Product Design

**Team 36**

**Members: Talib Siddiqui, Khushi Wadhwa, Rayaan Khan, Yash Shivhare Introduction**

We have to create a mobile application based on the Amita care website, ie, https://www.amitacare.com/.

The user should first log in/Register with his credentials.

The user can get a preliminary diagnosis based on his symptoms using the Knidian (Symptom Checker) AI

Our app allows the user to book a therapy session, choose the Therapist/Doctor by looking at profiles of all the available doctors, and choose the one best suited for their needs.

Moreover, If the user has provided his ABHA number at registration , his preliminary diagnosis and appointment are added to his medical history stored in the government database

**System Overview**

The app is used by users to get a preliminary diagnosis , and book appointments with their preferred therapist.

**Design Overview**

**System interfaces**

**User Interface**

The user needs to first log in/Register with his credentials.

The user can then get a preliminary diagnosis based on his symptoms using the Knidian (Symptom Checker) AI if he wants. The AI returns the ailment the user is most likely to

have based on the symptoms and a recommendation regarding how to proceed further.

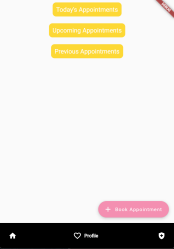
Our app allows the user to book a session with the user’s choice of Therapist/Doctor. The user can look at the profiles of all the available doctors, and choose the one best suited for their needs.

If the user has provided his ABHA number at the time of registration, his preliminary diagnosis and appointment are added to his medical history stored in the government database.

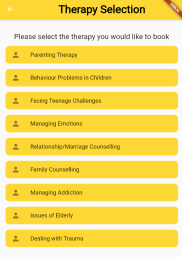
Some screenshots of the User face:

LOGIN/REGISTER PAGE

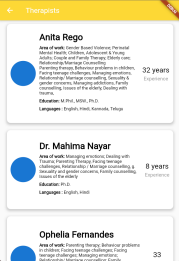
HOME PAGE



BOOKINGS PAGES



DOCTORS INFORMATION PAGE: **APIs**

****

**1) GET userRegistration(name, email, password, mobile):**

Creates a new account with the specified parameters.

Returns True OR False.

**2) POST forgotPassword(email):**

Sends a forgot password email to the user

**3) POST LoginCredentials(username,password):**

Used for authentication of the user and also returns the AUTH token which is further used throughout the application to fetch details.

**4) GET getCustomer():**

Fetches the customer details using the auth token.

**5) GET getAppointments():**

Fetches the appointments booked for the logged in user,

Returns 3 types of appointments, Past, Today and Future appointments.

**6) GET getTherapist(specialisation):**

At the bookings panel, we fetch the specific therapists that are specialized in the field selected by the user.

**7) GET getTherapistInfo(therapist\_id):**

After the specialization selection, we need to fetch the information of these specialized therapists for the user to choose from.

Returns the Name, Picture, Experience and a short bio.

**8) GET getTherapistSlots(therapist\_id):**

After the therapist selection we fetch the available slots for an appointment booking. Return the slots in Google Calendar Format.

**9) POST applyCoupon(obj):**

Used to apply discount or other offers on the payment gateway.

**10) POST bookAppointment(name,user\_id,age,sex,email,therapist,slot):**

After successful status code from the payment gateway, the appointment is booked.

**Model**

UML diagram -

https://lucid.app/lucidchart/ac2c9b65-e971-4e66-a35c-41b1da622e0c/edit?viewport\_loc=201%2 C399%2C1664%2C846%2C0\_0&invitationId=inv\_da7b8d80-e905-4c86-b457-c32087886f05

| Authentication | Class state  ● Authorizes the user.  Class behaviour  ● Able to logout and login the user |
| --- | --- |
| TherapistBox | Class state  ● Stores the information of the therapist in boxes like name, work, education, language, experience. |
| Appointment | Class state  ● Stores the details of the appointment like type, date, purpose, user, customer name, age, gender, number of sessions, created date. |
| Therapist | Class state  ● Stores name, age, gender and specialisation of the therapist Class behaviour  ● Fetch Therapist data, build therapy list |
| Knidian | Class state  ● To open the Knidian symptom checker on the application |
| Register | Class state  ● To register for the new user. Stores name, email, mobile, password.  Class behaviour  ● Checks if user is already registered |
| Reset | Class state  ● Used to reset the data |

|  | Class behaviour  ● Checks the mail, checks if it has been reset |
| --- | --- |
| Profile | Class state  ● Checks for the profile and displays it |
| Login | Class state  ● Stores email and password from the input  Class behaviour  ● Checks authorization and logins |
| Appointment  MobileLogin  User | Class state  ● Saves upcoming and previous appointments  Class state  ● saves otp sent and input otp from user  Class behaviour  ● Verifies OTP, gets OTP  Class state  ● Saves user information like name, email, number, education, address, occupation, married, dob, gender, password. |
| HomePage | Class state  ● Stores and displays the homepage data- login and knidian symptom checker |
| TherapySelection | Class state  ● Stores all different types of therapy and their data to be selected from the user |
| Customer | Class state  ● Stores name, age, gender of the customer |
| Knidian | Class state  ● Redirects to ABHA creation webpage |
| Landing | Class state  ● To get back to home page |

.

**Sequence Diagram(s)**

Use cases for the project are:

1. User signs up/login

2. User enters symptoms

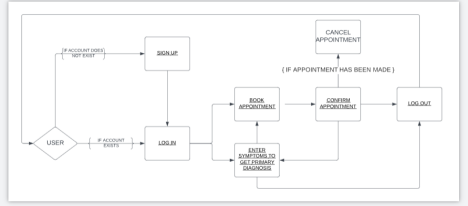
3. The app generates a primary prognosis

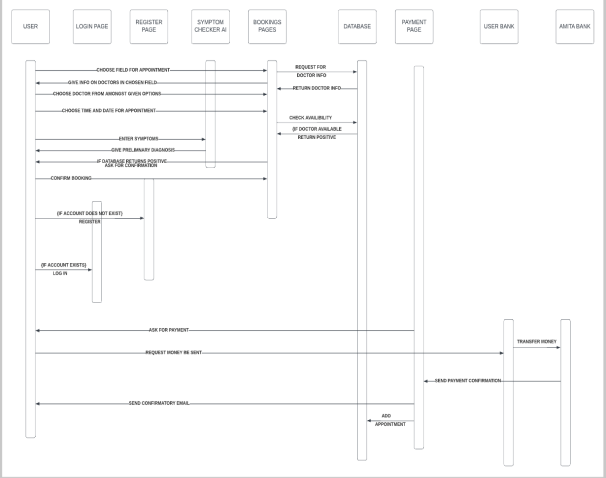
4. App books an appointment with a healthcare professional

5. User cancels an appointment

6. User logs out.

Flow Diagram :

Sequence Diagram :

**Design Rationale**

**1) Change of project**

At the start of the course, we were given the project to make a "Sign Language Translator" which aimed to create an on-the-go ML model which would process signs and interpret the meaning to the doctor(in the appointment or otherwise) right away.

After a few weeks of research and consideration our project was replaced with the undermentioned due to the Lack of Training data on Signs, especially the Indian Sign Language ( ISL ) and the time constraints which

**2) Choice of Stack for the App**

Initially we decided to go with a React Native + Mongo Stack which we had to ditch later after a few weeks of research and after considering the advantages of Flutter Development over React Native.

**3) Backend for the application**

We initially started out the backend with our dummy database which was due to a miscommunication with the client, later cleared and we got access to the Original Database making the application more production ready.

The final design is created in a mutual agreement and upon careful consideration of the time scope and the functionalities to be implemented.

It is better than the previous design, due to the planning and the methodologies followed in this sprint.

We have decided on a central control flow upon which further development is queued on the Version Control being used (GitLab), making the testing and integration phase easier than an ambiguous set of ideas.