

# AI Powered Plant Disease Detection System

## Software Requirements Specification

Version 1.0

26.12.2023

Ahmet Melih Kostak

Ekin Alaydın

Efe Özdemir

Nazlı Hilal Özer

Instructor: Dr. Sevgi Koyuncu Tunç  
Spring 2023/2024

# Revision History

Date	Description	Author	Comments
29.11.2023	Version 1.0	Ahmet Melih Kostak	<First Revision>
27.05.2024	Version 1.1	Ekin Alaydın	<Second Revision>
28.05.2024	Version 1.2	Nazlı Hilal Özer	<Third Revision>
29.05.2024	Version 1.3	Nazlı Hilal Özer	<Fourth Revision>

## Contents

1. Introduction	1
1.1. Purpose	1
1.2. Scope	1
1.3. Definitions, Acronyms, and Abbreviations	2
1.4. References	2
1.5. Overview	3
2. General Description	4
2.1. Product Perspective	4
2.2. Product Functions	5
2.2.1. Disease Detection	5
2.2.2. Community Forum	6
2.2.3. Map - Disease Visualization	7
2.2.4. Sign Up	8
2.2.5. Login	9
2.2.6. Edit Profile	10
2.3. User Characteristics	11
2.4. General Constraints	13
2.5. Assumptions and Dependencies	14
3. Specific Requirements	16
3.1. External Interface Requirements	16
3.1.1. User Interfaces	16
3.1.1.1. Login Page UI Design	16
3.1.1.2. Home Page UI Design	17
3.1.1.3. Report Page UI Design	18
3.1.1.4. Community Page UI Design	19
3.1.1.5. Disease Distribution Page UI Design	20
3.1.1.6. Post Page UI Design	21
3.1.1.7. Sign Up Page UI Design	22
3.1.1.8. Edit Profile Page UI Design	23
3.1.1.9. Camera Page UI Design	24
3.1.2. Hardware Interfaces	25
3.1.3. Software Interfaces	26
3.1.4. Communication Interfaces	28

3.2.	Functional Requirements	30
3.2.1.	Check Disease & Treatment	33
3.2.2.	Add Post to the Community	37
3.2.3.	Sign Up	41
3.2.4.	Login	45
3.2.5.	Edit Profile	49
3.2.6.	Analyze the Map	53
3.3.	Non-Functional Requirements	56
3.3.1.	Performance	56
3.3.2.	Reliability	57
3.3.3.	Availability	58
3.3.4.	Security	59
3.3.5.	Maintainability	61
3.3.6.	Portability	63
3.3.7	Design Constraints	64

# 1.Introduction

## 1.1. Purpose

The purpose of this document is to define the software requirements for the Plant Disease Detection project. It provides a detailed overview of the project, its parameters, and its goals. This SRS document outlines the specific target audience, including professional farmers, general gardeners, and agricultural students, ensuring the system meets their distinct needs. Additionally, it describes the user interface, hardware, and software requirements. It also defines the roles of the system users: **clients**, who refer to general users interacting with the system for disease detection and analysis, and **admins**, who manage and oversee the system's operations. The document provides an overview of the functionalities expected by each user group, offering clarity on how the product will address their requirements and use cases.

## 1.2. Scope

The Plant Disease Detection project will be a mobile application that leverages image processing and deep learning algorithms to analyze plant images and detect diseases. The system aims to provide users with disease information at a validated accuracy rate of at least 90%, measured using the F1-score to ensure clarity and consistency. The response time for image analysis will target a maximum of 5 seconds under optimal hardware and stable network conditions, ensuring efficient performance for most users.

After detecting a disease, the application will offer users recommended treatments to help cure their plants. These treatment descriptions will be retrieved from the GEMINIAI API. Disease descriptions will be securely stored using Google Cloud. To enhance community engagement, the application will include a **community forum** where users can share knowledge, ask questions, and respond to other users' inquiries by creating posts.

Additionally, the application will visualize the distribution of diseases across the country on an interactive map. This map will display real-time or periodically updated data, informing users about the most prevalent diseases in specific regions.

The application will also feature essential account management functionalities, including sign-up and login processes. Users must sign up to create an account before accessing the application. Once logged in, they can edit their profiles to update personal information as needed.

To ensure system reliability, fallback mechanisms will be implemented to handle potential downtimes of third-party services like the GEMINIAI API or Google Cloud. This ensures continuous operation even when dependencies are temporarily unavailable.

## 1.3. Definitions, Acronyms, and Abbreviations

### Definitions

- **ROI:** The specific area or part of an image where image processing and disease detection algorithms focus.
- **Deep Learning:** A subset of artificial intelligence focused on using neural networks with multiple layers to process data and improve accuracy through experience.
- **User Interface:** The graphical or textual elements through which users interact with the system.
- **Backend:** The server-side component of the system responsible for processing image data, executing disease detection algorithms, managing database operations, and facilitating API communication.
- **Database:** A structured collection of data, organized to allow easy retrieval, modification, and management. The system will use a NoSQL database to handle diverse and unstructured data, disease descriptions.
- **Confidence Level:** A measure of the system's certainty in the accuracy of disease detection results.

### Acronyms and Abbreviations

SRS: Software Requirements Specification

UI: User Interface

ROI: Region of Interest

DL: Deep Learning

DB: Database

## 1.4. References

- Zeynep, (2020). PLANT DETECTION IN AERIAL IMAGES USING DEEP NEURAL NETWORKS FOR SMART AGRICULTURAL APPLICATIONS M.Sc. THESIS. Bayraktar.
- Melis, (2022). SMART FARMING APPLICATIONS, M.S. Thesis. Siropyan.
- Furkan Alp, (2022). Derin Öğrenme Tabanlı Akıllı Tarım ve Uygulamaları Yazılım Mühendisliği Ana Bilim Dalı Yüksek Lisans Tezi. Esen.
- IEEE Guide for Software Requirements Specifications, in IEEE Std 830-1984, vol., no., pp.1-26, 10 Feb. 1984, doi: 10.1109/IEEESTD.1984.119205.

## 1.5. Overview

The remaining sections of this SRS document provide a comprehensive description of the Plant Disease Detection project, including its user characteristics, product perspective, functional requirements, constraints, assumptions, and dependencies.

- **Section 2** offers a general description of the project, discussing the target audience, scope, and product functionalities. It also includes user characteristics and outlines the product's overall objectives.
- **Section 3** provides detailed functional requirements, specific constraints, and the user perspective on the system. This section also describes the external interface requirements, user interfaces, hardware, and software integrations, and includes a comprehensive breakdown of the system's specific requirements.
- **Section 4** addresses non-functional requirements
- **Section 5** concludes with a summary of assumptions, dependencies, and constraints, ensuring a holistic view of the project's requirements and potential limitations.

## **2.General Description**

### **2.1. Product Perspective**

The Plant Disease Detection project aims to empower farmers, gardeners, and agricultural professionals by providing actionable insights about plant health. Users can simply take a photo of their plants to receive disease detection results and recommended treatments. This feature enables users to improve the efficiency and productivity of their agricultural activities while gaining valuable knowledge about plant health and disease prevention.

In many cases, accessing agricultural professionals can be challenging due to factors remote locations, limited availability, or cost constraints. By offering an accessible and reliable alternative, the Plant Disease Detection application simplifies the process of identifying plant diseases, saving users time and effort.

The system relies on modern technologies, including mobile platforms for ease of use, deep learning algorithms for disease detection, APIs for retrieving treatment recommendations, and cloud services for secure data storage. Together, these components ensure a seamless user experience and deliver timely, accurate insights into plant health.



## 2.2. Product Functions

### 2.2.1. Disease Detection

- **Image Uploading**

Users will upload images of their plants to the system to obtain information about potential diseases. The system will support commonly used image formats JPEG and PNG, with a maximum file size limit of 5 MB to ensure compatibility and performance. Users will be prompted to ensure their images are clear, well-lit, and focused to improve detection accuracy. If the uploaded image does not meet these quality standards, the system will display an error message and request a new image.

- **Image Processing**

The system will apply image processing algorithms to identify regions of interest (ROI) within the uploaded image. This process involves detecting key areas, leaves or stems, using techniques like bounding boxes or segmentation. The identified ROI will then be extracted and prepared for analysis by the disease detection module.

- **Disease Detection**

The system will classify and identify diseases by analyzing the regions of interest using a deep learning model, a convolutional neural network (CNN). The model will be trained on labeled plant disease datasets to recognize patterns and make predictions. The detection results will include a confidence level to indicate the certainty of the prediction. This ensures users can make informed decisions based on the system's analysis.

- **Performance Metrics**

The disease detection module is designed to achieve an accuracy rate of at least 90%, validated using the F1-score. In cases where the system cannot confidently detect a disease, it will provide users with suggestions for improving their image quality or direct them to consult an agricultural professional.

## 2.2.2. Community Forum

The Plant Disease Detection project will include a **community forum** where users can share information and experiences about specific plant diseases. This feature fosters collaborative knowledge-sharing within the application, enabling users to learn about diseases they may not have encountered before. By encouraging open discussion and collective learning, the forum contributes to advancing farming knowledge as a community.

### Forum Features and Structure

- **Threaded Discussions:** Users can create posts and participate in threaded discussions by replying to existing posts. Posts can also be liked or disliked to indicate their usefulness.
- **Search and Filtering Options:** The forum will include search functionality, allowing users to find discussions based on keywords, disease types, or post dates. Filters will help categorize posts for efficient navigation.

### User Roles and Permissions

The forum will define specific user roles to manage interactions:

- **Regular Users:** Can create posts, reply to discussions, and interact with content (like or dislike posts).
- **Moderators:** Monitor the forum to ensure posts adhere to community guidelines and filter inappropriate or irrelevant content.
- **Administrators:** Manage user roles, remove flagged content, and oversee overall forum activity.

### Content Moderation

To ensure the reliability of shared information, moderators will verify posts for accuracy and appropriateness. Community guidelines will be established to promote constructive and respectful discussions. Inappropriate or inaccurate content can be reported by users and will be reviewed by moderators or administrators.

### Data Storage and Retention

User-generated content will be securely stored using cloud-based services. A retention policy will be implemented to archive older posts that are no longer actively accessed, ensuring scalability and efficient storage management.

By implementing these features, the community forum will become a valuable resource for users to exchange knowledge, solve problems collaboratively, and contribute to a more informed farming community.

### 2.2.3. Map - Disease Visualization

The application will include a map feature that visualizes the distribution of plant diseases across the country. The map will be displayed on the **Disease Distribution Page** and provide users with insightful statistics to better understand regional and temporal patterns of plant diseases.

#### Location-Based Display

The map will adjust its view based on the user's location, determined through GPS or manually entered location data. Permissions will be requested to access location data, and privacy measures, anonymization, will be implemented to protect user information.

#### Disease Statistics and Data Source

The map will display the following statistics:

- **Regional Data:** The most frequently occurring diseases in specific locations across the country.
- **Temporal Data:** Seasonal trends showing which diseases occur most often during specific times of the year.

The data for these statistics will be sourced from verified agricultural databases and updated regularly to ensure accuracy and reliability. Updates will occur on a daily or weekly basis, depending on data availability, and will be validated to remove anomalies or incorrect entries.

#### Interactive Features

The map will include interactive functionalities to enhance user experience:

- **Zoom and Navigation:** Users can zoom in and out and navigate to specific regions for detailed views.
- **Click for Details:** Clicking on a specific region will display detailed information about prevalent diseases and their frequencies.
- **Timeline Feature:** Users can view disease trends over time using an adjustable timeline slider.

#### Data Visualization Methods

The statistics will be visualized using intuitive graphical elements:

- **Heatmaps:** To display the intensity of disease occurrences in different regions.
- **Markers:** To highlight specific areas with information pop-ups for detailed insights.
- **Charts:** To compare disease trends over time or across regions.

#### Privacy and Security

User location data will be collected only with explicit permission and will be anonymized before being used for statistical aggregation. No personally identifiable information will be associated with the displayed data.

By incorporating these features, the map will serve as a valuable tool for users to identify regional and temporal disease patterns, enabling informed decisions and proactive plant care.

## 2.2.4. Sign Up

To use the application, users must first create an account. Upon opening the application for the first time, the **Sign Up Page** will be displayed. Users will need to provide the following required attributes:

- **Name:** The user's first name.
- **Surname:** The user's last name.
- **Username:** A unique identifier that must not match any existing username in the system.
- **Password:** A secure password, requiring a minimum of 8 characters, including at least one uppercase letter, one number, and one special character.
- **Email:** A valid email address for account authentication and recovery.
- **City:** The user's location (optional).
- **Occupation:** The user's profession (optional).
- **Gender:** Male, female, or prefer not to specify (optional).

### Input Validation and Feedback

The system will validate all required fields before allowing account creation. For example:

- If the username is already taken, the user will be prompted to choose a different one.
- Weak passwords or invalid email formats will trigger error messages with specific guidance on how to correct the input.

### Confirmation Process

After successfully entering all required information, users will click the **Submit** button. The system will send a confirmation email containing a verification link. Users must verify their email address before their account becomes active. This mechanism ensures user identity and prevents spam registrations.

### Data Storage and Security

- All user information will be securely stored in the cloud using encrypted databases.
- Passwords will be hashed using industry-standard algorithms (bcrypt) to ensure security.
- Data privacy measures will comply with regulations like GDPR, ensuring user trust and legal compliance.

By implementing these measures, the sign-up process will ensure a secure and user-friendly experience, supporting seamless onboarding for new users.

## 2.2.5. Login

If a user already has an account, they do not need to create a new one. To log in, users must provide their registered **email address** and **password** on the Login Page.

### Authentication Process

1. The system will validate the provided email and password against stored credentials in the database.
2. If the credentials are correct, the user will be successfully logged in and redirected to the Home Page.
3. If the credentials do not match, a generic error message, *“Invalid email or password”*, will be displayed. This approach prevents exposing which part of the input is incorrect, enhancing security.

### Forgot Password Feature

If a user forgets their password, they can click on the **“Forgot Password”** link on the Login Page. The system will prompt the user to enter their registered email address. A password reset link will be sent to the email, allowing the user to create a new password securely.

### Security Measures

- **Encryption:** All passwords are securely hashed using industry-standard algorithms (bcrypt) before being stored in the database.
- **Account Lockout:** To prevent brute-force attacks, accounts will be temporarily locked after five consecutive failed login attempts. The lockout will reset automatically after 15 minutes.
- **Session Management:** After a successful login, the user session will remain active for a maximum of 24 hours. Users will be automatically logged out after 15 minutes of inactivity to protect account security.

By implementing these measures, the login process ensures a secure, efficient, and user-friendly experience while adhering to modern security practices.

## 2.2.6. Edit Profile

The **Edit Profile** feature allows users to update their account information securely and efficiently. Users can access this feature from the application's profile page to make changes to specific fields without being required to re-enter all their information.

### Editable Fields

Users can modify the following attributes:

- **Name:** Update their first name.
- **Surname:** Update their last name.
- **Username:** Choose a new username that is unique and not already in use by another account.
- **Password:** Change their password after entering the current password to verify their identity. New passwords must meet security requirements, including a minimum of 8 characters, at least one uppercase letter, one number, and one special character.
- **Email:** Update their email address, which will require verification through a confirmation email.
- **City:** Update their location (optional).
- **Occupation:** Update their profession (optional).
- **Gender:** Update or remove gender information (optional).
- **Profile Picture:** Upload or change their profile picture for personalization.

### Validation Rules and Feedback

- All updated fields will be validated according to the system's data integrity rules (ensuring unique usernames and strong passwords).
- If any field contains invalid data or duplicates, the user will receive a specific error message with guidance on how to resolve the issue.

### Confirmation and Notifications

After making changes, users must click the **Save Changes** button to apply updates. A confirmation message will notify users of successful updates, and changes will be reflected in real time. For security-critical updates, email or password changes, the user will receive an email notification to confirm the modification.

### Security and Privacy

- **Authentication:** Users must verify their identity by re-entering their current password when making sensitive updates, changing their password or email address.
- **Data Storage:** All updated user data will be securely stored in an encrypted database, ensuring compliance with privacy standards like GDPR.
- **Real-Time Updates:** Changes will be reflected immediately in the user's account to ensure a seamless experience.

By offering flexibility and robust security, the Edit Profile feature ensures users can manage their account information confidently and conveniently.

## 2.3. User Characteristics

The application supports two primary user roles: **Users** and **Admins**, each with distinct functionalities and responsibilities.

### Users

Users interact with the application to access its core features, which include:

1. Uploading images of their plants to receive disease detection results and treatment recommendations.
2. Creating posts or replying to discussions in the **Community Forum** to share knowledge and seek advice.
3. Viewing disease statistics via the **Disease Distribution Map**, which provides insights into regional and temporal patterns.
4. Editing their profile to update account details, username, password, email, and profile picture.
5. Viewing detailed reports about their plant's health and disease analysis.
6. Accessing general features like navigation between different sections of the application (Home Page, Help Page).

### Admins

Admins are responsible for overseeing the application's operations and maintaining its integrity. Their responsibilities include:

1. Moderating the **Community Forum** by managing posts, addressing flagged content, and ensuring adherence to community guidelines.
2. Managing user accounts, including granting or revoking access and handling user-reported issues.
3. Monitoring system performance and ensuring data consistency across all modules.
4. Performing data analysis to improve the accuracy of disease detection models and ensure the reliability of disease statistics.
5. Configuring and updating content, disease descriptions and treatment recommendations.

### Access Levels and Permissions

- **Registered Users:** Gain full access to the application's functionalities, including uploading images, participating in the community forum, and editing profiles.
- **Guest Users:** Can browse public sections of the application, the **Community Forum**, but cannot upload images or post until they register.

### User Skill Levels

The application is designed for users with varying technical skills and agricultural knowledge:

- **General Users:** Farmers, gardeners, and enthusiasts with minimal technical expertise can easily interact with the application due to its intuitive interface.
- **Advanced Users:** Agricultural professionals with domain knowledge can leverage advanced features, detailed statistics and disease reports.

### User-Application Interactions

Users and admins will interact with the system through a user-friendly graphical interface. Role-based access controls will ensure each user type can perform their specific tasks without overlapping permissions. For instance, admins will have additional controls for moderating content and managing user accounts, which are not visible to regular users.

This clear distinction of user roles ensures that the system meets the needs of its diverse audience while maintaining operational efficiency and security.



## 2.4. General Constraints

The system is subject to the following constraints, which guide its implementation and operation:

### 1. Language Support

The system must support both Turkish and English languages, allowing users to select their preferred language for all user interface elements.

### 2. Platform Requirements

The application must be developed as a mobile application and must be compatible with both iOS and Android platforms.

### 3. Design Constraints

The primary color for the application's user interface will be green, reflecting its agricultural theme. Accessibility concerns, ensuring sufficient contrast for colorblind users, will be addressed to comply with usability standards.

### 4. Deep Learning Model

The performance of the system is dependent on the accuracy of the deep learning model used for disease detection. The model must achieve a minimum accuracy of 80%, measured using the F1-score, to meet user expectations. Performance may vary based on the quality of input images and environmental factors.

### 5. Environmental Factors

The system's effectiveness is influenced by external factors weather conditions, which may affect image quality. For instance, low lighting or poor camera focus can reduce detection accuracy. To mitigate these issues, the application will prompt users to retake images in better lighting conditions or use clearer images.

### 6. Network Connectivity

A stable internet connection is required to use the application. Core functionalities, disease detection, rely on real-time communication with cloud servers and third-party APIs. Offline functionality will be limited to viewing previously downloaded reports or static content.

### 7. Image Processing Libraries

The system will utilize image processing libraries OpenCV or TensorFlow, which may introduce constraints related to performance, compatibility, or platform-specific dependencies. These constraints will be evaluated during implementation to ensure seamless integration.

By adhering to these constraints, the system will maintain compatibility, usability, and performance, delivering a reliable experience for users across different environments and devices.

## 2.5. Assumptions and Dependencies

The system relies on the following third-party services and assumes their availability and functionality under normal operating conditions. However, fallback mechanisms and contingency plans will be implemented to mitigate the risks associated with potential service interruptions.

### 1. GEMINI API

The GEMINI API will be used to retrieve treatment recommendations for detected plant diseases. While we assume the GEMINI API will function reliably, the following considerations address potential risks and dependencies:

- **Fallback Strategy:** In case the GEMINI API is unavailable, the system will display a predefined set of generic treatment suggestions stored locally.
- **Integration Details:** The system will integrate with the GEMINI API using RESTful calls. The API is expected to support JSON as the primary data format for requests and responses. Error-handling mechanisms will manage scenarios like invalid responses or timeouts.
- **Performance Expectations:** The GEMINI API should respond to treatment requests within 2 seconds to maintain overall system performance.

### 2. Google Cloud Service

Google Cloud Service will be used for secure storage of data, including disease descriptions, treatment recommendations, and user-generated content. The following assumptions and dependencies apply:

- **Availability:** While uninterrupted service is assumed, the system will cache frequently accessed data locally to reduce dependence on real-time connectivity with Google Cloud.
- **Data Security:** All data stored in Google Cloud will be encrypted using industry-standard protocols, ensuring compliance with privacy regulations GDPR.
- **Redundancy and Backup:** Regular backups of critical data will be maintained to prevent data loss during unexpected outages.

### 3. General Assumptions and Mitigation Strategies

- **Service Interruptions:** The system assumes that third-party services may occasionally experience downtime or degraded performance. Mitigation strategies, retry logic and user notifications, will ensure continuity of service.
- **Dependencies Across Modules:**
  - **GEMINI API:** Impacts treatment recommendations displayed in the disease report module.
  - **Google Cloud:** Affects data storage for reports, user-generated content in the community forum, and disease descriptions.

- **Latency Tolerance:** The system will allow a maximum API latency of 2 seconds for GEMINI API calls and ensure seamless user experience with local caching for critical operations.

By defining these assumptions and dependencies along with fallback mechanisms, the system ensures reliability, data security, and performance even in the face of third-party service interruptions.

## 3. Specific Requirements

### 3.1. External Interface Requirements

#### 3.1.1. User Interfaces

##### 3.1.1.1. Login Page UI Design

The Login Page offers a secure, user-friendly interface with essential components for access. It includes input fields for email and password, with email validation and a show/hide toggle for password visibility. The "Login" button activates only when valid data is entered, and a "Forgot Password?" link facilitates password recovery. Error messages like "Invalid email or password" appear below fields for immediate feedback. Security features include CAPTCHA verification and account lockout after five failed attempts, while accessibility options like ARIA attributes, high-contrast mode, and keyboard navigation enhance usability. Input validation ensures proper email format and password complexity (minimum 8 characters). Additional features include a "Remember Me" checkbox for session persistence and user feedback on login status, redirecting successful logins to the Home Page.

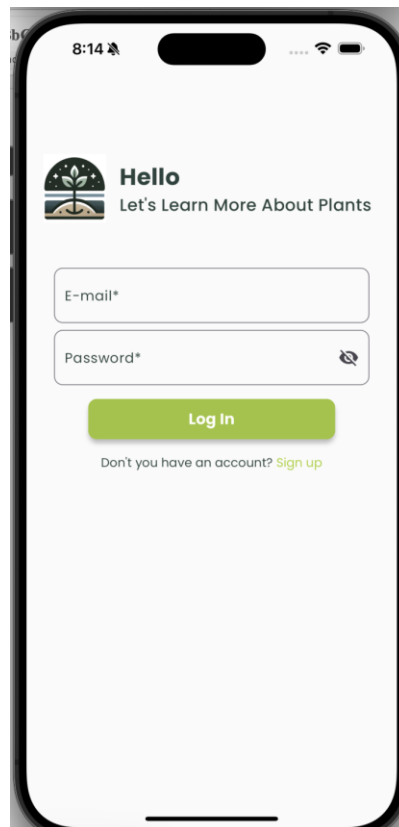


Figure 1 Login Page

### 3.1.1.2. Home Page UI Design

The Home Page serves as the application's central hub, offering intuitive navigation and personalized features. The header includes a clickable logo for home redirection, a user profile dropdown for settings, and a notifications bell for updates on treatments, forum posts, and trending diseases. A horizontal or collapsible navigation menu provides quick access to features like image upload, the community forum, disease maps, and reports, with hover effects for visual feedback. The main content area greets users with a welcome message, quick access cards, recent activity, and recommendations based on user interactions. A search bar enables easy feature or content discovery, while the footer links to About Us, Contact Us, and Help Center. Accessibility is prioritized with features like font scaling, high-contrast mode, ARIA roles, and a responsive design ensuring usability across devices. Interactive elements include hover effects, loading indicators, and a dynamic layout for seamless user experience.

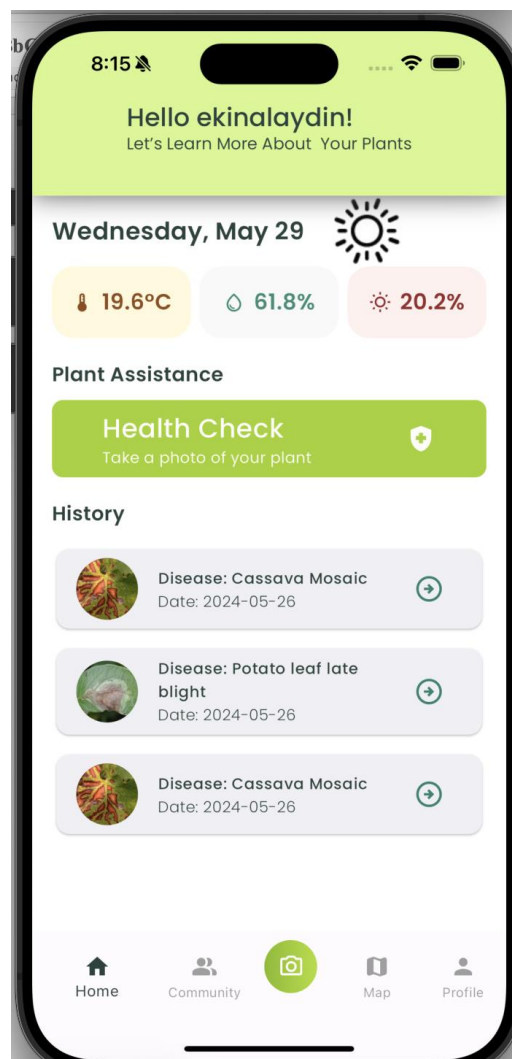


Figure 2 Home Page

### 3.1.1.3. Report Page UI Design

The Report Page provides a detailed summary of plant health analysis in an intuitive and accessible layout. The header highlights the report title, analysis date, and time. A disease information section displays the identified disease, confidence level with visual indicators, and detailed treatment suggestions sourced from the GEMINIAI API. The image analysis section shows both the uploaded and processed images with annotated areas of interest. Visualization features include confidence and comparison charts, while a feedback section offers retry options if no disease is detected. Export options allow saving the report as a PDF or sharing via email and social media. The responsive design ensures compatibility with various devices, supporting scrolling or pagination for extensive content. Accessibility features include multilingual support, high-contrast mode, and legible fonts. Error handling provides clear messages and guidance in case of detection issues or API downtime.

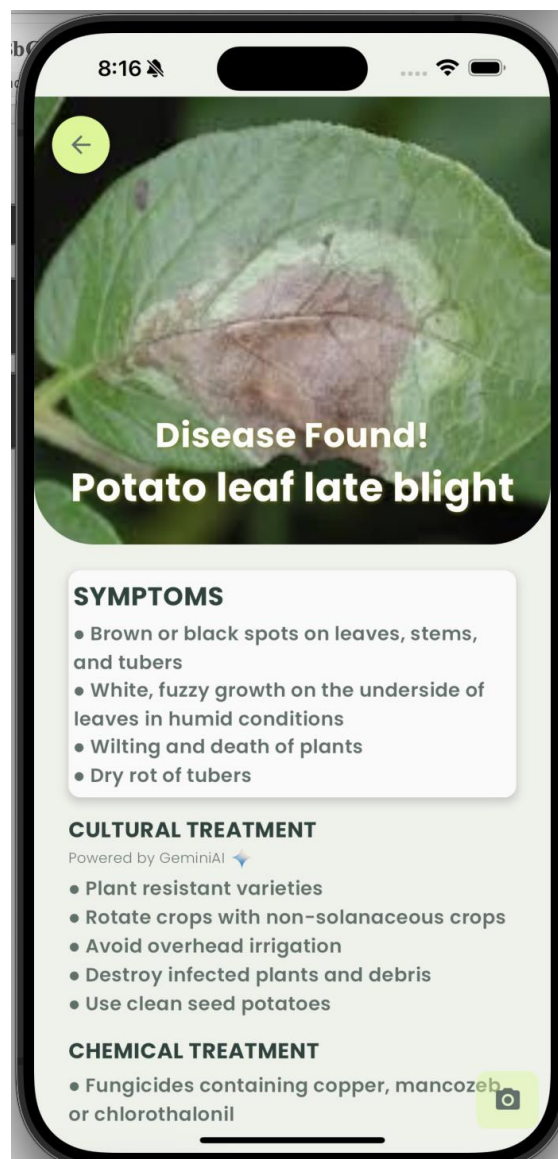


Figure 3 Report Page

### 3.1.1.4. Community Page UI Design

The Community Page fosters user interactions through sharing knowledge and discussions about plant diseases and treatments. A header features a search bar and filters for sorting posts by keywords, disease types, or activity. Posts are displayed in a list or card format, showing titles, content previews, timestamps, authors, and engagement metrics like likes, dislikes, and comments. Users can interact through likes, threaded comments, or by creating new posts via an "Add Post" button, which opens a post creation form with options for titles, tags, and file uploads. Notifications alert users to updates on their posts or replies. Accessibility is ensured with font resizing, high-contrast themes, and screen reader compatibility. Moderation tools allow admins to manage flagged content, while users can report posts violating community guidelines. The page supports infinite scrolling or "Load More" buttons for seamless navigation, alongside features like bookmarking and pinned posts for quick access to key content.

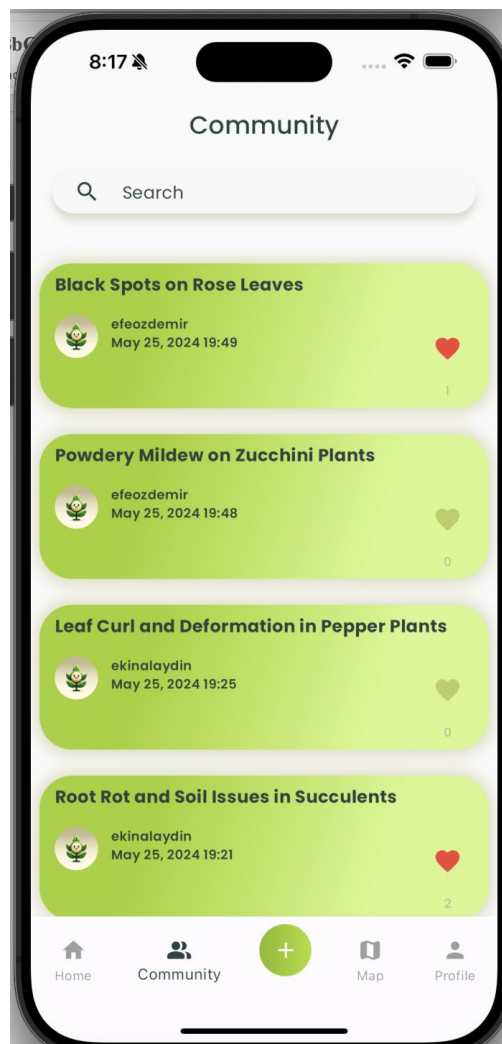


Figure 4 Community Page

### 3.1.1.5. Disease Distribution Page UI Design

The Disease Distribution Page provides an interactive map for visualizing plant disease occurrences across the country. Users can zoom, navigate, and click on regions to view detailed statistics, common diseases, and trends. Legends explain color codes for severity and markers for disease types, while filters allow data refinement by date range, disease type, or region. A heatmap highlights prevalence, and adjacent charts display trends and distributions. The map dynamically updates with real-time data from verified sources like the GEMINIAI API, with daily updates ensuring accuracy. In no-data scenarios, the map prompts users to refine filters while remaining interactive. Accessibility features include high-contrast mode, screen reader compatibility, and keyboard navigation. Users can opt for location-based notifications and either allow GPS detection or manually enter locations, with privacy ensured through anonymized data handling.

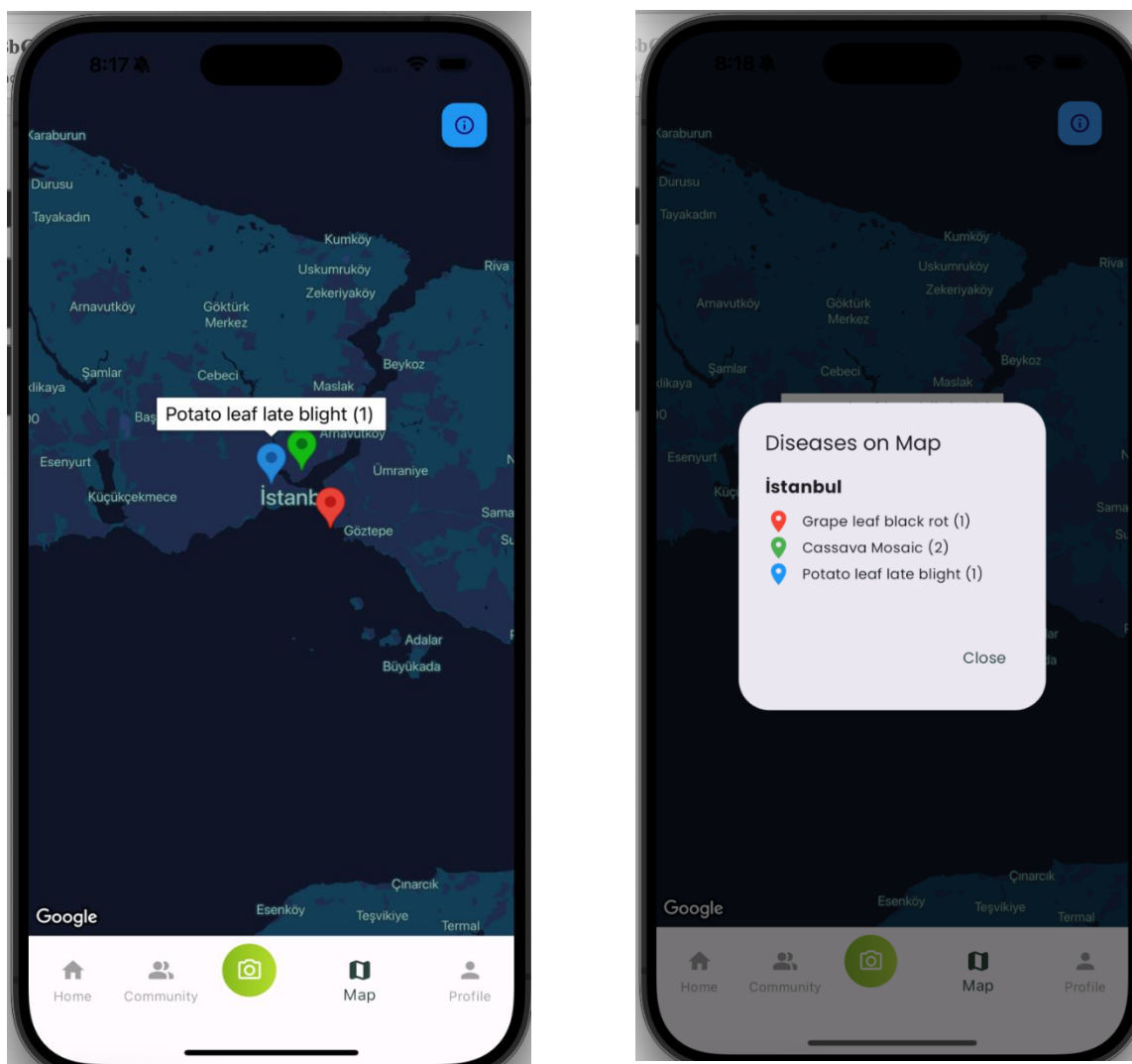


Figure 5 Disease Distribution Page



### 3.1.1.6. Post Page UI Design

The Post Page offers an intuitive interface for creating and sharing posts in the community. It features a header with a title, "Create a Post," and a back button to return to the Community Page. Users can input a post title (up to 100 characters) and content (up to 2,000 characters) with character counters. An optional image upload feature supports JPEG, PNG, and GIF files up to 5 MB. Action buttons include "Post" to submit, "Save as Draft" for later editing, and "Cancel" to discard the draft. Confirmation dialogs appear before submission, and users receive feedback on successful submissions or errors (image upload issues). Posts can be saved as drafts and edited later, with edited posts marked for transparency. Input validation ensures fields are filled, and error handling automatically saves drafts if issues arise. Accessibility features include ARIA roles, keyboard navigation, and a high-contrast mode. All content is subject to moderation to uphold community guidelines, with flagged posts reviewed by moderators.

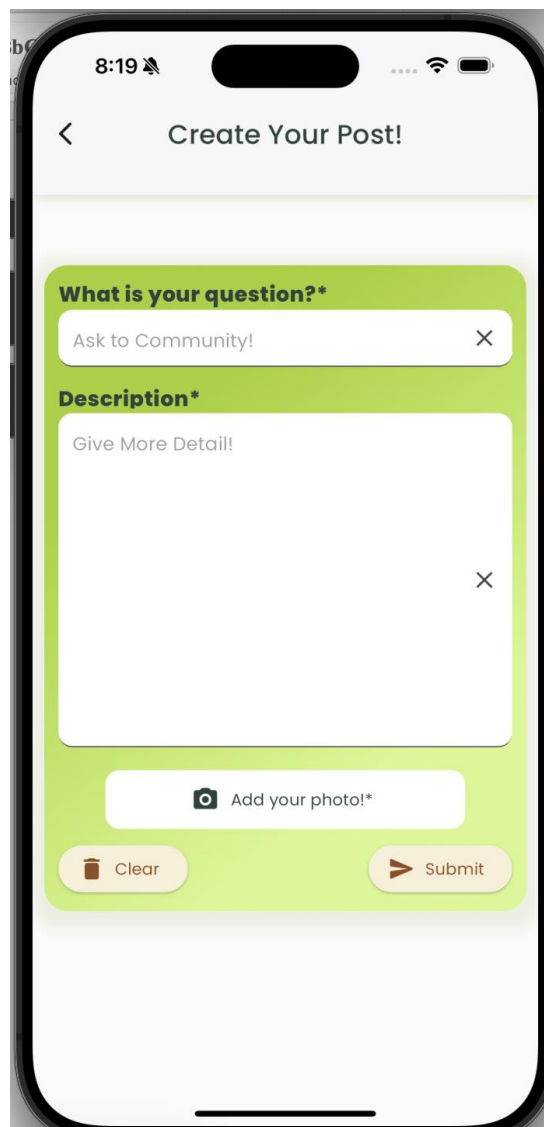



Figure 6 Create a Post Page

### 3.1.1.7. Sign Up Page UI Design

The Sign Up Page provides a secure and user-friendly interface for account creation. It features a header with the title "Create Your Account" and a back button to return to the Login Page. Input fields include mandatory fields like name, username (validated for uniqueness), email (validated for format and duplication), password (meeting complexity requirements), and confirm password, along with optional fields like city, occupation, and gender. Validation rules ensure all required fields meet criteria, displaying errors for issues like duplicate usernames or invalid passwords. Action buttons include "Sign Up" to submit the form and "Cancel" to return to the Login Page. After successful registration, users see a confirmation message and receive an email verification link. Accessibility features include ARIA roles, keyboard navigation, and high-contrast mode. Security is ensured through SSL encryption, hashed passwords, and CAPTCHA to prevent bot activity. Optional fields and clear feedback make the process inclusive and straightforward.

10:50


<

 **Hello**  
Let's Learn More About Plants

Name\* Surname\*

Username\*

E-mail\*

Password\* 

Your password must be between 4-12 characters.

City ▼

Occupation

Gender ▼

**Sign Up**

Figure 7 Sign up Page.

### 3.1.1.8. Edit Profile Page UI Design

The Edit Profile Page offers a secure and user-friendly interface for updating account information. Users can edit fields like name, username (validated for uniqueness), email (validated for format and duplication), password (with strict security requirements), and optional fields like city, occupation, and gender. A profile picture can also be uploaded or changed, with supported formats being JPEG, PNG, and GIF (up to 5 MB). Mandatory fields are marked, and real-time validation provides feedback for errors, such as duplicate usernames or invalid passwords. Action buttons include "Save Changes" to update the profile and "Cancel Changes" to discard edits. Security is ensured by requiring the current password for sensitive updates, encrypting all transmitted data. Accessibility features include ARIA roles, keyboard navigation, and high-contrast mode. Users receive feedback after saving changes or an error message if updates fail, ensuring a smooth and reliable experience.

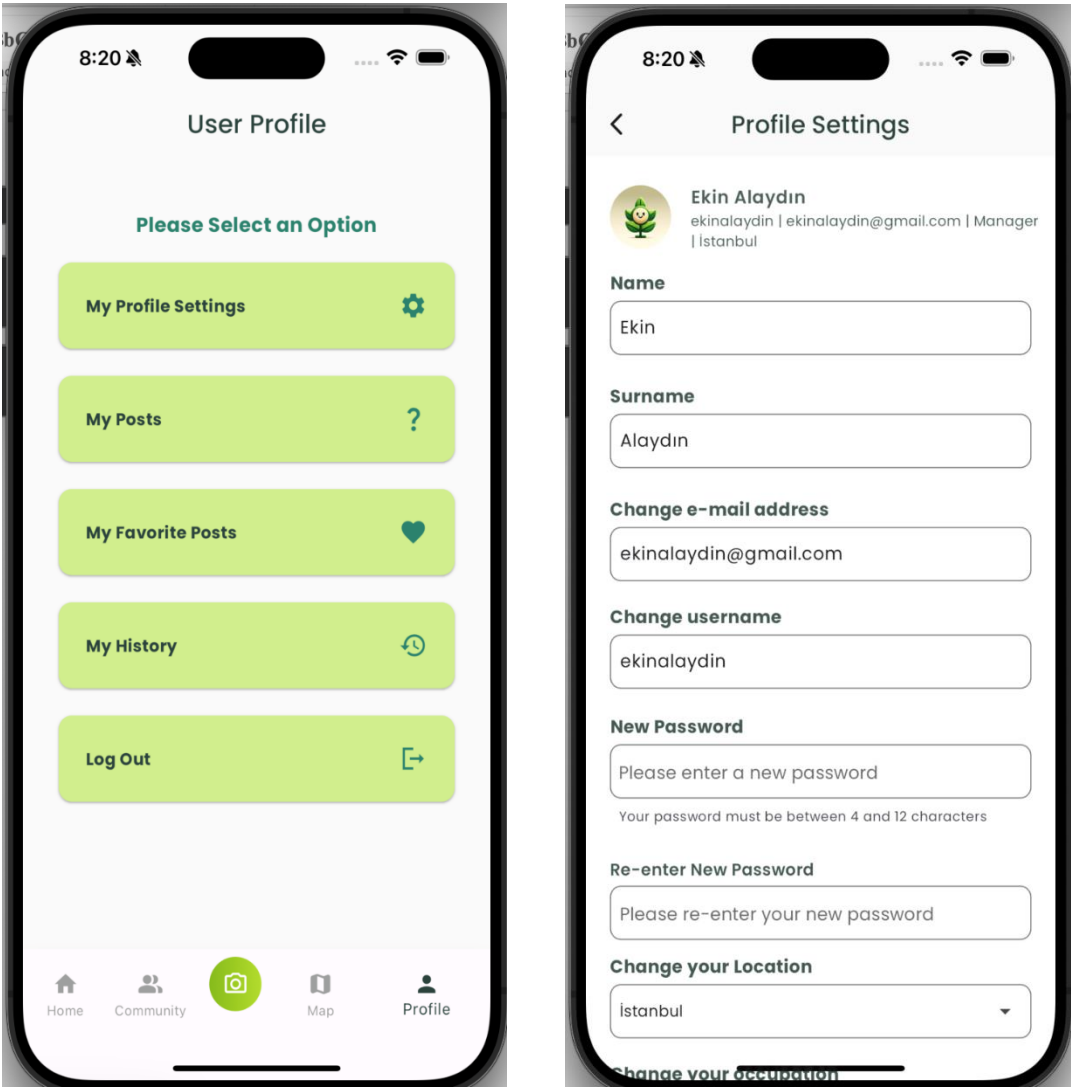


Figure 8 Edit Profile Page

### 3.1.1.9. Camera Page UI Design

The Camera Page enables users to capture and submit high-quality images for disease detection with an intuitive interface. The real-time camera viewfinder occupies most of the screen, guiding users to center their plant using a border overlay. Control buttons include a "Capture" button for taking photos, a "Retake" button for discarding and retaking images, and a "Confirm" button to proceed. Additional features include a flash toggle for lighting adjustments and a camera switch button for front or rear camera use. Feedback messages prompt users to improve image quality if blurry or poorly lit. Images must meet requirements like a minimum resolution of 1280x720 pixels, proper lighting, and a maximum file size of 10 MB. Accessibility is enhanced with voice commands, haptic feedback, and screen reader support. In case of camera issues, users are guided to enable permissions or upload images from their gallery. Captured images are stored temporarily until confirmation, securely uploaded to the cloud, and cached locally to prevent data loss. Poor-quality images trigger prompts for retakes with clear improvement suggestions.

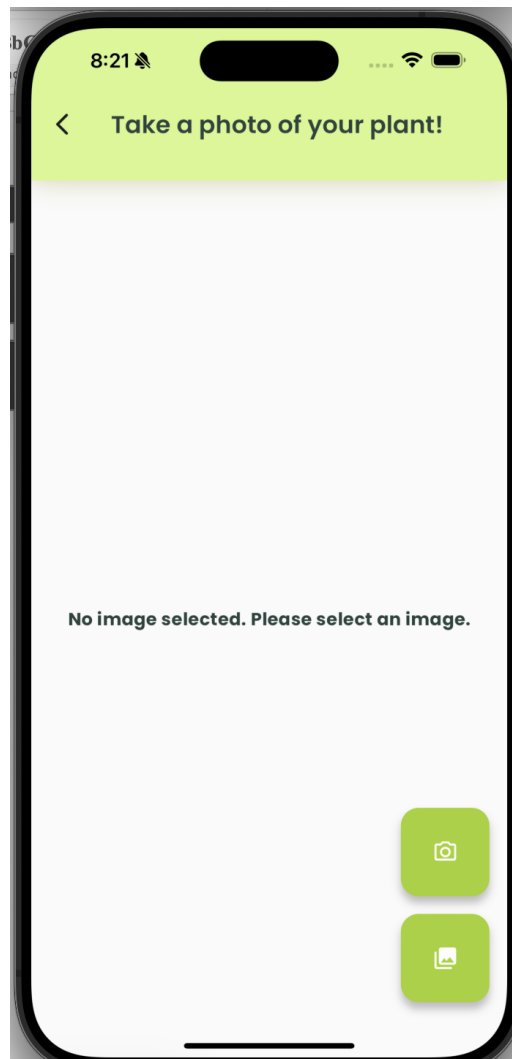


Figure 9 Camera Page

### 3.1.2. Hardware Interfaces

The system relies on the following hardware components to ensure compatibility, performance, and security:

#### 1. Camera Devices

- **Compatibility Requirements:**
  - The system must support built-in cameras on mobile devices, with the following minimum specifications:
    - Resolution: At least 8 megapixels for capturing detailed images.
    - Autofocus Capability: Required to ensure image clarity.
    - Flash Support: Optional but recommended for low-light conditions.
- **External Device Support:**
  - The system can optionally integrate with external hardware USB cameras or drones for image capture, provided they meet the resolution and focus requirements.
- **Access Permissions:**
  - The application will request camera permissions as per Android and iOS platform guidelines. Users must grant these permissions to capture images. If denied, the system will notify the user and provide instructions for enabling permissions in settings.

#### 2. Server Hardware

- **Performance Benchmarks:**
  - The server infrastructure must handle up to 10,000 concurrent user requests with a response time of less than 2 seconds under normal operating conditions.
  - Scalability: The server must support dynamic scaling to accommodate increased demand during peak usage.
- **Storage Requirements:**
  - Initial capacity for storing 1 TB of user-generated data, with options for expansion as needed.
- **Data Security:**
  - All stored data will be encrypted using industry-standard protocols (AES-256).
  - Access control mechanisms will ensure only authorized personnel can access the server infrastructure.
  - Regular security audits and updates will be conducted to address vulnerabilities and maintain compliance with data protection regulations.

### 3. Mobile Devices

- **Supported Platforms:**
  - Android devices running Android 8.0 (Oreo) or later.
  - iOS devices running iOS 12 or later.
- **Minimum Device Specifications:**
  - RAM: At least 2 GB.
  - Storage Space: Minimum of 100 MB available for application installation and data caching.
- **Offline Functionality:**
  - The application will allow users to capture images offline. These images will be stored locally and uploaded for analysis once an internet connection is available.

### 3.1.3. Software Interfaces

The system relies on the following software interfaces to ensure compatibility, performance, and secure operations:

#### 1. Operating Systems

- **Supported Platforms:**
  - The mobile application will be compatible with Android and iOS operating systems.
  - **Android:** Minimum version 8.0 (Oreo) or later.
  - **iOS:** Minimum version 12 or later.

#### 2. Image Processing Libraries

- **Libraries Used:**
  - OpenCV: For image preprocessing, resizing, denoising, and identifying regions of interest (ROIs).
  - TensorFlow Lite: For deploying machine learning models optimized for mobile devices.
- **Dependency Management:**
  - Versioning will ensure compatibility with the mobile platforms and provide consistent results across environments (OpenCV 4.x, TensorFlow Lite 2.x).

#### 3. Deep Learning Frameworks

- **Frameworks Used:**
  - TensorFlow and Keras: For training and deploying deep learning models to detect plant diseases.
  - PyTorch: For experimentation and benchmarking alternative model architectures.
- **Model Versions:**
  - All models will follow a versioning system to track updates and improvements (DiseaseNet v1.0).

#### 4. Database Management System

- **Database Type:**
  - Google Cloud Firestore, a NoSQL database, will be used for storing user data, reports, disease descriptions, and other metadata.
- **Performance Benchmarks:**
  - Query response time: Less than 500 milliseconds for typical operations under normal load.
  - Initial storage capacity: 1 TB, with scalability options for future growth.
- **Error Handling:**
  - In case of database downtime, the system will provide appropriate messages ( *“Service temporarily unavailable. Please try again later.”* ).
  - Cached data will be used for read-only operations during outages.

#### 5. External APIs

- **GEMINI API Integration:**
  - The system will use the GEMINI API to fetch disease descriptions and treatment suggestions.
  - **Data Format:** Communication with the API will use JSON for requests and responses.
  - **Error Handling:**
    - If the API is unavailable, the system will display a fallback message: *“Unable to fetch data. Please try again later.”*
    - Retry logic will attempt to reconnect before notifying the user.
- **Security:**
  - API interactions will use authentication tokens for secure access.
  - All communications will be encrypted using HTTPS to prevent unauthorized access and data breaches.

#### 6. Security and Version Control

- **Dependency Versioning:**
  - All software libraries, frameworks, and tools will have clearly defined versions to ensure compatibility (TensorFlow 2.10.0, OpenCV 4.5.5).
- **Security Measures:**
  - API keys and authentication tokens will be stored securely and rotated periodically.
  - Data transmission will use industry-standard encryption protocols, SSL/TLS.

### 3.1.4. Communication Interfaces

The system relies on stable and secure communication interfaces to enable seamless interaction between the mobile application, backend servers, and external APIs. Below is the revised description:

#### 1. Network Requirements

- **Internet Connection:** A stable internet connection is necessary for using the application's features, image uploads and retrieving disease information.
- **Minimum Network Specifications:**
  - Bandwidth: At least 2 Mbps for smooth operation.
  - Latency: Tolerates network latency up to 200 ms without affecting core functionalities.

#### 2. Data Transmission Protocols

- **Protocol:**
  - All data transmissions between the client application and backend servers use HTTPS, ensuring secure communication via TLS/SSL encryption.
- **Encryption:**
  - Sensitive information, login credentials and user data, is encrypted during transmission to protect against unauthorized access and man-in-the-middle attacks.

#### 3. Handling Connectivity Loss

- **Retry Mechanism:**
  - If a network error occurs during critical actions (uploading images or retrieving disease data), the system will automatically retry failed requests up to three times.
- **Offline Mode:**
  - Users can capture images and temporarily save them locally when offline. These images are uploaded automatically once a stable connection is re-established.

#### 4. Communication with External APIs

- **Integration with GEMINI API:**
  - The application uses REST API calls to retrieve disease descriptions and treatment suggestions.
  - Data Format: Communication with the API is conducted using JSON for both requests and responses.
- **Error Handling for External APIs:**
  - If API requests fail, the system will display a user-friendly message: *"Unable to retrieve data at this time. Please try again later."*
  - Fallback: Cached or default data will be displayed if available.



## 5. Error Feedback to Users

- **Notification Messages:**
  - When communication issues occur, the system displays clear and actionable feedback:
    - *“Network Error – Please check your connection.”*
    - *“Unable to reach the server. Retrying...”*
- **Connection Status Indicator:**
  - A small indicator in the app’s interface shows the current network status ( “Connected” or “Offline”).

## 3.2. Functional Requirements

Plant Disease Detection System - Use Case Diagram

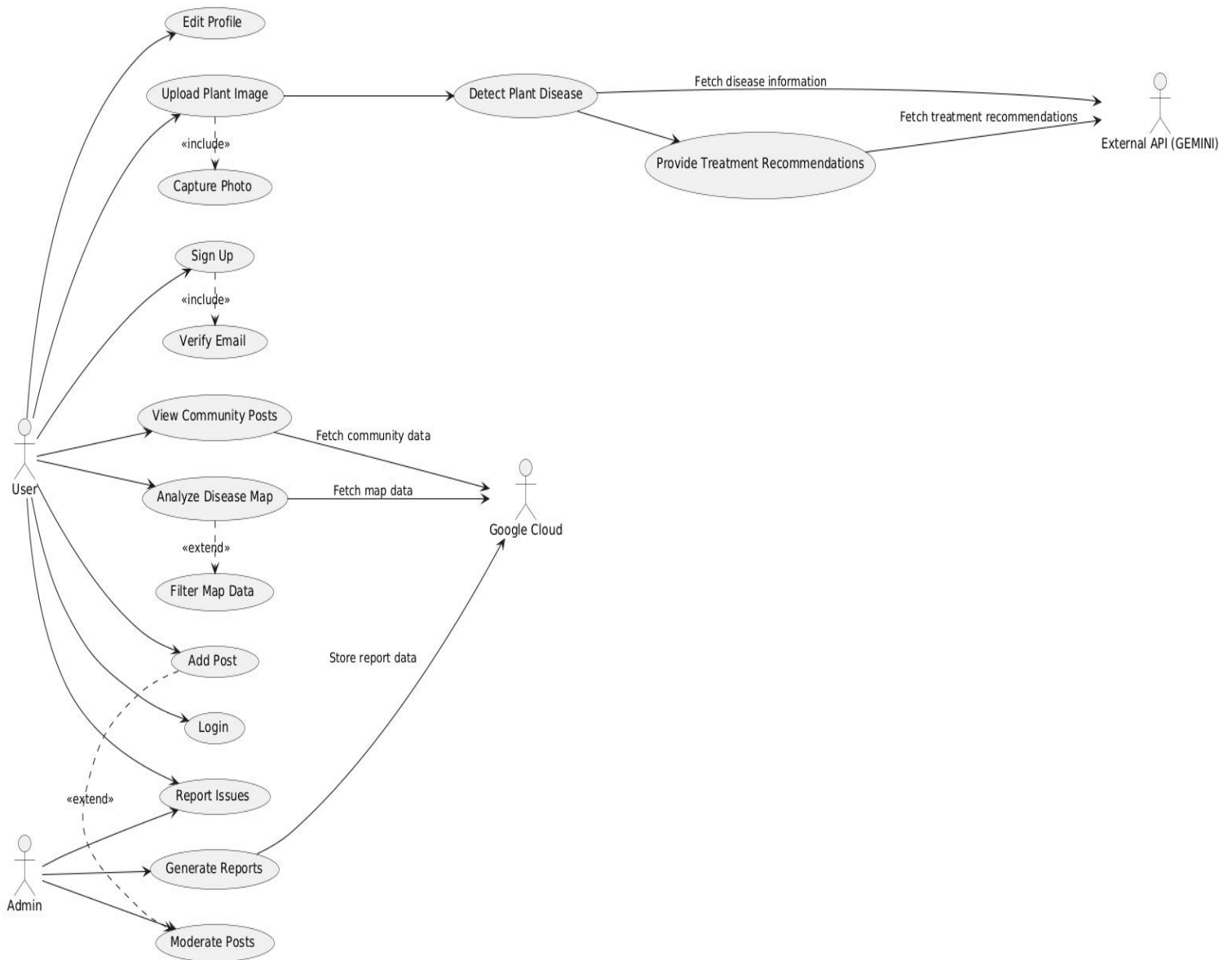


Figure 10 Use Case Diagram

# Plant Disease Detection System and Smart Agriculture

Detailed Data Model for Plant Disease Detection System

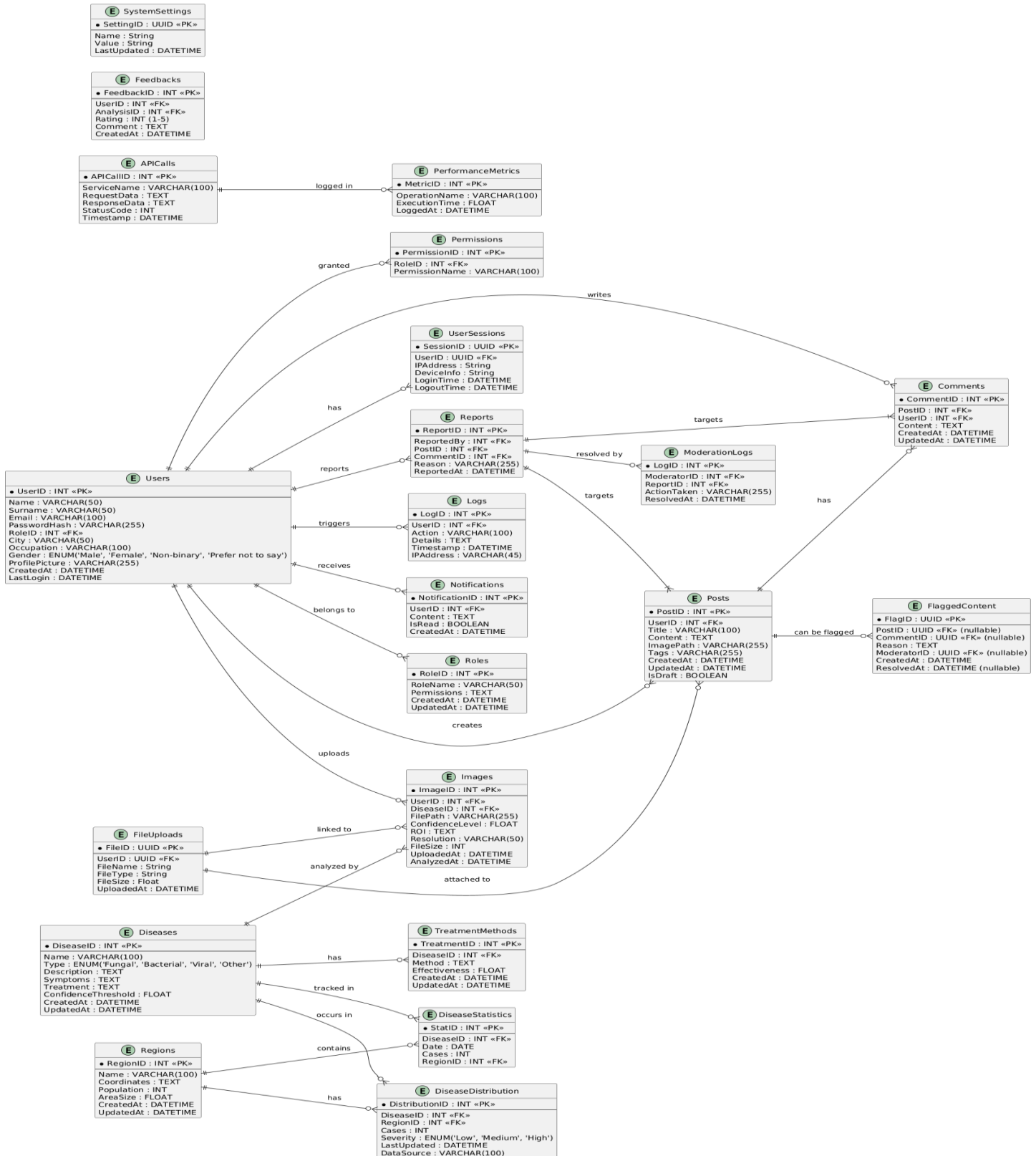


Figure 11 Data Model

# Plant Disease Detection System and Smart Agriculture

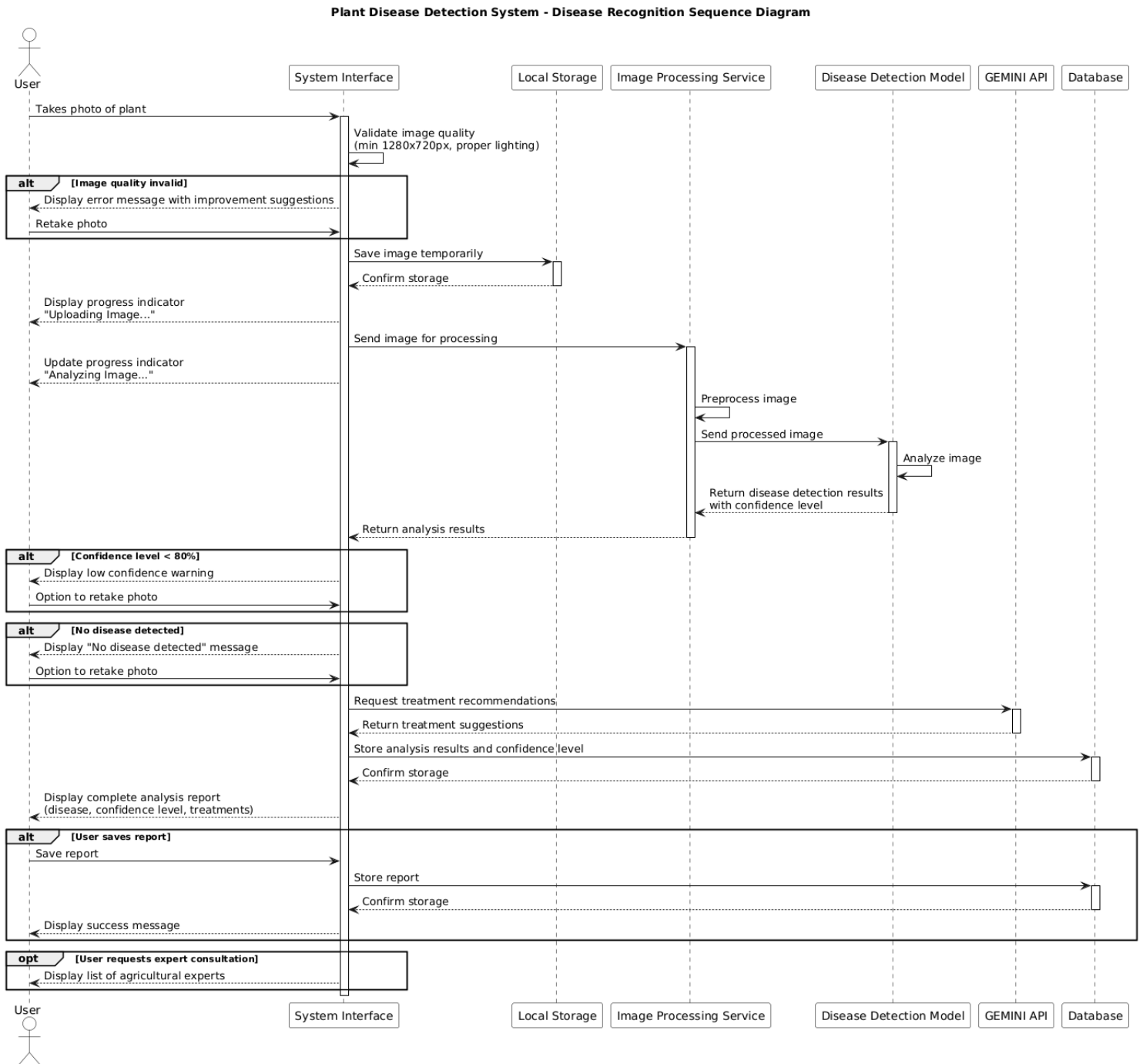


Figure 12 Sequence Diagram for Disease Recognition

### 3.2.1. Check Disease & Treatment

The following stimulus/response sequence outlines the interaction between the user and the system for checking diseases and treatments:

#### Stimulus/Response Sequence

1. **User:** Logs into the system.
2. **System:** Authenticates the user and grants access to the application.
3. **User:** Chooses one of the following options:
  - View previously diagnosed diseases and their treatments.
  - Analyze a new plant disease.
4. **System:**
  - If the input is *previous diagnoses*:
    - Retrieves previously diagnosed diseases from the server or local storage.
    - Displays a list of previously diagnosed diseases on the Home Page.
5. **User:** Selects one of the diagnosed diseases to view details.
6. **System:** Displays the treatment information and related details for the selected disease.
7. **System:**
  - If the input is *analyze a new disease*:
    - Redirects the user to the Camera Page.
8. **User:** Clicks the “**Camera**” button to activate the device camera.
9. **System:** Opens the camera interface.
10. **User:** Takes a photo of the diseased plant.
11. **System:** Prompts the user with the message:
  - “*Do you want to use this photo?*”
12. **User:** Responds to the prompt:
  - If *No*: The system allows the user to retake the photo.
  - If *Yes*: The system saves the photo securely in temporary storage.
13. **System:**
  - Displays the saved photo for user confirmation.
  - Provides the option to retake or proceed.
14. **System:** Begins the analysis process:
  - Displays a progress bar labeled “*Analyzing Image...*” to indicate the progress of disease detection.
15. **System:**
  - After successful analysis, displays the identified disease name and treatment options.
  - Provides a summary of the results, including confidence levels and additional recommendations.
16. **User:** Reviews the treatment information and can either:
  - Save the analysis report for future reference.
  - Navigate back to the Home Page for further actions.

## **Functional Requirements for Check Disease & Treatment Operation:**

**REQ-1:** The system will allow the user to choose whether to:

- View previously diagnosed diseases and their treatments, retrieved from the user's account stored in the cloud database.
- Analyze a new disease by capturing a photo of the plant.

**REQ-2:** The system will display a list of previously diagnosed diseases, allowing the user to select one to view details and treatment suggestions.

**REQ-3:** Upon selecting a previously diagnosed disease, the system will display the associated treatment information, including a description and confidence level.

**REQ-4:** The user will be able to like or dislike a treatment, limited to one feedback action (like or dislike) per treatment.

- A confirmation message will appear after the action: *"Your feedback has been saved."*

**REQ-5:** The system will store the status of liked or disliked treatments securely in the user's profile within the cloud database.

**REQ-6:** The system will allow the user to take a photo of their plant for disease analysis. The photo must meet the following validation criteria:

- Minimum resolution: 1280x720 pixels.
- Sufficient lighting and clarity (not blurry).

If the photo fails validation, the system will prompt the user to retake the photo.

**REQ-7:** The system will ask the user whether they want to use the captured photo with the following prompt:

*"Do you want to use this photo?"*

- If the user selects "No," they can retake the photo.
- If the user selects "Yes," the system will save the photo temporarily for analysis.

**REQ-8:** The system will securely save the validated photo for processing.

**REQ-9:** The system will display the saved photo to the user for confirmation before initiating analysis.

**REQ-10:** The system will display a progress bar labeled with its purpose:

- *"Uploading Image..."* during photo upload.
- *"Analyzing Image..."* during disease detection processing.

**REQ-11:** The system will handle errors gracefully and provide appropriate feedback:

- *“Image upload failed. Please try again.”* for upload errors.
- *“Unable to retrieve treatment information. Check your connection.”* for network-related issues.

**REQ-12:** The system will allow the user to navigate back to previous steps, retaking a photo or returning to the Home Page, without losing progress.

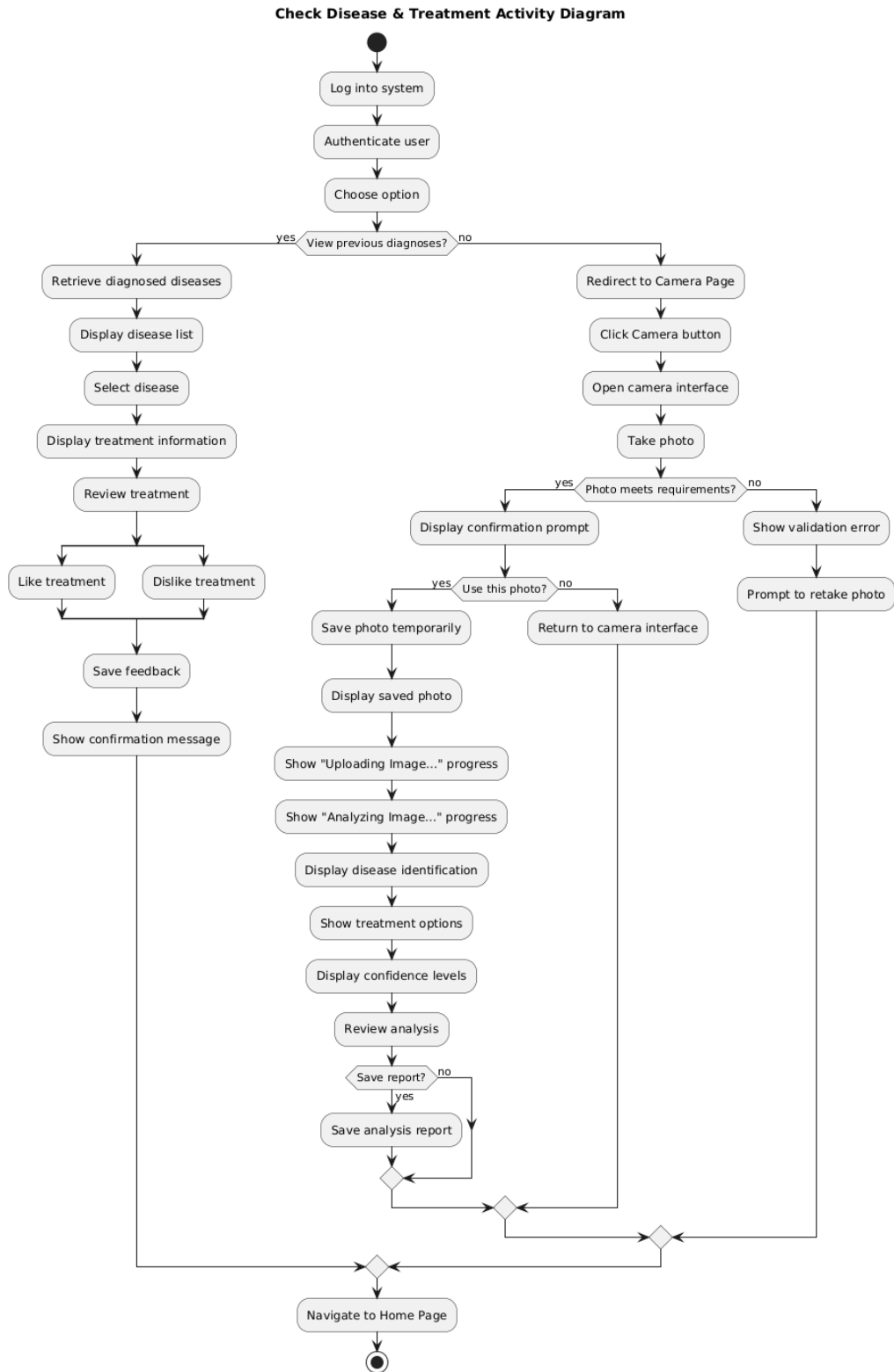


Figure 13 Activity Diagram for Check Disease Treatment



### 3.2.2. Add Post to the Community

#### Stimulus/Response Sequences:

1. **User:** Logs into the system.
2. **System:** Authenticates the user and grants access to the application.
3. **User:** Navigates to the community page.
4. **System:** Displays the community page with existing posts and an “**Add Post**” button.
5. **User:** Clicks on the “**Add Post**” button to create a new post.
6. **System:** Redirects the user to the post creation page.
7. **User:** Enters the following details:
  - **Title** (mandatory): Minimum of 3 characters, maximum of 100 characters.
  - **Description** (mandatory): Minimum of 10 characters, maximum of 5000 characters.
  - **Image Upload** (optional): Supports JPEG, PNG, and GIF formats with a maximum file size of 5 MB.
8. **System:** Verifies the entered information based on the following validation criteria:
  - Title and description must meet the specified character limits.
  - Uploaded images must adhere to format and size restrictions.
9. **System:**
  - If all information is valid:
    - Saves the post securely in the database.
    - Displays a success message:  
*“Your post has been successfully published!”*
    - Redirects the user to a **Post Preview Page**, allowing the user to review the post content and confirm publication.
  - If validation fails:
    - Displays error messages below the relevant fields:
      - *“Title must be at least 3 characters.”*
      - *“Uploaded image exceeds the size limit of 5 MB.”*
10. **User:**
  - If the post is valid, confirms publication on the **Post Preview Page** or chooses to edit the post.
  - If the post is invalid, corrects errors based on the feedback and resubmits.
11. **System:**
  - If saving the post fails due to server or network issues, displays an error message:  
*“Unable to save your post. Please check your connection and try again.”*
  - Provides an option to retry saving the post or save it as a draft for later submission.
12. **User:**
  - After successfully submitting the post, navigates back to the community page to view the newly published post.

## Functional Requirements for Add Post to the Community Operation:

**REQ-1:** The system will allow the user to navigate to the community page via the main navigation menu or Home Page.

**REQ-2:**

The system will provide an “**Add Post**” button on the community page that redirects users to the post creation interface.

**REQ-3:** The system will display a post creation form with the following fields:

- **Title** (mandatory): Minimum of 3 characters, maximum of 100 characters.
- **Description** (mandatory): Minimum of 10 characters, maximum of 5000 characters.
- **Image Upload** (optional): Supports one image per post in JPEG, PNG, or GIF formats, with a maximum file size of 5 MB.

**REQ-4:** The system will validate the submitted data as follows:

- The title and description must meet the specified character limits.
- Uploaded images must adhere to file format and size restrictions.
- The form must not contain empty mandatory fields.

If the input is invalid, the system will:

- Highlight the problematic fields.
- Display a specific error message:
  - *“Title must be at least 3 characters.”*
  - *“Uploaded image exceeds the size limit of 5 MB.”*

**REQ-5:** The system will allow the user to save their post in the following ways:

- **Publish Post:** If all input fields are valid, the post is saved to a cloud-based database (Google Cloud) and marked as published.
- **Save as Draft:** Users can save incomplete or unpublished posts as drafts to edit and publish later.

**REQ-6:** After successfully saving or publishing a post, the system will display a confirmation message:

- *“Your post has been successfully published!”*
- This message will appear as a popup or inline notification on the post creation page.

**REQ-7:** The system will redirect users to the community page after successful publication, displaying the new post at the top of the list.

**REQ-8:** If saving or publishing the post fails due to a system or network error, the system will:

- Display an error message:
  - *“Unable to save your post. Please check your connection and try again.”*
- Provide an option to retry saving the post or save it as a draft for later submission.

**REQ-9:** The system will provide a **Preview Post** option, allowing users to review their post before final submission to ensure accuracy and completeness.

**REQ-10:** The system will include basic content moderation mechanisms:

- Filtering offensive language using a predefined list of banned words.
- Detecting spam or excessive use of special characters in the title or description.

**REQ-11:** All posts, including drafts and published content, will be stored securely in a cloud-based database, ensuring scalability and accessibility.

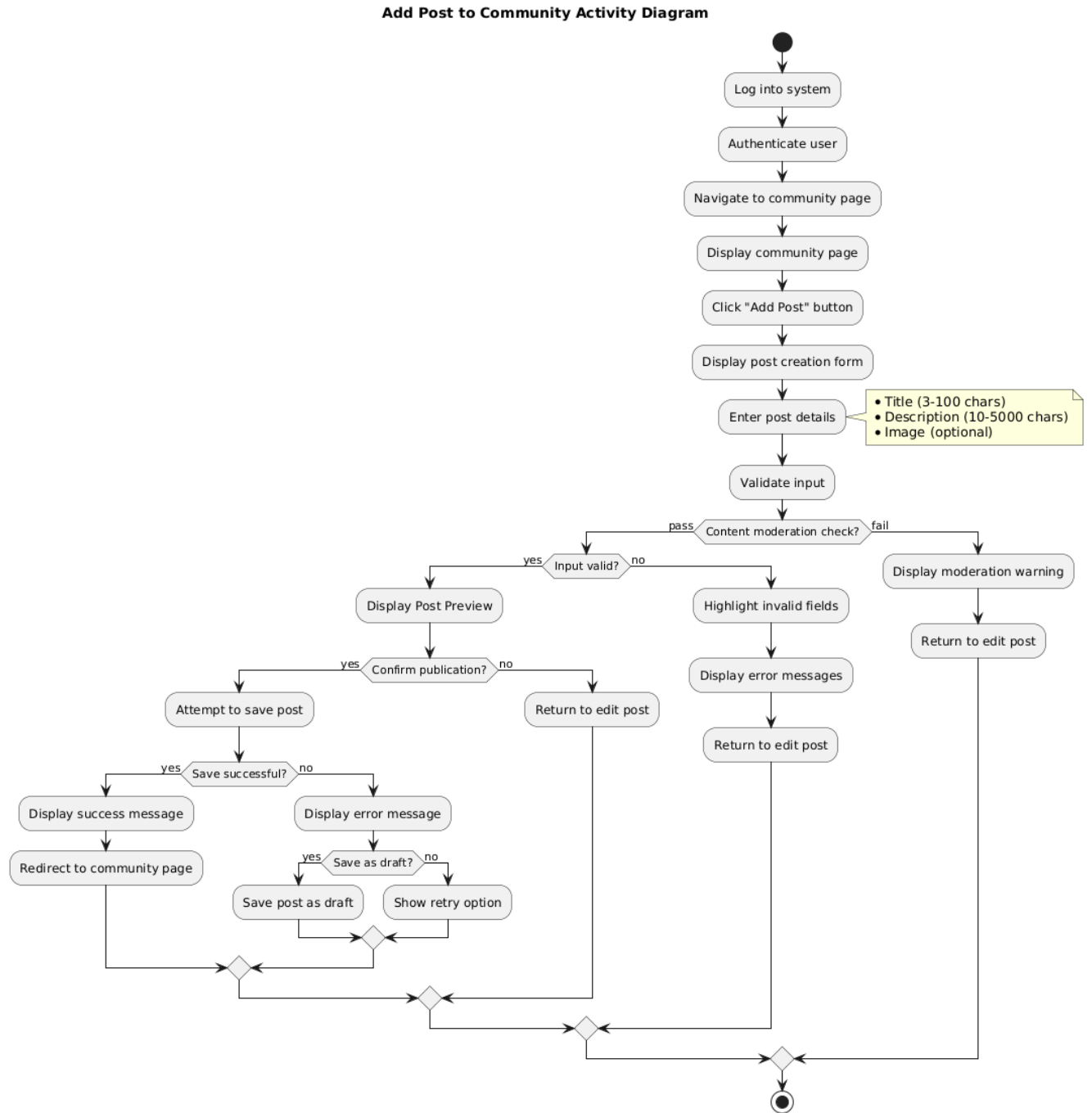


Figure 14 Activity Diagram for Add a Post to the Community

### 3.2.3. Sign Up

#### Stimulus/Response Sequences:

1. **User:** Opens the application.
2. **System:** Displays the **Sign-Up Page** with input fields for account creation.
3. **User:** Enters the following details:
  - **Name** (mandatory): Accepts up to 50 characters.
  - **Surname** (mandatory): Accepts up to 50 characters.
  - **Username** (mandatory): Must be unique, 5–20 alphanumeric characters.
  - **Password** (mandatory): Must meet strength criteria:
    - At least 8 characters.
    - Includes one uppercase letter, one number, and one special character.
  - **City** (optional): Dropdown selection for predefined cities.
  - **Occupation** (optional): Free text with a maximum of 100 characters.
  - **Gender** (optional): Dropdown selection (Male, Female, Non-binary, Prefer not to say).
4. **System:** Validates the entered information based on the following criteria:
  - Mandatory fields (Name, Surname, Username, Password) must not be empty.
  - Username must be unique and not already in use.
  - Password must meet strength criteria.
  - Optional fields, if filled, must conform to character limits and predefined formats.
5. **System:**
  - If all inputs are valid:
    - Creates the user account.
    - Encrypts the password securely using industry-standard encryption (bcrypt).
    - Displays a success message:  
*“Your account has been successfully created! Please verify your email to complete the sign-up process.”*
    - Sends a verification email to the user’s provided email address with a confirmation link.
  - If validation fails:
    - Highlights invalid fields with specific error messages:
      - *“Password must include at least one uppercase letter and one special character.”*
      - *“Username already exists. Please choose a different username.”*
6. **User:**
  - Reviews error messages and corrects invalid inputs, if any.
  - For valid submissions, checks their email and clicks the verification link to activate the account.
7. **System:**
  - After successful email verification, activates the user account and redirects them to the **Login Page** with a confirmation message:  
*“Email verified successfully! You can now log in.”*

8. **System** (Error Handling):

- If the system encounters a network or server issue during validation or account creation:
  - Displays an error message:  
*“Unable to create your account at the moment. Please check your internet connection and try again later.”*
  - Provides an option to retry submission without losing entered data.

**Functional Requirements for Sign Up Operation:**

**REQ-1:** The system will display a **Sign-Up Page** when the user opens the application for the first time or selects the **Sign Up** option from the Login Page.

**REQ-2:** The system will provide input fields for the following user details:

- **Mandatory Fields:**
  - **Name:** Maximum 50 characters.
  - **Surname:** Maximum 50 characters.
  - **Username:** Must be unique, 5–20 alphanumeric characters.
  - **Password:** Must meet strength criteria:
    - Minimum 8 characters.
    - At least one uppercase letter, one number, and one special character.
- **Optional Fields:**
  - **City:** Dropdown selection of predefined options.
  - **Occupation:** Free text, maximum 100 characters.
  - **Gender:** Dropdown options (Male, Female, Non-binary, Prefer not to say).

**REQ-3:** The system will validate the entered information based on the following rules:

- **Username:** Must be unique and not already in use.
- **Password:** Must meet the strength criteria mentioned above.
- **Mandatory Fields:** Must not be empty or exceed the character limits.
- **Optional Fields:** If provided, must adhere to the format and character limits.

**REQ-4:** The system will display actionable error messages if validation fails:

- *“Username already taken. Please choose a different username.”*
- *“Password must include at least one uppercase letter and one special character.”*
- *“Name cannot exceed 50 characters.”*

The system will highlight the invalid fields and allow the user to correct them.

**REQ-5:** The system will create a new user account only if all entered information is valid. The password will be securely hashed and stored using industry-standard encryption (bcrypt).

**REQ-6:** After successful account creation:

- The system will send a verification email with a confirmation link to the user's provided email address.
- The user will see a success message:
  - *"Account created successfully! Please verify your email to complete the registration process."*

**REQ-7:** The system will redirect the user to the **Login Page** after successful email verification and display a confirmation message:

- *"Email verified successfully! You can now log in."*

**REQ-8:** If the system encounters network or server issues during the sign-up process, it will display an error message:

- *"Unable to create your account at the moment. Please check your connection and try again later."*
- The system will provide a retry option and retain previously entered data.

**REQ-9:** The system will implement CAPTCHA or reCAPTCHA to prevent automated bot registrations during the sign-up process.

**REQ-10:** The system will ensure accessibility by providing:

- Proper field labels compatible with screen readers.
- Error messages that are clearly visible and assistive technology-friendly.
- Keyboard navigation for form fields and buttons.

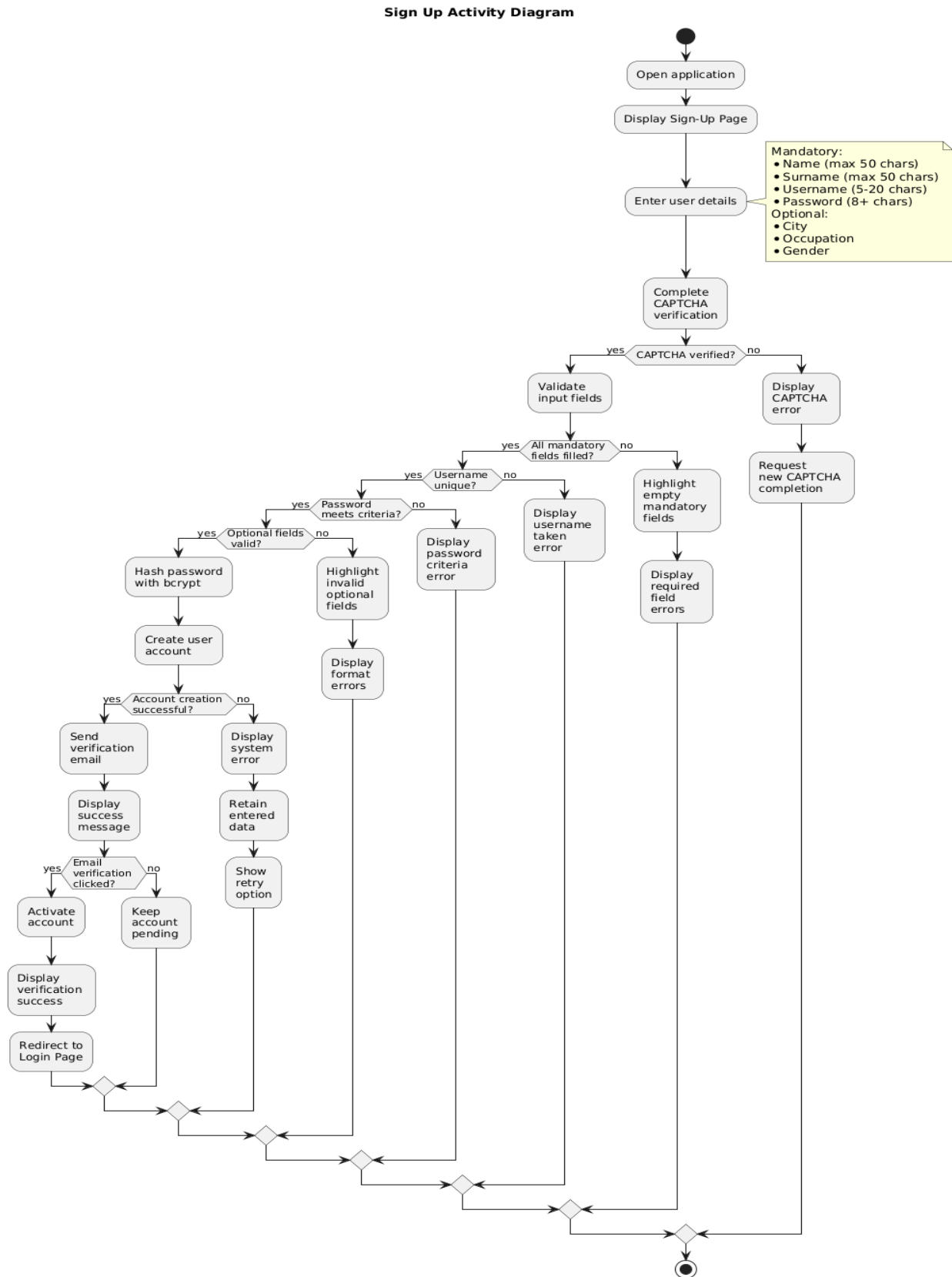


Figure 15 Activity Diagram for Sign Up



### 3.2.4. Login

#### Stimulus/Response Sequences:

1. **User:** Opens the application.
2. **System:** Displays the **Sign Up/Login Page** with buttons for signing up or logging in.
3. **User:** Clicks the “**Login**” button.
4. **System:** Redirects the user to the **Login Page** and prompts the user to enter:
5. **Email:** Must match the email used during registration (case-insensitive).
6. **Password:** Must match the password associated with the email.
7. **User:** Enters their registered email and password.
8. **System:** Validates the submitted credentials as follows:
  - Checks if the email exists in the database.
  - If the email does not exist, displays the error message:  
*“No such user exists. Please check your email or sign up.”*
  - If the email exists but the password is incorrect, displays the error message:  
*“Incorrect password. Please try again.”*
  - Allows the user to retry by re-entering their credentials in the input fields.
9. **System** (After Successful Validation):
  - If the email and password match:
    - Displays a “**Login Successful**” message.
    - Redirects the user to the **Home Page**.
  - During the redirection, shows a loading spinner or visual indicator to confirm the system is processing the request.
10. **System** (Account Security Measures):
  - If the user enters incorrect credentials more than three times consecutively:
    - Temporarily locks the account for 5 minutes.
    - Displays the message:  
*“Too many failed attempts. Please try again later or reset your password.”*
  - Provides an option to complete a CAPTCHA verification to prevent bot activity.
11. **User** (Forgot Password):
  - If the user cannot remember their password, clicks the “**Forgot Password**” link.
  - The system redirects the user to a password recovery page, where they can enter their email to receive a password reset link.
12. **System** (Error Handling):
  - If the system encounters a server or network error during login validation:
    - Displays an error message:  
*“Unable to log in. Please check your internet connection and try again later.”*
    - Provides an option to retry login without re-entering credentials.

## Functional Requirements for Login Operation:

**REQ-1:** The system will display the **Sign Up/Login Page** when the user opens the application. This page will provide options for logging in or signing up.

**REQ-2:** The system will display the **Login Page** when the user clicks the “**Login**” button on the **Sign Up/Login Page**.

**REQ-3:** The system will prompt the user to enter the following login credentials:

- **Email:** Case-insensitive input that matches the email used during registration.
- **Password:** Must match the encrypted password stored in the database.

**REQ-4:** The system will validate the entered credentials based on the following rules:

- **Email Validation:**
  - Must match a registered email in the database.
  - Case-insensitive input is allowed.
- **Password Validation:**
  - Must match the hashed password associated with the provided email.
  - Validation will not expose whether the email exists if the password is incorrect (generic error: *“Invalid credentials. Please try again.”*).

**REQ-5:** The system will display specific error messages for invalid inputs:

- If the email does not exist:
  - *“No account found with this email. Please sign up or check your email address.”*
- If the password is incorrect:
  - *“Invalid credentials. Please try again.”*
- For multiple consecutive incorrect attempts:
  - *“Too many failed attempts. Your account is temporarily locked. Please try again later or reset your password.”*

**REQ-6:** The system will redirect the user to the **Home Page** after successful validation and display a success message:

- *“Login Successful.”*  
During the redirection, a loading spinner or progress indicator will be shown.

**REQ-7:** If validation fails, the system will allow the user to retry by:

- Retaining the email input in the field for convenience.
- Clearing the password field for security.

**REQ-8:** The system will implement the following security measures to prevent brute-force attacks:

- Account lockout after three consecutive failed login attempts for 5 minutes.
- CAPTCHA verification after multiple failed attempts to ensure human interaction.

**REQ-9:** The system will provide a **Forgot Password** link on the **Login Page**. When clicked:

- The user will be redirected to a **Password Recovery Page** to enter their registered email.
- A password reset link will be sent to the user's email with instructions to reset their password.

**REQ-10:** The system will handle server or database failures gracefully by displaying an error message:

- *“Unable to log in at the moment. Please check your internet connection or try again later.”*  
Users will have the option to retry without re-entering their credentials.

**REQ-11:** The system will manage user sessions as follows:

- The session will remain active for a default duration of 30 minutes of inactivity.
- After session expiration, the user will be automatically logged out and redirected to the **Login Page** with a message:
  - *“Your session has expired. Please log in again.”*

**REQ-12:** The system will ensure accessibility by supporting:

- Keyboard navigation for login fields and buttons.
- Screen reader compatibility for form labels, error messages, and buttons.

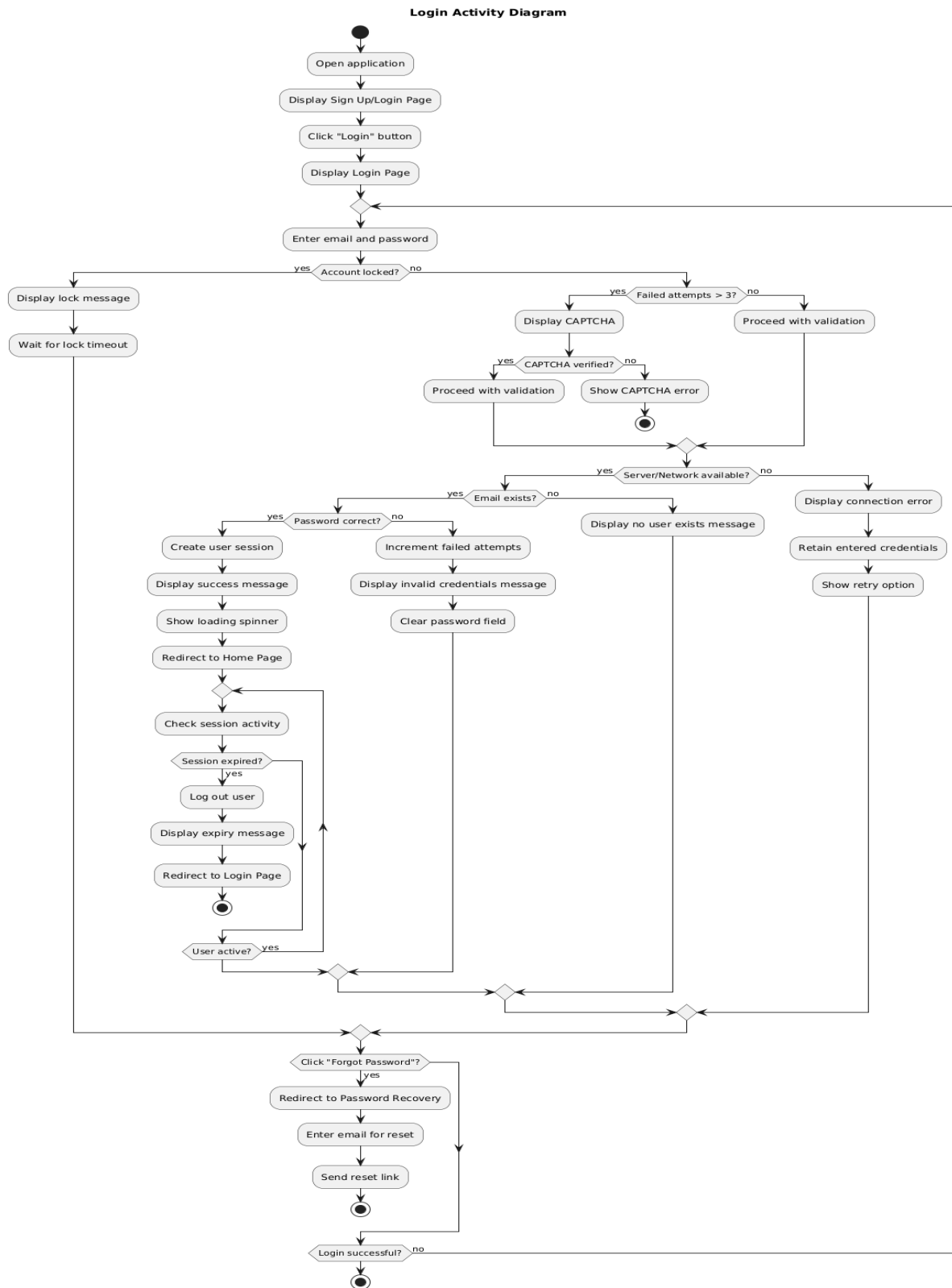


Figure 16 Activity Diagram for Login

### 3.2.5. Edit Profile

#### Stimulus/Response Sequences:

1. **User:** Logs in to the system.
2. **System:** Authenticates the user's credentials.
3. **User:** Navigates to the **Edit Profile Page** via the menu or navigation bar.
4. **System:** Displays the **Edit Profile Page** with the following editable fields:
  - **Name** (text input).
  - **Surname** (text input).
  - **Username** (text input with uniqueness validation).
  - **Password** (current password confirmation required).
  - **City** (dropdown list of predefined values).
  - **Occupation** (text input or dropdown list).
  - **Gender** (radio buttons or dropdown list).
5. **User:** Updates one or more fields and clicks the **"Save Changes"** button.
6. **System:** Validates the updated information based on the following criteria:
  - **Username:**
    - Minimum of 3 characters and maximum of 15 characters.
    - Must be unique (checks against existing usernames in the database).
  - **Password:**
    - Must include at least one uppercase letter, one lowercase letter, one number, and one special character.
    - Minimum length of 8 characters.
    - Requires confirmation of the current password before updating.
  - **Other Fields:**
    - **Name/Surname:** Text input only; no special characters or numbers.
    - **City:** Selected from a predefined list.
    - **Occupation/Gender:** Text input or selected from predefined options.
7. **System** (Validation Outcome):
  - **If Valid:**
    - Saves the updated information securely.
    - Displays a **success message** ( "Your profile has been updated successfully") as an inline notification or popup.
    - Provides an option to return to the **Home Page** or remain on the **Edit Profile Page**.
  - **If Invalid:**
    - Displays specific error messages next to the respective fields ( "Username already taken," "Invalid city selection").
    - Allows the user to correct the errors without clearing valid inputs.
8. **System** (Confirmation and Security):
  - Prompts the user with a confirmation dialog before saving changes:
    - *"Are you sure you want to save these changes?"*
  - Handles sensitive information securely:
    - Encrypts or hashes passwords before saving to the database.

9. **System** (Error Handling for Failures):

- If the system encounters a server or network issue during the update process:
  - Displays a failure message ( “Unable to save changes. Please try again later.”).
  - Provides an option to retry or cancel the operation.

10. **User:** After completing the updates:

- Can choose to return to the **Home Page** via a redirect or manually using the navigation bar.

**Functional Requirements for Edit Profile Operation:**

**REQ-1:** The system will display the **Edit Profile Page** when the user navigates to it from the navigation bar or menu.

**REQ-2:** Users will be able to update the following fields on the **Edit Profile Page**:

- **Name:** Mandatory; accepts alphabetic characters only.
- **Surname:** Mandatory; accepts alphabetic characters only.
- **Username:** Mandatory; must be unique and between 3–15 characters.
- **Password:** Optional; requires the user to enter their current password for verification before creating a new one. New password must include:
  - At least one uppercase letter.
  - At least one lowercase letter.
  - At least one number.
  - At least one special character.
  - Minimum length of 8 characters.
- **City:** Optional; selected from a predefined dropdown menu.
- **Occupation:** Optional; entered as free text or selected from predefined options.
- **Gender:** Optional; selected from a predefined list or entered as free text.

**REQ-3:** The system will validate all updated information based on the following criteria:

- **Username:** Must be unique and conform to allowed character formats.
- **Password:** Must meet the defined complexity and strength requirements.
- **City/Occupation/Gender:** Input must conform to predefined formats or be selected from predefined lists where applicable.

**REQ-4:** The system will display a **confirmation dialog** before saving any changes to ensure users have reviewed their updates. Example confirmation:

*"Are you sure you want to save these changes?"*

**REQ-5:** If all updated information is valid, the system will save the changes and display a **success message**:

- The success message will appear as a popup notification or inline message at the top of the page.
- Example: *"Your profile has been updated successfully."*

**REQ-6:** If any field fails validation, the system will display **field-specific error messages** next to the affected fields. Examples include:

- *"Username is already taken."*
- *"Password must include at least one special character."*
- *"Invalid city selection."*

**REQ-7:** The system will securely handle sensitive data:

- Passwords will be hashed and encrypted before storage.
- All data will be transmitted using secure protocols (HTTPS).

**REQ-8:** The system will handle network or server failures gracefully:

- If a save operation fails due to connectivity or server issues, the system will display an error message (*"Unable to save changes. Please try again later."*).
- Users will have the option to retry or cancel the operation.

**REQ-9:** Users will have flexibility in navigation after saving changes:

- The system will provide options to either remain on the **Edit Profile Page** or return to the **Home Page** after successful updates.

**REQ-10:** The system will ensure accessibility compliance by supporting:

- Screen reader compatibility for all fields and error messages.
- Keyboard navigation for form inputs and buttons.

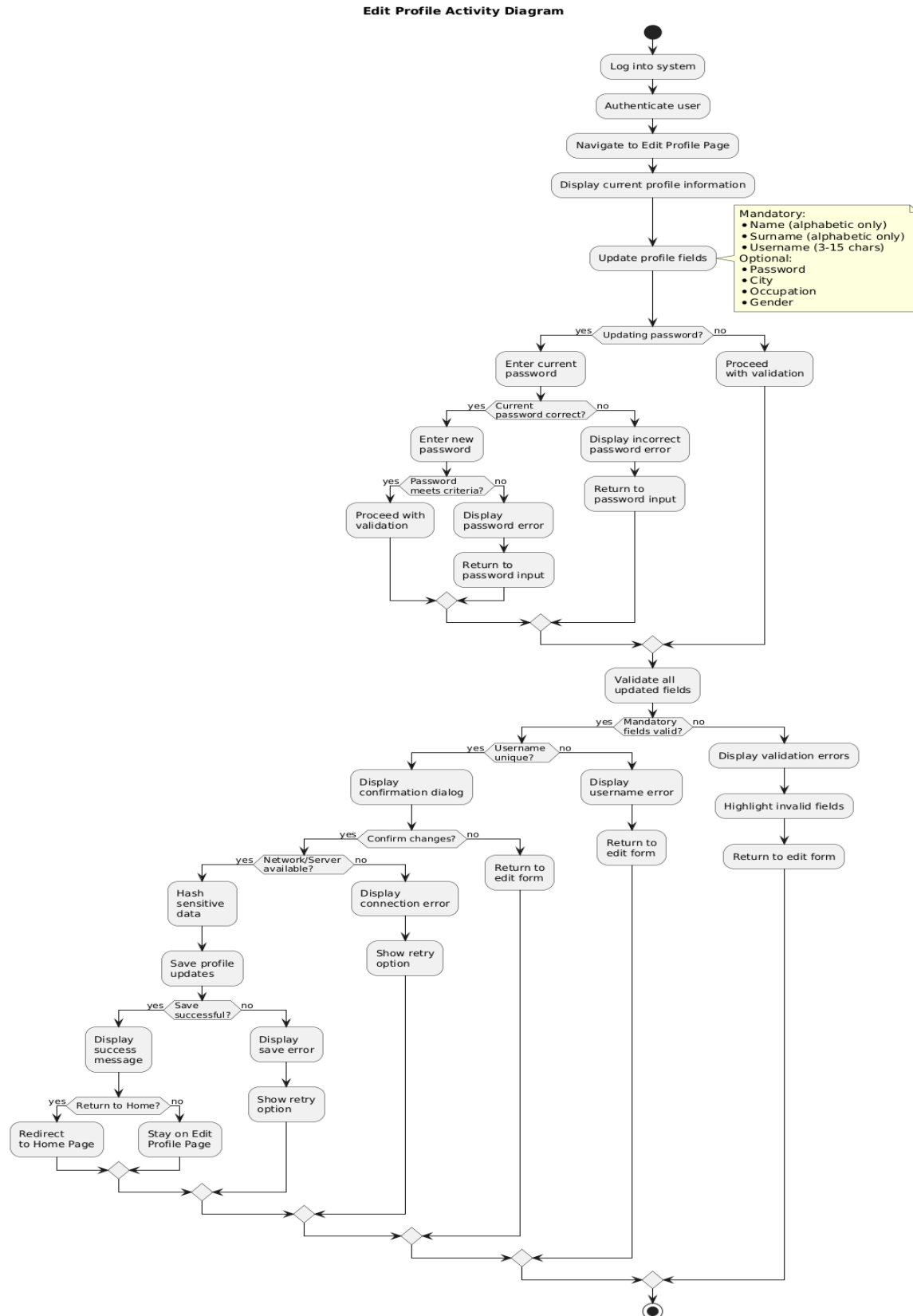


Figure 17 Activity Diagram for Edit Profile



## 3.2.6. Analyze the Map

### Stimulus/Response Sequences:

1. **User:** Login to the system.  
**System:** Authenticate the user.
2. **User:** Navigate to the map analysis page.  
**System:** Display the map interface with all interactive features, zoom controls, filters, and a legend.
3. **User:** Zoom in/out or pan across the map to explore specific regions.  
**System:** Adjust the map view and display disease markers and region-specific summaries dynamically.
4. **User:** Apply filters to customize the map view (by disease type, severity, or date range).  
**System:** Update the map display based on the selected filter criteria.
5. **User:** Search for a specific region or location using the search bar.  
**System:** Highlight the searched region and display relevant disease markers and details.
6. **User:** Click on a specific marker to view detailed disease information.  
**System:** Display a pop-up box or sidebar with the following information:
  - Disease name.
  - Severity level.
  - Number of cases in the region.
  - Treatment suggestions.
  - Last updated timestamp.
7. **System:** Notify the user if no disease data is available for the selected region or search criteria (*"No disease data available for this region."*).
8. **User:** Toggle additional map layers (historical data, weather conditions) or reset the map view to the default state.  
**System:** Update the map accordingly to reflect the selected layers or reset to the default view.
9. **System:** Provide accessibility features for map interactions:
  - Keyboard navigation for zoom and region selection.
  - Screen reader support to describe disease markers and details.
10. **User:** Attempt to load the map during poor network conditions or server downtime.  
**System:** Display an error message (*"Unable to load map. Check your internet connection and try again."*).
11. **User:** Navigate back to other pages (home page, community page) using the navigation bar or menu.  
**System:** Redirect the user to the selected page while saving the current state of the map for later use.

## Functional Requirements for Analyze the Map Operation:

- **REQ-1:** The system will display an interactive map when the user navigates to the map analysis page, with real-time updates for disease data and a load time of less than 2 seconds under normal network conditions.
- **REQ-2:** The user will be able to zoom in and out, pan across regions, and select specific areas for detailed analysis on the map.
- **REQ-3:** The user will be able to apply filters to refine map visualization based on:
  - Disease type.
  - Severity level (low, medium, high).
  - Date range.
  - Affected crop types.
- **REQ-4:** The system will display detailed disease information for selected markers, including:
  - Disease name.
  - Severity level.
  - Affected crop types.
  - Number of cases in the region.
  - Date of detection.
  - Treatment suggestions.
- **REQ-5:** The system will visually represent disease markers using color-coding for severity and unique icons for different disease types.
- **REQ-6:** The system will display a fallback message, *"No data available for this region,"* if no disease markers or data are available for the selected region or search criteria.
- **REQ-7:** The user will be able to export or share map data and views as reports or screenshots for professional use or offline analysis.
- **REQ-8:** The system will provide error handling for network or system failures with appropriate messages like *"Unable to load map. Please check your connection."*
- **REQ-9:** The system will include accessibility features:
  - Keyboard navigation for map controls.
  - Screen reader compatibility for marker and filter information.
  - High-contrast mode for better visibility.
- **REQ-10:** The map will allow the user to toggle additional layers, historical data or weather conditions, to enhance disease analysis.

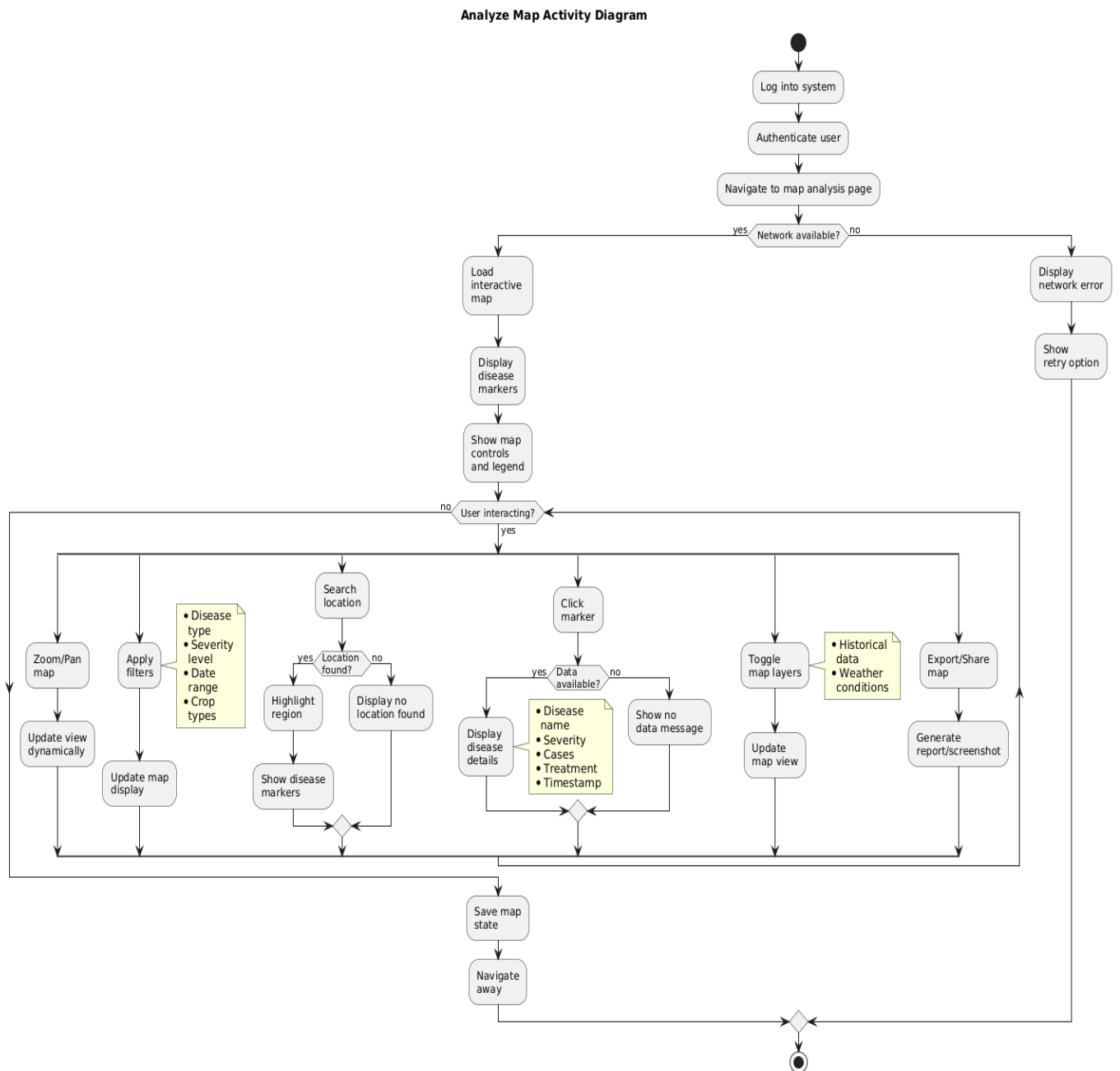


Figure 18 Activity Diagram for Analyze the Map

## 3.3. Non-Functional Requirements

### 3.3.1. Performance

- **Performance Benchmark:** The system's image processing phase, including uploading the image, analyzing it for disease detection, and returning treatment results, should be completed within **15 seconds** under the following conditions:
  - **Hardware Assumptions:** The processing time applies to devices meeting minimum specifications (4 GB RAM, 2 GHz processor) and running on stable networks with a bandwidth of at least 10 Mbps.
  - **Image Resolution:** The system will support images up to a resolution of 1920x1080 pixels, with larger images potentially requiring additional processing time. Users will be notified if the uploaded image exceeds the supported size or resolution limits.
  - **Concurrent Requests:** The 15-second benchmark applies to individual user requests, with the system designed to maintain this performance for up to 100 concurrent users.
- **Error Handling for Delays:** If processing exceeds 15 seconds:
  - The system will display a progress indicator with an estimated remaining time.
  - Users will receive a fallback message, *"Processing is taking longer than expected. Please wait or try again later."*
  - Automatic retries may be triggered for transient errors like network disruptions.
- **Scalability and Monitoring:**
  - The system will log image processing times and monitor performance metrics to ensure adherence to the 15-second target.
  - Alerts will be triggered if processing consistently exceeds this threshold, enabling proactive troubleshooting.

### 3.3.2. Reliability

- **Accuracy Benchmark:** The deep learning model must achieve a minimum accuracy rate of **80%** in identifying plant diseases, measured under the following conditions:
  - **Evaluation Metric:** Accuracy will be assessed using the **F1-score**, which balances precision and recall to provide a reliable measure of the model's performance.
  - **Dataset for Evaluation:** The accuracy benchmark applies to evaluations conducted on a **representative test dataset** that includes diverse plant species, diseases, and image conditions. The dataset will feature variations in:
    - Lighting conditions (bright, dim, and natural light).
    - Image resolutions and angles (high-resolution close-ups and distant shots).
    - Different stages of disease severity.
- **Handling Misclassifications:**
  - If the model misclassifies a plant disease (false positive/negative), the system will:
    - Display a **confidence score** for the prediction to help users make informed decisions.
    - Provide a disclaimer, *"Results are based on machine learning predictions and may not be fully accurate. Consult an expert for confirmation."*
  - Include a feedback mechanism for users to report incorrect predictions, which will contribute to improving the model.
- **Model Retraining and Updates:**
  - The model will be periodically retrained with updated datasets to maintain or improve its accuracy rate as new disease data becomes available.
  - Retraining will include newly reported diseases and edge cases provided through user feedback and external sources.
- **Contingency for Accuracy Below 80%:**
  - If the accuracy rate falls below 80% during performance validation:
    - The system will notify administrators to review and retrain the model.
    - Users will be provided with additional analysis options or recommendations, consulting a local agricultural expert.
- **Performance Validation:**
  - Accuracy will be validated using test conditions that mimic real-world scenarios, including:
    - Low-quality images (blurry or poorly lit).
    - Cases where multiple diseases appear in the same plant.
    - Partial plant visibility in the image.
  - Validation results will be documented and regularly reviewed to ensure reliability.

### 3.3.3. Availability

#### Defined Uptime Requirement:

The system must maintain an **availability rate of at least 99.9%**, adhering to industry standards. This allows for minimal planned or unplanned downtime, equivalent to approximately **8.76 hours of downtime per year**.

- **Availability Scope:**

- **Core Components:** The availability requirement applies to all core functionalities, including:
  - Login and user authentication.
  - Image upload and disease detection.
  - Map analysis and community page access.
- **Geographic Scope:** Availability will be ensured globally, with optimizations for time zone variances to minimize disruptions for users in different regions.

- **Redundancy and Failover Mechanisms:**

- The system will implement redundancy measures:
  - **Backup servers** to handle failover during primary server outages.
  - **Data replication** to ensure continuity and prevent data loss in case of system failures.
- Load balancing techniques will be utilized to distribute traffic evenly and reduce downtime risks.

- **Scheduled Maintenance:**

- The system will include defined maintenance windows, scheduled during **off-peak hours** based on user activity analytics.
- Users will be notified of upcoming maintenance at least **24 hours in advance**, via:
  - In-app notifications.
  - Email alerts (if email is provided).
  - A maintenance schedule displayed on the system's status page.

- **Downtime Contingency and Error Handling:**

- In the event of downtime (planned or unplanned), users will receive clear and user-friendly feedback:
  - A message displayed in the app: *"The system is currently unavailable. Please try again later."*
  - Estimated time to resolution when applicable.
- Users will have access to a **status page** that provides real-time updates on system availability.

- **Monitoring and Reporting:**

- Availability will be monitored continuously through automated tools that log and report uptime metrics.
- Performance reports will be reviewed regularly to ensure compliance with uptime targets and identify potential improvements.

- SLA (Service Level Agreement) compliance will be evaluated quarterly and shared with stakeholders.

### 3.3.4. Security

- **User Authentication:**

The system will implement secure user authentication processes to prevent unauthorized access, including:

- **Username and Password Authentication:** Users must provide a unique username and a secure password to log in.
- **Multi-Factor Authentication (MFA):** Additional authentication steps, OTP via email or mobile, will be used for enhanced security.
- **Password Security:**
  - Passwords will be hashed using a secure algorithm (bcrypt) before storage in the database.
  - All communications involving password transmission will be encrypted using **HTTPS/TLS** protocols to prevent interception.
- **Brute-Force Attack Prevention:**
  - Account lockout will be enforced after five consecutive failed login attempts, with a cooldown period of 30 minutes.
  - CAPTCHA verification will be required after multiple failed login attempts to prevent automated attacks.
- **Session Management:**
  - User sessions will automatically time out after **30 minutes** of inactivity.
  - Secure session tokens (JWT) will be issued and managed, ensuring they are securely signed and encrypted.
- **Data Security During Authentication:**
  - All data exchanged during authentication will be protected using secure communication protocols **HTTPS/TLS**.
  - Sensitive information (passwords, session tokens) will not be logged or stored in plaintext.
- **Role-Based Access Control (RBAC):**
  - The system will implement RBAC to restrict access to specific features or data based on user roles (admin, standard user).

- Permissions for sensitive actions, modifying system settings, will be granted only to authorized roles.
  
- **Audit and Logging:**
  - The system will maintain logs of all authentication attempts, including:
    - Successful and failed login attempts.
    - Timestamped records of suspicious activities (repeated failed logins from the same IP).
  - Logs will be regularly reviewed for security monitoring and incident response.
  
- **Password Recovery:**
  - A secure password recovery mechanism will be provided, requiring email-based verification for resetting passwords.
  - Recovery tokens will be time-limited (valid for 15 minutes) and encrypted to prevent misuse.
  
- **Compliance with Security Standards:**
  - The system will adhere to security best practices, including:
    - **OWASP** guidelines for secure authentication implementation.
    - **GDPR** and other applicable regulations for user data protection.
  - Regular security audits will be conducted to ensure ongoing compliance with these standards.



### 3.3.5. Maintainability

- **Modular Design:**

The system should be modular to allow for easy updates, enhancements, and maintenance. Modularity will be achieved through a clearly defined architecture that separates the front-end, back-end, and database components. Each module will be independently testable and deployable.

- **Impact of Updates:**

- Updates will ensure backward compatibility to avoid breaking existing functionalities.
- Minimal downtime will be prioritized during updates using techniques rolling updates or blue-green deployments.

- **Scalability:**

The modular design will support scalability, allowing new features or modules to be added seamlessly without affecting existing functionalities.

- **Code Documentation:**

- All modules will be accompanied by comprehensive and up-to-date documentation, including:
  - Code comments for clarity.
  - API documentation for integration details.
  - High-level architecture diagrams for module relationships.
- Documentation will be stored in a centralized, version-controlled repository for easy access.

- **Third-Party Integrations:**

The modular structure will support easy integration with third-party tools and APIs. Integration points will be clearly defined with robust interfaces to prevent disruptions during updates or enhancements.

- **Testing Strategy:**

- All updates and enhancements will undergo rigorous testing to ensure system stability, including:
  - **Unit Testing:** For individual module functionality.
  - **Integration Testing:** To validate interactions between modules.
  - **Regression Testing:** To verify that updates do not introduce new issues.
- Automated testing will be implemented where feasible to streamline the update process.

- **Version Control:**

A version control system (Git) will be used to manage code changes effectively. This will include:

- Branching strategies (feature branches, release branches).
- Version tagging for updates and releases.
- Change logs to document updates and enhancements for future reference.

- **Maintainability Metrics:**

Maintainability will be monitored through metrics:

- **Code complexity:** Measured using tools like SonarQube to ensure simplicity and readability.
- **Build success rate:** To track the stability of updates.
- **Time to implement updates:** To ensure enhancements are manageable within the expected timeframe.

### 3.3.6. Portability

- **Platform Compatibility:**

The system must function seamlessly on both Android and iOS platforms, with the following minimum OS version requirements:

- Android: Version 8.0 (Oreo) and above.
- iOS: Version 12 and above.

- **Device Form Factor Support:**

The system will support various device types, including smartphones and tablets. The design will adapt to different screen sizes and resolutions to ensure a consistent user experience across devices.

- **Development Approach:**

Portability will be achieved through the use of a cross-platform framework React Native or Flutter to ensure a unified codebase and efficient resource utilization. Native development will be considered for platform-specific features if required.

- **Performance Consistency:**

The system will maintain consistent performance metrics (response time, loading speed) across Android and iOS devices, ensuring an equitable user experience on both platforms.

- **Platform-Specific Features:**

The system will account for platform-specific behaviors and interactions, including:

- iOS gestures and navigation patterns.
- Android back button functionality and navigation.

- **Testing and Validation:**

Portability will be validated through comprehensive testing strategies, including:

- Device testing on a range of Android and iOS devices with varying OS versions.
- Addressing device-specific issues through targeted debugging and fixes.

- **Offline Capabilities:**

Offline functionality will be supported uniformly across both platforms to ensure users can continue accessing features in low or no connectivity scenarios, where applicable.

- **Update Management:**

Updates will be synchronized across Android and iOS platforms to ensure feature parity.

- Backward compatibility will be maintained for previously supported versions wherever possible.

- Users will receive consistent update notifications and access to updated functionality simultaneously on both platforms.

### **3.3.7 Design Constraints**

There are no specific design constraints in this project. Design will be done respect to the UI designs in “User Interfaces” part of the SRS document.