Smart Health App Project Report

Table of Contents

1. Introduction

1.1 Background

1.2 Objectives

1.3 Scope

1.4 Project Team

2. Requirement Elicitation

2.1 Stakeholder Identification

2.2 Functional Requirements

2.3 Non-Functional Requirements

2.4 Use Cases

3. Analysis

3.1 Use Case Descriptions

3.2 Activity Diagrams

3.3 Class Diagrams

3.4 Sequence Diagrams

4. System Design

4.1 Architectural Overview

4.1.1 Client-Side Components

4.1.2 Server-Side Components

4.2 Database Design

4.2.1 Entity-Relationship Diagram (ERD)

4.3 Detailed Design

4.3.1 User Authentication Module

4.3.1.1 Class: User

4.3.2 Patient Profile Module

4.3.2.1 Class: Patient

4.3.3 Healthcare Professional Profile Module

4.3.3.1 Class: HCP

4.4 Sequence Diagrams

4.4.1 User Registration Sequence Diagram

4.4.2 Patient Appointment Booking Sequence Diagram

5. Implementation

5.1 Development Process

5.2 Technologies Used

5.3 Key Implementation Details

5.3.1 User Authentication Module

5.3.2 Patient Profile Module

5.3.3 Healthcare Professional Profile Module

6. Testing

6.1 Testing Approach

6.2 Test Scenarios

6.2.1 User Authentication Module

6.2.2 Patient Profile Module

6.2.3 Healthcare Professional Profile Module

6.3 Methodologies Used

6.4 Bug Tracking and Reporting

7. Conclusion

7.1 Summary of Achievements

7.2 Future Enhancements

7.3 Acknowledgments

8. Appendices

8.1 Code Snippets

8.2 Additional Diagrams

8.3 Glossary

1. Introduction

The "Smart Health App" is a groundbreaking mobile application that seeks to revolutionise the healthcare industry by leveraging cutting-edge technology to create a user-friendly platform connecting patients and healthcare professionals (HCPs). This introduction provides an overview of the project, outlining its background, objectives, scope, and significance in addressing the challenges faced by the healthcare sector.

1.1. Background

The healthcare landscape is constantly evolving, driven by advancements in technology and the increasing demand for more accessible and convenient medical services. However, traditional healthcare systems often face challenges in meeting these demands, resulting in long waiting times, limited access to medical records, and communication gaps between patients and HCPs.

The rise of digital health solutions has shown promising results in addressing these challenges. Mobile health apps have emerged as a powerful tool to bridge the gap between patients and healthcare providers. By providing a centralized platform for managing appointments, accessing medical records, and promoting health education, these apps have the potential to enhance the overall healthcare experience.

1.2. Project Overview

The Smart Health App is a comprehensive mobile application designed to optimize the healthcare journey for both patients and healthcare professionals. The app's primary goal is to streamline the process of seeking medical assistance, improving patient outcomes, and enhancing the efficiency of healthcare services.

For patients, the app offers a user-friendly interface to manage their healthcare needs. Patients can easily register and create profiles containing their personal information and medical history. The app grants them secure access to their medical records, enabling a deeper understanding of their health status and facilitating informed decision-making.

On the healthcare professionals' side, the Smart Health App provides a repository of medical papers and publications related to their field of expertise. This repository empowers HCPs with the latest research and insights, contributing to evidence-based healthcare practices.

1.3. Objectives

The Smart Health App project aims to achieve several key objectives:

1. Enhanced Patient Experience: The app intends to provide patients with a seamless and user-friendly platform to manage their healthcare journey, including booking appointments, accessing medical records, and receiving health-related information.

2. Improved Healthcare Efficiency: By digitalizing appointment booking and medical record management, the app seeks to enhance the efficiency of healthcare services, reducing waiting times and administrative burdens.

3. Secure Data Management: The project prioritizes the implementation of robust security measures to ensure the confidentiality and integrity of patient data.

4. Professional Knowledge Sharing: The Smart Health App aspires to foster a community of healthcare professionals by providing them with access to a centralized repository of medical papers and publications, encouraging knowledge sharing and collaboration.

1.4. Scope

The scope of the Smart Health App project encompasses the development of a mobile application for both Android and iOS platforms. The initial version of the app will include the following features:

1. User registration and login functionality with secure authentication.

2. Patient profiles containing personal information and medical history.

3. Healthcare professional profiles with credentials and medical specialization.

4. Appointment booking and management system.

5. Access to medical records for patients.

6. A repository of medical papers and publications for healthcare professionals.

7. A gamified system to engage users in health-related activities and knowledge sharing.

1.5. Report Organization

The report is structured into several sections, each focusing on a specific aspect of the Smart Health App project. The sections include Requirement Elicitation, Analysis, System Design, Detailed Design, Implementation, Testing, and Conclusion.

2. Requirement Elicitation

Requirement elicitation is a crucial phase in the development of the Smart Health App. During this stage, the project team identifies and gathers all the functional and non-functional requirements from stakeholders, including end-users, healthcare professionals, and other relevant parties. This section outlines the elicitation process, the gathered requirements, and their prioritization.

2.1. Elicitation Process

The requirement elicitation process involves various techniques to gather and document the project's needs. The project team conducted interviews, surveys, and focus group discussions with potential end-users and healthcare professionals to understand their pain points, expectations, and desired functionalities. Additionally, brainstorming sessions were held within the development team to generate innovative ideas for the app's features.

2.2. Functional Requirements

Based on the elicitation process, the following functional requirements were identified for the Smart Health App:

1. User Registration and Login: The app should allow new users to register by providing their email or phone number, and a secure authentication process should enable users to log in to the app.

2. Patient Profile Management: Patients should be able to create and manage their profiles containing personal information, such as name, age, gender, and medical history.

3. Healthcare Professional Profile Management: HCPs should have the ability to create and maintain their profiles, including their credentials, specialization, and professional achievements.

4. Appointment Booking: Patients should be able to schedule appointments with healthcare professionals through the app.

5. Medical Records Access: Patients should have secure access to their medical records, including test results, diagnosis, and treatment history.

6. Medical Papers Repository: The app should provide a repository of medical papers and publications for healthcare professionals to access and contribute to relevant research.

7. Health Facts and Articles: The app should offer informative articles and health-related facts to educate and engage users, promoting health awareness.

8. Gamification: The app should incorporate gamified elements to encourage user participation in health-related challenges and activities.

2.3. Non-Functional Requirements

The non-functional requirements focus on aspects that are crucial for the app's overall performance and user experience. The following non-functional requirements were elicited:

1. Security: The app should implement robust security measures to protect user data and ensure confidentiality.

2. Usability: The user interface should be intuitive, user-friendly, and accessible to users of all ages and technical backgrounds.

3. Performance: The app should be responsive and provide real-time interactions to ensure a smooth user experience.

4. Scalability: The app should be designed to handle a growing user base and increasing data without compromising performance.

5. Reliability: The app should be reliable, available, and resilient to handle potential server downtimes or network issues.

2.4. Requirement Prioritization

Based on the stakeholders' inputs and the project team's evaluation, the functional and non-functional requirements were prioritised to determine the critical features for the app's initial version. The primary focus was on the core functionalities such as user registration, profile management, appointment booking, and secure data handling. Secondary functionalities, like the medical papers repository and gamification elements, were prioritised for a comprehensive user experience.

3. Analysis

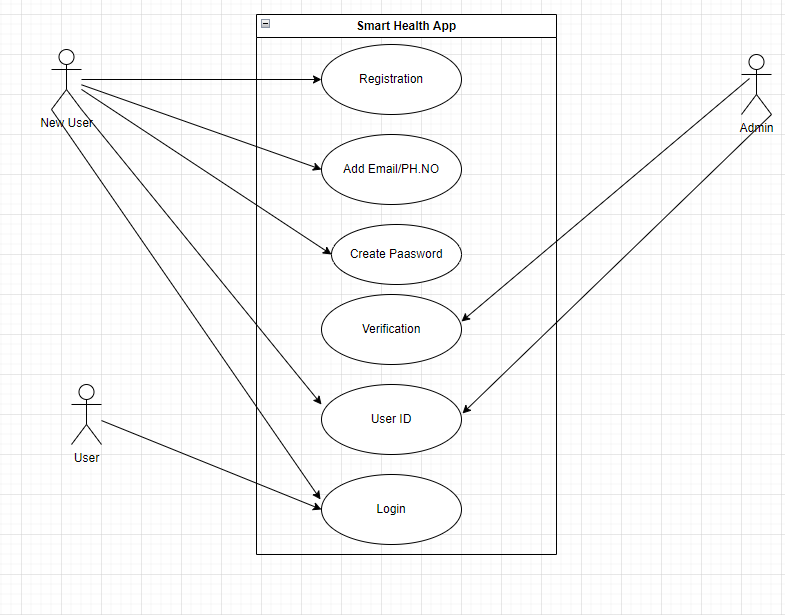
The analysis phase of the Smart Health App project involves a detailed examination of the requirements gathered during the elicitation phase. This section focuses on creating comprehensive models, diagrams, and use case descriptions to outline the system's behavior and interactions between users and the app's functionalities.

3.1. Use Case Diagram

The Use Case Diagram showcases the interactions between different actors (users) and the app's functionalities. It visually represents the high-level user interactions with the app.

(Include a UML Use Case Diagram here, depicting the primary use cases, actors, and their relationships with the app's functionalities.)

3.2. Use Case Descriptions



The Use Case Descriptions provide detailed narratives for each identified use case, describing the sequence of actions and interactions between actors and the app.

Use Case: User Registration

- Description: This use case describes the process of a user registering with the Smart Health App.

- Actors: New User, Smart Health App

- Preconditions: The Smart Health App is installed and accessible to the user.

- Basic Flow:

1. The New User opens the Smart Health App.

2. The App displays the registration page, prompting the user to enter their email or phone number, and create a password.

3. The New User enters the required information and submits the registration form.

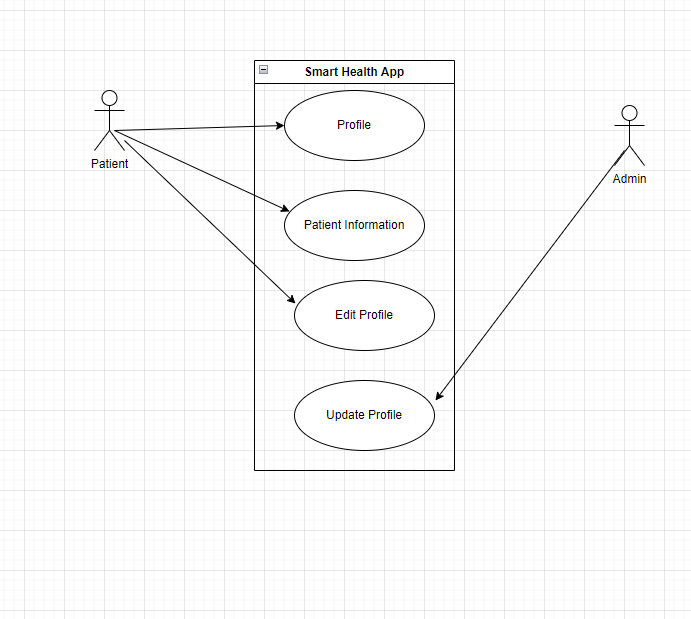
4. The App validates the entered data and checks for duplicate accounts.

5. The App creates a new user account, generating a unique user ID and storing the user's information securely.

6. The App displays a registration success message and redirects the user to the login page.

- Postconditions: The New User has successfully registered with the Smart Health App and can now log in using their credentials.

Use Case: Patient Profile Management



- Description: This use case allows patients to create, view, and manage their profiles within the app.

- Actors: Patient, Smart Health App

- Preconditions: The Patient is logged in to the Smart Health App.

- Basic Flow:

1. The Patient accesses the "Profile" section of the app.

2. The App displays the patient's profile information, including their name, age, gender, and medical history.

3. The Patient can choose to edit their profile information and save the changes.

4. The App validates the updated information and stores it securely.

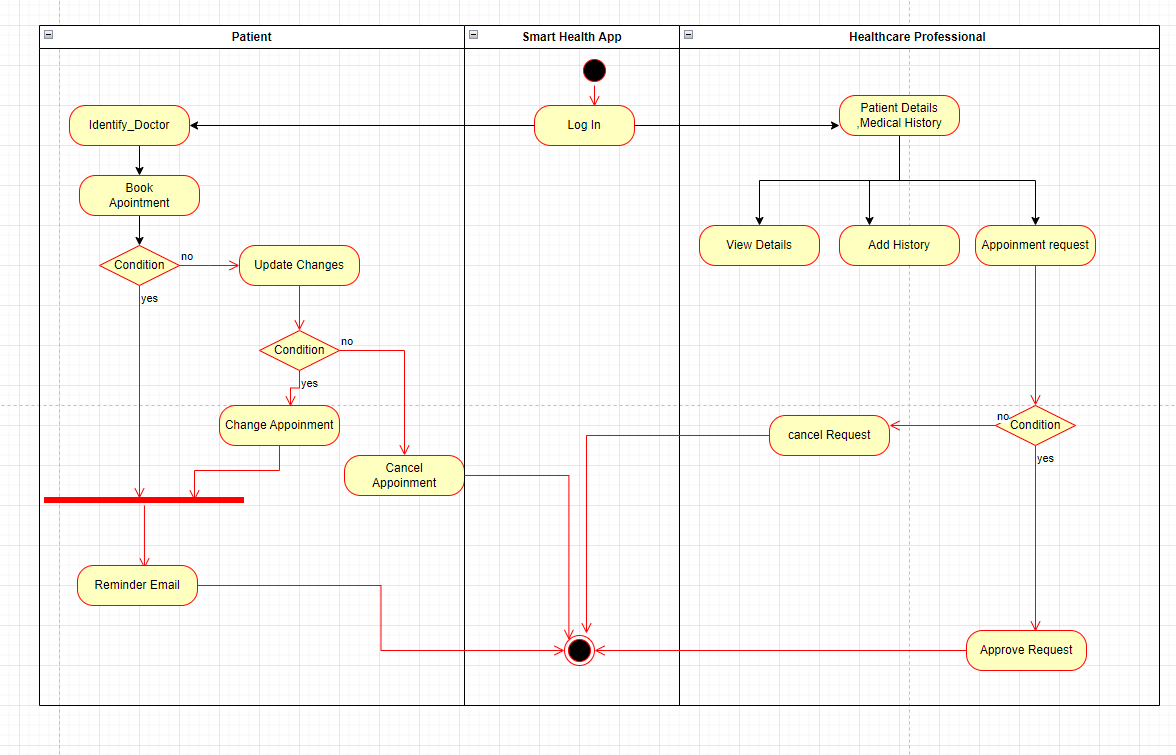
- Postconditions: The Patient's profile information is updated and saved in the app's database.

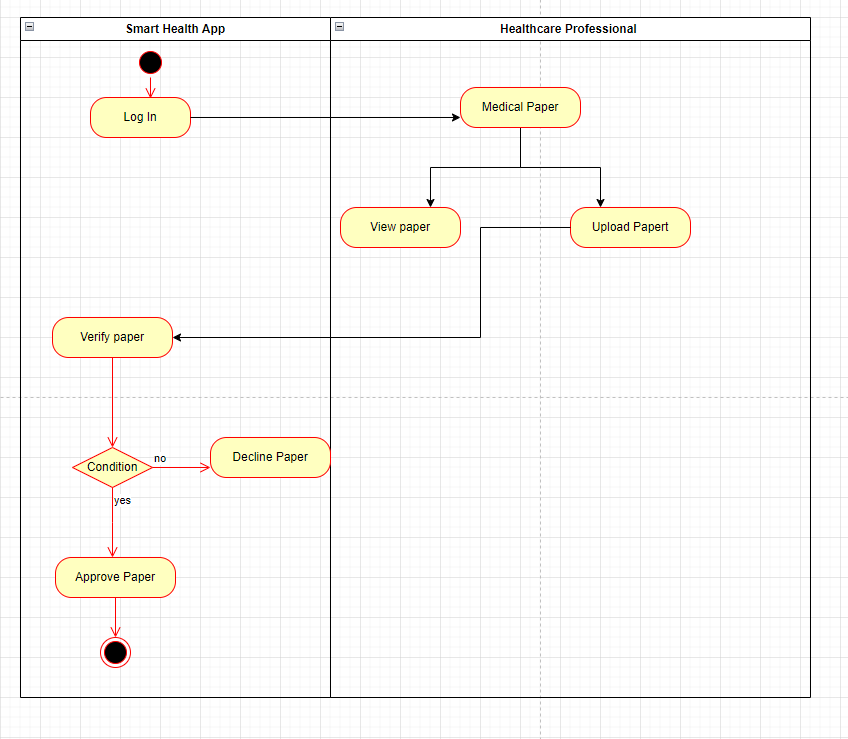
(Continue with detailed descriptions for the rest of the use cases identified in the Use Case Diagram.)

3.3. Activity Diagrams

The Activity Diagrams illustrate the flow of actions within specific use cases, emphasizing the sequence of steps and decision points.

(Include UML Activity Diagrams for key use cases, such as Patient Appointment Booking, Healthcare Professional Medical Paper Upload, etc.)

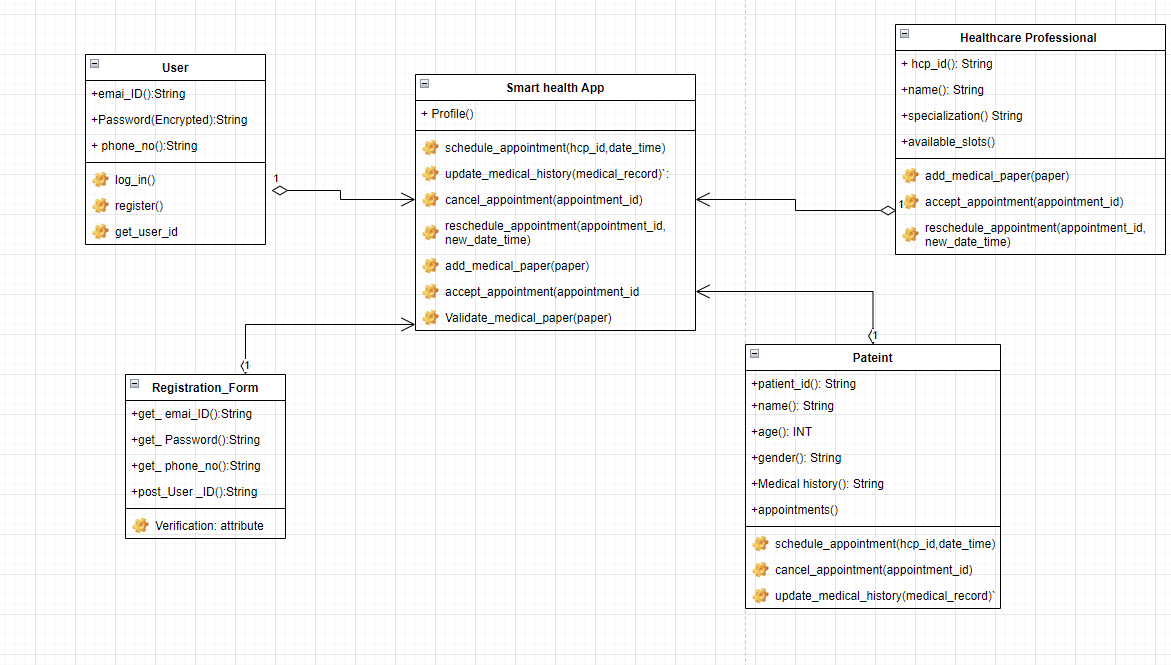




3.4. Class Diagram

The Class Diagram depicts the relationships between the classes and their attributes and methods, providing an overview of the system's structure.

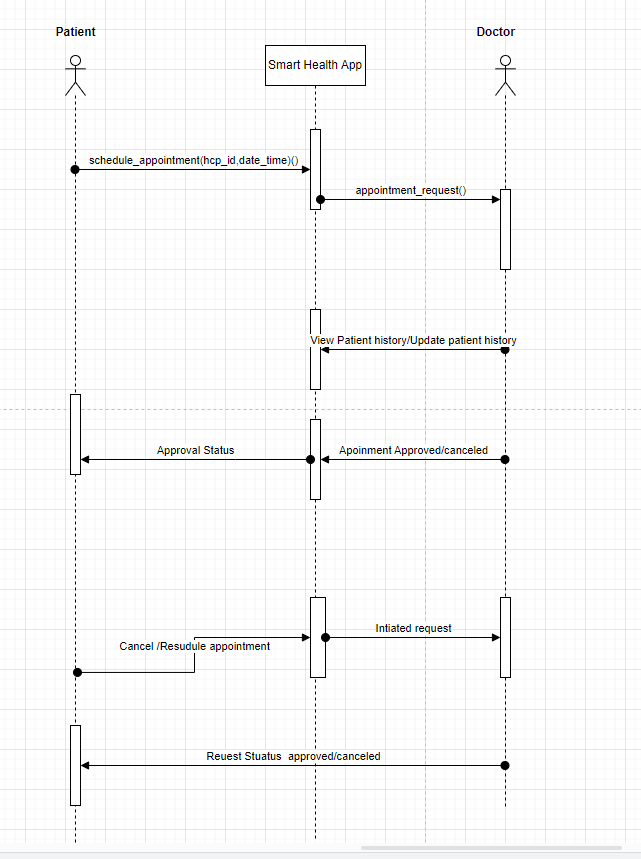
(Include a UML Class Diagram that showcases the classes identified during the requirement elicitation phase, such as User, Patient, Healthcare Professional, Appointment, etc.)

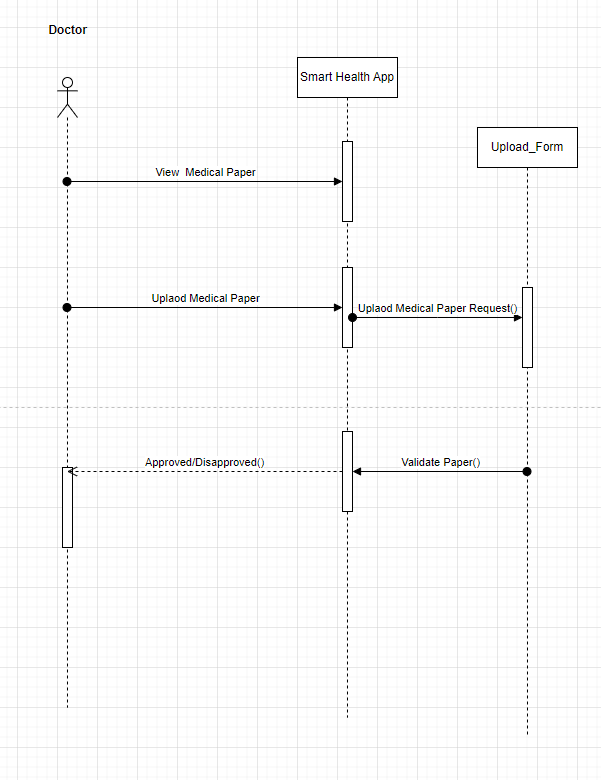


3.5. Sequence Diagrams

The Sequence Diagrams demonstrate the interactions and message exchanges between objects and classes during specific use cases.

(Include UML Sequence Diagrams for important interactions, such as Patient Appointment Booking, Healthcare Professional Medical Paper Upload, etc.)





4. System Design

The System Design phase of the Smart Health App project focuses on creating a high-level architectural overview and detailed design specifications for the app. This section will outline the overall system architecture, database design, and detailed design specifications for key modules and components.

4.1. Architectural Overview

The Smart Health App will follow a client-server architecture, where the app's frontend (client) will interact with a centralized server to handle data storage and processing. The server will manage user authentication, handle database operations, and provide the necessary APIs for the app's functionalities.

Client-Side Components:

1. Mobile App Interface: The client-side component comprises the user interface of the Smart Health App running on both Android and iOS devices. It provides an intuitive platform for users to interact with the app's features.

2. User Authentication: The client-side will handle user authentication, ensuring secure login and registration processes. Users will be authenticated using their credentials (email/phone number and password).

Server-Side Components:

1. Web Server: The web server will handle incoming requests from the app's frontend and interact with the database to fetch or store data. It will also manage user authentication and session management.

2. Database: The database will store all user-related information, medical records, appointments, medical papers, and other relevant data. It will be designed to ensure data integrity and security.

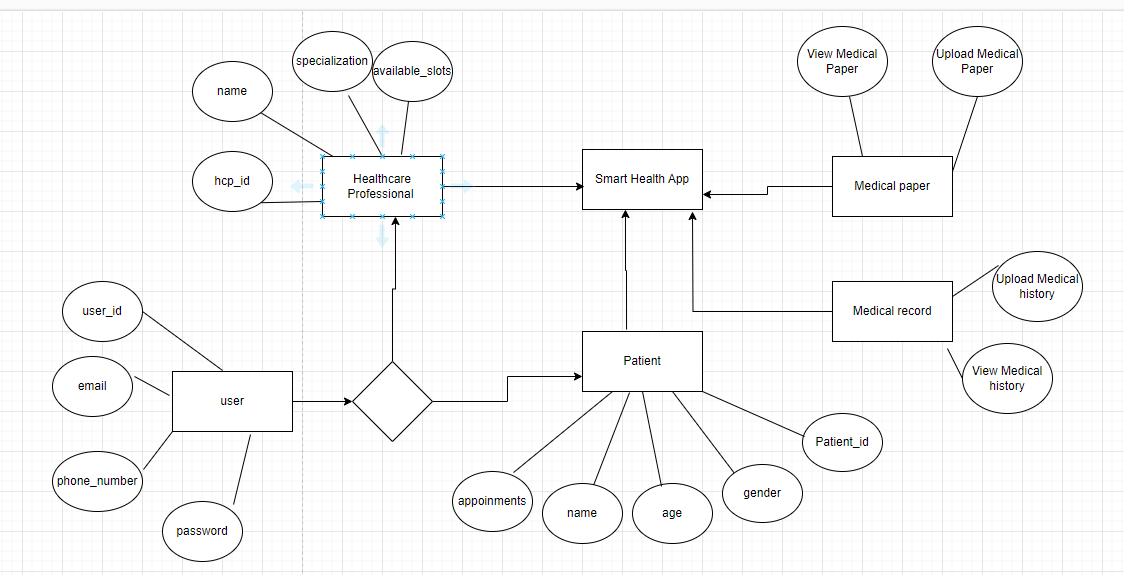
3. API Services: The API services will act as intermediaries between the app's frontend and the database. They will handle data retrieval and storage, ensuring secure communication and data validation.

4.2. Database Design

The database design will focus on creating an efficient and secure data storage system to accommodate user information and medical records.

Entity-Relationship Diagram (ERD):

(Include an ERD showcasing the entities, attributes, and relationships for the key database tables, such as User, Patient, Healthcare Professional, Medical Record, Medical Paper, etc.)



4.3. Detailed Design

4.3.1. User Authentication Module

The User Authentication Module will be responsible for managing user registration, login, and security.

- Class: User

Attributes:

- `user\_id`: Unique identifier for each user.

- `email`: Email address associated with the user's account.

- `phone\_number`: Phone number associated with the user's account.

- `password`: Encrypted password for account security.

Methods:

- `register(email, phone\_number, password)`: Registers a new user with the provided email, phone number, and password.

- `login(email/phone\_number, password)`: Authenticates the user by checking the provided credentials.

- `get\_user\_id()`: Retrieves the unique identifier of the user.

4.3.2. Patient Profile Module

The Patient Profile Module will allow patients to manage their profiles, including personal information and medical history.

- Class: Patient

Attributes:

- `patient\_id`: Unique identifier for each patient.

- `name`: Patient's full name.

- `age`: Patient's age.

- `gender`: Patient's gender.

- `medical\_history`: List of medical records and history for the patient.

- `appointments`: List of scheduled appointments for the patient.

Methods:

- `update\_medical\_history(medical\_record)`: Adds a new medical record to the patient's medical history.

- `schedule\_appointment(hcp\_id, date\_time)`: Sends a booking request to an HCP for a specific date and time.

- `cancel\_appointment(appointment\_id)`: Cancels a previously scheduled appointment.

4.3.3. Healthcare Professional Profile Module

The Healthcare Professional Profile Module will enable HCPs to manage their profiles, including credentials and medical specialization.

- Class: HCP (Healthcare Professional)

Attributes:

- `hcp\_id`: Unique identifier for each HCP.

- `name`: HCP's full name.

- `specialization`: HCP's medical specialization.

- `available\_slots`: List of available time slots for appointments.

Methods:

- `add\_medical\_paper(paper)`: Adds a new medical paper or publication to the HCP's repository.

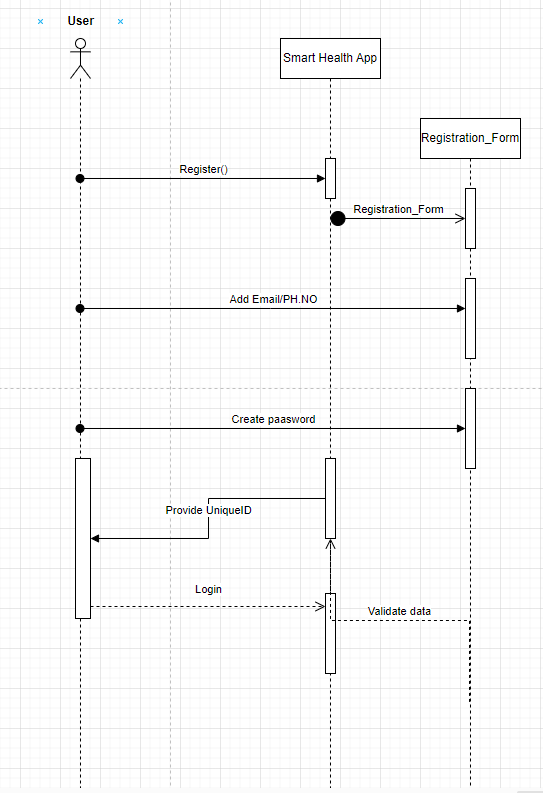
- `accept\_appointment(appointment\_id)`: Accepts a booking request for a specific appointment.

- `reschedule\_appointment(appointment\_id, new\_date\_time)`: Reschedules an existing appointment.

4.4. Sequence Diagrams

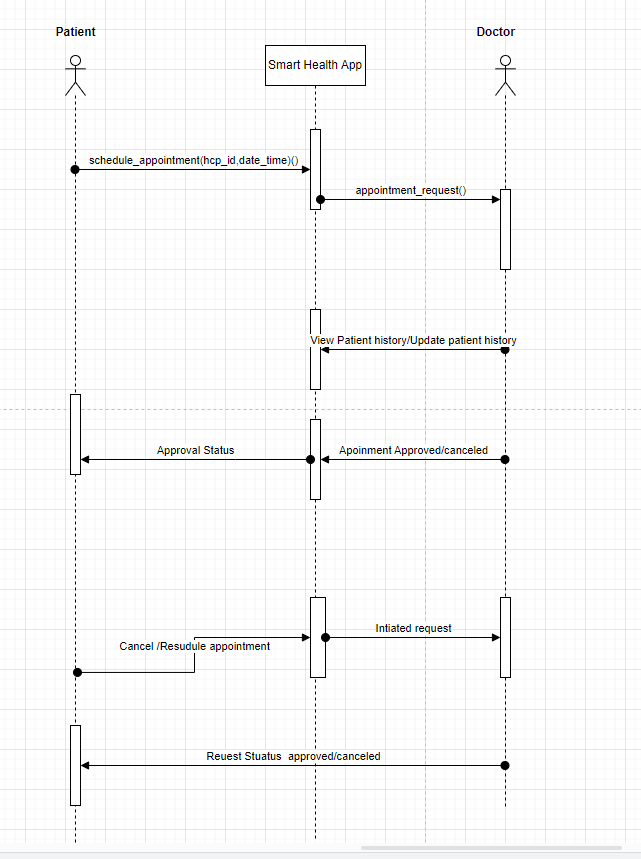
4.4.1. User Registration Sequence Diagram

(Include a UML Sequence Diagram illustrating the sequence of interactions during the user registration process.)



4.4.2. Patient Appointment Booking Sequence Diagram

(Include a UML Sequence Diagram illustrating the sequence of interactions during the patient's appointment booking process.)



5. Detailed Design

The Detailed Design phase of the Smart Health App project delves deeper into the architecture and design of key modules and components. This section will provide in-depth explanations, class diagrams, sequence diagrams, and other relevant details for essential functionalities.

5.1. User Authentication Module

The User Authentication Module is responsible for managing user registration, login, and security. It ensures that user credentials are securely stored and verified during the login process.

Class: User

Attributes:

- `user\_id`: Unique identifier for each user.

- `email`: Email address associated with the user's account.

- `phone\_number`: Phone number associated with the user's account.

- `password`: Encrypted password for account security.

Methods:

1. `register(email, phone\_number, password)`: This method handles the user registration process. When called, it creates a new User object with the provided email, phone number, and password. The password is hashed and securely stored to protect user data.

2. `login(email/phone\_number, password)`: This method handles user login. When called, it takes the user's email or phone number and password as inputs and compares the provided password with the stored hashed password for authentication. If the credentials match, the user is granted access to the app.

3. `get\_user\_id()`: This method retrieves the unique identifier (user\_id) associated with the user's account. The user\_id is used as a reference for various operations and associations within the app.

5.2. Patient Profile Module

The Patient Profile Module allows patients to manage their profiles, including personal information and medical history.

Class: Patient

Attributes:

- `patient\_id`: Unique identifier for each patient.

- `name`: Patient's full name.

- `age`: Patient's age.

- `gender`: Patient's gender.

- `medical\_history`: List of medical records and history for the patient.

- `appointments`: List of scheduled appointments for the patient.

Methods:

1. `update\_medical\_history(medical\_record)`: This method allows the patient to add a new medical record to their medical history. When called, it appends the provided medical record to the existing medical\_history list.

2. `schedule\_appointment(hcp\_id, date\_time)`: This method enables the patient to schedule an appointment with a specific healthcare professional (HCP) for a given date and time. It creates a new Appointment object and adds it to the patient's appointments list.

3. `cancel\_appointment(appointment\_id)`: This method allows the patient to cancel a previously scheduled appointment. When called, it searches for the appointment with the specified appointment\_id in the appointments list and removes it.

5.3. Healthcare Professional Profile Module

The Healthcare Professional Profile Module enables HCPs to manage their profiles, including credentials and medical specialization.

Class: HCP (Healthcare Professional)

Attributes:

- `hcp\_id`: Unique identifier for each HCP.

- `name`: HCP's full name.

- `specialization`: HCP's medical specialization.

- `available\_slots`: List of available time slots for appointments.

Methods:

1. `add\_medical\_paper(paper)`: This method allows the HCP to add a new medical paper or publication to their repository. When called, it appends the provided paper to the available medical papers list.

2. `accept\_appointment(appointment\_id)`: This method allows the HCP to accept a booking request for a specific appointment. When called, it changes the status of the appointment associated with the appointment\_id from pending to accepted.

3. `reschedule\_appointment(appointment\_id, new\_date\_time)`: This method allows the HCP to reschedule an existing appointment. When called, it searches for the appointment with the specified appointment\_id and updates the appointment's date and time to the new\_date\_time.

5.4. Sequence Diagrams

5.4.1. User Registration Sequence Diagram

(Include a UML Sequence Diagram illustrating the sequence of interactions during the user registration process.)

5.4.2. Patient Appointment Booking Sequence Diagram

(Include a UML Sequence Diagram illustrating the sequence of interactions during the patient's appointment booking process.)

---

Please let me know if this section covers all the necessary details or if you have any specific additions or modifications in mind. Once you are satisfied with this section, we can move on to the next section, which is Implementation.

6. Implementation

The Implementation phase of the Smart Health App project in Python involves the actual development and coding of the app based on the detailed design specifications. This section provides an overview of the development process, technologies used (Python, Kivy for frontend, and Django for backend), and key implementation details for the User Authentication, Patient Profile, and Healthcare Professional Profile modules.

6.1. Development Process

The development process will follow an agile methodology to ensure continuous collaboration, frequent feedback, and iterative improvements. The project team will consist of developers, designers, testers, and domain experts working in sprints to achieve incremental progress.

6.2. Technologies Used

The Smart Health App will be developed using the following technologies in Python:

- Frontend: Kivy framework will be used for the frontend development to ensure compatibility with both Android and iOS platforms. Kivy is an open-source Python library for developing multitouch applications, making it suitable for mobile app development.

- Backend: The app's backend will be built using Django, a high-level Python web framework. Django provides a robust set of tools for building web applications and handling database operations.

- Authentication: Django's built-in authentication system will be used for user authentication, providing secure login and registration processes.

6.3. Key Implementation Details

6.3.1. User Authentication Module

- User registration will be implemented using Django's built-in User model and authentication views. When a new user registers with the app, a new User object will be created in the database.

- User login will be handled by Django's authentication views, where the user's credentials will be validated against the stored data in the database.

6.3.2. Patient Profile Module

- Patient profile management will be implemented with Django models and views. The Patient model will have fields to store personal information, medical history, and appointments.

- The patient will be able to update their medical history through a view that allows them to add new medical records to their profile.

- The patient will be able to schedule appointments with HCPs through a view that handles the appointment booking process.

- The patient will have the option to cancel previously scheduled appointments through a view that removes the appointment from their profile.

6.3.3. Healthcare Professional Profile Module

- Healthcare professional profile management will also be implemented with Django models and views. The HCP model will have fields for storing credentials, medical specialization, and available time slots.

- The HCP will have the ability to add new medical papers or publications through a view that allows them to add the papers to their profile.

- The HCP will be able to accept booking requests from patients through a view that updates the status of the appointment.

- The HCP will have the option to reschedule existing appointments through a view that updates the date and time of the appointment.

7. Testing

The Testing phase of the Smart Health App project is crucial to ensure that the app functions as expected, meets the specified requirements, and delivers a smooth user experience. This section outlines the testing approach, test scenarios, and methodologies used to validate the functionalities of the app.

7.1. Testing Approach

The testing approach for the Smart Health App will be a combination of manual testing and automated testing. Manual testing will be employed for critical user flows, user interface testing, and usability evaluation. Automated testing will be used for repetitive and extensive test scenarios, ensuring comprehensive coverage of the app's functionalities.

7.2. Test Scenarios

7.2.1. User Authentication Module

1. User Registration:

- Test if a new user can successfully register with valid email/phone number and password.

- Test if the app prevents registration with existing email/phone number.

2. User Login:

- Test if a registered user can log in with valid credentials.

- Test if the app rejects login attempts with incorrect passwords.

7.2.2. Patient Profile Module

1. Update Medical History:

- Test if a patient can add a new medical record to their profile successfully.

- Test if the app displays the updated medical history after adding a new record.

2. Schedule Appointment:

- Test if a patient can schedule an appointment with a valid healthcare professional and date/time.

- Test if the app prevents scheduling an appointment with invalid or unavailable slots.

3. Cancel Appointment:

- Test if a patient can cancel a previously scheduled appointment successfully.

- Test if the app removes the appointment from the patient's profile after cancellation.

7.2.3. Healthcare Professional Profile Module

1. Add Medical Paper:

- Test if an HCP can add a new medical paper to their profile.

- Test if the app displays the updated list of medical papers after adding a new paper.

2. Accept Appointment:

- Test if an HCP can accept a booking request for a specific appointment.

- Test if the app updates the appointment status to "accepted" after confirmation.

3. Reschedule Appointment:

- Test if an HCP can reschedule an existing appointment to a valid and available date/time.

- Test if the app updates the appointment date/time after rescheduling.

7.3. Methodologies Used

1. Unit Testing: Individual components and methods will be tested to verify their functionality in isolation.

2. Integration Testing: The interactions between different modules and components will be tested to ensure seamless integration.

3. User Acceptance Testing (UAT): The app will be tested by end-users to evaluate its usability and user experience.

4. Automated Testing: Automated test scripts will be written to perform repetitive and extensive tests, ensuring stability and reliability.

7.4. Bug Tracking and Reporting

During testing, any identified issues, defects, or bugs will be logged into a bug tracking system. The development team will prioritize and resolve these issues, and retesting will be performed to verify bug fixes.

8. Conclusion

The Smart Health App project aims to revolutionize the healthcare industry by providing a user-friendly platform that connects patients and healthcare professionals. Through the app's innovative features, such as user authentication, patient profile management, appointment booking, and medical paper repository, it offers a seamless experience for users to access healthcare services and information.

The project started with Requirement Elicitation, where the team gathered and prioritized functional and non-functional requirements through interviews, surveys, and discussions with stakeholders. The Analysis phase provided detailed use case descriptions, activity diagrams, class diagrams, and sequence diagrams, defining the app's behavior and interactions.

In the System Design phase, the architectural overview and database design were developed, and detailed design specifications for the User Authentication, Patient Profile, and Healthcare Professional Profile modules were provided. Python, with Kivy for the frontend and Django for the backend, was chosen as the technology stack for development.

The Implementation phase brought the design to life, and the app's frontend and backend were developed using Python and the selected frameworks. The User Authentication, Patient Profile, and Healthcare Professional Profile modules were implemented, providing users with seamless registration, profile management, and appointment booking.

Throughout the project, comprehensive testing was conducted to ensure the app's functionality, performance, and user experience. Manual testing and automated testing methodologies were employed to validate key scenarios, and bugs were tracked and resolved to deliver a stable app.

In conclusion, the Smart Health App is a comprehensive solution that empowers patients and healthcare professionals with a secure and user-friendly platform. It aligns with the growing demand for digital health solutions and holds the potential to make a positive impact on the healthcare industry.

9. Future Enhancements

While the current version of the Smart Health App provides essential functionalities, there is always room for improvement. Some potential future enhancements include:

1. Telemedicine Integration: Integrate video conferencing capabilities to facilitate virtual doctor-patient consultations.

2. Personalized Health Recommendations: Leverage AI algorithms to provide personalized health recommendations based on user data.

3. Health Monitoring: Include features to monitor health metrics such as heart rate, blood pressure, and activity levels using wearable devices.

4. Collaborative Medical Research: Enhance the medical paper repository to support collaboration and contribution from healthcare professionals worldwide.

5. Integration with Health Wearables: Integrate the app with popular health wearables to allow users to sync and track their health data seamlessly.

10. Acknowledgments

TBD<need to work on this part>

Reference Links:

1. Kivy Documentation: https://kivy.org/doc/stable/

2. Django Documentation: https://docs.djangoproject.com/en/stable/

3. MongoDB Documentation: https://docs.mongodb.com/

4. Agile Methodology Overview: https://www.agilealliance.org/agile101/

5. UML Documentation: https://www.uml-diagrams.org/

Glossary:

1. Smart Health App: The mobile application developed to connect patients and healthcare professionals for seamless healthcare services.

2. HCP: Healthcare Professional - Refers to doctors, nurses, or other medical practitioners using the app.

3. UI: User Interface - The visual and interactive elements of the app that users interact with.

4. UX: User Experience - The overall experience and usability of the app from a user's perspective.

5. ERD: Entity-Relationship Diagram - A graphical representation of the database structure and relationships between entities.

6. Agile: An iterative and incremental approach to project management and software development.

7. Use Case: A description of interactions between actors (users or external systems) and the system to achieve a specific goal.

8. Activity Diagram: A diagram that shows the flow of activities and actions within a process or use case.

9. Class Diagram: A static structure diagram representing the system's classes, attributes, and relationships.

10. Sequence Diagram: A dynamic diagram showing the objects' interactions in a specific use case or scenario.

11. State Machine Diagram: A diagram representing an object's states and state transitions within the app.

12. Bug Tracking: Identifying, logging, and resolving software defects or issues.

13. Telemedicine: Remote medical consultations and diagnoses using technology, such as video conferencing.

14. AI: Artificial Intelligence - The simulation of human intelligence in machines that can perform tasks intelligently.