# Chapter 1: Introduction:

# **Background and Motivation:**

The prevalence and impact of mental health disorders are increasingly recognized as critical global health issues. Despite this growing awareness, significant barriers to accessing mental health care persist, including societal stigma, a lack of available resources, and the prohibitive cost of traditional therapy. The digital transformation offers a promising avenue to overcome these barriers, providing accessible, private, and cost-effective mental health support. The motivation behind this project is to harness the potential of digital solutions to fill the gap in mental health care. By integrating features such as personalized care plans, mood tracking, and cognitive-behavioral therapy (CBT) modules, the project aims to offer a comprehensive mental health care application that is both accessible and engaging, addressing the nuanced needs of individuals seeking support.

# **Problem Statement:**

Mental health care access remains limited for a significant portion of the global population due to various factors, including cost, availability, and societal stigma. Traditional mental health support systems often fail to meet the diverse and evolving needs of individuals, leaving many without the necessary tools to manage their mental health effectively. The lack of personalized, accessible, and cost-effective mental health resources presents a critical barrier to improving the overall well-being and quality of life for those affected by mental health issues. This project seeks to address the urgent need for an innovative solution that bridges the gap in mental health care, providing users with a holistic and integrated approach to managing their mental health.

# **Objectives:**

The primary objective of this project is to develop a comprehensive mental health application that:

1. Provides a Secure and Accessible Authentication System: Ensuring users can easily access their accounts while maintaining the highest levels of data privacy and security.

2. Enables Personalized Mental Health Care Plans: Based on users' preferences, interests, and ongoing mental health assessments to offer tailored support.

3. Incorporates Advanced Features for Mental Health Management: Including mood tracking, CBT modules, stress and depression detection through machine learning, and meditation exercises, aiming to cater to a wide range of mental health needs.

4. Offers Educational Resources on Mental Health: To increase awareness, understanding, and self-management skills concerning mental health issues.

5. Achieves a User-Friendly Interface Design: To ensure ease of use for a diverse user base, enhancing user engagement and satisfaction.

An example objective for this project could be: "To integrate at least five personalized features, such as mood tracking and meditation exercises, within the next six months, thereby enhancing user engagement and providing measurable improvements in users' mental health."

# **Project Scope and Limitations:**

## Scope:

The project will develop a mental health application designed to provide users with personalized mental health support through a variety of features, including assessment tools, mood tracking, educational content, and interactive CBT modules. The application aims to offer a holistic approach to mental health care, accessible via mobile devices, ensuring privacy and ease of use. The development will focus on integrating evidence-based practices and leveraging machine learning technologies for enhanced personalization and effectiveness.

## Limitations:

1. Technological Constraints: The application's effectiveness is contingent upon the integration of advanced machine learning algorithms, which may require substantial computational resources and expertise.

2. User Engagement: The success of mental health interventions is often dependent on consistent user engagement, which can be challenging to maintain in digital platforms.

3. Data Privacy: Ensuring the privacy and security of sensitive user data is paramount, necessitating rigorous data protection measures that comply with global standards.

4. Clinical Validation: While the application aims to support mental health, it is not a substitute for professional medical advice or treatment. Its impact must be evaluated through clinical research and feedback from mental health professionals.

5. Accessibility and Inclusivity: Ensuring the application is accessible and inclusive to a diverse user base, including those with disabilities, presents both a challenge and a priority for development.

By addressing these objectives within the defined scope and acknowledging its limitations, the project aspires to make a significant contribution to improving mental health care access and support through digital innovation.

The Agile Scrum methodology was chosen for the development of the mental health application due to its specific benefits that align with the project's needs:

1. Adaptability: Agile Scrum's iterative nature allows for rapid adjustments to changing requirements or user feedback, essential for a project targeting evolving user needs in mental health care.

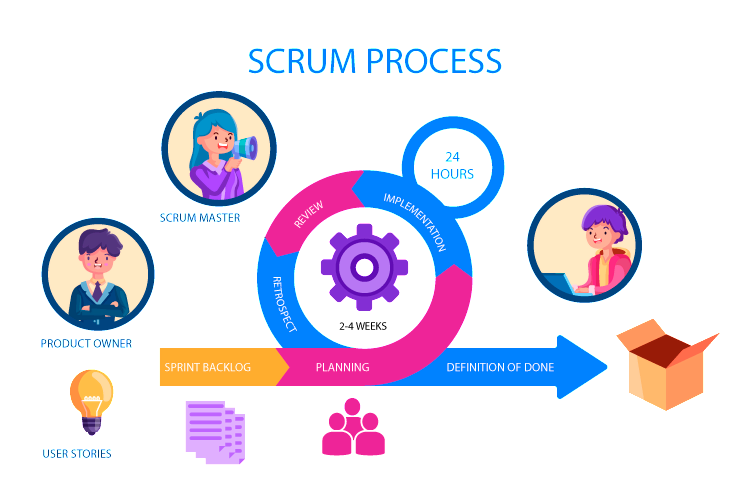
2. User Engagement: Through regular feedback cycles, the methodology ensures that the application evolves in direct response to user input, increasing satisfaction and effectiveness.

3. Risk Management: Early and continuous delivery inherent in Agile Scrum helps identify and mitigate risks promptly, ensuring the project remains on track and adaptable to unforeseen challenges.

4. Collaboration: Agile Scrum fosters a collaborative environment, integrating diverse stakeholder insights into the development process, which is crucial for a multidimensional project like a mental health application.

5. Quality and Continuous Improvement: The framework emphasizes ongoing assessment and refinement, ensuring high quality and relevance of the application through constant iteration and user feedback.

6. Resource Efficiency: Prioritizing work in sprints ensures that resources are focused on delivering the most value to users efficiently, a critical consideration for time-sensitive and budget-conscious projects.



# Chapter 2: Market and Literature Survey

# **Related works:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **App Name** |  | **Positives** | **Negatives** | | --- | --- | --- | --- | | Fabulous |  | Offers personalized coaching, programs, challenges, community support, and integration with other apps and devices. | Onboarding and some other tips are mandatory to watch and take a lot of time even for returning users. | | Habitica | Habitica Icon - UpLabs | Includes gamification and social features to enhance engagement and motivation, offering a wide range of habits and tasks. | May not suit everyone due to the gaming aspect and complex interface, requiring consistent use for gamification to be effective. | | Onrise | Onrise: Habit Tracker & Focus | App Price Intelligence by Qonversion | Incorporates gamification, social features, and a variety of tasks to make habit-building engaging and adaptable. | Interface may be overwhelming for new users, not ideal for those who prefer a straightforward approach, requiring consistent use. | | Not Boring (Habits) |  | Engaging graphics and animations, personalized habit tracking, and comprehensive progress overview. | Lacks advanced features like reminders and motivational messages, may not appeal to those seeking a more serious habit tracking. | | Productive |  | User-friendly interface with a variety of habit tracking options and detailed statistics for insights. | Limited customization options and some features are only available in the paid version, with reported technical issues. | | Habit Garden |  | Offers engaging gamification features, daily quotes, an easy-to-use interface for habit tracking, and progress visualization. | Limited customization options, no social features, only available on iOS, stability issues, and requires in-app purchases for all features. | |

# **System features:**

| **Feature / App** | **Fabulous** | **Habitica** | **Onrise** | **Not Boring (Habits)** | **Productive** | **Habit Garden** |
| --- | --- | --- | --- | --- | --- | --- |
| Personalized Coaching | Yes | No | No | No | No | No |
| Programs & Challenges | Yes | Yes | No | No | No | No |
| Community Support | Yes | Yes | No | No | No | No |
| App & Device Integration | Yes | No | No | No | No | No |
| Gamification | No | Yes | Yes | No | No | Yes |
| Social Features | No | Yes | Yes | No | No | No |
| Habit and Task Variety | No | Yes | Yes | No | Yes | Yes |
| Mood Tracking | No | No | No | No | No | No |
| CBT Modules | No | No | No | No | No | No |
| Stress & Depression Detection | No | No | No | No | No | No |
| Meditation Exercises | No | No | No | No | No | No |
| Educational Content on Mental Health | No | No | No | No | No | No |
| Progress Visualization | No | No | No | Yes | Yes | Yes |
| User-friendly Interface | No | No | No | Yes | Yes | Yes |
| Advanced Features (e.g., reminders) | No | No | No | No | No | No |

# User Personas

# Survey

# GANT CHART

# Chapter 3: Analysis

# Functional requirements

1. **Register**

Allow users to register by creating new account or through their Google account.

# Login

Allow users to login using their email and password or through their Google account.

# Reset Password

Allow users to initiate a password reset through email verification.

# Edit Password

Allow users to change their passwords.

# Edit profile

Allow users to edit their profile information (image, Date of birth, Gender, first name, last name).

# Turning on/off notifications

Allow users to toggle notifications on or off.

# Select starting day time

Allow users set their preferred start time for daily activities.

# Preferences Questions

The user answers these questions (true/false) only once when using the program for the first time.

# Depression test

The user answers these questions to find out whether he has depression or not.

# Select mood

Allow users to input their daily mood and each mood has some tips to deal with it.

# Select activities and reasons:

Allow users to input their daily activities and reasons and they can add more activities or reasons.

# Journal Entries

User can write how the day was.

# Provide customize plan based on preferences

The system creates a personalized plan for the users to help them improve

# User make his plan

Allow users to select topics they are interested in to make plan

# Pick the time for meditation

Allow users to set their preferred meditation time

# Play music audios for meditation

Provide a library of meditation audio for users to play during meditation.

# Challenge thoughts

Provide types of the thought and each type has some explanation for it, Allow users to identify their thought patterns.

# View Insights

Display insights on depression, stress, activities, and weekly checking.

# Weekly Checking

Send true/false questions for weekly checking and user answer them.

# Stress detection

Identify if the user has stress or not based on journaling.

# Depression detection

Identify if the user has depression or not based on journaling.

# Chat bot

Chat bot for users to interact with.

# Non-functional requirements

1. **Usability**
   * The application has an easy-to-use interface that has everything a user needs in an easy way to understand.
   * The colors used in the application are consistent and comfortable for the eye.
2. **Robustness**
   * The system can respond and handle invalid inputs with error messages and exception handling.
3. **Security**
   * The system will show only the data related to each user.
   * No other users can access another user's data.
4. **Reliability**
   * The application shows the user a clear error message when the user fails to enter a valid command
5. **Availability**
   * The system will be available 24/7 and the user can use it at any time.
6. **Response Time**
   * The system will respond to user requests in a minimum amount of time.
7. **Memorability**
   * Colors, themes and experience in our app will be unforgettable, so users can return to the interface after some time and start efficiently working with it right away.

# 3.1 Audience:

Given the context of this being a personal project with a more focused scope, the stakeholders and audience can be streamlined to reflect the scale and objectives more accurately.

## Stakeholders:

In the context of a personal project aimed at developing a mental health application, the stakeholders primarily include:

1. Project Owner/Developer: As the individual initiating and developing the application, the project owner is responsible for the vision, development, and deployment of the app. This role encompasses project management, development activities, and decision-making regarding the app's features and content.

2. End Users: Individuals interested in improving their mental health, managing emotions, and enhancing their mood. They are the direct beneficiaries of the app, engaging with its content and functionalities for personal growth and emotional well-being.

3. Content Contributors: This may include the project owner if they are also creating content, or any mental health professionals consulted to ensure the app's advice and resources are reliable and effective. They are responsible for providing accurate, helpful, and ethical content.

4. Technical Advisors: If the project owner seeks external advice, technical advisors (such as more experienced software developers or designers) could provide insights on the app’s design, functionality, and technical improvements.

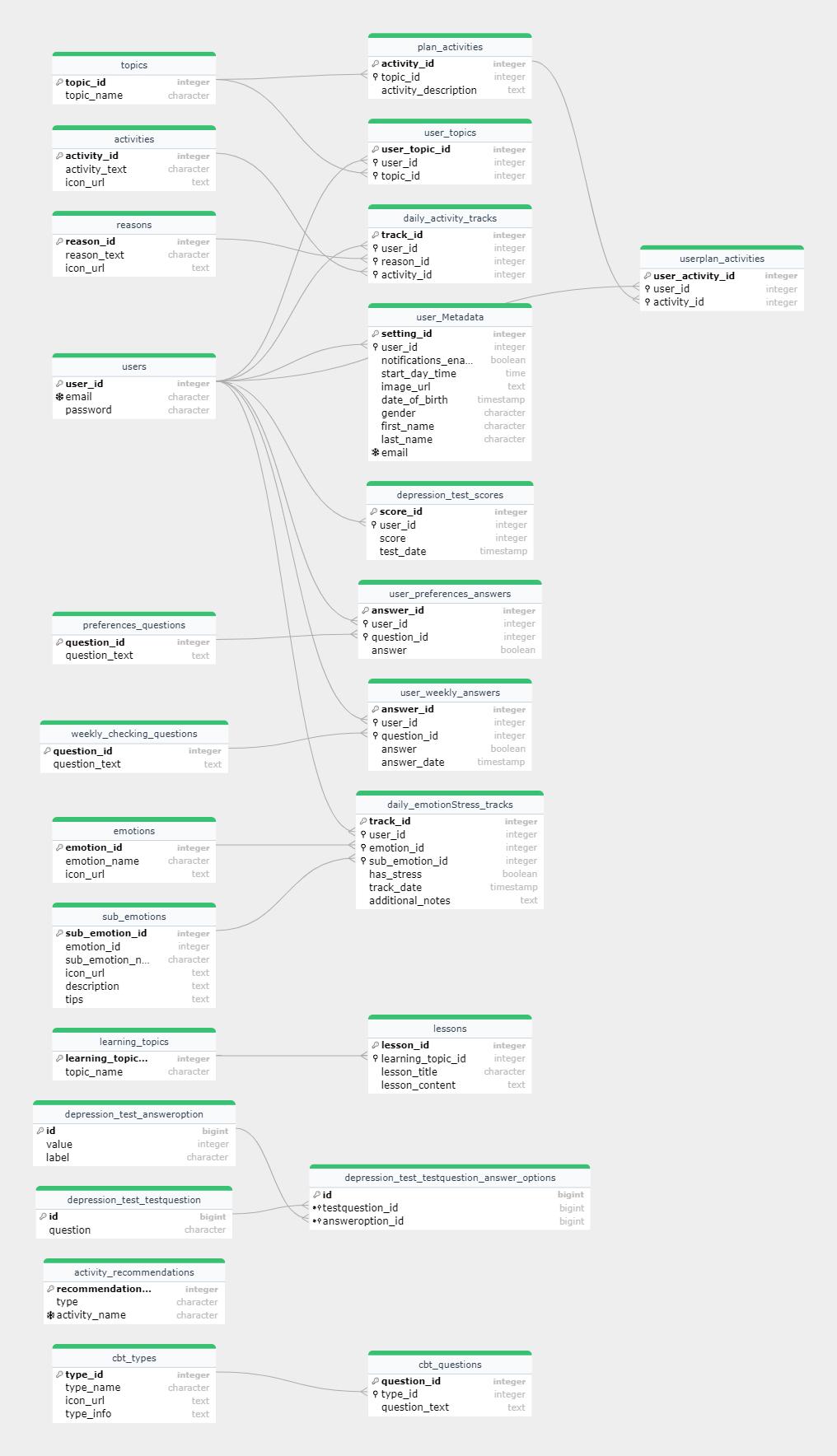
## Audience:

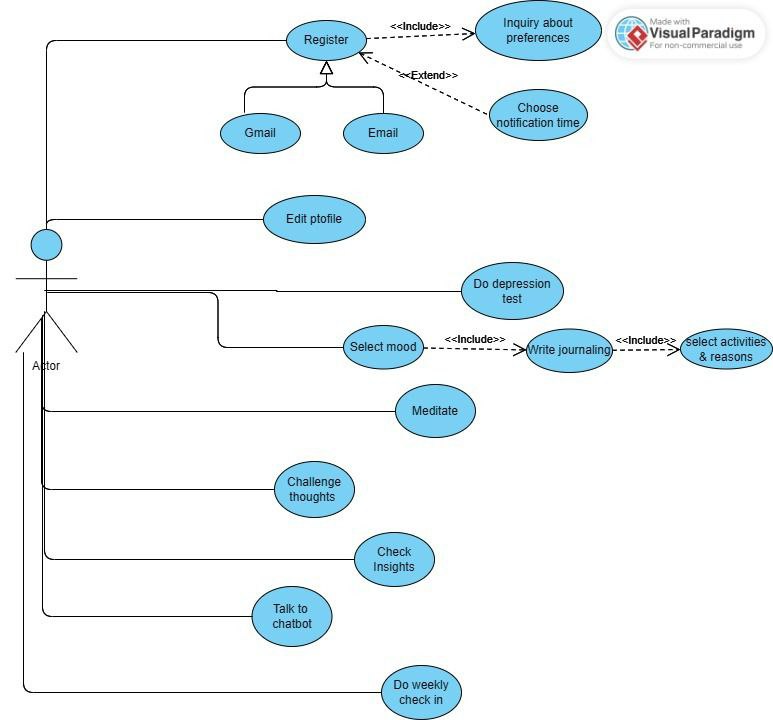
The audience for this personal project primarily consists of:

1. Individuals Seeking Mental Health Tools: People looking for accessible, user-friendly tools to help them understand their emotions, improve their mood, and learn more about mental health. This includes a wide demographic potentially ranging from young adults to older individuals, all seeking self-improvement through digital means.

2. Mental Health Enthusiasts: Those with a general interest in mental health and wellness who are always on the lookout for new tools, insights, and methods to enhance their well-being or the well-being of others around them.

3. Caregivers and Family Members: Those caring for individuals with mental health challenges may use the app to gain insights and strategies to support their loved ones effectively.

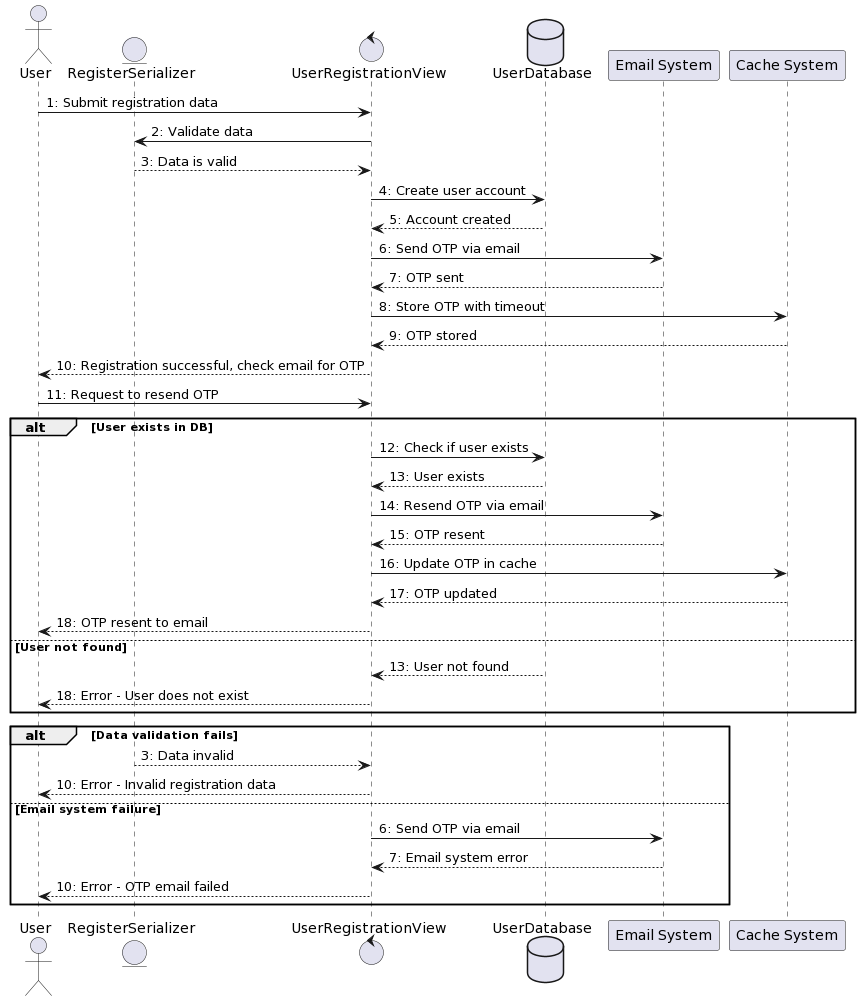
* ERD
* 
* Use case diagram



* Use case description & Sequence flows

**User Registration with Email Verification and OTP Resend Functionality**

|  |  |  |
| --- | --- | --- |
| **Use case name** | User Registration with Email Verification | |
| **Actors** | New User | |
| **Description** | This use case describes the process by which a new user registers for an account, receives an OTP for email verification, and has the option to resend the OTP if necessary. | |
| **Preconditions** | - The user has access to the registration interface.  - The user possesses a valid email address that is not already associated with an existing account. | |
| **Post conditions** | - The user is registered with a pending verification status.  - An OTP is sent to the user's email.  - The system is ready to verify the user's email upon submission of the OTP.  - The user has the ability to request a resend of the OTP if needed. | |
| **Flow of activities** | Actor | System |
| 1. Provide registration details.   5. Receive OTP via email.  6. (Optional) Request OTP resend | 2. Validate registration details.  3. Create user account with pending verification status.  4. Send OTP to the user's email.  7. Validate resend request.  8. Resend OTP to user's email. |
| **Exception conditions** | - If the user provides an email that is already in use, the system should notify the user and halt the registration process.  - If the system fails to send the OTP (e.g., due to an email service outage), the user should be informed of the failure and prompted to try again.  - If the user's details do not meet the validation criteria, the user should be prompted to correct the information and resubmit the registration form. | |



**Google Login for Both Registration and Login**

|  |  |  |
| --- | --- | --- |
| **Use case name** | Google Authentication | |
| **Actors** | User, Google Account? | |
| **Description** | This use case details the process where a user can register or log in to the system using their Google account credentials. | |
| **Preconditions** | - The user has selected the option to register or log in using Google.  - The user is not currently authenticated in the system.  - The user has a valid, active Google account. | |
| **Post conditions** | - The user is logged into the system.  - A new account is created if one did not previously exist, linked to the user's Google account.  - The user's email is marked as verified in the system. | |
| **Flow of activities** | Actor | System |
| 1. Initiate Google login/register.   7. Receive JWT tokens and confirmation of successful login or account creation. | 2. Redirect to Google sign-in and request access token.  3. Receive Google access token and user data.  4. Check if a user account exists with the given Google email.  5. Create a new user account if none exists; otherwise, proceed to login.  6. Generate and return JWT tokens for user authentication. |
| **Exception conditions** | - If the Google access token is invalid, the system should notify the user of the error.  - If there is a failure in creating a new user account or logging in, the system should provide an error message.  - If the user's Google account does not provide the necessary permissions, the system should request the required permissions again. | |

A screenshot of a diagram

Description automatically generated

**User Login**

|  |  |  |
| --- | --- | --- |
| **Use case name** | User Login | |
| **Actors** | User | |
| **Description** | This use case outlines the process by which a returning user logs into the system using their email and password. | |
| **Preconditions** | - The user has an existing account with a verified email.  - The user is at the login page.  - The user knows their email and password. | |
| **Post conditions** | - The user is authenticated and logged into the system.  - JWT tokens are generated and provided to the user for subsequent authenticated sessions. | |
| **Flow of activities** | Actor | System |
| 1. Submit email and password.   4. Receive JWT tokens and access to the system as an authenticated user. | 2. Validate login credentials against the database.  3. Generate JWT tokens for authentication. |
| **Exception conditions** | - If the user enters invalid login credentials, the system should notify the user of the invalid attempt.  - If the user's account is locked or disabled, the system should inform the user and deny access.  - If there is a system error preventing login, the user should be informed of the error and may be prompted to try again later. | |

A diagram of a software

Description automatically generated

**User Profile Retrieve and Update**

|  |  |  |
| --- | --- | --- |
| **Use case name** | User Profile Retrieve and Update | |
| **Actors** | User | |
| **Description** | This use case enables an authenticated user to view their current profile details and update them as needed. | |
| **Preconditions** | - The user must be logged in and authenticated.  - The user's profile already exists in the system. | |
| **Post conditions** | - The user's profile information is updated in the database.  - The user receives confirmation of the updates. | |
| **Flow of activities** | Actor | System |
| 1. Request to view or update profile. 2. Submit updated profile information.   6- View current profile information. | 2. Retrieve user's profile from the database  4. Validate and save updates to the database.  5. Confirm profile update to the user. |
| **Exception conditions** | - If the user is not authenticated, access to view or update the profile is denied.  - If the updated information fails validation, the user is prompted to correct it.  - If there is a database error while updating, the user is notified of the error. | |

A screenshot of a diagram

Description automatically generated

**Password Change for Authenticated Users**

|  |  |  |
| --- | --- | --- |
| **Use case name** | Password Change | |
| **Actors** | User | |
| **Description** | This use case describes the process by which an authenticated user can change their password. | |
| **Preconditions** | - The user is logged in and has access to the password change option.  - The user knows their current password. | |
| **Post conditions** | - The user's password is updated in the database.  - The user is notified that their password has been changed successfully. | |
| **Flow of activities** | Actor | System |
| 1. Submit current and new password. | 2. Verify current password and validate the new password.  3. Update the password in the database.  4. Notify the user of successful password change. |
| **Exception conditions** | - If the current password is incorrect, the system denies the password change request.  - If the new password does not meet the system's security requirements, the user is prompted to try again.  - If there is a system error during the update, the user is informed of the failure. | |

A diagram of a username

Description automatically generated

**Password Reset Process (Including OTP Verification)**

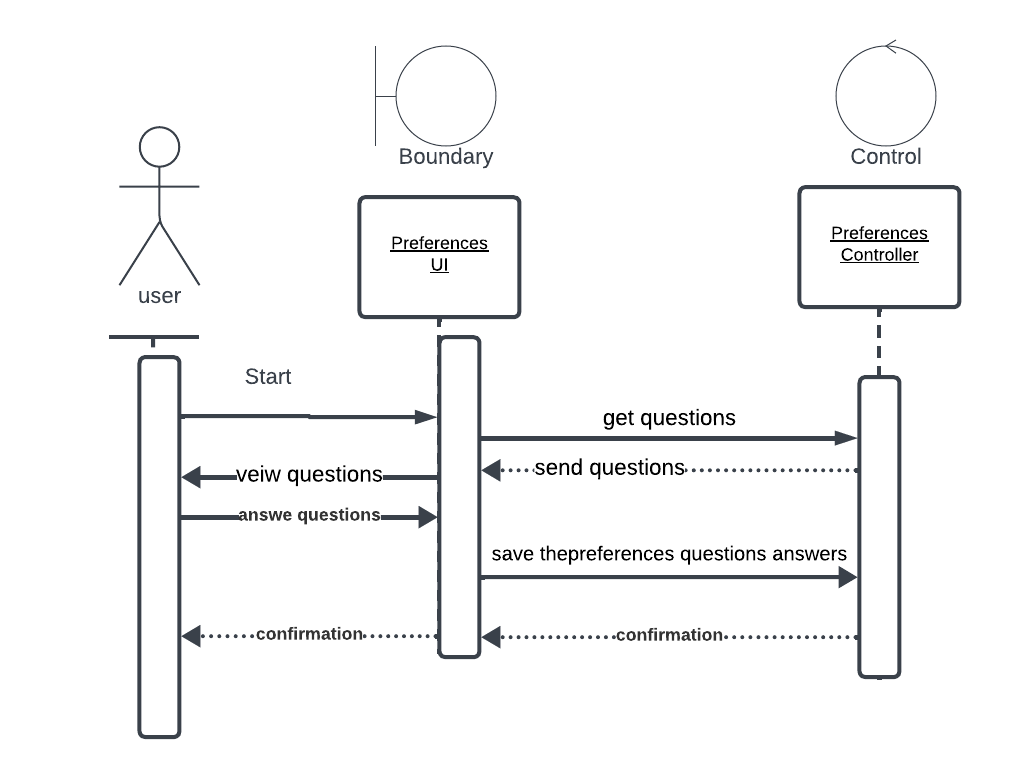
|  |  |  |
| --- | --- | --- |
| **Use case name** | Password Reset with OTP Verification | |
| **Actors** | User | |
| **Description** | This use case details the process for a user to reset their password, which includes receiving an OTP via email, verifying the OTP, and then entering a new password. | |
| **Preconditions** | - The user must have a previously registered account with a verified email address.  - The user must have access to the email associated with their account. | |
| **Post conditions** | - The user's password is successfully reset.  - The user can log in with the new password. | |
| **Flow of activities** | Actor | System |
| 1. Initiate password reset request.  3. Retrieve OTP from email.  4. Submit OTP for verification.  7. Enter and submit new password. | 2. Send OTP to user's email.  5. Validate OTP.  6. Prompt user to enter a new password.  8. Update user's password in the system.  9. Confirm successful password reset to the |
| **Exception conditions** | - If the OTP is incorrect or expired, the user is prompted to request a new OTP.  - If the new password does not meet the security requirements, the user is asked to provide a different password. | |

A screenshot of a computer screen

Description automatically generated

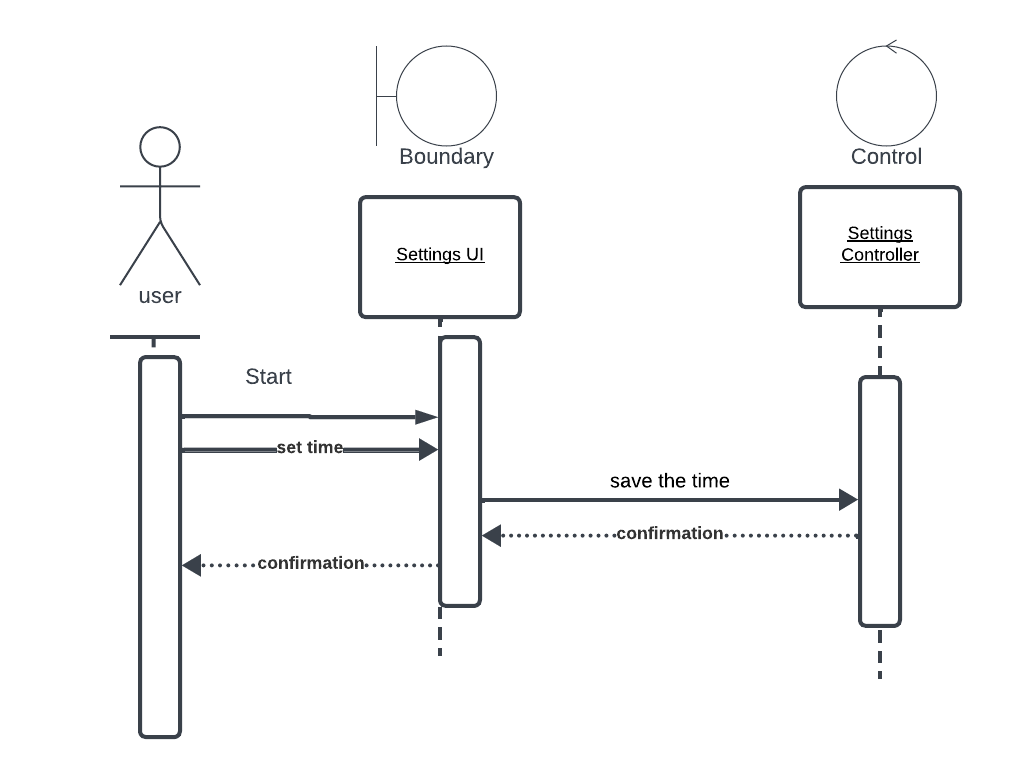
**-Inquiry about preferences:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Inquiry about preferences** | |
| **Actors** | User | |
| **Description** | True/false questions appear only once after user register to the application to get his preferences. | |
| **Preconditions** | The user must be registered. | |
| **Post conditions** | Store answers and then use it to help user. | |
| **Flow of activities** | Actor | System |
| Start  Answer questions  View confirmation | Send the questions  Save the answers  Send confirmation |
| **Exception conditions** | If the user didn’t fill all the question his/her data will not be saved | |



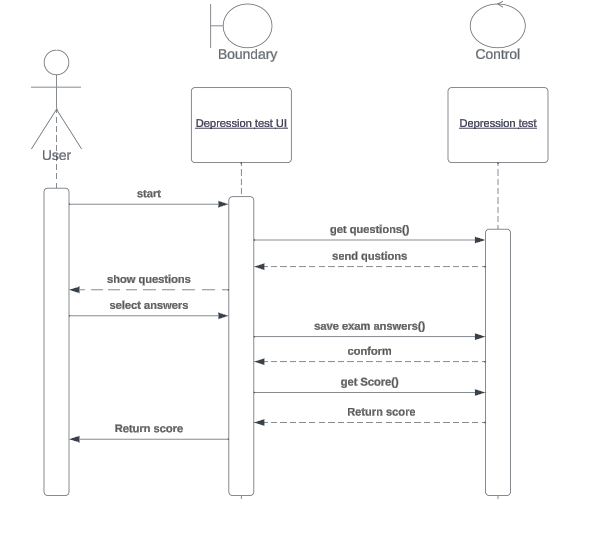
**-Choose notification time:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Choose notification time** | |
| **Actors** | User | |
| **Description** | The user chooses the appropriate time for the notification. | |
| **Preconditions** | The user must be registered. | |
| **Post conditions** | Sending the notification to the user at this time. | |
| **Flow of activities** | Actor | System |
| User go to the setting  User choose the time | Store the time  Send conformation |
| **Exception conditions** | No Exceptions | |



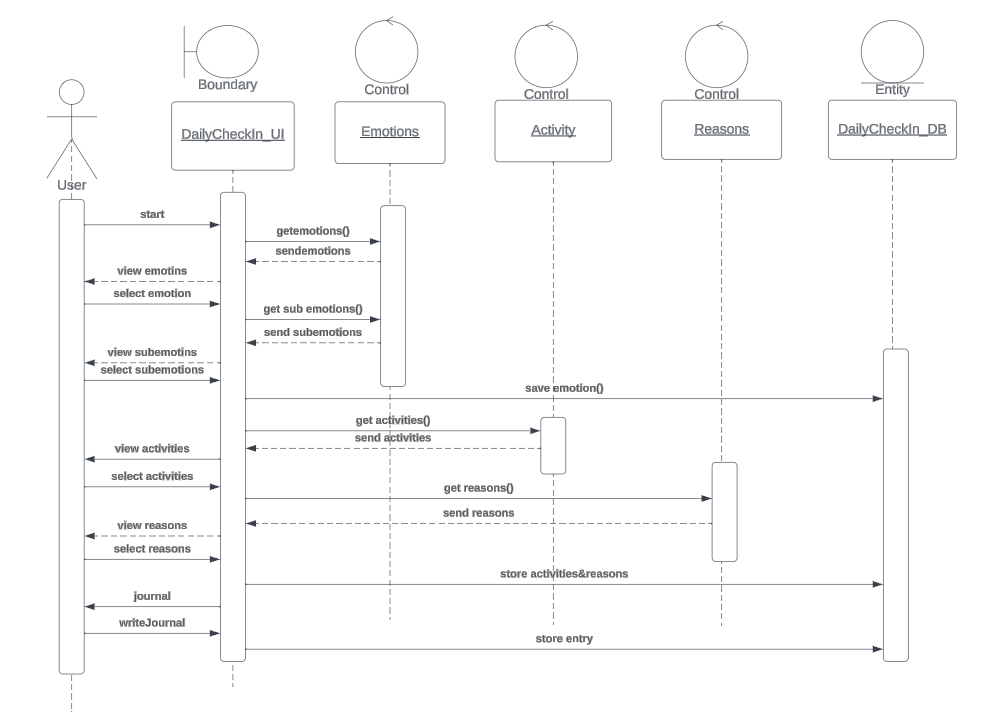
**-Do depression test:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Do depression test** | |
| **Actors** | User | |
| **Description** | User answers these test questions to find out whether he has depression or not. | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | Depression test result. | |
| **Flow of activities** | Actor | System |
| 1-User start Test  3-User selects answers to questions and submits.  5-User get score. | 2-Sent Questions and answers.  4-calculate the depression score.  Save answers.  Return scores. |
| **Exception conditions** | If user answers not submitted he tries get score before submitting answers. | |



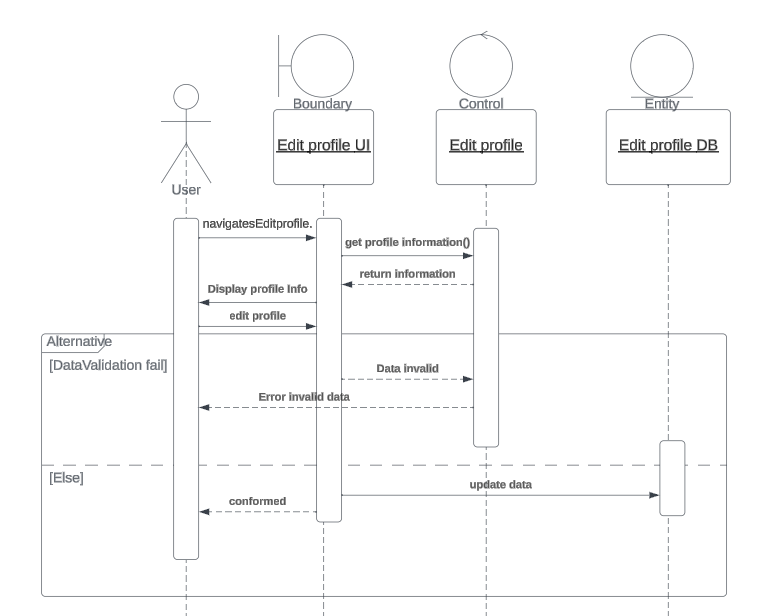
**-Daily check in:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Daily check in** | |
| **Actors** | User | |
| **Description** | Enabling users to express their emotional states easily by selecting emoji for their daily mood. Then users can enter their daily activities and reasons and they can add more activities or reasons. After that User can write how was his day. | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | Store mood and activities and reasons in app's database. Analyze sentiment in this journaling to discover if there is stress or depression. | |
| **Flow of activities** | Actor | System |
| 1-Navigate to mood section  3-Select specific emotion.  5-Select one.  9-User selects their daily activities and reasons from list  12-User writes journal entry that describing their day | 2-Display list of emotions to user.  4-Display the sub-emotions and tips.  6-store emoji in database.  7-Retrieve list of activities.  8-Retrieve list of reasons.  10-system store selected activities and reasons.  11-Prompt to write journal.  12-save this entry. |
| **Exception conditions** | Data retrieval failure. | |



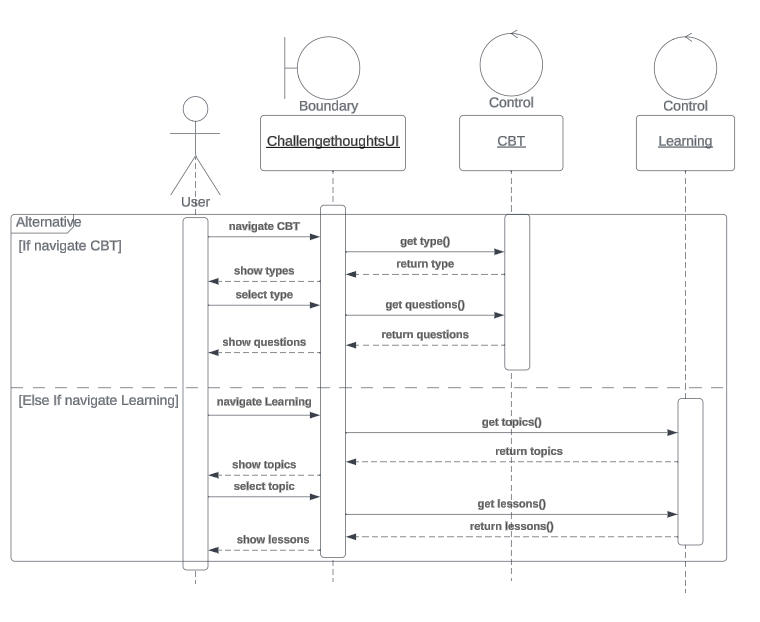
**-Edit profile:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Edit profile** | |
| **Actors** | User | |
| **Description** | Allow users to edit their profile information (image, Date of birth, Gender, first name, last name) | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | The user's information is updated in the database. | |
| **Flow of activities** | Actor | System |
| 1-User navigates to Edit profile.  3-make change to desired field.  5-conform. | 2-system retrieves and display the user current profile information.    4-check for valid input data.  -if invalid system notified user about invalid data.  -else update in DB. |
| **Exception conditions** | Invalid input | |



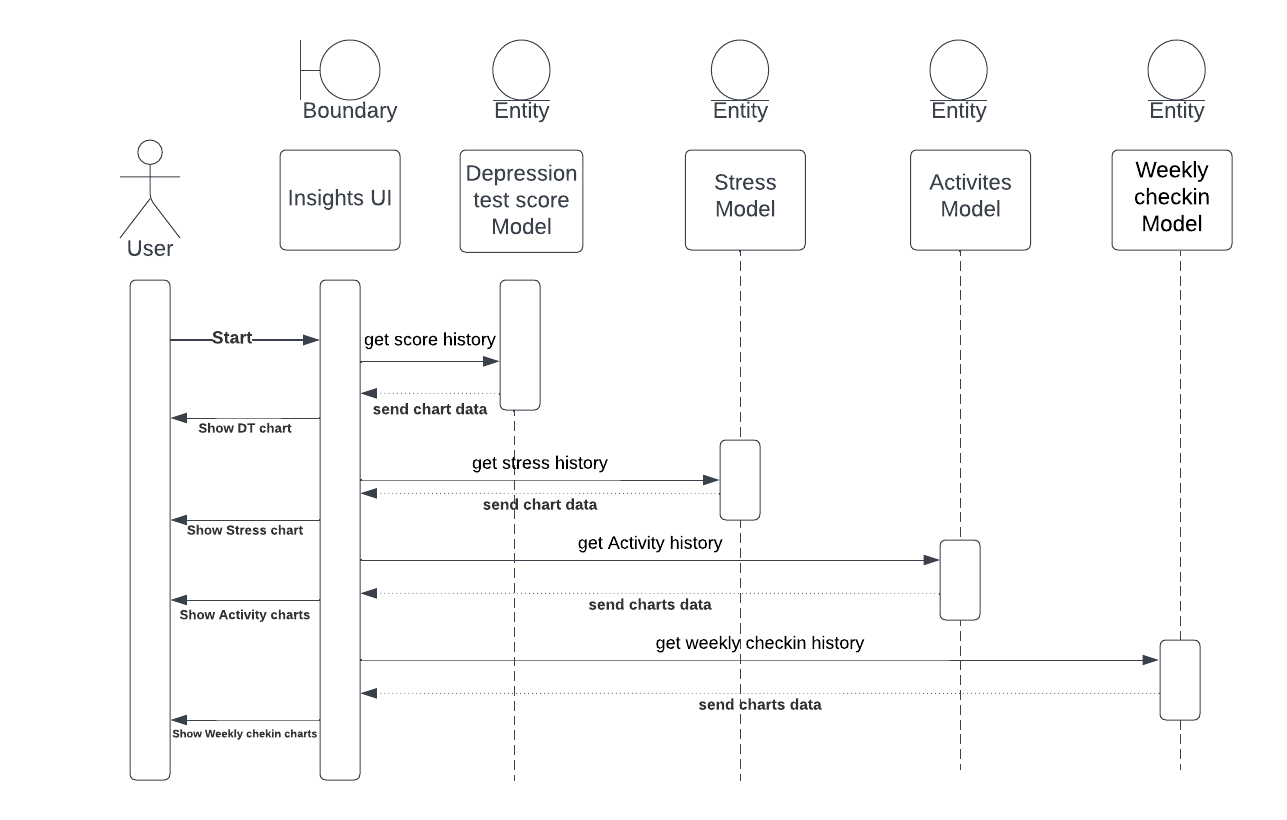
**-Challenge thoughts(CBT/learning):**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Challenge thoughts(CBT/learning)** | |
| **Actors** | user | |
| **Description** | Provide types of the thought and each type has some explanation for it. | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | users will identify their thought patterns.  ??? | |
| **Flow of activities** | Actor | System |
| 1-if user navigates CBT.  3-send thought type.  5-if User navigates Learning.  7-User Select specific topic. | 2-Send type of depressions.  4-return questions based on thought type.  6-Send Learning topics.  8-send list of lessons to this topic. |
| **Exception conditions** | No exceptions | |



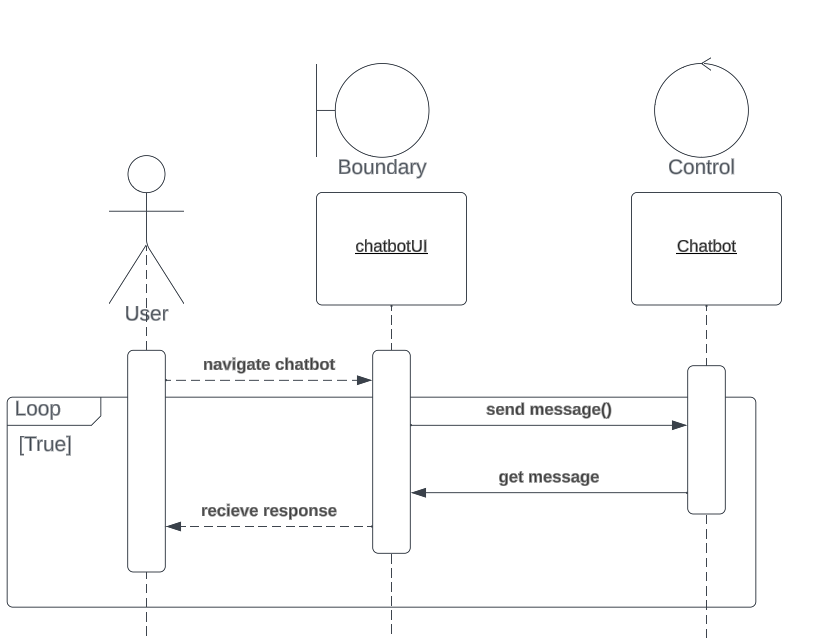
**-Check Insights:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Check Insights** | |
| **Actors** | User | |
| **Description** | Providing users with a comprehensive view of their mental health journey in application like activities | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | View charts | |
| **Flow of activities** | Actor | System |
| Start  View charts | Send activities charts  Send depression test score chart  Send weekly check-in chart  Send stress chart |
| **Exception conditions** | No exceptions | |



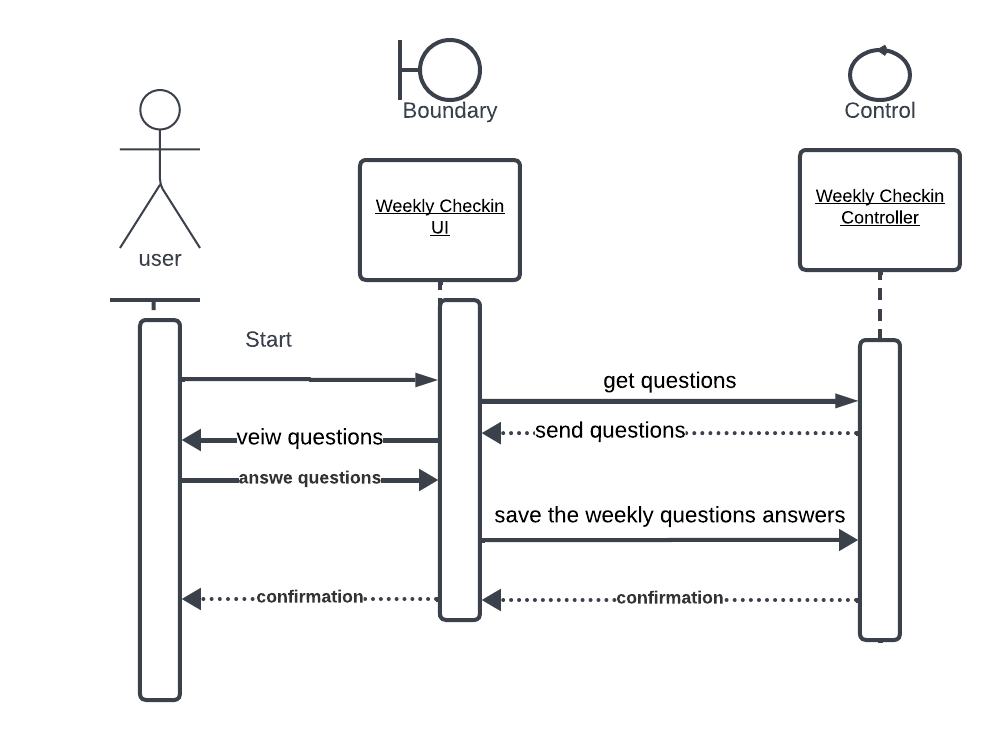
**-Talk to chatbot:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Talk to chatbot** | |
| **Actors** | User | |
| **Description** | Interactive and conversational engagement with the chatbot for mental health questions. | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | Find out the answers to his questions.  ??? | |
| **Flow of activities** | Actor | System |
| 1-User send message.  3-receive response. | 2-System get message and send response. |
| **Exception conditions** | If user provide an invalid or empty message. | |



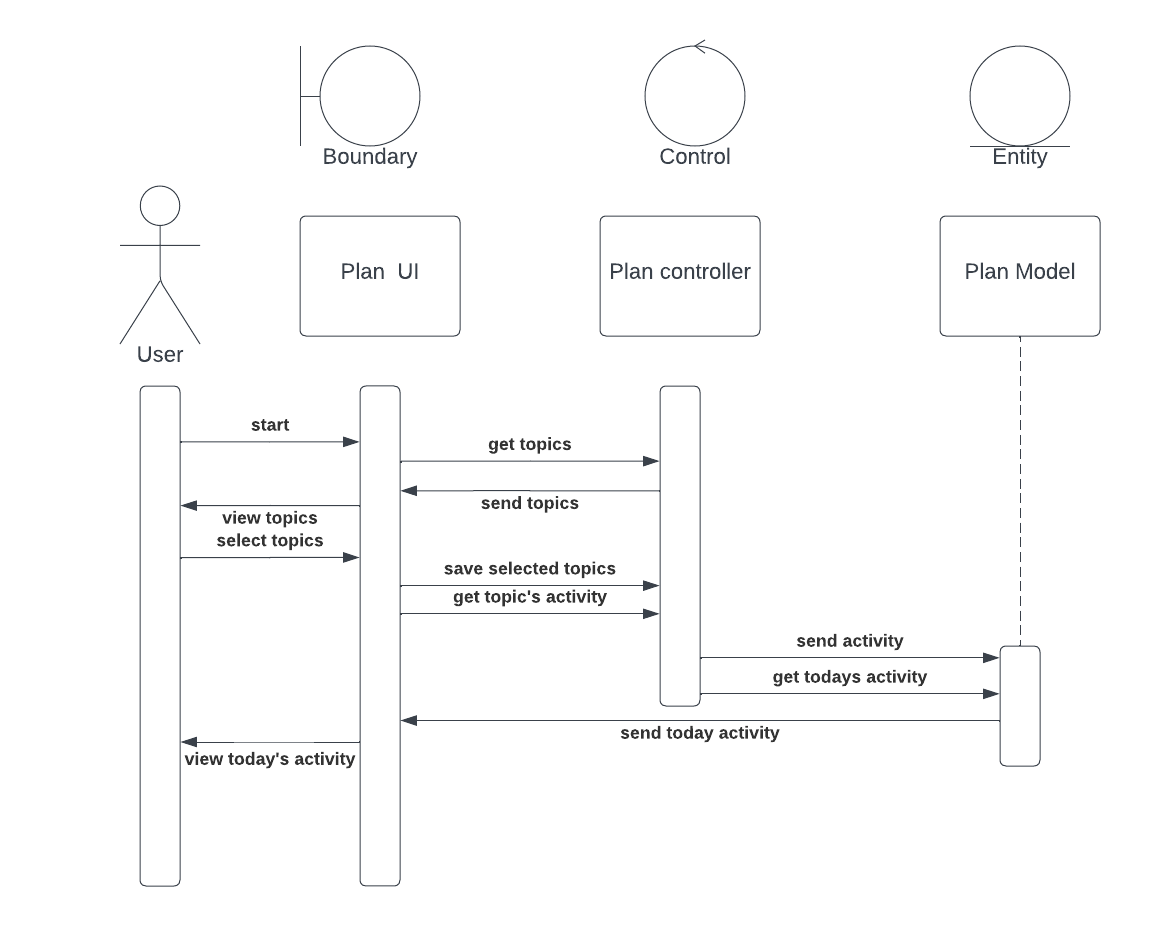
**-Do weekly check in:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Do weekly check in** | |
| **Actors** | User | |
| **Description** | true/false questions for weekly checking and user answer them. | |
| **Preconditions** | User must be logged in. | |
| **Post conditions** | Confirmation and Store it in app's database. | |
| **Flow of activities** | Actor | System |
| Start  Answer the Questions  View Confirmation | Send the questions  Save the answers  Send confirmation |
| **Exception conditions** | No Exception | |

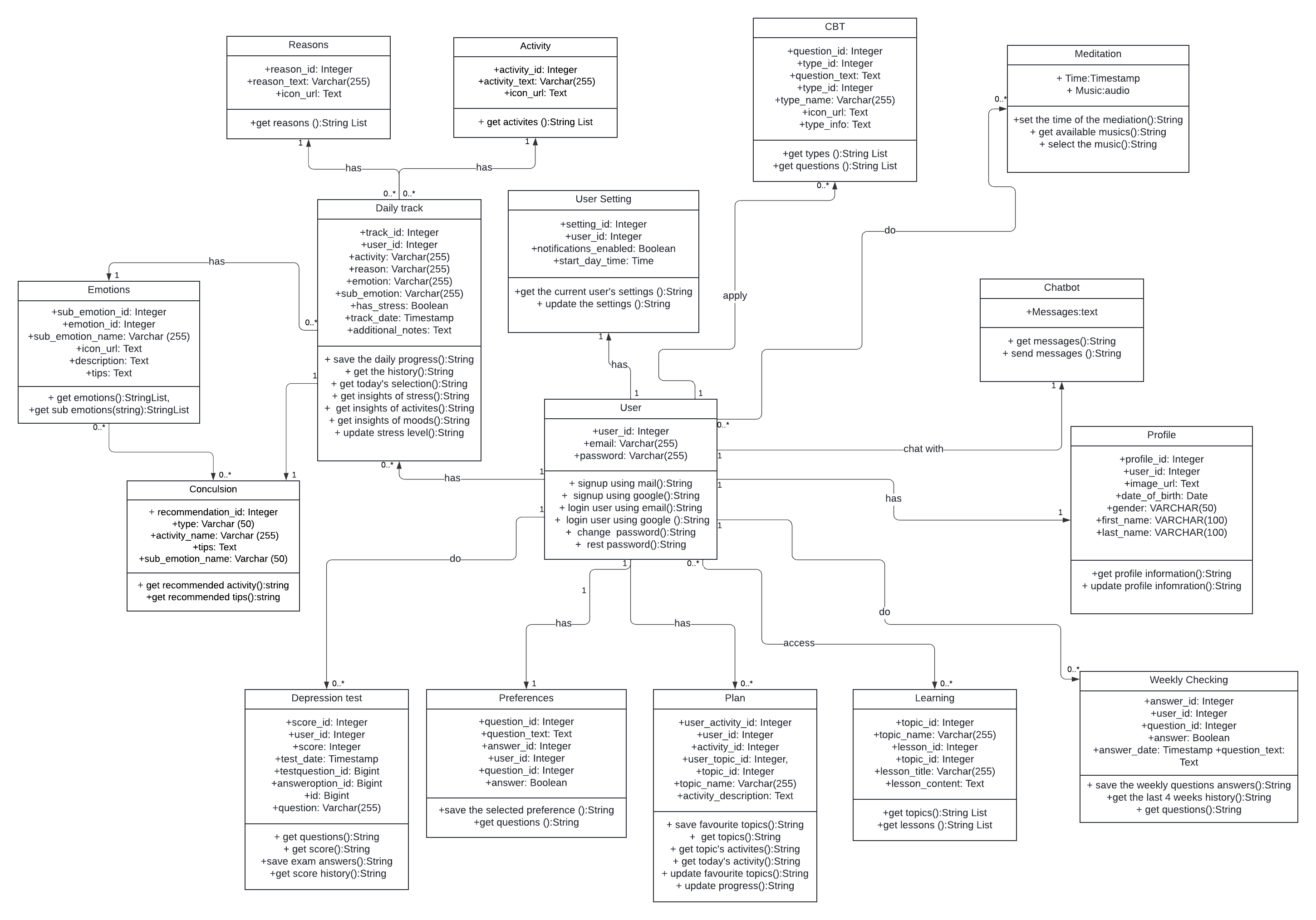


**-Customize plan:**

|  |  |  |
| --- | --- | --- |
| **Use case name** | **Customize plan** | |
| **Actors** | User | |
| **Description** | Customized plan for the user based on the faviourte topics. | |
| **Preconditions** | User must be logged in and did the preferences inquiry. | |
| **Post conditions** | Confirmation and customized plan. | |
| **Flow of activities** | Actor | System |
| Start  Select topics  View plan  View today task | Send the topics  Save the topics  Send activity based on the selected topics  Send daily activity task |
| **Exception conditions** | No Exception | |



* Class diagram



* System Architecture diagram

# Chapter 4: Design

* UI prototype (figma)
* Technologies and tools:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Technology** | **Reason for Using** | **Latest Version** | | --- | --- | --- | | scikit-learn | SVM for stress and depression detection | 1.2.2 | | nltk | Text cleaning and processing for data analysis | 3.8.1 | | spaCy | Advanced text cleaning and natural language processing | 3.7.2 | | Django | Rapid development and built-in admin panel for backend | 5.0.1 | | Flutter | Cross-platform development for Android and iOS | 3.7.0 | |