

1 2 9 0



UNIVERSIDADE D
COIMBRA

Filipa da Luz Moreira

**ISABELA – IoT STUDENT ADVISOR AND BEST
LIFE ANALYZER**

**Dissertação no âmbito do Mestrado em Engenharia Biomédica orientada
pelo Professor Doutor Jorge Miguel Sá Silva e pela Investigadora Doutora Ana
Telma Fernandes Pereira e apresentada ao Departamento de Física da
Universidade de Coimbra.**

Setembro de 2023

This work was developed in collaboration with:

INESC - *Instituto de Engenharia de Sistemas e Computadores de Coimbra*



INESC Coimbra
Instituto de Engenharia de Sistemas e Computadores de Coimbra

Faculty of Sciences and Technology - University of Coimbra



IPM - Institute of Psychological Medicine of the Faculty of Medicine - University of Coimbra



Esta cópia da tese é fornecida na condição de que quem a consulta reconhece que os direitos de autor são da pertença do autor da tese e que nenhuma citação ou informação obtida a partir dela pode ser publicada sem a referência apropriada.

This thesis copy has been provided on the condition that anyone who consults it understands and recognizes that its copyright belongs to its author and that no reference from the thesis or information derived from it may be published without proper acknowledgement.

Agradecimentos

Gostaria de começar por agradecer aos meus orientadores, Professor Doutor Jorga Sá Silva e Investigador Doutora Ana Telma Pereira, por todo o tempo que investiram no acompanhamento do meu trabalho, pela disponibilidade, ajuda, e todo o conhecimento que me transmitiram ao longo deste último ano de mestrado, que foi uma grande mais valia e permitiu não só a entrega desta tese, mas também um grande crescimento das minhas capacidades de interpretação, resolução de problemas e o meu espírito crítico.

Gostaria também de agradecer a todas as pessoas que me acompanharam desde o início do meu percurso académico. Mari, Costini, Martini, obrigada por todos os momentos e recordações, que vou levar sempre comigo. Piri e Bule, obrigada por terem estado sempre lá, em todos os momentos bons e não tão bons. Agradecer também às pessoas que se foram juntando pelo caminho, nomeadamente à minha afilhada Mariana, que sempre me motivou e com quem partilhei momentos incríveis, e ao meu afilhado Miguéis que, apesar de chegar sempre atrasado, nunca falhava um jantar de família. Um obrigado também à restante família de praxe, por terem escolhido fazer este percurso connosco.

Por último, mas de todo não menos importante, tenho de agradecer à minha família por todo o apoio, por terem estado lá mesmo quando estava demasiado stressada para dar atenção, por me terem sempre feito acreditar que eu podia fazer o que quisesse e ser quem eu quisesse. Não tiveram apenas um papel importante neste percurso, mas em tudo o que levou até ele. Mãe, Pai, Rodrigo e Joana, não estaria aqui sem vocês.

A todos, muito obrigada!

Resumo

A motivação dos estudantes é um fator importante para o seu sucesso acadêmico. Problemas de saúde mental, derivados de fatores como stresse excessivo, podem afetar a motivação e, conseqüentemente, os resultados obtidos pelos estudantes.

Atualmente, a grande maioria dos estudantes universitários tem acesso a *smartphones*, que por sua vez reúnem todo um conjunto de sensores e possíveis funcionalidades. No entanto, a grande maioria das aplicações existentes não consideram o ser humano como um fator central, mas apenas como um fator externo que interage com as suas funcionalidades.

Usando a evolução de conceitos como a Internet das Coisas (IoT) e *Cyber-Physical Systems (CPSs)*, é possível começar a desenvolver sistemas no sentido de considerarem o ser humano como uma peça essencial e, desta forma, criar sistemas *Human-in-the-loop (HITL)*, que permitam aos utilizadores ter interações mais personalizadas.

O principal objetivo desta tese é adicionar funcionalidade ao sistema *IoT Student Advisor and BEst Life Analyser (ISABELA)*, que já conta com vários questionários para avaliar dimensões psicológicas, e com a recolha de dados através de vários sensores dos smartphones. Nomeadamente, começar a fechar o ciclo de controlo, devolvendo informação ao utilizador com base nos dados recolhidos, aproximando esta aplicação daquilo que é considerado um sistema *Human-in-the-loop (HITL)*.

Para atingir esse objetivo, para além de adicionar novos questionários, foi pensado e desenvolvida uma prova de conceito de um algoritmo de *Reinforcement Learning (RL)* para aprender com os dados recolhidos e conseguir devolver alguma informação ao utilizador que lhe seja útil, e o ajude no dia a dia como estudante universitário.

Palavras chave: Ferramentas de aprendizagem, Inteligência Artificial, Motivação, Internet das Coisas

Abstract

Student's motivation is an important factor to their academic success. Mental health problems, derived from factors such as excessive stress, can affect motivation and, consequently, the results obtained by students.

Nowadays, the vast majority of college students have access to smartphones, which in turn bring together a set of sensors and possible functionalities. However, the vast majority of existing applications do not consider the human a central factor, but only an external factor that interacts with its functionalities.

Using the evolution of concepts like the Internet of Things (IoT) and Cyber-Physical Systems (CPSs), it is possible to start developing systems in the sense of considering the human as an essential piece and, this way, create systems Human-in-the-loop (HITL), that allow users to have more personalized interactions.

The main objective of this thesis is to add functionalities to the IoT Student Advisor and BEst Life Analyser (ISABELA) system, that already has multiple questionnaires to evaluate psychological dimensions, and collects data through various smartphone sensors. Namely, start to close the control loop, giving back information to the user based on the collected data, bringing this application closer to what is considered a HITL system.

To achieve this goal, besides adding new questionnaires, a Reinforcement Learning (RL) algorithm was tough out and a Proof of Concept (PoC) was developed to learn from the data collected and to be able to return some information to the users that is useful to them, and that helps them daily as college students.

Keywords: Learning tools, Artificial Intelligence, Motivation, Internet of Things

Contents

List of Tables	xiii
List of Figures	xv
List of Code Snippets	xvii
List of Acronyms	xix
1 Introduction	1
1.1 Context and Motivation	1
1.2 Objectives	1
1.3 General outline of the thesis	2
2 Concepts and technologies used	3
2.1 Xamarin/.NET MAUI	3
2.2 FIWARE	4
2.3 Dash	5
2.4 DialogFlow	5
2.5 Human-in-the-loop systems	7
2.6 Reinforcement Learning	8
2.7 Python	9
2.8 Privacy, Ethics and Security	10
3 Existing Solution	11
3.1 ISABELA	11
3.2 Previously implemented questionnaires	11
3.2.1 HEXACO Personality Inventory	12
3.2.2 PoMS-12	14
3.2.3 Other questionnaires	15
3.2.3.1 Transportation form	15

3.2.3.2	Surrounding form	16
3.2.3.3	Sleep form	17
3.2.3.4	App finality form	17
4	Implementation	19
4.1	New questionnaires implemented	19
4.1.1	Questionnaires description	19
4.1.1.1	Maslach Burnout Inventory - Student Survey	19
4.1.1.2	Big Three Perfectionism Scale - Short Form	20
4.1.2	Code implementation	20
4.1.2.1	Model-View-ViewModel and Resource Dictionaries	20
4.1.2.2	ISABELA's implementation	22
4.2	Chatbot	25
4.2.1	First approach - general recommendations through DialogFlow	26
4.2.2	Strategy improvement - reinforcement learning model	28
4.2.2.1	Questionnaires and mental state	29
4.2.2.2	Questionnaire's threshold values	32
4.2.2.3	Algorithm selection	33
4.2.2.4	Q-learning algorithm	35
4.2.2.5	First implementation using Python	36
4.2.2.6	Integrate on ISABELA using Flask server	42
5	Functional and non-functional tests	47
5.1	Functional tests	47
5.2	Non-functional tests	53
6	Conclusions and future work	57
6.1	Conclusions	57
6.2	Future work	57
	Appendices	59
A	HEXACO-60 Personality Inventory	61
B	HEXACO-60 Personality Inventory - Portuguese version	65
C	PoMS-12 - Portuguese version	69
D	Maslach Burnout Inventory	71
E	Maslach Burnout Inventory - Portuguese version	73
F	Big Three Perfectionism Scale for Self-critical perfectionism	75
G	Big Three Perfectionism Scale for Self-critical perfectionism - Portuguese version	77

H	List of recommendations	79
I	QLearning Python code	83
J	Resulting article	87

List of Tables

3.1	HEXACO’s domains, facets and respective description[28]	13
3.2	Mood states per dimension used in ISABELA’s PoMS-12 implementation [31], [32]	15
4.1	Means and Standard Deviations (SDs) of selected HEXACO personality traits [48], perfectionism [43], burnout [68] and Profile of Mood States (PoMS) [31] questionnaires	33
4.2	Binary values set for each dimension	38
5.1	<i>epsilon_greedy_policy</i> test results	48
5.2	<i>calculate_reward</i> test results	49
5.3	Helper functions test results	51
5.4	<i>train</i> test results	51
5.5	<i>getRecommendation</i> test results	52
5.6	<i>getRecommendation</i> test results	53
5.7	Performance testing on <i>Flask</i> endpoints	54
A.1	HEXACO-60 questionnaire [25]	61
B.1	HEXACO-60 questionnaire - Portuguese version [90]	65
C.1	Mood states per dimension used in ISABELA’s PoMS-12 implementation - Portuguese version [31]	69
D.1	Maslach Burnout Inventory (MBI) [36]	71
E.1	MBI - Portuguese version [35]	73
F.1	Big Three Perfectionism Scale (BTPS) [39]	75
G.1	BTPS - Portuguese version [43]	77
H.1	Recommendations available in the Reinforcement Learning (RL) model	79

List of Figures

2.1	Xamarin’s architecture [7]	3
2.2	FIWARE platform architecture [10]	4
2.3	Agent with weather intent [14]	6
2.4	Flow of conversation using context [14]	6
2.5	Human-in-the-loop system diagram [16]	7
2.6	Interactions in a Markov Decision Process (MDP) [17]	9
3.1	Some screens for ISABELA’s HEXACO personality questionnaire . . .	12
3.2	ISABELA’s screen for PoMS questionnaire	14
3.3	ISABELA’s screen for Transportation questionnaire	16
3.4	ISABELA’s screen for Surrounding questionnaire	16
3.5	ISABELA’s screen for Sleep questionnaire	17
3.6	ISABELA’s screen for App Finality questionnaire	17
4.1	Model-View-ViewModel (MVVM) pattern [44]	21
4.2	Screens of the new implemented forms	24
4.3	Questionnaire’s architecture	25
4.4	Intent for high values of exhaustion in DialogFlow	27
4.5	28
4.6	Reinforcement Learning method diagram	29
4.7	Exponential decay of ϵ values through the algorithm steps	37
4.8	Tables of database for RL algorithm	39
4.9	Screens of the Graphical User Interface (GUI) developed in <i>Tkinter</i> . .	41
4.10	Recommendation displayed in ISABELA’s chatbot	44
4.11	Warning message when PoMS is not answered frequently	44
4.12	RL process in ISABELA	46
5.1	Number of users over time	55
5.2	Requests per second over time	55

List of Code Snippets

4.1	View model function example - Perfectionism questionnaire	22
4.2	View code example - Perfectionism questionnaire	23
4.3	JSON schema for agent initialisation	43
I.1	Q-learning code in Python	83

List of Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

BATINA Best Assisting Tutor and INteractive Advisor.

BTPS Big Three Perfectionism Scale.

CPS Cyber-Physical System.

CSS Cascading Style Sheets.

DL Deep Learning.

DP Dynamic Programming.

EU European Union.

FCTUC *Faculdade de Ciências e Tecnologias da Universidade de Coimbra.*

FL Federated Learning.

GDPR General Data Protection Regulation.

GUI Graphical User Interface.

HITL Human-in-the-loop.

HTTP Hypertext Transfer Protocol.

INESC Instituto de Engenharia de Sistemas e Computadores de Coimbra.

IoT Internet of Things.

IPM Institute of Psychological Medicine.

ISABELA IoT Student Advisor and BEst Life Analyser.

MBI Maslach Burnout Inventory.

MBI-SS Maslach Burnout Inventory - Student Survey.

MC Monte Carlo.

MDP Markov Decision Process.

MVVM Model-View-ViewModel.

NLU Natural Language Understanding.

NN Neural Networks.

PoC Proof of Concept.

PoMS Profile of Mood States.

RL Reinforcement Learning.

SARSA State Action Reward State Action.

SD Standard Deviation.

TD Temporal-Difference.

UI User Interface.

XAML eXtensible Application Markup Language.

Introduction

1.1 Context and Motivation

Mental health is a topic that has been gaining more attention throughout the years, as its importance and impact on overall health is increasingly recognized. [1]

College students have high tendencies to mental health problems, mostly due to the high stress they are subjected to and the difficulty in keeping a good sleep routine, being that these factors are interconnected. Many of them present anxiety symptoms and, in more severe cases, depression symptoms. All these factors have a negative impact on academic performance. [2]

Motivated students are more likely to be academically successful, and motivation is highly correlated to mental health. [3]

In Portugal, the statistics show that there are a lot of students that drop out of college within the first year, with the 2023 percentage being around 10%. [4] A lot of factors can influence students to drop out, and some of the most relevant ones include anxiety, depression and lack of motivation. [5]

With the aim to help students with motivation, and by doing so, helping with their academic performance, the ISABELA system was developed. By leveraging the concepts of Internet of Things (IoT) and Cyber-Physical System (CPS), and the fact that smartphones are widely spread, and most students have access to one, the idea is to develop a system that can help students by analysing data collected by sensors and questionnaires that measure different psychological dimensions.

1.2 Objectives

The main objective of the work developed in scope of this thesis was to improve the functionality of ISABELA by adding new questionnaires, relevant to the application's purpose, and to start working on closing the loop of the system by utilizing the results obtained from different inputs into a feedback that can give students support in their day to day life. To achieve this, it was necessary to merge

psychology concepts with technology and development concepts.

This work connects two very distinct areas, psychological medicine and informatics, to achieve a common goal of helping college students achieve better results and remain motivated. Particularly, it starts to close the loop of what intends to be a HITL system, by starting to go deeper into the investigation of how the data that was already being collected can be leveraged to give relevant feedback, so the system starts to have more utility for its users and comes closer to its intended purpose.

This thesis was the result of a collaboration between *Instituto de Engenharia de Sistemas e Computadores de Coimbra (INESC)* and Institute of Psychological Medicine (IPM).

1.3 General outline of the thesis

This thesis is divided into six main chapters: Introduction, Concepts and technologies used, Existing Solution, Implementation, Functional and non-functional tests, and Conclusions and future work.

In the first chapter, the general context and motivation was described, as well as the objective of this work.

In the second chapter, the technologies and concepts used are better described.

In the third chapter, the existing solution is presented, with a general overview of ISABELA and what functionalities it had before the beginning of this work.

In the fourth chapter, the implementation done in scope of this thesis was described, focusing both on the psychology side, with more detailed explanations of the added questionnaires, what they evaluate and why they were added, and the programming side, describing methods and models used.

In the fifth chapter, the results from some functional and non functional tests developed for the RL code are presented.

Lastly, chapter six contains some conclusions from the work, as well as a discussion of topics that might be improved in future work.

Concepts and technologies used

2.1 Xamarin/.NET MAUI

Xamarin is an open-source platform that allows developers to share their code across different platforms (Android, iOS, Windows) resulting in native performance while only writing business logic code in one language (C#). [6]

Regarding the User Interface, Xamarin.Forms can be used to allow for shareable code either in eXtensible Application Markup Language (XAML) or C#. It supports databinding and simplifies the process of coding by leaving out the need to learn each platform's native language.[7]

Figure 2.1 is a general diagram of the architecture of an application developed in Xamarin. On the top layer we see the User Interface (UI) level, that will be native to each platform. Then, the general code is programmed in C# and XAML, and is shared across platforms. At runtime, the code will be processed and the shared elements will be converted into native controls. [7]

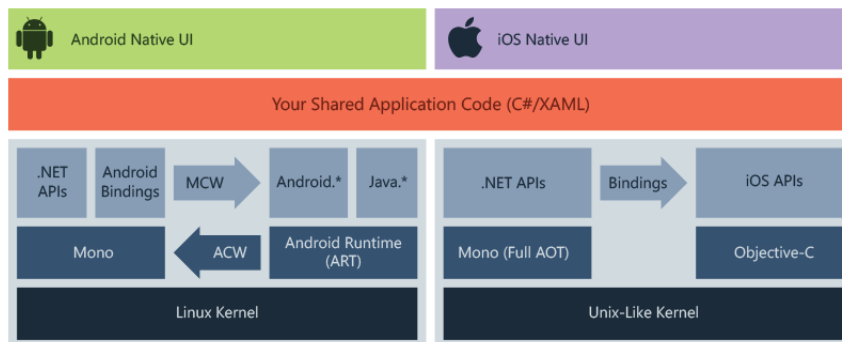


Figure 2.1: Xamarin's architecture [7]

As of 2022, .NET MAUI appeared as an evolution of Xamarin.Forms. It has the same purpose of cross-platform coding, but presents some new features and enhancements. [8] It would be interesting to explore this new framework and eventually migrate the code to it, since Xamarin will no longer have support from Microsoft as of May 2024. [9]

Given all the functionalities that Xamarin supports, it was selected as the platform to develop the application code for the ISABELA system.

2.2 FIWARE

FIWARE is an open source smart solution platform that includes a set of Application Programming Interfaces (APIs) available to the public. Its components are called Generic Enablers and can be integrated with each other and with third-party components, allowing for an easy and fast build of platforms.

The only Generic Enabler required to create a FIWARE platform is a Context Broker that will manage the context information, allowing for the creation of entities, and to update information, as well as receive notifications in real-time. There are a series of available Generic Enablers to build onto the Context Broker. In figure 2.2 we can see the Context Broker represented in the middle, and then different categories of Generic Enablers around it.

- Interface with Internet of Things (IoT), Robots and Third-Party Systems - get context updates and trigger actuations as needed.
- Data/API Management, Publication and Monetization - manage context data, facilitating usage control and providing the chance to publish and monetize it. The access to this data can be managed by using Security components.
- Context Processing, Analysis and Monitoring - process, analyze, and visualize context information, enabling smart applications to behave intelligently and give feedback when certain events occur.
- Deployment tools - components can generally be deployed with basic containerization, being that most of them are available as Docker Images

The available Generic Enablers can be used in different combinations with third-party components so that different platforms are designed and built. [10]

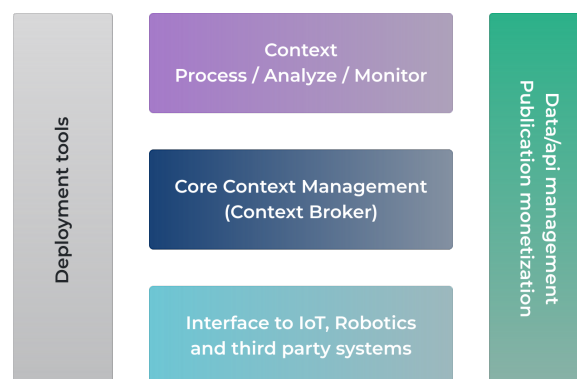


Figure 2.2: FIWARE platform architecture [10]

The ISABELA system uses various FIWARE modules in the backend to save information, allow for authentication, among other functionalities.

2.3 Dash

Dash is a platform based on Plotly.js and React.js that can be used to build and deploy web apps that allow for interactive data visualization. Said apps are cross-platform and mobile ready. It is possible to create different types of graphs, dropdowns, buttons, and other components that allow interaction with the user and visualization based on inputs.[11]

It is a simple tool that supports page customization through Cascading Style Sheets (CSS), multi-page apps and JavaScript. [12]

Dash is used in the context of ISABELA to present dashboards with the collected data.

2.4 DialogFlow

DialogFlow is a platform by Google that allows for the creation of chatbots by using Natural Language Understanding (NLU) mechanisms. It is composed of agents that are trained to understand human language, whether written or spoken. [13]

In order to build a chatbot in DialogFlow, it is necessary to create an agent and define intents. When the user gives an input, DialogFlow will match the information given with the closest intent and provide a response. In this process, it can also trigger actions and retrieve values as parameters that can be stored or used in responses. Parameters have types, called entity types, that can either be predefined (date, time, and so on) or custom.

In figure 2.3 there is an example of an agent which has an intent to answer questions about the weather. Words like 'weather' or equivalents, highlighted in blue, will be recognized and match the forecast intent. Then, the agent can give back an answer. In the process, it extracts parameters, time and location, that help identify what exact weather the user is requesting.

2. Concepts and technologies used

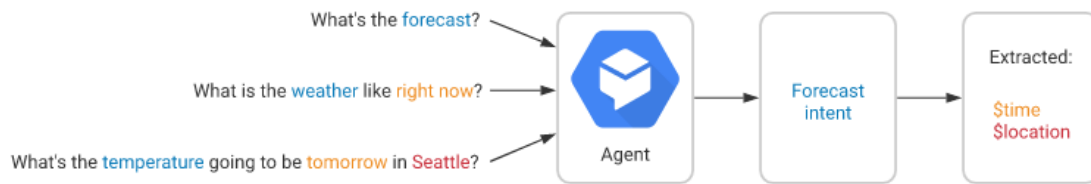


Figure 2.3: Agent with weather intent [14]

To generate more complex conversations, contexts can also be defined. Contexts are activated when an intent that contains them is matched and will influence the subsequent flow of the conversation by matching intents with the input context matching the active one. This also allows for the creation of follow-up intents that will provide answers based on the previous user input. [14]

In figure 2.4 it is possible to see how a flow of conversation works when there is a context. When the user asks about the checking account, it will match an intent that activates a context. From that moment on, during the conversation, the agent will most likely match intents that have the defined context, based also on the content of the messages received.

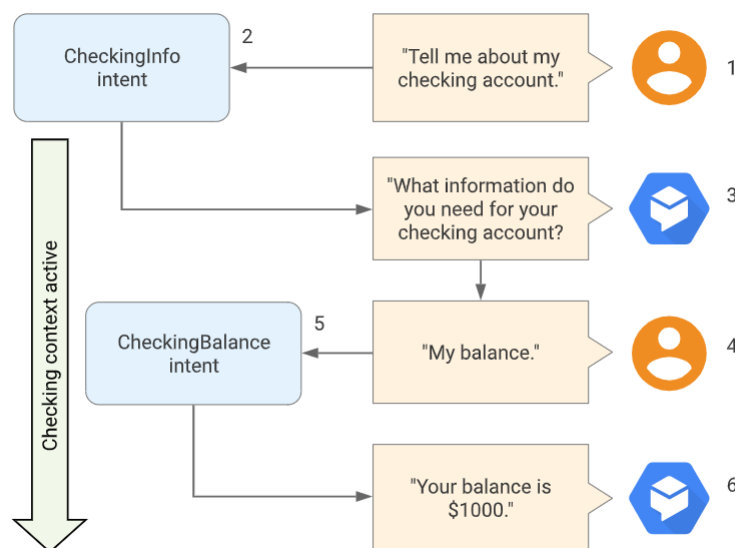


Figure 2.4: Flow of conversation using context [14]

There are a lot of options to build and train chatbots with DialogFlow.

It is also important to mention that DialogFlow has different options for integration with other systems. However, Xamarin is not supported.[15] In order to allow for communication between Xamarin and DialogFlow there is a server developed in node.js.

DialogFlow is used in the ISABELA's chatbot, given that it provides all the mentioned resources and it is easy to learn and utilize.

2.5 Human-in-the-loop systems

HITL systems consider the human as a central factor, by analysing data collected by sensors and mobile devices, and tailoring the actions taken by the system to the users' needs. This kind of system can have many applications, namely, for the purpose of this work, in psychology by understanding the users' state and taking action according to it.

In figure 2.5 there is a diagram of how a HITL system generally works. It is centered on the human, from who it collects data and then analyses a state and takes an action directed back to the human.

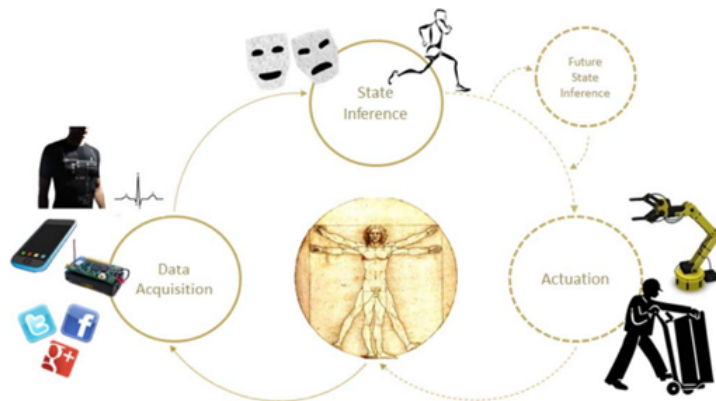


Figure 2.5: Human-in-the-loop system diagram [16]

There are various limitations to the applications of this type of systems, given that humans are complex and it is hard to integrate all the information in order for a system to make accurate decisions and be useful and reliable. However, as technology evolves, this becomes a closer reality, and research in these areas has shown promising results. [16]

The ISABELA system aims to be a HITL system, able to provide useful feedback to the users by taking its interactions and information into consideration. In chapter 3 there is an explanation of what has been done so far. The main purpose of this work is to start developing more to close the loop, by starting to take actions based on the already collected information, and on some more added questionnaires.

2.6 Reinforcement Learning

RL is a machine learning method that learns by interacting with its environment and trying to achieve the maximum reward possible. This type of algorithm simulates the general way in which humans and animals learn by evaluating their environment and trying to assess the best actions to take to achieve certain goals.

In a RL algorithm, a learning agent assesses its environment and takes actions that influence that environment's state. Its objective is to reach a certain goal state, and for each action there is a reward that will indicate if its actions are getting it closer to that goal.

Reinforcement Learning differs from both supervised and unsupervised learning. In supervised learning, there is a collection of examples provided to the system so that it knows which actions to take in each situation. This kind of machine learning is not practical when we are talking about complex environments where it is difficult to label every possible scenario. Unsupervised learning algorithms deal with a set of unlabeled data and try to find patterns and structure to group them into categories. Even though RL also deals with data that is not classified, its goal is not to find structure but instead to lead the environment to a certain goal state and maximize the reward by taking actions, as previously mentioned.

One important aspect to take into account is the balance between exploitation, which leads the agent to take advantage of previous actions with a good reward and repeat them, and exploration, which means the agent should also try actions that it hasn't tried before in order to learn what gives the best reward. This is one of the big challenges when developing a RL model.

Some important elements to consider during the development of a RL method are:

- Action - defines what the agent can do in each situation, what options he has
- State - the current circumstances of the environment
- Reward - defines what is a good and what is a bad action for the agent, as its purpose is to maximize it

With this basic information, an agent can be trained and define a policy that will map each state to a possible action. [17]

Reinforcement Learning is strongly based on MDPs. While MDPs are a mathematical idealization for which it is possible to make exact theoretical claims, RL adds the concepts of missing information and approximation to model real world complex problems.

MDPs take into account not only the immediate reward received when taking

an action, but also the potential future reward, formalizing the process of making sequential decisions. If a certain state only depends on the previous one and not on the ones prior to that, given that the previous state contains all relevant information to move forward, then the state has the Markov property. So, in a MDP, the environment's dynamics can be fully characterized by probabilities, given that the chance to achieve a certain state and reward on the next step depends only on the previous state and action.

The main principle is that problems of learning through interaction can be reduced to three main signals traded: the action, sent from the agent to the environment, and the reward and state, given to the agent by the environment. This does not apply to every case scenario, but it is a flexible and useful approach to take. It is possible to see the basic interactions in figure 2.6. [17]

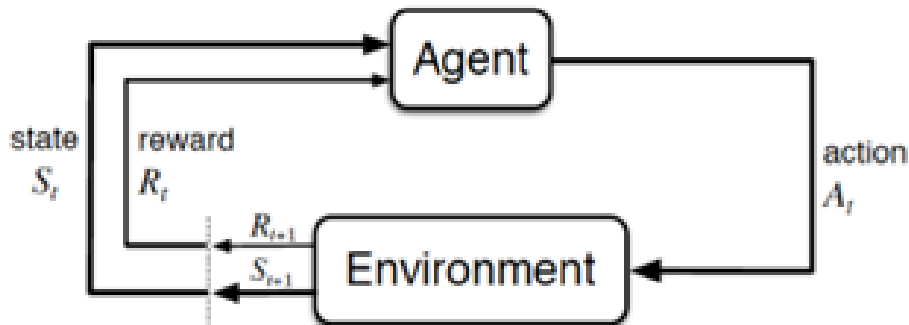


Figure 2.6: Interactions in a MDP [17]

With the goal of making ISABELA able to give feedback based on the user's interaction, a RL algorithm was developed within the scope of this work. For this first approach to make ISABELA personalized to each user, RL algorithms come as a good way to start this process, and to test possible solutions.

2.7 Python

Python is an open source programming language that includes a series of libraries with different functionalities, as well as various third-party extensions, making it usable for many different types of problems. It has a clear syntax and allows for different programming paradigms, making it very flexible and adaptable. It can run in different operating systems and can interface with other systems using protocols such as Hypertext Transfer Protocol (HTTP), and it also has different training frameworks available. [18]

Given all of these features, Python is a good language to implement machine learning algorithms, including reinforcement learning.

In the scope of this thesis, like it was mentioned before, a RL algorithm was implemented. The final goal was to have it integrated into the ISABELA system. However, since Python is a language widely used in this kind of problems, it was chosen as a first approach to code development.

2.8 Privacy, Ethics and Security

The ISABELA app collects a series of data for analysis and feedback purposes. As it aims to be a personal helper to students in their day to day life, it needs to gather various information to process and to associate it with a specific user.

This, while being really important for the application's functionality, raises concerns regarding privacy, ethics and security of the collected data. Particularly in the European Union (EU), the General Data Protection Regulation (GDPR) applies, and the data privacy of the users needs to be guaranteed. The data needs to be safely stored, the users must be informed of the purpose of the data collection, and the data needs to be used only for that purpose. [19]

With that in mind, there is a group within the project that is developing security measures, particularly Federated Learning (FL). With FL, multiple users can be connected with one or more central servers, but the data belonging to each user is stored locally. The Artificial Intelligence (AI) models can be trained locally, and then the parameters can be shared with the servers and contribute to the overall mechanism, while assuring privacy and having the users as owners of their own data. [20], [21]

Existing Solution

3.1 ISABELA

ISABELA is a system based on several technologies, that have been previously mentioned, developed by our research group at *Faculdade de Ciências e Tecnologias da Universidade de Coimbra* (FCTUC). Its main goal is to help university students with academic performance by analysing multiple factors, both physical and psychological, and use the data collected to provide recommendations to the users.

To accomplish this, ISABELA is being developed to be available for both Android and iOS devices and collects data in multiple ways:

- without user's interaction - using sensors available in smartphones such as accelerometer, light sensor, proximity sensor, gyroscope, among others
- with user's interaction - by answering the app's available questionnaires such as personality, emotional, sleep, among others, as well as interacting with a chatbot

ISABELA is available in both Portuguese and English, being that every questionnaire's Portuguese version has been validated and revealed good psychometric properties.

Besides ISABELA, there is other application developed by this research group, named Best Assisting Tutor and INteractive Advisor (BATINA). Its purpose is more towards academic goals, providing a link between students and teachers. Teachers can put materials and questionnaires in the app, for students to answer, and students can get some information about performance and interact with a chatbot.

3.2 Previously implemented questionnaires

Before discussing the new questionnaires and functionalities developed in scope of this thesis, the already existing questionnaires will be described to give some context of what already existed and its purpose.

3.2.1 HEXACO Personality Inventory

The term personality can be described as a series of traits that characterise individual patterns of thinking and feeling, resulting in certain responses or behaviors when facing specific situations. Those patterns tend to be relatively stable during an individual's life.

Several explanatory models of the relationship between personality and health have been validated, which propose ways and cognitive, emotional, behavioural, social and physiological processes through which traits influence different health outcomes. [22] Personality traits have also been found to be able to predict overall academic performance and achievement, seemingly even more than intelligence. [23], [24]

Given these relations, personality is an important factor to consider in the development of ISABELA as an application that intends to monitor and improve academic performance. With this in mind, the HEXACO Personality Inventory - Revised was included as one of the questionnaires available in order to evaluate the six major personality dimensions. It is composed of sixty questions to be answered with five levels according to the how much the student agrees with them given that 1 means "strongly disagree" and 5 means "strongly agree". [25]

To implement this in the code there are six pages with 10 questions each that can be answered with a slider. The student can move forwards and backwards through the pages. This questionnaire is only answered once since personality traits are considered to be stable during a considerable amount of time. [26], [27]

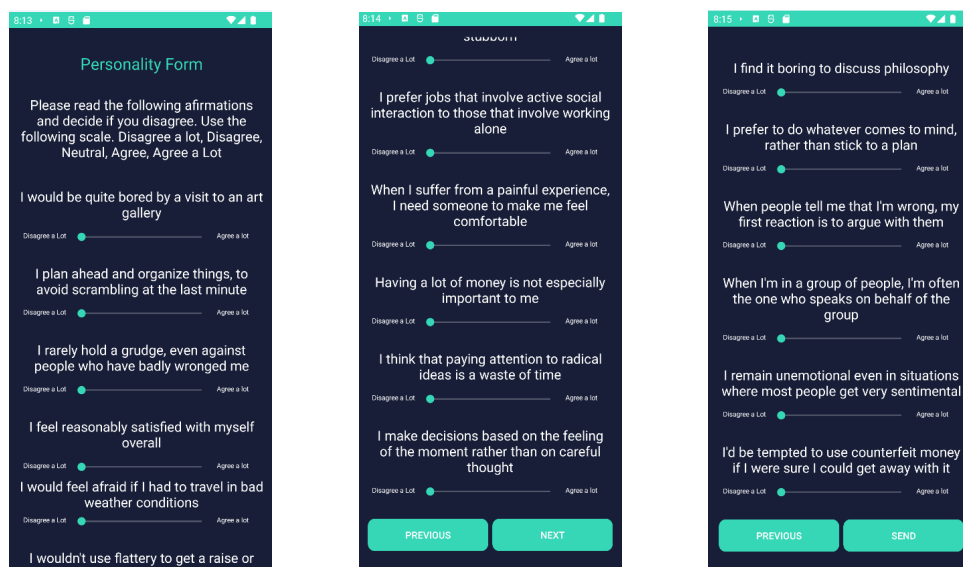


Figure 3.1: Some screens for ISABELA's HEXACO personality questionnaire

After submitting their answers, the student can see his/her level on each of the personality dimensions evaluated, with them being:

Table 3.1: HEXACO's domains, facets and respective description[28]

Domains	Facets	Descriptions
Honesty/Humility	Sincerity Fairness Greed avoidance Modesty	How much an individual avoids manipulating other people, feels no entitlement, or is not tempted to break the rules.
Emotionality	Fearfulness Anxiety Dependence Sentimentality	How much an individual fears physical harm, experiences anxiety, needs emotional support and has feelings of attachment to others.
Extraversion	Social self-esteem Social boldness Sociability Liveliness	How much confidence about themselves and leading others an individual feels, as well as enjoying social interactions and feeling enthusiastic and energetic.
Agreeableness (vs Anger)	Forgiveness Gentleness Flexibility Patience	How easily an individual controls their temper, forgives wrongdoing, is tolerant, cooperative, and able to compromise.
Conscientiousness	Organization Diligence Perfectionism Prudence	How organized and disciplined an individual is, striving for perfection and accuracy and making deliberated decisions.
Openness to Experience	Aesthetic appreciation Inquisitiveness Creativity Unconventionality	How imaginative an individual is, taking interest in unusual things as well as appreciating beauty and having intellectual curiosity.

To see the full version of the questionnaires and how the scores are calculated, check appendices A (English version) and B (Portuguese version).

3.2.2 PoMS-12

The Profile of Mood States (PoMS) is a questionnaire that, in its original version [29], includes 65 adjectives describing mood states and, for each one, the individual can choose how he feels regarding that state, from a scale from 0 (Not at all) to 4 (Extremely). These states relate to six dimensions: tension, depression, anger, vigor, fatigue, and confusion [29].

There is evidence that positive and negative mental states can influence an individual's academic performance. While negative emotions can reduce one's ability to concentrate and reduce working memory, positive emotions have the opposite effect, increasing focus, improving working memory and having an overall positive impact in learning. [30]

Given the impact that positive and negative emotions have on students' well-being, motivation and academic performance, it is relevant to consider affective states in ISABELA. As such, there is a short version of this questionnaire implemented in the application. It contains 12 adjectives that describe feelings and evaluate three mood states: depression, anxiety and positive affect. The sum of the first two is a negative affect proxy. While still being reliable and able to give valid results, the time spent filling this shorter version of PoMS is much shorter than it would be if we were to answer all original 65 items, making it easier for users to answer the questionnaire at any time during their day. [31] Table 3.2 shows the mood states present in the questionnaire, as well as the mood dimensions they relate to.

This questionnaire is presented in a screen where, for each mood state, there is a slider with values that go from "Not at all" to "Extremely" and should be answered every day.

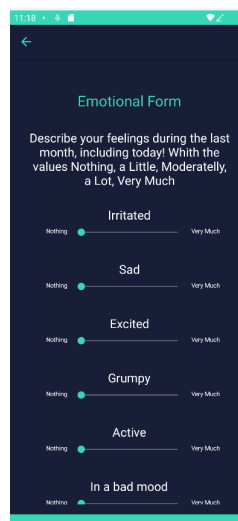


Figure 3.2: ISABELA's screen for PoMS questionnaire

Table 3.2: Mood states per dimension used in ISABELA’s PoMS-12 implementation [31], [32]

Mood dimensions	Mood states
Depression	Sad Pessimistic Desperate Useless
Anxiety	Irritated Grumpy In a bad mood Anxious
Positive Affect	Excited Active Cheerful Self-confident

The Portuguese version of this questionnaire can be found in appendix C, as well as how the calculation is made.

3.2.3 Other questionnaires

Besides HEXACO and PoMS, some other forms were implemented in order to collect a series of data that can be related with a student’s mental health. Here they are briefly described.

It is important to note that the transportation and surrounding forms were added in the scope of a project developed during the COVID pandemic, with the purpose of identifying if people were traveling and having contact with each other.

3.2.3.1 Transportation form

The transportation questionnaire is composed by two questions, one about the vehicle in which the user was and one about how many people were there with them.

Both questions are answered by selecting an option from a multiple choice. The available options for vehicles are personal vehicle, friend/colleague’s vehicle, taxi/TVDE, bus, boat, subway/train/tram. The choices for how many people were

3. Existing Solution

with the person depend on the vehicle chosen. For instance, if a personal vehicle is chosen, the options available are 0, 1, 2 or 2+, as it is a smaller mean of transportation. However, if a larger vehicle, like a bus, is selected, the options will be less than 10, 10 to 20, 20 to 30 or more than 30.

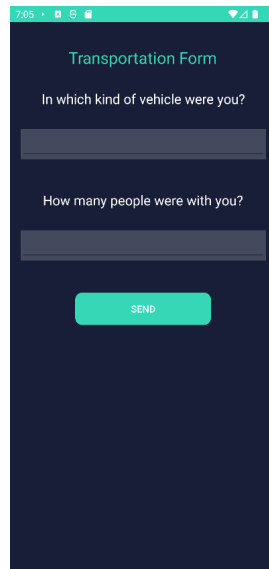
A screenshot of a mobile application interface titled "Transportation Form". The screen has a dark blue background. At the top, the title "Transportation Form" is displayed in a light blue font. Below the title, there are two questions: "In which kind of vehicle were you?" and "How many people were with you?". Each question is followed by a dark grey rectangular input field. At the bottom of the form, there is a light blue button with the word "SEND" in white capital letters. The status bar at the top shows the time as 7:05 and various system icons.

Figure 3.3: ISABELA's screen for Transportation questionnaire

3.2.3.2 Surrounding form

For the surrounding questionnaire, there is only one question: *How many people are less than 2 meters away from you?*. To answer that, the user types in a number.

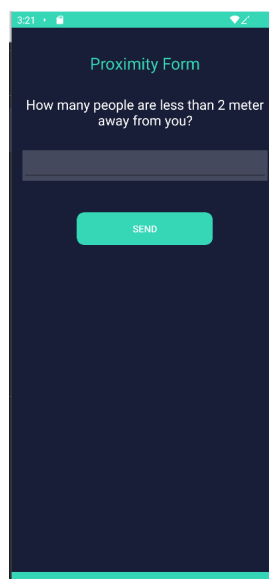
A screenshot of a mobile application interface titled "Proximity Form". The screen has a dark blue background. At the top, the title "Proximity Form" is displayed in a light blue font. Below the title, there is a single question: "How many people are less than 2 meter away from you?". This question is followed by a dark grey rectangular input field. At the bottom of the form, there is a light blue button with the word "SEND" in white capital letters. The status bar at the top shows the time as 3:21 and various system icons.

Figure 3.4: ISABELA's screen for Surrounding questionnaire

3.2.3.3 Sleep form

The sleep questionnaire includes questions about hours of sleep, how the user felt socially the day before, how they slept that night, and how many hours they spent exercising and studying the previous day. These questions are answered by using sliders and indicating a time.

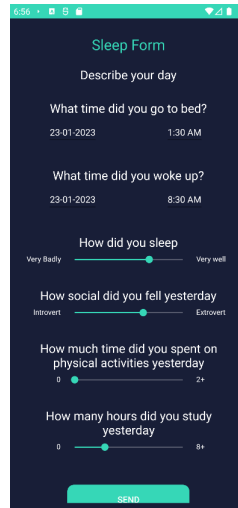
The screenshot shows a mobile application interface titled "Sleep Form". The background is dark blue. At the top, there's a status bar with the time 6:56 and various icons. Below the title, it says "Describe your day". The form contains five questions, each with a corresponding input field or slider: 1. "What time did you go to bed?" with a date and time picker showing "23.01.2023" and "1:30 AM". 2. "What time did you wake up?" with a date and time picker showing "23.01.2023" and "8:30 AM". 3. "How did you sleep" with a horizontal slider between "Very Badly" and "Very well", with a green dot indicating a position. 4. "How social did you feel yesterday" with a horizontal slider between "Introvert" and "Extrovert", with a green dot indicating a position. 5. "How much time did you spent on physical activities yesterday" with a horizontal slider between "0" and "2+", with a green dot indicating a position. The final question is "How many hours did you study yesterday" with a horizontal slider between "0" and "8+", with a green dot indicating a position. At the bottom, there is a green "SEND" button.

Figure 3.5: ISABELA's screen for Sleep questionnaire

3.2.3.4 App finality form

In this questionnaire, the user is presented with the five most used applications and he can choose between four different types of possible reasons to utilize the app in question: communication, leisure, research or work.

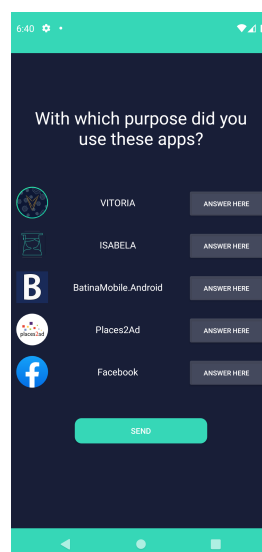
The screenshot shows a mobile application interface titled "With which purpose did you use these apps?". The background is dark blue. At the top, there's a status bar with the time 6:40 and various icons. Below the title, there are five rows, each representing an application: 1. "VITORIA" with a globe icon and an "ANSWER HERE" button. 2. "ISABELA" with an hourglass icon and an "ANSWER HERE" button. 3. "BatimaMobile.Android" with a large letter 'B' icon and an "ANSWER HERE" button. 4. "Places2Ad" with a location pin icon and an "ANSWER HERE" button. 5. "Facebook" with the Facebook 'f' icon and an "ANSWER HERE" button. At the bottom, there is a green "SEND" button.

Figure 3.6: ISABELA's screen for App Finality questionnaire

4

Implementation

4.1 New questionnaires implemented

One of the purposes of this work was to add some more functionalities to the application. The first new implementation included adding two new questionnaires to the ones already existing, described in the previous chapter.

4.1.1 Questionnaires description

To introduce the rational of the added questionnaires, here is presented a description from the psychology's perspective.

4.1.1.1 Maslach Burnout Inventory - Student Survey

Burnout is characterized by feeling emotionally exhausted, detached from the job, and unaccomplished, which arises from continuous stress and can have a significant negative impact on the individual and their work. The MBI was originally developed to assess burnout levels in human services workers, i.e., people whose job involves interacting with other people in some sense. [33]

However, burnout also affects workers in other fields and even students. The activities in which students engage are very similar to work, as they are subject to schedules and timelines to develop assignments that are submitted to a performance evaluation. Burnout has a negative impact on academic performance and achievement. [34]

Maslach Burnout Inventory - Student Survey (MBI-SS) is an adapted version of the MBI to be used by students. It is composed by 15 items divided into 3 scales: exhaustion, cynicism and professional efficacy. Each item is a statement and is scored on a 7-point rating from 0 (Never) to 6 (Always). While elevated scores in exhaustion and cynicism indicate higher degrees of burnout, the professional efficacy works with reversed values, which means that higher degrees of burnout will result in lower scores in this scale. [35]

In ISABELA the MBI-SS is implemented in both English [35] and Portuguese [36], and the full versions can be found in appendices D (English version) and E (Portuguese version), as well as how the results are calculated.

4.1.1.2 Big Three Perfectionism Scale - Short Form

Perfectionism is a personality trait that can be described by multiple facets/dimensions. Perfectionist individuals have unreasonably high standards and goals for themselves, and do not leave any space for mistakes, being constantly worried that they have not done things well enough. It includes feelings of self-doubt, being overly critical, placing value in meeting other's expectations and being precise. There is a relation between perfectionism and several psychological conditions, and even with procrastination. [37]

The BTPS is a recent model used to evaluate perfectionism in three major factors: rigid perfectionism, self-critical perfectionism and narcissistic perfectionism. It was originally composed by 45 items[38], but a shorter version with only 16 items was developed. [39]

Self-critical perfectionism in particular is related to negative emotions towards imperfect performing and believing that other people expect perfection. [39] It is the perfectionism dimension more strongly associated with psychological distress, burnout, difficulty in coping and more academic struggles, making student more prone to perceived failure and criticism. [40]–[42]

For ISABELA the items of the BTPS shorter version that characterize self-critical perfectionism were used. It was implemented in both English [39] and Portuguese [43]. The full version of the questionnaire can be found in appendices F (English version) and G (Portuguese version) with how the results are calculated.

4.1.2 Code implementation

Now that there is a clearer understanding of why these questionnaires are relevant and were implemented, the description of the code developed is presented below.

4.1.2.1 Model-View-ViewModel and Resource Dictionaries

When developing apps in Xamarin, it is recommended to organize the code in a MVVM pattern in order to clearly separate the business logic from the UI and facilitate maintenance, testing, and development. It can also make some code reusable, avoiding repetition and making collaborative work easier.

A MVVM pattern is composed of a model, a view, and a view model that perform different functions and interact with each other to compose a functional app.

In order to successfully implement a MVVM pattern, it is important to understand each component's function and how they relate.

Starting with the view, it should mostly be developed in XAML, even though it can have some code-behind to define complex behaviour as long as it does not define any business logic. It defines the layout of what the user sees when interacting with the application.

The view model is developed in C# and has commands and properties to which the view can be bound. So, the view model introduces functionality to the application, and the view can interact with it to display different things to the user. The view binds to the view model, however the view model is not aware of the view code.

On the other hand, the view model also exposes model data, occasionally performing some data conversion so that the view can easily consume it. This way, it acts like a middle layer between the view and the model, abstracting them from each other in such a way that they can be independently developed. The model is not aware of the view model's code; only the view model binds to and gets information from the model. An example can be, if there is a model with properties that represent answers to some questions, and in the view there are input fields to answer said questions, the view model will then bind these two and can also take care of other actions such as sending data to a server.

Lastly, the model classes are also defined in C# and usually consist of a data model and some logic. [44]

In figure 4.1 there is a diagram of the MVVM pattern, where it is possible to see what was mentioned so far. The view binds to the view model, the view model can update and interact with the model.

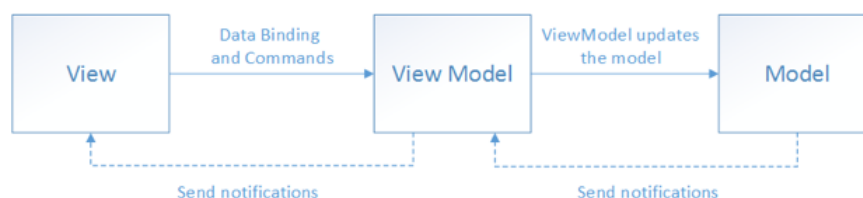


Figure 4.1: MVVM pattern [44]

By using this pattern, it becomes possible to, for instance, update the UI without interfering with the code. It is also possible to unit test the view model and model, without interfering with the view. Another advantage is that it is easier for

developers and designers to cooperate, since they can work separately on different components. [44]

It is also possible to integrate resource dictionaries into the code. In these dictionaries, it is possible to input different resources and attribute a key to each of them. Then, that key can be used to map to the resource value and use it in the XAML.[45] This can be used to translate resources presented in the application. If multiple dictionaries are configured in different languages, the preferences set by the user can define to which of them the view is bound, and in that way present different languages.

4.1.2.2 ISABELA's implementation

With the current code structure, the questionnaires follow a MVVM pattern and there are resource dictionaries for translation purposes. To add new questionnaires, it is possible to use this structure to maintain the business code separate from the visual implementation, as previously explained, and use the resource dictionaries to input the text to show in English and Portuguese and use this approach to make the application able to work in both languages.

In the model, the structure of the questionnaire answers to be saved is defined. Since the answers are a number in a certain range, the attributes are represented as numeric properties. The timestamp of the questionnaire's answer is also saved.

In the view model, it is where the logic is implemented, mostly to handle saving the data. There is a connection with the model, so its information can be used to deal with the data that its send in by the view. In code snippet 4.1, it is possible to see an example of a function that is built inside the view model of the perfectionism's questionnaire, that uses the model structure to save the data coming from the user's interaction with the view. This function is called when the 'Send' button is pressed.

```
1 public async Task<bool> SendData()
2 {
3     var db = new FormsService();
4
5     try
6     {
7         var form = new Models.Forms.Perfectionism
8         {
9             id = $"perfectionism_{RestFiwareUtils.GetUserId()}",
10            item1 = PerfectionismQ1,
11            item2 = PerfectionismQ2,
12            item3 = PerfectionismQ3,
13            item4 = PerfectionismQ4,
```

```

14         item5 = PerfectionismQ5,
15         item6 = PerfectionismQ6,
16         timestamp = NgisiUtils.DatetimeToString(DateTime.UtcNow)
17             };
18         await db.AddPerfectionismForm(form);
19         return true;
20     }
21     catch (Exception ex)
22     {
23         Console.WriteLine("Something went wrong ex");
24     }
25
26     return false;
27 }

```

Code snippet 4.1: View model function example - Perfectionism questionnaire

In the view, there is a 'StackLayout' structure, as per the previously implemented questionnaire pages, and the color scheme used throughout the application is maintained. The view can then bind to the resource dictionaries to get the text and to the view model to be able to access the logic. By using the resource dictionaries like this, the app is available in both Portuguese and English, as previously mentioned.

In code snippet 4.2 it is possible to see a small piece of a view's code for the perfectionism questionnaire. In there, a question is defined, being the text given by 'PerfectionismFp1', and the values presented at the each end of the slider given by 'PerfectionismFa1' and 'PerfectionismFa5', being that all these fields are defined in the resource dictionaries. Then, the slider itself allows values between 1 and 5, and they will bind to 'PerfectionismQ1', a field defined in the view model.

```

1
2 <StackLayout Margin="0,25,0,8">
3     <Label Text="{DynamicResource PerfectionismFp1}"
4         HorizontalOptions="CenterAndExpand"
5         HorizontalTextAlignment="Center"
6         TextColor="{DynamicResource White_ISABELA}"
7         FontSize="18"/>
8     <Grid>
9         <Grid.ColumnDefinitions>
10            <ColumnDefinition Width="*" />
11            <ColumnDefinition Width="3*" />
12            <ColumnDefinition Width="*" />
13        </Grid.ColumnDefinitions>
14        <Label Grid.Column="0" Text="{DynamicResource

```

4. Implementation

```
PerfectionismFa1}"
15         TextColor="{DynamicResource White_ISABELA}" FontSize
           ="10" HorizontalOptions="End"/>
16         <Label Grid.Column="2" Text="{DynamicResource
           PerfectionismFa5}"
17         TextColor="{DynamicResource White_ISABELA}" FontSize
           ="10" HorizontalOptions="Start"/>
18         <Slider Grid.Column="1" Maximum="5" Minimum="1"
19         Value="{Binding PerfectionismQ1, Mode=TwoWay}"
20         ValueChanged="OnSliderValueChanged"
21         ThumbColor="{DynamicResource Green_ISABELA}"
22         MinimumTrackColor="{DynamicResource Green_ISABELA}"
           />
23     </Grid>
24 </StackLayout>
```

Code snippet 4.2: View code example - Perfectionism questionnaire

In figure 4.2 there are the screens of the new questionnaires implemented.

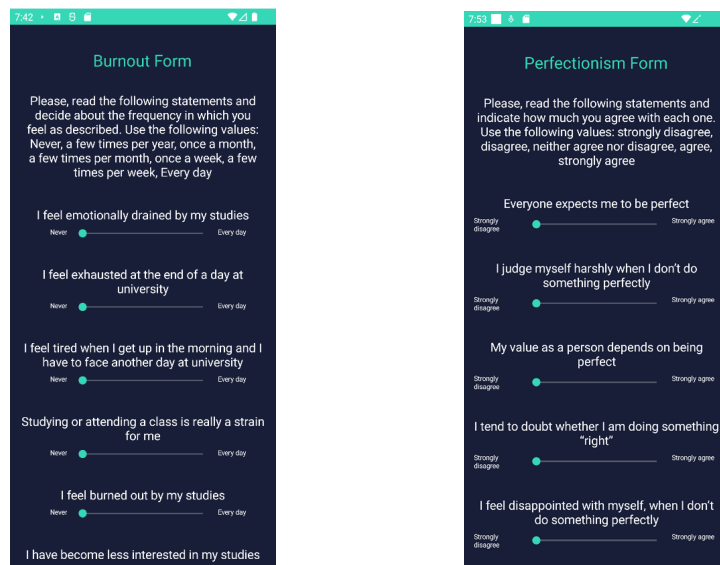


Figure 4.2: Screens of the new implemented forms

Once this all has been done, the new questionnaires work properly, from user interface to backend code.

In figure 4.3 the general process of saving the questionnaire's data after the 'Send' button is pressed is described.

Every time a user finishes answering a questionnaire and presses the 'Send' button, a function is called to send the data to the database. The answers are saved in the local SQLite database, based on the date structure defined in the model.

Then, if there is an Internet connection, a POST HTTP call will be made to the server to try and save the data in its database, which is in MongoDB. If it succeeds, then the answers are deleted from the local database. If there is no Internet connection, then the answers are only saved locally, but there is a library, the RestFiwareAPI, running in the background that will periodically try to send the request to the server. Once it succeeds, the data is deleted locally. This way, it is assured that the answers from the users are always stored, even without an Internet connection.

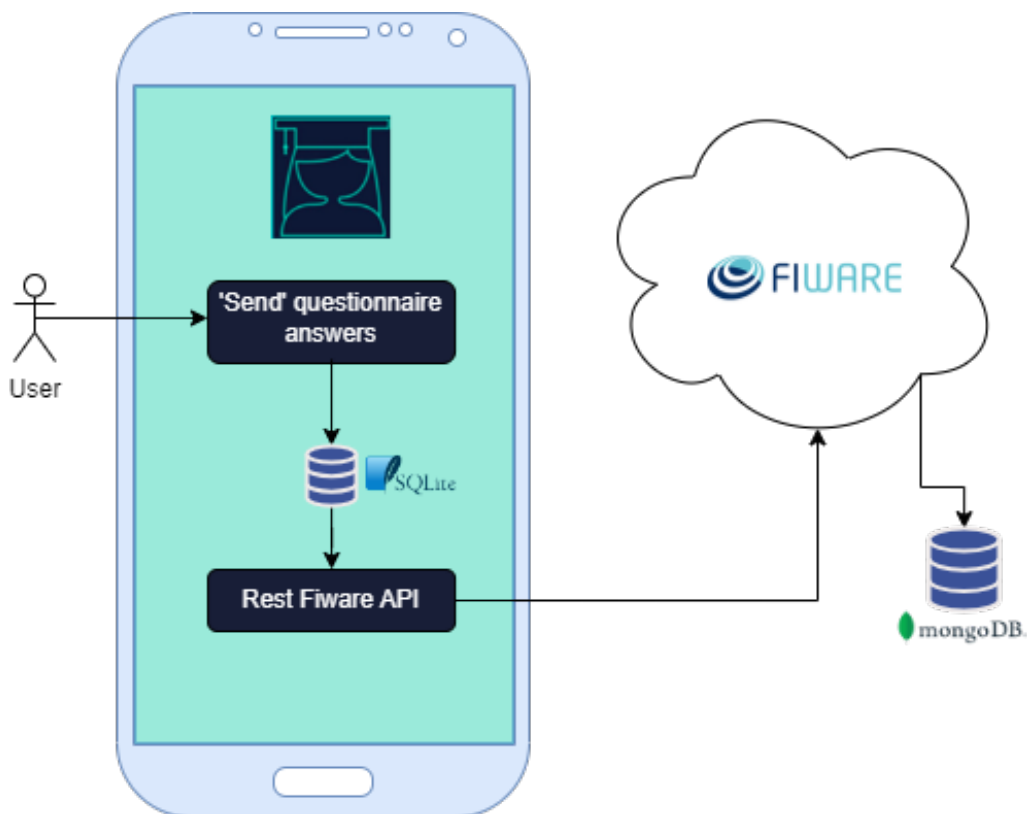


Figure 4.3: Questionnaire's architecture

4.2 Chatbot

The access to a mental health professional can be hard and take some time. Due to that, more people seek alternative means to get help, one of them being AI based systems. While they do not replace the experience and knowledge of a mental health specialist, such as a psychologist, they can be helpful when used to complement and give some suggestions in between sessions or while waiting for medical counseling. [46]

With that in mind, it would be useful to the purpose of the system to add the

ability to provide some recommendation regarding mental health to the chatbot's functionalities in ISABELA.

4.2.1 First approach - general recommendations through DialogFlow

As a first approach, DialogFlow functionalities were directly used.

The main idea was that, by considering each questionnaires' dimensions, and using known mean and SD values for each one, it would be possible to create triggers when a result for a certain dimension was considered concerning.

The way this process would work would be that, each time a user answers a questionnaire, the results are calculated and then compared to the mean value. For some dimensions, the results are considered concerning when they go above $mean + SD$, while for others it is when they go below $mean - SD$. When the condition to be considered concerning was met by the calculated result, a specific intent was sent to DialogFlow.

The mean and SD values for each dimension that was taken into account can be found at table 4.1. The dimensions considered were selected based on known correlations with students' motivation and performance, further described in subsection 4.2.2.1.

As an example, here is the explanation on how the process works when a user had results above $mean + SD$ for the exhaustion dimension in MBI.

The application calculates the result and detects it is above the threshold. Then, it sends a specific intent to DialogFlow. This intent has very specific and unlikely to be matched training phrase, so it is only matched by this automatic setting. A message is sent back to the user warning that a high value of exhaustion was detected. An output context is activated so a user can interact later by asking questions, and they can be correctly matched to other intents. A custom entity type was created, named '*Psychology_types*' to save the dimension that was triggered, and eventually this could be used in other conversation stages. In figure 4.4 it is possible to see how this was configured in DialogFlow.

Contexts

Add input context

5 help_exhaustion Add output context

Events

Training phrases Search training phrases

⚠️ Template phrases are deprecated and will be ignored in training time. More details [here](#).

When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use [annotations](#) with available [system](#) or [custom](#) entity types.

Add user expression

HIGH_EXHAUSTION_DETECTED

(a)

Action and parameters

Alarm.High_exhaustion

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
<input type="checkbox"/>	Psychology_type	@Psychology_types	\$Psychology_type	<input type="checkbox"/>
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

+ New parameter

Responses

DEFAULT +

Text Response

1 Your answers to the Burnout Form have revealed a high level of exhaustion.

2 Enter a text response variant

ADD RESPONSES

Set this intent as end of conversation

(b)

Figure 4.4: Intent for high values of exhaustion in DialogFlow

The user can then interact in one of two ways:

1. Ask to know more about what it means to have a high level of exhaustion
2. Ask about what actions to take to help

These questions will match new intents, based on what the user says but also on the context that was set by the first intent. When a user asks about what actions to take, then it will be given some recommendations, chosen randomly from the set

4. Implementation

defined in the responses field. Below are some images that show possible questions and the defined answers.

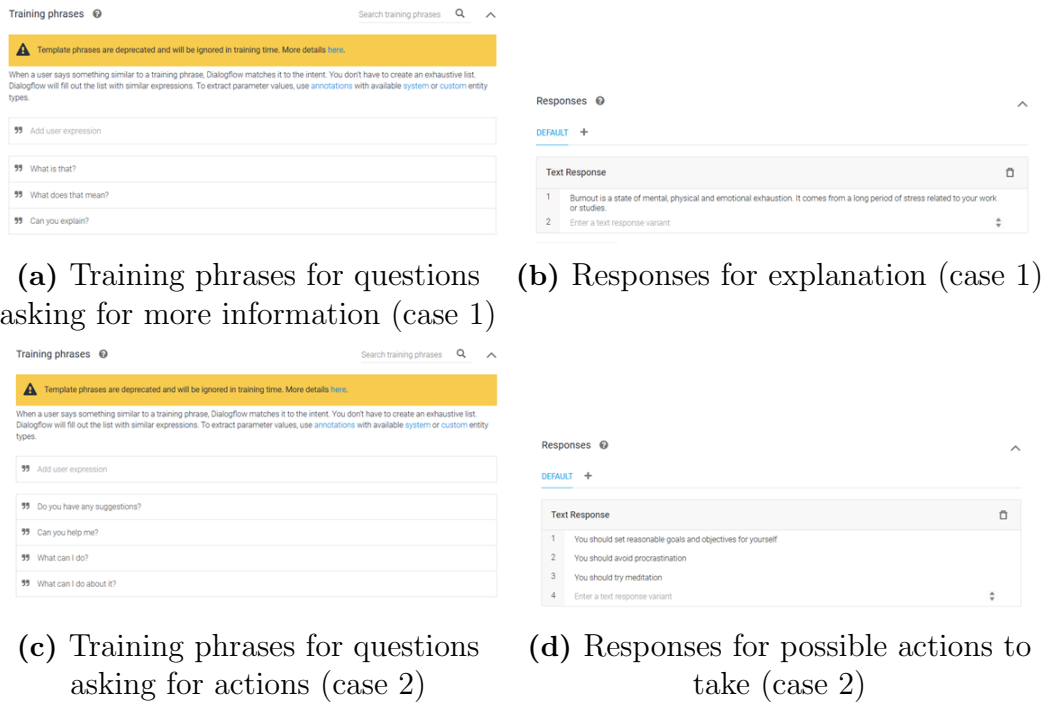


Figure 4.5

It is important to note that, by doing this, the answer given back each time would only be dependent on a randomization done by DialogFlow when a set of possible answers is given.

4.2.2 Strategy improvement - reinforcement learning model

The main problem with the first approach is that it only gave random recommendations from a preselected set, which made it too general and ignored the individuality of each user. It was a way of getting some information to the users about possible actions to take in order to improve their mental health, but it was not tailored to their specific characteristics measured by the questionnaires and by the various sensors that are collecting information in the application.

As a way to improve how ISABELA interacts with the user and use the information collected to its advantage, a reinforcement learning model was thought out to gather the information that comes from both the questionnaires and the application sensors in a meaningful way and transform it into personalized recommendations tailored to each individual's needs.

To start developing a RL model, it is important to understand the main con-

cepts and define how they apply to our specific situation.

The intention is to allow the chatbot in ISABELA to provide personalized recommendations depending on the user’s answers to questionnaires and data collected by the sensors. Using this intention to determine the main components of the model, the state will be defined by the user’s answers and the sensor’s data, and the actions will be the possible recommendations that the user can follow to change its state. There is one main component left to define, which is the reward. In this case, it will relate to the evolution of the values that define the state, namely the questionnaire’s results. If they evolve in the direction of less concerning values, then the recommendations are effective, and the agent will receive a positive reward. On the other hand, if the values remain the same or show a tendency to get worse, then the agent is negatively rewarded as the recommendations are not having the desired effect.

The following diagram (figure 4.6) shows the basic composition of the intended ISABELA’s RL model. The red arrows represent constant values, since both the personality and perfectionism questionnaires are only answered once. However, they can influence the state, so they need to be included in the algorithm.

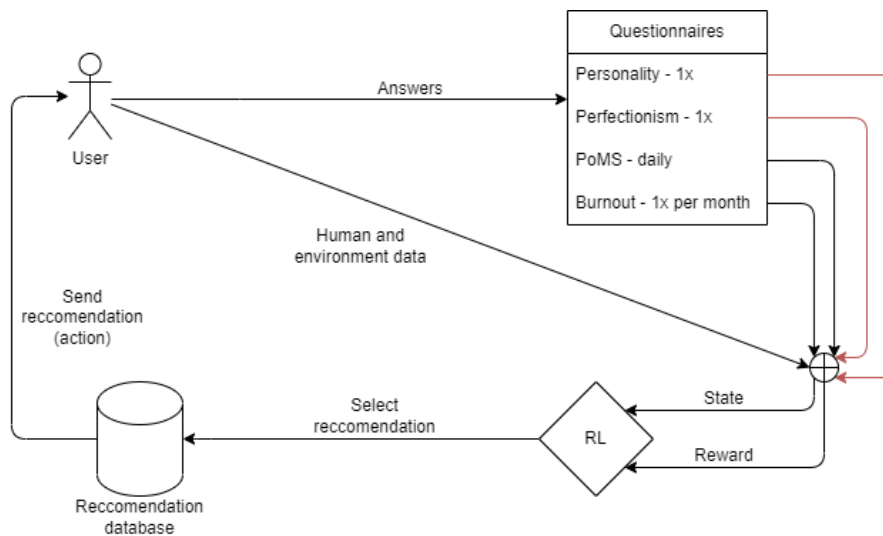


Figure 4.6: Reinforcement Learning method diagram

4.2.2.1 Questionnaires and mental state

As shown in the previous diagram, the questionnaires taken into account to the user’s state were the HEXACO Personality Inventory, the BTPS, the MBI and the PoMS-12.

The main goal of this reinforcement learning model is to help users improve their mental state by providing useful and personalized recommendations. An important step to being able to start implementing this is understanding how the results of the

questionnaires relate to an individual's emotions and overall mental condition.

The HEXACO Personality Inventory and the BTPS available in ISABELA are only answered once. Even though people mature and evolve their personality as they age, there is evidence of a rank-order stability, meaning that individuals tend to score around the same values, that increase throughout the years and stabilize around young adulthood.[26], [27] Perfectionism, being a personality trait, shares the same characteristics. Moreover, HEXACO shows a high test-retest reliability, so results tend to be consistent over time.[47]–[49] Given that the target audience of this app are college students, who are mostly young adults, and the duration of the study, at least in an initial phase, should be around one semester, it can be considered that the answers given are valid and reliable throughout this time span.

Despite being constants in this context, personality and perfectionism results need to be taken into account in the model as they affect the general state.

Regarding personality, from HEXACO's answers it is possible to calculate the values of six dimensions: Honesty/Humility, Emotionality, Extraversion, Agreeableness, Conscientiousness and Openness to Experience [25].

Personality traits have been shown to have a correlation with stress, academic achievement and overall emotional states in students [50], [51]. From these six dimensions, three were considered in the RL model due to having a stronger correlation to mental states. [52]–[54]

One of them is neuroticism, which in the HEXACO questionnaire is mainly represented by emotionality. It is positively correlated with stress and anxiety, which means that an individual with a high value in the emotionality dimension will be more prone to stressful situations and to have negative emotional reactions. It has the strongest association to stress among the personality traits, which is coherent with the fact that some aspects of their concepts overlap.[51]

The others are conscientiousness and extraversion, which are negatively correlated to mental disorders. [54] Individuals with higher values of conscientiousness and extraversion have higher regulation abilities, are more prone to be reward focus, be organized and plan ahead, tending to feel more positive emotions. These factors contribute to coping better with stressful situations and lower the levels of the perceived stress.[51], [55] Conscientiousness has also been studied as a factor that can influence general health throughout life. [56]

Regarding perfectionism, it is a particularly relevant personality characteristic and is measured by the BTPS. It relates to having extremely high standards and striving for perfection. Even though these are trends that are increasingly frequent and promoted in today's society, since most people try to achieve different kinds of

'perfection', depending on their life goals, when it is in an excessive and/or obsessive way, it might become unhealthy. [57], [58]

The BTPS measures three kinds of perfectionism: rigid, self-critical and narcissistic. Rigid perfectionism is more related to striving for perfection, self-critical perfectionism is more related to concern over mistakes and narcissistic is directing the demand for perfectionism towards others. [38] In ISABELA, the questions regarding the self-critical perfectionism were included, since there is evidence that this is the dimension most strongly (and prospectively) associated with psychological distress, namely in university students. [41] This is highly relevant for the purpose of the system.

Then there is MBI, that should ideally be answered at least once a month, but during exam periods it should be answered more frequently, every two weeks. This is due to the fact that there are prospective studies, including with Portuguese students, that show moderate to high within-subjects correlations in periods separated by twelve to eighteen months. [41], [59] This indicates that burnout is not variable in really short periods of time. However, burnout levels tend to be higher around exam periods, indicating that its severity can fluctuate throughout the school year. [60], [61] By answering the questionnaire with the previously mentioned frequency, it is possible to detect these fluctuations and take them into consideration for the reward calculation. All three dimensions are considered: efficacy, exhaustion and cynicism.

Lastly there is PoMS. As previously explained, the version used in this application has 12 mood states, that can be divided into three dimensions: depression, anxiety and positive affect. This questionnaire is usually answered in one of two ways:

- weekly - "How have you felt during the past week including today?"
- daily - "How do you feel right now?"

The reference to the past week is due to the fact that it can be considered a long enough period to capture people's typical and persistent emotional reactions to daily life events, but still short enough to assess reactive fluctuations and acute effects of an intervention. [62]

The scope of the reference period, either "past week" or "right now", can affect an individual's perception of the intensity, seriousness and frequency. [63] Evidence indicates that scores obtained when referencing "past week" were higher than the average scores in multiple "right now" instructions. [64], [65] This suggests that recall of mood can be influenced by mood at the time of recall and possible significant events. Thus, it is suggested by several authors that the "right now" response time

frame should be the method of choice when it is intended to evaluate the mood state in a valid and ecological way. [65]–[67] Taking that into consideration, in ISABELA the questionnaire is set to be answered daily, being the most frequently answered questionnaire.

These last two questionnaires will then be variable and contribute to the algorithm by influencing the reward as they change over time. So, an individual’s general state will be taking into account all four questionnaires and their dimensions, but the state evolution will depend on the user’s answers to the variable questionnaires.

4.2.2.2 Questionnaire’s threshold values

As described in section 4.2.1, to trigger the intents in DialogFlow, some threshold values were taken into consideration. The same threshold values were taken into consideration in the RL algorithm, for the definition of states. To understand how that works, it is important to understand what are those values and when they represent cases for concern or warning.

For the purpose of a threshold, the mean and SD of each questionnaire’s dimension was considered.

When a certain dimension calculated in a questionnaire is a positive correlate of psychological distress, the results should be considered as concerning when their value goes above $mean + SD$. On the other hand, if the dimension has a negative correlation with them, then the results should be considered concerning when their value goes below $mean - SD$. This applies to both personality and perfectionism.

Burnout and the PoMS questionnaire, on the other hand, measure states directly, instead of relating to them. For those, burnout, depression and anxiety should be considered concerning above $mean + SD$, while positive affect is concerning when it is below $mean - SD$.

It is important to mention that for the MBI in specific there are cut-off points, that were defined in the third edition of the MBI Manual [33], but eventually it was noticed that they did not align with some average values obtained in different populations. So, for this purpose, the values considered were the mean and SD from a study conducted with Portuguese students. [42]

In table 4.1 there are the values for mean and SD of every dimension considered in the algorithm.

Table 4.1: Means and SDs of selected HEXACO personality traits [48], perfectionism [43], burnout [68] and PoMS [31] questionnaires

Dimensions	Mean	SD
Personality		
Emotionality	34.06	5.74
Extraversion	33.81	5.94
Conscientiousness	37.36	5.17
Perfectionism		
Self-critical perfectionism	20.04	5.43
Burnout		
Efficacy	18.1803	6.00925
Exhaustion	15.1671	6.57678
Cynicism	7.6158	6.16717
PoMS		
Depression	4.7761	2.08308
Anxiety	5.7085	3.12693
Positive affect	5.2322	2.53220

By considering these values and comparing the obtained results against them, in the first approach a trigger was generated for a DialogFlow intent, and in this improved approach it is possible to define states.

4.2.2.3 Algorithm selection

There are numerous algorithms that can be used to try to achieve the best policy in a RL model. To develop the model's first version, the intention is to select a simple algorithm to try to prove the concept. Nevertheless, it is important to understand how the algorithms work to see if they are applicable to this case.

Dynamic Programming (DP) includes various algorithms to achieve the best policy; however, it requires the environment to have a perfect model, such as a MDP, and has a big computational cost [17].

In ISABELA's case, it is possible to define state, action, and a way to give reward. However, to have a perfect model, it was necessary that the transition probabilities could be accurately defined [17]. Since this problem deals with mental states described by answering questionnaires and collecting user and environmental data, it is not possible to define such probabilities with great certainty. Given that, DP algorithms are not fit for this problem.

Monte Carlo (MC) methods are a different type of algorithm that can achieve optimal policies without knowing the transition probabilities but instead only using experience. Said experience can come from actually interacting with the environment or by simulating interactions with the environment. These methods are based on averaging the returns and, in order for clear returns to be available, they should only be used for episodic tasks, being that the policy and value estimation will only be updated at the end of each episode [17]. As the intention in ISABELA is to be able to progressively learn through time while the user interacts with the app, the task is not episodic, but rather continuous.

Temporal-Difference (TD) methods combine the two previous categories. Like Dynamic Programming, they don't need to wait for an end to improve the policy, instead they can do that throughout the process. And like Monte Carlo methods, they don't need to have a perfect model to begin, but rather learn from experience. These kind of methods align better with the problem discussed here.

Inside the TD category, there are two main methods: State Action Reward State Action (SARSA) and Q-learning. The main difference is that SARSA is an on-policy method, meaning that it will always follow its own policy while training, and Q-learning is an off-policy method, meaning that it uses a different strategy for training while defining the optimal policy. A Q-learning method will directly approximate to the optimal value for the action-value function, no matter what the policy is, which makes it easier to analyse. Q-learning will always assume the maximum reward in the next action taken for the Q-value calculation, while SARSA is more conservative and assumes the reward if the current policy is followed. [17]

For the purpose of this work, which is to build a Proof of Concept (PoC) for a RL algorithm integrated in the application, Q-learning was used since it ends up being simpler and more straightforward.

It is important to reiterate that the purpose of this analysis was to get to a simple, yet appropriate, approach to start a RL algorithm for our use case. Of course

there are a lot more options for methods, including some that work together with Deep Learning (DL) and Neural Networks (NN), and it could be interesting to test different approaches as future work.

4.2.2.4 Q-learning algorithm

Starting with a more in-depth explanation of the Q-learning method itself, it is defined by [69]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (4.1)$$

- $S_t \rightarrow$ state at a certain time t
- $A_t \rightarrow$ action at a certain time t
- $\alpha \rightarrow$ learning rate, generally set between 0 and 1. The higher the value, the more impact each learning step will have on the Q-value and the quicker is the learning. [70]
- $R_{t+1} \rightarrow$ reward
- $\gamma \rightarrow$ discount factor, set between 0 and 1. It relates to how much the agent will consider future reward. The higher it is, the more importance the agent sets on future reward. [70]
- $\max_a Q(S_{t+1}, a) \rightarrow$ maximum Q-value for the action taken [70]

Equation 4.1 will update the Q-values, and will be included in an algorithm with the following steps:

Parameters: define parameters' values

Initialize: Q-table randomly, with the exception that $Q(\text{terminal}, \cdot) = 0$

for each episode do

Initialize state

for each step do

Choose action using policy (e.g ϵ -greedy)

Take action and observe the next state and the reward obtained

Apply equation 4.1

Replace state with the next state

end

Until state is terminal

end

Algorithm 1: Q-Learning algorithm pseudo-code [17]

An episode represents an independent event, such as a play of a game, or any sort of sequence of interactions that ends in a terminal state. A step is each interaction inside the episode. [17]

For the specific purpose of this work, the goal is to have the agent permanently learning from the user interactions. So, even though it was possible to consider the situation where all the questionnaires reveal answers within the expected values as a 'terminal state', for the purpose of developing an algorithm that is always interacting with the user and giving feedback, episodes were not considered, and the agent is always updating Q-values and doing the iterative process for each step. The implementation in itself will be further explained in the next subsection.

4.2.2.5 First implementation using Python

As a first step in implementing the RL algorithm, a code was developed in Python. This way, it was completely independent from the application, which made it easier to experiment and develop.

The algorithm used was Q-Learning and, in order to get a simple interface to interact with the code, a GUI was created using the *Tkinter* library available in Python.

In this section some main considerations will be made regarding its functionality and parameters. As previously mentioned, instead of considering separate episodes, the agent will update the Q-values at each iteration and keep learning continuously, instead of stopping at a terminal state and then restarting.

Starting off with the parameters defined. For the learning rate α , the value considered was 0.7. This value will define how fast the agent learns, given that the closer it is to 1, the faster the learning will be. However, if the value is too big, it might make the results less accurate. [70] By using 0.7, the intention is to still have a good accuracy, but a relatively fast learning, so the agent can start adapting to a user. Then for the discount factor γ , 0.95 was considered. The discount factor will determine how important are the short-term rewards. The closer it gets to 1, the more the agent will be concerned about the future and delay the rewards. [70] These values were defined for a first approach.

Now that the values of the needed parameters are defined, it is necessary to define a strategy to select the actions to take. A general strategy is the ϵ -greedy. This strategy tries to balance exploration and exploitation, by defining an ϵ value that will determine how much of the action selection is random (exploration) and how much of it is by the maximum Q-value available (exploitation). The value of ϵ should be between 0 and 1 will define the probability of choosing a random action, leaving the choice of the maximum reward with a probability of $1 - \epsilon$. [17]

In the beginning, the agent does not know much (barely anything) about the environment. As such, it needs to explore more, which means choosing more random

actions. On the other hand, once the Q-table starts to be more fulfilled, and the agent knows more about the environment, it should start to choose more based on what it knows, and less randomly. With this in mind, the ϵ value was defined as an exponential decay that starts at 0.95 and can go as low as 0.05, with a decay rate of 0.005. The evolution of the ϵ values through the algorithm steps is presented in figure 4.7

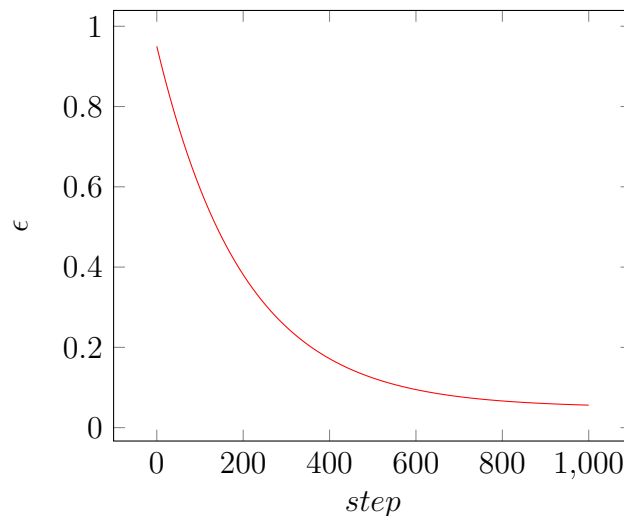


Figure 4.7: Exponential decay of ϵ values through the algorithm steps

In this algorithm, the actions will be mental health recommendations. As of this time, there are 35 recommendations, meaning there is a total of 35 actions. The complete list of actions can be consulted in appendix H.

As for the states, there are some things to take into consideration. For this first version, the factors taken into consideration are only the dimensions of the questionnaires mentioned in subsection 4.2.2.1. Values collected from sensors are not yet considered. For each of the questionnaires, there are various results. As an example, the HEXACO-60 has 10 questions per dimension, and each of them can be answered with integer values from 1 to 5. Considering one of the dimensions, such as neuroticism, the value of the results could go from 10 to 50, which totals 41 possible results for this dimension alone. Considering this for each of the 10 dimensions across the different questionnaires, some of them with even more possibilities, and combining them into overall states, there would be numerous possibilities. Taking into consideration the chosen algorithm and how it works, and also the fact that this is a PoC of the utility of this kind of approach, a simplification was made to reduce the number of states and make the algorithm simpler and clearer.

4. Implementation

This simplification takes into account the threshold values presented in table 4.1 and when the values are considered concerning or not, according to the explanation in subsection 4.2.2.2. This way, for each dimension, there are only two options, which are being above or below the threshold. Considering 2 options for each of the 10 dimensions, there is a total of 1024 possible combinations, so there are 1024 states. Each dimension will have a 0 if the value is considered concerning, and a 1 if it is not. Table 4.2 shows how the values will be for each dimension.

Table 4.2: Binary values set for each dimension

Dimensions	Below threshold	Above threshold
Emotionality	1	0
Extraversion	0	1
Conscientiousness	0	1
Self-critical perfectionism	1	0
Efficacy	0	1
Exhaustion	1	0
Cynicism	1	0
Depression	1	0
Anxiety	1	0
Positive affect	0	1

With this approach, there are 35 actions and 1024 states. This means that the Q-table will be 1024x35. This table will contain the Q-values for each pair state-action, and will be continuously updated. To start, it is initialized to all zeros. This means that the first recommendations will be random, as the system knows nothing about a user. With some time, it will start to give more personalized recommendations. The ϵ value also plays a part into this, by balancing exploration and exploitation as previously explained.

Having all of these definitions, there are enough bases to start developing the code.

The main function developed is *train()*, in which the general logic of the algorithm is defined. All the other functions were built around this. All the functions in the general Q-learning code can be fully seen in Appendix I.

The general idea is that there is a main loop that is always running. In this main loop, an action is selected based on an ϵ -greedy policy, where ϵ has the previously explained values. Then, there is a *yield* in between this part of the loop, and the remaining part that applies the algorithm. This is because, in order to calculate the Q-values, there needs to be a next state, and that next state will be user-dependent, since it is a combination of the questionnaire results. The *yield* expression suspends the execution and retains the current state until it is called again. [71]

Once there is a new state, the execution proceeds from the *yield* point, and will get the new state, calculate the reward, update the Q-value, and update the ϵ value. Then it goes back to the start of the loop, giving back a new recommendation (action) and waiting at *yield*.

To do all these operations, the *train()* function relies on various other functions and a database structure that withholds all the needed data for the process.

The database is composed of several tables, presented in figure 4.8.

recommendations			
id	Int	PK	AU
recommendation	Text		

personality			
id	Int	PK	AU
emotionality	Int		
extraversion	Int		
conscientiousne..	Int		
date	Date		

burnout			
id	Int	PK	AU
cynicism	Int		
efficacy	Int		
exhaustion	Int		
date	Date		

emotions			
id	Int	PK	AU
depression	Int		
anxiety	Int		
positive affect	Int		

perfectionism			
id	Int	PK	AU
perfectionism	Int		
date	Date		

states			
id	Int	PK	AU
emotionality	Bool		
extraversion	Bool		
conscientiousne..	Bool		
perfectionism	Bool		
depression	Bool		
anxiety	Bool		
positive_affect	Bool		
cynicism	Bool		
efficacy	Bool		
exhaustion	Bool		

user_state			
id	Int	PK	AU
emotionality	Int		
extraversion	Int		
conscientiousne..	Int		
perfectionism	Int		
depression	Int		
anxiety	Int		
positive_affect	Int		
cynicism	Int		
efficacy	Int		
exhaustion	Int		

Figure 4.8: Tables of database for RL algorithm

There is one table per questionnaire, to save the result values. Then there is a

table that saves all the recommendations. The recommendation to give is selected by its id, and then the text can be recovered by mapping through the database table. Lastly, there are two tables that save states. While *user_state* saves the states of a user, composed by the questionnaire results turned into boolean values as per table 4.2, the *states* table contains all possible 1024 states. This way, it is possible to match a user's state by comparing it to the possibilities in *states* table, and refer to each state by a single id.

To facilitate access and insertion of some database values, there are some queries defined inside functions that can then be used in the algorithm.

Describing some of the functionality around the main cycle, there are 2 functions that work as a trigger:

- *initial_trigger()* → will check if there is a state, which is only true once all the questionnaires have been fulfilled, and if so initialize the while loop
- *trigger_next_step()* → will trigger next run of the loop by continuing after the *yield*

These functions work as helpers, since answering questionnaires is what should trigger next steps, and the questionnaires are outside the loop. By calling these functions when a questionnaire is answered, it is possible to interact with the loop from outside.

Then, there is *epsilon_greedy_policy(state)* that will apply the policy by getting a random number between 0 and 1. If this number is below the defined ϵ value at that moment, then a random action is selected. Otherwise, the action with the highest Q-value is selected.

Lastly, there is the *calculate_reward()* function. As the naming suggests, this is where the reward is defined for a certain state-action pair. This is an essential part of the algorithm since it defines how the agent will be rewarded/punished, and that will have an impact on how the agent will learn the best actions to choose. There are some things to consider:

- Only the PoMS and MBI will be taken into consideration for the reward calculation, since they are the only ones that vary through time
- PoMS measures emotional states, which can be influenced by a lot of factors on a daily basis. To try to compensate for daily fluctuations due to random life events, it was defined a condition that the reward will only be calculated, and consequently a new Q-value, after the user answers this questionnaire 4 times
- Considering the previous decision, there was another factor to take into account. If a user does not answer regularly, then it is not possible to evaluate

the evolution of the results. To avoid considering answers with a long time in between, a threshold of one week was set. From the moment the user gets the recommendation, only questionnaire answers within a week will be taken into account. If there are no answers within a week, it is considered that there is not enough information.

- Since MBI is filled in less times, it is only considered for reward if it has been answered after the last recommendation was given. Otherwise it is a constant, so there is no need to include it as it will give a zero difference.
- When the questionnaire's results go from one side of the questionnaire's threshold values to the other, the reward/punishment should be bigger.

With all the previous information, the function was developed to calculate the reward based on the difference between the questionnaire's results for each dimension since the last recommendation was given and make a mean. The reward will be positive if the values move towards better values, and negative otherwise.

It is important to note that all these decisions were made based on deduction from the knowledge of how the questionnaires work. They are not definite, and extensive testing would be needed to prove they are the most efficient approach to reward calculation.

Once all this was functional, the *Tkinter* app was defined as a GUI for easy interaction. It has a simple layout, with four buttons, one to access each questionnaire. As this was not the end goal, but only a first implementation, and the objective was to later integrate this in ISABELA, instead of having the full questionnaires, the *Tkinter* app only has an input field for each dimension, where it is possible to directly write a result value. In figure 4.9 it is possible to see some screens of this GUI.



Figure 4.9: Screens of the GUI developed in *Tkinter*

To facilitate all of this developments, some Python libraries were used:

- **numpy** - offers a lot of mathematical tools. [72] It was mainly used to access

functions such as *max*, *exp*, and for arrays

- **datetime** - for date and time manipulation. [73] It was used to deal with the calculation of time between questionnaire’s answers.
- **sqlite3** - to create and interact with the databases directly in Python. [74]
- **pandas** - good for data analysis and manipulation, it was used for a dataframe to save the possible values of each questionnaire’s dimension, so they can be combined into all possibilities. [75]
- **itertools** - provides tools for iteration, and was used to combine possibilities and define all 1024 states. [76]
- **tkinter** - to develop the GUI. [77]

4.2.2.6 Integrate on ISABELA using Flask server

Once the first version in Python was finalized, and it seemed to provide the desired functionality, the next step was to integrate this algorithm into the ISABELA app so it could be available to the users.

Since the first implementation was in Python, in order to allow for reuse of some of the code, *Flask* was used. *Flask* is a Python module that allows for creation of HTTP methods. This can be used for web app development, or in this case, to develop a server that can be called from the mobile app, and implement the developed RL algorithm. [78]

To allow for best control over the data that is saved to the databases, these were defined inside the app by using SQLite, available for local storage in Xamarin. [79] So, in the *Flask* server, there are only methods related to the algorithm itself.

By having the databases on the application’s side, the data for reward calculation and to update the Q-table needs to be sent to the server, and the functions need to be adapted to receive it and use it.

As a way to validate the data that is sent to the server, it is possible to use the *jsonschema* library, to implement a JSON Schema validation in Python. [80] Let’s take as an example the initialization of the agent. To initialize the agent there is a POST method defined, and it will create an instance of the agent’s class with all the necessary parameters. Said parameters will be sent to the server from the app. To make sure all the mandatory data is sent, it is possible to define the schema with required fields, as well as their types and some restrictions they might have, such as minimum values. The schema for this initialization is presented in code snippet 4.3 as an example.

```

1 initialize_schema = {
2     "type" : "object",
3     "properties" : {
4         "num_states" : {"type": "integer", "exclusiveMinimum": 0},
5         "num_actions" : {"type": "integer", "exclusiveMinimum": 0},
6         "learning_rate" : {"type": "number"},
7         "gamma" : {"type": "number"},
8         "max_epsilon" : {"type": "number"},
9         "decay_rate" : {"type": "number"},
10        "min_epsilon" : {"type": "number"}
11    },
12    'required': ['num_states', 'num_actions', 'learning_rate', 'gamma',
13               'max_epsilon', 'decay_rate', 'min_epsilon']
14 }

```

Code snippet 4.3: JSON schema for agent initialisation

Another change that needed to be done was figure out a way to remove the *while(True)* loop, since in the server the idea is that the requests are received, processed, and an answer is sent back. Having an infinite loop in a server may cause issues, specially since it is accessed by several users at different times.

As a way to break this loop, what was previously the *train()* function, was split into two:

- The first part is the equivalent of what went on up until the *yield* was reached, which is retrieve a recommendation based on the ϵ -greedy policy
- The second part is the update of the Q-values with the reward calculation

Once the *train()* function was split into two, it was possible to create two different methods at two different endpoints that can be called by ISABELA in order to interact with the algorithm.

In total there will be three methods, the initialization one, defined as a POST for /initialize endpoint, the one to get recommendations, defined as a GET for /recommendation endpoint, and the one to update the Q-values, defined a PATCH to /qtable endpoint.

To make the whole process work, the initialization needs to be called when all four questionnaires have been answered at least once. Then, a first recommendation can be sent.

From then on, each time the PoMS is answered four times, the equivalent of a loop in the Python implementation needs to be done: update Q-values and send a new recommendation. For that, both the update Q-values method and the get recommendation method need to be called. The recommendations and states are passed as integers, and can be translated by using the databases showed in figure

4. Implementation

4.8, on ISABELA's side. Each time the get recommendations is called, ISABELA gets back an integer, that can be translated into the corresponding text by querying the correspondent table in database. Then, that text can be displayed to the user in the chatbot. An example can be seen in figure 4.10

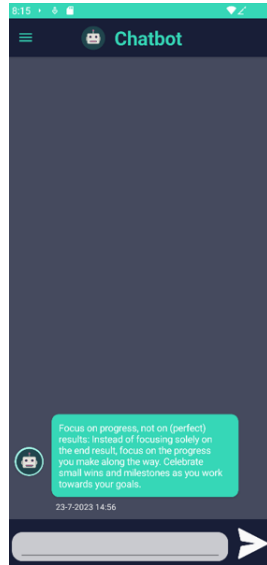


Figure 4.10: Recommendation displayed in ISABELA's chatbot

If the user is not answering the PoMS questionnaire with the required frequency, the RL algorithm cannot learn properly. Besides the general reminders the app sends to answer the questionnaires, an extra one was added to send a message to the chatbot, along with a notification, if the user is not answering at least four times a week. This message can be seen in figure 4.11.

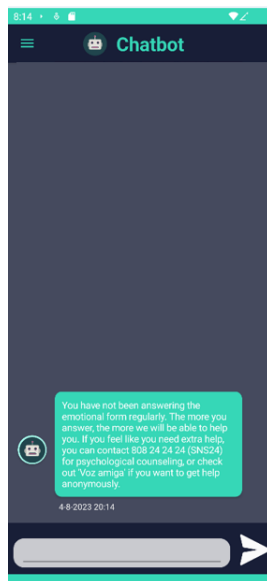


Figure 4.11: Warning message when PoMS is not answered frequently

To make sure the whole process works, the server will keep track of the algorithm information, such as the defined parameters, the Q-table, the last suggested action, in order to be able to know where to update the Q-value, and also how many times the PoMS has been answered. To make sure this information remains associated with the right user, the *Session* class was used. *Session* allows for the storage of data in the server, associated with a specific user, so it can be recovered the next time the same user interacts with it. It is necessary to define a secret key, to allow for encryption, and each client needs an unique id to be uniquely identified. [78], [81]

To better illustrate the whole process described, figure 4.12 has a sequence diagram with the overall steps.

The steps defined are the following:

1. The user will start answering questionnaires
2. Once all 4 questionnaires considered have been answered at least once, a call will be made to the server to initialize the RL agent
3. If the agent initializes successfully, a call is made to receive the first recommendation
4. If a recommendation id is successfully received, it will be mapped to the correspondent text and presented to the user in the chatbot
5. The user will continue to answer PoMS (and MBI, but that is not considered as a trigger to RL)
6. Every time a questionnaire is answered, a call is made to /initialize. This method, besides initializing the agent, also verifies if there is already an existing one.
7. If there is an existing agent, the Q-table will be updated
8. Call to get a new recommendation - the get recommendation method verifies if the questionnaire has been answered 4 times. If it has, gives back a recommendation id and restarts the count
9. If a recommendation id is received, show the recommendation to the user through the chatbot
10. Throughout the whole process, it is being verified if the user answers PoMS frequently. If not, a message will be posted in the chatbot and a recommendation sent.

In figure 4.12 there is a sequence diagram of this process.

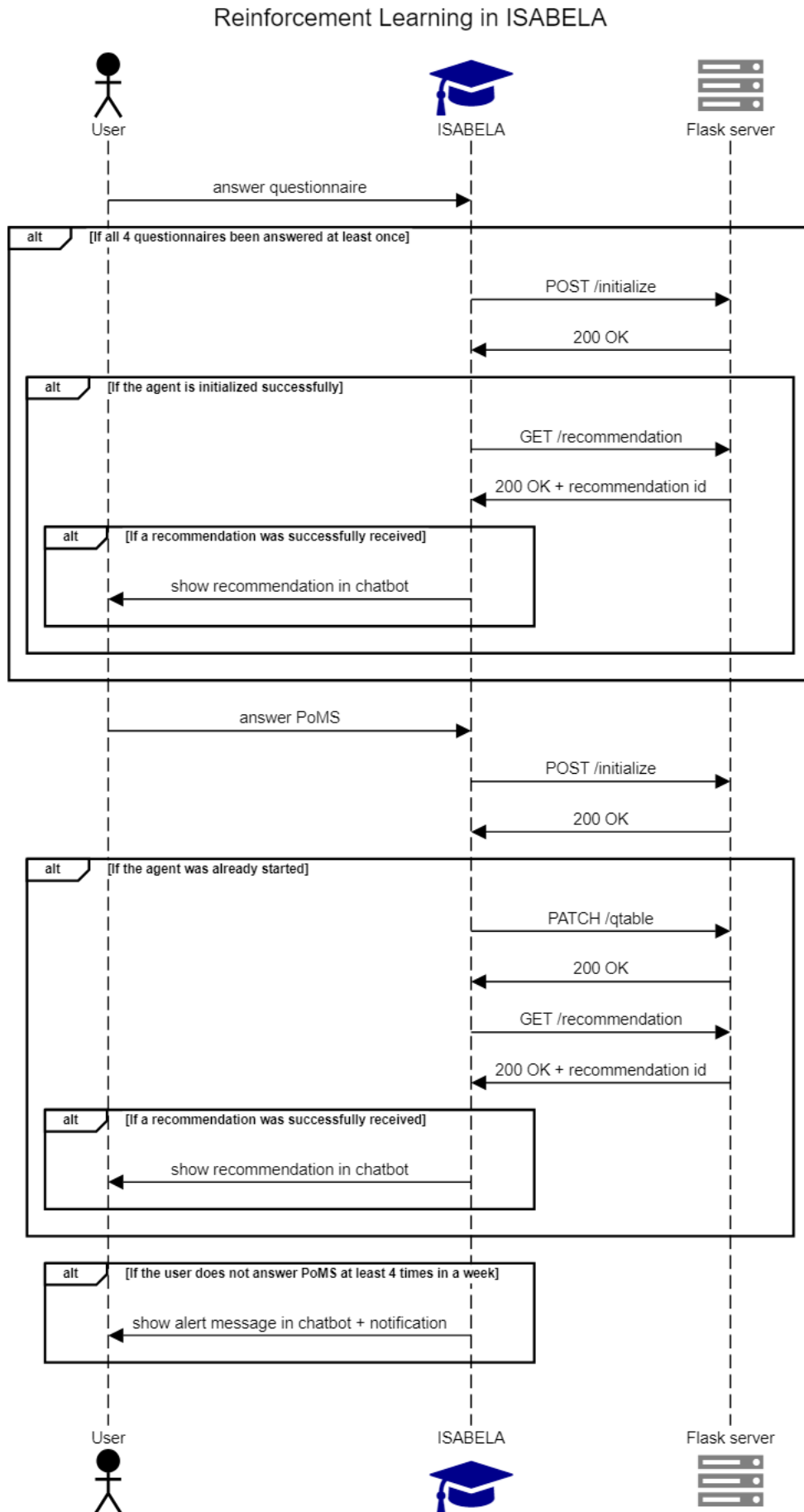


Figure 4.12: RL process in ISABELA

Functional and non-functional tests

This chapter focuses on the tests developed for the RL code, both for the first version in Python and for the version in Flask that was connected to the ISABELA application. Ideally, some real tests would have been done, with real users testing the application's new functionalities for some amount of time. That way, it would be possible to see if the code revealed good functionality when used as intended. However, due to the time available after the developments were concluded, this was not possible. The ideal testing period would be a semester, so all semester activities, including exam seasons, could be evaluated.

As real testing was not possible, to still validate functionality and usability to some extent, functional and non functional tests were developed, using Python libraries available for that effect.

5.1 Functional tests

Functional tests, as the name implies, aim to test the code's functionality. To make sure the code works as expected, different scenarios are tested and the expected output is compared with the actual output.

Python provides a library, *unittest*, that includes a lot of different options for automated tests. [82] This library was used to develop some functional tests to the initial Python code, as well as to some functions in the *Flask* code.

Starting with the initial Python code, tests were designed to test each individual function, as a way to understand if they worked well individually. Each function has its own complexity, being that the most complex one is *train*, as it is the central one, and calls other functions.

The goal here is to cover as much code as possible, by making small granular tests. In this section there is a description of what tests were made to each function, and those tests' results.

Function: *epsilon_greedy_policy*

The *epsilon_greedy_policy* is the function that will select the next action to be taken.

Test cases:

- 1 Random value is smaller than defined ϵ - exploration is selected, and a random action should be given back
- 2 Random value is bigger than defined ϵ - exploitation is selected, and the action with the best Q-value so far should be given back

To test these two cases, the random value generator needs to be simulated, so it is possible to test specific values and see if they follow the correct route in the choice.

For testing purposes, 100 states and 15 actions were considered. For the random action selection, there is a validation that the result is an integer and that is in range of the number of actions. For the bigger Q-value selection, in the test, a row of the Q-table is filled with values, and then there is a validation if the value obtained matches the maximum value in that row.

The ϵ value for testing is 0.95. Different values above and below were tested.

Table 5.1 shows values tested, the expected result and if the test was passed or failed.

Test case	Mocked value	Expected result	Actual result	Status
1	0.2	Integer in range [0,15)	8	Pass
1	0.7	Integer in range [0,15)	13	Pass
1	0.94	Integer in range [0,15)	0	Pass
2	0.95	10	10	Pass
2	0.96	7	7	Pass
2	1	2	2	Pass

Table 5.1: *epsilon_greedy_policy* test results

Function: *calculate_reward*

The *calculate_reward* is the function used to calculate the reward, based on the assumptions explained previously.

Test cases:

- 1 The emotional questionnaire was not answered in the 7 days following the last

recommendation, so there is not enough information

- 2 The emotional questionnaire was answered more than once, but all in the same day of the last recommendation, so there is not enough information
- 3 The emotional questionnaire was answered more than once in the 7 days following the last recommendation, but the burnout was not answered in that time span
- 4 The emotional questionnaire was answered more than once in the 7 days following the last recommendation, and the burnout was also answered in that time span, so it needs to be considered

Since this function gets the questionnaire answers from a database, to test each of the cases, the function to get the answers from the database was mocked with different values. The expected results were manually calculated, and then compared to what the function was actually giving back.

For test cases 3 and 4, some different values were tested, to test situations with positive reward, negative reward, and with values passing through the threshold.

Table 5.2 shows the test cases, with expected and actual results, and if the test passed.

Test case	Expected result	Actual result	Status
1	0	0	Pass
2	0	0	Pass
3	1	1	Pass
3	-1	-1	Pass
3	6	6	Pass
4	2	2	Pass
4	-2	-2	Pass
4	12	12	Pass

Table 5.2: *calculate_reward* test results

The *epsilon_greedy_policy* function is exactly the same in the *Flask* code, so these tests are valid for both. The *calculate_reward* only differs as it takes the questionnaires' results as inputs instead of getting them directly from the database.

However, all the logic and options are the same, so these tests were also considered valid.

Now, three functions remain: *initial_trigger*, *trigger_next_step* and *train*. The *initial_trigger* and *trigger_next_step* are helper functions, to make the *train* function work properly. They are really simple, and do not take different inputs, neither give back different results. They do have *if* statements. For *trigger_next_step* the option is based directly on a boolean and it does not make anything else besides running the next iteration of *train* or nothing of false. For the *initial_trigger*, there is an *if/else*, based on if there is a full state or not, but the output is just starting *train* or logging a warning that not all forms are fulfilled.

Function: *initial_trigger*

Test cases:

- 1 The questionnaires have not yet been all answered
- 2 The questionnaires have all been answered

Function: *trigger_next_step*

Test cases:

- 1 The training has started
- 2 The training has not yet started

Opposite to the functions tested so far, these ones do not give back results, they simply interact with the *train* function, by making the generator created due to the *yield* statement go to the next iteration. So, there aren't any expected results. However, by mocking a generator and making the conditions to go to each of the previous test cases, it is possible to assert that the *next* function was or was not called, and that way verify the right condition is being performed.

Table 5.3 shows the expected actions, and the actual actions, as well as the test status.

Function	Test case	Expected action	Actual action	Status
<i>initial_trigger</i>	1	Print information	Print information	Pass
<i>initial_trigger</i>	2	Start agent and call 'next'	Start agent and call 'next'	Pass
<i>initial_trigger</i>	1	Call 'next'	Call 'next'	Pass
<i>initial_trigger</i>	2	No action	No action	Pass

Table 5.3: Helper functions test results

Function: *train*

Finally, there is the *train* function left. This is the central function, and it calls others. For testing purposes, *epsilon_greedy_policy* and *calculate_reward* were mocked, since they were tested separately. This function does not have any conditions, it is simply a continuous loop that should give back actions at *yield* and updates Q-values. There are no different test cases, so the test consisted in running the complete loop, and checking if all values involved were being correctly updated.

First, the function runs up until the *yield* and it is possible to check if the value returned is the same as the one given by the mocked *epsilon_greedy_policy*. Then, the function continues after *yield* and it is possible to check if the value in the Q-table is correctly calculated given the value of the mocked *calculate_reward* and if *epsilon* was properly updated.

Table 5.4 shows the results of this loop test.

Parameter	Expected value	Actual value	Status
Action	2	2	Pass
Q-table value	0.7	0.7	Pass
Epsilon	0.9455	0.9455	Pass

Table 5.4: *train* test results

For the *Flask* code, the *train* function was split into two, and there are no helper functions. So, different tests were made for each of the functions.

Function: *getRecommendation*

This function takes the state as input, checks to see if the emotional questionnaire was answered 4 times, and then uses the *epsilon_greedy_policy* to give back a recommendation.

Test cases:

- 1 There are not yet 4 answers to the emotional questionnaire, so no recommendation is given and one answer is added to the count
- 2 The emotional questionnaire was answered 4 times, so the *epsilon_greedy_policy* is applied to get a recommendation

Table 5.5 shows the results for each test case.

Test case	Expected result	Actual result	Status
1	None	None	Pass
2	2	2	Pass

Table 5.5: *getRecommendation* test results

Function: *updateQTable*

This function takes the new state and the answers to emotional and burnout questionnaires (if burnout was answered since last recommendation), calculates the reward, and updates the Q-table.

Test cases:

- 1 Agent is not started, so it is not possible to update Q-values
- 2 There are less than 4 answers to the emotional questionnaire, so the Q-table will not be updated
- 3 The agent is training and there are 4 answers to emotional questionnaire, so Q-value is updated

For all the test cases, the return from the function is only a string that indicates what happened. But for the last case, besides the string return, there are also updates made to the Q-table and to other class parameters. So, all of those were checked to make sure they were updated correctly.

Table 5.6 shows the results of the tests for each test case.

Test case		Expected result	Actual result	Status
1		"Agent is not training so it is not possible to update Q-Table"	"Agent is not training so it is not possible to update Q-Table"	Pass
2		"Not enough answers to update Q-Table"	"Not enough answers to update Q-Table"	Pass
3	return value	"The update of the Q-table was successful"	"The update of the Q-table was successful"	Pass
	Q-table value	0.7	0.7	Pass
	epsilon value	0.9455	0.9455	Pass

Table 5.6: *getRecommendation* test results

This concludes the functional tests, for all functions developed. In the next section, non-functional tests are presented. For the *Flask* endpoints only non-functional tests were performed, as they make use of already tested functions.

5.2 Non-functional tests

Besides testing functionality, it is also relevant to test non-functional aspects such as performance and load testing. These tests were only done for the *Flask* code, since it is where it is more relevant, as this code is intended to be available to multiple users and it should be able to provide fast responses.

There are three endpoints, as explained previously, and these tests were performed for all three of them.

Firstly, some performance tests were made. To perform these tests, the libraries *pytest*, *requests*, *csv* and *time* were used. The library *pytest* was simply used as a test runner, and by making use of of a custom 'mark' it was possible to categorize

the tests as performance tests, for code organization. [83] The *requests* library was used to call the *Flask* endpoints, as it allows to call different endpoints and send headers, query parameters and body. [84] The *csv* and *time* libraries were helpers, *csv* allowed for saving test results into a csv file for analysis, [85] and *time* allows for calculation of the time differences. [86]

For performance testing, a loop was created that calls an endpoint 100 times. The time that each call takes to be completed is recorded and saved to a *csv* file. Then, the mean, minimum value and maximum value were calculated and are here presented in table 5.7.

Endpoint	Minimum value	Maximum value	Mean
/initialize	0.02419	0.08807	0.03646
/recommendations	0.047141	0.186566	0.066628
/qtable	0.081525	0.257489	0.105654

Table 5.7: Performance testing on *Flask* endpoints

The values obtained are fairly fast, since they are all below 1 second.

Besides checking for the response time of the endpoints when they are called individually, it is also important to check if they can handle different users interacting with them at the same time. To check for that, some load tests were performed.

To develop some load tests, the *locust* library was used. With this library, it is possible to create tasks that should be run, in this case, calls to the endpoints. When a test is ran, a class is created for each simulated user. Simulated users can execute tasks, on random orders and with some interval in between. The initialize endpoint is called inside *on_start*. That way, each time a user is started, it will run the initialize. This way, errors by lack of the learning agent are avoided, since they would have an impact in the report, but they do not actually matter in the context of load testing. [87]

When a load test is run using *locust*, there are some parameters that can be defined. In these test, the parameters used were the number of users, that indicates the maximum number of concurrent users, the spawn rate, that defines how many users per second should spawn, and the run time, that defines for how long the test will run. The test results can be exported to an html file with tables and graphics that summarize the test results. [88]

The run time for each test was 15 minutes, with 100 as number of users and 10 as spawn rate. This is a fairly small number. In reality, there is no data of how

many concurrent requests are expected. The reason a small number was selected was because the *Flask* code hasn't been yet deployed to a production or production-like server. So, the tests were performed locally, using localhost. This brings limitations, as the availability is highly related to the available resources of the machine hosting. So, the results are not realistic. However, the tests were still performed as a way to observe the behaviour of the endpoints, if any of them tends more to fail than the others, and to have a general overview of concurrent performance.

Figure 5.1 shows the number of users over time. As the users grew at a rate of 10 per second, they went up to 100 quickly, and remained at 100 throughout the rest of the time.

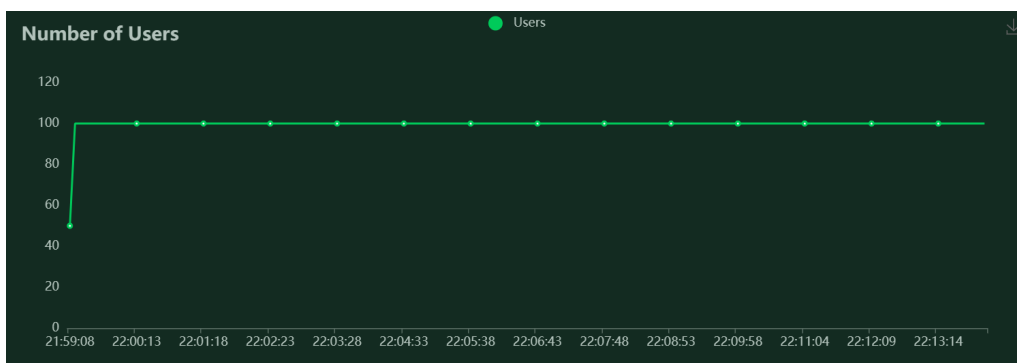


Figure 5.1: Number of users over time

Figure 5.2 shows the requests per second as well as the failures per second. It is possible to see they are directly correlated by the way the lines resemble each other. This indicates that the failures observed can be related to the load.

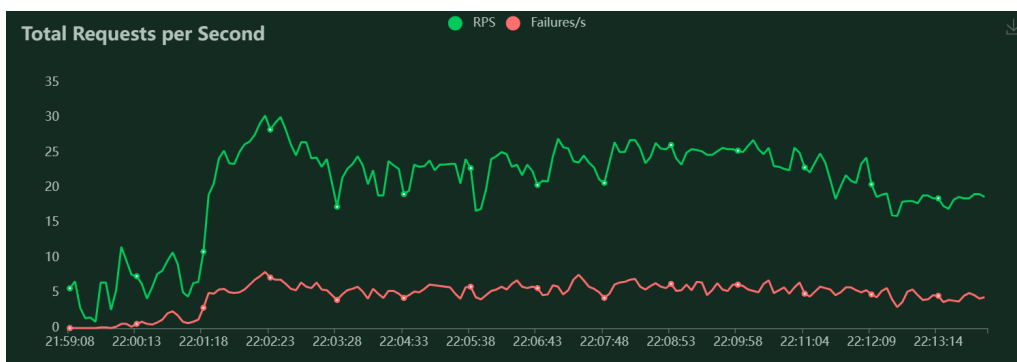


Figure 5.2: Requests per second over time

Checking the statistics, it is possible to see that there was a total of 18429 requests, with 4281 being failures, which comes to about 23.23%. From the 4281 total failures, 4239 were when calling `/qtable` endpoint. This indicates that `/qtable` might need more resources to fully run, and has the number of requests increases,

it starts to fail more. This also adds to the fact that it has the largest average size (in bytes). The code for `/qtable` is also the one that makes more calculations, since it calculates reward, the Q-value and a new epsilon value.

Like previously mentioned, these tests were performed in a localhost server, so they do not reflect the real behaviour in a production server. Once this code is deployed, more accurate tests can be performed. However, for the performance tests it is possible to see that when the code runs individually, it is fast. From the load testing, it is clear that `/qtable` takes more resources, and this could be further verified in a production server, to make sure there are enough resources for it to run properly.

Conclusions and future work

6.1 Conclusions

The purpose of this work was to improve the ISABELA system by adding features and start giving the users feedback based on the collected data, and that way close the loop of aims to be a personalized HITL application that provides students with useful recommendations according to their specific situations.

To achieve this, some more psychology related questionnaires were added, that measure important dimensions to analyze a student's mental well-being and general health, and a PoC of a RL system to give recommendations through the chatbot was implemented.

All of this was accomplished by using Xamarin as the main platform, as it allows to develop shared code between Android and iOS, and takes away the need to learn OS-specific languages, but also Python as it is a language that offers a lot of different libraries for all kinds of functionalities, and is widely used in machine learning, namely in RL.

Overall, the goals proposed for this dissertation were achieved, and will contribute to the next steps on making ISABELA a full HITL system, that can contribute to students' motivation on a daily basis, by providing an interface that is easy to use but also helpful.

6.2 Future work

This work presents the first steps towards having a full HITL system, that can gather the users' information and use it for their own benefit.

Some decisions were made throughout the development to work towards a simple PoC that could be functional and show the potential use of RL systems in this kind of application. Each of those decisions has an impact that needs to be further tested so all the validity can be verified, and changes can be made to improve as necessary.

First of all, the selected algorithm to use was Q-learning. This algorithm is vastly used in a lot of RL applications, however it can have some limitations and it would be interesting to try out more complex algorithms, such as using DL with application of NN, that can be more complex to develop, but also more adequate to complex environments [89], such as the one considered here, that is related to students' states.

Another thing that can be tested and improved are the parameters values. Also, in the calculation of reward, new approaches can be tested, such as a different number of days in between recommendations, the way the reward itself is calculated, which for now is a mean of the differences in values since the last time, but other ideas discussed included making a linear approximation of the last four values to define a tendency of evolution.

It is also possible to have some pre-training. Instead of initialising the Q-table with all zeros, some values could be introduced to make the algorithm tend towards recommendations that are scientifically known to be better for certain states.

Taking the PoC developed in the scope of this thesis, it is possible to make several new versions and test them, moving towards improving the methods and the algorithm's performance. This requires further testing, ideally with a groups of students using the apps during an extended period of time, such as a semester, to have reliable information to make an analysis, which goes over the time available in the scope of this thesis.

Appendices

A

HEXACO-60 Personality Inventory

For these questionnaire, the answers go from 1 to 5. To calculate the values for each dimension evaluated, some questions are considered directly, and some are considered in reverse, meaning that 1 converts to 5, 2 converts to 4, and vice-versa, while 3 remains 3. The values considered in reverse are presented with *R*. [25]

Here the calculations used for each dimension are presented. This is valid for both English and Portuguese versions.

$$\text{Honesty-humility} = 6 + 12R + 18 + 24R + 30R + 36 + 42R + 48R + 54 + 60R$$

$$\text{Emotionality} = 5 + 11 + 17 + 23 + 29 + 35R + 41R + 47 + 53R + 59R$$

$$\text{Extraversion} = 4 + 10R + 16 + 22 + 28R + 34 + 40 + 46R + 52R + 58$$

$$\text{Agreeableness} = 3 + 9R + 15R + 21R + 27 + 33 + 39R + 45 + 51 + 57R$$

$$\text{Conscientiousness} = 2 + 8 + 14R + 20R + 26R + 32R + 38R + 44R + 50 + 56R$$

$$\text{Openness to experience} = 1R + 7 + 13 + 19R + 25 + 31R + 37 + 43 + 49R + 55R$$

Now that the calculation of results is described, here is the full questionnaire.

Please read the following affirmations and decide if you disagree. Use the following scale. Strongly disagree, Disagree, Neutral, Agree, Strongly agree.

Table A.1: HEXACO-60 questionnaire [25]

Statement
1. I would be quite bored by a visit to an art gallery.
2. I plan ahead and organize things, to avoid scrambling at the last minute.
3. I rarely hold a grudge, even against people who have badly wronged me.
4. I feel reasonably satisfied with myself overall.
5. I would feel afraid if I had to travel in bad weather conditions.

A. HEXACO-60 Personality Inventory

Statement
6. I wouldn't use flattery to get a raise or promotion at work, even if I thought it would succeed.
7. I'm interested in learning about the history and politics of other countries.
8. I often push myself very hard when trying to achieve a goal.
9. People sometimes tell me that I am too critical of others.
10. I rarely express my opinions in group meetings.
11. I sometimes can't help worrying about little things.
12. If I knew that I could never get caught, I would be willing to steal a million dollars.
13. I would enjoy creating a work of art, such as a novel, a song, or a painting.
14. When working on something, I don't pay much attention to small details.
15. People sometimes tell me that I'm too stubborn.
16. I prefer jobs that involve active social interaction to those that involve working alone.
17. When I suffer from a painful experience, I need someone to make me feel comfortable.
18. Having a lot of money is not especially important to me.
19. I think that paying attention to radical ideas is a waste of time.
20. I make decisions based on the feeling of the moment rather than on careful thought.
21. People think of me as someone who has a quick temper.
22. On most days, I feel cheerful and optimistic.
23. I feel like crying when I see other people crying.
24. I think that I am entitled to more respect than the average person is.
25. If I had the opportunity, I would like to attend a classical music concert.

Statement
26. When working, I sometimes have difficulties due to being disorganized.
27. My attitude toward people who have treated me badly is “forgive and forget”.
28. I feel that I am an unpopular person.
29. When it comes to physical danger, I am very fearful.
30. If I want something from someone, I will laugh at that person’s worst jokes.
31. I’ve never really enjoyed looking through an encyclopedia.
32. I do only the minimum amount of work needed to get by.
33. I tend to be lenient in judging other people.
34. In social situations, I’m usually the one who makes the first move.
35. I worry a lot less than most people do.
36. I would never accept a bribe, even if it were very large.
37. People have often told me that I have a good imagination.
38. I always try to be accurate in my work, even at the expense of time.
39. I am usually quite flexible in my opinions when people disagree with me.
40. The first thing that I always do in a new place is to make friends.
41. I can handle difficult situations without needing emotional support from anyone else.
42. I would get a lot of pleasure from owning expensive luxury goods.
43. I like people who have unconventional views.
44. I make a lot of mistakes because I don’t think before I act.
45. Most people tend to get angry more quickly than I do.
46. Most people are more upbeat and dynamic than I generally am.
47. I feel strong emotions when someone close to me is going away for a long time.

A. HEXACO-60 Personality Inventory

Statement
48. I want people to know that I am an important person of high status.
49. I don't think of myself as the artistic or creative type.
50. People often call me a perfectionist.
51. Even when people make a lot of mistakes, I rarely say anything negative.
52. I sometimes feel that I am a worthless person.
53. Even in an emergency I wouldn't feel like panicking.
54. I wouldn't pretend to like someone just to get that person to do favors for me.
55. I find it boring to discuss philosophy.
56. I prefer to do whatever comes to mind, rather than stick to a plan.
57. When people tell me that I'm wrong, my first reaction is to argue with them.
58. When I'm in a group of people, I'm often the one who speaks on behalf of the group.
59. I remain unemotional even in situations where most people get very sentimental.
60. I'd be tempted to use counterfeit money, if I were sure I could get away with.

B

HEXACO-60 Personality Inventory - Portuguese version

For this questionnaire, the results are calculated using the same formulas presented in appendix A. Here is the full questionnaire in its Portuguese version.

Por favor, leia-as seguintes afirmações e decida o quanto concorda ou discorda com cada uma. Use a seguinte escala de avaliação Discordo Fortemente, Discordo, Neutro, Concordo, Concordo Fortemente

Table B.1: HEXACO-60 questionnaire - Portuguese version [90]

Statement
1. Ficaria muito entediado(a) ao visitar uma galeria de arte.
2. Planeio e organizo as coisas com antecedência, para evitar deixar tudo para a última hora.
3. Raramente guardo rancor, mesmo contra pessoas que me enganaram bastante.
4. No geral, sinto-me razoavelmente satisfeito(a) comigo mesmo.
5. Ficaria assustado se tivesse que viajar em más condições meteorológicas.
6. Eu não usaria a bajulação para obter um aumento ou uma promoção no trabalho, mesmo quando acho que se o fizesse teria sucesso.
7. Interesso-me em aprender a História e Política de outros países.
8. Geralmente exijo bastante de mim quando pretendo atingir um objetivo.
9. As pessoas às vezes dizem-me que eu sou demasiado crítico(a) em relação aos outros.

Statement
10. Raramente expresso as minhas opiniões em reuniões de grupo.
11. Por vezes, não consigo deixar de me preocupar com pequenas coisas.
12. Se soubesse que nunca seria apanhado(a), eu roubaria um milhão de euros.
13. Gostaria de criar uma obra de arte como um romance, uma canção ou uma pintura.
14. Quando estou a trabalhar nalguma coisa, não presto atenção aos pequenos detalhes.
15. Às vezes dizem-me que eu sou demasiado teimoso(a).
16. Prefiro trabalhos que envolvam interação social ativa do que trabalhar sozinho(a).
17. Quando eu sofro uma experiência dolorosa, necessito de alguém para me fazer sentir confortável.
18. Ter muito dinheiro não é especialmente importante para mim.
19. Acho que prestar atenção a ideias radicais é uma perda de tempo.
20. Tomo decisões baseadas mais na sensação do momento do que após pensar cuidadosamente
21. As pessoas acham que eu sou uma pessoa que se irrita com facilidade.
22. Na maior parte dos dias, eu sinto-me animado(a) e otimista.
23. Sinto vontade de chorar quando vejo outras pessoas chorar.
24. Acho que mereço mais respeito do que a média das pessoas.
25. Se eu tivesse oportunidade, gostaria de ir a um concerto de música clássica.
26. Quando eu estou a trabalhar, às vezes tenho dificuldades por ser muito desorganizado(a).
27. A minha atitude para com as pessoas que me trataram mal é esquecer e perdoar.
28. Sinto-me uma pessoa impopular.

Statement
29. Quando se trata de perigo físico, eu tenho bastante medo.
30. Se eu quiser alguma coisa de uma determinada pessoa, vou rir-me até das suas piores piadas.
31. Nunca gostei muito de procurar coisas numa enciclopédia.
32. Eu faço apenas o mínimo de trabalho necessário para ir vivendo.
33. Eu tendo a ser benevolente ao julgar as outras pessoas.
34. Em situações sociais, normalmente sou eu quem dá o primeiro passo.
35. Preocupo-me menos do que a maioria das pessoas.
36. Eu nunca aceitaria um soborno, mesmo que fosse de um valor muito lato.
37. As pessoas já me disseram que eu tenho uma boa imaginação.
38. Procuro ser sempre preciso(a) no meu trabalho, mesmo que à custa do tempo.
39. Habitualmente, sou bastante flexível nas minhas opiniões quando não concordam comigo.
40. A primeira coisa que eu faço quando estou num lugar novo é fazer novos amigos.
41. Sou capaz de lidar com situações difíceis, sem necessitar do apoio emocional de ninguém.
42. Teria bastante prazer em possuir bens de luxo.
43. Gosto de pessoas com pontos de vista não convencionais.
44. Cometo muitos erros, porque eu não penso antes de agir.
45. A maioria das pessoas costuma zangar-se mais facilmente do que eu.
46. A maioria das pessoas são mais bem-humoradas e dinâmicas do que eu geralmente.
47. Sinto emoções fortes quando alguém que me é próximo vai estar longe duante muito tempo.

Statement
48. Eu quero que as pessoas saibam que eu sou uma pessoa importante de elevado estatuto.
49. Não me considero uma pessoa artística ou criativa.
50. As pessoas geralmente chamam-me perfeccionista.
51. Mesmo quando as pessoas cometem muitos erros, eu raramente digo algo de negativo.
52. Por vezes sinto-me uma pessoa sem valor.
53. Mesmo numa emergência, nunca entro em pânico.
54. Não iria fingir gostar de uma pessoa só para obter favores da mesma.
55. Acho aborrecido discutir filosofia.
56. Prefiro fazer aquilo que me vem à cabeça do que me fixar num plano.
57. Quando as pessoas me dizem que eu estou errado, a minha primeira reação é começar a discutir com elas.
58. Quando eu me encontro num grupo de pessoas, geralmente sou eu quem fala em nome do grupo.
59. Permaneço imperturbável mesmo em situações em que a maioria das pessoas se tornam bastante sentimentais.
60. Sentir-me-ia tentado(a) a usar dinheiro falsificado, se tivesse a certeza de que nunca me apanhariam.

C

PoMS-12 - Portuguese version

For this questionnaire, the answers go from 0 to 4. The results are simply calculated by summing the values of each individual item, for each dimension.

Here it is presented the Portuguese version of the 12 moods of the questionnaire.

Table C.1: Mood states per dimension used in ISABELA's PoMS-12 implementation - Portuguese version [31]

Mood dimensions	Mood states
Depressão	Triste Pessimista Desesperado(a) Inútil
Ansiedade	Irritado(a) Rabugento(a) Com a neura Ansioso(a)
Afeto positivo	Animado(a) Ativo(a) Alegre Seguro de si

D

Maslach Burnout Inventory

For this questionnaire, the answers are scored from 0 to 6, and the results are calculating by adding the answers to each individual question being that questions 1 to 5 relate to the exhaustion dimension, questions 6 to 9 relate to cynicism and the remaining questions relate to efficacy.

Here is the English version of the MBI for students.

Please, read the following statements and decide about the frequency in which you feel as described. Use the following values: Never, a few times per year, once a month, a few times per month, once a week, a few times per week, every day.

Table D.1: MBI [36]

Statement
1. I feel emotionally drained by my studies.
2. I feel used up at the end of a day at university.
3. I feel tired when I get up in the morning and I have to face another day at university.
4. Studying or attending a class is really a strain for me.
5. I feel burned out from my studies.
6. I have become less interested in my studies since my enrollment at the university.
7. I have become less enthusiastic about my studies.
8. I have become more cynical about the potential usefulness of my studies.
9. I doubt the significance of my studies.
10. I can effectively solve the problems that arise in my studies.

D. Maslach Burnout Inventory

Statement
11. I believe that I make an effective contribution to the classes that I attend.
12. In my opinion, I am a good student.
13. I feel stimulated when I achieve my study goals.
14. I have learned many interesting things during the course of my studies.
15. During class I feel confident that I am effective in getting things done.

E

Maslach Burnout Inventory - Portuguese version

Here it is presented the Portuguese version of the MBI for students. The results are calculated as described in appendix D.

Por favor, leia as seguintes afirmações e decida sobre a frequência com que se sente da forma descrita. Use a seguinte escala de avaliação: Nunca, Quase nunca, Algumas vezes, Regularmente, Bastantes vezes, Quase sempre, Sempre

Table E.1: MBI - Portuguese version [35]

Statement
1. Os meus estudos deixam-me emocionalmente exausto.
2. Sinto-me de 'rastos' no final de um dia na universidade.
3. Sinto-me cansado quando me levanto de manhã e penso que tenho de enfrentar mais um dia na universidade.
4. Estudar ou assistir a uma aula deixam-me tenso.
5. Os meus estudos deixam-me completamente esgotado.
6. Tenho vindo a desinteressar-me pelos meus estudos desde que ingressei na universidade.
7. Sinto-me pouco entusiasmado com os meus estudos.
8. Sinto-me cada vez mais cínico relativamente à utilidade potencial dos meus estudos.
9. Tenho dúvidas sobre o significado dos meus estudos.

Statement
10. Consigo resolver, de forma eficaz, os problemas que resultam dos meus estudos.
11. Acredito que participo, de forma positiva, nas aulas a que assisto.
12. Sinto que sou um bom aluno.
13. Sinto-me estimulado quando atinjo os meus objetivos escolares.
14. Tenho aprendido muitas matérias interessantes durante o meu curso.
15. Durante a aula, sinto que consigo acompanhar as matérias de forma eficaz.

F

Big Three Perfectionism Scale for Self-critical perfectionism

For the BTPS only the self-critical dimension is considered. As such, the calculation of results is as simple as adding every answer's value. Here are all the questions in the English version of the questionnaire.

Please, read the following statements and indicate how much you agree with each one. Use the following values: strongly disagree, disagree, neither agree nor disagree, agree, strongly agree.

Table F.1: BTPS [39]

Statement
1. Everyone expects me to be perfect.
2. I judge myself harshly when I don't do something perfectly.
3. My value as a person depends on being perfect.
4. I tend to doubt whether I am doing something "right".
5. I feel disappointed with myself, when I don't do something perfectly.
6. Making even a small mistake would upset me.

G

Big Three Perfectionism Scale for Self-critical perfectionism - Portuguese version

Here is the Portuguese version of the BTPS questionnaire. The results are calculated as explained in appendix F.

Por favor, leia as seguintes afirmações e indique o quanto concorda com cada uma. Use a seguinte escala de avaliação: discordo muito, discordo, não concordo nem discordo, concordo, concordo muito.

Table G.1: BTPS - Portuguese version [43]

Statement
1. Toda a gente espera que eu seja perfeito.
2. Critico-me duramente quando faço alguma coisa que não esteja perfeita.
3. O meu valor como pessoa depende de ser perfeito.
4. Tenho tendência a duvidar se estou a fazer as coisas bem.
5. Fico desapontado comigo quando não faço as coisas de modo perfeito.
6. Cometer um pequeno erro, mesmo que pequeno, iria incomodar-me.

H

List of recommendations

In this chapter there is the list of all recommendations available at the moment in the RL model. They were developed by the IPM research group, based off of knowledge on how to help people with different mental health issues.

For now, they are only available in English, as this thesis work was developed in English, but they can be translated and implemented in Portuguese as well.

Table H.1: Recommendations available in the RL model

Recommendation
Practice mindfulness: it can help you become more aware of your thoughts and feelings and regulate them.
Set achievable goals: break down big tasks into smaller, more manageable goals. This will help you stay focused and motivated, and you'll be more likely to follow through.
Look for activities that you enjoy and where you feel comfortable. The more you enjoy an activity, the more likely you are to engage with it.
Set reasonable goals / realistic expectations: rather than striving for perfection, set realistic goals and expectations for yourself.
Exercise regularly: it is a great way to reduce anxiety and stress, and it can also help boost your mood and self-confidence.
Get enough sleep: lack of sleep can exacerbate feelings of anxiety and depression, so it's important to prioritize good sleep hygiene.

H. List of recommendations

Recommendation
Practice self-compassion: be kind to yourself and practice self-compassion. Recognize that everyone makes mistakes and experiences setbacks, and don't be too hard on yourself when things don't go as planned. What would you say to a good friend in a moment of suffering? say that to yourself.
Seek professional help: if you're struggling with high levels of anxiety or depression, consider seeking help from a mental health professional.
Connect with others: social support is important for mental health. Spend time with friends and family who are supportive.
Remember: anxiety makes us believe in worst case scenarios.
Remember: we can function and accomplish tasks even when we don't feel our best.
Remember: postponing or avoiding what causes us anxiety or discomfort can help in the short term, but it harms well-being and increases stress in the medium and short term.
Create a routine: establishing a routine can help you develop good habits and stay organized. Try to schedule your day and stick to a consistent routine as much as possible.
Use reminders: set reminders for important tasks or events to help you stay on track. Use a calendar or an app to help you remember deadlines and appointments.
Practice self-discipline: build self-discipline by setting limits on activities that may distract you from achieving your goals. For example, limit your time on social media or TV.
Seek feedback: ask for feedback from people you trust and respect. Their input can help you see blind spots and areas where you need to improve.
Reward yourself: celebrate your accomplishments and reward yourself for achieving your goals. This will help reinforce positive behaviors and keep you motivated
Look for activities that you enjoy and where you feel comfortable. The more you enjoy an activity, the more likely you are to engage with it

Recommendation
Join a group: joining a group or club with like-minded people can be a great way to make social connections and build relationships. Look for groups that align with your interests or hobbies.
Practice active listening: when you do engage with others, practice active listening. This means paying attention to what the other person is saying, asking questions, and showing empathy.
Build social skills: if you struggle with social skills, consider seeking help from a therapist who specializes in social skills training. They can help you develop strategies for initiating and maintaining conversations.
Practice self-care: make sure to take care of your physical and emotional needs. This includes getting enough sleep, exercise, and healthy food, as well as engaging in activities that promote relaxation and stress relief.
Set realistic goals: don't push yourself too hard to be more outgoing or social than you're comfortable with. Set small, achievable goals and build on them over time.
Remember: there's nothing inherently wrong with being introverted. It's just a different way of engaging with the world. Focus on finding activities and social situations that work for you, rather than trying to fit into someone else's idea of what it means to be sociable or outgoing.
While Perfectionism can help you achieve success in some areas, it can also lead to burnout, anxiety, and depression. Recognize the costs of perfectionism and the impact it has on your life.
Remember: perfection doesn't exist, so it's impossible to be perfect or do perfect things. Striving for perfection leads to constant dissatisfaction and frustration.
Embrace imperfection: accept that perfection is an unattainable goal and that imperfection is a natural part of the human experience. Embrace the imperfections in yourself and others.
Practice self-compassion: be kind to yourself when you make mistakes or fall short of your goals. Treat yourself with the same compassion and understanding that you would offer to a friend who was struggling. Self-compassion is the antidote to the poison that is perfectionism.

H. List of recommendations

Recommendation
Focus on progress, not on (perfect) results: Instead of focusing solely on the end result, focus on the progress you make along the way. Celebrate small wins and milestones as you work towards your goals.
Seek help if needed: if your perfectionism is causing significant distress or interfering with your daily life, consider seeking help.
Remember: breaking free from perfectionism takes time and effort, but it is possible. Be patient and kind to yourself as you work towards developing a healthier relationship with yourself and your goals.
Develop assertiveness: sometimes it is important to set limits and say no.
Develop creativity as a form of expression of identity.
Break the link between performance/achievements and personal value.
Distinguish between self-confidence and personal worth. Self-confidence has fluctuations. Our value as a person does not.

I

QLearning Python code

```
1
2 import numpy as np
3 import databaserecommendations as dr
4 from datetime import datetime
5
6 class QAgent():
7     def __init__(self, num_states, num_actions, learning_rate, gamma,
8                 max_epsilon, decay_rate, min_epsilon):
9         self.num_states = num_states
10        self.num_actions = num_actions
11        self.learning_rate = learning_rate
12        self.gamma = gamma
13        self.epsilon = max_epsilon
14        self.max_epsilon = max_epsilon
15        self.decay_rate = decay_rate
16        self.min_epsilon = min_epsilon
17        self.q_table = np.zeros((num_states, num_actions))
18        self.training = False
19        print("Agent created with {} states and {} actions.".format
20              (self.num_states, self.num_actions))
21
22 #this one to start the training, because we only want to start if
23 #all questionnaires have answers
24 def initial_trigger(self):
25     current_state = dr.get_current_state()
26     if (bool(current_state)):
27         global train
28         train = self.train()
29         next(train)
30     else:
31         print("Can't start learning agent until all forms are
32               fulfilled")
33
34 def trigger_next_step(self):
35     if self.training:
```



```
32     next(train)
33
34 def train(self):
35     step = 0
36     self.training = True
37     #calculate initial state
38     state = dr.get_current_state() #last state in the table
39     while (True):
40         #choose an action
41         action = self.epsilon_greedy_policy(state)
42         recommendation = dr.get_recommendation(action)
43         print("RECOMMENDATION: {}".format(recommendation[0][0]))
44         yield action
45         #now apply the action and wait for a new state and for the
reward
46         new_state = dr.get_current_state()
47         reward = self.calculate_reward()
48         print("REWARD: {}".format(reward))
49         self.q_table[state][action] = self.q_table[state][action] +
self.learning_rate * (reward + self.gamma * np.max(self.q_table[
new_state])) - self.q_table[state][action])
50         print("New q-value: {}".format(self.q_table[state][action]))
51         self.epsilon = self.min_epsilon + (self.max_epsilon - self.
min_epsilon)*np.exp(-self.decay_rate*step)
52         step += 1
53         #loop so next time the state is the new state
54         state = new_state
55
56 #epsilon greedy policy to choose next action
57 def epsilon_greedy_policy(self, state):
58     random_int = np.random.uniform(0,1)
59     #if the random is smaller than epsilon, then random action
60     if random_int < self.epsilon:
61         action = np.random.randint(0, self.num_actions)
62     #else select the most promising action so far
63     else:
64         action = np.argmax(self.q_table[state])
65     return action
66
67 def calculate_reward(self):
68     date_format = '%Y-%m-%d'
69     #we want to calculate a reward based on how much the user
improves/got worse since last time
70     #for that, I would say using the points from the questionnaire,
and give an extra big reward if it comes to a positive value,
```

```

or extra negative if it goes to worse
71 #making it work for emotional and burnout only, since
personality and perfectionism should only be fulfilled once
72 answers = dr.get_emotions()
73 first_day = datetime.strptime(answers[-1][-1], date_format) #
last time recommendation was given
74 last_day = datetime.strptime(answers[0][-1], date_format) #most
recent questionnaire
75 count = 0
76 while ((last_day-first_day).days > 7):
77     count += 1
78     last_day = datetime.strptime(answers[count][-1], date_format)
79     if (count==3):
80         print("The questionnaire was not filled in the 7 days
following the last recommendation. Not enough information")
81         break
82     if (last_day!=first_day and count!=3):
83         #depression should be as low as possible, so we should
consider positive reward if it decreases
84         depression_change = - (answers[count][0] - answers[-1][0])
85         #if the user passes the threshold from a 'negative' to a '
positive' state, we want a specially big reward
86         #if it is the other way around, a specially negative one (
since we already calculate the signal, we can just multiply
anyway)
87         if ((answers[count][0] < 6.85918 and answers[-1][0] >
6.85918) or (answers[count][0] > 6.85918 and answers[-1][0] <
6.85918)):
88             depression_change = 3*depression_change
89             #the same applies for anxiety
90             anxiety_change = - (answers[count][1] - answers[-1][1])
91             if ((answers[count][1] < 8.83543 and answers[-1][1] >
8.83543) or (answers[count][1] > 8.83543 and answers[-1][1] <
8.83543)):
92                 anxiety_change = 3*anxiety_change
93                 positive_affect_change = answers[count][2] - answers[-1][2] #
positive affect has the opposite logic, so it should be high,
meaning we want a positive reward if it increases
94                 if ((answers[count][2] > 2.7 and answers[-1][2] < 2.7) or (
answers[count][2] < 2.7 and answers[-1][2] > 2.7)):
95                     positive_affect_change = 3*positive_affect_change
96                     answers_burnout = dr.get_burnout()
97                     burnout_date = datetime.strptime(answers_burnout[0][-1],
date_format)
98                     burnout_mean = 0

```

```
99     #if the burnout questionnaire has been answered since the
100     last time we answered the emotional
101     if (burnout_date >= first_day):
102         cynicism_change = - (answers_burnout[0][0] -
103         answers_burnout[-1][0])
104         if ((answers_burnout[0][0] < 13.78297 and answers_burnout
105         [-1][0] > 13.78297) or (answers_burnout[0][0] > 13.78297 and
106         answers_burnout[-1][0] < 13.78297)):
107             cynicism_change = 3*cynicism_change
108             efficacy_change = answers_burnout[0][1] - answers_burnout
109             [-1][1]
110             if ((answers_burnout[0][1] > 12.17105 and answers_burnout
111             [-1][1] < 12.17105) or (answers_burnout[0][1] < 12.17105 and
112             answers_burnout[-1][1] > 12.17105)):
113                 efficacy_change = 3*efficacy_change
114                 exhaustion_change = - (answers_burnout[0][2] -
115                 answers_burnout[-1][2])
116                 if ((answers_burnout[0][2] < 21.74388 and answers_burnout
117                 [-1][2] > 21.74388) or (answers_burnout[0][2] > 21.74388 and
118                 answers_burnout[-1][2] < 21.74388)):
119                     exhaustion_change = 3*exhaustion_change
120                     burnout_mean = (cynicism_change + efficacy_change +
121                     exhaustion_change)/3
122                     #the reward will be the sum of the means for each
123                     questionnaire
124                     reward = (depression_change + anxiety_change +
125                     positive_affect_change)/3 + burnout_mean
126                     return reward
127             else:
128                 print("Not enough information")
129                 return 0
```

Code snippet I.1: Q-learning code in Python

J

Resulting article

From the work developed in scope of this thesis, resulted an article presented in this appendix.

It will be submitted for publication to the "Computers & Education" journal.

An innovative solution based on chatbots and RL mechanisms to improve academic performance and student well-being

Filipa Moreira^a, Ana Telma Pereira^{b,c}, André Rodrigues^d, António Macedo^{b,c}, Fernando Boavida^a, Jorge Sá Silva^{a,e}

^a*Faculty of Sciences and Technology, University of Coimbra, Portugal*

^b*Institute of Psychological Medicine, Faculty of Medicine, University of Coimbra, Portugal*

^c*Coimbra Institute for Biomedical Imaging and Translational Research, Portugal*

^d*Coimbra Polytechnic Institute, Portugal*

^e*INESC Coimbra, Portugal*

Abstract

Academic success is directly correlated with a student's motivation. Mental health problems, often induced by excessive stress, can affect motivation and, consequently, academic performance.

A new and innovative way to help improve students' academic performance and well-being is proposed. By using machine learning, particularly reinforcement learning, and chatbots, a model was developed to give students' feedback based on their psychometric parameters, measured by scientifically accepted questionnaires.

The main goal is to explore new possibilities on how to help students in an efficient and convenient way, by combining psychological medicine with informatics, namely application programming, by trying to build a Human-in-the-loop system combining user inputs with data collected from sensors.

Keywords: Learning tools; Artificial Intelligence; Motivation; Internet of Things

1. Introduction

Throughout the years, awareness about mental health has been increasing, as its importance and impact on overall health is continuously more recognized. (World Health Organization (WHO), 2019)

Motivation is highly correlated to mental health, and motivated students have a higher academic performance. (Mahdavi, Valibeygi, Moradi & Sadeghi, 2023) College students have high tendencies to mental health problems, mostly due to the high stress they are subjected to and the difficulty in keeping a good sleep routine, being that these factors are interconnected. (Gardani, Bradford, Russell, Allan, Beattie, Ellis, et al., 2022)

In Portugal, as of 2023, the percentage of students that drop out of college within the first year is around 10%. (*Dados e Estatísticas de Cursos Superiores*, n.d.) A lot of factors can influence students to drop out, and some of the most relevant ones include anxiety, depression and lack of motivation. (Emily, 2023)

This article presents a new approach, based on Reinforcement Learning algorithms, to help students with their well-being, and by doing so, helping with their academic performance.

To do so, a first model of a Reinforcement Learning is proposed, based on results collected from psychometric questionnaires that measure different psychological dimensions. It is a proposal to unify the areas of psychological medicine and technology, in favor of mental health and well-being improvement.

This article is composed by six sections. The first one has a brief introduction of the main goal and rationale. The second presents some information about how universities are becoming more connected with their students by using technology. The third one describes the proposed solution and what was developed towards the main goal. The fourth chapter shows some evaluation results of the built system. The fifth chapter describes some real platforms where the solution can be hosted to be used by the students. The sixth and last chapter advances some conclusions about the work developed.

2. University of the future

Universities have considerably evolved in the past years. Nowadays, their roles are essential for any society, not only in pedagogical terms - for preparing our people for the job market - but also in terms of their relationship with companies, other members of their own society and other cultures, with the proposal of new development models based on knowledge and innovation. This evolution is associated with the Bologna process, and with the change in the educational paradigm that it advocates, “from a passive model, based on the acquisition of knowledge, to a model based on the development of competences”. (European Commission, 2020)

However, despite this continuous development and this social contribution of the University, very little has changed in the “classroom” throughout History. We continue to use magisterial classes where the teacher “outpours” the theoretical material to dozens, and sometimes hundreds, of students without the necessary proximity in a real learning process. This “one-size-fits-all” paradigm hampers the achievement of the full potential of each individual student and may even generate psychological distress.

In the context of the COVID-19 pandemic, the “classroom” was forced to rethink itself. In the period of confinement, distance learning had to be resorted to. Tools such as Zoom, Teams, Skype, Webex, among others, were of great importance in responding quickly to this contingency and supporting these distance classes during confinement. However, the University can never become a simple collection of online courses. The University is all the wealth of experiences, emotions, multiculturalism that can never be replaced by just some remote processes.

In terms of recent History, there was a very important stage with the democratization/massification of Higher Education teaching. Today the challenge is different: taking advantage of the Internet, AI, Internet of Things (IoT) and Intelligent Personal Assistant Agents (IPAA) such as chatbots (eg Siri, Alexa, ChatGPT, Bard), we should propose customization techniques and teaching by profiles, so that the professor can meet the physical-psychological characteristics of each student and their interests, even in hybrid “classrooms” with hundreds of students.

Previous work at Dartmouth College has presented a system named StudentLife, (Wang, F. Chen, Z. Chen, Li, Harari, Tignor, et al., n.d.) which uses Android phones to continuously sense the day-to-day impact of workload on stress, sleep, activity, mood, sociability, psychological well-being and academic performance of a single class of 48 students across a 10-week term. StudentLife identified a strong correlation between the automatic sensing data and several psychological well-being measures, specifically, PHQ-9 depression, perceived stress, flourishing and loneliness subscales. Their results indicated that automatically sensed conversation, activity, mobility, and sleep have significant correlations with psychological well-being outcomes and academic performance.

While StudentLife provided a great example on how smartphones can be very helpful in acquiring a student’s context, the scope of the system was limited. The objective of the study was only to obtain a statistical dataset, containing the behavioral and psychological health outcomes of the students. The system did not take any measures to counteract the student’s negative academic performance or undesirable lifestyles. Another relevant and similar contribution is the work presented in Eskes, Spruit, Brinkkemper, Vorstman & Kas, 2016; Harari, Gosling, Wang, F. Chen, Z. Chen & Campbell, 2017; Tibshirani, 1996; Wang et al., 2014; Wang, Harari, Hao, Zhou & Campbell, 2015.

Our work intends to extend StudentLife’s ideas by taking a proactive approach to “close the loop”. In other words, we intend to create a system that not only automatically infers students’ status and attitudes during classes, but also closes the loop - feeding this information to a decision-making process.

3. Proposed solution

The solution proposed in this article is based on scientifically recognized psychology questionnaires that validly measure different dimensions, chatbots and methods of Reinforcement Learning (RL). In the next sections, we aim to present more details about the questionnaires, a first approach to the problem of using the collected data to give back recommendations to the user, and finally about the RL method designed and developed.

3.1. Implemented questionnaires

3.1.1. HEXACO Personality Inventory

The HEXACO Personality Inventory - Revised is composed of sixty items to be answered with a five points scale according to the student's level of agreement from "strongly disagree" (1) to "strongly agree" (5). (Ashton & Lee, 2009) This questionnaire is only answered once since personality traits are considered to be stable during a considerable amount of time. Even though people mature and their personality evolves over time, there is evidence of a rank-order stability, meaning that individuals tend to score around the same values, that stabilize around young adulthood. (Bleidorn, Schwaba, Zheng, Hopwood, Sosa, Roberts, et al., 2022; Damian, Spengler, Sutu, & Roberts, 2019)

The dimensions measured by HEXACO can be found at table 1.

Table 1. HEXACO's domains and facets [16]

Domains	Facets
Honesty/Humility	Sincerity
	Fairness
	Greed avoidance
	Modesty
Emotionality	Fearfulness
	Anxiety
	Dependence
	Sentimentality
Extraversion	Social self-esteem
	Social boldness
	Sociability
	Liveliness
Agreeableness (vs Anger)	Forgiveness
	Gentleness
	Flexibility
	Patience
Conscientiousness	Organization
	Diligence
	Perfectionism
	Prudence
Openness to experience	Aesthetic appreciation
	Inquisitiveness
	Creativity
	Unconventionality

From these six dimensions, only three were considered for the approaches presented in this article due to their stronger correlation with mental states and psychological distress: neuroticism (which in the HEXACO questionnaire is mainly represented by emotionality), conscientiousness and extraversion. (Kang, Steffens, Pineda, Widuch & Malvaso, 2023; Kotov, Gamez, Schmidt & Watson, 2010; Lewis & Cardwell, 2020)

Emotionality is positively correlated with stress and anxiety, which means that an individual with a high emotionality will be more prone to stressful situations and to negative emotional reactions. Among the main personality traits, it has the strongest association to stress, which is consistent with the fact that some aspects of their concepts overlap. (Luo, Zhang, Cao & Roberts, 2023)

Conscientiousness and extraversion are negatively correlated to psychological distress and symptoms of mental disorders. (Kang et al., 2023) Individuals with higher conscientiousness and extraversion have higher regulation abilities, are more prone to be reward focused, organized and plan ahead, tending to experience more positive emotions. This contributes to have better coping strategies in dealing with stressful situations and, thus, to present lower levels perceived stress. (Luo et al., 2030; Goodwin & Friedman, 2006) Conscientiousness has also been studied as a factor that can influence general (mental) health throughout life. (Kern & Friedman, 2008)

3.1.2. PoMS-12

PoMS (Profile of Mood states) is a questionnaire that, in its original version, (McNair, Lorr & Droppleman, 1971) includes 65 adjectives describing mood states and, for each one, the individual can choose how he feels regarding that state in five levels, where 0 means "Not at all" and 4 means "Extremely". These states relate to six dimensions: tension, depression, anger, vigor, fatigue, and confusion. (McNair et al., 1971)

Given the impact that positive and negative emotions have on students' well-being, motivation and academic performance, it is relevant to consider affective states.

For this work, however, a shorter version of the questionnaire was considered. It contains 12 adjectives that describe feelings and evaluate three mood states: depression, anxiety and positive affect.

While still being valid and able to give reliable results, the time spent filling this abbreviated version of PoMs is much shorter, which favors adherence to filling it regularly. (Pereira, Araújo, Cabaços, Brito, M, Fernandes, Rodrigues, et al., 2023) Table 2 shows the mood states present in the questionnaire, as well as the affect dimensions they relate to.

Table 2. Moods per dimension in PoMS-12 questionnaire. (Pereira, Araújo et al., 2023; Shacham, 1983)

Mood dimensions	Mood states
Depression	Sad
	Pessimistic
	Desperate
	Useless
Anxiety	Irritated
	Grumpy
	In a bad mood
Positive affect	Anxious
	Excited
	Active
	Cheerful
	Self-confident

This questionnaire may be answered with reference to different timeframes; some of the most commonly used are:

- weekly - "How have you felt during the past week including today?"
- daily - "How do you feel right now?"

Referencing the past week, is considered a long enough period to capture people's typical and persistent emotional reactions to daily life events, but still short enough to assess reactive fluctuations and acute effects of an intervention. (McNair, Lorr & Droppleman, 1992)

It is suggested by several authors that the "right now" response time frame should be the method of choice to evaluate the mood state in a valid and ecological way, since recall of mood can be influenced by mood at the time of recall and possible significant events. (Beedie, Terry & Lane, 2005; Bolger, Davis & Rafaeli, 2003; Cranford, Shrout, Iida, Rafaeli, Yip & Bolger, 2006) Taking that into consideration, this questionnaire is presented with a 'right now' scope, and to contribute to the whole system it should be answered every day.

3.1.3. *Big Three Perfectionism Scale (BTPS)*

The BTPS is based on a recent measurement model used to evaluate perfectionism in three major factors: rigid perfectionism, self-critical perfectionism and narcissistic perfectionism. Originally, it was composed by 45 items, (Smith, Saklofske, Stoeber & Sherry, 2016) but a shorter version with only 16 items was developed. (Feher, Smith., Saklofske, Plouffe, Wilson & Sherry, 2020) The answers are given in a five points scale, where 1 means "strongly disagree" and 5 means "strongly agree".

Self-critical perfectionism is the dimension most associated with psychological distress (anxiety, depression, stress), burnout, difficulty in coping and academic struggles, making student more prone to perceived failure and self-criticism. (Cabaços, Macedo, Carneiro, Brito, Amaral, Araújo, et al., 2023; Dunkley, Zuroff & Blankstein, 2003; Pereira, Brito, Cabaços, Carneiro, Carvalho, Manão, et al., 2022) Because of that, only this dimension was considered and only the items related with it are used.

3.1.4. *Maslach Burnout Inventory (MBI)*

While the MBI was originally developed to assess burnout levels in human services workers, (Maslach, Jackson & Leiter, 1997) it also affects workers in other fields and even students. Students perform activities really similar to work, as they have to follow schedules and timelines to develop assignments that are submitted to a performance evaluation. Burnout has a negative impact on academic performance. (Madigan & Curran, 2021)

Maslach Burnout Inventory - Student Survey (MBI-SS) is an adapted version of the MBI to be used by students. It has 15 items divided into 3 scales: emotional exhaustion, cynicism and academic efficacy. Each item is a statement to be scored on a 7-point system, where 0 means "Never" and 6 means "Always". While elevated scores in exhaustion and cynicism indicate higher degrees of burnout, the academic efficacy is reversed, meaning that lower scores in this subscale indicate higher degrees of burnout. (Schaufeli, Martinez, Pinto, Salanova, & Bakker, 2002)

3.2. *Chatbot and first approach*

As previously mentioned, before starting on the RL method, there was a first approach to using the collected data to give back recommendations to the user. Said first approach was developed using the capabilities of DialogFlow directly.

The main idea was to create DialogFlow intents that would be triggered when the results for any of the dimensions considered were concerning. To determine when the results could be considered concerning, known mean and standard deviation (SD) values for each dimensions were used.

For this process to work, each time a user answers a questionnaire, the results are calculated and then compared to the mean value. When the condition to be considered concerning was met by the calculated result, a specific intent was sent to DialogFlow.

HEXACO selected personality traits and perfectionism questionnaires measure facets that are related with psychological distress. When there is a direct correlation, the results should be considered as concerning when their value goes above mean+SD. On the other hand, if the dimension has a negative correlation with them, then the results should be considered concerning when their value goes below mean-SD.

Burnout and the PoMS questionnaire, measure states more directly, instead of predispose to them. For those, burnout, depression and anxiety should be considered concerning above mean+SD, while positive affect is concerning when it is below mean-SD.

It is important to mention that for the MBI there are cut-off points, that were defined in the third edition of the MBI Manual, (Maslach et al., 1997) but eventually it was noticed that they did not align with some average values obtained in different populations. So, the values considered were the mean and SD from a study conducted with Portuguese students. (Pereira et al., 2022)

The mean and SD values for each dimension that was taken into account can be found at table 3. The dimensions considered were selected based on known correlations with students' motivation and performance, as described in subsection 3.1.

Table 3. Means and SD of selected HEXACO personality traits, (Sousa, 2018) perfectionism, (Pereira, Brito, Marques, Araujo, Cabaços, Carneiro, et al., 2023), burnout (Manão, Carneiro, Carvalho, et al., 2021) and PoMS questionnaires. (Pereira, Araújo et al., 2023)

Dimensions	Mean	SD
Personality		
Emotionality	34.06	5.74
Extraversion	33.81	5.94
Conscientiousness	37.36	5.17
Perfectionism		
Self-critical	20.04	5.43
Burnout		
Efficacy	18.1803	6.00925
Exhaustion	15.1671	6.57678
Cynicism	7.6158	6.16717
PoMS		
Depression	4.7761	2.08308
Anxiety	5.7085	3.12693
Positive affect	5.2322	2.53220

As an example, here is the explanation on how the process works when a user had results above mean+SD for the emotional exhaustion (EE) dimension in MBI.

The results for the EE dimension are calculated, and it is detected that they are above the established threshold. This sends a very specific intent to DialogFlow, with an unlikely to be matched training phrase, so it is only matched by this automatic process. A message is sent back with a warning that a high value of EE was detected. An output context is activated so a user can interact later by asking questions through the chatbot, and they can be correctly matched to other intents in DialogFlow. In figure 1 it is possible to see how this was configured in DialogFlow.

DialogFlow is prepared to then answer questions in the sense of better explaining what this means or give recommendations, depending on the user interaction. New intents are matched, based on what the user says but also on the context that was set by the first intent.

When the intent with recommendations is matched, some are chosen randomly from the set defined in the responses field. Below are some images that show possible questions and the defined answers.

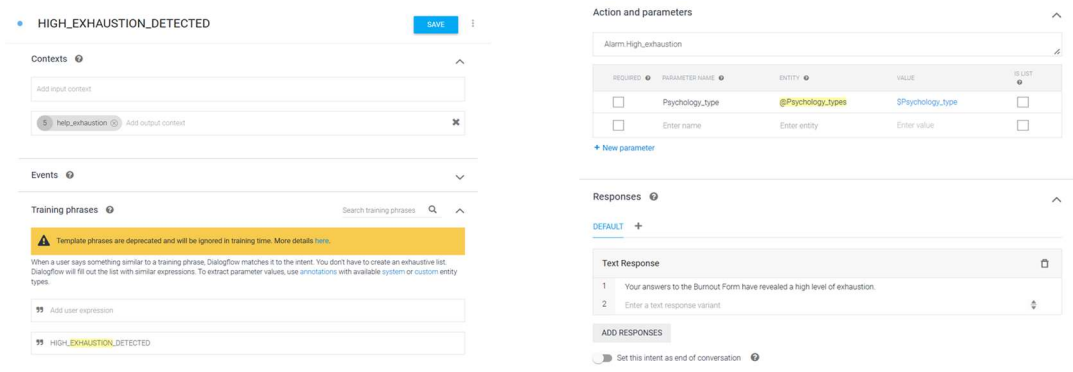


Figure 2 - Intent for high values of exhaustion in DialogFlow

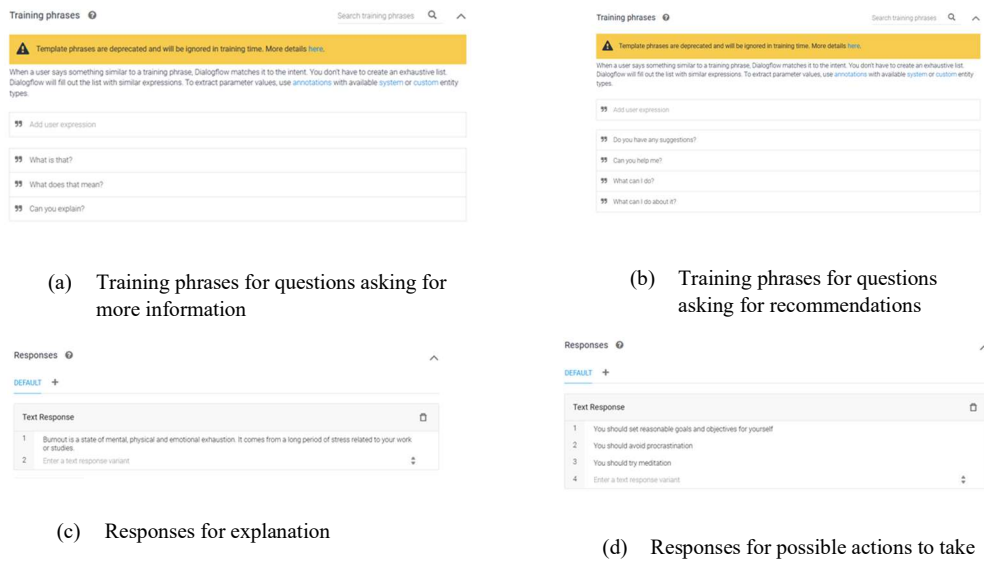


Figure 1 - Possible questions and answers after first intent

It is important to note that, by doing this, the answer given back each time would only be dependent on a randomization done by DialogFlow when a set of possible answers is given.

3.3. Personalized approach based on RL mechanisms

The main problem with using DialogFlow functionalities directly is that it only gave random recommendations from a preselected set, which makes it too general and ignores the individuality of each person.

As a way to improve interaction with the user and use the information collected to its advantage, a RL model was thought out to gather the information that comes from both the questionnaires and from smartphone sensors in a

meaningful way and transform it into personalized recommendations tailored to each individual's needs, instead of the random selection of recommendations done by DialogFlow.

The smartphone sensors can be used for various purposes that relate to the student's mental health such as, per instance: measuring the level of physical activity, to understand how active a student is; tracking if a student spends too much time at home or too little time at the university, to check if there is a high isolation and low engagement with lessons and activities; check ambient noise to evaluate study quality, sleep time, among others; checking the battery and how long the screen is locked to evaluate phone usage; proximity sensors can help determine if the student spends a lot of time around other people, to check for social activity. All of these can be relevant in understanding a student's mental state.

In this first development that will be described here, to simplify the process, only the questionnaires were considered. Sensors data can later be included in future versions.

Before implementation, it is important to understand the main concepts and define how they apply to our specific situation so a proper design can be made. We need to define state, actions and reward to be able to develop a RL model.

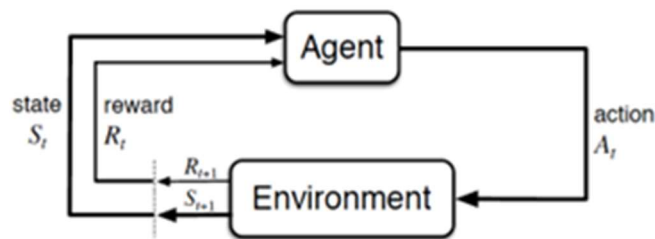


Figure 3 - General diagram of a RL model (Sutton & Barto, 2018)

The state here is defined by the user's answers to the questionnaires. The actions are the possible recommendations the user can get. The reward will relate to the evolution of the values that define the state, i.e., the questionnaires' results for the considered dimensions. When the results evolve towards less concerning values, then the agent receives a positive reward to learn that was a good recommendation for a user. If the values get more concerning, the opposite is true.

The following diagram (figure 4) shows the basic composition of the intended RL model. The red arrows represent constant values, since both the personality and perfectionism questionnaires are only answered once. As mentioned, personality can be considered relatively constant during some time, and since perfectionism is a personality trait, it shares the same property. Since they can still influence the state, they need to be included in the algorithm.

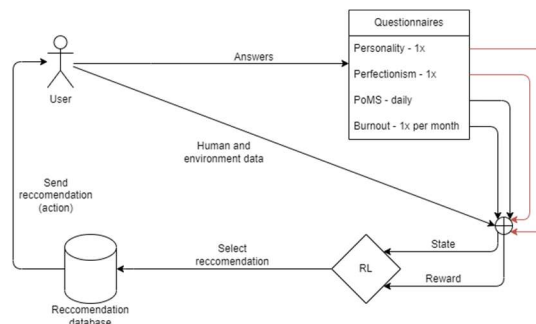


Figure 4 - Reinforcement Learning method diagram

For a threshold, the mean and SD of each questionnaire's dimension was considered. The values are the same as the ones considered for the first approach, available in table 3, and the way to determine when the values are concerning is also the same as described in the corresponding subsection.

By considering these values and comparing the obtained results against them, in the first approach a trigger was generated for a DialogFlow intent, and in this improved approach it is possible to define states.

Now that we have the basic concepts defined, an algorithm needs to be selected. There are numerous algorithms that can be used to try to achieve the best policy in a RL model. To develop the model's first version, the intention is to select a simple algorithm to try to prove the concept. However, some of the simpler ones need to have a perfect model of the environment, which is not possible in this situation, or only give back proper results in episodic tasks. (Sutton & Barto, 2018)

For the case discussed here, we need a model that can learn from experience, only knowing the state, actions and considering the rewards, and also that can be updated and learn after each interaction so it learns continuously throughout the process. One algorithm that aligns with these conditions and is commonly used in a lot of RL scenarios is the Q-learning. The Q-learning algorithm will directly approximate to the optimal value for the action-value function, no matter what the policy is, which makes it easier to analyze, and will always assume the maximum reward in the next action taken for the Q-value calculation. (Sutton & Barto, 2018)

For the purpose of this work, which is to build a first version of a RL algorithm that can open new possibilities to help students with academic performance and motivation, Q-learning was used since it ends up being simple and straightforward, while having the needed characteristics.

Explaining the algorithm a little bit more in-depth, it is defined by: (Watkins & Dayan, 1992)

$$Q(S_t, A_t) \rightarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (1)$$

S_t and A_t represent, respectively, the state and the action at a given time t . α is the learning rate, R_{t+1} is the reward, γ is the discount factor and $\max_a Q(S_{t+1}, a)$ represents the maximum Q-value for the action taken. (Zhou, 2022)

Equation 1 will update the Q-values, and will be included in an algorithm with the following steps: (Sutton & Barto, 2018)

Parameters: define parameters' values

Initialize: Q-table randomly, with the exception that $Q(\text{terminal}, \cdot) = 0$

for each episode do

 Initialize state

for each step do

 Choose action using policy (e.g ϵ -greedy)

 Take action and observe the next state and the reward obtained

 Apply equation 1

 Replace state with the next state

end

 Until state is terminal

end

An episode represents an independent event, such as a play of a game, or any sort of sequence of interactions that ends in a terminal state. A step is each interaction inside the episode. (Sutton & Barto, 2018)

For the specific purpose of this work, the goal is to have the agent permanently learning from the user interactions. So, for the purpose of developing an algorithm that is always interacting with the user and giving feedback, episodes were not considered, and the agent is always updating Q-values and doing the iterative process for each step.

3.3.1. First implementation using Python

As a first step in implementing the RL algorithm, a code was developed in Python. Python is widely used for lots of algorithms, including for RL, due to the wide range of libraries it provides.

In order to get a simple interface to interact with the code, a Graphical User Interface (GUI) was created using the *Tkinter* library available in Python.

As previously mentioned, instead of considering separate episodes, the agent will update the Q-values at each iteration and keep learning continuously.

Starting off with the parameters defined. The learning rate α was defined as 0.7. This value will determine how fast the agent learns. The closer it is to 1, the faster the learning will be, but, if the value is too big, it might make the results less accurate. (Zhou, 2022) By using 0.7, the intention is to still have a good accuracy, but a relatively fast learning. Then for the discount factor γ , 0.95 was considered. The discount factor will determine how important are the short-term rewards. The closer it gets to 1, the more the agent will be concerned about the future and delay the rewards. (Zhou, 2022)

It is also necessary to define a strategy to select which actions to take. A general strategy is the ϵ -greedy. This strategy tries to balance exploration and exploitation, by defining an ϵ value that will determine how much of the action selection is random (exploration) and how much of it is by the maximum Q-value available (exploitation). The value of ϵ should be between 0 and 1 will define the probability of choosing a random action, leaving the choice of the maximum reward with a probability of $1-\epsilon$. (Sutton & Barto, 2018)

As times goes by, and the agent learns more about the environment, the choices should be more based on knowledge and less random. With this in mind, the ϵ value was defined as an exponential decay that starts at 0.95 and can go as low as 0.05, with a decay rate of 0.005. The evolution of the ϵ values through the algorithm steps is presented in figure 1.

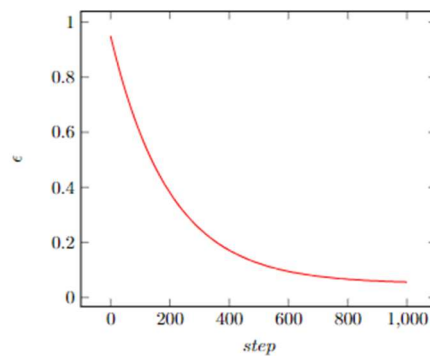


Figure 5 - Exponential decay of ϵ values through the algorithm steps

In this algorithm, the actions will be mental health recommendations. As of this time, there are 35 recommendations, meaning there is a total of 35 actions.

As for the states, there are some things to take into consideration. For each of the questionnaires, there are various results. As an example, the HEXACO-60 has 10 questions per dimension, and each of them can be answered with integer values from 1 to 5. Considering one of the dimensions, such as neuroticism, the value of the results could go from 10 to 50, which totals 41 possible results for this dimension alone. Considering this for each of the 10 dimensions across the different questionnaires, some of them with even more possibilities, and combining them into overall states, there would be numerous possibilities. Taking into consideration the chosen algorithm and how it works, and also the fact that this is a proof of concept (PoC) of the utility of this kind of approach, a simplification was made to reduce the number of states and make the algorithm simpler and clearer.

This simplification takes into account the threshold values presented in table 3 and, for each dimension, defines two options, which are being above or below the threshold. Considering 2 options for each of the 10 dimensions, there is a total of 1024 possible combinations, so there are 1024 states. Each dimension will have a 0 if the value is considered concerning, and a 1 if it is not. Table 4 shows how the values will be for each dimension.

Table 4. Binary values set for each dimension.

Dimensions	Below threshold	Above threshold
Emotionality	1	0
Extraversion	0	1
Conscientiousness	0	1
Self-critical perfectionism	1	0
Efficacy	0	1
Exhaustion	1	0
Cynicism	1	0
Depression	1	0
Anxiety	1	0
Positive affect	0	1

With this approach, there are 35 actions and 1024 states, which means that the Q-table will be 1024x35. This table will contain the Q-values for each state-action pair, and will be continuously updated. To start, it is initialized to all zeros, meaning that the first recommendations will be random, as the system knows nothing about a user. With some time, it will start to give more personalized recommendations. The ϵ value also plays a part into this, by balancing exploration and exploitation as previously explained.

Having all of these definitions, there are enough bases to start developing the code.

The main function developed is *train()*, in which the general logic of the algorithm is defined. All the other functions were built around this.

The general idea is that there is a main loop that is always running. In this main loop, an action is selected based on an ϵ -greedy policy, with the previously explained ϵ values. Then, there is a *yield* that will suspend the execution until it is called again. (Python documentation, n.d.) This is needed because, in order to calculate the Q-values, there needs to be a next state, and that next state will be user-dependent, since it is a combination of the questionnaire results.

Once there is a new state, the execution proceeds from the *yield* point, and will get the new state, calculate the reward, update the Q-value, and update the ϵ value. The loop will then get back to the start and stop again at *yield*.

To do all these operations, the *train()* function relies on various other functions and a database structure that withholds all the needed data for the process.

The database is composed of several tables, presented in figure 6.

recommendations			
id	Int	PK	AU
recommendation	Text		

personality			
id	Int	PK	AU
emotionality	Int		
extraversion	Int		
conscientiousne.	Int		
date	Date		

burnout			
id	Int	PK	AU
cynicism	Int		
efficacy	Int		
exhaustion	Int		
date	Date		

emotions			
id	Int	PK	AU
depression	Int		
anxiety	Int		
positive affect	Int		

perfectionism			
id	Int	PK	AU
perfectionism	Int		
date	Date		

states			
id	Int	PK	AU
emotionality	Bool		
extraversion	Bool		
conscientiousne.	Bool		
perfectionism	Bool		
depression	Bool		
anxiety	Bool		
positive_affect	Bool		
cynicism	Bool		
efficacy	Bool		
exhaustion	Bool		

user_state			
id	Int	PK	AU
emotionality	Int		
extraversion	Int		
conscientiousne.	Int		
perfectionism	Int		
depression	Int		
anxiety	Int		
positive_affect	Int		
cynicism	Int		
efficacy	Int		
exhaustion	Int		

Figure 6 - Tables of database for the RL algorithm

There is one table per questionnaire, to save the result values. Then there is a table that saves all the recommendations and two tables that save states. While *user_state* saves the states of a user, composed by the questionnaire results turned into boolean values as per table 4, the *states* table contains all possible 1024 states. This way, it is possible to match a user's state by comparing it to the possibilities in *states* table, and refer to each state by a single id.

To facilitate access and insertion of some database values, there are some queries defined inside functions that can then be used in the algorithm.

Describing some of the functionality around the main cycle, there are 2 functions that work as a trigger, *initial_trigger()* and *trigger_next_step*, being the first to initialize the while loop once it verifies if all questionnaires were answered at least once, and the second to trigger the next run by continuing after the *yield*. By calling these functions when a questionnaire is answered, it is possible to interact with the loop from outside.

Then, there is *epsilon_greedy_policy(state)* that will apply the policy by getting a random number between 0 and 1. If this number is below the defined *epsilon* value at that moment, then a random action is selected. Otherwise, the action with the highest Q-value is selected.

Lastly, there is the *calculate_reward()* function where the reward is defined for a certain state-action pair. This is an essential part of the algorithm as it defines how the agent will be rewarded/punished, and that will have an impact on how the agent will learn the best actions to choose. There are some things to consider:

- Only the PoMS and MBI will be taken into consideration for the reward calculation, since they may vary considerably through time.
- PoMS measures emotional states, which can be influenced by a lot of factors daily. To try to compensate for daily fluctuations due to random life events, it was defined a condition that the reward will only be calculated, and consequently a new Q-value, after the user answers this questionnaire 4 times.
- Considering the previous decision, there was another factor to take into account. If a user does not answer regularly, then it is not possible to evaluate the evolution of the results. To avoid considering answers with a long time in between, a threshold of one week was set. From the moment the user gets the recommendation, only questionnaire answers within a week will be taken into account. If there are no answers within a week, it is considered that there is not enough information.
- Since MBI is filled in less times, it is only considered for reward if it has been answered after the last recommendation was given. Otherwise, it is a constant, so there is no need to include it as it will give a zero difference.
- When the questionnaire's results go from one side of the questionnaire's threshold values to the other, the reward/punishment should be bigger.

With all the previous information, the function was developed to calculate the reward based on the difference between the questionnaire's results for each dimension since the last recommendation was given and make a mean. The reward will be positive if the values move towards better values, and negative otherwise.

Once all this was functional, the *Tkinter* app was defined as a GUI for easy interaction. It has a simple layout, with four buttons, one to access each questionnaire. Instead of having the full questionnaires, the *Tkinter* app only has an input field for each dimension, where it is possible to directly write a result value, for testing purposes. In figure 7 it is possible to see some screens of this GUI.



Figure 7 - Screens of the GUI developed in *Tkinter*

3.3.2. Integrate using Flask server

Once the first version in Python was finalized, and it seemed to provide the desired functionality, the next step was to make this algorithm able to be integrated into an application.

Since the first implementation was in Python, in order to allow for reuse of some of the code, *Flask* was used. *Flask* is a Python module that allows for creation of HTTP methods. This can be used for web app development, or in this case, to develop a server that can be called from the mobile app and implement the developed RL algorithm. (Flask, n.d.) So, in the *Flask* server, there are only methods related to the algorithm itself, that can then be called and orchestrated inside an application.

As a way to validate the data that is sent to the server, it is possible to use the *jsonschema* library, to implement a JSON Schema validation in Python. (Jsonschema, n.d.) To make sure all the mandatory data is sent to a method, it is possible to define the schema with required fields, as well as their types and some restrictions they might have, such as minimum values.

Another change that needed to be done was figure out a way to remove the *while(True)* loop, since by using *Flask* and deploying the code in a server the idea is that the requests are received, processed, and an answer is sent back. Having an infinite loop in a server may cause issues, especially since it is accessed by several users at different times.

As a way to break this loop, what was previously the *train()* function, was split into two by the *yield* expression. The part before and after the *yield* became separate functions, one to send a recommendation and the other to update the Q-values.

Then different methods were created to interact with different functions. In total there are three methods, the initialization one, defined as a POST for */initialize* endpoint, the one to get recommendations, defined as a GET for */recommendation* endpoint, and the one to update the Q-values, defined a PATCH to */qtable* endpoint.

To make the whole process work, the initialization needs to be called when all four questionnaires have been answered at least once. Then, a first recommendation can be sent.

From then on, each time the PoMS is answered four times, the equivalent of a loop in the Python implementation needs to be done: update Q-values and send a new recommendation. The recommendations can then be displayed to the user.

To make sure the whole process works, the server will keep track of the algorithm information, such as the defined parameters, the Q-table, the last suggested action, in order to be able to know where to update the Q-value, and also how many times the PoMS has been answered. To make sure this information remains associated with the right user, the *Session* class was used. *Session* allows for the storage of data in the server, associated with a specific user, so it can be recovered the next time the same user interacts with it. It is necessary to define a secret key, to allow for encryption, and each client needs an unique id to be uniquely identified. (Flask, n.d.; Python basics, n.d.)

4. Evaluation results

To use these methods in a server it is relevant to test non-functional aspects such as performance and load testing. This code is intended to be available to multiple users and it should be able to provide fast responses.

There are three endpoints, as explained previously, and these tests were performed for all three of them.

Firstly, some performance tests were made. To perform these tests, *pytest* library was used as a test runner, and by making use of a custom 'mark' it was possible to categorize the tests as performance tests, for code organization. (Pytest, n.d.) Then, the *requests* library was used to call the *Flask* endpoints, as it allows to call different endpoints and send headers, query parameters and body. (Requests: HTTP for Humans™, n.d.) The results were then saved in a *csv* file for analysis.

A loop was created that calls an endpoint 100 times. The time that each call takes to be completed is recorded and saved to a *csv* file. Then, the mean, minimum value and maximum value were calculated and are here presented in table 5.

Table 5. Performance testing on *Flask* endpoints (in seconds).

Endpoint	Minimum value	Maximum value	Mean
/initialize	0.02419	0.08807	0.03646
/recommendations	0.047141	0.186566	0.066628
Conscientiousness	0.081525	0.257489	0.105654

The values obtained are fairly fast, since they are all below 1 second.

Besides checking for the response time of the endpoints when they are called individually, some load tests were performed to check if they can handle different users interacting with them at the same time.

For this purpose, the *locust* library was used. With this library, it is possible to create tasks that should be run, in this case, calls to the endpoints. When a test is run, a class is created for each simulated user. Simulated users can execute tasks, on random orders and with some interval in between. The initialize endpoint is called inside *on_start*, so that each time a user is started, it will run the initialize. This way, errors by lack of the learning agent are avoided since they do not actually matter in the context of load testing. (*Writing a locustfile*, n.d.) When a load test is run using *locust*, there are some parameters that can be defined. In this test, the parameters used were the number of users, that indicates the maximum number of concurrent users, the spawn rate, that defines how many users per second should spawn, and the run time, that defines for how long the test will run. (*Configuration*, n.d.)

The run time for each test was 15 minutes, with 100 as number of users and 10 as spawn rate. This is a fairly small number, that was selected because the *Flask* code hasn't been yet deployed to a production or production-like server. So, the tests were performed locally, using localhost, which brings limitations, as the availability is highly related to the available resources of the machine hosting. However, the tests were still performed as a way to observe the behaviour of the endpoints and to have a general overview of concurrent performance.

Figure 8 shows the number of users over time. As the users grew at a rate of 10 per second, they went up to 100 quickly, and remained at 100 throughout the rest of the time.

Figure 9 shows the requests per second as well as the failures per second. It is possible to see they are directly correlated by the way the lines resemble each other. This indicates that the failures observed can be related to the load.

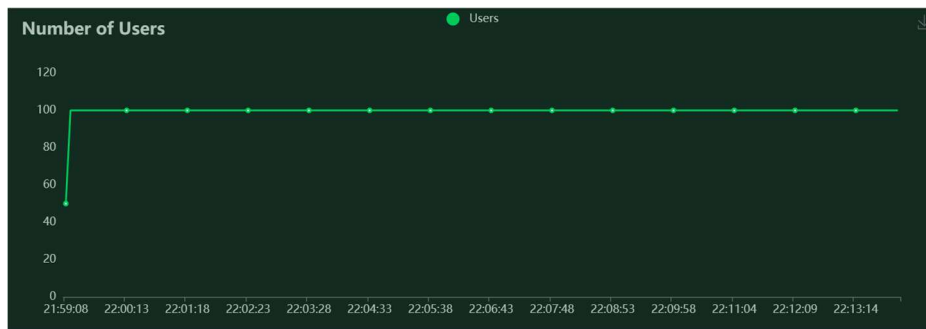


Figure 9 - Number of users over time

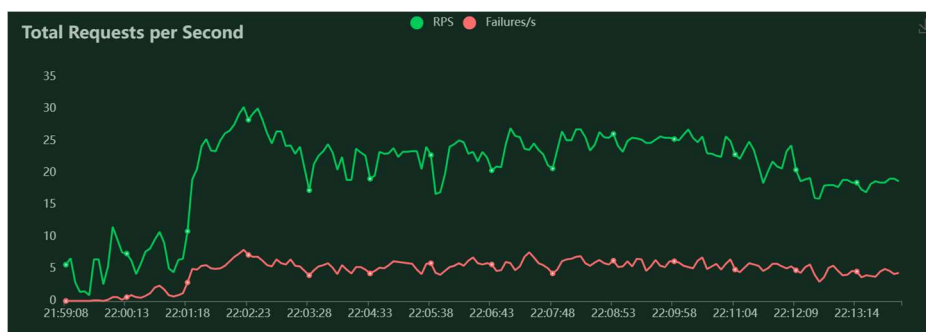


Figure 8 - Requests per second over time

Checking the statistics, it is possible to see that there was a total of 18429 requests, with 4281 being failures, which comes to about 23.23%. From the 4281 total failures, 4239 were when calling /qtable endpoint, which indicates /qtable might need more resources to fully run, and as the number of requests increases, it starts to fail more. This also adds to the fact that it has the largest average size (in bytes). The code for /qtable is also the one that makes more calculations.

Like previously mentioned, these tests were performed in a localhost server, so they do not reflect the real behaviour in a production server. Once this code is deployed, more accurate tests can be performed. However, for the performance tests it is possible to see that when the code runs individually, it is fast. From the load testing, it is clear that /qtable takes more resources, and this could be further verified in a production server, to make sure there are enough resources for it to run properly.

5. Real platforms

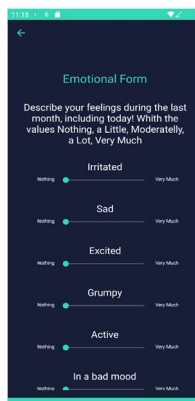
To test the solution presented, and have a clear vision about how the algorithm works when used by multiple different people, it is important to make some real tests with volunteers. However, due to the nature of the solution, it is meant to be used during an extended period of time, ideally at least for a semester, so tests are dependent on academic years. To do such tests in upcoming semesters, there are some platforms available where the algorithm is already embedded and can be used.

5.1. ISABELA

ISABELA (IoT Student Advisor and BESt Life Analyser) is a system based on several technologies, developed by our research group at FCTUC (*Faculdade de Ciências e Tecnologias da Universidade de Coimbra*). Its main goal is to help university students with academic performance by analysing multiple factors, both physical and psychological. It aims to be a full Human-in-the-loop system, by analysing users' data and using it to provide recommendations and, that way, making the user a central piece of the application. (Nunes, Zhang & Silva, 2015)

To accomplish this, ISABELA is being developed using Xamarin, so it can be available for both Android and iOS devices and collects data in different ways:

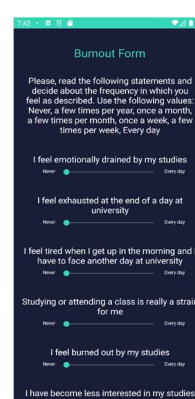
- without user's interaction - using smartphone sensors such as the accelerometer, light sensor, proximity sensor, gyroscope, to name a few
- with user's interaction - by answering the app's psychology questionnaires, as well as interacting with a chatbot



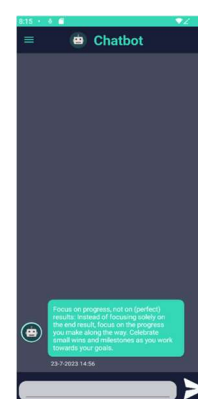
(a) Screen of the PoMS questionnaire



(b) Screen of the perfectionism questionnaire



(c) Screen of the burnout questionnaire



(d) Screen of the chatbot with a recommendation

ISABELA is available in both Portuguese and English, and already includes all the questionnaires mentioned in subsection 3.1. Every questionnaire's Portuguese version has been validated and revealed good psychometric properties. In figure 9 it is possible to see some screens from ISABELA. The questionnaires are easy to fill out, as they have sliders between the possible values for each question. There is also a chatbot already implemented that makes use of DialogFlow to answer simple questions, and can now also interact with the RL model, by using the *Flask* server with the methods.

5.2. BATINA

Besides ISABELA, there is other application developed by this research group, named BATINA (Best Assisting Tutor and INteractive Advisor). Its purpose is more related with academic goals, providing a connection between students and teachers. Teachers can put materials and questionnaires in the app, for students to answer, and students can get some information about performance and interact with a chatbot.

In figure 10 some screens from the application are presented.

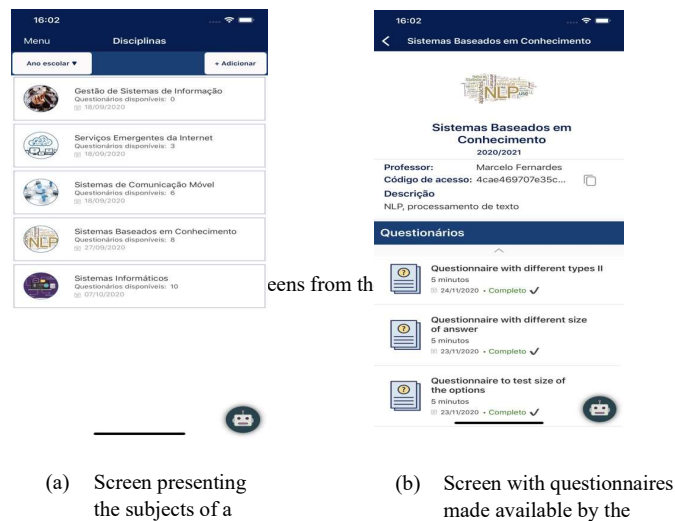


Figure 11 - Some screens of BATINA

6. Conclusions

The work presented on this article is a step towards new approaches to improve mental well-being and academic performance in college students. By using mobile applications, that are easy and convenient, making use of Reinforcement Learning methods and combining psychological medicine with technology, it can be possible to achieve full Human-in-the-loop applications that add value to a student's daily life.

Further testing needs to be done to prove this approach's utility and usability. Such testing is being prepared in the group to be conducted in future academic semesters, by making use of the real platforms available and completing all the infrastructure developments needed to host the RL model in a way it supports interaction from lots of users at the same time.

Acknowledgements

This work is funded by the project 2023 - PDS & VDS from INESC Coimbra.

References

- World Health Organization (WHO) (2019, December 19). *Mental health*. Available: https://www.who.int/health-topics/mental-health#tab=tab_1 Accessed August 5, 2023
- Mahdavi, P., Valibeygi, A., Moradi, M., & Sadeghi, S. (2023). Relationship between achievement motivation, mental health and academic success in university students. *Community Health Equity Research & Policy*, 43(3), 311-317.
- Gardani, M., Bradford, D. R., Russell, K., Allan, S., Beattie, L., Ellis, J. G., & Akram, U. (2022). A systematic review and meta-analysis of poor sleep, insomnia symptoms and stress in undergraduate students. *Sleep medicine reviews*, 61, 101565.
- Dados e Estatísticas de Cursos Superiores*. (n.d.). Available: <https://infocursos.medu.pt/> Accessed August 5, 2023
- Emily, J. (2023). Cognitive strategies, motivation to learn, levels of well-being, and risk of dropout: an empirical longitudinal study to qualify ongoing university guidance services. *ENDLESS: INTERNATIONAL JOURNAL OF FUTURE STUDIES*, 6(1), 1-10.
- European Commission. (2020). Resetting education and training for the digital age. https://ec.europa.eu/education/education-in-the-eu/digital-education-action-plan_en Accessed September 12, 2023
- R. Wang, F. Chen, Z. Chen, T. Li, G. Harari, S. Tignor, X. Zhou, D. Ben-Zeev, and A. T. Campbell, (2014) "Studentlife: assessing mental health, academic performance and behavioral trends of college students using smartphones," in Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, pp. 3–14.
- R. Wang, G. Harari, P. Hao, X. Zhou, and A. T. Campbell, (2015) "SmartGPA," in Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15.
- R. Tibshirani, (1996) "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288.
- G. M. Harari, S. D. Gosling, R. Wang, F. Chen, Z. Chen, and A. T. Campbell, (2017) "Patterns of behavior change in students over an academic term: A preliminary study of activity and sociability behaviors using smartphone sensing methods," *Comput. Human Behav.*
- R. Wang et al. (2014), "StudentLife," in Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct.
- P. Eskes, M. Spruit, S. Brinkkemper, J. Vorstman, and M. J. Kas, (2016) "The sociability score: App-based social profiling from a healthcare perspective," *Comput. Human Behav.*, vol. 59, pp. 39–48.
- Ashton, M. C., & Lee, K. (2009). The HEXACO-60: A short measure of the major dimensions of personality. *Journal of personality assessment*, 91(4), 340-345.
- Bleidorn, W., Schwaba, T., Zheng, A., Hopwood, C. J., Sosa, S. S., Roberts, B. W., & Briley, D. A. (2022). Personality stability and change: A meta-analysis of longitudinal studies. *Psychological bulletin*, 148(7-8), 588.
- Damian, R. I., Spengler, M., Sutu, A., & Roberts, B. W. (2019). Sixteen going on sixty-six: A longitudinal study of personality stability and change across 50 years. *Journal of Personality and Social Psychology*, 117(3), 674.
- K. Lee and M. C. Ashton, The hexaco personality inventory - revised. [Online]. Available: <https://hexaco.org/scaledescriptions> (visited on Jan. 22, 2023).
- Lewis, E. G., & Cardwell, J. M. (2020). The big five personality traits, perfectionism and their association with mental health among UK students on professional degree programmes. *BMC psychology*, 8, 1-10.
- Kotov, R., Gamez, W., Schmidt, F., & Watson, D. (2010). Linking "big" personality traits to anxiety, depressive, and substance use disorders: a meta-analysis. *Psychological bulletin*, 136(5), 768.
- Kang, W., Steffens, F., Pineda, S., Widuch, K., & Malvaso, A. (2023). Personality traits and dimensions of mental health. *Scientific Reports*, 13(1), 7091.
- Luo, J., Zhang, B., Cao, M., & Roberts, B. W. (2023). The stressful personality: A meta-analytical review of the relation between personality and stress. *Personality and social psychology review*, 27(2), 128-194.
- Goodwin, R. D., & Friedman, H. S. (2006). Health status and the five-factor personality traits in a nationally representative sample. *Journal of health psychology*, 11(5), 643-654.
- Kern, M. L., & Friedman, H. S. (2008). Do conscientious individuals live longer? A quantitative review. *Health psychology*, 27(5), 505.
- McNair, D. M., Lorr, M., Droppleman, L. F., (1971) "Manual profile of mood states," San Diego, CA: Educational and Industrial Testing Service.
- Pereira, A. T., Araújo, A. I., Cabaços, C., Brito, M. J., Fernandes, M., Rodrigues, A., ... & Macedo, A. (2023). Profile of mood states-12: same validity, more usability. *European Psychiatry*, 66(S1), S553-S554.
- Shacham, S. (1983). A shortened version of the Profile of Mood States. *Journal of personality assessment*, 47(3), 305-306.
- McNair, D. M., Lorr, M., & Droppleman, L. F. (1992). *EdITS Manual for the Profile of Mood States (POMS)*. Educational and industrial testing service.
- Bolger, N., Davis, A., & Rafaeli, E. (2003). Diary methods: Capturing life as it is lived. *Annual review of psychology*, 54(1), 579-616.
- Cranford, J. A., Shrout, P. E., Iida, M., Rafaeli, E., Yip, T., & Bolger, N. (2006). A procedure for evaluating sensitivity to within-person change: Can mood measures in diary studies detect change reliably?. *Personality and Social Psychology Bulletin*, 32(7), 917-929.
- Beedie, C., Terry, P., & Lane, A. (2005). Distinctions between emotion and mood. *Cognition & Emotion*, 19(6), 847-878.
- Smith, M. M., Saklofske, D. H., Stoeber, J., & Sherry, S. B. (2016). The big three perfectionism scale: A new measure of perfectionism. *Journal of Psychoeducational Assessment*, 34(7), 670-687.
- Feher, A., Smith, M. M., Saklofske, D. H., Plouffe, R. A., Wilson, C. A., & Sherry, S. B. (2020). The Big three perfectionism scale-short form (BTPS-SF): development of a brief self-report measure of multidimensional perfectionism. *Journal of Psychoeducational Assessment*, 38(1), 37-52.

- Dunkley, D. M., Zuroff, D. C., & Blankstein, K. R. (2003). Self-critical perfectionism and daily affect: dispositional and situational influences on stress and coping. *Journal of personality and social psychology, 84*(1), 234.
- Cabaços, C., Macedo, A., Carneiro, M., Brito, M. J., Amaral, A. P., Araújo, A., ... & Pereira, A. T. (2023). The mediating role of self-compassion and repetitive negative thinking in the relationship between perfectionism and burnout in health-field students: A prospective study. *Personality and Individual Differences, 213*, 112314.
- Pereira, A. T., Brito, M. J., Cabaços, C., Carneiro, M., Carvalho, F., Manão, A., ... & Macedo, A. (2022). The protective role of self-compassion in the relationship between perfectionism and burnout in Portuguese medicine and dentistry students. *International Journal of Environmental Research and Public Health, 19*(5), 2740.
- Maslach, C., Jackson, S., and Leiter, M., (1997, Jan.) The Maslach Burnout Inventory Manual., vol. 3, pp. 191–218
- Madigan, D. J., & Curran, T. (2021). Does burnout affect academic achievement? A meta-analysis of over 100,000 students. *Educational Psychology Review, 33*, 387-405.
- Schaufeli, W. B., Martinez, I. M., Pinto, A. M., Salanova, M., & Bakker, A. B. (2002). Burnout and engagement in university students: A cross national study. *Journal of cross-cultural psychology, 33*(5), 464-481.
- Sousa, P. R. S. (2018). *A versão Portuguesa do HEXACO-60: Estudo psicométrico numa amostra de estudantes universitários* (Master's thesis).
- Pereira, A. T., Brito, M. J., Marques, C., Araújo, A. I., Cabaços, C., Carneiro, M., ... & Macedo, A. (2023). Development and first validation of the Portuguese version of the Big Three Perfectionism Scale–Short Form (BTPS-SF). *European Psychiatry, 66*(S1), S174-S175.
- Manão, A., Carneiro, M., Carvalho, F., et al., (2021) “Maslach burnout inventory – students survey: Validation for portuguese medical students.” Poster presented at 21st WPA World Congress of Psychiatry.
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning, 8*, 279-292.
- Zhou, X. (2022, December). Optimal Values Selection of Q-learning Parameters in Stochastic Mazes. In *Journal of Physics: Conference Series* (Vol. 2386, No. 1, p. 012037). IOP Publishing.
6. Python Documentation (n.d.) *Expressions* Available: <https://docs.python.org/3/reference/expressions.html> Accessed August 3, 2023
- Flask (n.d.) *Quickstart — Flask Documentation (2.3.x)*. Available: <https://flask.palletsprojects.com/en/2.3.x/quickstart/> Accessed August 4, 2023
- Jschema (n.d.) *jschema* Available: <https://python-jschema.readthedocs.io/en/stable/> Accessed August 4, 2023
- Python Basics (n.d.) *Session data in Python Flask - Python Tutorial*. Available: <https://pythonbasics.org/flask-sessions> Accessed August 4, 2023
- Pytest (n.d.) *API Reference — pytest documentation*. Available: <https://docs.pytest.org/en/7.4.x/reference/reference.html> Accessed August 31, 2023
- Requests: HTTP for Humans™ (n.d.) *Requests 2.31.0 documentation*. Available: <https://requests.readthedocs.io/en/latest> Accessed August 31, 2023
- Writing a locustfile* (n.d.). Available: <https://docs.locust.io/en/stable/writing-a-locustfile.html> Accessed September 1, 2023
- Configuration* (n.d.). Available: <https://docs.locust.io/en/stable/configuration.html#command-line-options> Accessed September 1, 2023
- Nunes, D. S., Zhang, P., & Silva, J. S. (2015). A survey on human-in-the-loop applications towards an internet of all. *IEEE Communications Surveys & Tutorials, 17*(2), 944-965.

Bibliography

- [1] *Mental health, world health organization (who)*, Jul. 2023. [Online]. Available: https://www.who.int/health-topics/mental-health#tab=tab_1 (visited on Aug. 5, 2023).
- [2] M. Gardani, D. R. Bradford, K. Russell, *et al.*, “A systematic review and meta-analysis of poor sleep, insomnia symptoms and stress in undergraduate students”, *Sleep medicine reviews*, vol. 61, p. 101565, 2022. DOI: <https://doi.org/10.1016/j.smr.2021.101565>.
- [3] P. Mahdavi, A. Valibeygi, M. Moradi, and S. Sadeghi, “Relationship between achievement motivation, mental health and academic success in university students”, *Community Health Equity Research & Policy*, vol. 43, no. 3, pp. 311–317, 2023. DOI: <https://doi.org/10.1177/0272684X211025932>.
- [4] *Dados e estatísticas de cursos superiores - edição 2023*. [Online]. Available: <https://infocursos.medu.pt/> (visited on Aug. 5, 2023).
- [5] J. Emily, “Cognitive strategies, motivation to learn, levels of well-being, and risk of dropout: An empirical longitudinal study to qualify ongoing university guidance services.”, *ENDLESS: INTERNATIONAL JOURNAL OF FUTURE STUDIES*, vol. 6, no. 1, pp. 1–10, 2023.
- [6] *What is xamarin? - xamarin*, 2022. [Online]. Available: <https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin> (visited on Jan. 21, 2023).
- [7] *What is xamarin.forms? - xamarin*, 2021. [Online]. Available: <https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms> (visited on Jan. 22, 2023).
- [8] *What is .net maui? - .net maui*, Jan. 2023. [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui> (visited on Jul. 26, 2023).
- [9] *Xamarin support policy*. [Online]. Available: <https://dotnet.microsoft.com/en-us/platform/support/policy/xamarin> (visited on Jul. 26, 2023).

- [10] *Developers catalogue*. [Online]. Available: <https://www.fiware.org/catalogue/> (visited on Jul. 7, 2023).
- [11] *Introduction to dash*. [Online]. Available: <https://dash.plotly.com/introduction> (visited on Jan. 15, 2023).
- [12] *Frequently asked questions*. [Online]. Available: <https://dash.plotly.com/faqs> (visited on Jan. 15, 2023).
- [13] *Dialogflow es documentation - google cloud*. [Online]. Available: <https://cloud.google.com/dialogflow/es/docs> (visited on Mar. 11, 2023).
- [14] *Dialogflow es basics - google cloud*. [Online]. Available: <https://cloud.google.com/dialogflow/es/docs/basics> (visited on Mar. 11, 2023).
- [15] *.net client library - google cloud*. [Online]. Available: <https://cloud.google.com/dotnet/docs/reference/help/platforms> (visited on Mar. 11, 2023).
- [16] D. S. Nunes, P. Zhang, and J. S. Silva, “A survey on human-in-the-loop applications towards an internet of all”, *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 944–965, 2015. DOI: 10.1109/COMST.2015.2398816.
- [17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [18] *General python faq*. [Online]. Available: <https://docs.python.org/3/faq/general.html> (visited on Jul. 7, 2023).
- [19] *General data protection regulation (gdpr)*. [Online]. Available: <https://gdpr-info.eu/> (visited on Jul. 26, 2023).
- [20] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning”, *Computers & Industrial Engineering*, vol. 149, p. 106 854, 2020. DOI: <https://doi.org/10.1016/j.cie.2020.106854>.
- [21] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, “A survey on federated learning”, *Knowledge-Based Systems*, vol. 216, p. 106 775, 2021.
- [22] A. Pereira, A. Araújo, and A. Macedo, “Personalidade, saúde e doença”, *A Macedo, AT Pereira & Madeira N.(Coords.) Psicologia na Medicina*, pp. 241–260, 2018.
- [23] M. Komarraju, S. J. Karau, R. R. Schmeck, and A. Avdic, “The big five personality traits, learning styles, and academic achievement”, *Personality and individual differences*, vol. 51, no. 4, pp. 472–477, 2011. DOI: <https://doi.org/10.1016/j.paid.2011.04.019>.
- [24] A. E. Poropat, “A meta-analysis of the five-factor model of personality and academic performance.”, *Psychological bulletin*, vol. 135, no. 2, p. 322, 2009. DOI: <https://psycnet.apa.org/doi/10.1037/a0014996>.

-
- [25] M. C. Ashton and K. Lee, “The hexaco–60: A short measure of the major dimensions of personality”, *Journal of personality assessment*, vol. 91, no. 4, pp. 340–345, 2009. DOI: <https://doi.org/10.1080/00223890902935878>.
- [26] W. Bleidorn, T. Schwaba, A. Zheng, *et al.*, “Personality stability and change: A meta-analysis of longitudinal studies.”, *Psychological bulletin*, vol. 148, no. 7-8, p. 588, 2022. DOI: <https://psycnet.apa.org/doi/10.1037/bu10000365>.
- [27] R. I. Damian, M. Spengler, A. Sutu, and B. W. Roberts, “Sixteen going on sixty-six: A longitudinal study of personality stability and change across 50 years.”, *Journal of Personality and Social Psychology*, vol. 117, no. 3, p. 674, 2019. DOI: <https://psycnet.apa.org/doi/10.1037/pspp0000210>.
- [28] K. Lee and M. C. Ashton, *The hexaco personality inventory - revised*. [Online]. Available: <https://hexaco.org/scaledescriptions> (visited on Jan. 22, 2023).
- [29] D. M. McNair, M. Lorr, L. F. Droppleman, *et al.*, “Manual profile of mood states”, 1971.
- [30] J. Rogaten, G. B. Moneta, and M. M. Spada, “Academic performance as a function of approaches to studying and affect in studying”, *Journal of happiness studies*, vol. 14, no. 6, pp. 1751–1763, 2013. DOI: <https://doi.org/10.1007/s10902-012-9408-5>.
- [31] A. Pereira, A. Araújo, C. Cabaços, *et al.*, “Profile of mood states-12: Same validity, more usability.”, *European Psychiatry*, vol. 66, no. S1, S553–S554, 2023. DOI: 10.1192/j.eurpsy.2023.1168.
- [32] S. Shacham, “A shortened version of the profile of mood states”, *Journal of personality assessment*, vol. 47, no. 3, pp. 305–306, 1983. DOI: https://doi.org/10.1207/s15327752jpa4703_14.
- [33] C. Maslach, S. Jackson, and M. Leiter, *The Maslach Burnout Inventory Manual*. Jan. 1997, vol. 3, pp. 191–218.
- [34] D. J. Madigan and T. Curran, “Does burnout affect academic achievement? a meta-analysis of over 100,000 students”, *Educational Psychology Review*, vol. 33, pp. 387–405, 2021. DOI: <https://doi.org/10.1007/s10648-020-09533-1>.
- [35] W. B. Schaufeli, I. M. Martinez, A. M. Pinto, M. Salanova, and A. B. Bakker, “Burnout and engagement in university students: A cross-national study”, *Journal of cross-cultural psychology*, vol. 33, no. 5, pp. 464–481, 2002. DOI: <https://doi.org/10.1177/0022022102033005003>.
- [36] F. Carvalho, C. Cabaços, M. Carneiro, *et al.*, “Mindfulness and self-compassion based intervention program to prevent burnout in medical and dentistry stu-

- dents”, *European Psychiatry*, vol. 64, no. S1, S459–S460, 2021. DOI: [10.1192/j.eurpsy.2021.1228](https://doi.org/10.1192/j.eurpsy.2021.1228).
- [37] R. O. Frost, P. Marten, C. Lahart, and R. Rosenblate, “The dimensions of perfectionism”, *Cognitive therapy and research*, vol. 14, pp. 449–468, 1990. DOI: <https://doi.org/10.1007/BF01172967>.
- [38] M. M. Smith, D. H. Saklofske, J. Stoeber, and S. B. Sherry, “The big three perfectionism scale: A new measure of perfectionism”, *Journal of Psychoeducational Assessment*, vol. 34, no. 7, pp. 670–687, 2016. DOI: <https://doi.org/10.1177/0734282916651539>.
- [39] A. Feher, M. M. Smith, D. H. Saklofske, R. A. Plouffe, C. A. Wilson, and S. B. Sherry, “The big three perfectionism scale–short form (btps-sf): Development of a brief self-report measure of multidimensional perfectionism”, *Journal of Psychoeducational Assessment*, vol. 38, no. 1, pp. 37–52, 2020. DOI: <https://doi.org/10.1177/0734282919878553>.
- [40] D. M. Dunkley, D. C. Zuroff, and K. R. Blankstein, “Self-critical perfectionism and daily affect: Dispositional and situational influences on stress and coping.”, *Journal of personality and social psychology*, vol. 84, no. 1, p. 234, 2003. DOI: <https://psycnet.apa.org/doi/10.1037/0022-3514.84.1.234>.
- [41] C. Cabaços, A. Macedo, M. Carneiro, *et al.*, “The mediating role of self-compassion and repetitive negative thinking in the relationship between perfectionism and burnout in health-field students: A prospective study”, *Personality and Individual Differences*, vol. 213, p. 112314, 2023. DOI: <https://doi.org/10.1016/j.paid.2023.112314>.
- [42] A. T. Pereira, M. J. Brito, C. Cabaços, *et al.*, “The protective role of self-compassion in the relationship between perfectionism and burnout in portuguese medicine and dentistry students”, *International Journal of Environmental Research and Public Health*, vol. 19, no. 5, p. 2740, 2022. DOI: <https://doi.org/10.3390/ijerph19052740>.
- [43] A. Pereira, M. Brito, C. Marques, *et al.*, “Development and first validation of the portuguese version of the big three perfectionism scale–short form (btps-sf)”, vol. 66, no. S1, S174–S175, 2023. DOI: [10.1192/j.eurpsy.2023.420](https://doi.org/10.1192/j.eurpsy.2023.420).
- [44] *The model-view-viewmodel pattern - xamarin*, Jul. 2021. [Online]. Available: <https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm> (visited on Feb. 26, 2023).
- [45] *Xamarin.forms resource dictionaries - xamarin*, Sep. 2022. [Online]. Available: <https://learn.microsoft.com/en-us/xamarin/xamarin-forms/xaml/resource-dictionaries> (visited on Mar. 16, 2023).

-
- [46] A. S. Neto, *Tempo de espera para consultas de saúde mental motiva a procura de ajuda em chatbots*, Feb. 2023. [Online]. Available: <https://pplware.sapo.pt/high-tech/tempo-de-espera-para-consultas-de-saude-mental-motiva-a-procura-de-ajuda-em-chatbots/> (visited on Mar. 20, 2023).
- [47] S. Henry, I. Thielmann, T. Booth, and R. Mõttus, “Test-retest reliability of the hexaco-100—and the value of multiple measurements for assessing reliability”, *PloS one*, vol. 17, no. 1, e0262465, 2022. DOI: <https://doi.org/10.1371/journal.pone.0262465>.
- [48] P. R. S. Sousa, “A versão portuguesa do hexaco-60: Estudo psicométrico numa amostra de estudantes universitários”, M.S. thesis, University of Coimbra, <http://hdl.handle.net/10316/82008>, 2019.
- [49] F. M. V. Lino, “A versão portuguesa do big three perfectionism scale: Estudo psicométrico numa amostra de estudantes universitários”, M.S. thesis, University of Coimbra, <http://hdl.handle.net/10316/82423>, 2018.
- [50] S.-W. Yang and M. Koo, “The big five personality traits as predictors of negative emotional states in university students in taiwan”, *International Journal of Environmental Research and Public Health*, vol. 19, no. 24, p. 16 468, 2022. DOI: <https://doi.org/10.3390/ijerph192416468>.
- [51] J. Luo, B. Zhang, M. Cao, and B. W. Roberts, “The stressful personality: A meta-analytical review of the relation between personality and stress”, *Personality and social psychology review*, vol. 27, pp. 128–194, 2 2023. DOI: <https://doi.org/10.1177/10888683221104002>.
- [52] E. G. Lewis and J. M. Cardwell, “The big five personality traits, perfectionism and their association with mental health among uk students on professional degree programmes”, *BMC psychology*, vol. 8, pp. 1–10, 2020. DOI: <https://doi.org/10.1186/s40359-020-00423-3>.
- [53] R. Kotov, W. Gamez, F. Schmidt, and D. Watson, “Linking “big” personality traits to anxiety, depressive, and substance use disorders: A meta-analysis.”, *Psychological bulletin*, vol. 136, no. 5, p. 768, 2010. DOI: <https://psycnet.apa.org/doi/10.1037/a0020327>.
- [54] W. Kang, F. Steffens, S. Pineda, K. Widuch, and A. Malvaso, “Personality traits and dimensions of mental health”, *Scientific Reports*, vol. 13, no. 1, p. 7091, 2023. DOI: <https://doi.org/10.1038/s41598-023-33996-1>.
- [55] R. D. Goodwin and H. S. Friedman, “Health status and the five-factor personality traits in a nationally representative sample”, *Journal of health psychology*, vol. 11, no. 5, pp. 643–654, 2006. DOI: <https://doi.org/10.1177/1359105306066610>.

- [56] M. L. Kern and H. S. Friedman, “Do conscientious individuals live longer? a quantitative review.”, *Health psychology*, vol. 27, no. 5, p. 505, 2008. DOI: <https://psycnet.apa.org/doi/10.1037/0278-6133.27.5.505>.
- [57] J. Stoeber, *The psychology of perfectionism: Theory, research, applications*. Routledge, 2017.
- [58] T. Curran and A. P. Hill, “Perfectionism is increasing over time: A meta-analysis of birth cohort differences from 1989 to 2016.”, *Psychological bulletin*, vol. 145, no. 4, p. 410, 2019. DOI: <https://psycnet.apa.org/doi/10.1037/bul0000138>.
- [59] K. Boersma and K. Lindblom, “Stability and change in burnout profiles over time: A prospective study in the working population”, *Work & Stress*, vol. 23, no. 3, pp. 264–283, 2009. DOI: <https://doi.org/10.1080/02678370903265860>.
- [60] M. Carneiro, A. Macedo, E. Loureiro, *et al.*, “Inventory of sources of stress during medical education-further validation”, *European Psychiatry*, vol. 65, no. S1, S619–S619, 2022. DOI: <https://doi.org/10.1192/j.eurpsy.2022.1586>.
- [61] A. Pereira, C. Cabaços, A. Araújo, *et al.*, “Covid-19 pandemic: Another source of stress for medical students”, *European Psychiatry*, vol. 65, no. S1, S495–S496, 2022. DOI: <https://doi.org/10.1192/j.eurpsy.2022.1260>.
- [62] D. McNair, M. Lorr, and L. Droppleman, “Its manual for the profile of mood states san diego”, *CA: Educational and Industrial Testing Service*, 1992.
- [63] D. Watson and J. Vaidya, “Mood measurement: Current status and future directions”, *Handbook of psychology: Research methods in psychology*, vol. 2, pp. 351–375, 2003.
- [64] K. Petrowski, C. Albani, M. Zenger, E. Brähler, and B. Schmalbach, “Revised short screening version of the profile of mood states (poms) from the german general population”, *Frontiers in psychology*, vol. 12, p. 631668, 2021. DOI: <https://doi.org/10.3389/fpsyg.2021.631668>.
- [65] C. Beedie, P. Terry, and A. Lane, “Distinctions between emotion and mood”, *Cognition & Emotion*, vol. 19, no. 6, pp. 847–878, 2005. DOI: <https://doi.org/10.1080/02699930541000057>.
- [66] N. Bolger, A. Davis, and E. Rafaeli, “Diary methods: Capturing life as it is lived”, *Annual review of psychology*, vol. 54, no. 1, pp. 579–616, 2003. DOI: <https://doi.org/10.1146/annurev.psych.54.101601.145030>.
- [67] J. A. Cranford, P. E. Shrout, M. Iida, E. Rafaeli, T. Yip, and N. Bolger, “A procedure for evaluating sensitivity to within-person change: Can mood measures in diary studies detect change reliably?”, *Personality and Social*

- Psychology Bulletin*, vol. 32, no. 7, pp. 917–929, 2006. DOI: <https://doi.org/10.1177/0146167206287721>.
- [68] A. Manão, M. Carneiro, F. Carvalho, *et al.*, “Maslach burnout inventory – students survey: Validation for portuguese medical students.”, *Poster presented at 21st WPA World Congress of Psychiatry.*, 2021. DOI: <http://dx.doi.org/10.13140/RG.2.2.17827.12323>.
- [69] C. J. Watkins and P. Dayan, “Q-learning”, *Machine learning*, vol. 8, pp. 279–292, 1992. DOI: <https://doi.org/10.1007/BF00992698>.
- [70] X. Zhou, “Optimal values selection of q-learning parameters in stochastic mazes”, in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2386, 2022, p. 012037. DOI: <https://doi.org/10.1088/1742-6596/2386/1/012037>.
- [71] *Expressions*. [Online]. Available: <https://docs.python.org/3/reference/expressions.html> (visited on Aug. 3, 2023).
- [72] *Numpy*, Aug. 2023. [Online]. Available: <https://numpy.org/> (visited on Aug. 4, 2023).
- [73] *Datetime - basic date and time types*. [Online]. Available: <https://docs.python.org/3/library/datetime.html> (visited on Aug. 4, 2023).
- [74] *Sqlite3 - db-api 2.0 interface for sqlite databases*. [Online]. Available: <https://docs.python.org/3/library/sqlite3.html> (visited on Aug. 4, 2023).
- [75] *Intro do data structures*. [Online]. Available: https://pandas.pydata.org/docs/user_guide/dsintro.html (visited on Aug. 4, 2023).
- [76] *Itertools - functions creating iterators for efficient looping*. [Online]. Available: <https://docs.python.org/3/library/itertools.html> (visited on Aug. 4, 2023).
- [77] *Tkinter - python interface to tcl/tk*. [Online]. Available: <https://docs.python.org/3/library/tkinter.html> (visited on Aug. 4, 2023).
- [78] *Quickstart flask documentation*. [Online]. Available: <https://flask.palletsprojects.com/en/2.3.x/quickstart/> (visited on Aug. 4, 2023).
- [79] *Store data in a local sqlite.net database - xamarin*, Dec. 2022. [Online]. Available: <https://learn.microsoft.com/en-us/xamarin/get-started/quickstarts/database> (visited on Aug. 4, 2023).
- [80] *Jschema*. [Online]. Available: <https://python-jschema.readthedocs.io/en/stable/> (visited on Aug. 4, 2023).
- [81] *Session data in python flask*. [Online]. Available: <https://pythonbasics.org/flask-sessions/> (visited on Aug. 4, 2023).

- [82] *Unittest - unit testing framework*. [Online]. Available: <https://docs.python.org/3/library/unittest.html> (visited on Aug. 25, 2023).
- [83] *Api reference - pytest documentation*. [Online]. Available: <https://docs.pytest.org/en/7.4.x/reference/reference.html> (visited on Aug. 31, 2023).
- [84] *Requests: Http for humans*. [Online]. Available: <https://requests.readthedocs.io/en/latest/> (visited on Aug. 31, 2023).
- [85] *Csv — csv file reading and writing*. [Online]. Available: <https://docs.python.org/3/library/csv.html> (visited on Aug. 31, 2023).
- [86] *Time — time access and conversions*. [Online]. Available: <https://docs.python.org/3/library/time.html> (visited on Aug. 31, 2023).
- [87] *Writing a locustfile*. [Online]. Available: <https://docs.locust.io/en/stable/writing-a-locustfile.html> (visited on Sep. 1, 2023).
- [88] *Command line options*. [Online]. Available: <https://docs.locust.io/en/stable/configuration.html#command-line-options>.
- [89] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey”, *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017. DOI: 10.1109/MSP.2017.2743240.
- [90] P. Sousa, C. Marques, A. Pereira, *et al.*, “Confirmatory factor analysis of the hexaco-60 in a sample of portuguese university students”, in *European Psychiatry*, ELSEVIER FRANCE-EDITIONS SCIENTIFIQUES MEDICALES ELSEVIER 65 RUE CAMILLE . . . , vol. 48, 2018, S283–S283.