## “ DreadEase - Phobia Detection and Therapy using ML”

Mini Project submitted in partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

`

*Under the esteemed guidance of*

## M. Supriya

Associate Professor, CSE(AI&ML) Department

Submitted by

**B.REETHIKA 21R11A66A2**

**N.SRI DURGA 21R11A66C7**

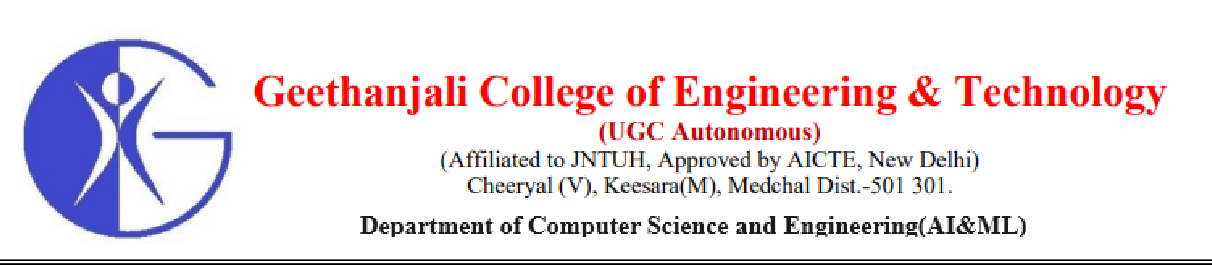


### Department of Computer Science and Engineering CSE(AI&ML) Accredited by NBA

**Geethanjali College of Engineering and Technology (UGC Autonomous)**

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi) Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

### 2024-2025



**CERTIFICATE**

This is to certify that the Mini Project Report entitled **“DreadEase - Phobia Detection and Therapy using ML”** is a bonafide work done and submitted by

#### REETHIKA 21R11A66A2 N.SRIDURGA 21R11A66C7

during the academic year **2023 - 2024**, in partial fulfilment of requirement for the award of Bachelor of Technology degree in **“Computer Science and Engineering (AI&ML)”** from Jawaharlal Nehru Technological University Hyderabad, is a bonafide record of work carried out by them under my guidance and supervision.

Certified further that to my best of the knowledge, the work in this dissertation has not been submitted to any other institution for the award of any degree or diploma.

**Project Guide Project Co-Ordinator**

**M.Supriya T. Kranthika**

**Associate Professor CSE(AI&ML) Associate Professor CSE(AI&ML)**

**Head of the Department CSE(AI&ML) External Examiner Dr. L. Venkateswarlu**

**DECLARATION**

We hereby declare that the Mini Project report entitled **“DreadEase - Phobia Detection and Therapy using ML”** is an original work done and submitted to **Computer Science and Engineering (AI&ML)** Department, **Geethanjali College of Engineering & Technology**, affiliated to Jawaharlal Nehru Technological University Hyderabad, in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering (AI&ML) and it is a record of bonafide project work carried out by us under the guidance of **M. Supriya**, Associate Professor, Department of Computer Science and Engineering (AI&ML).

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma.

Signature of the Student B.Reethika 21R11A66A2

Signature of the Student

1. Sri Durga

21R11A66C7

**ACKNOWLEDGEMENT**

The satisfaction of completing this project would be incomplete without mentioning our gratitude towards all the people who have supported us. Constant guidance and encouragement have been instrumental in the completion of this project.

First and foremost, we thank the Chairman, Principal, Vice Principal for availing infrastructural facilities to complete the major project in time.

We offer our sincere gratitude to our Project guide **M. Supriya**, Associate Professor, Computer Science and Engineering (AI&ML) Department, Geethanjali College of Engineering & Technology for her immense support, timely co- operation and valuable advice throughout the course of our Mini Project work.

We would like to thank the Project Co-Ordinator, **T.Kranthika**, Associate Professor, Computer Science and Engineering (AI&ML) for his valuable suggestions in implementing the project.

We would like to thank the Head of Department CSE(AI&ML), **Dr. L. Venkateswarlu**, for his meticulous care and cooperation throughout the project work.

#### 

#### 

# ABSTRACT

In the midst of escalating mental health concerns such as depression and panic attacks, this project provides a novel web application designed to provide critical assistance to persons dealing with these issues. Using modern machine learning where the program takes a novel strategy of tailored questioning to identify and categorize users' concerns. Using technology, the platform provides personalized daily instructions and advice for overcoming these phobias, suited to each user's specific needs and circumstances. Furthermore, the program recognizes the need of professional help in such trips, effortlessly connecting users with experienced psychiatrists and motivational speakers who provide important support and guidance. The process is based on the delicate interplay of prodding users with individualized questions, methodically assessing their replies, and classifying them according to certain phobias. The final outcome of this procedure is an ingenious tool that may help individuals confront and overcome their phobias, considerably improving their mental health. With the goal of bridging the gap between diagnostic tools and treatment resources, this application aims to transform mental health care by increasing its accessibility and effectiveness. Ultimately, the objective is to empower individuals by providing them with the resources and assistance they need to live fulfilling and resilient lives.

***Keywords: Mental health, Depression,Panic attacks,Web application,Machine learning,Natural language processing,Tailored questioning,Personalized daily instructions,Professional help,Psychiatrists,Motivational speakers,Individualized ,Overcome phobias.***

# LIST OF FIGURES

Fig. 1.1(a) Non – Pneumonia Chest X-ray 1

Fig. 1.1(b) Pneumonia Chest X-ray 1

[Fig. 1.2.1 Causes of Pneumonia 2](#_TOC_250026)

[Fig. 1.4.1 Symptoms of Pneumonia 4](#_TOC_250025)

[Fig. 4.3.1 Python logo 18](#_TOC_250024)

[Fig. 4.3.2 TensorFlow logo 18](#_TOC_250023)

[Fig. 4.3.3 scikit learn logo 19](#_TOC_250022)

[Fig. 4.3.4 SQLite logo 19](#_TOC_250021)

[Fig.4.3.5 VS Code logo 20](#_TOC_250020)

[Fig. 4.3.6 jupyter, colab logo 20](#_TOC_250019)

[Fig. 4.3.7 PyQt designer logo 21](#_TOC_250018)

[Fig. 5.1.1 Process Cycle 23](#_TOC_250017)

[Fig. 5.2.1 User Flow Diagram. 24](#_TOC_250016)

[Fig. 5.3.1 E R Diagram 25](#_TOC_250015)

Fig. 5.3.2 Schema in SQLinte3 25

Fig. 6.2.1 efficientNetB0 Architecture 27

[Fig. 6.2.2 Internal Process of Dense Net 27](#_TOC_250014)

[Fig. 6.3.1 Model Architecture 28](#_TOC_250013)

[Fig. 6.4.1 Internal Implementation 29](#_TOC_250012)

[Fig. 7.1.1 process and training of pre-trained model 32](#_TOC_250011)

Fig. 7.2.1 Training for severity prediction 33

[Fig. 7.2.2 Training for Disease prediction 33](#_TOC_250010)

Fig. 7.2.3 Training for medicine recommendations 33

Fig. 7.3.1 find accuracy for severity prediction 34

[Fig. 7.3.2 Best epoch value for Disease prediction 35](#_TOC_250009)

[Fig. 7.3.3 confusion matrix 35](#_TOC_250008)

[Fig. 7.4.1 Evaluation results 36](#_TOC_250007)

[Fig. 10.1 GUI Home Page 43](#_TOC_250006)

[Fig. 10.2 Login Page 44](#_TOC_250005)

Fig. 10.3 Signup Page 45

[Fig. 10.4 Options Page 46](#_TOC_250004)

[Fig. 10.5 Severity Prediction Page 47](#_TOC_250003)

Fig. 10.6 Disease Prediction Pahe 48

[Fig. 10.7 Recommendations Page 49](#_TOC_250002)

TABLE OF CONTENTS

Certificate Page Declaration Acknowledgement

Abstract i

List of Figures ii

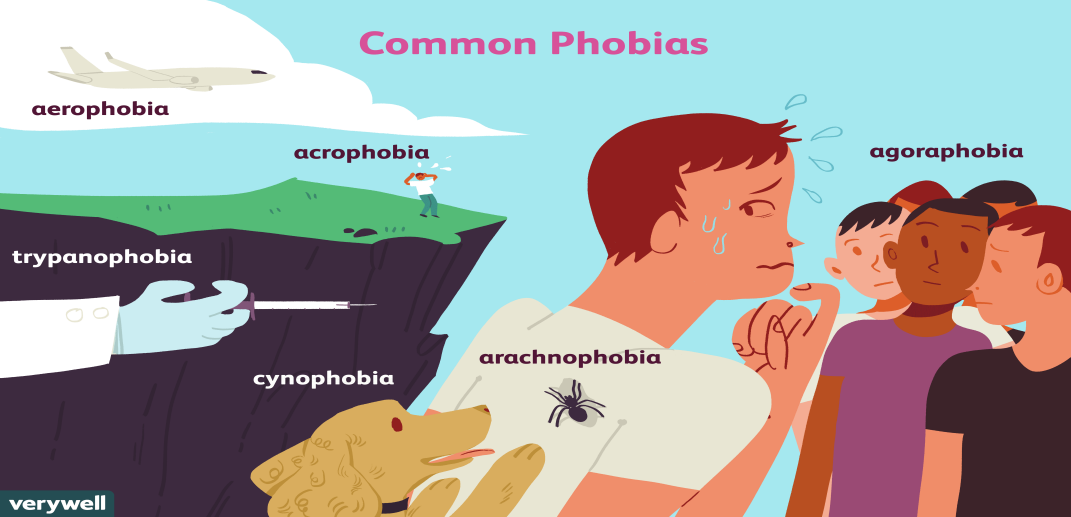
Table of Contents iii - v

1. INTRODUCTION
   1. About Pneumonia 1
   2. Causes of Pneumonia 2
   3. Risk Factors 3
   4. Symptoms of Pneumonia 4
   5. Existing System 5
   6. Limitations of Existing Systems 6-7
   7. Proposed System 8
   8. Advantages of Proposed System 8
2. AIM and OBJECTIVES
   1. AIM of proposed system 9
   2. Objectives of proposed system 10-11
3. LITERATURE SURVEY
   1. Introduction 12
   2. Pneumonia Detection using various Techniques 12-15
4. SOFTWARE / HARDWARE REQUIREMENTS
   1. Functional Requirements 16
   2. Non – Functional Requirements 17
   3. Software Requirements 18-21
   4. Hardware Requirements 22
   5. Tech stack 22
5. SOFTWARE DESIGN
   1. process Cycle 23
   2. User Flow Diagram 24
   3. E R Diagram 24-25
6. SYSTEM IMPLEMENTATION
   1. Methods 26
   2. Model 27
   3. Model Implementation 28
   4. Internal Implementation 28-31
7. TRAINING and TESTING
   1. Training 32
   2. Model Fit and epochs 33
   3. Testing and Evaluation 34-35
   4. Evaluation 36
8. USER INTERFACE INTEGRATION
   1. PyQt Designer Interface 37
9. SAMPLE CODE
   1. level-1.ipynb 38-40
   2. level-2.ipynb 41
   3. level-3.ipynb 42
10. OUTPUT SCREENS
    1. GUI Home Page 43
    2. Login Page 44
    3. Signup Page 45
    4. Options Page 46
    5. Severity Prediction Page 47
    6. Disease Prediction Page 48
    7. Recommendations Page 49
11. [CONCLUSION 50](#_TOC_250001)
12. [FUTURE SCOPE 51](#_TOC_250000)
13. **BIBILOGRAPHY 52-53**

# INTRODUCTION

### About Phobia

Phobias are extreme, irrational dread about specific items, circumstances, or activities that can cause severe suffering and avoidance behavior. It is projected that around 10% of the world's population, or 800 million people, will experience some type of phobia during their lives. Phobias can cause varying degrees of impairment in daily functioning, limiting people's ability to work, socialize, and participate in regular activities.   
Specific phobias, such as acrophobia, arachnophobia are among the most frequent. When people encounter the feared object or circumstance, they can have panic episodes and acute anxiety.Phobias in children might take the form of an intense fear of the dark, separation anxiety, or a fear of specific animals. While many children overcome their phobias, some can last until adulthood, resulting in chronic anxiety disorders.  
According to research, genetic, environmental, and psychological variables can all have an impact on phobias. Phobias can emerge as a result of traumatic experiences, acquired behaviors, or even familial history.Advances in technology and psychology have resulted in a variety of phobia treatment alternatives, including cognitive-behavioral therapy (CBT), exposure therapy, and the use of virtual reality to replicate frightening events in a controlled setting. These approaches try to assist individuals in confronting and managing their fears, thereby increasing their quality of life.



**Fig. 1.1(a) Common phobia types**

### Causes of Phobia

**1. Genetic Factors**   
Genetic predisposition can have a substantial impact on the development of phobias. Individuals with a family history of anxiety disorders or phobias may be more susceptible to developing similar worries. Certain hereditary features can affect how a person reacts to stress and anxiety, making them more prone to phobias.   
  
**2. Environmental factors**   
Environmental factors, such as painful experiences or significant life events, might cause the emergence of phobias. For example, a person who has had a scary encounter with dogs may develop a dog fear (cynophobia). Furthermore, exposure to phobic stimuli throughout childhood can result in the development of specific phobias later in life.   
  
**3. Learned behaviors**   
Phobias can also be acquired through observation and conditioning. If a youngster witnesses a parent or sibling expressing dread of a specific object or circumstance, they may develop comparable concerns. This taught habit may perpetuate the phobia, making it more difficult to overcome.   
  
**4. Psychological Factors**Certain psychological characteristics, such as anxiety sensitivity and cognitive errors, can aid in the development of phobias. Individuals who are more sensitive to anxiety may be more likely to acquire phobias because they see body sensations as harmful. Cognitive distortions, such as catastrophizing or exaggerating the possibility of unfavorable consequences, can aggravate phobic responses.

* Some common triggers that might lead to phobias are:   
  > Traumatic experiences (such as accidents or attacks)   
  > Stressful life events, such as the loss of a loved one, divorce)  
  > Childhood Experiences (e.g., bullying, parental over protection)

### Risk Factors

#### Age

Phobias can afflict anybody, however particular age groups may be more likely to develop specific forms of phobias.

#### Babies and children (2 years old or younger)

Children and adolescents are more vulnerable to develop phobias due to their greater sensitivity and inexperience with coping techniques. Childhood trauma or negative events, such as being bitten by a dog or being trapped in a confined space, can result in long-term phobias.

#### Older adults (age 65 or older )

Adults If not treated, childhood phobias can remain into adulthood. In other circumstances, new phobias might develop as individuals experience new problems, such as health issues or social isolation.

#### Environment or Occupation

Living through or seeing traumatic or stressful events, such as accidents, natural disasters, or violence, can raise the risk of acquiring phobias. Certain phobias, such as social phobia, are more likely to develop in people who grew up in situations where social interactions were closely monitored or were subjected to bullying or criticism. Working in high-stress areas, such as hospitals, disaster response teams, or with animals or insects (for people who have specific animal phobias) may increase the likelihood of developing associated phobias.

#### Lifestyle habits

* + - * Substance Use: Alcohol, narcotics, and stimulants can worsen anxiety and lead to the development of phobias because they impair the brain's ability to control fear and stress.
      * Avoidance Individuals who postpone confronting their concerns or progressively expose themselves to anxiety-provoking events may reinforce the phobia, exacerbating it over time.

#### Other medical conditions

* + - * Anxiety Disorders or Depression: Individuals with generalized anxiety disorder, panic disorder, or depression are more likely to develop specific phobias.
      * Traumatic Brain Injury (TBI): Certain types of brain injuries can alter emotional control, potentially leading to the development of phobias.  
        Chronic sickness can cause people to develop health-related phobias, such as a fear of suffocation, hospitals, or needles.
      * Genetic Factors: People with a family history of anxiety disorders or phobias are more prone to develop them themselves due to inherited genetic vulnerabilities.

### 1.4 Symptoms of Phobia

The signs and symptoms of a phobia can vary from moderate to severe, depending on the type of phobia, the severity of dread, and the individual's overall mental health. Mild symptoms may resemble generalized anxiety, but they worsen when confronted with the feared object or circumstance.

Signs and symptoms of phobia may include:

* Extreme fear or anxiety when confronted with the object or circumstance.
* Panic episodes include rapid heartbeat, perspiration, shaking, or shortness of breath.
* Avoidance behavior is going out of one's way to avoid the feared thing or scenario.
* Feelings of dread or losing control, especially when contemplating about confronting the phobic trigger
* When confronted with fear, one may experience nausea, dizziness, or upset stomach.
* Difficulty functioning in everyday activities owing to severe dread.
* Overwhelming sense of doom or danger despite the fact that the fear is illogical

#### 

#### IMG_256

#### Fig. 1.4.1 symptoms of phobia

### 1.5 Existing System

#### **1. Psychological Evaluation Tools and Diagnostic Assessments** Clinicians can use many standardized psychological evaluations, such as the Fear Questionnaire or fear Diagnostic Tests, to determine the severity and kind of fear. These methods aid in the identification of specific phobic responses, although their interpretation is normally done by a licensed therapist.

#### Electronic Health Records (EHR) Systems

EHR systems are used in mental health to record patients' psychological histories, including the presence of confirmed phobias, their severity, and the treatments they received. This helps to streamline care and monitor patient development over time.

#### Telemedicine Platforms

Teletherapy programs allow patients to consult with psychologists or therapists from a remote location. These platforms facilitate symptom evaluation and therapy interventions for phobias. They may also incorporate self-reporting options for continuous symptom monitoring.

#### Machine Learning and AI Tools

Various AI-powered systems are emerging to diagnose and forecast phobias using patient-reported symptoms, physiological data, and behavior. These technologies attempt to help therapists detect phobia patterns using advanced data analysis, saving diagnostic time and improving therapy planning.

#### Clinical Decision Support Systems

Clinical decision support systems in mental health can make individualized treatment suggestions for phobias based on patient information including symptom severity, previous therapies, and concomitant illnesses. These platforms enable physicians to create more personalized therapy interventions.

#### Healthcare Mobile Apps

Several mental health applications provide services for phobia management, including relaxation exercises, exposure therapy assistance, and symptom tracking. Some may include educational content or allow users to measure their anxiety levels, so functioning as additional tools for phobia self-management.

### 1.6 Limitations of Existing Systems

### The healthcare landscape is continually changing, notably in mental health and phobia detection, thanks to advances in AI, tele health, and data protection. However, there are numerous limitations to current phobia diagnosing techniques. These problems must be considered while evaluating or designing new phobia detection techniques.

#### Dependence on Mental Health Expertise

#### Current systems frequently rely on mental health professionals such as psychologists and psychiatrists to provide correct diagnoses. Diagnosing phobias usually necessitates in-depth patient interviews and psychometric testing, which can cause delays and subjective variances in diagnosis quality.

#### Limited Access to Mental Health Services. Access to mental health specialists, particularly in rural or poor locations, is frequently limited. Tele medicine platforms can help, but a lack of infrastructure and internet connectivity continues to hinder service delivery, making it difficult for patients to receive timely phobia diagnosis and treatment.

#### Data Privacy and Security Concerns

Phobia diagnosis and therapy require sensitive psychological data. Ensuring the security and privacy of personal mental health information on platforms such as Electronic Health Records (EHRs) and tele therapy services is critical, and any compromise can have serious ramifications for individuals.

#### Interoperability Issues

The absence of standardized systems across different mental health platforms can impede collaboration and information sharing among healthcare professionals. This fragmentation has an impact on the continuity of mental health care and might cause delays in treatment for phobia cases that require coordination among multiple professionals.

#### Accuracy of AI Models

#### AI-Artificial intelligence-based phobia detection systems are still in their early stages of development. While substantial progress has been achieved in detecting anxiety disorders by natural language processing and behavior analysis, these models are inaccurate due to the subjective and highly individualized character of phobia symptoms.

#### Lack of Personalization

Many existing phobia diagnostic methods offer generalist assessments rather than tailored evaluations based on a person's specific mental health history or triggers. This lack of customisation can result in insufficient or unsuccessful phobia treatment plans.

#### Resource Constraints

In resource-constrained countries, access to mental health experts and modern diagnostic techniques like cognitive exams and psychometric evaluations is limited. This constraint affects the accuracy and speed of phobia detection and management.

#### Patient Engagement

Many mental health apps or platforms that assist patients track symptoms or manage their phobias struggle to maintain long-term user engagement. It is still difficult to encourage patients to use self-assessment tools or participate in online therapy programs on a frequent basis.

#### Research and Clinical Validation

#### Many of the current phobia detection techniques or technologies are experimental or at the prototype stage, with limited clinical validation. Their effectiveness and accuracy cannot be fully demonstrated without thorough real-world testing, limiting their adoption into mainstream mental healthcare.

#### Regulatory and Ethical Challenges

### Compliance Compliance with mental health standards, ethical concerns about patient permission, and the handling of sensitive data all pose substantial problems. Variability in regulatory norms among regions may impede the use of digital technologies for phobia diagnosis and therapy.

### 1.7 Proposed System

The proposed system aims to transform phobia detection and treatment through a web-based platform that leverages advanced machine learning and AI. The system engages users with personalized questions to assess their fears, categorizing phobias such as social phobia, acrophobia, and claustrophobia. Using machine learning algorithms, it accurately diagnoses the phobia based on user responses and provides tailored daily instructions and coping strategies. The platform adapts its recommendations dynamically, tracking user progress over time. To ensure a holistic approach, the system integrates professional mental health support by connecting users with psychiatrists, psychologists, and motivational speakers for online consultations and therapy sessions.

This seamless combination of AI-driven diagnosis and human intervention ensures effective treatment, even for severe cases. The platform prioritizes user privacy, employing strong data encryption and security protocols to safeguard sensitive mental health information. User engagement is maintained through progress tracking, motivational messages, and interactive mental health tips. By making mental health care more accessible, personalized, and engaging, the system aims to empower individuals to overcome their phobias and improve their overall mental well-being, providing them with the necessary tools and support for a fulfilling life.

### 1.8 Advantages of Proposed System

**Table 1.8.1 Advantages of Proposed System**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Proposed System** | **Existing Systems** |
| **Personalized Daily**  **Task Management** | Provides personalized daily tasks based on specific phobia type and severity. | Offers general information on phobias without tailored daily tasks. |
| **Interactive User Interface** | Features an interactive UI with check boxes for task tracking | Lacks an interactive platform for user engagement. |
| **Streak Tracking Mechanism** | Implements streak tracking to encourage daily logins and task completion. | Does not include gamification or engagement tracking |
| **User-Friendly Task Reset** | Automatically resets daily task completion status at the start of each day. | No automated reminders or task resets. |
| **Data Storage and Accessibility** | Utilizes SQLite to security store user information and task status. | Focuses on providing general health information with the personalized data and storage. |
| **Focus on Symptoms Reporting** | Allows real-time symptom reporting to track changes and adapt tasks. | Emphasizes diagnosis and treatment recommendations without direct symptom report. |
| **Integration of Technology** | Integrates modern technologies like Python and UI frameworks. | Offers traditional healthcare services without interactive software. |
| **Comprehensive Task Management** | Provides a holistic approach to managing phobias with daily tasks and feedback. | Focuses primarily on therapy and medication without structured task management. |
| **Access to Community Support** | Could integrate community support features for sharing experiences. | Primarily delivers professional treatment options, lacking community support. |

## 2.AIM and OBJECTIVES

### 2.1 AIM of proposed System

### The proposed system's major purpose is to address the growing mental health concerns about phobias and anxiety disorders by developing an innovative platform for diagnosis and individualized therapy. This system uses cutting-edge machine learning techniques to provide a complete, accessible, and user-friendly solution for people suffering from a variety of phobias, including social anxiety, specific phobias (such as acrophobia or claustrophobia) and complex phobias. The approach starts with a personalized, question-based examination that examines each user's anxieties, symptoms, and psychological triggers. The technology uses targeted questioning to identify the exact phobia type and assess its severity, providing a highly personalized experience. The system also acknowledges the value of professional mental health care. It smoothly incorporates connections to qualified psychiatrists, psychologists, and motivational speakers, allowing users to seek expert guidance and therapy as necessary. Furthermore, the platform monitors user progress over time and adjusts recommendations based on changes in the user's condition.

The approach, which is implemented through a user-friendly web application, increases accessibility to mental health care and encourages people to take control of their anxieties. While it can help manage mental health, it cannot replace expert diagnosis and therapy. The system supplements the complete mental health care process, ensuring that users receive comprehensive assistance and resources for conquering phobias and enhancing their well-being.

### 2.2 Objectives of proposed system

#### 1. Develop a Symptom-Based Prediction Layer

The first objective is to create a symptom-based prediction layer that enables the system to assess the presence and severity of phobias based on user-reported symptoms and fears. By developing algorithms that analyze user responses, this layer provides an initial evaluation, which serves as the entry point for users. The prediction layer determines whether users require immediate coping strategies or should seek further evaluation from mental health professionals. Rigorous data analysis and model training are essential to ensure accurate detection across various types of phobias.

#### Implement Machine Learning Models for Phobia Diagnosis

#### The second objective is to incorporate machine learning models into the system for the accurate diagnosis and classification of phobias. These models will analyze patterns in user behavior, responses, and symptom severity to diagnose phobias such as social anxiety, specific phobias (e.g., claustrophobia), or complex phobias. By training these models on extensive datasets that cover a wide range of phobia cases and severities, the system can offer reliable diagnostic insights, helping users understand the nature of their fears.

#### Provide Personalized Coping Strategies and Daily Guidance

#### The third objective is to deliver personalized coping strategies and daily mental health exercises tailored to each user’s specific phobia and situation. Once a user's phobia type and severity are determined, the system should generate customized recommendations, including relaxation techniques, gradual exposure methods, and motivational tips. The system integrates psychological research and best practices to ensure these suggestions are safe, effective, and suited to the individual’s mental health needs.

#### Create a User-Friendly Application

The fourth objective is to design and develop an intuitive, user-friendly web application that allows seamless interaction with the system. The application should guide users through the phobia detection process, presenting questions, providing results, and offering recommendations in a clear, comprehensible format. Accessibility is key, ensuring the platform can be used by individuals with various levels of technical expertise and mental health literacy.

#### Facilitate Professional Support and Consultation

#### The fifth objective focuses on integrating professional mental health support into the system. While the platform provides valuable self-help resources, it also connects users with licensed psychiatrists, psychologists, and motivational speakers for personalized consultations. This feature enables users to access professional advice when necessary, ensuring they receive expert guidance alongside AI-driven diagnosis and treatment suggestions.

#### **Promote Early Phobia Detection and Mental Health Management**

The final overarching objective is to promote early detection of phobias and improve mental health management. By implementing the aforementioned layers, the system aims to provide timely, accurate, and personalized support to individuals facing phobias, ultimately helping them confront their fears and improve their quality of life. This system complements professional mental health care by offering users immediate tools to manage their condition while encouraging them to seek further help when necessary.

# LITERATURE SURVEY

### 3.1 Introduction

### Phobias, a prevalent but frequently devastating mental health disorder, provide substantial issues for individuals and healthcare systems worldwide. Addressing these difficulties necessitates novel techniques that combine advanced technologies, psychological insights, and tailored treatment. To improve phobia detection and treatment, we must first have a complete awareness of current research and improvements in mental health care. This literature review provides a basic analysis of relevant studies, emphasizing major findings, trends, and gaps that drive our project's aims.

The literature review is structured to include essential topics such as phobia detection using machine learning approaches, individualized therapy interventions, user-friendly digital health platforms, and the role of mental health experts in treatment. By rigorously reviewing existing research, we hope to give a comprehensive framework that emphasizes the significance of our project's objectives and potential to improve mental health treatment results.  
  
This evaluation of the present corpus of information provides a solid foundation for the development of an improved phobia detection and individualized treatment system. This review allows us to better match our initiative with recognized psychological techniques, cutting-edge technologies, and the need for early intervention. Finally, it ensures that our system can contribute effectively to the mental health landscape, by offering accessible, individualized care for individuals coping with phobias and anxiety disorders.

### 3.2 Phobia Detection using various Techniques

#### 1.Utilization of VR and AR Exposure Therapy for Specific Phobia Treatment

- Exposure therapy is a common method for treating specific phobias.

- VR exposure therapy (VRET) and AR exposure therapy (ARET) show promise in reducing anxiety through gradual exposure.

- A systematic review across multiple databases selected 18 studies on VRET and ARET.

- VRET was effective for many phobias, though some responded better to traditional treatments.

- The study also compared the effectiveness of VR and AR technologies.

- Limitations and future improvements in VRET and ARET are discussed.

#### 2.Attention to Visual Threat in Spider Phobics

- People with anxiety disorders attend more to threat-relevant stimuli than non-threat stimuli.

- This study compared spider-phobic and non-fearful participants using a rapid visual presentation task.

- Spider phobics showed quicker detection of spider stimuli compared to negative and neutral stimuli. Signal detection analysis indicated that spider-phobics had greater sensitivity to threat features.

- There was a trend toward a lower threshold for detecting spider stimuli in phobic participants.

- No evidence of slowed disengagement from threat-relevant stimuli was found. These insights enhance understanding of threat processing in specific phobias.

1. **Neurobiology of Fear and Specific Phobias**

- Fear, either innate or conditioned, is triggered by perceived danger, preparing the body to respond.

- Dysfunction in fear processing can result in psychiatric disorders where fear is disproportionate to actual danger.

- This article reviews the neurobiological mechanisms behind normal and pathological fear, with a focus on the amygdala.

- Specific phobias involve both innate (learning-independent) and experiential (learning-dependent) mechanisms.

- Poor habituation and poor extinction are identified as key dysfunctional processes in the persistence of these phobias.

1. **Toward Fear Detection using Affect Recognition**

- The Affective Internet of Things (AIoT) uses sensors to detect or predict a person's emotional or affective state.

- Wearable systems, particularly personal area networks, play a crucial role in affect recognition research.

- This paper presents a PhD thesis focusing on a fear detection system using wearable-ready sensors, specifically blood volume pulse and galvanic skin resistance.

- A multi-modal approach based on arousal and valence scores is proposed to recognize emotions like fear.

- The results are compared with current methods, and future steps for developing a fear-specific detection system are outlined.

1. **Human Fear Analysis using Signal and Image Processing**

- This paper explores human emotion detection, focusing on fear analysis through image and signal processing.

- It employs multimodal emotion analysis using fused facial image processing, emotional speech processing, and physical parameters.

- Statistical feature extraction is conducted using various algorithms, including Mel Frequency Cepstral Coefficients for speech and deep learning techniques for classification.

- The study reports a sensitivity of 97.36% and specificity of 91.67% for fear detection, significantly improving upon single-feature analyses.

- Results indicate that combined feature sets enhance classification accuracy, with the modified Mel Frequency Cepstral Coefficients algorithm proving effective for fear emotion detection.

1. **Multi-Input CNN-LSTM Deep Learning Model for Fear Level Classification Based on EEG and Peripheral Physiological Signals:**

- This study aims to classify human fear levels using a deep learning model, crucial for developing treatments for anxiety disorders and phobias.

- A Multi-Input CNN-LSTM model, integrating Convolutional Neural Networks and Long Short-Term Memory, is employed to analyze multichannel EEG and peripheral physiological signals from the DEAP dataset.

- The model achieved an impressive accuracy of 98.79% and an F1 score of 99.01% through 10-fold cross-validation.

- Key contributions include demonstrating high-accuracy fear emotion recognition without manual feature extraction and investigating effective deep learning structures for fear recognition.

- The study also explores the model's ability to handle individual differences in physiological signals and suggests potential improvements through further training.

# SOFTWARE / HARDWARE REQUIREMENT

### Functional Requirements

#### User Authentication and Retrieval:

#### Users should be able to log in to the system securely.

#### The system should retrieve the most recently logged-in user's email for personalized.

#### Phobia Data Retrieval:

#### The system should fetch the predicted phobia type and level for the logged-in user from the database.

#### The retrieved data should be used to tailor the daily tasks displayed to the user.

#### Precautionary Task Display:

#### Users should be presented with a list of daily tasks (precautions) based on their specific phobia type and level.

#### The system should load and filter tasks from an Excel sheet for personalized display.

**4. Task Checkbox Feature:**

* Users should be able to interact with daily tasks by marking them as completed using check boxes.
* The system should reflect the current state of each task based on user interaction.

#### 5.Task Completion Persistence:

#### Previously checked tasks should be displayed as pre-checked when the user logs back in.

#### The system should retain the user's progress across sessions for a seamless experience.

#### 6. Daily Task Reset:

#### The system should reset the task completion status at the start of each new day.

#### Users should be encouraged to complete their daily tasks with a fresh start each day.

#### Task Progress Update:

* The system should update the database with the checked tasks once the user clicks the "Save Checked Precautions" button.
* A success message should be displayed to confirm the tasks have been saved.

1. **Completion Feedback:**

* Users should receive positive feedback upon completing all daily tasks.
* The system should include motivational messages and visual feedback to encourage continued engagement.

1. **Streak Tracking:**

* The system should track whether the user logs in daily and completes their tasks.
* Streaks should be updated to encourage consistent user engagement with the application.

**10 . Data Storage and Handling:**

* The system should securely store and retrieve all user information, including phobia type, task completion, and streak data using SQLite.
* Persistent storage should ensure that user data is available across sessions for continuity

### Non – Functional Requirements

#### Performance

#### The system should provide quick responses for phobia symptom assessment.

#### It should handle concurrent user interactions efficiently.

#### Scalability

* Design the system to accommodate a growing user base and increasing data volume related to phobia assessments.
* Ensure scalability to meet potential future demands as more users seek phobia-related support.

#### Accuracy

#### The models should achieve high accuracy in phobia detection and level assessment (mild, moderate, severe).

#### Minimize false positives and false negatives in phobia diagnosis.

#### Reliability

#### Ensure system uptime and reliability for critical mental healthcare support.

#### Implement backup and recovery mechanisms to prevent data loss or disruptions.

#### Usability

#### The user interface should be intuitive and accessible to a diverse user base, including individuals with phobias.

#### Consider usability testing and user feedback for continuous improvement, especially with users who may experience anxiety.

#### Security

#### Protect user data, personal details, and sensitive psychological records with robust security protocols.

#### Regularly update security measures to address emerging threats and ensure confidentiality.

#### Compliance

#### Comply with relevant healthcare regulations, such as HIPAA in the United States or equivalent laws in other regions, concerning mental health data.

#### Adhere to ethical standards in data handling and patient care, especially with sensitive psychological information.

#### Interoperability

### Ensure compatibility with existing healthcare systems and standards for data exchange related to mental health records.

### Facilitate seamless integration with other healthcare IT solutions, including teletherapy or psychological support platforms.

### 4.3 Software Requirements

#### Python



#### Fig, 4.3.1 Python logo

Python is a widely-used high-level programming language known for its simplicity and readability. It features clean and easy-to-understand syntax, making it a favorite among developers. Python is interpreted, cross-platform, and comes with an extensive standard library, reducing the need for custom code. It supports dynamic typing and object-oriented programming principles and is highly interoperable with other languages. Python has a thriving community and offers versatility, making it suitable for web development, data analysis, machine learning, and more. Its open-source nature and ease of learning have contributed to its widespread adoption.

#### scikit learn



#### Fig. 4.3.2 scikit learn logo

Scikit-learn is a powerful machine learning library in Python that simplifies the process of building and applying machine learning models. It is widely used for tasks like classification, regression, clustering, and dimensionality reduction. Scikit-learn provides a user-friendly and consistent interface for various machine learning algorithms, making it an essential tool for data scientists and developers. It also offers comprehensive functionality for tasks like data preprocessing, model evaluation, and hyperparameter tuning. This library's simplicity, efficiency, and integration with other Python libraries make it a valuable asset in the field of machine learning and data analysis.

#### SQLite



#### Fig. 4.3.3 SQLite logo

SQLite is a lightweight, self-contained, and serverless relational database management system (RDBMS). It is widely used for embedded systems, mobile applications, and small-scale database solutions due to its minimal setup requirements and portability. SQLite stores its entire database as a single file, simplifying management and deployment. It supports SQL (Structured Query Language) for data manipulation and retrieval and provides transactional capabilities for data consistency. SQLite is known for its efficiency and speed, making it a suitable choice for scenarios where a full-fledged RDBMS may be unnecessary or impractical.

#### VScode



#### Fig. 4.3.4 VS code logo

Visual Studio Code (VSCode) is a widely-used integrated development environment (IDE) known for its lightweight and versatile nature. It's developed by Microsoft and is favored by developers for its efficiency and extensibility. VSCode is highly customizable, allowing developers to tailor it to their specific needs with the help of numerous extensions available in its marketplace. Its key features include a powerful code editor with syntax highlighting, debugging spport, and integrated version control.

#### Colab and Jupyter



#### Fig. 4.3.6 jupyter, colab logo

Jupyter is an open-source interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports various programming languages, including Python, R, and Julia. Jupyter notebooks, which are a key component of Jupyter, enable users to combine code execution with explanatory text, making it an excellent tool for data analysis, scientific research, and educational purposes.Jupyter notebooks run in a web browser and can integrate with data visualization libraries, making it easier to explore and present data-driven insights. Google Colab (Colaboratory) is a cloud-based platform provided by Google that allows users to create and run Jupyter notebooks without the need for local installation or powerful hardware. Colab offers free access to GPU and TPU resources, making it well-suited for machine learning and deep learning tasks. Users can collaborate on Colab notebooks in real-time, share them with others, and access pre-installed libraries and packages, simplifying the setup process for data analysis and machine learning projects. Colab notebooks are stored on Google Drive, ensuring easy access and version control.

1. **Streamlit**



Streamlit is an open-source Python framework designed for creating interactive, data-driven web applications with minimal effort. Tailored for data scientists and machine learning engineers, Streamlit simplifies the process of turning Python scripts into shareable web apps without requiring extensive front-end development knowledge. With its intuitive API, users can quickly build dashboards, visualizations, and interfaces to interact with data models and machine learning workflows. Streamlit supports real-time updates, making it ideal for prototyping and deploying data applications. Its lightweight, fast, and easy-to-use nature allows for rapid iteration and deployment, making it a popular choice for building and sharing data-centric applications.

### 4.4 Hardware Requirements

* **CPU:** A multi-core processor, preferably with support for parallel processing, is essential for running deep learning models efficiently. A modern CPU with multiple cores (e.g., quad-core or higher) is recommended.
* **RAM:** Adequate RAM is crucial, especially when dealing with large datasets and deep learning. Depending on your specific needs, a minimum of 16 GB of RAM is a good starting point, but more may be necessary for larger deployments.
* **GPU (Graphics Processing Unit):** To accelerate deep learning model training, consider using one or more GPUs. NVIDIA GPUs are commonly used for this purpose. High-end GPUs with CUDA support can significantly speed up the training of complex models, but we have used colab T4 Pro version of GPU.
* **Storage:** You will need sufficient storage capacity to store user data, medical records, deep learning model weights, and other system data. Consider fast SSDs for improved data access speed.
* **Compatibility:** Ensure that the user interface of your application is responsive and compatible with a variety of devices and screen sizes.
* **Network Bandwidth:** Adequate network bandwidth is essential, especially if your system handles a large number of image uploads and downloads. High-speed internet connectivity and sufficient internal network bandwidth are required for efficient data transfer.

### Tech stack

* **Programming language**  Python
* **Development Fremework**  Tensorflow, scikit – learn
* **Database**  SQLite
* **IDE**  VScode, colab, jupyter
* **Data visualization**  matplotlib, seaborn
* **Operating System**  Window 11
* **Model Development**  keras, EfficeientNet
* **Interface Development**  PyQt Designer

# SOFTWARE DESIGN

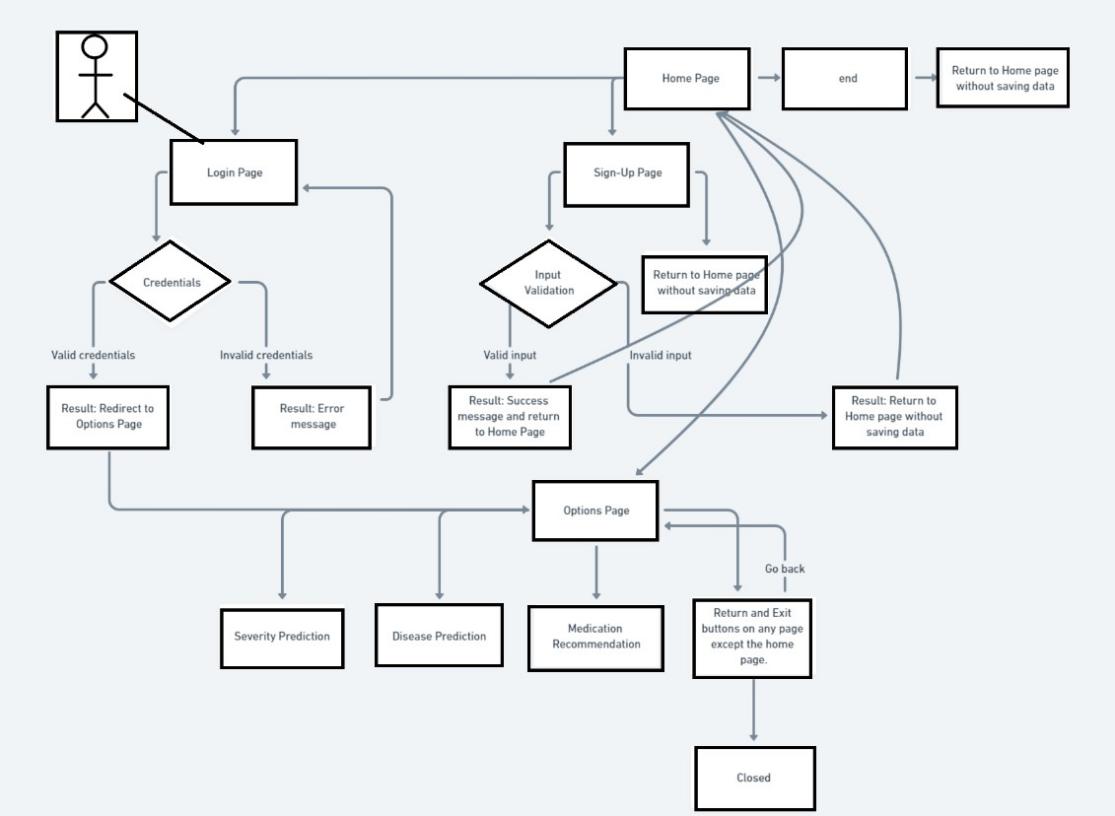
### Process Cycle

The application begins with a Home Page where users can choose to "Login," "Sign Up," or "Exit." If they opt for "Login," they're directed to the Login Page to enter their credentials, and successful login leads to the Options Page. Registration occurs on the Sign-Up Page, with user data stored upon success.

The Options Page serves as the central hub, offering functionalities like Severity Prediction, Disease Prediction, and Medication Recommendation. Users can exit the app from here.

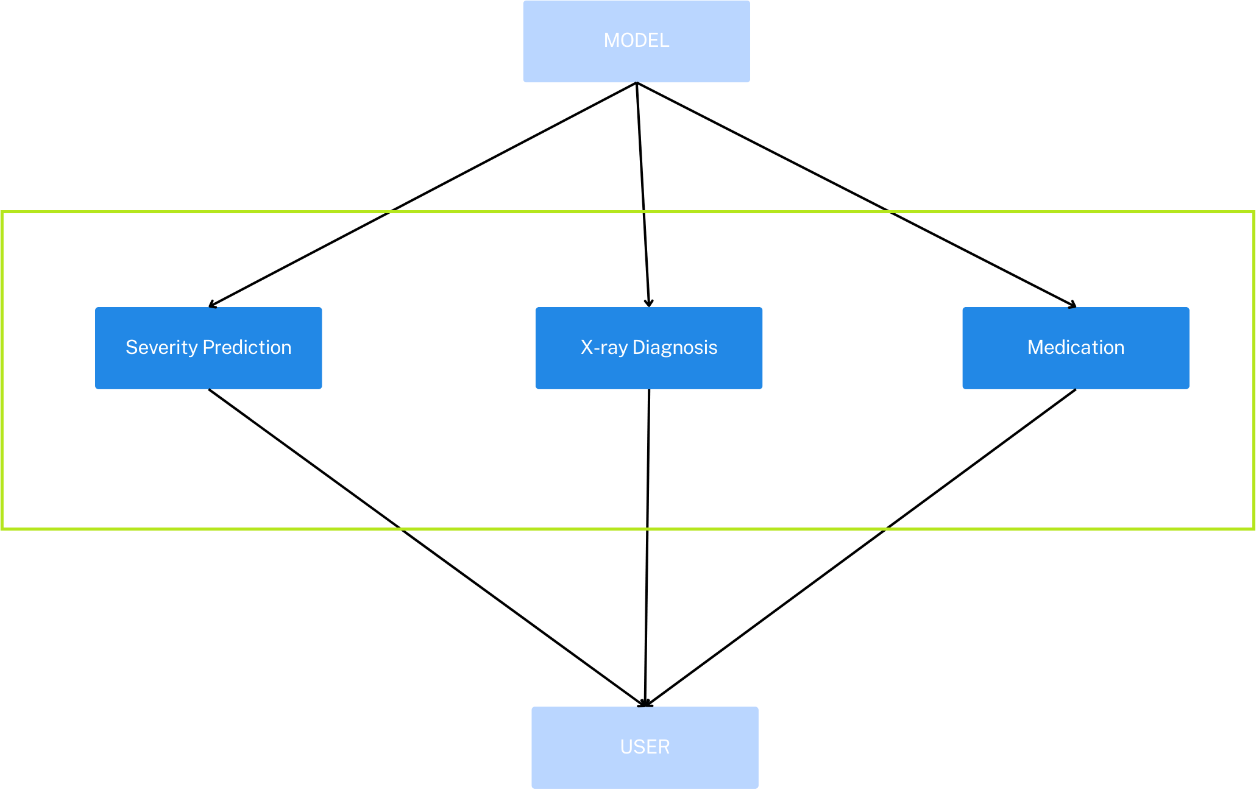
The Severity Prediction Page lets users input age and symptoms to predict pneumonia severity, with visual feedback based on results. Disease Prediction Page allows users to upload X-ray images for "Normal" or "Pneumonia" predictions.

In the Medication Page, users select severity and diagnosis to receive medication recommendations. The process cycle enables users to navigate, interact, and make informed healthcare decisions regarding pneumonia prediction and treatment, all within the user-friendly graphical interface of the application.



#### Fig. 5.1.1 Process cycle

### User Flow Diagram

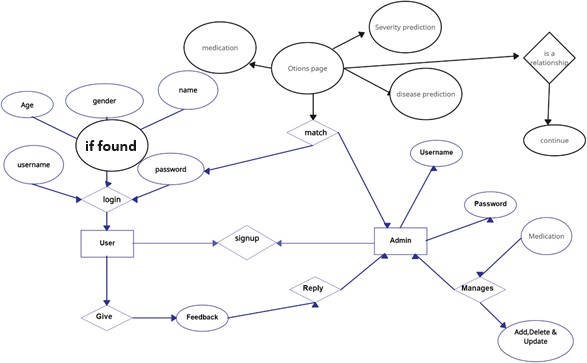


#### Fig. 5.2.1 User Flow Diagram

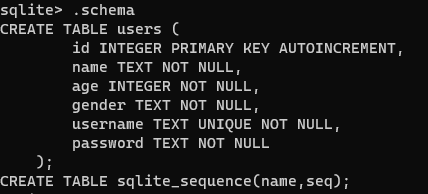
### E R Diagram

#### users Table Schema Description:

* + 1. id (INTEGER PRIMARY KEY AUTOINCREMENT): This field serves as the primary key for the table, ensuring each record has a unique identifier. The AUTOINCREMENT attribute automatically generates a unique ID for each new record.
    2. name (TEXT NOT NULL): This field stores the user's full name as text. It's marked as "NOT NULL," meaning that a user's name must be provided and cannot be left empty.
    3. age (INTEGER NOT NULL): This field stores the user's age as an integer. Similar to the "name" field, it's marked as "NOT NULL," indicating that age information must be provided.
    4. gender (TEXT NOT NULL): This field stores the user's gender as text. Like the previous fields, it's marked as "NOT NULL," ensuring that gender information is required.
    5. username (TEXT UNIQUE NOT NULL): This field stores the user's chosen username for authentication. It must be unique to each user, preventing duplicate usernames. It's marked as "NOT NULL" to ensure that a username is provided.
    6. password (TEXT NOT NULL): This field stores the user's password for authentication. Like other sensitive data fields, it's marked as "NOT NULL."



#### Fig. 5.3.1 E R Diagram



**Fig. 5.3.2 Schema in SQLite3**

**6.SYSTEM IMPLEMENTATION**

**6.1 Overview**

The phobia prediction project is designed to analyze various symptoms and factors associated with different phobias. By leveraging machine learning techniques, the system aims to accurately predict the type of phobia a user may have based on input data. The implementation encompasses data collection, pre processing, model training, and deployment phases.

### **6.2. Data Collection and Pre processing**

Data plays a crucial role in the performance of any machine learning model. For our project, we gathered a substantial amount of data from diverse sources, specifically targeting research papers, phobia-related websites, and hospital records. The data collection process involved the following steps:

#### **6.2.1 Data Sources**

**1.Research Papers**: Academic journals and publications were reviewed to extract relevant data on phobia symptoms, classifications, and patient demographics. This provided a theoretical framework and insights into the common patterns observed in phobia cases.

**2.Phobia and Hospital Websites**: Websites specializing in mental health, psychological disorders, and phobia treatments were utilized to obtain real-world data. Information regarding patient experiences, symptoms descriptions, and treatment responses was collated from these sources. This empirical data adds depth and validity to the dataset.

#### **6.2.2 Data Compilation**

The collected data consisted of various attributes, including:

* **Age**: The age of the individual experiencing the phobia.
* **Current Stressors**: Any existing stress factors that might contribute to the phobia.
* **Symptoms**: Descriptions of the symptoms experienced by the individual.
* **Duration**: The time span for which the individual has experienced the phobia.
* **Fear Of**: Specific fears associated with the phobia.
* **Frequency**: How often the symptoms manifest.
* **Phobia**: The type of phobia, if diagnosed.
* **Type**: Categorization of symptoms and conditions.

#### **6.2.3 Data Cleaning**

Once the data was collected, it was subjected to a rigorous cleaning process to ensure accuracy and consistency. This involved:

* **Handling Missing Values**: Missing data entries were identified and addressed through imputation or removal, ensuring the dataset remains robust.
* **Normalization**: Data normalization techniques were applied to standardize the scale of numerical features. This is critical for machine learning algorithms that are sensitive to the scale of input data.
* **Encoding Categorical Variables**: Categorical attributes, such as 'Phobia' and 'Frequency,' were encoded into numerical formats using techniques like one-hot encoding, allowing the model to interpret these features effectively.

#### **6.2.4 Feature Engineering**

To enhance model performance, feature engineering techniques were employed:

* **Combining Features**: New features were created by combining existing attributes that showed potential correlation. For instance, a composite feature could be developed from age and duration to capture more intricate relationships.
* **Dimensionality Reduction**: Techniques such as Principal Component Analysis (PCA) were considered to reduce the dimensionality of the dataset, improving the efficiency and accuracy of the model.

### **6.3 Model**

The Phobia Type Prediction Model utilizes the Random Forest Classifier, a robust ensemble learning technique known for its effectiveness in classification tasks. This model is particularly suitable for our project due to its ability to handle both categorical and numerical data efficiently. Additionally, Random Forest is less prone to overfitting compared to many other algorithms, which is advantageous given the relatively small size of our dataset.

### **6.3.1 Model Architecture**

#### **6.3.1.1 Input Layer**

The input features for the Random Forest Classifier were meticulously selected based on their relevance to phobia identification. The features included:

* **Age**: Represents the age of the individual, which can influence the likelihood and type of phobia experienced.
* **Stress Levels**: Captures the current stressors affecting the individual, as elevated stress can exacerbate phobia symptoms.
* **Symptoms**: A detailed description of the symptoms experienced by the individual, which provides critical insights into the specific nature of the phobia.
* **Fear Factors**: Lists specific fears associated with the individual's phobia, helping to delineate the type of phobia.
* **Duration**: Indicates how long the individual has been experiencing symptoms, as chronic conditions may suggest specific phobia types.
* **Frequency**: Describes how often the symptoms occur, providing context on the severity and management of the phobia.

#### **6.3.1.2 Feature Importance**

One of the key advantages of the Random Forest model is its ability to assess feature importance, which identifies the most significant factors contributing to phobia type prediction. By analyzing the importance scores, we can gain insights into which features play a crucial role in influencing the model's predictions. This information can guide future research and therapeutic approaches by highlighting the critical areas for intervention.

**6.4 Model Implementation**

Both models were developed using Python, leveraging the powerful scikit-learn library to facilitate machine learning tasks. The dataset was divided into training and testing subsets using a standard train-test split of 80% for training and 20% for testing. This approach ensures that the models are trained on a substantial portion of the data while reserving a sufficient amount for validation, allowing us to assess their performance effectively.

### **6.4.1 Training the Models**

#### **6.4.1.1 Phobia Type Model**

The **Phobia Type Prediction Model** employs a Random Forest Classifier, which is particularly suited for this task due to its ability to manage a mix of categorical and numerical data while mitigating the risk of overfitting.

1. **Model Configuration**:
   * The Random Forest classifier was initialized with a specific number of trees, typically ranging from 100 to 500, to ensure a robust ensemble learning approach.
   * Hyperparameters such as the maximum depth of trees and the minimum number of samples required to split an internal node were tuned through cross-validation techniques to achieve optimal performance.
2. **Training Process**:
   * The model was trained on the training dataset, learning to recognize patterns and relationships within the input features (age, stress levels, symptoms, fear factors, duration, and frequency) to classify the different types of phobias.
   * During training, the model adjusted its internal parameters to minimize prediction errors, improving its accuracy in classifying phobia types.

#### **6.4.1.2 Phobia Level Model**

The **Phobia Level Prediction Model** utilizes a second Random Forest Classifier to assess the severity of the identified phobia, categorized as mild, moderate, or major.

1. **Model Configuration**:
   * Similar to the phobia type model, this classifier was configured with an appropriate number of trees and hyperparameters to enhance learning and performance.
   * Cross-validation was employed again to ensure the model's reliability and robustness against overfitting.
2. **Training Process**:
   * The model was trained on the same input features, allowing it to learn the nuances that distinguish varying severity levels of phobias.
   * Post-training, the predicted severity level was adjusted based on the most frequently occurring severity levels for the corresponding phobia types to enhance clinical relevance.

### **6.4.2 Model Evaluation**

Once the models were trained, their performance was evaluated using a variety of metrics:

#### **6.4.2.1 Phobia Type Prediction Accuracy**

* The Phobia Type Model achieved an impressive **99.3% accuracy** on the test set. This high accuracy indicates that the model effectively classified the different types of phobias, with very few misclassifications.
* A classification report was generated, providing insights into precision, recall, and F1-score for each class of phobia. This report highlighted the model's strengths and areas for potential improvement, ensuring a comprehensive understanding of its performance.

#### **6.4.2.2 Phobia Level Prediction Accuracy**

* The Phobia Level Model demonstrated even greater performance, with a **99.9% accuracy** for predicting the severity of phobia. This near-perfect accuracy underscores the model's ability to discern subtle distinctions between mild, moderate, and major phobias.
* Similar to the phobia type model, a detailed classification report was produced, showcasing the model's effectiveness across different severity levels, ensuring that it could reliably assist in clinical assessments and interventions.

**6.5 Internal Implementation**

The phobia type and severity prediction models were seamlessly integrated into a user-facing web application built with **Streamlit**. This interactive application is designed to facilitate user engagement by allowing individuals to input their data and receive immediate feedback regarding their phobia type and severity. The implementation is structured around several key components that enhance user experience and model performance.

### **6.5.1 Key Components**

#### **6.5.1.1 User Input Interface**

The user input interface is a crucial aspect of the application, ensuring that users can easily provide their information for analysis.

* **Input Collection**:
  + The application collects user input through a series of intuitive text boxes and dropdown menus, enabling users to enter various factors, including:
    - **Age**: Users input their age to help the model tailor predictions based on demographic data.
    - **Current Stressors**: Users select from predefined options or enter specific stressors affecting their mental health.
    - **Symptoms**: A list of common phobia-related symptoms allows users to specify their experiences.
    - **Fear Factors**: Users describe their fears in detail, which is essential for accurate classification.
    - **Duration**: Users specify how long they have been experiencing these phobias or related symptoms.
    - **Frequency**: A dropdown menu enables users to select how often they experience these fears (e.g., occasional, regular, persistent).
* **Data Preprocessing**:
  + To ensure consistency with the training data, the application preprocesses user input by applying the same transformations used during model training. This includes:
    - **Lowercasing**: All text inputs are converted to lowercase to maintain uniformity and reduce discrepancies in data processing.
    - **Label Encoding**: Categorical inputs are transformed into numerical labels, allowing the models to process the data effectively. This step is crucial for handling categorical variables like symptoms and fear factors.

#### **6.5.1.2 Prediction Logic**

Once the user inputs their data, the application utilizes the trained models to deliver predictions.

* **Phobia Type Prediction**:
  + The input data is first fed into the **Phobia Type Model**, which analyzes the information to determine the most likely phobia type. This model employs the learned patterns from training to classify the user's input accurately.
* **Severity Level Prediction**:
  + Following the initial classification, the application leverages the **Phobia Level Model** to predict the severity of the identified phobia. Adjustments are made based on established patterns associated with the predicted phobia type, enhancing the model's reliability. For instance, if the model predicts a specific phobia type, it can utilize data about typical severity levels for that phobia to refine its prediction.

#### **6.5.1.3 Result Display**

The application provides users with clear and actionable insights based on the model's predictions.

* **Predicted Outputs**:
  + After processing the input data, the app displays the predicted phobia type alongside the severity level. This information is presented in an easy-to-understand format, allowing users to gain immediate insights into their condition.
* **User Feedback**:
  + Additionally, the application can provide recommendations or resources for coping strategies, helping users to better understand their situation and seek appropriate support.

#### **6.5.1.4 Streak Tracking System**

To encourage regular engagement with the application, a **streak tracking system** has been implemented.

* **Daily Login Incentives**:
  + The streak system motivates users to log in daily and complete designated tasks aimed at overcoming their phobia. Users are prompted with tasks tailored to their needs, enhancing their engagement with the application.
* **Rewards Mechanism**:
  + Upon completing all daily tasks, users receive visual feedback in the form of a streak icon, which signifies their commitment to overcoming their phobia. Additionally, users earn coins as rewards for their consistent efforts, which can be used within the application for unlocking additional resources or features.
* **Progress Tracking**:
  + The application includes a visual representation of the user's progress over time, such as a calendar view with color-coded days to indicate activity levels. This feature helps users to remain motivated and accountable in their journey toward managing their phobia.

## TRAINING and TESTING

### Training

* Training a machine learning (ML) model for phobia detection involves identifying patterns in input data, such as symptoms and demographic factors, to classify different types and severity levels of phobias. This process typically utilizes algorithms from supervised learning, where labeled data is used to teach the model.
* **Dataset Preparation:** A comprehensive dataset consisting of symptom descriptions, patient details (such as age), and corresponding phobia diagnoses is required. For this project, a dataset of labeled phobia cases was collected, categorizing them by phobia type (e.g., Acrophobia, Arachnophobia) and severity levels (mild, moderate, major).
* **Model Selection:** Various machine learning models were explored, including logistic regression, decision trees, and random forest classifiers. These models were selected based on their ability to handle categorical data and perform multi-class classification.
* **Data Split:** The data set was split into a 70:30 ratio, with 70% of the data used for training the model and 30% reserved for testing. This ensures a balanced distribution for both model training and evaluation, helping to avoid over fitting.
* **Feature Engineering:** Features such as age, symptom duration, and intensity were normalized to enhance the model’s learning. Feature selection techniques were applied to identify the most important factors influencing phobia detection.
* **Training Process:** The selected models were trained using cross-validation techniques to ensure robust performance. Hyper parameters like maximum tree depth, regularization strength, and the number of estimators were fine-tuned during training to improve accuracy and reduce bias.

### Model Fit and epochs

* **Training for Severity Prediction:** Severity levels (mild, moderate, major) were predicted based on input data using a multi-class classification approach. Several models were iteratively trained to minimize prediction error, with each model's performance evaluated based on accuracy and recall.
* **Training for Phobia Type Prediction:** A separate classifier was trained to predict the type of phobia based on input symptoms. Techniques like random forest and support vector machines were tested for their ability to classify multiple phobia types.
* **Training for Treatment Recommendation:** Based on the predicted phobia type and severity, a model was developed to recommend suitable treatment options, which can include therapy types, medication, or other interventions.

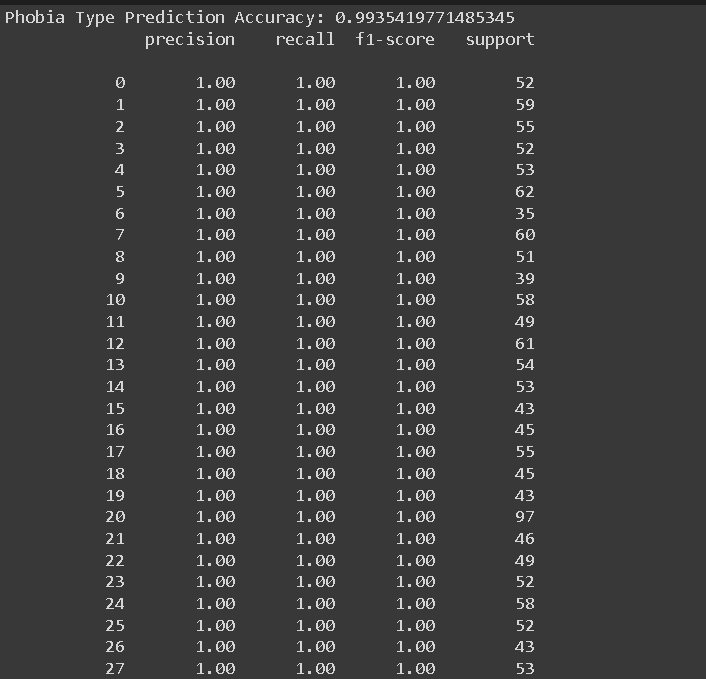
### Testing and Evaluation

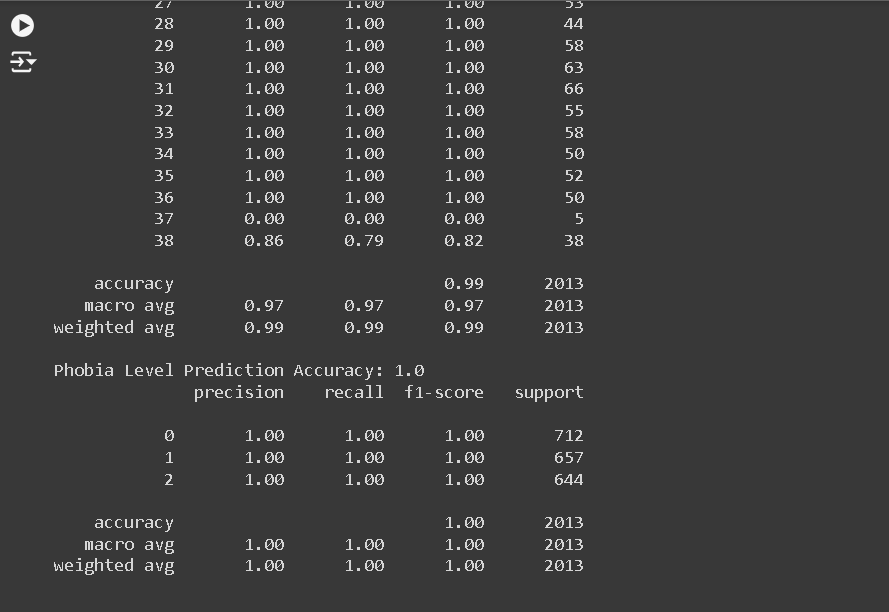
Testing and evaluation are critical steps in measuring the effectiveness of phobia detection models. By assessing their performance on unseen data, we ensure the models generalize well to new cases.

* + 1. **Testing**
* **Data Splitting**: The dataset was divided into three subsets: training (70%), validation (20%), and test (10%). The test set was used to perform an unbiased evaluation of the trained models.
* **Model Training**: During training, models were optimized based on a loss function (e.g., cross-entropy for classification tasks) to accurately predict phobia types and severity. The validation set was used to monitor overfitting and fine-tune hyperparameters like learning rate and model complexity.

### Evaluation

1. **Performance Metrics:** Specific metrics are used to measure the model's performance. Common metrics include accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). The choice of metrics depends on the problem type (classification, regression, etc.) and goals.
2. **Model Validation:** The model's performance is assessed on the validation set. This helps in tuning hyperparameters and detecting issues like overfitting, where the model performs well on the training data but poorly on unseen data.
3. **Hyperparameter Tuning:** Hyperparameters (e.g., learning rate, batch size, model architecture) are adjusted to optimize model performance. Techniques like grid search or random search can be used to find the best combination of hyperparameters.
4. **Cross-Validation :** Cross-validation involves splitting the dataset into multiple subsets (folds) and training and evaluating the model on different combinations of these subsets. It provides a more robust estimate of model performance.
5. **Overfitting Analysis:** Evaluating whether the model is overfitting (learning noise in the data) or underfitting (not capturing important patterns) is crucial. Regularization techniques and proper model architecture can help mitigate overfitting.
6. **Final Model Evaluation:** The model's performance is assessed on the test set, which it has never seen before. This evaluation provides an unbiased estimate of the model's generalization ability and how well it will perform on new, unseen data.
7. **Baseline Comparison:** Models are often compared to baseline models or simple algorithms to gauge their effectiveness. A good machine learning model should outperform these baselines.





#### Fig. 7.4.1 Evaluation results

# USER INTERFACE INTEGRATION

### Streamlit Interface Integration

Integrating a user interface (UI) using Streamlit involves creating an interactive, web-based application that allows users to interact with the phobia management system and view the predictions generated by the machine learning model. The steps to achieve this integration are as follows:

* **Installation:** Ensure that you have Streamlit installed in your Python environment. You can install it using the pip package manager:

***pip install streamlit***

* **UI Design and Layout:** Streamlit simplifies UI design by allowing you to create the layout using Python scripts. It provides various widgets such as text inputs, buttons, checkboxes, and sliders to build an interactive user interface. The UI for this project includes:

1. Input fields for demographic details, symptoms, and other relevant data.
2. A task list feature for users to complete daily steps related to phobia management.
3. A streak system to track user activity over time.
4. A dashboard for visualizing user progress and predictions.

* **Application Script:** The UI and backend are combined in a single Python script that uses Streamlit to generate the user interface and handle interactions. This script includes:

1. Methods for receiving user inputs.
2. Functions for processing data and interacting with the machine learning model.
3. Components for displaying predictions, such as phobia levels, task completions, and streaks.

* **Logic Implementation:** The backend logic for data processing and model predictions is implemented in Python, using pandas and SQLite for managing data, and a Random Forest model for making predictions. Streamlit widgets are bound to these functions to capture inputs and display results dynamically.
* **Binding UI Elements:** Streamlit’s declarative nature simplifies binding UI elements to functions. For example, user inputs for symptoms and demographics are linked directly to the phobia prediction model. Similarly, user activity is tracked and displayed as streaks using Streamlit’s calendar component and the streak logic.
* **Deployment:** The Streamlit app is deployed using streamlit.io/share, allowing users to access the application via a web browser. This makes the phobia management system widely accessible, without the need for installing software locally.
* **Execution:** To launch the user interface, the following command is executed:

***streamlit run app.py***

This command runs the application locally or on the cloud, displaying an interactive dashboard where users can input data, view predictions, and track their progress over time.

* **User Experience:** The UI has been designed to be simple and user-friendly, ensuring that users can easily input data, complete tasks, and view results without requiring extensive technical knowledge. It integrates various components like the streak system, task completion tracking, and phobia level predictions, making it a comprehensive platform for managing phobias.

# 9.SAMPLE CODE

### 9.1 DreadEase.py

import streamlit as st

import login

import create

import dashboardt

import dashboardl

import test

import database

import user

import daily\_tasks

import settings

import password

import splash

import major\_tasks

st.set\_page\_config(

page\_title="DreadEase",

layout="wide", # Use the entire screen width

#initial\_sidebar\_state="collapsed"

)

# Call the function to create the database and tables

database.create\_db()

# Get the current page from query parameters

query\_params = st.query\_params.to\_dict()

current\_page = query\_params.get('page', 'splash')

# Function to handle navigation using query parameters

def navigate\_to(page):

st.query\_params.from\_dict({"page": page})

# Display the appropriate page based on the query parameter

if current\_page == 'splash':

splash.splash\_page()

if current\_page == 'login':

login.login\_page(navigate\_to)

elif current\_page == 'password':

password.reset\_password\_page(navigate\_to)

elif current\_page == 'create':

create.create\_account\_page(navigate\_to)

elif current\_page == 'dashboardt':

dashboardt.dashboardt\_page()

elif current\_page == 'test':

test.prediction\_page()

elif current\_page == 'dashboardl':

dashboardl.dashboardl\_page()

elif current\_page == 'user':

user.profile()

elif current\_page == 'daily\_tasks':

daily\_tasks.daily\_tasks()

elif current\_page == 'settings':

settings.settings()

elif current\_page == 'major\_tasks':

major\_tasks.major\_tasks()

### 9.2 Splash.py

### import streamlit as st

### import time

### from PIL import Image,ImageDraw,ImageOps

### from streamlit\_extras.stylable\_container import stylable\_container

### from streamlit\_js\_eval import streamlit\_js\_eval as sj

### import os

### import requests

### def navigate\_to(page):

### st.query\_params.from\_dict({"page": page})

### def create\_rounded\_image(image, size=(800, 800)):

### image = image.resize(size, Image.LANCZOS)

### # Create a mask for the rounded corners

### mask = Image.new('L', size, 0)

### draw = ImageDraw.Draw(mask)

### draw.ellipse((0, 0, size[0], size[1]), fill=255)

### # Create a new image with a transparent background

### rounded\_image = ImageOps.fit(image, mask.size, Image.LANCZOS)

### rounded\_image.putalpha(mask)

### return rounded\_image

### def splash\_page():

### logo\_url = "https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/dreadease\_logo.png"

### response = requests.get(logo\_url, stream=True)

### logo = Image.open(response.raw)

### #logo = Image.open(logo\_path)

### rounded\_image = create\_rounded\_image(logo)

### col1, col2= st.columns([0.4, 1]) # Creates 3 equal columns

### with col2: # Use the center column

### st.image(rounded\_image, use\_column\_width=False, width=450)

### 

### time.sleep(3)

### navigate\_to('login')

### sj(js\_expressions="parent.window.location.reload()")

### 9.3 login.py

import streamlit as st

import sqlite3

import datetime

from streamlit\_js\_eval import streamlit\_js\_eval as sj

# Function to connect to the database

def connect\_db():

return sqlite3.connect('users.db')

# Function to check login credentials

def login\_user(email, password):

conn = connect\_db()

c = conn.cursor()

# Check the users table for valid credentials

c.execute('SELECT \* FROM users WHERE email=? AND password=?', (email, password))

user\_result = c.fetchone()

# If login is successful, check for account in user\_predictions table

if user\_result:

c.execute('SELECT \* FROM user\_predictions WHERE email=?', (email,))

account\_check = c.fetchone()

conn.close()

return True, account\_check is not None # Return login success and account existence

conn.close()

return False, False

# Function to save the login session and log the login date

def save\_login\_session(email):

conn = connect\_db()

c = conn.cursor()

# Log the login date

login\_date = datetime.datetime.now()

logout\_time=None

c.execute('INSERT INTO activity\_log (email, login\_date,logout\_time) VALUES (?,?,?)', (email, login\_date,logout\_time))

conn.commit()

conn.close()

def page\_footer():

# Add custom CSS for styling the footer

st.markdown(

"""

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #f1f1f1;

text-align: center;

padding: 10px 0;

font-size: 14px;

color: #333;

border-top: 1px solid #eaeaea;

}

.footer a {

color: #0366d6;

text-decoration: none;

}

.footer a:hover {

text-decoration: underline;

}

</style>

""",

unsafe\_allow\_html=True

)

# Add footer content

st.markdown(

"""<div class="footer">

<p>© 2024 DreadEase. All rights reserved.</p>

<p>Powered by Streamlit | <a href="mailto:dreadease.18@gmail.com" target="\_blank">Mail Id</a> |

<a href="https://github.com/Reethz30" target="\_blank">GitHub</a> |

<a href="https://www.linkedin.com/in/buddi-reethika-chovudary-3382a0255/" target="\_blank">LinkedIn</a> |

<a href="#contact\_us">Contact Us</a></p>

</div>

""",

unsafe\_allow\_html=True

)

# Login page layout function

def login\_page(navigate\_to):

st.title("Login Page")

# Email and Password Input

email = st.text\_input("Email", placeholder="Enter your email", key="email\_input")

password = st.text\_input("Password", placeholder="Enter your password", type="password")

if st.button("Login"):

if not email:

st.error("Email field is empty.")

elif not password:

st.error("Password field is empty.")

else:

login\_success, account\_exists = login\_user(email, password)

if login\_success:

if account\_exists:

st.success("Login successful!")

save\_login\_session(email) # Save session with expiration

navigate\_to('dashboardl') # Navigate to the dashboard page

sj(js\_expressions="parent.window.location.reload()")

else:

st.error("You have no account associated with this email.")

else:

st.error("Invalid email or password")

# Line to separate the options

st.markdown("---")

# Forgot Password and Create Account buttons

col1, col2 = st.columns(2)

with col1:

if st.button("Forgot Password"):

navigate\_to('password') # Navigate to the reset password page

sj(js\_expressions="parent.window.location.reload()")

with col2:

if st.button("Create Account"):

navigate\_to('create') # Navigate to the create account page

sj(js\_expressions="parent.window.location.reload()")

page\_footer()

### 9.4 Create.py

import streamlit as st

import sqlite3

import uuid

import datetime

import time

from streamlit\_js\_eval import streamlit\_js\_eval as sj

# Function to connect to the database

def connect\_db():

return sqlite3.connect('users.db')

# Function to create a new user

def create\_user(email, password, secret\_key):

conn = connect\_db()

c = conn.cursor()

try:

# Insert the user into the database

c.execute('INSERT INTO users (email, password, secret\_key) VALUES (?, ?, ?)', (email, password, secret\_key))

login\_date = datetime.datetime.now().date()

c.execute('INSERT INTO activity\_log (email, login\_date) VALUES (?, ?)', (email, login\_date))

conn.commit()

st.success("Account created successfully!")

return True

except sqlite3.IntegrityError:

# Handle the case where the email already exists

st.error("An account with this email already exists.")

return False

finally:

conn.close()

def page\_footer():

# Add custom CSS for styling the footer

st.markdown(

"""

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #f1f1f1;

text-align: center;

padding: 10px 0;

font-size: 14px;

color: #333;

border-top: 1px solid #eaeaea;

}

.footer a {

color: #0366d6;

text-decoration: none;

}

.footer a:hover {

text-decoration: underline;

}

</style>

""",

unsafe\_allow\_html=True

)

# Add footer content

st.markdown(

"""<div class="footer">

<p>© 2024 DreadEase. All rights reserved.</p>

<p>Powered by Streamlit | <a href="mailto:dreadease.18@gmail.com" target="\_blank">Mail Id</a> |

<a href="https://github.com/Reethz30" target="\_blank">GitHub</a> |

<a href="https://www.linkedin.com/in/buddi-reethika-chovudary-3382a0255/" target="\_blank">LinkedIn</a> |

<a href="#contact\_us">Contact Us</a></p>

</div>

""",

unsafe\_allow\_html=True

)

# Create Account page layout function

def create\_account\_page(navigate\_to):

st.title("Create Account")

# Input for new email and password

new\_email = st.text\_input("New Email", placeholder="Enter a new email")

new\_password = st.text\_input("New Password", placeholder="Enter a new password", type="password")

if st.button("Submit"):

if not new\_email:

st.error("Email field is empty.")

elif not new\_password:

st.error("Password field is empty.")

else:

# Generate a unique secret key for the user

st.write("The Secret Key is crucial to keep safe for future reference, as it is necessary for actions like password recovery and account verification.")

secret\_key = str(uuid.uuid4())

# Create the user and log them in if successful

if create\_user(new\_email, new\_password, secret\_key):

# Display the secret key for future password resets

with st.expander("Your Secret Key"):

st.write(f"Secret Key: {secret\_key}")

with st.spinner("Processing..."):

time.sleep(5)

# Redirect to the dashboard (or any other page)

navigate\_to('test') # Assuming 'dashboard' is the page to navigate after login

sj(js\_expressions="parent.window.location.reload()")

# Button to go back to the login page

if st.button("Back to Login"):

navigate\_to('login')

sj(js\_expressions="parent.window.location.reload()")

page\_footer()

### 9.5 Password.py

import streamlit as st

import sqlite3

from streamlit\_js\_eval import streamlit\_js\_eval as sj

# Function to connect to the database

def connect\_db():

return sqlite3.connect('users.db')

# Function to validate the secret key and email

def validate\_secret\_key(email, secret\_key):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT \* FROM users WHERE email=? AND secret\_key=?', (email, secret\_key))

result = c.fetchone()

conn.close()

return result

# Function to reset the password

def reset\_password(email, new\_password):

conn = connect\_db()

c = conn.cursor()

c.execute('UPDATE users SET password=? WHERE email=?', (new\_password, email))

conn.commit()

conn.close()

# Reset Password page layout

def reset\_password\_page(navigate\_to):

st.title("Reset Password")

email = st.text\_input("Email", placeholder="Enter your email")

secret\_key = st.text\_input("Secret Key", placeholder="Enter your secret key")

new\_password = st.text\_input("New Password", placeholder="Enter your new password", type="password")

col1, col2 = st.columns(2)

with col1:

if st.button("Reset Password"):

if not email:

st.error("Email field is empty.")

elif not secret\_key:

st.error("Secret Key field is empty.")

elif not new\_password:

st.error("New Password field is empty.")

else:

if validate\_secret\_key(email, secret\_key):

reset\_password(email, new\_password)

st.success("Password has been reset!")

navigate\_to('login') # Navigate to the login page after reset

sj(js\_expressions="parent.window.location.reload()")

else:

st.error("Invalid email or secret key.")

with col2:

if st.button("Back to Login"):

navigate\_to('login') # Navigate back to the login page

sj(js\_expressions="parent.window.location.reload()")

### 9.6 Dashboardt.py

import streamlit as st

import sqlite3

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from streamlit\_js\_eval import streamlit\_js\_eval as sj

def navigate\_to(page):

st.query\_params.from\_dict({"page": page})

# Function to connect to the database

def connect\_db():

return sqlite3.connect('users.db')

def retrieve\_users(email):

conn=connect\_db()

c = conn.cursor()

c.execute('SELECT email,name,age,gender,frequency,fear\_of,selected\_symptoms,duration,predicted\_phobia\_type,predicted\_phobia\_level FROM user\_predictions WHERE email=?', (email,))

result = c.fetchone()

conn.close()

return result

# Function to fetch the latest email from the account creation database

def fetch():

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT email FROM user\_predictions ORDER BY rowid DESC LIMIT 1')

email = c.fetchone()

conn.close()

return email[0] if email else None

def fetch\_name(email):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT name FROM user\_predictions WHERE email=?', (email,))

result = c.fetchone()

conn.close()

return result[0]

def insert\_users(email, name, age, gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level):

with connect\_db() as conn:

c = conn.cursor()

# Check if the email already exists

c.execute('SELECT email FROM dashboard\_users WHERE email=?', (email,))

existing\_email = c.fetchone()

if existing\_email:

c.execute('SELECT checked\_precautions,last\_checked\_date FROM dashboard\_users WHERE email=?',(email,))

#res=c.fetchone()

x,y=c.fetchone()

c.execute('''

DELETE FROM dashboard\_users WHERE email = ?

''', (email,))

c.execute('INSERT INTO dashboard\_users (email, name, age, gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level,checked\_precautions,last\_checked\_date) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)',

(email, name, age, gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level,x,y))

return

#st.warning("This email is already registered in the database.")

# If the email does not exist, insert the new user

c.execute('INSERT INTO dashboard\_users (email, name, age, gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level) VALUES (?,?,?,?,?,?,?,?,?,?)',

(email, name, age, gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level))

conn.commit()

import requests

def contact\_us():

st.subheader("Contact Us")

st.write("Have any questions or feedback? Please fill out the form below to reach us!")

# User provides their email and message

user\_email = st.text\_input("Your Email")

user\_message = st.text\_area("Your Message")

# Send message button

if st.button("Send Message"):

if user\_email and user\_message:

data = {

'user\_email': user\_email,

'user\_message': user\_message

}

try:

response = requests.post("http://192.168.31.228/dreadease/send\_email.php", data=data)

if response.status\_code == 200:

st.success("Your message has been sent successfully! We will get back to you soon.")

else:

st.error("Failed to send the message. Please try again later.")

except Exception as e:

st.error(f"An error occurred: {e}")

else:

st.error("Please fill in all fields.")

def fetch\_coins(email):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT coins FROM dashboard\_users WHERE email=?', (email,))

coins = c.fetchone()

conn.close()

return coins[0] if coins else 0

def update\_coins(email, new\_coins):

conn = connect\_db()

c = conn.cursor()

c.execute('UPDATE dashboard\_users SET coins = coins + ? WHERE email = ?', (new\_coins, email))

conn.commit()

conn.close()

def check\_precautions\_completed(email):

# Implement your logic here to check if all precautions have been completed

# Return True if completed, else False

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT checked\_precautions FROM dashboard\_users WHERE email=?', (email,))

completed = c.fetchone()

conn.close()

if completed and completed[0]: # Check if completed is not None

precautions\_list = completed[0].split(',') # Split by comma

return len(precautions\_list) >= 4 # Return True if all precautions are completed

return False # Assuming checked\_precautions is a text column

def fire\_streak(email):

if check\_precautions\_completed(email):

current\_coins = fetch\_coins(email)

new\_coins = 5 # Reward for completing all precautions

if current\_coins==None:

current\_coins=0

x=current\_coins+new\_coins

update\_coins(email, x)

y="🔥" \* (x // 5)

if (x//5)>=5:

y="❤️"\*(x//25)

st.success(y)#Fire Streak Activated! You've earned 5 coins!")

st.write(f"Total Coins: {x}")

else:

st.warning("Complete all precautions to activate Fire Streak!")

def page\_footer():

# Add custom CSS for styling the footer

st.markdown(

"""

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #f1f1f1;

text-align: center;

padding: 10px 0;

font-size: 14px;

color: #333;

border-top: 1px solid #eaeaea;

}

.footer a {

color: #0366d6;

text-decoration: none;

}

.footer a:hover {

text-decoration: underline;

}

</style>

""",

unsafe\_allow\_html=True

)

# Add footer content

st.markdown(

"""<div class="footer">

<p>© 2024 DreadEase. All rights reserved.</p>

<p>Powered by Streamlit | <a href="mailto:dreadease.18@gmail.com" target="\_blank">Mail Id</a> |

<a href="https://github.com/Reethz30" target="\_blank">GitHub</a> |

<a href="https://www.linkedin.com/in/buddi-reethika-chovudary-3382a0255/" target="\_blank">LinkedIn</a> |

<a href="#contact\_us">Contact Us</a></p>

</div>

""",

unsafe\_allow\_html=True

)

def fetch\_phobia\_data(email):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT predicted\_phobia\_type, predicted\_phobia\_level FROM user\_predictions WHERE email=?', (email,))

result = c.fetchone()

conn.close()

return result

def sidebar\_menu(email):

st.sidebar.title(f'Hello, {str(fetch\_name(email))}')

page = st.sidebar.radio("Go to", ("User Account", "Daily Tasks", "Settings"),index=None, key="sidebar\_menu")

#st.stop()

return page

def dashboardt\_page():

email = fetch() # Get email from create\_account database

#st.write("HIII")

email,name,age,gender,frequency,fear\_of,selected\_symptoms,duration,predicted\_phobia\_type,predicted\_phobia\_level=retrieve\_users(email)

insert\_users(email,name,age,gender,frequency,fear\_of,selected\_symptoms,duration,predicted\_phobia\_type,predicted\_phobia\_level)

'''if email:

st.title("Phobia Prediction Dashboard")'''

# Fire Streak Feature

col1, x,col2 = st.columns([1.5,0.4,0.5]) # Divide the page into two columns

with col1:

if email:

st.title("Phobia Prediction Dashboard")

#st.success(f"Welcome, {name}!")

# Display user data or other relevant information

'''st.write(f"Age: {age}")

st.write(f"Gender: {gender}")'''

st.success(f"Predicted Phobia Type: {predicted\_phobia\_type}")

st.success(f"Predicted Phobia Level: {predicted\_phobia\_level}")

else:

st.warning("No user found in account creation database.")

with col2:

if predicted\_phobia\_level != 'Major':

# Fire Streak Feature

st.write('\n')

st.write('\n')

st.write('\n')

fire\_streak(email) # Check and display fire streak

if st.button('User Account'):

navigate\_to('user')

sj(js\_expressions="parent.window.location.reload()")

'''else:

st.warning("No prediction has been made yet. Please go back and complete the prediction.")'''

'''else:

st.warning("No user found in account creation database.")'''

selected\_page = sidebar\_menu(email)

# Conditional logic for rendering the selected page

if selected\_page == "User Account":

navigate\_to('user')

sj(js\_expressions="parent.window.location.reload()")

elif selected\_page == "Daily Tasks":

if predicted\_phobia\_level=='Major':

navigate\_to('major\_tasks')

sj(js\_expressions="parent.window.location.reload()")

else:

navigate\_to('daily\_tasks')

sj(js\_expressions="parent.window.location.reload()")

elif selected\_page == "Settings":

#st.stop()

navigate\_to('settings')

sj(js\_expressions="parent.window.location.reload()")

contact\_us()

page\_footer()

### 9.7 User.py

import streamlit as st

import sqlite3

from datetime import datetime, timedelta

import calendar

from PIL import Image, ImageDraw,ImageOps

from streamlit\_extras.stylable\_container import stylable\_container

from streamlit\_js\_eval import streamlit\_js\_eval as sj

import os

import requests

def navigate\_to(page):

st.query\_params.from\_dict({"page": page})

# Function to connect to the database

def connect\_db():

return sqlite3.connect('users.db')

def fetch\_sec(email):

conn=connect\_db()

c=conn.cursor()

c.execute('SELECT secret\_key FROM users WHERE email=?',(email,))

sec=c.fetchone()

conn.close()

return sec[0] if sec else None

def fetch():

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT email FROM dashboard\_users ORDER BY rowid DESC LIMIT 1')

email= c.fetchone()

conn.close()

return email[0] if email else None

# Function to fetch user data from user\_predictions

def fetch\_user\_data(email):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT name, age,gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level FROM user\_predictions WHERE email=?', (email,))

result = c.fetchone()

conn.close()

return result

# Function to fetch user activity streak

def fetch\_activity\_streak(email):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT login\_date FROM activity\_log WHERE email=? ORDER BY login\_date DESC', (email,))

activity\_dates = [row[0] for row in c.fetchall()]

conn.close()

streak\_days = {}

#st.write(activity\_dates)

for date in activity\_dates:

streak\_days[date] = True # Mark the date as active

return streak\_days

def display\_calendar(year, month, active\_days):

# Get the first day of the month and the number of days in the month

first\_day = datetime(year, month, 1)

last\_day = (first\_day + timedelta(days=31)).replace(day=1) - timedelta(days=1)

days\_in\_month = [first\_day + timedelta(days=i) for i in range((last\_day - first\_day).days + 1)]

active\_days\_set = set(active\_days)

# Display the calendar

calendar\_html = '<table style="border-collapse: collapse;">'

calendar\_html += '<tr>' + ''.join(['<th style="border: 1px solid black; padding: 5px; ">' + day.strftime('%a') + '</th>' for day in days\_in\_month[:7]]) + '</tr>'

for week in range(0, len(days\_in\_month), 7):

calendar\_html += '<tr>'

for day in days\_in\_month[week:week + 7]:

is\_active = day.strftime('%Y-%m-%d') in active\_days\_set

cell\_style = f"background-color: {'lightgreen' if is\_active else 'white'}; color: black;"

calendar\_html += f'<td style="border: 1px solid black; padding: 5px; {cell\_style}">{day.day}</td>'

calendar\_html += '</tr>'

calendar\_html += '</table>'

st.markdown(calendar\_html, unsafe\_allow\_html=True)

# Usage in your user profile page

def create\_rounded\_image(image, size=(300, 300)):

image = image.resize(size, Image.LANCZOS)

# Create a mask for the rounded corners

mask = Image.new('L', size, 0)

draw = ImageDraw.Draw(mask)

draw.ellipse((0, 0, size[0], size[1]), fill=255)

# Create a new image with a transparent background

rounded\_image = ImageOps.fit(image, mask.size, Image.LANCZOS)

rounded\_image.putalpha(mask)

return rounded\_image

def page\_footer():

# Add custom CSS for styling the footer

st.markdown(

"""

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #f1f1f1;

text-align: center;

padding: 10px 0;

font-size: 14px;

color: #333;

border-top: 1px solid #eaeaea;

}

.footer a {

color: #0366d6;

text-decoration: none;

}

.footer a:hover {

text-decoration: underline;

}

</style>

""",

unsafe\_allow\_html=True

)

# Add footer content

st.markdown(

"""<div class="footer">

<p>© 2024 DreadEase. All rights reserved.</p>

<p>Powered by Streamlit | <a href="mailto:dreadease.18@gmail.com" target="\_blank">Mail Id</a> |

<a href="https://github.com/Reethz30" target="\_blank">GitHub</a> |

<a href="https://www.linkedin.com/in/buddi-reethika-chovudary-3382a0255/" target="\_blank">LinkedIn</a> |

<a href="#contact\_us">Contact Us</a></p>

</div>

""",

unsafe\_allow\_html=True

)

def profile():

st.title("User Profile")

with stylable\_container(

key="user\_b",

css\_styles="""\*{

font-size:15px;

/\*background-color:#e5a7b3;\*/}""",

):

email = fetch()

user\_data = fetch\_user\_data(email) # Get active days for the logged-in user

streak\_days = fetch\_activity\_streak(email)

#st.title("User Profile")

col1, x,col2 = st.columns([1.2,0.1,1])

if user\_data:

name, age,gender, frequency, fear\_of, selected\_symptoms, duration, predicted\_phobia\_type, predicted\_phobia\_level = user\_data

with col1:

st.subheader("User Details")

if selected\_symptoms:

# Remove extra spaces and split by comma

selected\_symptoms = selected\_symptoms.replace("'", "").strip()

selected\_symptoms\_list = [symptom.strip() for symptom in selected\_symptoms.split(',') if symptom.strip()]

#st.subheader("User Details")

st.write(f"\*\*Name:\*\* {name}")

st.write(f"\*\*Age:\*\* {age}")

st.write(f"\*\*Gender:\*\* {gender}")

st.write(f"\*\*Frequency:\*\* {frequency}")

st.write(f"\*\*Fear Of:\*\* {fear\_of}")

st.write(f"\*\*Selected Symptoms:\*\* {selected\_symptoms}")

st.write(f"\*\*Duration:\*\* {duration}")

st.write(f"\*\*Predicted Phobia Type:\*\* {predicted\_phobia\_type}")

st.write(f"\*\*Predicted Phobia Level:\*\* {predicted\_phobia\_level}")

with col2:

st.subheader("Upload Profile Photo")

# Default image based on gender

default\_image\_path = ""

if gender.lower() == "male":

default\_image\_path = "https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/2.png" # Update with your image path

elif gender.lower() == "female":

default\_image\_path = "https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/1.png" # Update with your image path

else:

#st.write("hiii")

default\_image\_path = "https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/3.jpeg" # Default for others or no gender specified

# File uploader for user to upload their own picture

#default\_image = st.image(default\_image\_path, caption="Default Profile Photo", use\_column\_width=False, width=150)

response = requests.get(default\_image\_path, stream=True)

default\_image = Image.open(response.raw)

rounded\_image = create\_rounded\_image(default\_image)

st.image(rounded\_image, caption="Default Profile Photo", use\_column\_width=False, width=150)

st.subheader("Activity Streak")

current\_year = datetime.now().year

current\_month = datetime.now().month

year = st.selectbox("Select Year", range(2023, current\_year + 1), index=current\_year-2023)

month\_names = list(calendar.month\_name)[1:] # Exclude the first entry (empty string)

month\_mapping = {name: index for index, name in enumerate(month\_names, start=1)}

# User selects month by name

month\_name = st.selectbox("Select Month", month\_names,index=current\_month-1)

month = month\_mapping[month\_name]

display\_calendar(year,month,streak\_days)

st.subheader("Account Details:")

st.write(f"\*\*Email:\*\* {email}")

st.write(f"\*\*Secret Key:\*\* {fetch\_sec(email)}")

st.write("\n")

st.write("\n")

st.write("\n")

# Now use the standard Streamlit button

with stylable\_container(

key="user\_home",

css\_styles="""

div[data-testid="stButton"]{

display: block;

margin: 0 auto;

padding: 0px 2px; /\* Reduced padding for height and width \*/

text-align: center;

text-decoration: none; /\* No underline \*/

font-size: 25px; /\* Smaller font size \*/

border-radius: 5px; /\* Rounded corners \*/

cursor: pointer; /\* Pointer on hover \*/

width: 100px;

border: #45a049;/\* Set a fixed width \*/}

div[data-testid="stButton"] button:hover{

background-color: #45a049;

color: 2px solid #ffffff;}""",

):

if st.button("Go to Home"):

navigate\_to('dashboardl')

sj(js\_expressions="parent.window.location.reload()")

else:

st.warning("No user data found.")

page\_footer()

### 9.8 Daily\_tasks.py

import streamlit as st

import sqlite3

import pandas as pd

from datetime import datetime

from streamlit\_js\_eval import streamlit\_js\_eval as sj

def navigate\_to(page):

st.query\_params.from\_dict({"page": page})

# Function to connect to the database

def connect\_db():

return sqlite3.connect('users.db')

def fetch():

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT email FROM dashboard\_users ORDER BY rowid DESC LIMIT 1')

email = c.fetchone()

conn.close()

return email[0] if email else None

def fetch\_phobia\_data(email):

conn = connect\_db()

c = conn.cursor()

c.execute('SELECT predicted\_phobia\_type, predicted\_phobia\_level FROM user\_predictions WHERE email=?', (email,))

result = c.fetchone()

conn.close()

return result

def load\_precautions():

# Load the Excel file

excel\_file\_path = 'https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/symptoms.xlsx'

df = pd.read\_excel(excel\_file\_path)

return df

def page\_footer():

# Add custom CSS for styling the footer

st.markdown(

"""

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #f1f1f1;

text-align: center;

padding: 10px 0;

font-size: 14px;

color: #333;

border-top: 1px solid #eaeaea;

}

.footer a {

color: #0366d6;

text-decoration: none;

}

.footer a:hover {

text-decoration: underline;

}

</style>

""",

unsafe\_allow\_html=True

)

# Add footer content

st.markdown(

"""<div class="footer">

<p>© 2024 DreadEase. All rights reserved.</p>

<p>Powered by Streamlit | <a href="mailto:dreadease.18@gmail.com" target="\_blank">Mail Id</a> |

<a href="https://github.com/Reethz30" target="\_blank">GitHub</a> |

<a href="https://www.linkedin.com/in/buddi-reethika-chovudary-3382a0255/" target="\_blank">LinkedIn</a> |

<a href="#contact\_us">Contact Us</a></p>

</div>

""",

unsafe\_allow\_html=True

)

def fetch\_checked\_precautions(email):

conn = connect\_db()

c = conn.cursor()

'''c.execute('SELECT age,fear\_of,last\_checked\_date FROM dashboard\_users WHERE email=? ',(email,))

res=c.fetchone()

st.write(res)'''

c.execute('''SELECT name,checked\_precautions, last\_checked\_date FROM dashboard\_users WHERE email=?''', (email,))

result = c.fetchone()

#st.write(f"Fetched data for {email}: {result}")

conn.close()

# Return (None, None) if no results, or if either checked\_precautions or last\_checked\_date is missing

if result:# and result[0] and result[1]:

return result

return (None, None)

def update\_checked\_precautions(email, checked\_precautions):

conn = connect\_db()

c = conn.cursor()

checked\_precautions=','.join(checked\_precautions)

st.write(checked\_precautions)

c.execute('UPDATE dashboard\_users SET checked\_precautions=?, last\_checked\_date=? WHERE email=?',

(checked\_precautions, datetime.now().date(), email))

conn.commit()

conn.close()

def daily\_tasks():

email = fetch() # Fetch the user's email

if email:

phobia\_data = fetch\_phobia\_data(email)

if phobia\_data:

phobia\_type, phobia\_level = phobia\_data

st.title("Daily Tasks to Overcome Phobia")

# Load the precautions from the Excel sheet

precaution\_df = load\_precautions()

if precaution\_df.empty:

st.warning("No precautions data loaded from Excel.")

return

phobia\_type = str(phobia\_type.strip().lower())

phobia\_level = phobia\_level.lower()

precaution\_df['phobia'] = precaution\_df['phobia'].str.strip()

precaution\_df['phobia'] = precaution\_df['phobia'].str.lower()

precaution\_df['level'] = precaution\_df['level'].str.lower()

#st.write(precaution\_df,precaution\_df['phobia'],precaution\_df['level'])

#test\_filter = precaution\_df[precaution\_df['phobia'] == 'nomophobia']

#st.write("Test Filter Results:", test\_filter, phobia\_type)

precautions\_df=precaution\_df

filtered\_precautions = precautions\_df[

(precautions\_df['level'] == phobia\_level)]

#st.write("Filtered Precautions:", filtered\_precautions)

filtered\_precautions = filtered\_precautions[(precautions\_df['phobia']==phobia\_type)]

#st.write("Filtered Precautions:", filtered\_precautions)

# Fetch previously checked precautions and the last checked date

x,checked\_precautions, last\_checked\_date = fetch\_checked\_precautions(email)

#st.write(x,checked\_precautions)

# Reset checked precautions if the day has changed

if last\_checked\_date and str(last\_checked\_date) != str(datetime.now().date()):

#st.write(str(datetime.now().date()))

checked\_precautions = [] # Clear checked precautions for the new day

update\_checked\_precautions(email, checked\_precautions)

# Display the precautions as checkboxes

if not filtered\_precautions.empty:

# Convert checked\_precautions from a comma-separated string to a list

checked\_precautions = checked\_precautions.split(',') if checked\_precautions else []

#st.write(checked\_precautions)

current\_checked = []

for index, row in filtered\_precautions.iterrows():

for i in range(1, 5): # Loop through precaution1 to precaution4

precaution = row.get(f'precaution{i}')

if pd.notna(precaution): # Ensure the precaution exists (not NaN)

# If the precaution was previously checked, lock it

if precaution in checked\_precautions:

st.checkbox(

precaution,

key=f'checkbox\_{index}\_{i}',

value=True,

disabled=True # Lock the checkbox if it was previously checked

)

else:

# If not previously checked, allow it to be checked

checkbox\_state = st.checkbox(

precaution,

key=f'checkbox\_{index}\_{i}',

value=False

)

# Store current checked state in a list

if checkbox\_state:

current\_checked.append(precaution)

# Save button

if st.button("Save Checked Precautions"):

# Update the database with the newly and previously checked precautions

#st.write(checked\_precautions)

update\_checked\_precautions(email, current\_checked + checked\_precautions)

sj(js\_expressions="parent.window.location.reload()")

#st.success("Your checked precautions have been saved!")

# Check if all precautions are checked

if len(current\_checked) + len(checked\_precautions) >= 4:

st.success("Great job! You've completed all your tasks! Keep it up!")

st.balloons()

else:

st.warning("Keep going! Every small step you take brings you closer to overcoming your phobia.")

else:

st.warning("No precautions available for your phobia.")

else:

st.warning("No phobia data found.")

else:

st.warning("No user found in the database.")

if st.button("Back to Home"):

navigate\_to('dashboardl')

sj(js\_expressions="parent.window.location.reload()")

page\_footer()

### 9.9 Major\_tasks.py

import pandas as pd

import sqlite3

import streamlit as st

from streamlit\_js\_eval import streamlit\_js\_eval as sj

def navigate\_to(page):

st.query\_params.from\_dict({"page": page})

def connect\_db():

return sqlite3.connect('users.db')

def fetch\_email():

conn = connect\_db()

cursor = conn.cursor()

cursor.execute('SELECT email FROM dashboard\_users ORDER BY rowid DESC LIMIT 1')

email = cursor.fetchone()

conn.close()

return email[0] if email else None

def major\_tasks():

st.title("DreadEase - Consultant Finder")

excel\_file\_path = 'https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/Major\_Links.xlsx'

df = pd.read\_excel(excel\_file\_path)

email = fetch\_email()

if not email:

st.error("No user email found.")

return

available\_states = df['Location'].unique().tolist()

selected\_state = st.selectbox('Select your state:', available\_states)

state\_consultants = df[df['Location'] == selected\_state]

if state\_consultants.empty:

st.warning("No consultants found for the selected state.")

return

st.write(f"## Consultants in {selected\_state}")

for index, consultant in state\_consultants.head(3).iterrows():

st.write(f"\*\*Consultant\*\*: {consultant['Suffix']} {consultant['Consultant']}")

st.write(f"\*\*Link\*\*: [Click here]({consultant['Link']})")

st.write("---")

if st.button('Back to Home'):

navigate\_to('dashboardl')

sj(js\_expressions="parent.window.location.reload()")

### 9.10 Settings.py

import streamlit as st

import sqlite3

import time

from PIL import Image, ImageOps, ImageDraw

from datetime import datetime

from streamlit\_extras.stylable\_container import stylable\_container

from streamlit\_js\_eval import streamlit\_js\_eval as sj

import os

import requests

def navigate\_to(page):

st.query\_params.from\_dict({"page": page})

# Database connection

def connect\_db():

return sqlite3.connect('users.db')

# Function to verify if the secret key matches

def verify\_secret\_key(email, secret\_key):

conn=connect\_db()

c=conn.cursor()

c.execute("SELECT secret\_key FROM users WHERE email = ?", (email,))

data = c.fetchone()

if data:

return data[0] == secret\_key

return False

# Logout functionality

def logout\_user(email):

if st.button("Logout"):

conn=connect\_db()

c=conn.cursor()

logout\_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S') # Get current time

c.execute("UPDATE activity\_log SET logout\_time = ? WHERE email = ?", (logout\_time, email))

conn.commit()

conn.close()

with st.expander("Notification", expanded=True):

st.success("Logged out successfully!")

time.sleep(3)

navigate\_to('splash')

sj(js\_expressions="parent.window.location.reload()")

# Delete account functionality

def delete\_account(email):

conn=connect\_db()

c=conn.cursor()

if st.button("Delete Account"):

c.execute("DELETE FROM users WHERE email = ?", (email,))

conn.commit()

c.execute("DELETE FROM user\_predictions WHERE email=?",(email,))

conn.commit()

c.execute("DELETE FROM activity\_log WHERE email=?",(email,))

conn.commit()

c.execute("DELETE FROM dashboard\_users WHERE email=?",(email,))

conn.commit()

with st.expander("Notification", expanded=True):

st.success("Deleted Account successfully!")

time.sleep(3)

navigate\_to('splash')

sj(js\_expressions="parent.window.location.reload()")

# Change password functionality using secret key

def change\_password(email):

secret\_key\_input = st.text\_input("Enter your secret key", type='password')

new\_password = st.text\_input("Enter new password", type='password')

confirm\_password = st.text\_input("Confirm new password", type='password')

if st.button("Change Password"):

if new\_password != confirm\_password:

st.error("Passwords do not match.")

elif not verify\_secret\_key(email, secret\_key\_input):

st.error("Secret key is incorrect.")

else:

st.expander(f"Your New Password: {new\_password}")

conn=connect\_db()

c=conn.cursor()

c.execute("UPDATE users SET password = ? WHERE email = ?", (new\_password, email))

conn.commit()

st.success("Password changed successfully!")

sj(js\_expressions="parent.window.location.reload()")

def create\_rounded\_image(image, size=(400, 400)):

image = image.resize(size, Image.LANCZOS)

# Create a mask for the rounded corners

mask = Image.new('L', size, 0)

draw = ImageDraw.Draw(mask)

draw.ellipse((0, 0, size[0], size[1]), fill=255)

# Create a new image with a transparent background

rounded\_image = ImageOps.fit(image, mask.size, Image.LANCZOS)

rounded\_image.putalpha(mask)

return rounded\_image

def fetch\_email():

conn=connect\_db()

c=conn.cursor()

c.execute('SELECT email FROM dashboard\_users ORDER BY rowid DESC LIMIT 1')

email = c.fetchone()

conn.close()

return email[0] if email else None

def page\_footer():

# Add custom CSS for styling the footer

st.markdown(

"""

<style>

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: #f1f1f1;

text-align: center;

padding: 10px 0;

font-size: 14px;

color: #333;

border-top: 1px solid #eaeaea;

}

.footer a {

color: #0366d6;

text-decoration: none;

}

.footer a:hover {

text-decoration: underline;

}

</style>

""",

unsafe\_allow\_html=True

)

# Add footer content

st.markdown(

"""<div class="footer">

<p>© 2024 DreadEase. All rights reserved.</p>

<p>Powered by Streamlit | <a href="mailto:dreadease.18@gmail.com" target="\_blank">Mail Id</a> |

<a href="https://github.com/Reethz30" target="\_blank">GitHub</a> |

<a href="https://www.linkedin.com/in/buddi-reethika-chovudary-3382a0255/" target="\_blank">LinkedIn</a> |

<a href="/contact">Contact Us</a></p>

</div>

""",

unsafe\_allow\_html=True

)

# Settings page implementation

def settings():

st.title("Settings")

email = fetch\_email()

if email:

# Create two columns

col1, col2 = st.columns([1, 1]) # Adjust the column width if needed

# Column 1: Logout and Delete Account buttons

with col1:

st.subheader("Logout")

logout\_user(email)

st.subheader("Delete Account")

delete\_account(email)

# Column 2: Logo display

with col2:

# Load and display the logo

logo\_url = "https://raw.githubusercontent.com/Reethz30/DreadEase/main/Code/dreadease\_logo.png"

response = requests.get(logo\_url, stream=True)

logo = Image.open(response.raw) # Replace with the actual path to your logo

rounded\_image = create\_rounded\_image(logo)

st.image(rounded\_image, use\_column\_width=False, width=250)

# Below both columns: Change Password option

st.subheader("Change Password")

change\_password(email)

else:

st.error("You need to be logged in to access settings.")

if st.button('Back to Home'):

navigate\_to('dashboardl')

sj(js\_expressions="parent.window.location.reload()")

page\_footer()

### 9.11 Database.py

import sqlite3

def create\_db():

conn = sqlite3.connect('users.db')

c = conn.cursor()

# Create table for users

c.execute('''

CREATE TABLE IF NOT EXISTS users (

email TEXT PRIMARY KEY,

password TEXT NOT NULL,

secret\_key TEXT

)

''')

# Create table for sessions

c.execute('''

CREATE TABLE IF NOT EXISTS activity\_log (

id INTEGER PRIMARY KEY AUTOINCREMENT,

email TEXT NOT NULL,

login\_date TEXT NOT NULL,

logout\_time TEXT

)

''')

#c.execute('''ALTER TABLE activity\_log ADD COLUMN logout\_time TEXT''')

c.execute('''

CREATE TABLE IF NOT EXISTS user\_predictions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

email TEXT NOT NULL,

name TEXT,

age INTEGER,

gender TEXT,

frequency TEXT,

fear\_of TEXT,

selected\_symptoms TEXT,

duration TEXT,

predicted\_phobia\_type TEXT,

predicted\_phobia\_level TEXT,

timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (email) REFERENCES users (email)

)

''')

c.execute('''

CREATE TABLE IF NOT EXISTS dashboard\_users(

id INTEGER PRIMARY KEY AUTOINCREMENT,

email TEXT NOT NULL UNIQUE,

name TEXT,

age INTEGER,

gender TEXT,

frequency TEXT,

fear\_of TEXT,

selected\_symptoms TEXT,

duration TEXT,

predicted\_phobia\_type TEXT,

predicted\_phobia\_level TEXT,

checked\_precautions TEXT,

last\_checked\_date DATE,

timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

coins INTEGER ,

FOREIGN KEY (email) REFERENCES users (email))''')

#c.execute('''ALTER TABLE dashboard\_users ADD COLUMN coins INTEGER''')

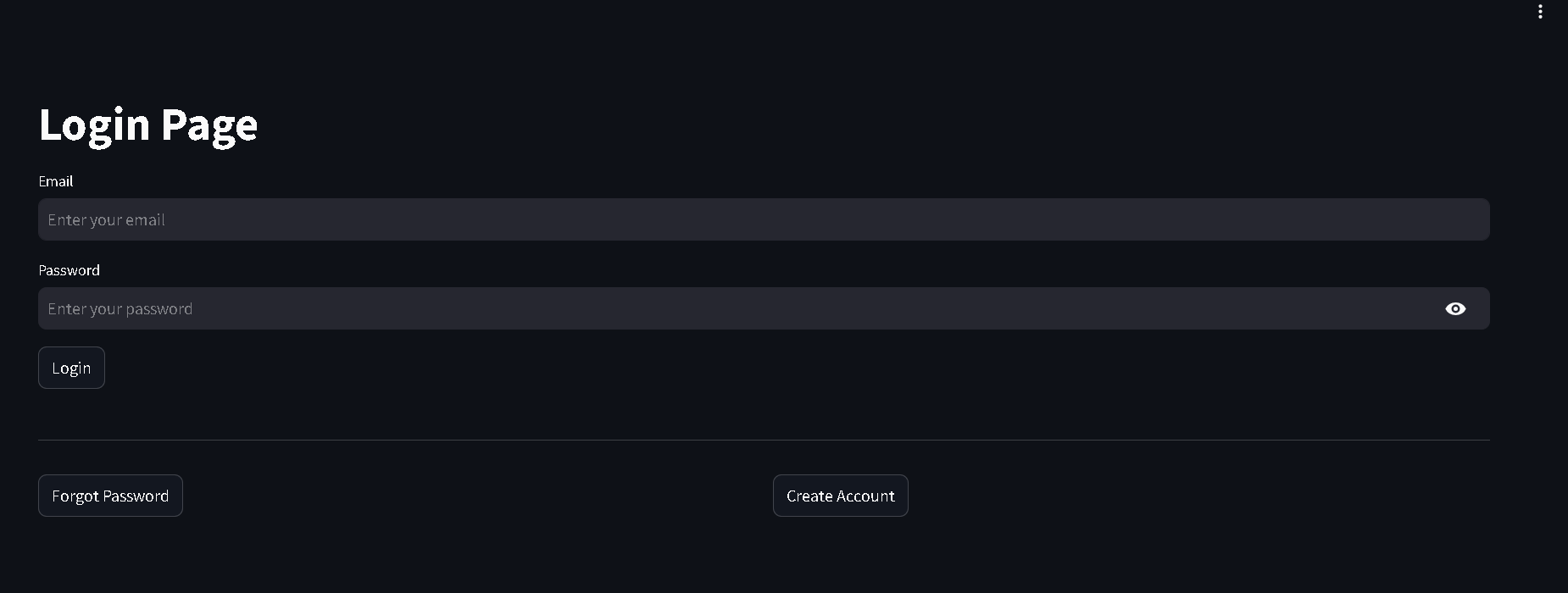
# 10.OUTPUT SCREENS

### 10.1 Logo page



#### Fig. 10.1 Logo Page

### 10.2 Login Page



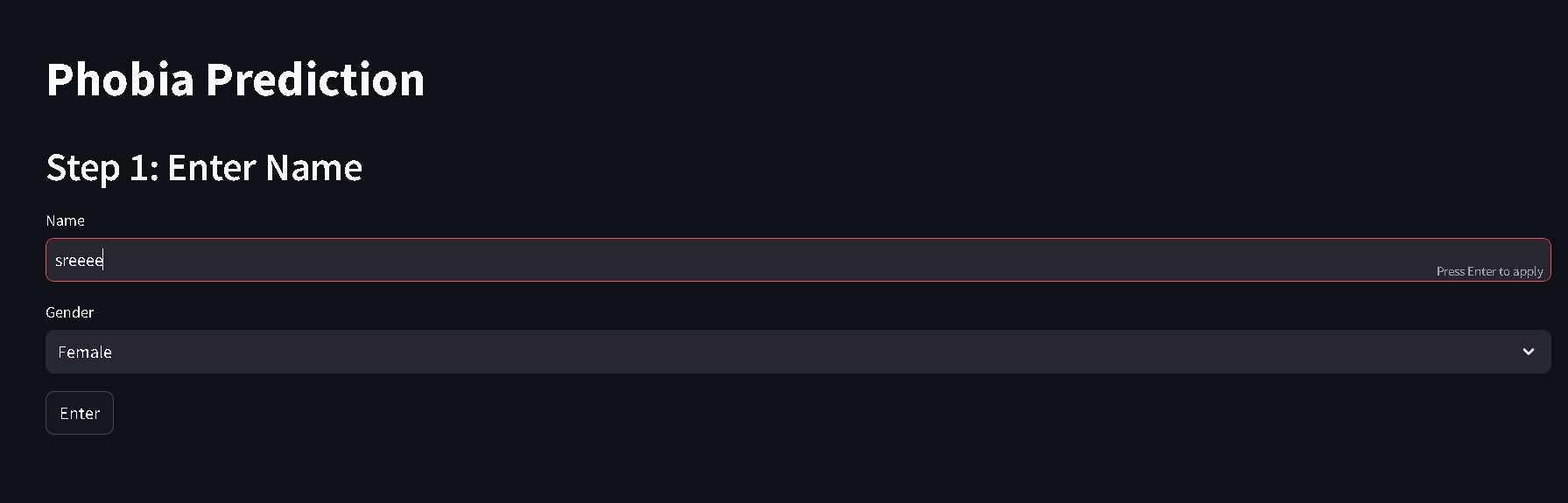
#### Fig. 10.2 Login Page

### 10.3 SignUp Page



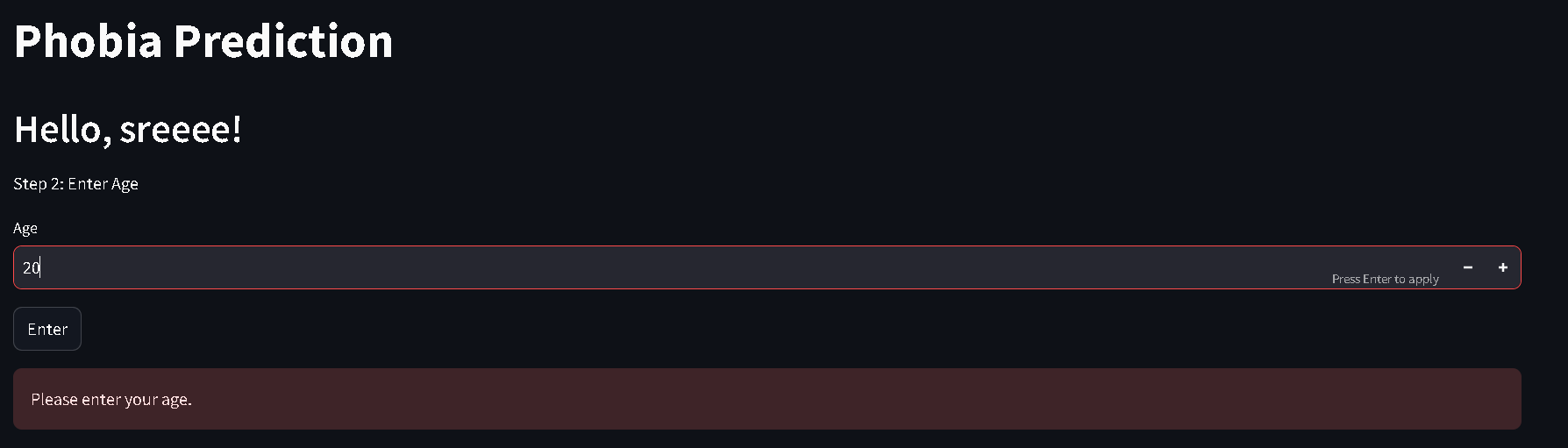
**Fig. 10.3 Signup/ Registration page**

### 10.4 Step-1 Page



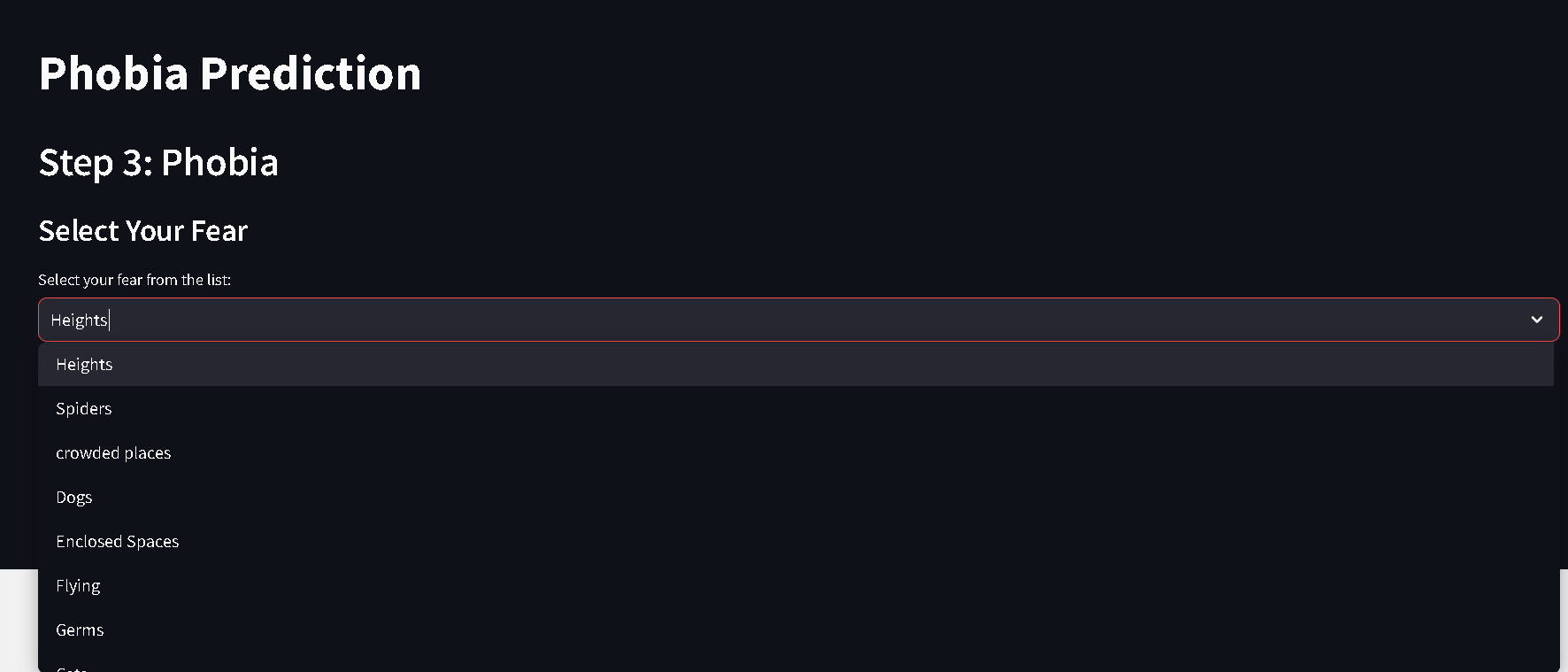
**Fig. 10.4 Step-1 page**

### 10.5 Step-2 Page



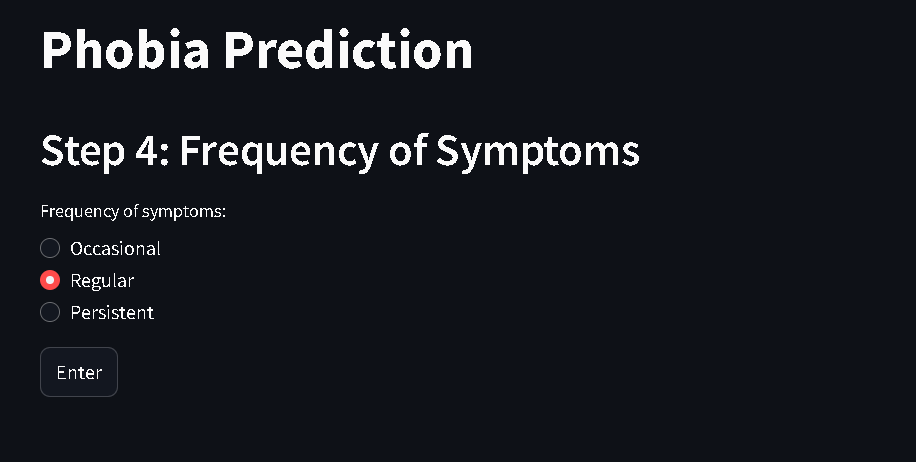
**Fig. 10.5 Step-2 page**

### 10.6 Step-3 Page



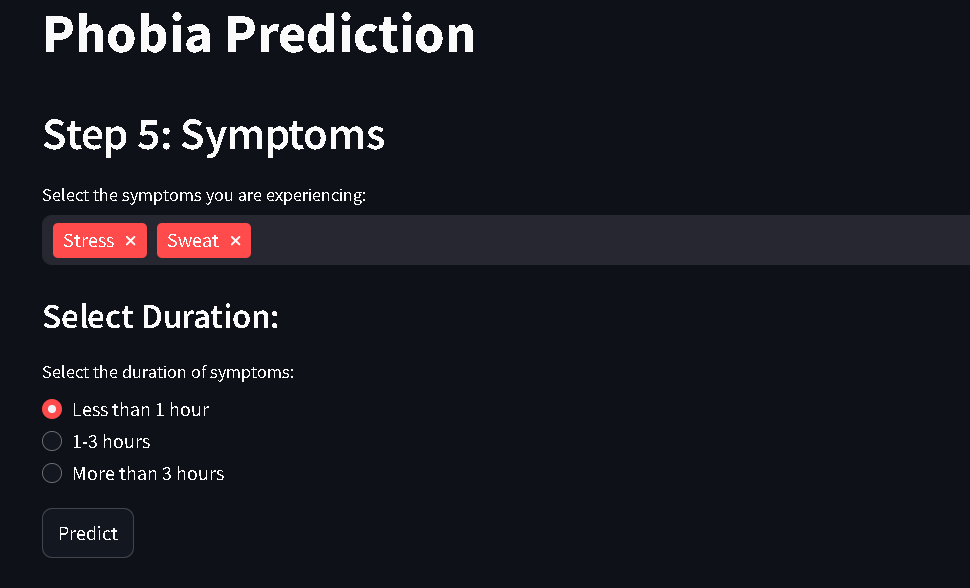
**Fig. 10.6 Step-3 page**

### 10.7 Step-4 Page



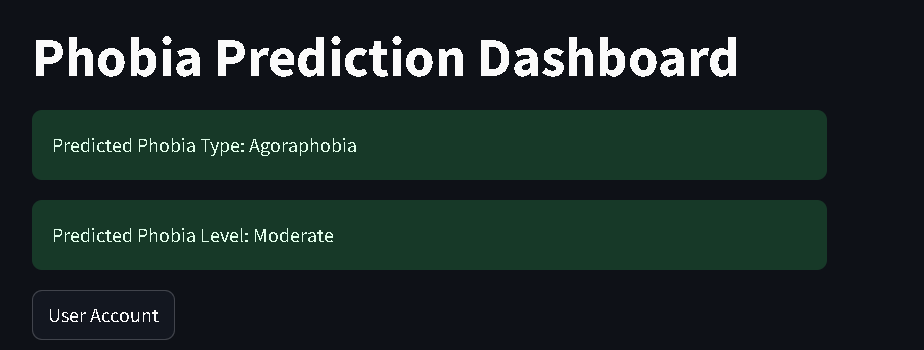
**Fig. 10.7 Step-4 page**

### 10.8 Step-5 Page



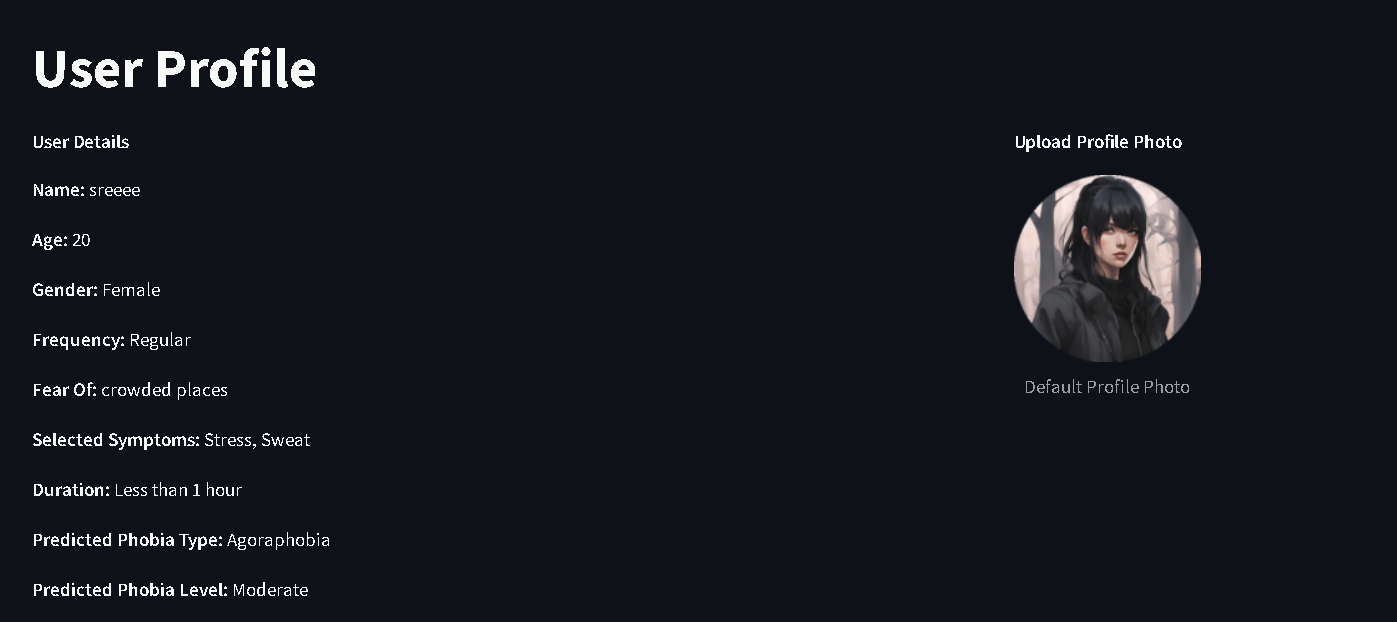
**Fig. 10.8 Step-5 page**

### 10.9 Phobia prediction dashbord

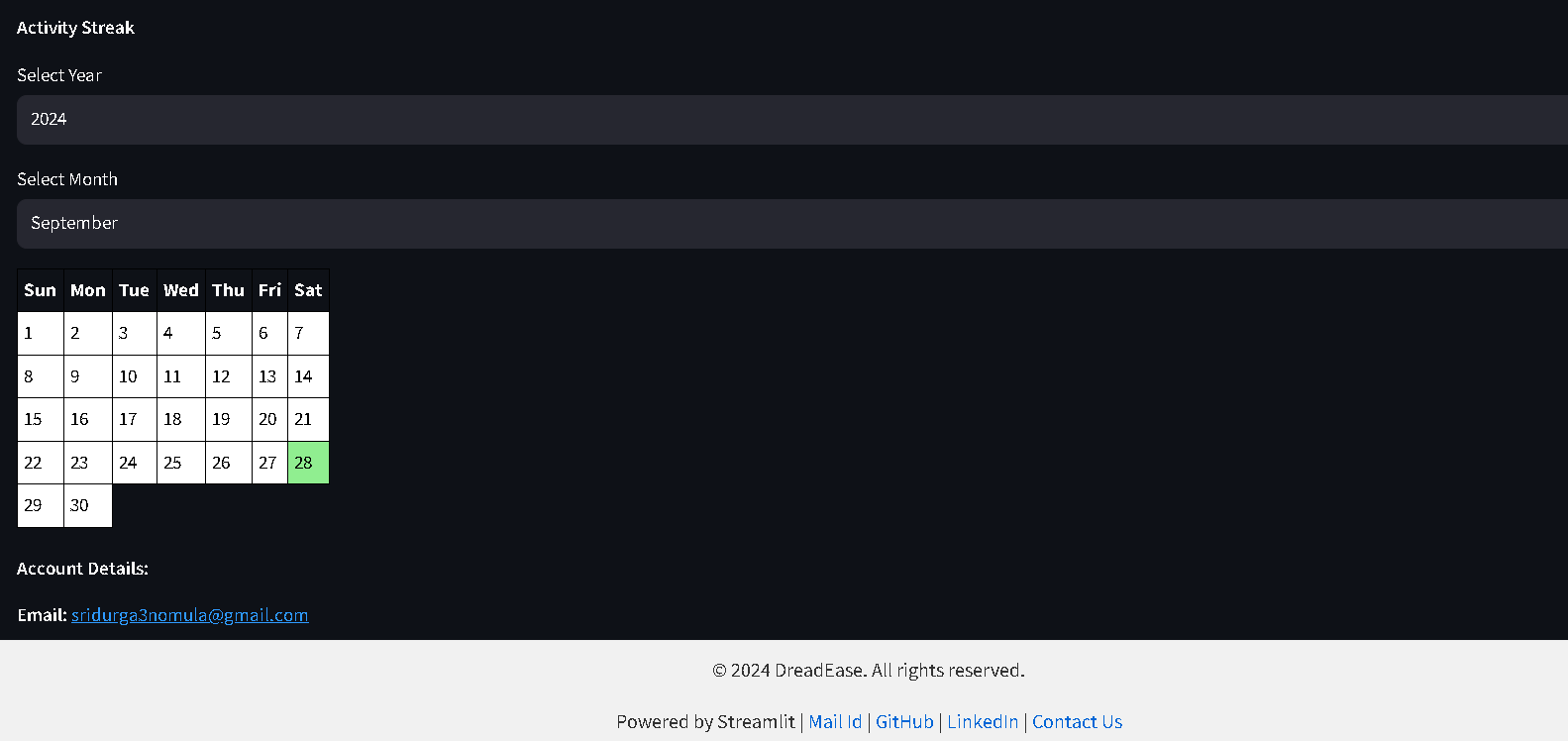


**Fig. 10.9 phobia prediction dashboard**

### 10.10 User Profile page

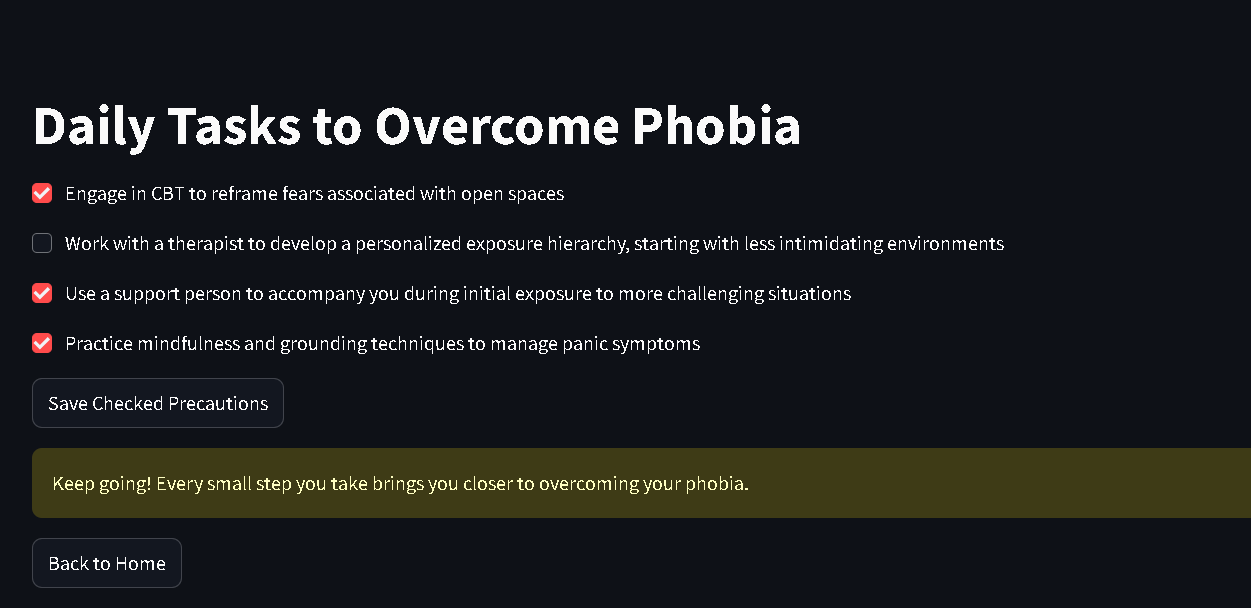


**Fig. 10.10 .1User profile page**



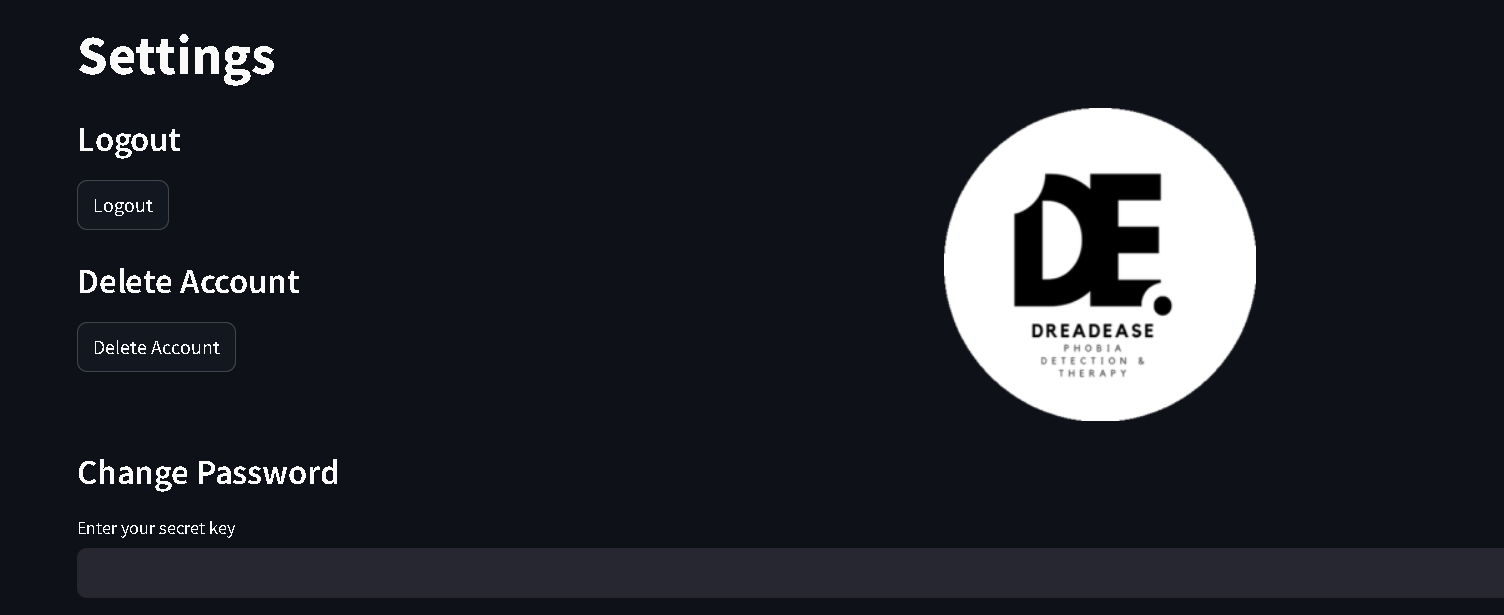
**Fig. 10.10 .2User profile page**

### 10.11 Daily tasks Page

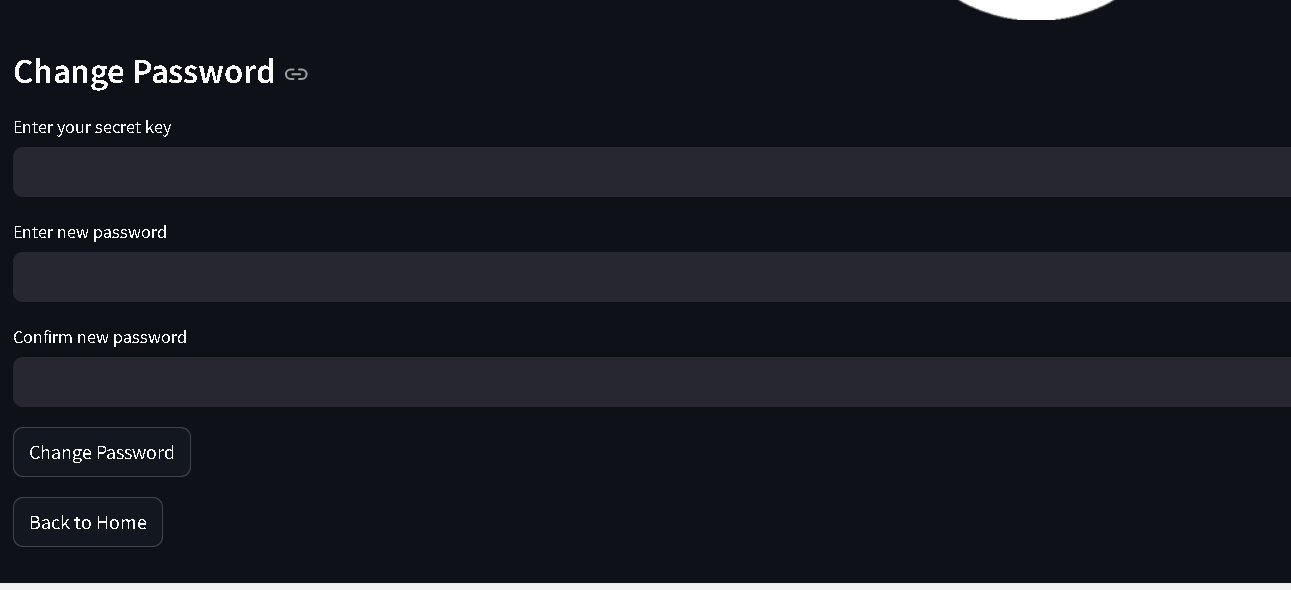


**Fig. 10.11 Daily tasks page**

### 10.12 Settings Page



**Fig. 10.12 .1 setting page**



**Fig. 10.12 .2 setting page**

# 11.CONCLUSION

In conclusion, this initiative represents a big step forward in tackling the severe issues faced by mental health concerns, particularly depression and panic attacks, which are becoming increasingly common in today's society. Our new web application uses cutting-edge machine learning techniques to provide individualized support to those struggling with these challenges, bridging the gap between diagnostic tools and treatment resources in the field of psychology.

The project's goals were carefully designed to deliver targeted recommendations based on user-reported symptoms via an interactive questioning system that identifies and categorizes users' worries. The website provides continuing support for users' mental health journeys by delivering individualized daily instructions and coping tactics tailored to their specific situations. Furthermore, the seamless integration of professional aid, which connects users with skilled psychiatrists and motivational speakers, guarantees that crucial guidance is available when it is needed.

Our user-friendly platform enables people to take control of their mental health by providing accessible, individualized support for addressing and overcoming phobias, thereby improving their overall well-being. While the method is an effective self-care tool, it is vital to note that professional intervention is still required for a thorough treatment plan.  
  
This project has helped to improve mental health care by boosting its accessibility and efficacy. Moving forward, continuous research and collaboration with mental health specialists will be critical to fine-tuning and increasing the system's capabilities, ensuring that it continues to have a significant impact on mental health outcomes for individuals worldwide.

We believe integrating modern technology, targeted data-driven insights, and mental health expertise can improve the way mental health treatment is delivered, providing individuals with the resources and support they need to live better and more resilient lives. This initiative is a significant step toward attaining that vision, with the potential for broader applications in mental health support.

# 12.FUTURE SCOPE

While this project represents a significant milestone in providing mental health support, especially for those dealing with depression and panic attacks, there are several promising directions for future enhancement:

* **Expansion of Mental Health Conditions**: Consider expanding the platform to address a broader spectrum of mental health disorders beyond phobias, such as generalized anxiety disorder, obsessive-compulsive disorder, and post-traumatic stress disorder. This could improve the platform’s versatility and help a wider range of users.
* **Real-time Emotional Monitoring**: Explore integrating wearable technology or smartphone sensors to continuously monitor emotional states in real time. This could provide early detection of anxiety or panic attacks and offer immediate interventions to manage these episodes more effectively.
* **Teletherapy and Remote Counseling**: In alignment with the growing demand for teletherapy, integrating the application with telehealth platforms could enable users to easily access mental health professionals for live counseling sessions, fostering a more holistic approach to mental health care.
* **Cultural and Linguistic Adaptation**: Extend the application’s reach globally by localizing the content and interface for various languages and cultures. This would make the platform more accessible and culturally relevant, ensuring its effectiveness across different regions and populations.
* **Enhanced Personalization with AI**: Invest in advanced AI techniques to enhance the level of personalization. By continuously learning from users' interactions, the platform could evolve to provide even more tailored advice, coping strategies, and recommendations, based on individual progress and changing emotional states.
* **Collaboration with Mental Health Experts**: To ensure credibility, collaborate more extensively with psychiatrists, psychologists, and mental health experts to refine the questioning algorithms and ensure that the advice and strategies align with best clinical practices.
* **Machine Learning Interpretability**: Further research can be dedicated to making machine learning models more interpretable. Providing transparent explanations of how the platform diagnoses mental health concerns and offers advice would build greater trust with both users and mental health professionals.
* **Community and Peer Support Networks**: Develop a feature for users to engage in peer support communities. Safe, moderated spaces for individuals with similar mental health challenges could foster mutual encouragement and shared learning, which can be instrumental in mental health recovery.

By advancing in these areas, this project has the potential to become an even more comprehensive tool for mental health management, offering deeper support and accessibility to individuals around the world.

# 13.BIBLIOGRAPHY

1. A. K. Jaiswal, P. Tiwari, S. Kumar, D. Gupta, A. Khanna, and J. J. P. C. Rodrigues, “Identifying pneumonia in chest X-rays: a deep learning approach,” Measurement, vol. 145, pp. 511–518, 2019.
2. A. Bhandary, G. A. Prabhu, V. Rajinikanth et al., “Deeplearning framework to detect lung abnormality—a study with chest X-ray and lung CT scan images,” Pattern Recognition Letters, vol. 129, pp. 271–278, 2020.
3. G. Liang and L. Zheng, “A transfer learning method with deep residual network for pediatric pneumonia diagnosis,” Computer Methods and Programs in Biomedicine, vol.187, Article ID 104964, 2019.
4. H. Behzadi-khormouji, H. Rostami, S. Salehi et al., “Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images,” Computer Methods and Programs in Biomedicine, vol. 185, Article ID 105162, 2020.
5. I. Sirazitdinov, M. Kholiavchenko, T. Mustafaev, Y. Yixuan, R. Kuleev, and B. Ibragimov, “Deep neural network ensemble for pneumonia localization from a large- scale chest X-ray database,” Computers & Electrical Engineering, vol. 78, pp. 388–399, 2019.
6. Jaiswal, A.K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., Rodrigues, J.J. (2019). Identifying pneumonia in chest x-rays: a deep learning approach. Measurement, 145, 511–518.
7. Johnson, L. (2018). Deep learning applications in medical image analysis (Doctoral dissertation). Stanford University.
8. M. D. Alshehri and F. K. Hussain, “A fuzzy security protocol for trust management in the internet of things (fuzzy-IoT),” Computing, vol. 101, no. 7, pp. 791–818, 2019.
9. M. D. Alshehri, F. Hussain, M. Elkhodr, and B. S. Alsinglawi, “A distributed trust management model for the internet of things (DTM-IoT),” Recent Trends and Advances in Wireless and IoT-Enabled Networks, Springer, Berlin, Germany, 2019.
10. M. To˘gaçar, B. Ergen, Z. C¨omert, and F. Ozyurt, “A deep feature learning model for pneumonia detection applying a combination of MRMR feature selection and machine learning models,” IRBM, vol. 41, 2019.
11. Orbann, C., Sattenspiel, L., Miller, E., Dimka, J. (2017). Defining epidemics in computer simulation models: how do definitions influence conclusions? \*Epidemics, 19\*, 24–32.
12. TensorFlow (2021). TensorFlow: An open-source machine learning framework (Version 2.7.0) [Software]. https://[www.tensorflow.org/](http://www.tensorflow.org/)
13. UNICEF. (2019). Child Health - Pneumonia. https://data.unicef.org/topic/child- health/pneumonia/ accessed on 15 July 2019.
14. World Health Organization. Pneumonia. https:/[/www.who.int/news](http://www.who.int/news-room/fact-)-[room/fact-](http://www.who.int/news-room/fact-) sheets/detail/pneumonia.
15. Y. Ge, Q. Wang, L. Wang et al., “Predicting post-stroke pneumonia using deep neural network approaches,” International Journal of Medical Informatics, vol. 132, Article ID 103986, 2019.