Software Requirements Specification

for

Health-Sync

Prepared by < Amy Freij-Camacho(211912090), **Prabsimran Badwal**(169033506), **Jayden Lo**(169037199), **Pratyaksh Kumar**(169016114), **Jashandeep Singh**(169018282)>

Wilfrid Laurier University

27 July 2024

Finished Product: https://github.com/simranbadwal/HealthSync

Table of Contents

Summary

Health Sync is a web-based application aimed at solving a global challenge faced by healthcare management, which is ineffective communication with doctors and difficulties in managing medical records on existing platforms. By enabling effective communication with doctors and simplifying medical record management, Health Sync helps staff concentrate more on patient care and maintain more accurate patient information. A user can just visit the landing page, create an account, and register as a doctor or patient. If he's a patient, the patient can create medical records, add doctors to profiles, and view and update medical records. If he's a doctor, he can view and update medical records. As you can see, a click of a button on Health Sync can make health care management more efficient and effective. In order to make health sync possible, the architecture is built around three key subcomponents. The Account Management Sub-component handles user registration, login, and logout, interfacing with the User Database to manage and secure user profiles. The Patient Management Sub-component supports patients by allowing them to update and view their medical records and add doctors to their profiles, working closely with the Notes/Health Form Database to ensure accurate management of medical records. Meanwhile, the Doctor Management Sub-component enables doctors to view and update patient records, interacting with the Health Form Database to store and retrieve the latest patient information. Together, these components ensure efficient and secure management of healthcare data and user interactions. For data protection and confidentiality, it also needs to be encrypted in transit and at rest. All of these contribute to the smooth operation and success of Health Sync.

Introduction

Purpose

The purpose of this Software Requirements Specification (SRS) and Design Document is to provide a comprehensive description of the Health-Sync system. This document outlines the functional and nonfunctional requirements, system features, constraints, and other key aspects necessary for the development, deployment, and detailing of the architectural design for Health-Sync. The intended audience for this document includes project managers, developers, testers, and stakeholders involved in the Health-Sync project.

Project Scope

Health-Sync is a web-based application designed to streamline the management of medical records and help improve communication between patients and healthcare providers. The system allows patients to manage their health records and share them with doctors. Doctors can view and update patient medical records and manage patient profiles. The primary objectives are to make sure a patient's medical records are as accurate as possible and improve patient care.

Problem Definition

Users often struggle with ineffective communication with doctors and difficulties managing their medical records on current platforms. Health-Sync addresses these issues by providing comprehensive tools for real-time interaction and medical record management. This ensures that both patients and doctors can keep a patient's medical records accurate and up-to-date,and this allows for better communication between them. This ultimately improves the overall efficiency and effectiveness of medical record management while at the same time allowing healthcare staff to focus more on patient care.

Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **HTTP:** Hypertext Transfer Protocol
- **UML:** Unified Modeling Language
- **Health-Sync:** The web-based application being developed

References

- Apple Health App (Inspiration for Front End)
- Verismo Health App

- TELUS Health MyCare

Overall Description

Product Perspective

Health-Sync is designed as a standalone web application aimed at streamlining healthcare management. It is intended to replace or complement existing medical record management systems by providing a modern, user-friendly interface.

Product Features

The main functions of Health-Sync include:

- **User Registration and Authentication:** Allows users to create accounts and securely log in to the system.
- **Doctor Search:** Allows users to search for their doctors on the platform.
- **Medical Record Management:** Enables patients to create, view, manage ,and comment on their medical records, and allows doctors to update records.
- **Patient-Doctor Communication:** Allows Patients to enter notes that seem relevant for the doctor to know.
- **Role-Based Access Control:** Ensures that users have appropriate access to system functionalities based on their roles.

User Classes and Characteristics

- **Patients**: Individuals who use the system to manage their medical records and add notes. They need an intuitive interface to view and update their medical information and communicate with doctors.
- **Doctors:** Healthcare providers who use the system to manage patient profiles and update medical records. They require secure access to patient information and efficient communication tools.

Operating Environment

- **Client-Side:** Web browsers (Chrome, Firefox, Safari, Edge) on desktop and mobile devices.
- **Server-Side:** Cloud-based servers running on Linux, with a database system i.e. SQLAlchemy
- **Communication:** Secure HTTP protocol for data transmission.

Constraints

- All data must be encrypted in transit and at rest to ensure security and confidentiality.

- The system must implement strong authentication and authorization mechanisms.
- The system should be designed to support a growing number of users without performance degradation.

User Documentation

Data Flow Diagram: Refer to **Appendix A**. This diagram illustrates the contextual level view of the system and the flow of data between the various components of the system.

Use Case Diagrams: Refer to **Appendix B**. This diagram illustrates the interactions between different user roles (Visitor, RegisteredUser, Patient, Doctor) and the system functionalities (e.g., registration, login, medical record management)

Class Diagrams: Refer to **Appendix C.** This diagram provides a detailed view of the system's data structure, including classes for User, Health Form ,and Note with associated attributes and methods.

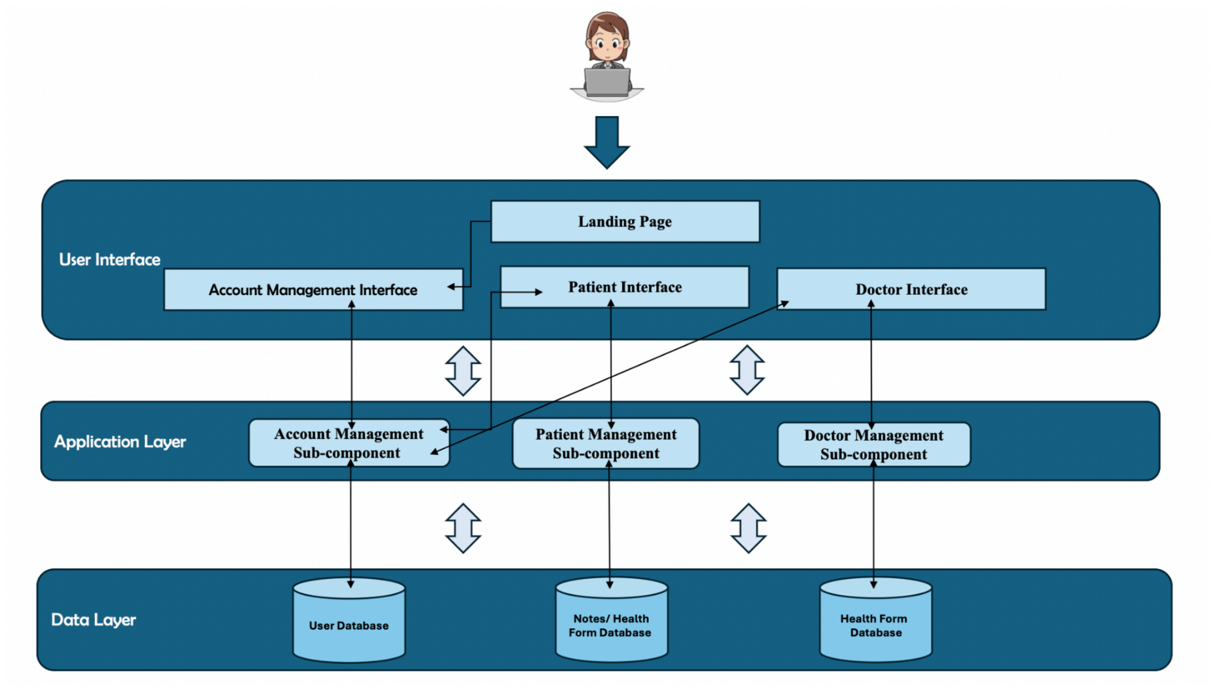
Activity Diagrams- Authentication: Refer to **Appendix D.** Visualises the process flow for user registration and login. Steps involved: Entering credentials, submitting credentials, validating credentials, and redirecting the dashboard.

Sequence Diagram-Searching for Doctors: Refer to **Appendix E.** Provides a detailed view of the interactions between components in a time sequence while searching for a doctor in the health sync database.

Sequence Diagram- Rendering Landing Page: Refer to **Appendix F.** Provides a detailed view on interaction between components of the system to return a landing page to users to show information about the health sync platform.

Component Diagram: Refer to **Appendix G.** Models the structure and interactions of the major components of our project.

System Architecture



User Interface
Landing Page
Account Management Interface
Patient Interface
Doctor Interface
Application Layer
Account Management Sub-component
Patient Management Sub-component
Doctor Management Sub-component
Data Layer
User Database
Notes/ Health Form Database
Health Form Database

Overview of Top-Level Component Functionalities and Feature Enablement and Overview of Sub-Component Functionalities and Interactions:

User Interface	
Landing Page	Visitors can learn about the platform and see a list of doctors,and it enables access to create an account and login features.
Account Management Interface	Users are allowed to create an account and specify if they are a patient or a doctor and this enables the login and logout features.
Patient Interface	This allows patients to be part of the medical record management.It also enables patients to add doctors to profiles,and view, create, and update medical records.
Doctor Interface	This allows doctors to be part of the medical record management.It also enables doctors to add patients to their profiles,and view and update medical records.

Application Layer	
Account Management Sub-Component	Account Management Sub-component enables the User Registration, Login, and Logout, interfacing with the User Database for managing User Registration and Profiles.
Patient Management Sub-Component	The Patient Management Sub-component enables the creation, update, and viewing of medical records, as well as the addition of doctors in profiles and comments on medical records, and interfaces with the Notes/Health Form Database in managing and storing health information.
Doctor Management Sub-Component	The Doctor Management Sub-component enables doctors to view and update patient records. It also manages a patient's profile while interfacing with the Health Form Database. All the updated medical records are stored and retrieved in the Health Form Database.

Data Layer	
User Database	It stores user registration information and profiles and it enables User Registration,Login,and Logout.
Notes/Health Form Database	It stores medical records,comments from patients, and it enables the managing of the patient comments and medical records.
Health Form Database	It stores detailed updated patient medical records and it enables the management of medical records.

Interfaces Between Top-Level Architecture and Internal Structures:

User Interface and Application Layer

Landing Page and Account Management Interface:

The landing page provides options to create an account or log in, and when users interact with these options by registering or logging in, they are directed to the Account Management Interface.

Account Management Interface and Account Management Sub-component:

The Account Management Interface collects user input for registration, login, and logout. Then this input is sent to the Account Management Sub-component, which processes the registration and then communicates with the user database to manage user profiles. The Account Management Sub-component also interacts with the Account Management Interface to show authentication results.

Account Management Sub-component with Patient and Doctor Interfaces:

The Account Management Sub-component enables the user to register and login as a doctor or patient, and once the role of the user is determined, the Account Management Sub-component directs them to the correct interface. If they are doctors, the Account Management Sub-component directs them to the Doctor Interface, and if the user is a patient, the Account Management Sub-component directs them to the Patient Interface. When the doctor or patient wants to log out, the Patient or Doctor interface directs them to the account management sub-component for the log-out process.

Patient Interface with Patient Sub-component:

The Patient Interface allows users to create, view, and update their medical records, as well as add doctors to their profiles and comments on their medical records, and all the actions taken in the patient interface are sent to the Patient Sub-component. Then the Patient Sub-component performs the necessary operations and then updates the Notes/Health Form databases. This allows the Patient Management interface to always display the updated medical results from the Patient Management sub-component.

Doctor Interface with Doctor Sub-component:

The Doctor Interface allows users to view and update a patient's medical records, and all the actions taken in the Doctor Interface are sent to the Doctor Sub-component. Then the Doctor subcomponent performs the necessary operations and then updates the Health Form database. As a result, the Doctor Interface is able to edit and display to the user the updated medical records from the Doctor Management sub-component.

Application Layer and Data Layer

Account Management Sub-component and User Database:

The Account Management Sub-component updates the user database based on the input from user Registration or login, and once registration or login occurs, the sub-component interacts with the user database to store new user information and verify existing credentials.

Patient Management Sub-component and Notes/Health Form Database:

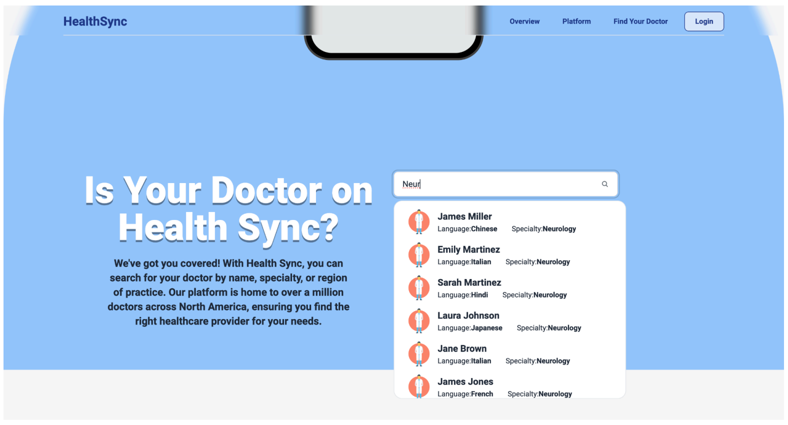
The Patient Management Sub-component interfaces with the Notes/Health Form Database to manage and store patient medical records and comments, and once a patient creates or updates medical records, the Patient Management Sub-component gives this information to the Health Form Database and retrieves it when needed.

Doctor Management Sub-component and Health Form Database:

The Doctor Management Sub-component interfaces with the Health Form Database to view and update patients medical records. The interaction can be seen when a doctor updates patient medical records; the changes are sent to be stored in the Health Form Database. Then the sub-component retrieves the updated data from the database to present it to the doctor when needed.

Functional Requirements

- The system must allow the user to look for doctors on the landing page.



Is Your Doctor on Health Sync?

We've got you covered! With Health Sync, you can search for your doctor by name, specialty, or region of practice. Our platform is home to over a million doctors across North America, ensuring you find the right healthcare provider for your needs.

Neur

James Miller
Language:Chinese Specialty:Neurology

Emily Martinez
Language:Italian Specialty:Neurology

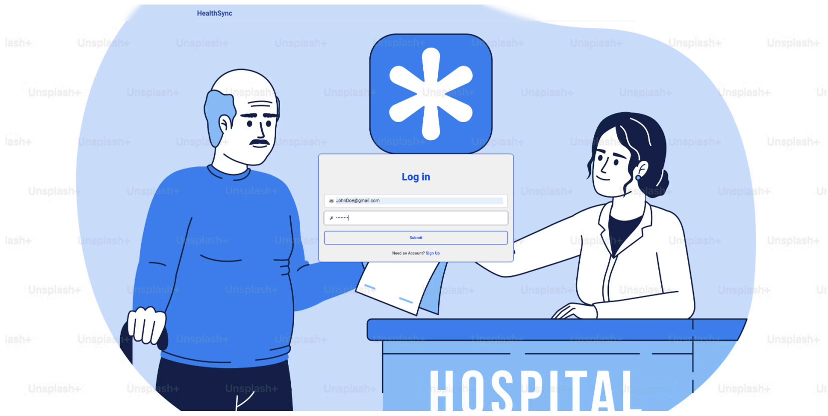
Sarah Martinez
Language:Hindi Specialty:Neurology

Laura Johnson
Language:Japanese Specialty:Neurology

Jane Brown
Language:Italian Specialty:Neurology

James Jones
Language:French Specialty:Neurology

- The system must allow users to input their email and password to login their account.



Log in

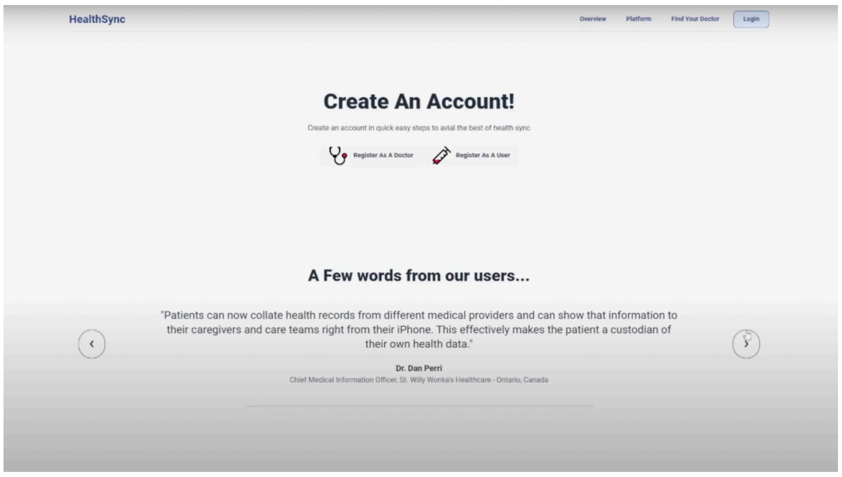
JohnDoe@gmail.com

••••••••

Submit

Need an Account? Sign Up

- The system must allow users to choose rather to create an account as a doctor or patient.



Create An Account!

Create an account in quick easy steps to avail the best of health sync

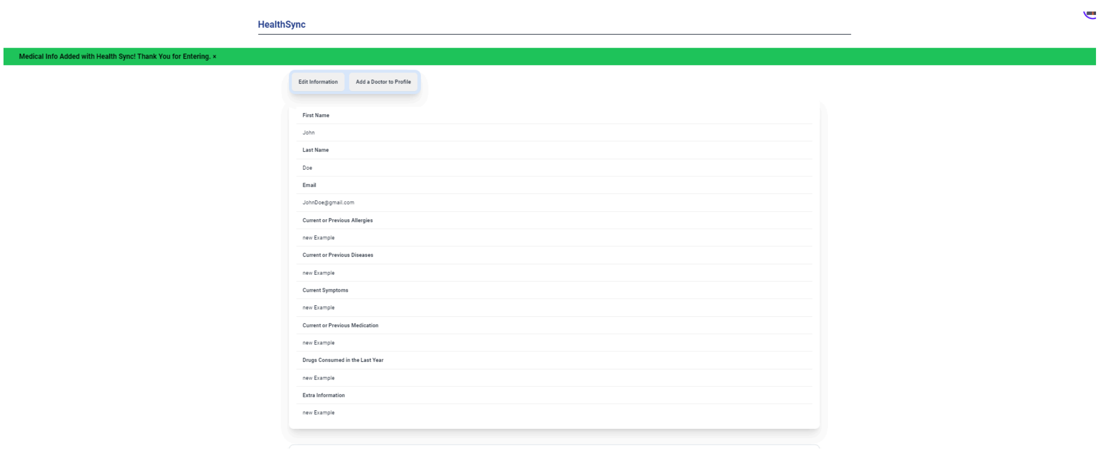
Register As A Doctor   Register As A User

A Few words from our users...

"Patients can now collate health records from different medical providers and can show that information to their caregivers and care teams right from their iPhone. This effectively makes the patient a custodian of their own health data."

Dr. Dan Perri
Chief Medical Information Officer, St. Willy Wonka's Healthcare - Ontario, Canada

- The system must also redirect users who are patients to the patient interface,and redirect patients who are doctors to the doctor interface.

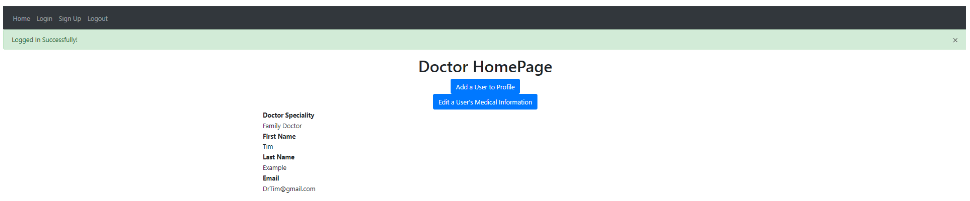


HealthSync

Medical Info Added with Health Sync! Thank You for Entering. ×

Edit Information | Add a Doctor to Profile

First Name
John
Last Name
Doe
Email
JohnDoe@gmail.com
Current or Previous Allergies
new Example
Current or Previous Diseases
new Example
Current Symptoms
new Example
Current or Previous Medication
new Example
Drugs Consumed in the Last Year
new Example
Extra Information
new Example



Home Login Sign Up Logout

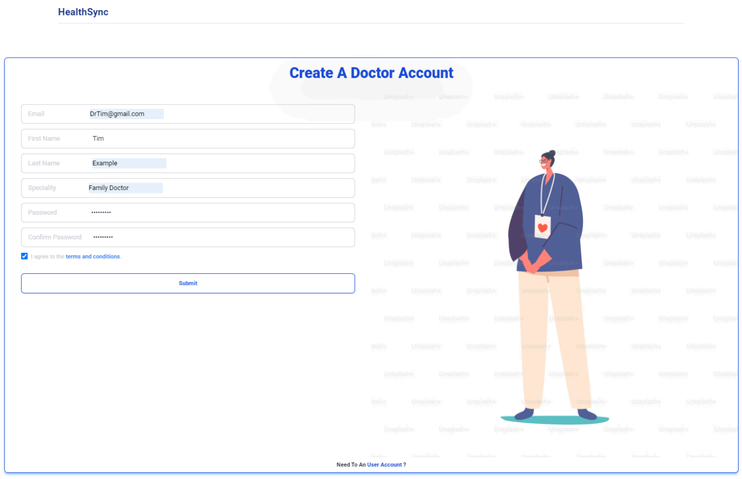
Logged In Successfully! ×

Doctor HomePage

Add a User to Profile
Edit a User's Medical Information

Doctor Speciality
Family Doctor
First Name
Tim
Last Name
Example
Email
DrTim@gmail.com

- The system must also allow doctors who are creating their account to input their email,first name,last name,specialty in medicine, password, and confirmation of that password.



HealthSync

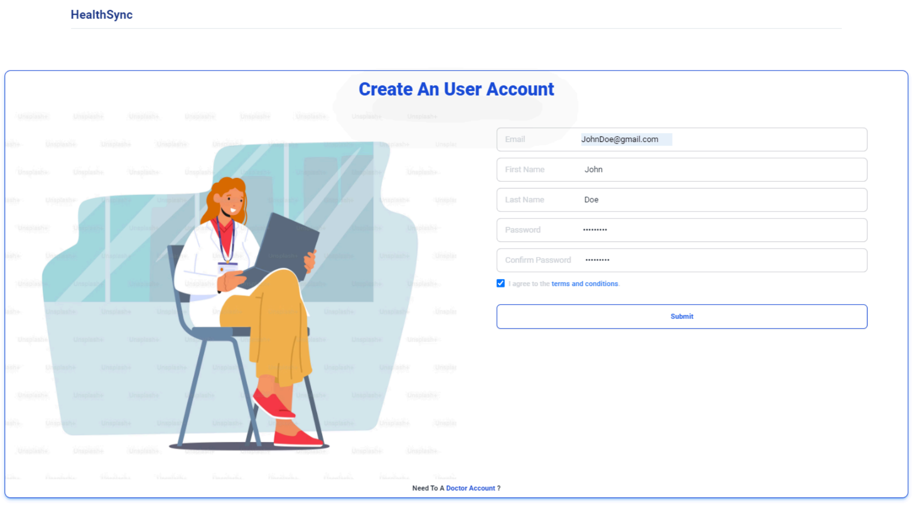
Create A Doctor Account

Email: DrTim@gmail.com
First Name: Tim
Last Name: Example
Speciality: Family Doctor
Password: ••••••••••
Confirm Password: ••••••••••
[✓] I agree to the terms and conditions.

Submit

Need To An User Account ?

- The system must also allow patients who are creating their accounts to input their email,first name, last name, password ,and confirmation of password.



HealthSync

Create An User Account

Email JohnDoe@gmail.com

First Name John

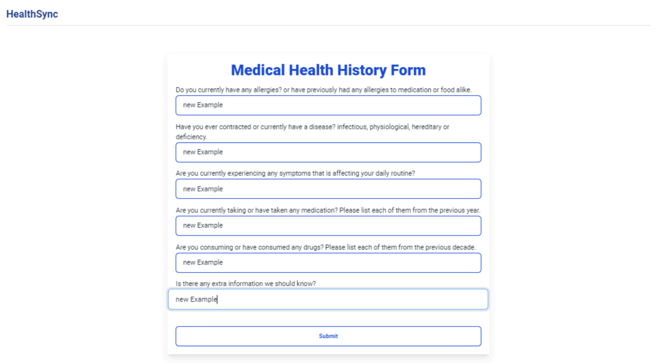
Last Name Doe

Password •••••••••

Confirm Password •••••••••

☑ I agree to the terms and conditions.

- The system must allow patients to input their allergies, diseases, symptoms, medications they are taking or have taken, drugs they have or are taking, as well as extra relevant information to create their medical records.



HealthSync

Medical Health History Form

Do you currently have any allergies? or have previously had any allergies to medication or food alike.
new Example

Have you ever contracted or currently have a disease? infectious, physiological, hereditary or deficiency.
new Example

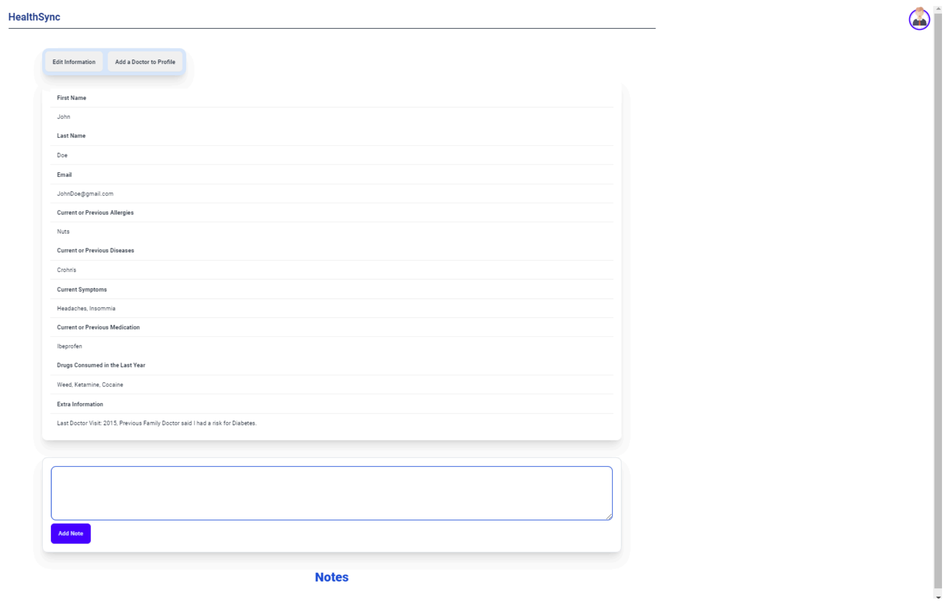
Are you currently experiencing any symptoms that is affecting your daily routine?
new Example

Are you currently taking or have taken any medication? Please list each of them from the previous year.
new Example

Are you consuming or have consumed any drugs? Please list each of them from the previous decade.
new Example

Is there any extra information we should know?
new Example

- The system must allow patients to view their medical records.


HealthSync

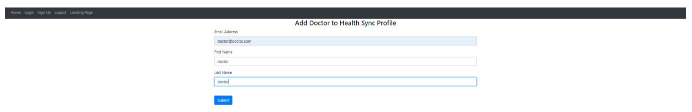
Edit Information | Add a Doctor to Profile

First Name
John
Last Name
Doe
Email
JohnDoe@gmail.com
Current or Previous Allergies
Nuts
Current or Previous Diseases
Crohn's
Current Symptoms
Headaches, Insomnia
Current or Previous Medication
Ibeprofen
Drugs Consumed in the Last Year
Weed, Ketamine, Cocaine
Extra Information
Last Doctor Visit: 2015, Previous Family Doctor said I had a risk for Diabetes.

Add Note

Notes

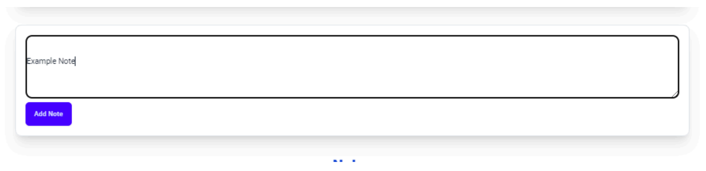
- The system must also allow the patients to update their medical records,add doctors to their profiles, and add comments.


Home Login Sign Up Logout Landing Page

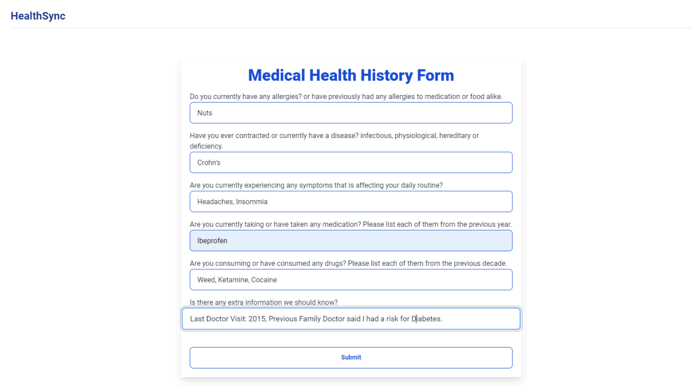
Add Doctor to Health Sync Profile

Email Address
doctor@doctor.com
First Name
doctor
Last Name
doctor

Submit


Example Note

Add Note

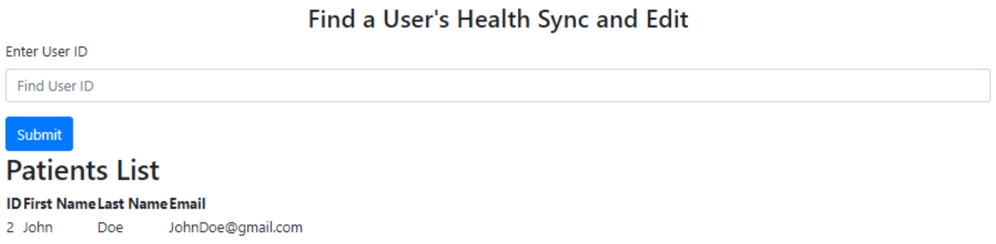

HealthSync

Medical Health History Form

Do you currently have any allergies? or have previously had any allergies to medication or food alike.
Nuts
Have you ever contracted or currently have a disease? infectious, physiological, hereditary or deficiency.
Crohn's
Are you currently experiencing any symptoms that is affecting your daily routine?
Headaches, Insomnia
Are you currently taking or have taken any medication? Please list each of them from the previous year.
Ibeprofen
Are you consuming or have consumed any drugs? Please list each of them from the previous decade.
Weed, Ketamine, Cocaine
Is there any extra information we should know?
Last Doctor Visit: 2015, Previous Family Doctor said I had a risk for Diabetes.

Submit

- The system must allow doctors to find and edit the patient’s medical records.



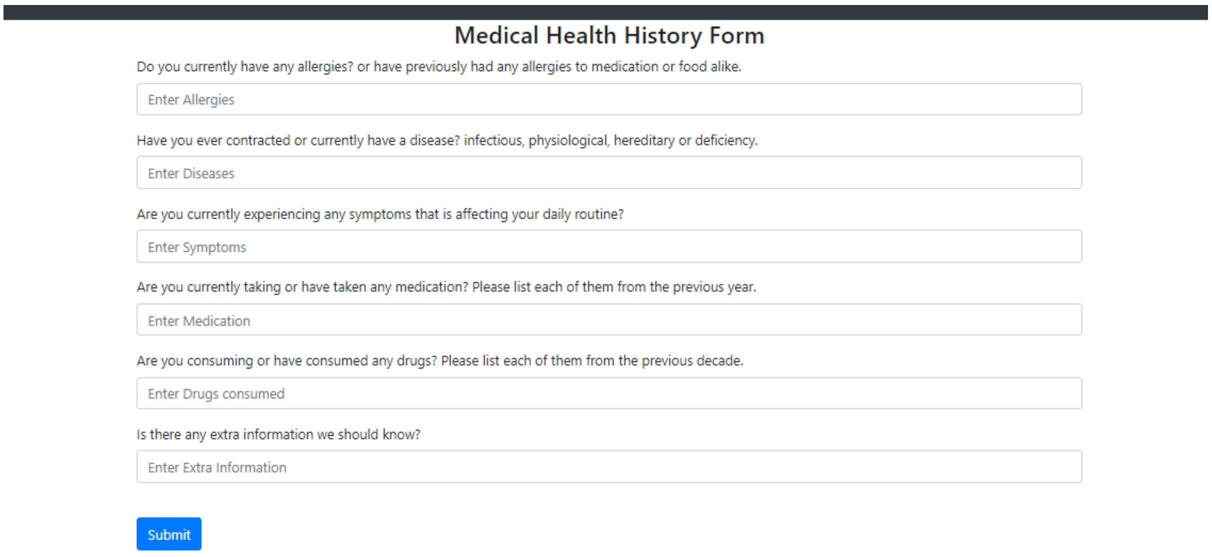
Find a User's Health Sync and Edit

Enter User ID

Find User ID

Patients List

ID	First Name	Last Name	Email
2	John	Doe	JohnDoe@gmail.com



Medical Health History Form

Do you currently have any allergies? or have previously had any allergies to medication or food alike.

Enter Allergies

Have you ever contracted or currently have a disease? infectious, physiological, hereditary or deficiency.

Enter Diseases

Are you currently experiencing any symptoms that is affecting your daily routine?

Enter Symptoms

Are you currently taking or have taken any medication? Please list each of them from the previous year.

Enter Medication

Are you consuming or have consumed any drugs? Please list each of them from the previous decade.

Enter Drugs consumed

Is there any extra information we should know?

Enter Extra Information

- The system must store the user's login information and profile in the User Database.
- The system must store the updated medical records and patient’s notes in the Notes/Health Form Database.
- The system must always display the updated patient’s medical record with the patient's notes.
- All of the system’s user interactions, such as logins, logouts, account creation, medical record updates, and viewing of medical records, must be processed and reflected in real-time within 1 second.
- The system must ensure that all updated data, such as patients' comments and updated medical records, are immediately available and consistent across all interfaces.

Nonfunctional Requirements

Performance Requirements

The system must be designed to handle at least 100 concurrent users accessing and interacting with it impacting the overall performance of the system. This involves ensuring that the server infrastructure, database management, and network bandwidth are all capable of supporting this load.

Maintainability Requirements

The system must be built using a modular architecture to make it easier to maintain and enhance in the future. This means that the system's components should be loosely coupled and highly cohesive, allowing for individual modules to be developed, tested, deployed, and maintained independently.

Security Requirements

- The system must ensure that users are authenticated before accessing any protected resources and that their actions are authorised based on their roles and permissions.
- Implement role-based access control (RBAC) to restrict user actions based on their roles and permissions.
- Enforce strong password policies, including complexity requirements.
- Implement secure session management, including session expiration and renewal mechanisms.

Solution Features

Patient Management

User Login

- Users can input their email and password to login to their account.

Account Creation

- Users can choose to create an account as a doctor or patient.
- Doctors can input their email, first name, last name, specialty in medicine, password, and password confirmation during account creation.
- Patients can input their email, first name, last name, password, and password confirmation during account creation.

User Interface

User Redirection
- The system redirects users to the patient dashboard if they are patients.
- The system redirects users to the doctor dashboard if they are doctors.

Patient Management

Medical Records History Form
- Patients can input allergies, diseases, symptoms, medications (current and past), drugs (current and past), and other relevant, extra information to create their medical records.

Medical Records Viewing
- Patients can view their medical records.
- Doctors can view the patient's medical records.

Medical Records Updating
- Patients can update their medical records.
- Doctors can update the patient's medical records.
- Updated medical records and patient's notes are always displayed.

Profile Management

- Patients can add doctors to their profiles.
- Patients can add comments to their medical records.

Data Management

- All Patient and Doctor login, and profiles are stored in the User Database.
- Updated medical records and patient's notes are stored in the Notes/Health Form Database.
- All user interactions, such as logins, logouts, account creation, medical record updates, and viewing of medical records, are processed and reflected in real-time within 1 second.
- All updated data, including patients' comments and updated medical records, are immediately available and consistent across all interfaces.

Solution Limitations and assumptions

Solution Limitations:

Document Upload Feature: With the document upload feature, doing it in Flask requires additional setup and libraries to handle file uploads securely and efficiently. Without proper handling, file uploads can pose security risks, such as malicious file uploads. To avoid this and extensive additional testing fall throughs with files it was decided best not to move forward with this feature. Flask's default configuration is not optimised for handling large file uploads or high traffic, which made us reject the use of additional tools or configurations in order to get this feature through.

Appointment Scheduling Feature: With the Appointment Scheduling Feature, Flask doesn't have built-in state management, making it challenging to handle complex user interactions and scheduling logic without additional setup. Implementing real-time updates and notifications for appointments required integrating with asynchronous communication protocols, which was tried and too complex to set up in Flask. Navigating through this we learn it heavily slowed down user runtimes, which led us to not implement this feature. This was due to the Complex scheduling logic, which often required sophisticated database queries and transactions to avoid conflicts and ensure data integrity. It allowed us to work with forms pretty easily, but the Appointment Management and Scheduling feature did not make it because of the harm it had on user runtimes and its complexity when featured.

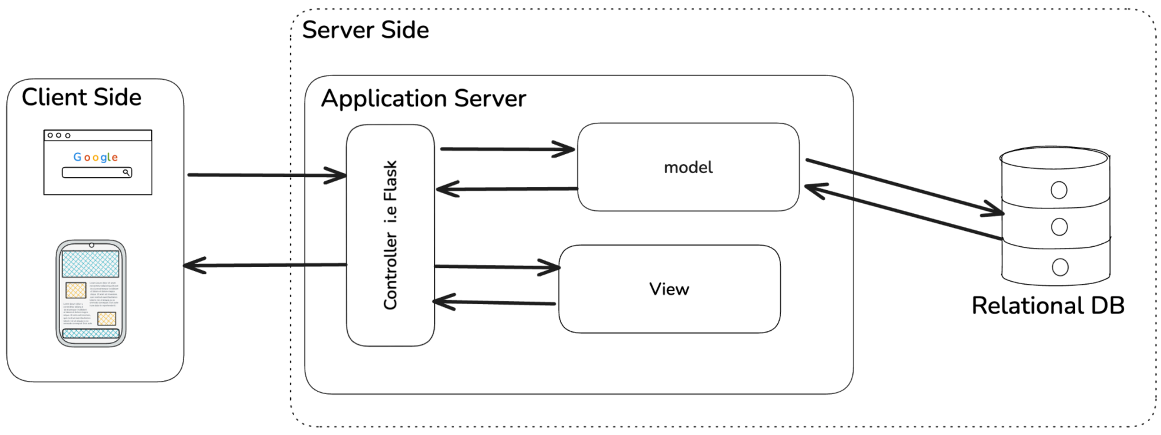
Doctor Notes Management (Viewing and Editing): We used Javascript to implement the notes featured on this project, but learned that allowing another user to achieve permissions caused security issues with data safely secured under encryption (passwords) needing decryption in order for other users to view said notes. Implementing a robust authentication and authorization system to ensure that only authorised users (doctors) can view and edit notes was dangerous for the security of our product. External libraries would have helped, but had issues integrating with other code in our software product, leading us to abandon this software feature.

Solution Assumptions:

1. Users will have access to a stable internet connection and a modern web browser.
2. Users will have basic knowledge of using web applications and managing health information online.

Appendices

Appendix A: Data Flow Diagram Level 0



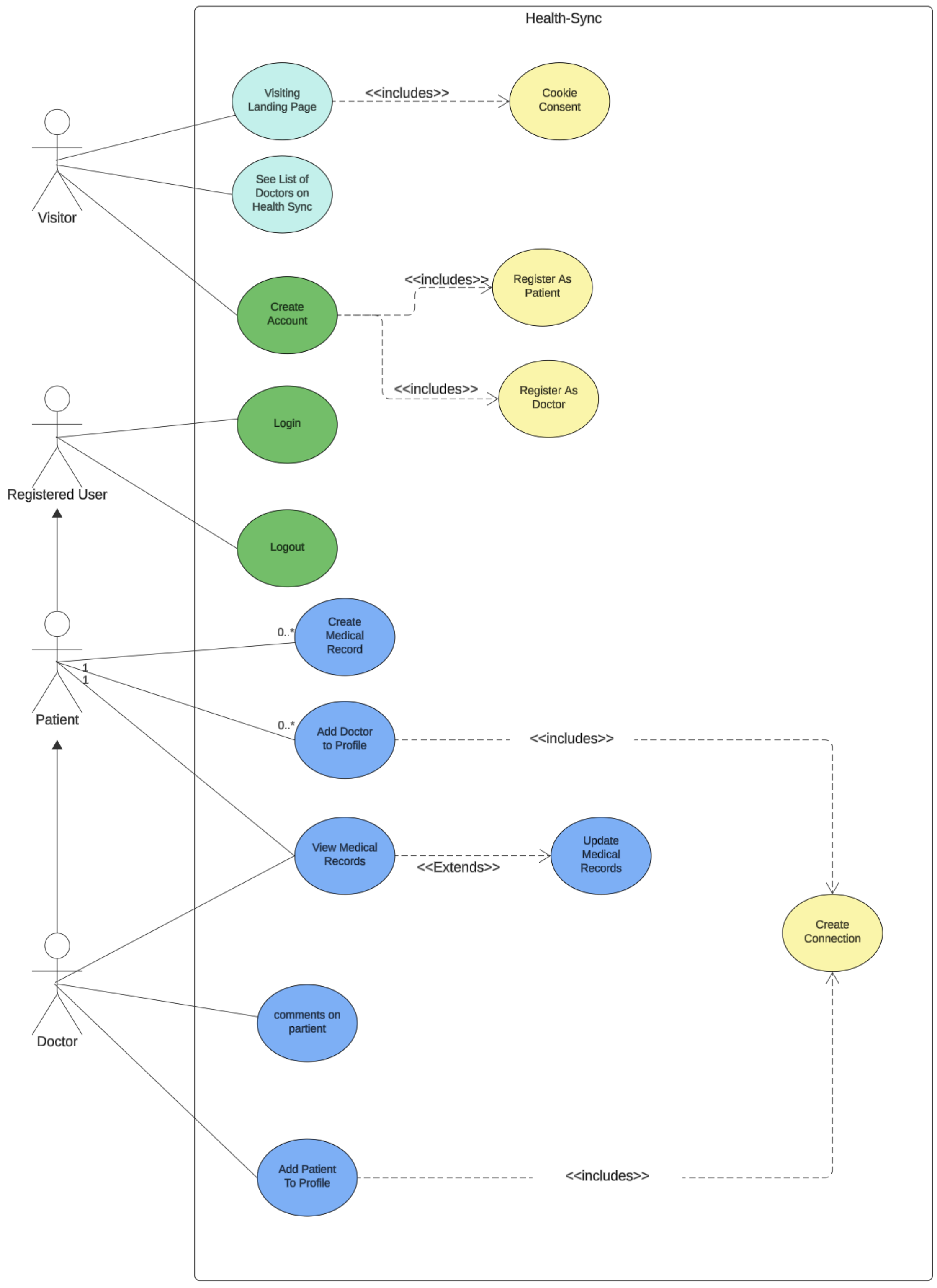
Description:

This diagram illustrates the overall data flow within the Health Sync System, depicting the interactions between the client side and the server side. The server side consists of an application which includes the controller (via Flask), model, and view. The relational database is connected to the model and view.

Components:

- Client Side:
  - Interacts with the server with a web interface
- Server Side:
  - Application:
    - Controller (via Flask): Manages the flow of data between the view and the model
    - Model: Handles the business logic and database interactions
    - View: Manages the presentation layer and user interface
- Relational DB:
  - Stores and retrieves data for application

Appendix B: Use Case Diagrams



Description:

This diagram shows the different use cases for various actors in the Health Sync System, including visitors, registered users, patients, and doctors. Each use case represents a functional requirement of the system.

Actors:

1. Visitor
    - A person who visits the Health Sync web application but does not have an account.
2. Registered User
    - A user who has created an account and can log in and out of the system.
3. Patient
    - A registered user who can create and view medical records, and add doctors to their profile.
4. Doctor
    - A registered user who can add patients to their profile and comment on patients.

Use Cases:

1. Visiting Landing Page (Visitor)
    - The visitor views the main landing page of the Health Sync application.
2. See List of Doctors on Health Sync (Visitor)
    - The visitor can view a list of doctors available on the Health Sync platform.
3. Create Account (Visitor)
    - The visitor can sign up and create a new account on the platform.
4. Login (Registered User)
    - The user logs into their account on the Health Sync system.
5. Logout (Registered User)
    - The user logs out of their account.
6. Create Medical Record (Patient)
    - The patient creates a new medical record in their profile.
7. Add Doctor to Profile (Patient)
    - The patient adds a doctor to their profile.
8. View Medical Records (Patient)
    - The patient views their existing medical records.
        - Extends: Update Medical Records
9. Update Medical Records (Patient)
    - The patient updates their existing medical records.
10. Add Patient to Profile (Doctor)
    - The doctor adds a patient to their profile.
11. Comments on Patient (Doctor)
    - The doctor comments on the patient's profile.

Relationships:

- Include: This indicates that a use case uses the functionality of another use case.
    - ‘Create Account’ includes ‘Register as Patient’ and ‘Register as Doctor’
    - ‘Visiting Landing Page’ includes ‘Cookie Consent’
- Extend: This indicates that a use case can be extended by the functionality of another use case.
    - ‘View Medical Records’ extends ‘Update Medical Records’

Details of Two Use Cases

- Create Account

  Actor: Visitor

  Description: This use case allows a visitor to sign up and create a new account on the Health Sync platform. It includes filling out personal details and setting up login credentials.

  Steps:

    - Visitor clicks on the "Create Account" button on the landing page.
    - System displays a form to enter personal details (name, email, password, etc.).
    - Visitor fills in the form and submits it.
    - System validates the details and creates a new account.
    - Visitor is notified of successful account creation and can now log in.

  Product Features: User registration, Account management

- View medical Records

  Actor: Patient

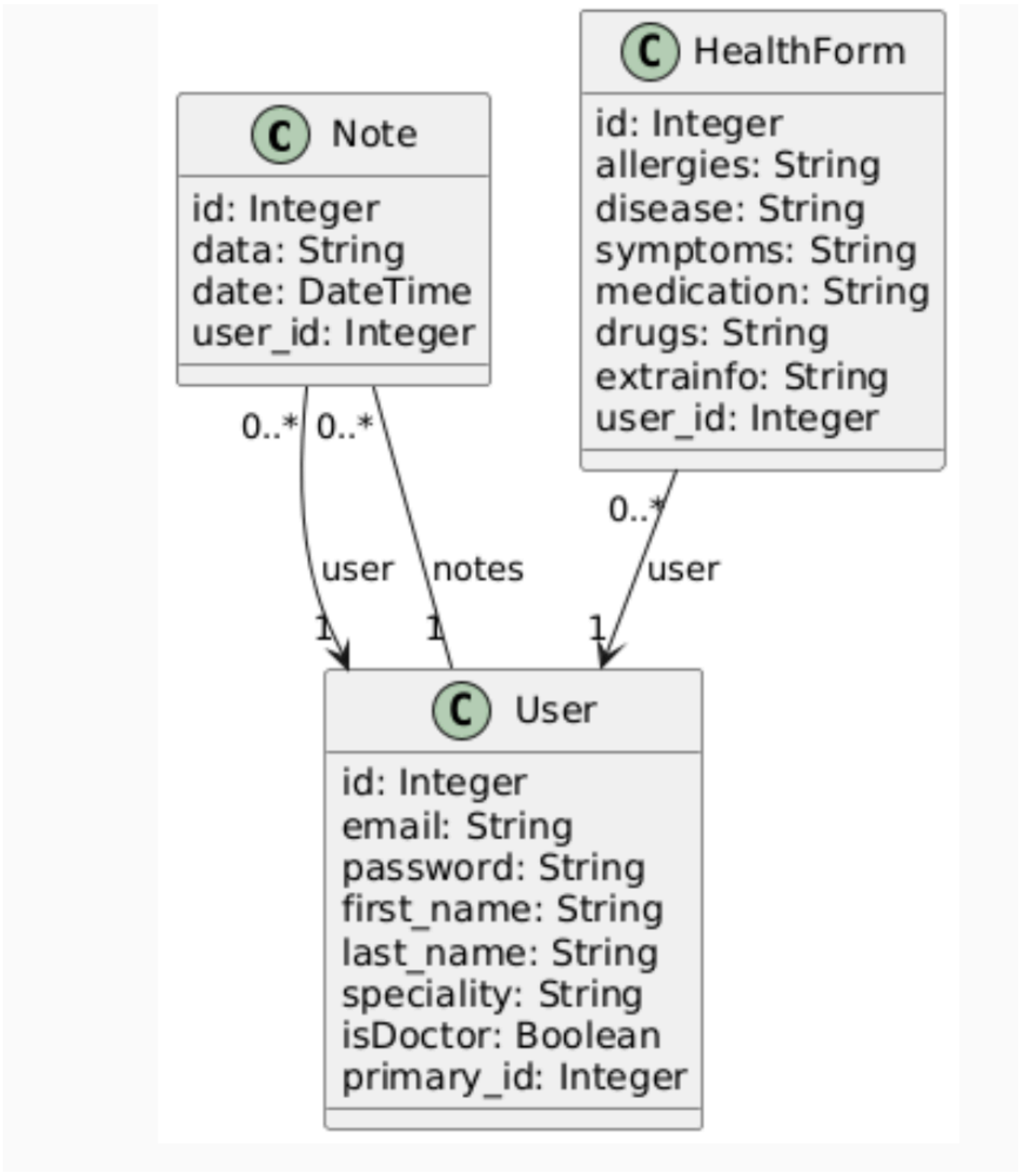
  Description: This use case allows a patient to view their existing medical records stored in the Health Sync system. It includes accessing the medical records section from the patient’s profile.

  Steps:

    - Patient logs into their account.
    - Patient navigates to the "Medical Records" section.
    - System retrieves and displays the patient's medical records.
    - Patient can view the details of each record.
    - If needed, patient can choose to update any record (extends to Update Medical Records).

  Product Features: Medical record management, Data retrieval and display

Appendix C: Class Diagrams



Description:

Classes and their Responsibilities:

1. RegisteredUser
    a. Attributes:
        - ‘id: integer’
        - ‘email: String’
        - ‘password: String’
        - ‘first_name: String’
        - ‘last_name: String’
        - ‘speciality: String’
        - ‘isDoctor: Boolean’
        - ‘primary_id: Integer
    b. Responsibilities: This class manages the user information like user authentication and profile details and it also acts as a base class for other users present in the system.

2. Note
    a. Attributes:
        - ‘id: Integer’
        - ‘data: String’
        - ‘date: DateTime’
        - ‘user_id: Integer’
    b. Responsibilities: This class encapsulates the note made by the doctors on the medical records.

3. HealthForm
    a. Attributes:
        - ‘id: Integer’
        - ‘allergies: String’
        - ‘disease: String’
        - ‘symptoms: String’
        - ‘medication: String’
        - ‘drugs: String’
        - ‘extrainfo: String’
        - ‘user_id: Integer’
    b. Responsibilities: This class encapsulates the health form details of a user and also records the medical information related to the user’s health status.

Relationships:

1. User - Note
    a. Type: Association
    b. Multiplicity: 1 to 0..* (One User can have multiple notes)
    c. Description: Each registered user can create multiple notes, but a note belongs to a specific user.

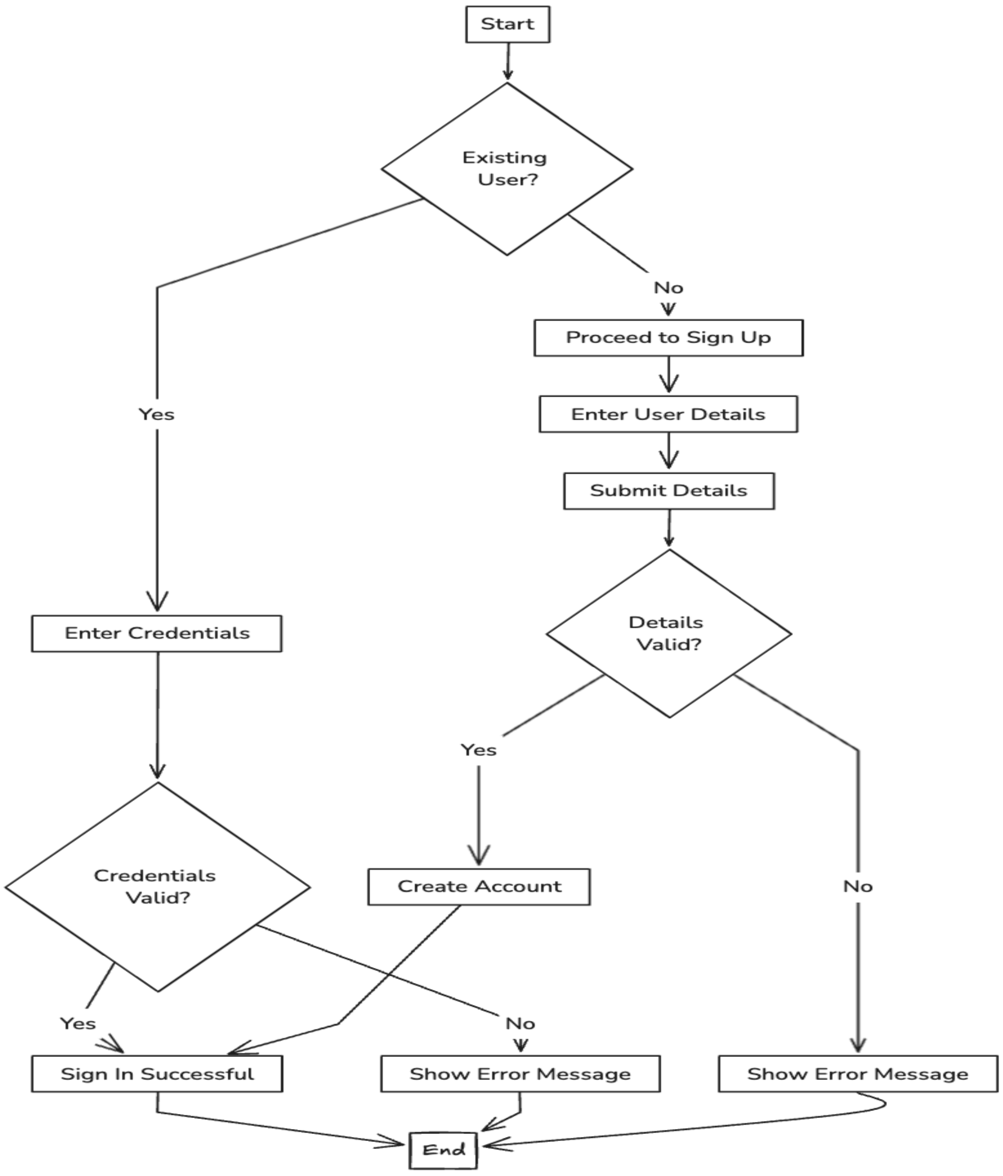
2. User - HealthForm
    a. Type: Association
    b. Multiplicity: 1 to 0..*(One user can multiple HealthForms)
    c. Description: Each registered user can have multiple health forms, but a health form is associated with one user.

Design Decisions

- Cohesion: Each class has a single responsibility that is increasing cohesion. For example, the 'Note' class focuses solely on medical record information and the 'HealthForm' class focuses solely on the medical information of the user.
- Coupling: The system is designed in a way so that it reduces coupling by using clear associations. For instance, the 'Gender' enumeration decouples gender representation from the 'RegisteredUser' class.

Code Reuse: The base 'User' class encapsulates the common attributes and operations for all types of users which promotes code reuse and also avoids duplication as well.

Appendix D: Activity Diagram for Authentication Process



Description:

This diagram outlines the steps involved in the authentication process for a user in the Health Sync System.

The process begins with determining if the user is an existing user.

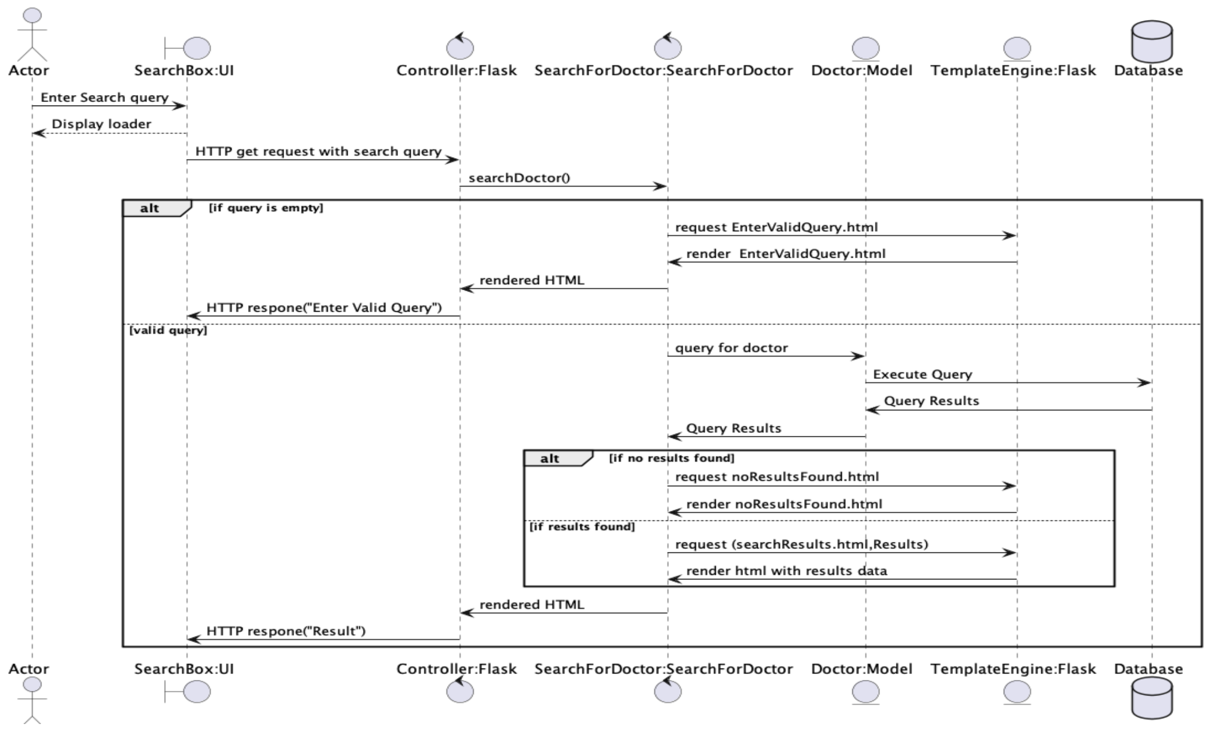
If the user is not an existing user, they proceed to sign up by entering their details and submitting them. The system then checks if the details are valid and if they are valid, an account is created; otherwise, an error message is shown.

For existing users, they enter their credentials, which are validated by the system and if the credentials are valid, the user successfully signs in. If not, an error message is displayed.

Steps:

1. Start
2. Check if the user is an existing user
    - 2.1. If no, then proceed to sign up
    - 2.2. If yes, then enter the required credentials
3. For Sign Up:
    - 3.1. Enter user details
    - 3.2. Submit the details
    - 3.3. Check if details are valid
        - 3.3.1. If valid, then create an account
        - 3.3.2. If not valid, show error message
4. For Existing Users:
    - 4.1. Validate credentials
        - 4.1.1. If valid, sign in successfully
        - 4.1.2. If not valid, show error message
5. End

Appendix E: Sequence Diagram:Searching for doctor



Description:

The sequence diagram shows the interaction between various components when a visitor on the landing page wants to search for a doctor. This process involves multiple components within the model view controller architecture implemented using flask.

1. **User/Actor Interaction:**
   - The user starts the process by entering either name or specialty of the doctor he/she is looking for.
   - To indicate that a search is being performed on the backend server, a loader is displayed asynchronously.

2. **HTTP Request:**
   - HTMX is used to perform a GET request containing the search query to the flask controller responsible for managing incoming and outgoing requests.
   - The controller, in turn, invokes the searchDoctor () utility.

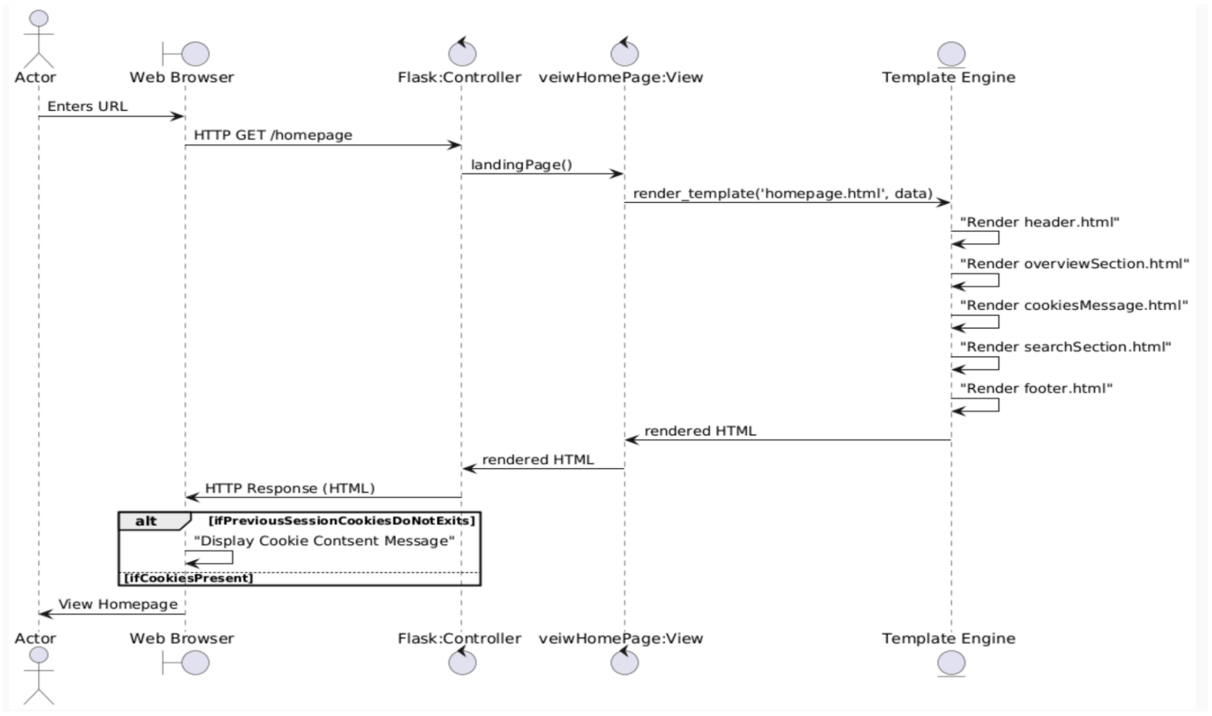
3. **SearchDoctor Utility:**
   - This handles the processing of the searching query. Checks if the entered query is valid alphabetical value and is not an empty string.

- If the query is valid then only the next operation is performed. The valid search query is sent to the model which in turn performs the search operation on the database and returns the result.
- This utility function is designed with data coupling and high cohesion in mind, ensuring that dependent functions are modular and independent based on functionality.
- This centralises specific functionality, promoting reusability and maintainability.

4. HTTP Response:

- If the results are found in the database the results are rendered to the user, else no results found message is rendered to the user.
- The model sends the result to the controller which in turn renders the results to the user.

Appendix F: Sequence Diagram: Rendering Landing Page



Description:

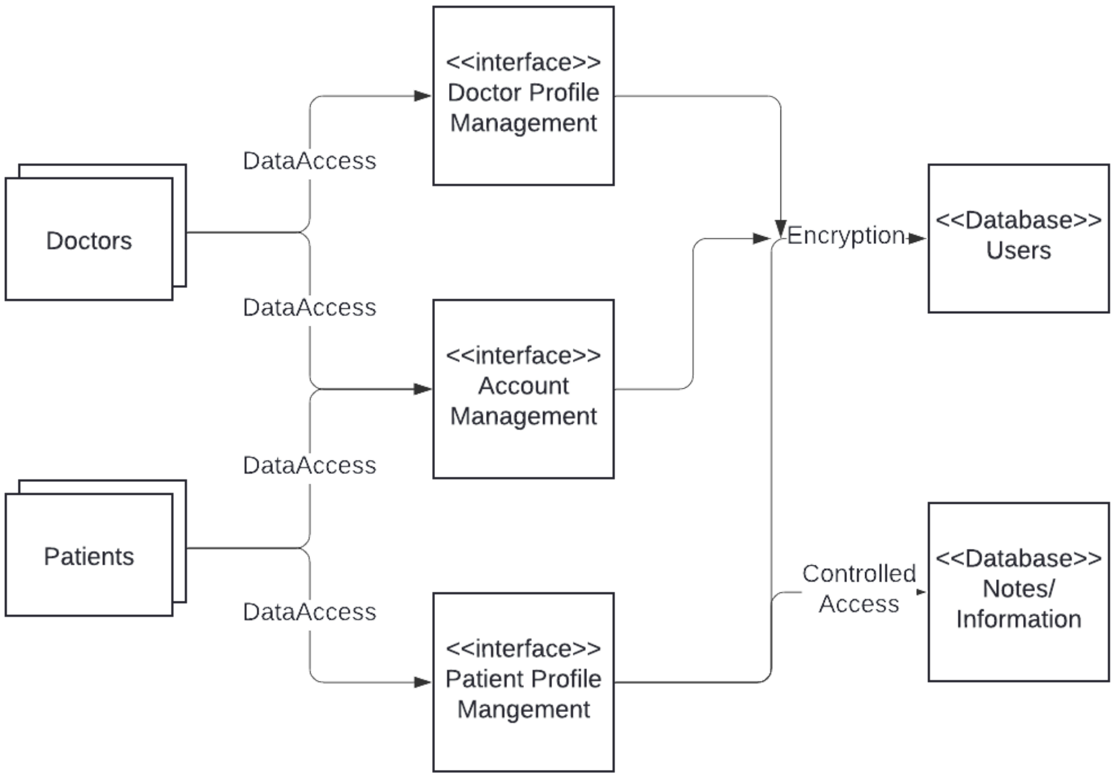
This sequence diagram details the process flow for loading the homepage of the Health Sync App, highlighting the interactions between the user, Flask's controller, views, and template engine.

1. **User/Actor Interaction:**
   · To access the web app the user must enter the web app's URL in the browser to access the homepage.
2. **HTTP Request:**
   · The web browser sends the HTTP GET request to the '/homepage' endpoint of the flask controller.
3. **Template Rendering:**
   · The flask controller in turn invokes the getLandingPage() utility which triggers the flask's template engine to generate a homepage.
   · During this process, the template engine renders and includes various section of the homepage, that are:
      i. Header
      ii. Overview Section
      iii. Cookie Consent Message
      iv. Doctor Search Section
      v. Footer

· Each section is split into individual components based on functionality. This is done to ensure highest possible cohesion and low coupling. This modular approach allows for easy updates to any section with minimal risk of introducing regression faults.

4. **HTTP Response:**
    · Fully rendered HTML web page with styling is rendered back to the user.
5. **Cookie Consent Message:**
    · On the client's web browser, a cookie consent message is displayed to the user.
    · If there are no previous session cookies, the system displays a cookie consent message to the user.
    · If cookies are present, this step is bypassed, and the homepage is displayed directly

Appendix G: Component Diagram


Doctors
Patients
DataAccess
DataAccess
DataAccess
DataAccess
<<interface>> Doctor Profile Management
<<interface>> Account Management
<<interface>> Patient Profile Mangement
Encryption
Controlled Access
<<Database>> Users
<<Database>> Notes/ Information

Users:
The main users of our application are doctors and patients, allowing patients to manage medical records themselves and connect with doctors.

Profile Management:
Doctors and patients are able to create accounts to provide, edit and change their information. This includes personal information and specifically for patients, a health form of general health/information form. Patients are also able to add personal notes. Also a part of their profiles is adding associated patients/doctors. Doctors are able to add patients they've worked with to their profile and patients are able add doctors they've worked with to their profile.

Databases:
To store users, an encrypted and secure database is used to store user information. Notes are only able to be seen by the patients, hence why a database with controlled access for the patient user is applied here. The patient's medical record are also stored in this database.