**DEVRELAX: STRESS DETECTING AND RELIEVING APPLICATION**

Bartholomeusz S.V

B.Sc. (Hons) Degree in Information Technology

(Specialization in Software Engineering)

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

Feb 2023

**DEVRELAX: STRESS DETECTING AND RELIEVING APPLICATION**

Bartholomeusz S.V IT20274702

The dissertation was submitted in partial fulfilment of the requirements

for the B.Sc. Special Honors degree in Information Technology

(Specialization in Software Engineering)

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

Feb 2023

# DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Signature |
| Bartholomeusz S.V | IT20274702 |  |

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor Date



…………………………. …………….…………

(Mr. Samadhi Rathnayake)

# ABSTRACT

This report introduces an innovative Stress-Relief and Emotion-Alleviation Activity Recommendation System that tackles the challenge of providing personalized activity suggestions for stress reduction and emotion alleviation. The recommendation task is cast as a sequential decision-making problem, leading to the adoption of Reinforcement Learning (RL) techniques for its implementation. The study leverages SlateQ, a recommendation slate generation algorithm developed by Google researchers. To achieve the dual objectives of stress reduction and emotion alleviation, the research integrates Multi-Objective Reinforcement Learning (MORL) with a weighted sum strategy. To evaluate the efficacy of the proposed system, experiments are conducted within the RecSim simulation environment. Comparative analyses are performed between the performance of the novel agent and a q-learning agent. The results show the proposed recommendation system outperforms the the q learning based recommendation system by looking at average rewards for time steps. By synergizing RL, MORL, and SlateQ, this study expands the landscape of recommendation systems, catering to the intricacies of managing stress and emotions through personalized activity recommendations. The designed recommendation system finds its application as a component of a desktop application we developed, tailored for workers, particularly those who spend extended durations in front of computers. By aiding in the enhancement of employee productivity, this research takes a step towards fostering emotional well-being. Ultimately, this work contributes to the advancement of intelligent systems dedicated to improving emotional welfare, offering a comprehensive framework adaptable across diverse contexts.

Keywords: Reinforcement Learning, Slate Recommendation, Deep Learning, Machine Learning, Recommendation Systems

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my dissertation supervisor Mr. Samadhi Rathnayake, and co-supervisor Ms. Devanshi Ganegoda from the Faculty of Computing, for their valuable guidance and support throughout the research process. Their perceptive comments and helpful critique assisted me in refining my research and improving the overall quality of my work.

I would also like to thank the CDAP team for their assistance and encouragement throughout the course of my studies. Their dedication to teaching and research has been an inspiration to me. I would also like to thank our panel members for their valuable comments and feedbacks throughout the research.

I am also grateful to my fellow team members for their support and collaboration. Our discussions and exchanges of ideas have enriched my understanding of the subject matter. I would like to acknowledge the assistance provided by Mrs. Arosha Dassanayake, as our external supervisor specialized in the field of psychology, for her generous support in the stress related domain, for this research project. Their contributions have been instrumental in making this research possible. I would also like to express my gratitude to the research team of the "xīnlĭ" mobile app for generously providing the datasets they have created, which was instrumental in the completion of my research.

Finally, I would like to express my appreciation to all those who participated in this study and contributed their time and insights. Their willingness to share their knowledge has been invaluable in helping me to gain a deeper understanding of the subject matter.

# TABLE OF CONTENTS

[DECLARATION i](#_Toc145479976)

[ABSTRACT ii](#_Toc145479977)

[ACKNOWLEDGEMENT iii](#_Toc145479978)

[TABLE OF CONTENTS iv](#_Toc145479979)

[LIST OF FIGURES vi](#_Toc145479980)

[LIST OF TABLES vii](#_Toc145479981)

[LIST OF ABBREVIATIONS vii](#_Toc145479982)

[1. INTRODUCTION 1](#_Toc145479983)

[**1.1.1.** **Background** 1](#_Toc145479984)

[**1.1.2** **Literature Survey** 2](#_Toc145479985)

[**1.2** **Research Gap** 8](#_Toc145479986)

[**1.3** **Research Problem** 12](#_Toc145479987)

[2. RESEARCH OBJECTIVES 14](#_Toc145479988)

[**2.1 Main Objectives** 14](#_Toc145479989)

[**2.2 Specific Objectives** 14](#_Toc145479990)

[3. METHODOLOGY 17](#_Toc145479991)

[**3.1.** **Methodology** 17](#_Toc145479992)

[3.1.1. Main Task 19](#_Toc145479993)

[3.1.2. Sub Tasks 19](#_Toc145479994)

[3.1.3. Data Collection 20](#_Toc145479995)

[3.1.4. Tools and Technologies 21](#_Toc145479996)

[**3.2.** **Commercialization Aspects of the Application** 21](#_Toc145479997)

[**3.3.** **Implementation and Testing** 23](#_Toc145479998)

[3.3.1. Implementation 23](#_Toc145479999)

[3.3.2. Testing 41](#_Toc145480000)

[4. RESULTS AND DISCUSSION 51](#_Toc145480001)

[**4.1.** **Results** 51](#_Toc145480002)

[**4.2.** **Discussion** 55](#_Toc145480003)

[5. CONCLUSION 57](#_Toc145480004)

[6. REFERENCES 59](#_Toc145480005)

[7. APPENDICES 63](#_Toc145480006)

[Appendix A: Online Survey for user preferences on activities 63](#_Toc145480007)

[Appendix B: Plagiarism Report 68](#_Toc145480008)

# LIST OF FIGURES

[Figure 1.1 : Paper publication in RL and DRL 3](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480009)

[Figure 1.2 : RL and DRL paper publication conferences and journals 4](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480010)

[Figure 1.3 : RecSim framework components 7](#_Toc145480011)

[Figure 3.1 : Recommendation system workflow 18](#_Toc145480012)

[Figure 3.2 : Activity Categorization 25](#_Toc145480013)

[Figure 3.3 : Reinforcement Learning Workflow 26](#_Toc145480014)

[Figure 3.4 : Q learning TD update algorithm 28](#_Toc145480015)

[Figure 3.5 : Q table overview of RS development attempt 1 29](#_Toc145480016)

[Figure 3.6 : Create Q table function 30](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480017)

[Figure 3.7 : Initialize Q table function 30](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480018)

[Figure 3.8 : Choose action function 31](#_Toc145480019)

[Figure 3.9 : Calculate reward function 32](#_Toc145480020)

[Figure 3.10 : Update Q value function 32](#_Toc145480021)

[Figure 3.11 : Neural network overview 34](#_Toc145480022)

[Figure 3.12 : Neural network build model 36](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480023)

[Figure 3.13 : Train network function 38](#_Toc145480024)

[Figure 3.14 : Replay function 38](#_Toc145480025)

[Figure 3.15 : Update target model function 38](#_Toc145480026)

[Figure 3.16 : act function 39](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480027)

[Figure 3.17 : Get recommendations API 39](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480028)

[Figure 3.18 :Update network API 40](#_Toc145480029)

[Figure 3.19 : Document model 42](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480030)

[Figure 3.20 : Document sampler 42](file:///C:\Users\barth\Downloads\IT20274702_Final_Report_Draft.docx#_Toc145480031)

[Figure 3.21 : User model 43](#_Toc145480032)

[Figure 3.22 : User response model 44](#_Toc145480033)

[Figure 3.23 : Simulate user response function 45](#_Toc145480034)

[Figure 3.24 : Generate response function 45](#_Toc145480035)

[Figure 3.25 : State updating and user model logic 46](#_Toc145480036)

[Figure 3.26 : Creating agents in RecSim 47](#_Toc145480037)

[Figure 3.27 : Test suite 1 47](#_Toc145480038)

[Figure 3.28 : Test suite 2 48](#_Toc145480039)

[Figure 3.29 : Test suite 3 48](#_Toc145480040)

[Figure 3.30 : Get recommendations API test 48](#_Toc145480040)

[Figure 3.31 : Update recommendations API test 48](#_Toc145480040)

[Figure 4.1 : Average episode reward in our agent 51](#_Toc145480041)

[Figure 4.2 : Average episode reward in DQN agent 52](#_Toc145480042)

[Figure 4.3 : Activities view 53](#_Toc145480040)

[Figure 4.4 : Attempt activity view 53](#_Toc145480040)

[Figure 4.5 : Activity feedback form 54](#_Toc145480040)

# LIST OF TABLES

[Table 1.1 : Research Gap based on previous research vs conducted research. 11](#_Toc145284834)

[Table 3.2 :Tools and Technologies used. 20](#_Toc145284835)

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| Abbreviation | Description |
| IT | Information Technology |
| CNN | Convolutional Neural Network |
| SC | Single Choice |
| RTDS | Reward/transition dependence on Selection |
| MDP | Markov Decision Process |
| RS | Recommendation System |
| MSE | Mean Squared Error |
| TQ | Total Q |
| AI | Artificial Intelligence |
| DRL | Deep Reinforcement Learning |
| API | Application Programming Interface |
| RL | Reinforcement Learning |
| MORL | Multi-Objective Reinforcement Learning |
| DQN | Deep Q Network |
| IOT | Internet of Things |
| ML | Machine Learning |

# INTRODUCTION

## **Background**

In the modern digital age, Information Technology (IT) professionals play a critical role in shaping the way businesses and society operate. The accelerated development of technology has resulted in an ever-increasing demand for IT expertise, transforming IT professionals into indispensable assets for businesses across various industries. However, the IT sector's dynamic nature, coupled with the constant pressure to adapt to emerging technologies and meet stringent deadlines, has led to a unique set of challenges for IT professionals. This has resulted in a growing concern about stress and emotion management within the IT workforce [1].

The IT industry is comprised of various roles that exhibit a wide range of responsibilities. These roles include software developers, network engineers, system administrators, data scientists, and IT managers, among others. These professionals are responsible for designing, implementing, and maintaining the digital infrastructure that underpins modern businesses. The scope of their work encompasses various tasks such as coding, troubleshooting, system maintenance, and prompt response to critical incidents, typically within high-pressure environments.

The nature of IT work is often associated with several factors that have the potential to induce stress. These factors include long working hours, tight project deadlines, and the constant need for continuous learning. [2].The emotional well-being of IT professionals is closely tied to stress levels. Prolonged stress can lead to emotional burnout, anxiety, and depression. Additionally, IT professionals may experience emotions related to their work, such as frustrations when troubleshooting difficult issues or satisfaction when solving complex problems. Emotional well-being is crucial as it affects job satisfaction, performance, and overall quality of life.

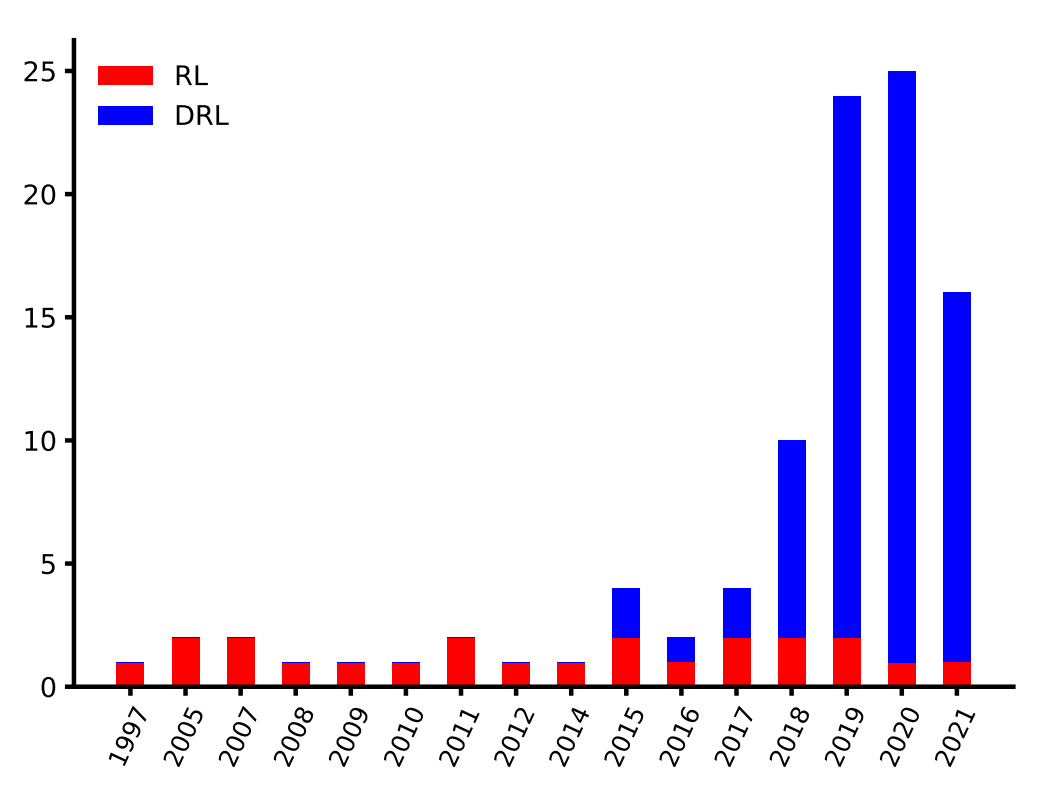
Hence it is very important to focus on developing a solution which addresses stress and emotion management of workers in the IT industry. This thesis discusses the stress relief and emotion alleviation recommendation system (RS) developed for the DevRelax application.

## **Literature Survey**

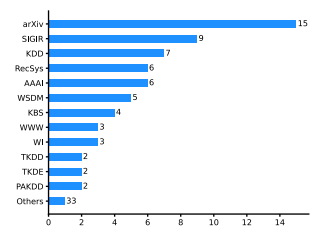
In the past, Conventional recommendation systems were developed based on Collaborative filtering – an approach that recommends items to a user by identifying other users who exhibit similar preferences based on their historical interactions with items. The system subsequently suggests items that have been favored or interacted by similar users but have not yet been encountered by the target user [3]. Content based filtering – an approach that recommends items to users based on the characteristics and attributes of the items themselves, as well as the preferences or history of the user [4]. The majority of conventional techniques, including the ones mentioned, approach the recommendation problem from the perspective of either prediction or classification.

Nevertheless, there is a growing consensus that representing the recommendation problem as a sequential decision-making problem would provide a more accurate depiction of user-system engagement [5]. RL has emerged as the predominant strategy for tackling sequential decision-making issues in the field of machine learning [6]. Several recommendation systems have been built using RL techniques [7] [8] [9]. These systems approach the suggestion problem by formulating it as a sequential decision-making problem. RL enables RSs to effectively manage consecutive, dynamic user-system interactions and effectively account for long-term user engagement.

Although RL techniques may offer a more effective framework for recommendation problems, the scalability of the RS is hindered by the behavior of the RL algorithm. Managing an extensive action-value space poses challenges for conventional RL methods. To overcome this issue, function approximation methods can be used to approximate action-value function. One possible approach involves utilizing a neural network to estimate the action-value function. DRL involves the integration of deep learning methodologies within the framework of RL. To be more precise, the utilization of a convolutional neural network (CNN) is employed for the purpose of function approximation. The utilization of DRL techniques in the development of RSs has shown a notable increase in recent years. [5]. According to a survey conducted by Mehedi et al. [5] there has been an observed increase in the number of research paper publications focused on RL based RSs compared to those based on DRL.



*Figure 1.1 : Paper publication in RL and DRL*



*Figure 1.2 : RL and DRL paper publication conferences and journals*

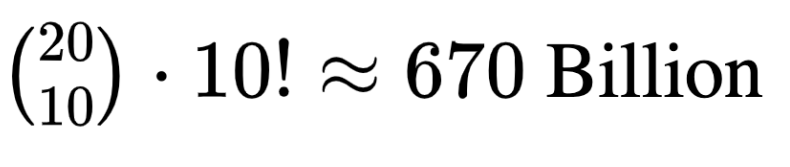
RL algorithms are designed to optimize a single objective and it usually deals with one reward function. However, in cases where many objectives need to be optimized or achieved, traditional RL methods may not be the most suitable choice. In such scenarios, MORL techniques can be employed. MORL is a specialized domain within the area of RL, wherein a smart agent acquires the ability to make optimal decisions by concurrently optimizing numerous objectives.

If the objectives inside the MORL domain exhibit no interdependence, they can be individually optimized, followed by the development of a strategy that optimizes all objectives collectively. In the event that the objectives are incompatible, it becomes necessary to optimize a singular strategy that prioritizes a single target while simultaneously negotiating a trade-off between objectives. [10]. MORL approaches can be divided into two groups based on the number of policies they learn. One group is focused on identifying a single-policy MORL approach [11] that reflects either the individual's preferences or the requirements of the problem. The other group aims to establish a combination of policies, referred to as a multiple-policy MORL method [12], that closely approximates the best possible outcomes for diverse objectives.

According to different scenarios different types of MORL approaches have been developed. Among these approaches, weighted sum approach, W-learning, ranking approach, and geometric approach are some of the single-policy approaches. The convex hull approach, varying parameter approach are some of the multi-policy approaches [13].

When designing a RS using RL, two key components are considered: the action space and the states. The action space refers to the set of items or activities that can be recommended to the user. On the other hand, the states encompass various factors such as user interests, user history, and any other relevant information that can assist in delivering more personalized and relevant recommendations. As a consequence of the inherent characteristics of RL algorithms, it is common for a RS constructed using this framework to be limited to suggesting a single action for a given condition. Several researchers have attempted to come up with solutions to this problem. Only two studies [14] [15] in the realm of RL for RSs have deeply explored this issue [5]. Slate-MDP[14] tries to tackle this by searching for a policy for each slot in the list separately. SlateQ [15] proposes to calculate the combination of the action set and consider each combination as an action.

In the context of offering slate recommendations, when considering a pool of 50 items and selecting 10 items for recommendation at each instance, it is estimated that there exist approximately 670 billion unique combinations of activity slates.

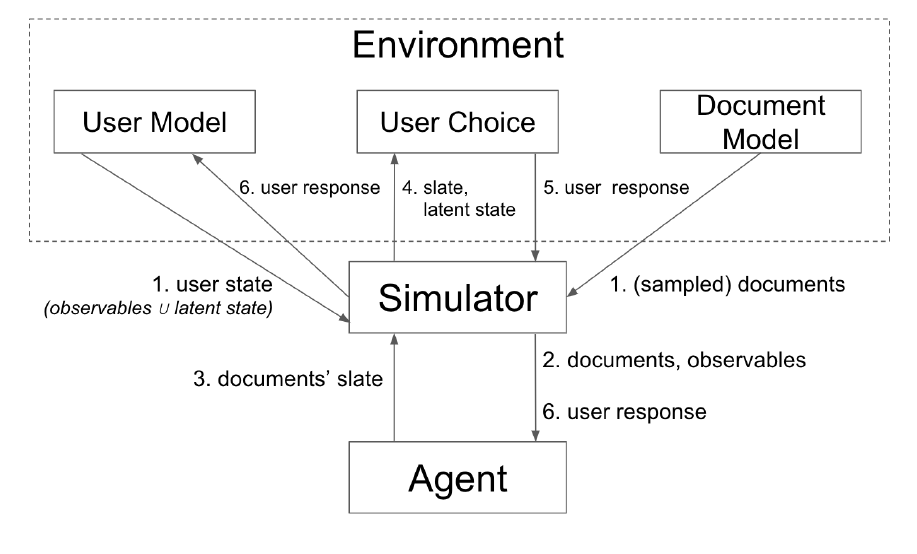


The expansive nature of the action space presents significant challenges in terms of manageability. SlateQ decomposes each of the items in those slates and considers Q values item wise.  By implementing this approach, there is a notable reduction in the overall size of the activity space. In order to analyze user choice behavior of items from slates that support the SLATEQ decomposition, two assumptions are made by SlateQ.

1. Single choice (SC) - A user consumes a single item from each slate
2. Reward/transition dependence on selection (RTDS) - The realized reward depends only on the item chosen from each slate.

This study focuses on the utilization of MORL techniques and Slate recommendation techniques, particularly SlateQ, in conjunction with DRL to create a scalable activity RS that aims to reduce stress and alleviate emotions while considering users' long-term engagement. Additional details regarding SlateQ's application in a RS will be discussed in the methodology section.

One key issue with establishing such online learning-based RSs is that they are extremely difficult to evaluate because there is no prior data to compare the system to. The only solution is to test the system with actual users and collect feedback, or to run simulations. However, such general-purpose simulation platforms specifically designed for sequential recommender setting are lacking. RecSim [16] is Google research project, in which they have developed a configurable simulation platform for RL recommender systems.



*Figure 1.3 : RecSim framework components*

Recsim consists primarily of two components:

1. Environment.
2. Agent.

Recsim Framework provides users with the ability to customize the environment and agent settings prior to conducting simulations and obtaining results. The configuration of three primary components of the environment is crucial. Those are ,

1. User Model – Responsible for configuring how user states change and how users react to recommendations.
2. User choice modals – Responsible for configuring how users make choices.
3. Document model – Responsible for sample documents (recommending items)

This research further discusses how RecSim framework was configured to run simulations to fine-tune the suggested RS and results were obtained.

## **Research Gap**

When developing a system to assist individuals in managing stress and enhancing their emotional well-being, it's crucial to consider various key factors. This section delves into the comparison of our proposed system with existing solutions in the field. It breaks down the specifics of the proposed system and evaluates how they stack up against previous approaches. This analysis aims to refine and enhance the system, ensuring that it not only aligns with existing options but also establishes itself as a superior choice for stress relief and emotional well-being improvement.

A work by Sriram et al. has developed a mobile application, "A Chatbot Mobile Quarantine App for Stress Relief" (Study A) to relieve stress for people who are in quarantine using music. They have used a camera to identify stress from facial features, followed by a questionnaire administered using a chatbot interface. In the event that the user does not respond as anticipated, the system will employ music as a means of alleviating stress [17]. One major difference between their stress-relieving technique and ours is that our solution works towards relieving stress and alleviating emotions in a personalized manner by learning about the user sequentially. The system we have developed offers stress-relieving activities that are tailored to individual user preferences and efficacy, considering long-term user engagement. In contrast, their approach delivers generalized recommendations that are not personalized for individual users. This personalized approach can potentially enhance the effectiveness of stress relief and emotion alleviation techniques, as such techniques can vary based on individual differences such as personality, lifestyle, and stressors. By taking these factors into account, our solution has the potential to provide more effective and tailored recommendations for stress relief and emotion alleviation.

According to the study conducted by Shin et al., they have developed a music RS, "Automatic stress-relieving music RS (RS) based on photoplethysmography-derived heart rate variability analysis," (Study B)  to relieve stress by identifying stress based on photoplethysmography-derived heart rate variability [18]. The music RS in this system has been created using a pre-trained machine learning model, which allows it to provide personalized music recommendations for every user of the system. However, pre-trained machine learning models do not adapt to changes in the environment like RL techniques do. To overcome this limitation, we developed our RS using RL techniques. Our system has the ability to adapt to environmental changes, resulting in more effective recommendations that are tailored to each individual's specific needs in specific scenarios. As a result, our system is more effective in helping users manage stress and alleviate emotions by providing personalized and adaptive recommendations.

In addition, the study by Chang et al. has led to the development of a personalized music RS based on electroencephalography feedback using collaborative filtering techniques (Study C) [17]. Their RS, powered by collaborative filtering techniques, functions by suggesting similar music to user groups with similar tastes. As they suggest, this approach can offer personalized recommendations to users. However, collaborative filtering has limitations in providing truly personalized recommendations. In cases where a user has unique interests that don't align with any existing user groups, they may not receive accurate recommendations. While such scenarios are uncommon, our RS addresses this possibility thanks to the way RL algorithms operate. Our RS has the ability to focus on individual preferences, ensuring that even outlier users receive relevant recommendations. Furthermore, our RS places a strong emphasis on long-term user engagement rather than solely relying on the user's group affiliation when making recommendations. This approach ensures a more comprehensive and personalized user experience.

Furthermore Anthony et al. have developed a music RS using content-based filtering techniques (Study D) [20]. They have shown that they were able to achieve cosine similarity results reaching up to 80% for songs and 50% similarity for artists. Content-based filtering techniques work by recommending similar items that the user likes. Although this technique works for some scenarios, in scenarios like ours, where recommending stress-relieving and emotion-alleviating activities is crucial, it is essential to formulate the recommendation problem as a sequential decision-making problem. This is because the user needs to progressively improve their coping skills for stress and emotional well-being, which cannot be achieved with content-based filtering alone. Our system not only provides activities similar to the user's preferences but also considers the necessity for recommendations to occur in a sequential decision-making manner.

A work by Shadini et al. has developed a social media application called "xīnlĭ The Social Media App to Replenish Mental Health with the Aid of an Egocentric Network" (Study E) to identify a person's emotional state and provide activity recommendations to alleviate negative emotions [21]. The RS they used is a personalized RS developed with Q-learning, a RL technique. Our RS is an upgraded version of this system, which improves their system architecture by using DRL techniques. Our RS uses a DQN to approximate the action-value function, which makes it more scalable than their solution. One noteworthy improvement is that our system also recommends a collection of activities (slate recommendations) for a given state rather than just one activity. This is an open research area, as discussed in the literature survey section, because current RL algorithms cannot handle this problem [22]. Also, our proposed system is designed to provide activity recommendations for both stress relief and negative emotion alleviation purposes; hence, it has two objectives. Therefore, we used MORL techniques, which are discussed in the methodology section.

The table below compares the five previously mentioned existing solutions to our proposed system in detail.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Study A | Study B | Study C | Study D | Study E | Proposed solution |
| RS adapts to environmental changes. | No | No | No | No | Yes | Yes. |
| Recommendation problem is formulated as a sequential decision-making problem | No | No | No | No | Yes | Yes |
| Recommends similar items (activities). | No | No | No | Yes | Yes | Yes |
| Recommends according to similar user groups. | No | No | Yes | No | No | Yes |
| RS learns in an online learning approach. | No | No | No | No | Yes | Yes |
| Provide slate recommendations | No | No | No | No | No | Yes |
| Highly scalable | No | No | Yes | No | No | Yes |

*Table 1.1 : Research Gap based on previous research vs conducted research.*

## **Research Problem**

As discussed in the background section, IT professionals suffer a lot from stress due to a high workload and sitting in front of computers all day [23]. Hence, it is necessary to come up with techniques to manage stress and emotional wellbeing. As described, the motivation of this research is to come up with a solution to stress management and emotion alleviation for IT professionals. In order to help IT professionals manage their stress and emotional wellbeing, firstly it is necessary to come up with a way to measure their stress levels and emotions while they work without impacting their day-to-day work. Then the problem left is to find a way to help them reduce their measured stress levels and alleviate emotions.

Many available solutions [24] [25] [26] [27], designed to identify stress and emotions and manage them, including the research discussed in the research gap section, are mainly focused on general purpose usage. In our research, we scoped this down to only IT professionals’ usages. Hence, there are many aspects to be considered when specializing like that. Whatever solution we develop, it has to work without interrupting the user’s day-to-day work.

Therefore, the main research problem of our research can be stated as: How to detect users stress levels and emotions without interrupting or impacting the user’s day-to-day work by utilizing the simple equipment they use every day and assist in reliving the detected stress levels and alleviating the detected emotion?

This report is mainly focused on finding out how to assist in reliving detected stress levels and alleviating the detected emotions. The proposed answer is to develop computer-based stress-relieving and emotion-alleviation activities and recommend them to users of our solution.

In the research gap section, it is discussed what the available RSs are and the differences between them and our proposed RS. Summing up everything discussed in the research gap, the following research questions need to be answered: How these questions are addressed is discussed in the objectives and methodology section.

1. What sort of RS is suitable in this context?
2. How to adapt recommendations as environmental factors change?
3. How to develop a scalable RS?
4. Is slate recommendation important for the context?
5. How to do both stress relief and emotion alleviation simultaneously?
6. How to evaluate the RS?

# RESEARCH OBJECTIVES

## **2.1 Main Objectives**

The main objective of the research project is to present a novel approach to address the research problem which is the lack of Stress relieving and emotion alleviation solution specifically focused for IT professionals, which will not interrupt their day-to-day work. The main objective is divided into 4 parts and addressed separately as separate research.

Such that main objectives of the research project include:

1. Implement a way of identifying stress levels of users via key board dynamics.
2. Implement a way of identifying stress levels of users via a mouse equipped with an IOT device which track heart rate variables.
3. Implement a way of identifying emotions of users via the webcam by considering user facial features.
4. Build a RS which has the capability to recommend stress relieving and emotion alleviation activities to users.

## **2.2 Specific Objectives**

This report focuses on the objective of building the stress relieving and emotion alleviation RS. This objective is divided into 2 main parts.

* + - 1. **Developing the cross-platform desktop application and the stress relieving and negative emotion alleviation activity pool / list.**

Develop a cross-platform desktop application: To ensure the stress management system is accessible to a wide range of users, a cross-platform desktop application had to be developed. This objective aimed to provide a user-friendly interface that is accessible to a wide range of users.

Develop a pool of stress-relieving and negative emotions alleviating computer-based activities: To ensure that the stress management system provides effective stress-relieving activities, a pool of computer-based activities has been developed. This objective aims to provide a wide range of activities capable of relieving stress for different stress levels and different individuals. Online forms were used to collect data regarding activities, and our external supervisor, who is a psychologist, had to be consulted to categorize activities and create an activity pool that would relieve stress and alleviate negative emotions.

* + - 1. **Developing the Recommendation System**
* In the context of stress management and emotional well being of IT professionals, it is important to consider the fact that user interaction with the system and user progress towards managing stress and emotions. Thus, it is more suitable to formulate the recommendation problem as a sequential decision-making problem, rather considering it as a prediction or a classification problem. Thus, the very first objective can be stated as formulating the recommendation problem as a sequential decision-making problem and implement the RS using RL techniques.
* RL based RSs learn everyday as user interact with the system which makes the RS capable of changing according to user changes and other environmental factor. Thus, to address the research problem of how make the RS adaptable to environmental changes, RL techniques can be used.
* Conventional RL algorithms are not able to scale well when the state-action space is large. To make it scalable function approximation methods can be applies with RL. Hence to address the third research problem of making a scalable RS, using function approximation techniques such as DQN can be stated as third objective.
* Without forcing the user to do an activity we provide, it is important to let the user choose whatever the activity from recommendation slate for better user satisfaction and engagement. SlateQ is an RL based algorithm designed to use with RL based RSs which have the need to providing slate recommendation. Hence applying SlateQ in our RS can be stated as the next objective.
* DevRelax app is designed with the purpose of both relieving stress and alleviate emotions. So it is important to consider both the objectives for recommendations. MORL can optimize multiple objectives simultaneously. Hence incorporating MORL techniques with our RS can be stated as the next objective.
* Evaluating online learning-based RSs such as ours is a complicated task because it is not easy to know what the user would do without real user interactions. RecSim is a configurable simulation platform specifically designed for RL based RSs. Hence using this platform and evaluate the implemented RS can be considered as the next objective.

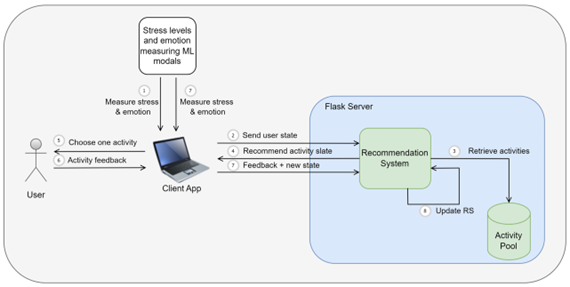
# METHODOLOGY

This section comprises three sub-sections that explain the methodology of the entire system. These sub-sections encompass the system diagram, commercialization facets of the product, and the testing and implementation of the product.

## **Methodology**

This section presents the methodology for stress relieving and emotion alleviation activity RS.

DevRelax desktop application has three machine learning based components to identify stress levels and emotions. We implemented a way of identifying stress levels by analyzing keyboard dynamics, which enables our app to identify stress levels while the user types using his/her keyboard. We also developed a mouse which incorporates an IOT device to measure the user's heart rate. Then through that identify stress levels which enables our application to identify stress levels while using the mouse. We implemented an ML solution to identify emotions using the webcam by looking at the user’s face. In order for artificial intelligence to make progress in understanding the world around us, it needs to be able to interpret multimodal signals together [28]. Then we implemented a way of aggregating the results of all these modalities and output aggregated stress levels and emotions which are taken as input for the stress relieving and emotion alleviation activity RS discussed in this report.



*Figure 3.1 : Recommendation system workflow*

As previously indicated, the initial stage of our application involves the assessment of the user's stress levels and emotions. The study aims to capture stress levels and emotions in a near real-time manner through the analysis of one-minute cycles. The three machine learning modalities employed for this task operate seamlessly in the background, ensuring uninterrupted user engagement with their computer during day-to-day activities. In order for the DevRelax application to generate a notification prompting the user to engage in a recommended activity, it is necessary for the user's negative emotions and elevated stress levels to be detected.

If the user clicks okay and proceeds with the app’s recommendations, the app will send the user state, which consists of the user’s emotion, stress level, past interaction data with the application, time, screen time, and other user details, to the RS, which is shown in the second step of the above diagram.

Based on the user state, the RS would look at the activity pool and recommend a slate of suitable activities to the user, which is shown in the third and fourth steps. A slate consists of four activities. How these recommendations happen, how these activities are chosen, the algorithm, and all the technical details are discussed in the implementation section.

Then the user would choose one activity from the activity slate and attempt that activity, which is shown in the fifth step. Each activity can be completed in no more than 20 minutes. After completion of the activity, stress levels and emotions are again measured. Also, three types of scores namely, satisfaction score, completion score, and effectiveness scores are calculated based on how the user attempted that activity and how stress levels and emotions changed. These scores and the updated user state are then sent to the RS again to let the RS evaluate itself, to know if the recommendations that it provided are any good or not, and to update the recommendation settings, which are shown in the seventh and eighth steps. All of these technical specifics are discussed in the section on implementation.

### 3.1.1. Main Task

The primary objective of this research is to develop the DevRelax desktop application which is capable of identifying stress levels and emoting of users accurately while they work on their computers without causing any disruptions. Additionally, the application seeks to mitigate high stress levels and negative emotions by offering computer-based activity recommendations, allowing users to engage in these activities.

### 3.1.2. Sub Tasks

As discussed in the research gap, research problem and objectives sections, in the light of RS, the sub tasks of the DevRelax desktop application are listed below,

* Conduct surveys and analyze the findings.
* Analyze literature findings.
* Identifying the computer-based activities suitable for the context and categorize them.
* Formulate recommendation problem as a sequential decision-making problem, thus, apply RL techniques.
* Use DRL techniques to achieve higher scalability.
* Using MORL techniques to balance both the stress relieving and emotion alleviation objectives.
* Develop the RS with SlateQ to achieve slate recommendation capability with high performance.
* Develop a simulation environment with RecSim to evaluate the RS.
* Develop the backend and frontend logics of the application related to the RS.

### 3.1.3. Data Collection

The proposed RS was developed using an online learning approach, meaning that the training process would occur while the system is in use [29]. Therefore, a prior dataset was not used to train the machine learning algorithm. (However, a simulation was run to train and evaluate the algorithm, which will be discussed further.) In order to create a

stress-relieving and negative-emotion coping activity pool, there were three primary methods of data collection:

1. Gathered activity preferences from potential users via Google Form - an online questionnaire.

2. Collected emotion alleviation activities dataset from researchers who had conducted the 'Xilini' [21] research.

3. Created an activity pool with the consultation of our external supervisor, who is a clinical psychologist.

By utilizing these methods, we were able to effectively gather the necessary data to create a comprehensive activity pool that will help users cope with stress and negative emotions.

### 3.1.4. Tools and Technologies

The table below shows the tools and technologies used to develop the RS of the DevRelax application.

|  |  |
| --- | --- |
| **Description** | **Tools and Technologies** |
| Programming IDE | Visual Studio Code |
| Programming language for desktop application development | ReactJs, ElectronJs |
| Machine learning algorithm-based Recommendation Engine | Python language |
| Programming language for backend development | Node JS, Flask |
| Database for store data | Mongo DB |
| Hosting the API | Flask and AWS |
| Version Controlling | Gitlab |
| Team connectivity | Teams and WhatsApp |

*Table 3.2 :Tools and Technologies used.*

## **Commercialization Aspects of the Application**

Target Market: The focus is on targeting corporate professionals, especially those in the IT sector, who are known to experience high levels of stress. The application should be marketed to both IT companies and individual IT professionals. The plan is to expand the target audience to college students in the near future since they are another demographic with high stress levels.

Revenue streams: The application will be offered as a subscription-based service, with different pricing tiers based on the number of users and functionality. Additionally, the plan is to offer corporate partnerships that provide access to the application for all employees at a discounted rate. To generate more revenue, the application will include in-app advertising and partnerships with stress-reducing product companies.

The commercialization of the application done in 5 phases. Currently we are at the phase 1. The following are the listed phases.

PHASE 01: The first step is to launch the first version of the software and test it by installing it at an IT services company. This is to obtain feedback and reviews about the product.

PHASE 02: Launch a free version of the software with limited activities, and a professional version with all features unlocked, which will require a subscription. The target clients are IT/Software companies that will benefit from the advanced capabilities of the professional version.

PHASE 03: The plan is to engage in targeted marketing campaigns through online advertising, social media platforms, and industry conferences to reach potential clients. Additionally, the company will collaborate with HR departments and employee wellness programs to promote the software as a tool for stress management and employee well-being.

PHASE 04: Continuously gather feedback from clients and use it to update and improve the software to maintain customer satisfaction, retention and attract new clients through positive word-of-mouth recommendations.

PHASE 05: The company plans to explore potential partnerships with health insurance providers to offer the software as a wellness benefit to their clients. This will create a new revenue stream and increase the accessibility of the software to a wider audience.

## **Implementation and Testing**

This section is divided into two parts, namely, implementation and evaluation. The implementation section focuses on discussing the details of developing the RS and integrating it with the desktop application. The evaluation section focuses on discussing the details of training and evaluating the RS in a simulation environment.

### Implementation

Before diving into the implementation of our solution, it was essential to clearly define what the users needed, what functions the system must perform, and what other requirements must be met. The following are the identified requirements.

1. User Requirements
2. Computer based activities that are most appropriate for users should be recommended.
3. Users should be able to click on the activity and attempt to do it.
4. Users should be able to provide feedback after attempting an activity.
5. User should be recommended 4 activities at a time.
6. After engaging in activities, users should be able to track their progress.
7. Functional Requirements
   1. The system should be able to identify emotions based on the measured inputs.
   2. The system should provide personalized recommendations for stress-relieving activities based on the identified stress and emotion levels.
   3. The system should be able to track user progress and adjust recommendations accordingly.
   4. The system should be able to provide real-time feedback to the user on their stress levels and progress.
   5. The system should allow the user to provide feedback on the recommended activities and adjust the recommendations accordingly.
8. Non-Functional Requirements
   1. The system should have high accuracy and reliability in measuring stress levels and identifying emotions.
   2. The system should have a low response time to provide recommendations.
   3. The system should maintain user privacy and confidentiality of their data.
   4. The system should be compatible with different operating systems.
   5. The system should have a secure and stable infrastructure to prevent data breaches and system failures.
   6. The RS should be highly scalable.

After considering the previously mentioned requirements, we decided to create a desktop application that employs innovative methods to detect stress levels and emotions. It also provides recommendations for stress relief and emotion management activities without disrupting the user's daily work routine. To kickstart the development process, I divided the workload into two key areas.

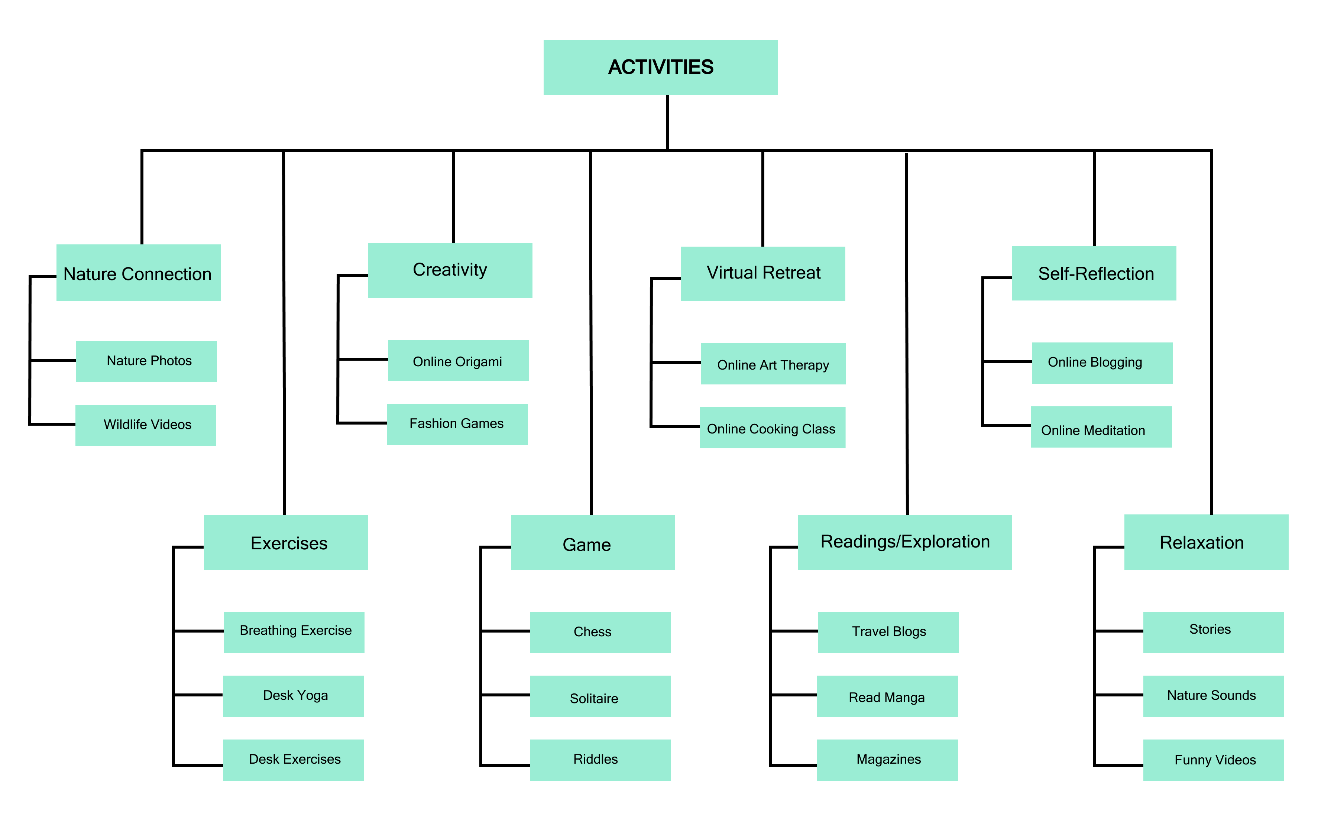
Recommendation System Development

Desktop Application Development

**Recommendation System Development**

1. **Activity Pool Development**

This section provides a description of the process by which the activities were selected and created. When contemplating stress reduction and emotional alleviation activities, it is essential to acknowledge that these activities are intended for IT professionals. Furthermore, it is essential that these activities can be performed while seated in front of a computer. A study [21] conducted by Kalansooriya et. al., had created a dataset of emotion alleviation activities. We collected these data from them and filtered out the most suitable activities for our context. In order to address the issue of stress relief, a list of computer-based activities was developed with the assistance of our external supervisor, who possesses expertise in clinical psychology. A total of 20 activities were developed in 8 categories, in collaboration with our external supervisor, with the objective of relieving stress and enhancing emotional well-being.



*Figure 3.2 : Activity Categorization*

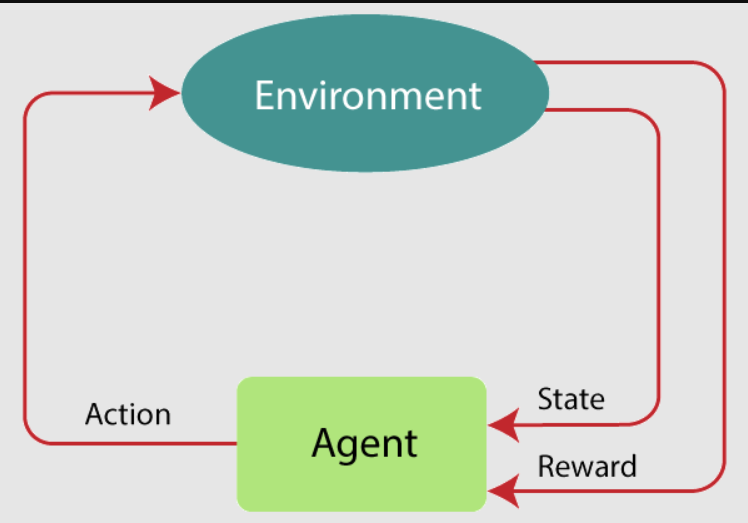
We looked up providers of such activities from the Internet and with their consent we integrated their products with our DevRelax desktop application.

1. **Recommendation Engine Development**

Once the construction of the activity pool was completed successfully, it became essential to set up the recommendation engine.

**Recommendation problem formulation to RL**

As discussed in the research gap and the objective section, our recommendation problem was formulated as a sequential decision-making problem rather than a classification or prediction problem. RL is the de facto approach in ML to address sequential decision-making problems.



*Figure 3.3 : Reinforcement Learning Workflow*

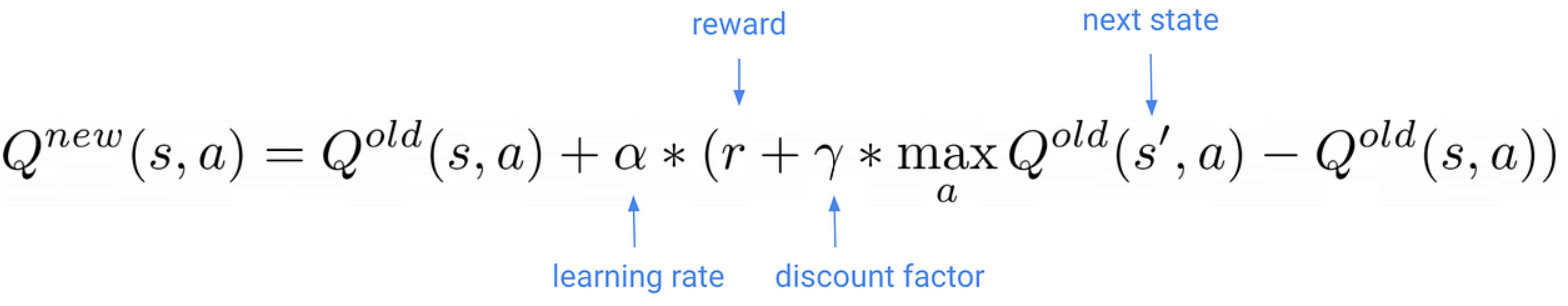
RL has two major components, namely the environment and agent. User interactions with the application, how the user's states are changing, and other environmental factors come under environment. The agent is the one who’s making the decision to take actions that would impact the environment. When the agent chooses action, the state coming from the environment changes. It will be given a reward based on how the state change occurred so that the agent can decide if the action agent chose to undertake was a good one or not.

Our recommendation problem was formulated to be addressed within the above framework. User states are the stress levels, emotions, and other user characteristics, which will be further discussed in the implementation section, and activity recommendations are actions. For the user’s state, the agent (recommendation algorithm) will choose an action or a slate of activities (which will be discussed in the implementation section), then the user will attempt the chosen activity and provide feedback and the user’s next state to the agent. Also, a reward is calculated based on user feedback.

**Recommendation Engine Implementation Attempt 1**

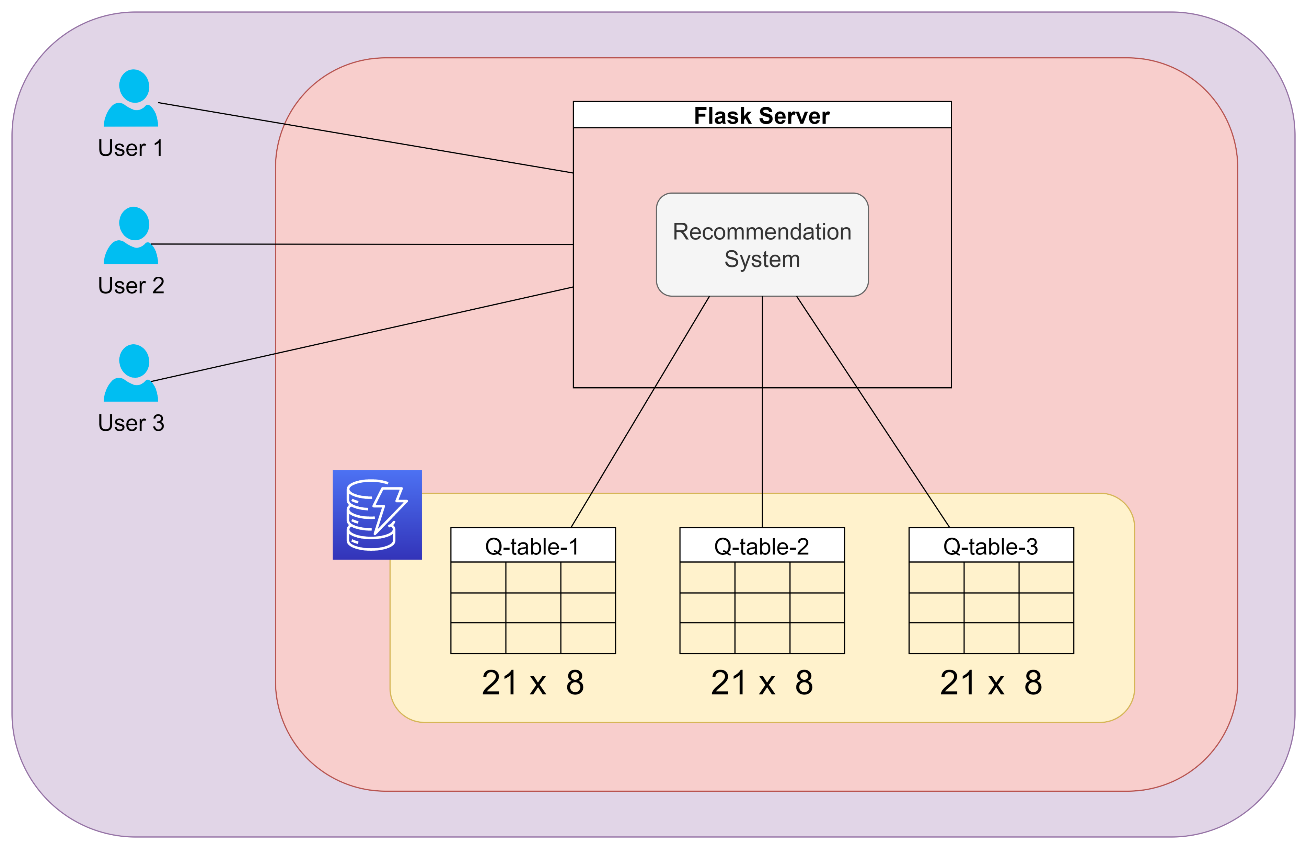
The implementation of the RS occurred in two attempts. This section will discuss the initial attempt and the reasons why a second attempt was necessary.

The recommendation agent was implemented using a tabular Q learning agent. Q learning is a model-free, off-policy learning technique that's particularly useful for problems where an agent needs to make a series of decisions to maximize a cumulative reward over time [30]. Q learning was chosen to develop our RS agent because it enables the system to learn optimal actions in an environment where the user's preferences and activity availability may change over time. By modeling the recommendation process as a Markov Decision Process (MDP), Q-learning can continuously update its recommendations based on user interactions, ensuring that it adapts to evolving user tastes and item popularity. This dynamic learning capability makes Q-learning a robust choice for RSs, providing personalized and relevant suggestions in real-time, even in small-scale scenarios where traditional methods may struggle to capture changing user behaviors [31]. The Q learning algorithm can be formulated as below,



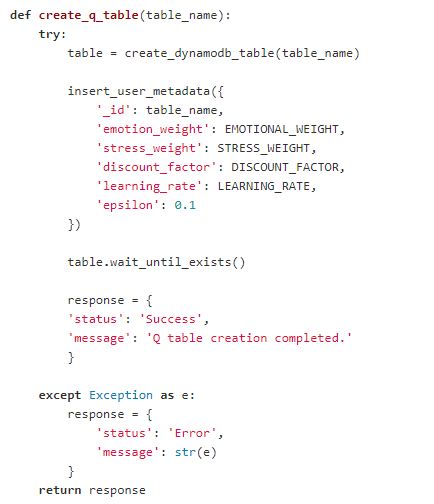
*Figure 3.4 : Q learning TD update algorithm*

The initial approach was to maintain one Q table per user. Research [21] done by Kalansooriya et al. also followed a similar approach to achieve personalization, which is discussed in the research gap section. Our RS was designed to create a new Q table in AWS DynamoDB when a new user registers for the system. Initially, there were 8 activity categories, which were considered 8 actions, and the state space was 21 in size. Hence, each Q table had a 168-sized state-action space. The business logic was implemented using Flask, a Python microweb server framework.

1

*Figure 3.5 : Q table overview of RS development attempt 1*

The business logic of Q table creation is implemented below. The create\_q\_table function is responsible for creating a new table in DynamoDB and updating user metadata in MongoDB. The initialize\_q\_table function is responsible for initializing the newly created Q table in DynamoDB with all zero values in each state action.



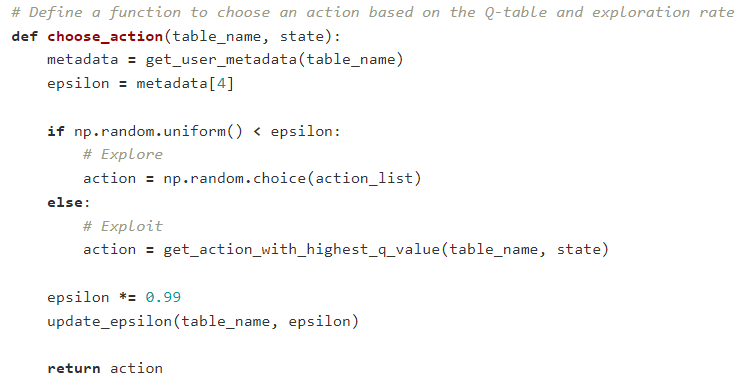
*Figure 3.6 : Create Q table function*

A computer screen shot of a code

Description automatically generated

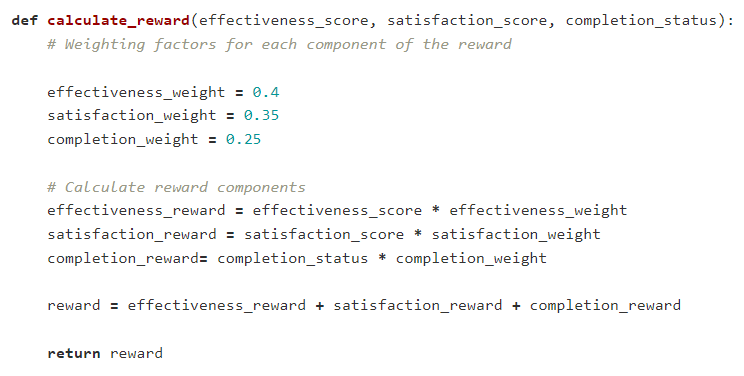
*Figure 3.7 : Initialize Q table function*

The choose\_action function is responsible for retrieving the current user’s epsilon value from the MongoDB database and choosing the activity or action with the highest Q value in the Q table in DynamoDB. The Epsilon value is used to control exploration and exploitation. When exploring, a random activity will be returned so that the user can try or explore that activity and provide feedback. When exploiting, an activity with the highest Q value, meaning the best activity suitable for that user, is chosen.



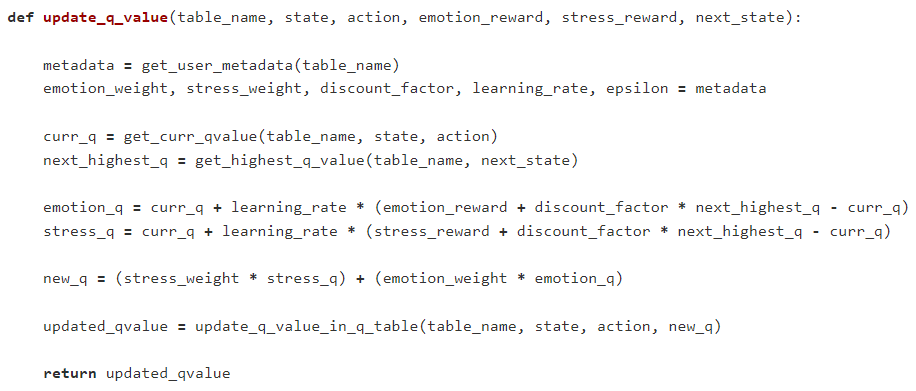
*Figure 3.8 : Choose action function*

The calculate\_reward function is responsible for calculating the reward after a user attempts an activity. This function has three parameters: effectiveness\_score, satisfaction\_score, and completion\_score, which are calculated from the frontend using user feedback information. A higher reward will be given for an activity if the user feedback is good, and likewise.



*Figure 3.9 : Calculate reward function*

The update\_q\_value function implements the Q learning algorithm. In this function, the Q learning Q value was calculated for both emotion alleviation and stress relieving objectives using a weighted sum, which is a single policy MORL approach [28].



*Figure 3.10 : Update Q value function*

Although this framework formulates the RS as a sequential decision-making problem and provides decent recommendations, one huge problem with this framework is that it does not scale and is very costly due to the fact that one new table has to be created when a new user registers for the system. Also, each new user has a newly created Q table that has no relation to other users’ Q tables. This creates a new problem named the cold start problem. This problem occurs in situations where the system has limited or no information about a new user or item, making it difficult to provide relevant recommendations. To overcome these problems, Recommendation Engine Implementation Attempt 2 was initiated, where I used DRL techniques.

**Recommendation Engine Implementation Attempt 2**

**Deep Reinforcement Learning**

To overcome the above-mentioned problems Recommendation Engine Implementation, Attempt 2, was initiated. As discussed in the literature review section, the behavior of conventional RL algorithms makes it difficult to scale applications like ours. To make it scalable, function approximation methods can be applied to approximate the action-value function. Neural networks are powerful machine learning models that can be used for various tasks, including function approximation [32]. Function approximation refers to the process of finding an approximate mapping from input data to output values. Neural networks excel at this task because they can model complex and nonlinear relationships between inputs and outputs.

A diagram of a network

Description automatically generated

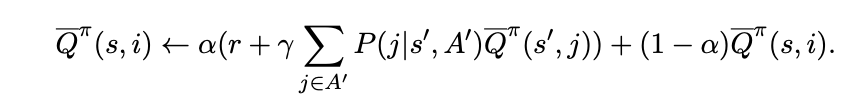
*Figure 3.11 : Neural network overview*

In DQN, a neural network is used to approximate the action-value function (Q-function) in RL. The Q-network takes the state as input and outputs the estimated expected future rewards for each possible action. This allows the agent to make decisions by selecting the action with the highest Q-value [33].

**Slate Recommendations**

However, with slate recommendation in mind, formulating our recommendation problem to be addressed with a conventional DQN is impractical. The DevRelax application is designed to recommend four activities per recommendation slate. We created an activity pool consisting of 20 activities. Meaning there are 4845 possible combinations of activity slates in total. When the activity count in the activity pool and the slate size increases, the activity slates combinations will also increase drastically, which cannot be managed at all as each slate has to be taken as action. To overcome this limitation, Google researchers have come up with a new algorithm named SlateQ [15] – which uses a decomposition technique to calculate Q values per activity rather than per slate.

TD Update for greedy policy improvement can be represented as below,



Where, α is the learning rate, r is immediate reward received after attempting activity a in state s, s' is next state, P(j|s',A') is probability of selecting activity j from next slate A’.

In our RS, the above TD Update algorithm was utilized to update Q values per action. Our RS has to adhere to specific requirements in order to use SlateQ such as,

Users can only pick at most one activity at a time from a slate of activities. This also means users can also choose none if they wanted.

When a user goes back to the recommendations for a second activity, it is treated as a separate action. The user's progress is updated as they attempt each activity, and this information is used to suggest a new activity slate.

* + - 1. User state transitions and the rewards are solely dependent on the activity that is chosen. Other activities in the slate don’t affect these values.
      2. The probability of a user selecting an activity is known for a given slate.

**Multi Objective Optimization**

Also, considering both the objectives of stress relief and emotion alleviation, some modifications had to be made to our agent. When there is more than one long-term objective to be optimized in RL, MORL approaches can be used. As discussed in the literature survey, MORL approaches can be divided into two categories: single policy approaches and multi-policy approaches. If the objectives are not conflicting with each other, a single policy can optimize both objectives simultaneously. The weighted sum approach is a single policy-based approach used in MORL when the weight of each objective is known [13]. DevRelax is designed to relieve stress and alleviate emotions; hence, the weight of each objective is similar. In the weighted sum approach, TQ (total Q) can be calculated by getting the weighted sum of the individually calculated Q value of each objective, as shown below.

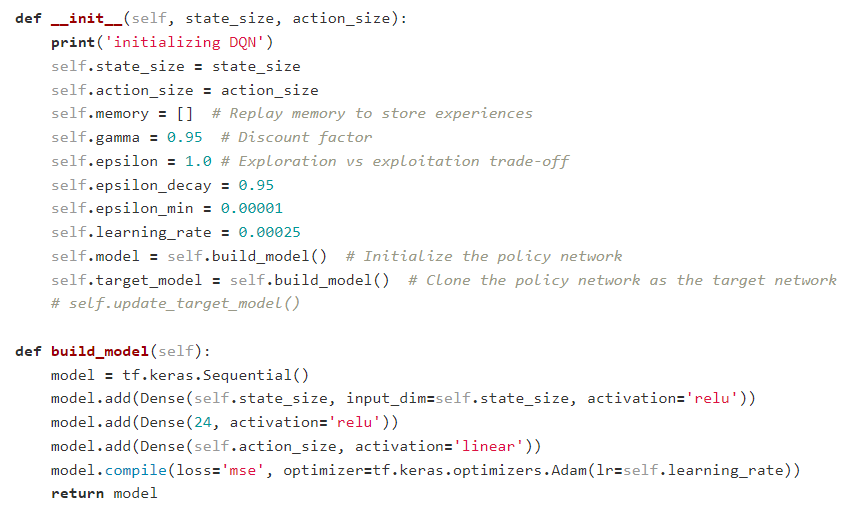
A mathematical equation with numbers and symbols

Description automatically generated

**Code Implementation of Agent**

Our RS agent was developed by combining the logics of SlateQ and MORL with DQN.

The first step of agent implementation is to develop and implement the logic to build a CNN, which will act as the DQN with hyperparameters.

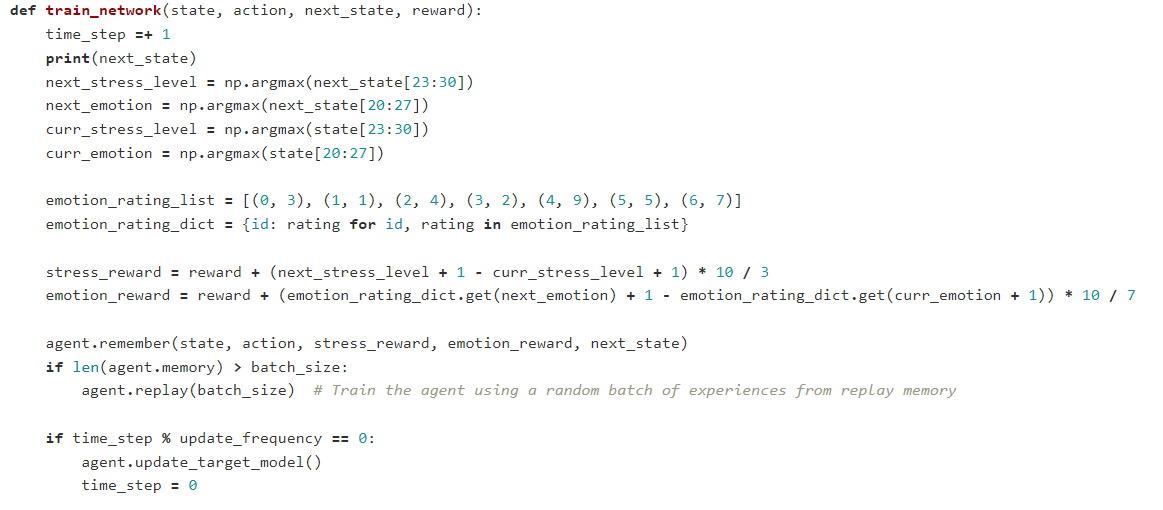


*Figure 3.12 : Neural network build model*

Our RS trains the network through an online learning approach. We implemented the logic of experience replies with a memory of 32. Experience replay is a critical technique that enhances the training efficiency and stability of the DQN. By storing past experiences in a replay buffer and randomly sampling from it during training, it helps break the temporal correlation between consecutive experiences, reducing the risk of learning from noisy or biased data. This prevents the model from overfitting to recent experiences, leading to more robust and accelerated learning, making it a fundamental component of DQNs [34]. Usually, replay memory stores state, action, reward, and next state. However, since we are using MORL, rewards calculated for each objective are also stored in the replay memory.

The training process of DQN involves the use of two neural networks: the policy network and the target network. The policy network approximates the Q-values for each possible action based on the input states. The target network is used to find the highest Q-value for the next state. The SlateQ algorithm is used to calculate the Q values for both stress relieving and emotion alleviation objectives and uses the weight sum MORL approach to get the total Q (TQ). During training, the policy network's weights are updated based on the difference between the predicted Q-values and the target Q-values. The loss function used to quantify this error is typically the Mean Squared Error (MSE) loss,

In this formula, Loss represents the overall loss value, N is the total number of training examples in a batch, Q\_target is the target Q-value obtained from the target network, and Q\_predicted is the predicted Q value obtained from the policy network. The logic of storing experiences in replay memory and the logic of training the network are implemented as below.



*Figure 3.13 : Train network function*

A computer code on a white background

Description automatically generated

*Figure 3.14 : Replay function*

A close-up of a computer code

Description automatically generated

*Figure 3.15 : Update target model function*

The logic for retrieving the list activities with their Q values is implemented as below. Then the activities with the 4 highest Q values are selected as the recommendation slate.

A computer code with black text

Description automatically generated

*Figure 3.16 : act function*

**Application – Recommendation Agent Communication**

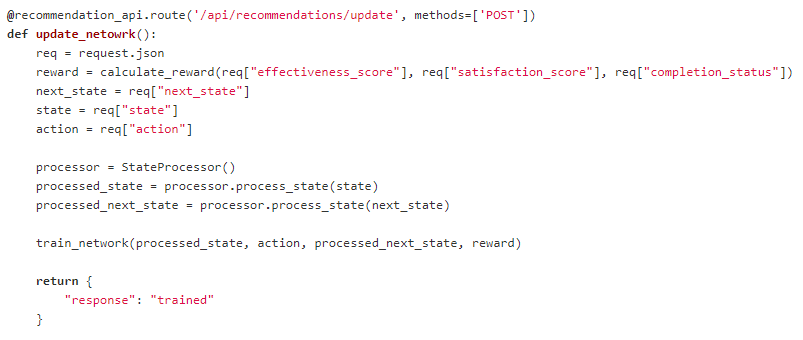
The implementation of the RS logic was done inside a Flask server. We implemented two APIs to communicate the desktop application with the RS on the Flask server.

The first API is to get recommendation slates when the current state is passed.



*Figure 3.17 : Get recommendations API*

The second API is to update RS when the previous state, next state, and user feedback are passed.



*Figure 3.18 :Update network API*

When the user initially loads the activities tab in the desktop application, the first API is called to get the activity recommendation slate for the current state. After the user chooses and attempts an activity, the user will give feedback on that activity, and user states will be changed. User feedback is sent as a completion score, satisfaction score, and effectiveness score. These scores will be sent to the RS along with the updates the new user has stated using the second API.

### Testing

**Evaluation with a simulation environment**

Before deploying the RS in production, where real users will use our RL, it is of utmost importance to evaluate it. One major problem with this is the lack of general-purpose simulation platforms available for sequential recommender settings. To address this, Google has developed RecSim, a configurable simulation platform for RL recommender systems.

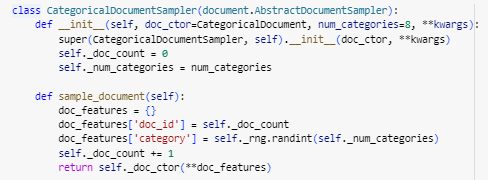
To evaluate the performance of our RS, we utilized RecSim. The framework facilitated the simulation of user interactions and enabled us to measure the system's effectiveness in providing recommendations. We configured the environment using RecSim's provided classes, implementing a document model, a user model, and a user choice model for the recommended slate to simulate the real-world environment and user behaviors. We employed the "slate\_decomp\_q\_agent," which implements the SlateQ algorithm, to simulate our RS's agent and allow it to interact with the created environment. Additionally, we used the "full\_slate\_q\_agent" provided with RecSim, which is a DQN without SlateQ’s decomposition, to interact with our environment as well, in order to compare and evaluate the differences.

Each simulated user was allocated a fixed budget. Each selected activity in the recommended slate reduces the time budget by 1. The experiment concludes when the time budget reaches 0. In our simulations, each agent chooses four documents (activities) to recommend from a corpus of 20 candidate documents. We executed both simulations in the Google Colab runtime environment and recorded the statistics using Tensorboard. These results are discussed further in the Results and Discussion section. Below is the implementation of the RecSim environment.

A screen shot of a computer code

Description automatically generated

*Figure 3.19 : Document model*



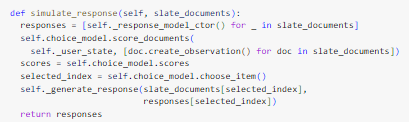
*Figure 3.20 : Document sampler*



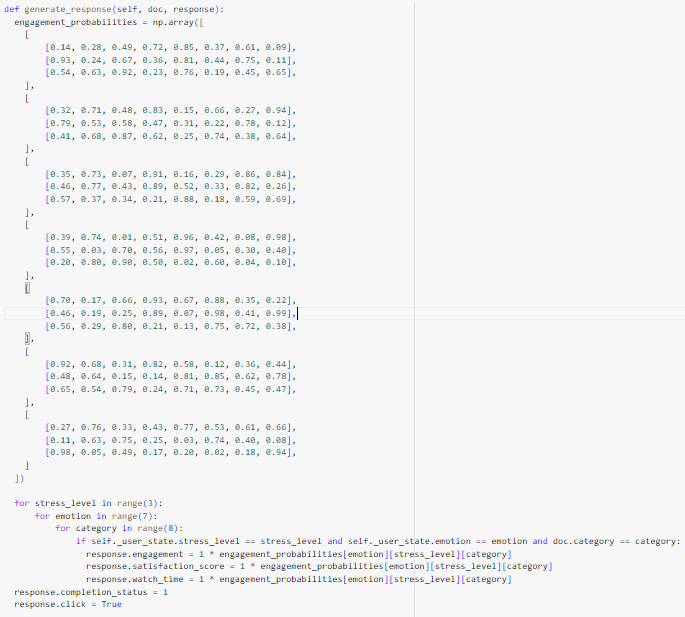
*Figure 3.21 : User model*



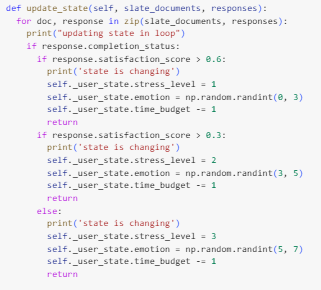
*Figure 3.22 : User response model*



*Figure 3.23 : Simulate user response function*



*Figure 3.24 : Generate response function*

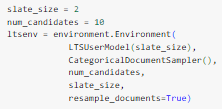


A close-up of a text

Description automatically generated

A screenshot of a computer program

Description automatically generated



*Figure 3.25 : State updating and user model logic*



*Figure 3.26 : Creating agents in RecSim*

The product is tested using a range of testing approaches, such as unit testing, integration testing, and user acceptability testing, to ensure that faults and bugs are detected early in the life cycle of the application.

The Figures show the test cases that were used to test the application at each interface.

A screenshot of a computer screen

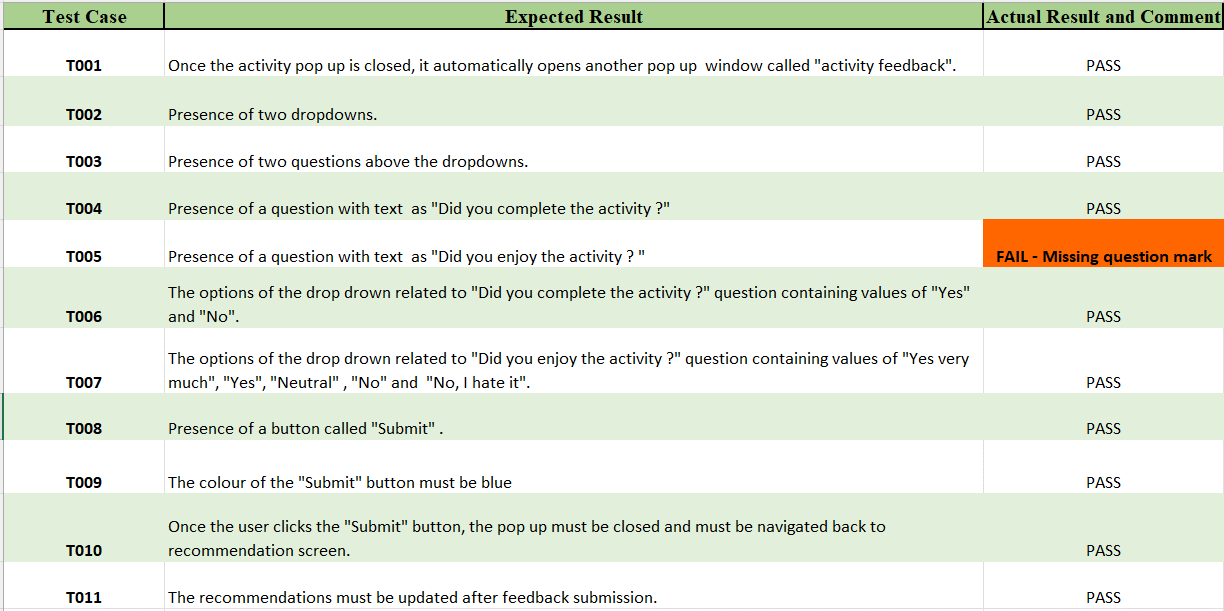
Description automatically generated

*Figure 3.27 : Test suite 1*

A screenshot of a computer

Description automatically generated

*Figure 3.28 : Test suite 2*



*Figure 3.29 : Test suite 3*

As mentioned earlier, the recommendation engine is developed inside a Flask server exposing 2 APIs which are used to communicate with our DevRelax client desktop application. These two APIs were tested locally using Postman.

A screenshot of a computer

Description automatically generated

*Figure 3.30: Get recommendation API test*

A screenshot of a computer

Description automatically generated

*Figure 3.31: Update recommendation API test*

# RESULTS AND DISCUSSION

This section will systematically provide the important aspects of the product's outcomes, research findings, and final discussion of the product with the appropriate diagrams.

## **Results**

As outlined in the testing section, a series of tests were conducted utilizing the developed environment in RecSim. These experiments involved the comparison of two agents: our proposed agent and a DQN agent that does not employ SlateQ decomposition techniques. The following findings have been obtained by the execution of summation calculations.

A graph on a screen

Description automatically generated

*Figure 4.1 : Average episode reward in our agent*

A graph with lines and numbers

Description automatically generated

*Figure 4.2 : Average episode reward in DQN agent*

The diagrams that you can see above illustrate the typical episodic rewards that were awarded onto our agent and the DQN agent that we used to generate the simulation environment. The use of an off policy and a greedy Q learning algorithm in the development of the DQN agent is what's responsible for the trend seen in the average episodic rewards graph. Because of the user's greedy strategy, which involved recommending things to the user that the user already desires, the incentive has been significantly raised. On the other hand, users eventually become bored with repeating the same activity, which is why the reward they receive for that activity gradually decreases over time.

In the meantime, our agent is being built with the concept of long-term engagement, which is why rewards are becoming enhanced in a modest but steady manner.

This tested model has been seamlessly integrated into the frontend to create the desktop application "DevRelax." Users can access and use the functionalities of this application through a user-friendly interface. To begin using DevRelax, users need to log in to the application. Once logged in, they can access the "Activities" tab, which directs them to the recommendation screen personalized specifically to their background.

A screenshot of a computer

Description automatically generated

*Figure 4.3 : Activities view*

Within the recommendation interface, users will be presented with a curated set of personalized recommendations that have been generated by the integrated model. The recommendations provided have been carefully selected so as to be consistent with the user's historical interactions with the application. In order to improve user engagement, an additional collection of activities can be accessed in the "More Activities" area. User can access an activity by a simple click on an activity tile.

After users have chosen an activity from the recommendations, a pop-up window will open, providing them with the opportunity to fully engage in the selected activity. The interface has been intentionally developed to optimize user experience, ensuring smooth and pleasurable interaction.

A screenshot of a web page

Description automatically generated

*Figure 4.4 : Attempt activity view*

After the activity has been completed, users have the option to close the pop-up window, which thereafter initiates a prompt for feedback. The feedback stage plays a critical role in the iterative process of improving the RS and boosting user satisfaction. The feedback form consists of two inquiries.

A screenshot of a computer

Description automatically generated

*Figure 4.5 : Activity feedback form*

1. "Did you complete the activity?"
   * Users can select from the following options: "Yes" or "No."
2. "Did you enjoy the activity?"
   * Users can choose from a range of responses: "Yes, very much," "Yes," "Neutral," "No," or "I hated it."

Once users have made their selections, they can proceed to submit the form. The integrated model utilizes these facts in order to adjust and enhance its recommendations for the user. Upon the user's activation of the submit button, the feedback pop-up ceases to be visible, and the user is subsequently sent to the recommendation page. At this point, the user receives the opportunity to browse an updated collection of recommendations. The approach of iteration not only offers customers individualized recommendations but also consistently enhances the model's comprehension of user preferences and participation.

## **Discussion**

One of the significant findings of this research was the recognition that novel recommendation settings can be effectively formulated as sequential decision-making problems. This approach proves to be particularly suitable in scenarios where user-system interaction is crucial, such as in the context of our RS. Unlike traditional collaborative filtering and content-based filtering techniques, which may not capture long-term user engagement effectively, this approach focuses on optimizing user engagement over time.

To address the scalability issues encountered with tabular RL algorithms, the research transitioned to DRL techniques. Deep RL allowed for the approximation of action-value functions, enabling more robust and scalable RSs. However, it was observed that deep RL approaches are data-hungry and require substantial amounts of data for training. This issue was somewhat mitigated by the shared network architecture across users in our RS, facilitating quicker learning as the user base grows.

It is crucial to note that the deep RL approach's effectiveness is contingent on the ability to train the network frequently. In scenarios where training cannot be performed regularly, the efficacy of this approach may be compromised. This consideration is important when planning for the long-term maintenance and usability of the RS.

Beyond the specific research findings related to RSs, this research underscored the importance of creating tech tools to support IT professionals in managing stress and emotions effectively. The "DevRelax" application not only assists users in discovering suitable activities but also helps prevent monotony by offering diverse recommendations. Moreover, the application's multi-modal support, including written, audio, and video guidance, enhances its utility in assisting users with daily tasks and providing a comprehensive wellness experience.

In conclusion, the research conducted on the "DevRelax" application has yielded valuable insights into the application of sequential decision-making and DRL for recommendation settings. It has also highlighted the significance of addressing challenges related to data requirements and training frequency. Furthermore, the application's potential to support IT professionals in managing stress and enhancing their overall well-being makes it a promising tool for the tech industry. These findings pave the way for further advancements in RSs and the development of technology-driven solutions to promote mental and emotional well-being among IT professionals.

# CONCLUSION

In conclusion, our team has made significant strides with the creation and implementation of DevRelax, a desktop application tailored to the needs of IT professionals seeking stress management and emotional well-being support while maintaining their productivity. A pivotal user requirement was to ensure the application seamlessly operates in the background, allowing users to focus on their daily tasks. To meet this requirement, we successfully integrated three machine learning-based components, providing nearly real-time assessments of stress levels and emotions. These assessments lay the foundation for our personalized recommendations aimed at relieving stress and addressing emotional well-being.

Our approach involved framing the recommendation problem as a sequential decision-making challenge and employing RL techniques. To tackle scalability issues, we harnessed DRL methods to approximate the action-value function. Considering DevRelax's dual objective of stress relief and emotion alleviation, we adopted a MORL approach. We also implemented the SlateQ algorithm to offer a slate of activity recommendations, enhancing the system's effectiveness.

Given the absence of historical user interaction data, we crafted a simulation environment using RecSim, a versatile simulation platform for RSs based on RL techniques. Through rigorous simulations, we were able to demonstrate that our developed agent outperformed a DQN agent, providing solid evidence of DevRelax's efficacy in assisting IT professionals in managing their emotional well-being seamlessly alongside their daily work tasks.

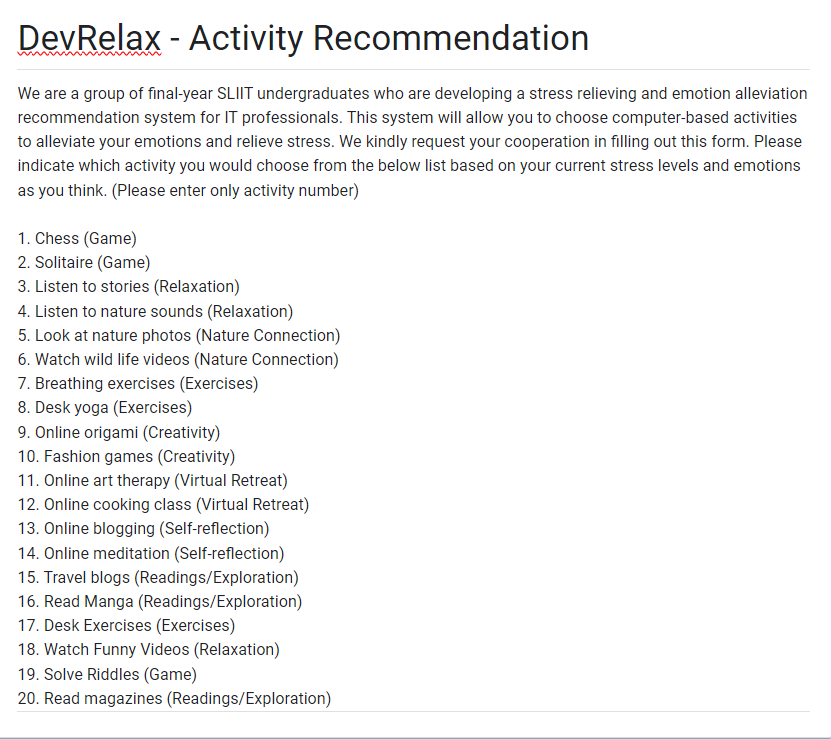
In summary, DevRelax stands as a groundbreaking solution, not only meeting the user's demand for uninterrupted operation but also leveraging state-of-the-art machine learning and RL techniques to deliver personalized and effective stress and emotion management recommendations. This project exemplifies the potential of AI-driven solutions to enhance the lives and well-being of IT professionals and serves as a testament to our team's dedication and expertise in this field.

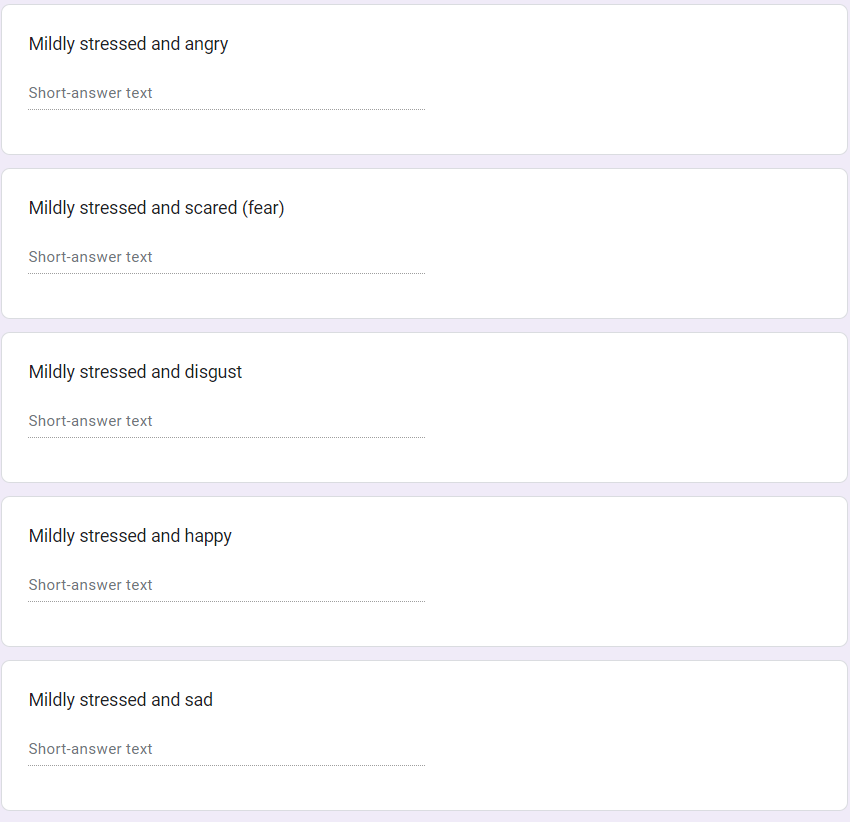
# REFERENCES

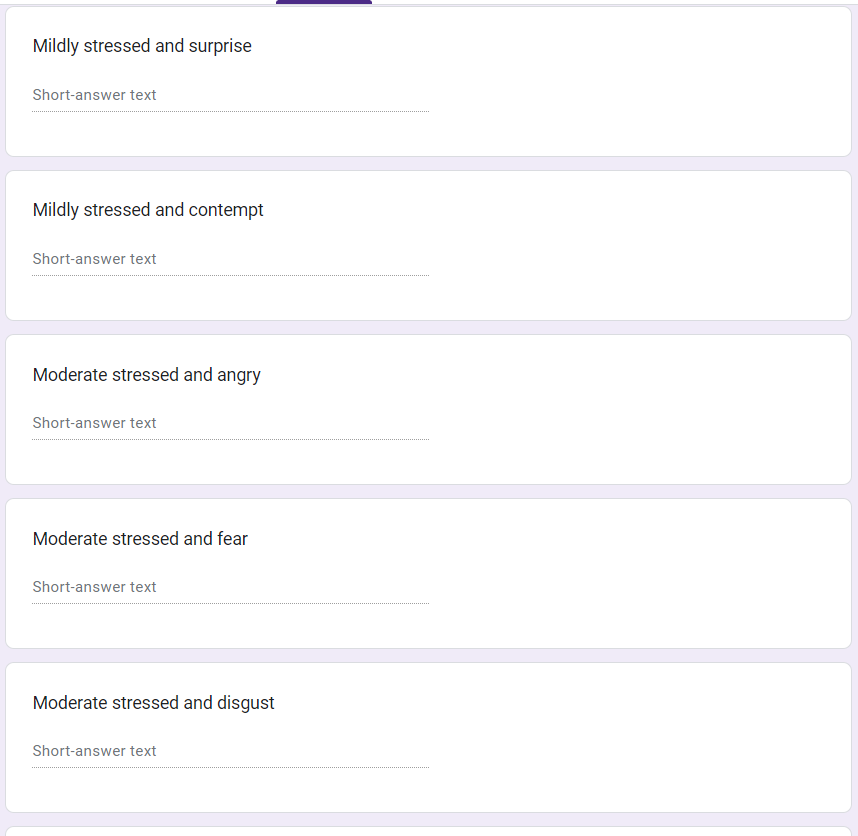
|  |  |
| --- | --- |
| [1] | S. Mochamad, L. Putri, A. Harefan and P. Ratyuhono, "The Effect of Role Conflict and Burnout Toward Turnover Intention at Software Industries, Work Stress as Moderating Variables," in *ICMEB*, 2019. |
| [2] | J. Sahni, "An assessment of employee perception regarding workplace Stress-causes and remedies: Evidence from Software Industry," *International Journal of Management, IT and Engineering,* vol. 6(7), pp. 23-39, 2016. |
| [3] | X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence 2019,* 2019. |
| [4] | M. J. Pazzani and D. Billsus, "Content-based recommendation systems," *The adaptive web: methods and strategies of web personalization,* pp. 325-341, 2007. |
| [5] | M. Asfar, C. Trafford and F. Behrouz, "Reinforcement learning based recommender systems: A survey," *ACM Computing Surveys,* vol. 55(7), pp. 1-38, 2022. |
| [6] | V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare and J. Pineau, "An introduction to deep reinforcement learning.," *Foundations and Trends® in Machine Learning,* Vols. 11(3-4), pp. 219-354, 2018. |
| [7] | G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie and Z. Li, "DRN: A deep reinforcement learning framework for news recommendation," *Proceedings of the 2018 world wide web conference,* pp. 167-176, 2018. |
| [8] | P. Wang, Y. Fan, L. Xia, W. X. Zhao, S. Niu and J. Huang, "A knowledge-guided reinforcement learning model for sequential recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020. |
| [9] | L. Zou, L. Xia, P. Du, Z. Zhang, T. Bai, W. Liu and D. Yin, "Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020. |
| [10] | C. Liu, X. Xu and D. Hu, "Multiobjective Reinforcement Learning: A Comprehensive Overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 45, pp. 385-398, 2015. |
| [11] | P. Vamplew, R. Dazeley and A. Berry, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Mach Learn,* p. 51–80, 2011. |
| [12] | D. M. Roijers, P. Vamplew, S. Whiteson and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research,* vol. 48, pp. 67-113, 2013. |
| [13] | C. Liu, X. Xu and D. Hu, "Multiobjective Reinforcement Learning: A Comprehensive Overview," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2015. |
| [14] | P. Sunehag, R. Evans, G. Dulac-Arnold, Y. Zwols, D. Visentin and B. Coppin, "Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions," *arXiv preprint,* 2015. |
| [15] | E. Ie, V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu and C. Boutilier, "SlateQ: A tractable decomposition for reinforcement learning with recommendation sets," 2019. |
| [16] | H. E. Ie, W. C., M. Mladenov, V. Jain, S. Narvekar, J. Wang and C. Boutilier, "Recsim: A configurable simulation platform for recommender systems," *arXiv preprint,* 2019. |
| [17] | S. Sriram, J. Manikandan, P. Hemalatha and G. L. Roselin, "A chatbot mobile quarantine app for stress relief," in *ICSCAN*, 2021. |
| [18] | I. Shin, C. Jaepyeong, L. Choonghee, L. Yup, Y. Hyung-Jin and K. Chan, "Automatic stress-relieving music recommendation system based on photoplethysmography-derived heart rate variability analysis," in *EMBC*, 2014. |
| [19] | H. Y. Chang, S. C. Huang and J. H. Wu, "A personalized music recommendation system based on electroencephalography feedback," *Multimedia Tools and Applications,* pp. 19523-19542, 2017. |
| [20] | J. T. Anthony, G. E. Christian, V. Evanlim, H. Lucky and D. Suhartono, "The Utilization of Content Based Filtering for Spotify Music Recommendation," in *ICIEE*, 2022. |
| [21] | S. Kalansooriya, A. Kaluarachchi, C. Weerawickrama, D. Nanayakkara, D. Kasthurirathna and D. Adeepa, "“xīnlĭ” The Social Media App to Replenish Mental Health with the Aid of an Egocentric Network," in *2022 IEEE 10th Region 10 Humanitarian Technology Conference*, 2022. |
| [22] | M. M. Asfar, T. Crumb and B. Far, "Reinforcement Learning based Recommender Systems: A Survey," *Association for Computing Machinery,* vol. 55, 2022. |
| [23] | K. Koneru, "A study on job stress among employees of software industries in Hyderabad," *Available at SSRN 3109468,* 2018. |
| [24] | D. L. Liong, S. M. Chit and C. L. Thong, "TimeAlone-The Stress Relieving Android App that Combines Diary and Music," in *International Conference on Human-Computer Interaction*, 2022. |
| [25] | M. McDaniel and M. Anwar, "Zen\_Space: a smartphone app for individually tailored stress management support for college students.," in *ICSH* , Hong Kong, 2017. |
| [26] | A. Rastogi, N. Shrivastav, A. Suryavanshi, P. Wadhwa, R. Kalantri and R. Shagufta, "Stress Reliving Application for Personal Wellbeing," in *International Conference on Internet of Things and Connected Technologies*, Singapore, 2022. |
| [27] | K. Chandrasiri, G. Chandrasena, A. De Silva, L. Jayasinghe, H. Dassanayake and O. Seneweera, "Mellow: Stress Management System For University Students In Sri Lanka," in *ICITR*, 2021 . |
| [28] | T. Baltrušaitis, C. Ahuja and P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 41, pp. 423-443, 2019. |
| [29] | S. C. Hoi, D. Sahoo, J. Lu and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing,* pp. 249-289, 2021. |
| [30] | J. Fan, Z. Wang, Y. Xie and Z. Yang, "A theoretical analysis of deep Q-learning," *Learning for dynamics and control,* pp. 486-489. |
| [31] | P. Rojanavasu, P. Srinil and O. Pinngern, "New recommendation system using reinforcement learning," *Special Issue of the Intl. J. Computer, the Internet and Management,* 2005. |
| [32] | S. Liang and R. Srikant, "Why deep neural networks for function approximation," *arXiv preprint,* 2016. |
| [33] | J. Fan, Z. Wang, Y. Xie and Z. Yang, "A theoretical analysis of deep Q-learning," *Learning for dynamics and control,* pp. 486-489, 2020. |
| [34] | S. Adam, L. Busoniu and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 42(2), pp. 201-212, 2011. |

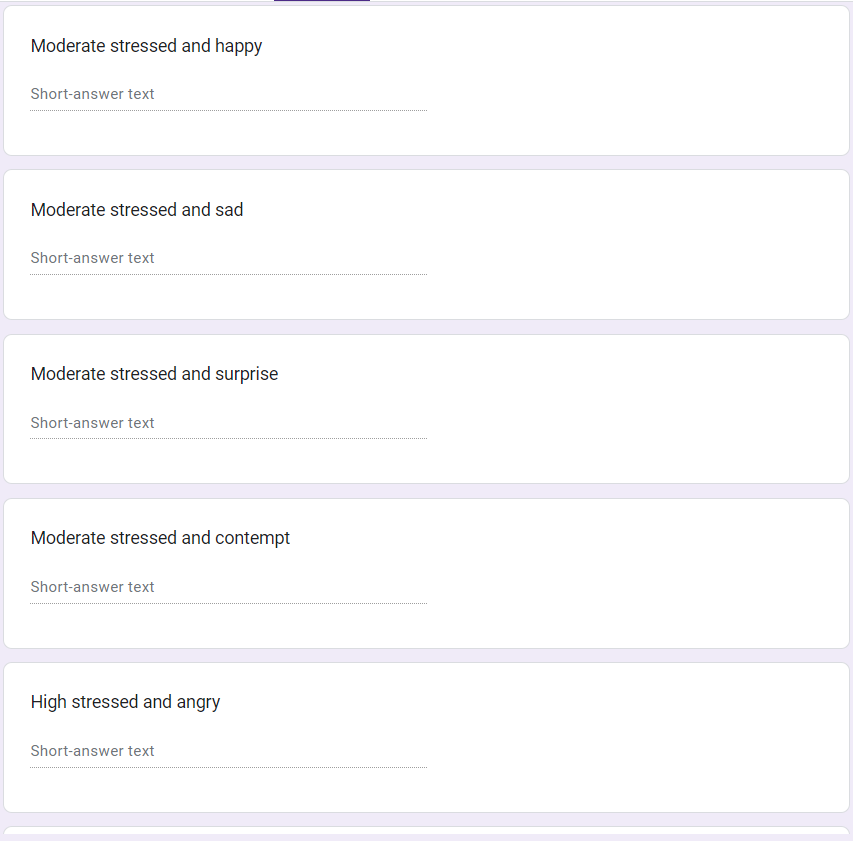
# APPENDICES

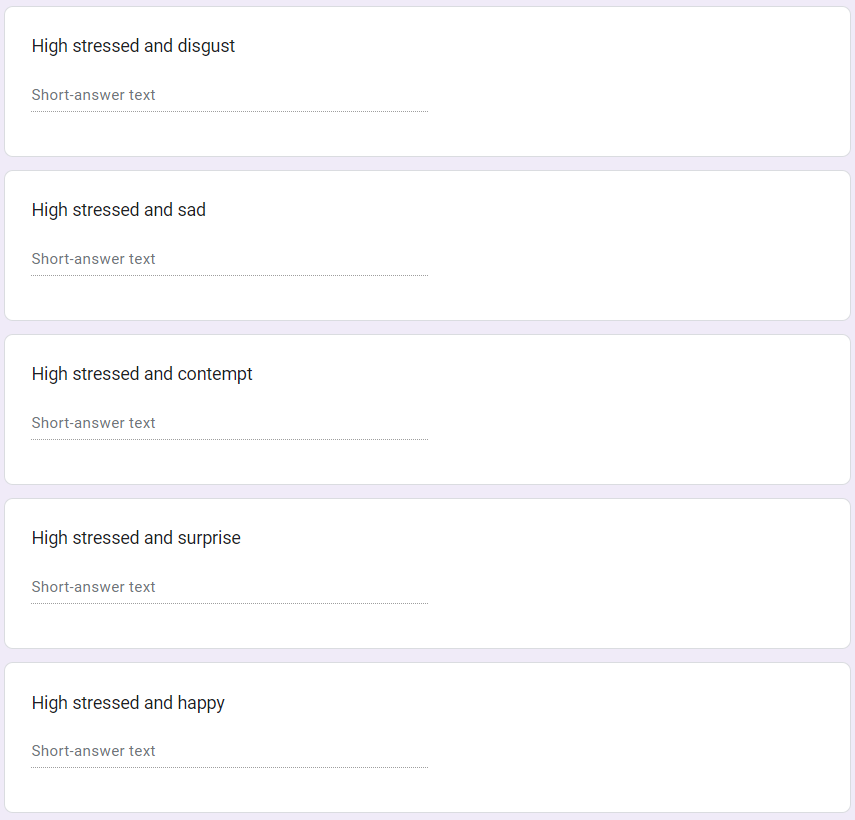
# Appendix A: Online Survey for user preferences on activities











# Appendix B: Plagiarism Report

