

# **DESIGN DOC**

## **Team 49**

- **Pranav Agrawal**
  - **Nikunj Garg**
  - **Aman Raj**
  - **Divij D**

## **Introduction**

We are building a prototype app for our client “IkkyMinds” to provide the service of being able to book a Franchised Wellbeing center. The user would be able to select a center based on their preferences and choose a time based on the availability.

The purpose of this document is to give a high level overview of the current technical details of our project along with reasoning behind broad design decisions

## **System Overview**

Our software system is a mobile application for both IOS and android operating systems. Our application allows users to create a profile using their email address. We then ask for some auxiliary details like mobile number and such and after verifying the email, the profile creation process is then.

For our application, our core assumption is that the software is being used by a parent/Guardian to book a wellness center for their ward. Thus, we provide the user with a main profile page with the capability of changing most of the details(Except the registered email address) and adding details of their wards.

There is then a booking tab where the user is prompted to choose a booking type. Currently in our implementation, we allow for two types of bookings, namely activity booking and party booking and each type is handled in separate ways. Upon choosing the type of booking, the user is asked to provide the details of participants and they are then taken to the payment interface to confirm their booking.

Some potential use cases for this app are-

- 1) Ability for a guardian to drop off their child at a safe and engaging place while they complete their own tasks
- 2) Ability for a Guardian to host parties/get togethers for their children and their friends at a wellness center
- 3) Providing information about nearby wellness centers to parent
- 4) Providing a convenient and user friendly interface for finalizing bookings and handling payments

Our Plan is to hold all the information about the logistics and availability of slots along with data about the centers in a centralized database. All the information that a user would need/desire to know will be provided on our application from the database. Thus, not only will it be a convenient way for the user to make bookings, it will make management of the franchisee system easier for the organization

# Design Overview

## Architectural design

Our project is split up into a range of high-level subsystems. These subsystems are as follows -

1. Profile creation/login
2. User profile
3. Activity Booking
4. Party booking
5. Center selection
6. Payment interface

### Responsibilities assigned to Profile creation/Login are as follows

- Allowing user to enter Personal details into our system
- Verifying the Email address through an OTP based system
- Preventing Spam/dummy account creation
- Providing a way for the user to logout of the app
- Providing a way for the user to login to the app through their registered email

### Responsibilities assigned to User profile are as follows

- Providing a graphical interface for the user to verify their details
- Providing the ability for the user to edit their details
- Ability for the user to add details about their children

### Responsibilities assigned to Activity booking are as follows

- Specifying which of the children is the booking for
- Specifying the time of drop off for the booking
- Specifying the duration of the booking
- Selecting the activities associated with the booking
- Providing details about the prices for each activity

### Responsibilities assigned to Party booking are as follows

- Specifying the number of adults which shall be attending the event
- Specifying the number of children which shall be attending the event
- specifying the duration of the booking
- Specifying the start of the event
- Specifying the activities for the event
- Providing the details about the prices of each activity

### Responsibilities assigned to Centre selection are as follows

- Providing a list of centers to the user based on location proximity
- Providing details about the centers
- Allowing the user to select a center based on their preferences

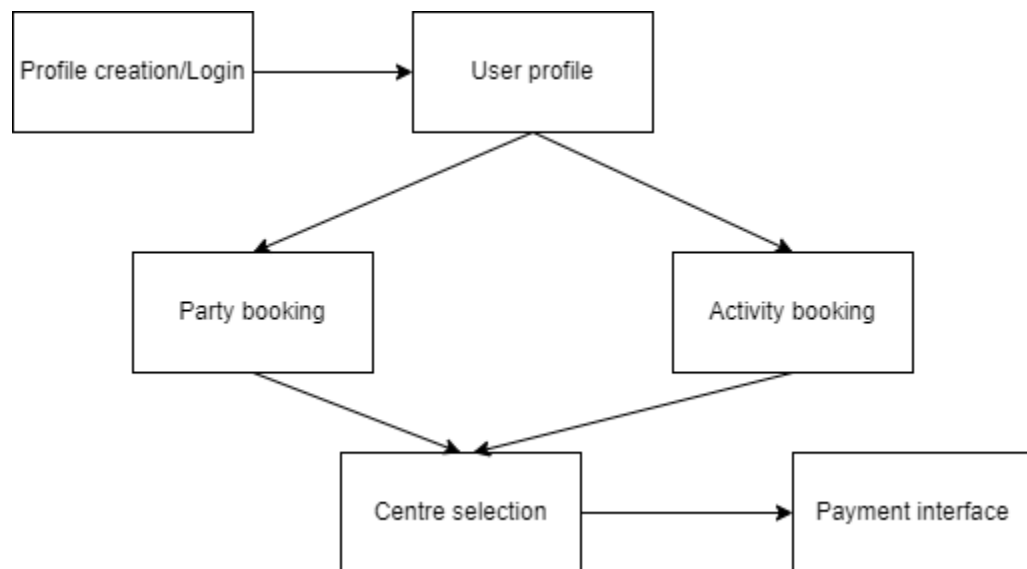
### Responsibilities assigned to The payment interface are as follows

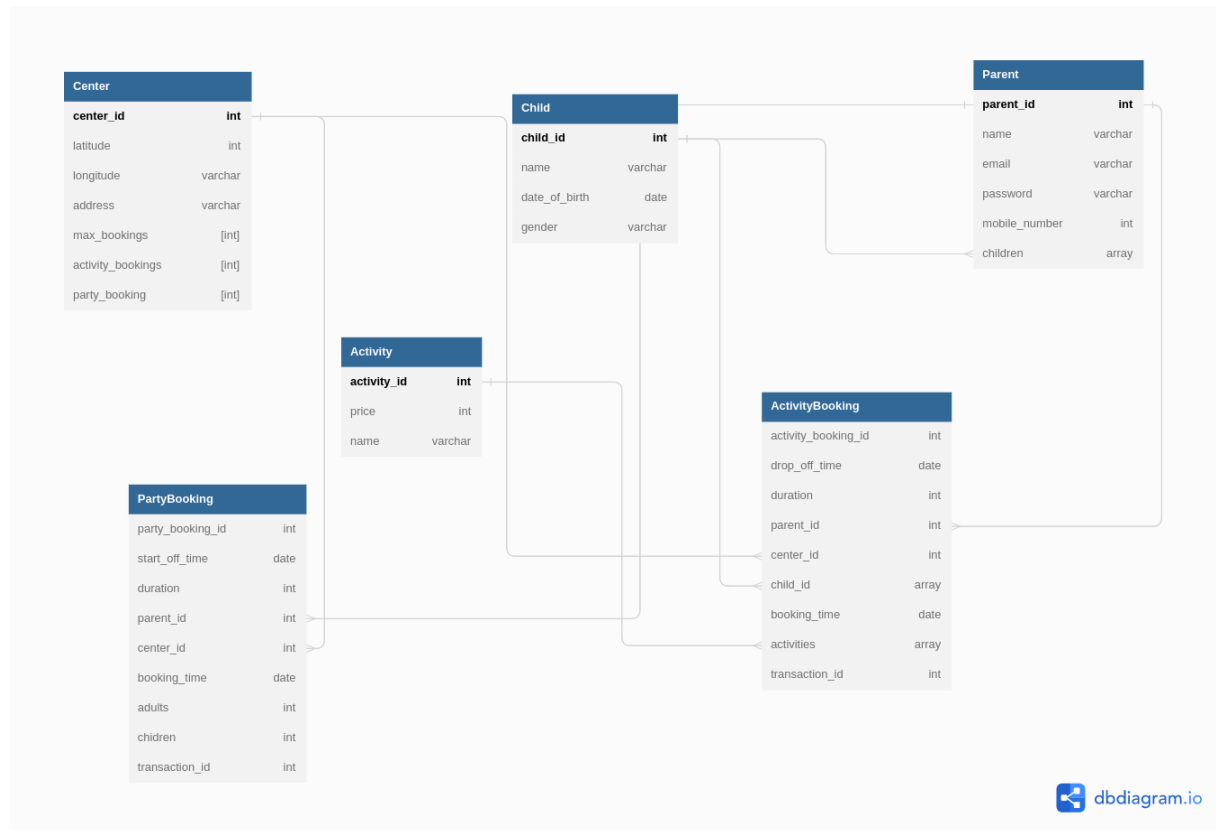
- Providing finalized information about the booking
- Providing details of taxes and convenience fees

- Providing options for making the payment
- Redirecting to the selected payment option

The profile subsystem interacts with both the profile creation system and the activity booking system. First, the details provided in the profile creation are directly shown in the profile page, unless they are edited later. Moreover, the details of the children provided in the profile page are used directly for making the Activity bookings

The Activities selected in the activity booking and party booking along with ot tabs are directly used to calculate the final cost of the booking. Moreover, for the center selection, only those centers are shown to the user which have the selected activities





## System interfaces

## User Interface

There is a simple Signup and signin pages for the user to access the app

LOGIN

SIGNUP

Email ID

Pranav.example@gmail.com |

Password

abcd1234 |

Confirm Password

abcd1234 |

SUBMIT

OTP will expire in 5 minutes

Enter OTP

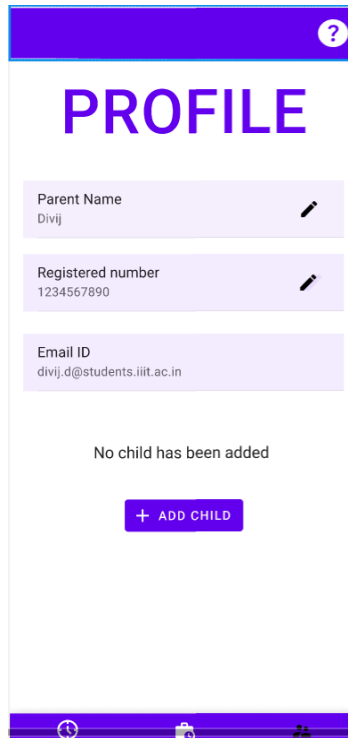
123456 |

SUBMIT

Request another OTP in 5 seconds..

RESEND OTP

On the Profile page, the user can change the editable details by tapping on the associated pencil icon



The image shows a mobile application interface for a 'PROFILE' page. At the top, there is a purple header bar with a white question mark icon. Below the header, the word 'PROFILE' is displayed in large, bold, purple capital letters. Underneath, there are three light purple rectangular boxes, each containing a label and a value, with a small pencil icon to the right of each value for editing. The first box is labeled 'Parent Name' with the value 'Divij'. The second box is labeled 'Registered number' with the value '1234567890'. The third box is labeled 'Email ID' with the value 'divij.d@students.iiit.ac.in'. Below these boxes, the text 'No child has been added' is centered. Underneath this text is a purple button with a white plus sign and the text '+ ADD CHILD'. At the bottom of the screen, there is a purple navigation bar with three white icons: a clock, a house, and a person.

Parent Name  
Divij

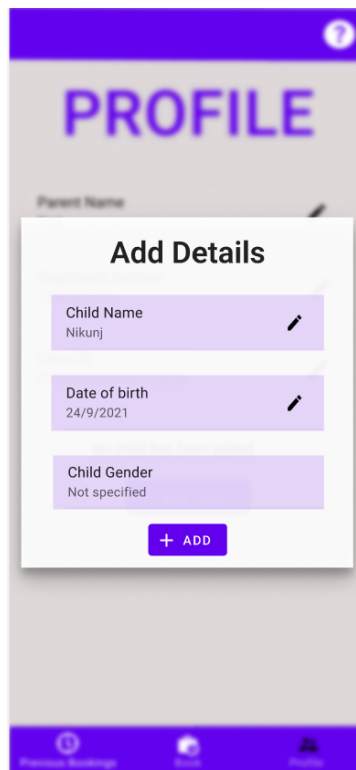
Registered number  
1234567890

Email ID  
divij.d@students.iiit.ac.in

No child has been added

+ ADD CHILD

They can also add details about their children through the button at the bottom



The image shows a mobile application interface for an 'Add Details' dialog. The dialog is a white rectangular box with a rounded top, overlaid on a blurred background of the 'PROFILE' page. The dialog has a title 'Add Details' in bold black text. Below the title, there are three light purple rectangular boxes, each containing a label and a value, with a small pencil icon to the right of each value for editing. The first box is labeled 'Child Name' with the value 'Nikunj'. The second box is labeled 'Date of birth' with the value '24/9/2021'. The third box is labeled 'Child Gender' with the value 'Not specified'. Below these boxes is a purple button with a white plus sign and the text '+ ADD'. The background page shows the 'PROFILE' title and the 'Parent Name' field.

Child Name  
Nikunj

Date of birth  
24/9/2021

Child Gender  
Not specified

+ ADD

They can provide details about their bookings through the respective buttons in the activity booking and party booking tabs

The image displays two side-by-side mobile app screens. Both screens have a purple header with a white question mark icon. The left screen is for the 'ACTIVITY' tab, and the right screen is for the 'PARTY' tab. Both screens have a purple bar at the bottom with three icons: a clock for 'Previous Bookings', a camera for 'Book', and a person for 'Profile'.

**Left Screen (ACTIVITY):**

- Header: ACTIVITY (selected), PARTY
- Section: SELECTED CHILDREN
  - Child 1
  - Child 2
  - + ADD CHILD
- Section: DROP OFF TIME
  - 11:30AM
  - [Edit icon]
- Section: DURATION
  - 5 hours
  - [Clock icon]
- Section: SELECT ACTIVITIES
  - Play area [✓]
  - Art and Craft [✓]
  - Storytelling [✓]
  - Origami [✓]
  - SEE CENTRES

**Right Screen (PARTY):**

- Header: ACTIVITY, PARTY (selected)
- Section: Adults
  - 7
  - [Edit icon]
- Section: Children
  - 10
  - [Edit icon]
- Section: SELECT TIMINGS
  - Start Time
    - 11:30AM
    - [Edit icon]
  - DURATION
    - 5 hours
    - [Clock icon]
  - SEE CENTRES

On the payment Tab, the user will be shown a detailed breakdown of the expenses of their booking along with taxes and convenience fees. There will then be options between multiple payment options for the user to make the payment. After a payment option is selected and finalized, they will be redirected to the associated payment gateway

## APIs

The following APIs are exposed to enable the users to interact with our system:

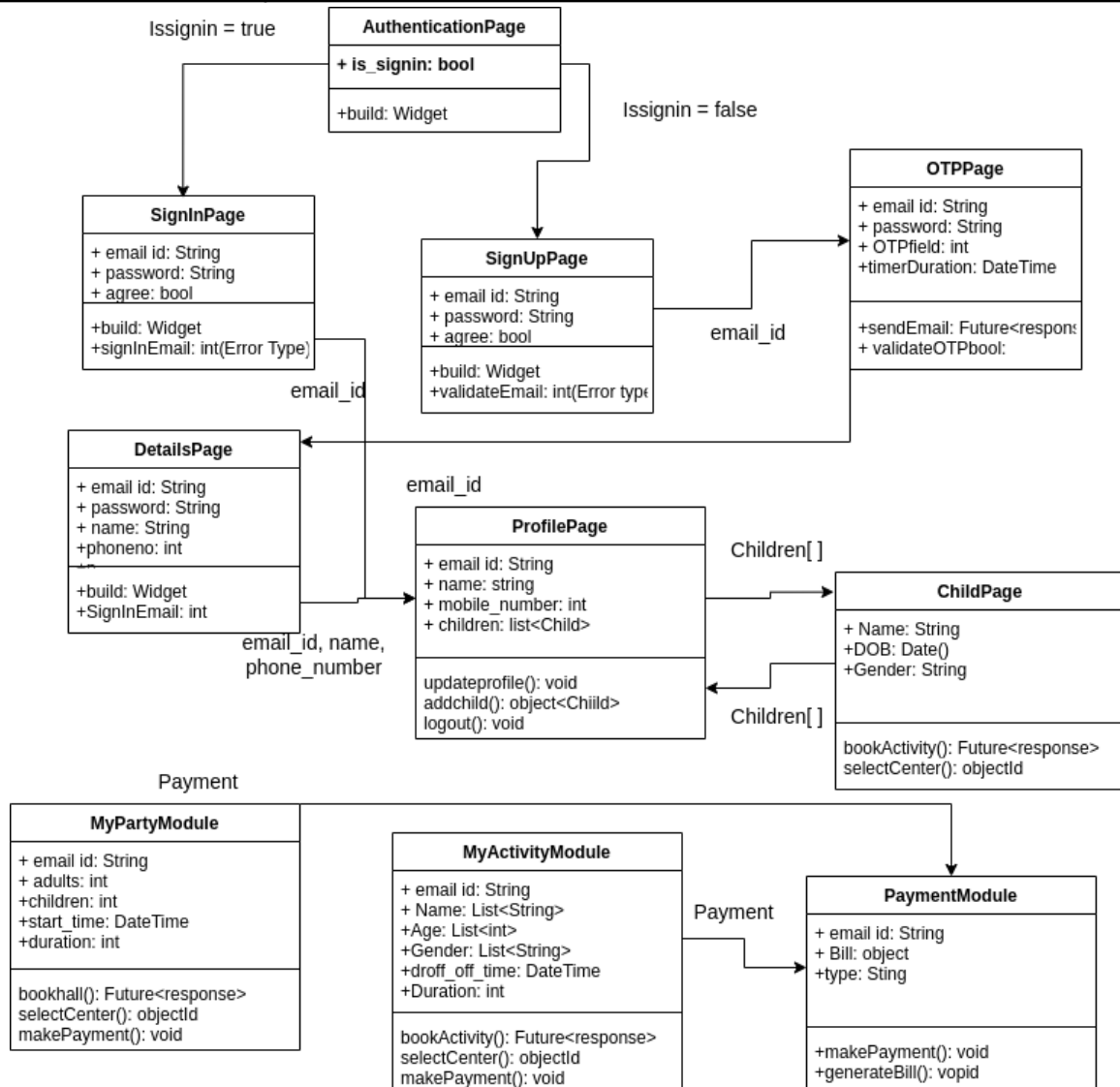
- **smtp.js API:** This is used to send the OTP to the user during the Sign Up page to confirm the email of the user.
- **Mongoose API:** This API is used to interface our server with the MongoDB database and perform the database transactions as demanded by the frontend requests. This API was used on the server.

- **geolocator:** This is a Flutter library that is used in our project to get the current location of the user and suggest nearby centers to the user during Party/Activity booking.
- **geocoding:** This Flutter library is used in our project to convert the location obtained by using geolocator to a human-readable form, so that it can be used.

## Model

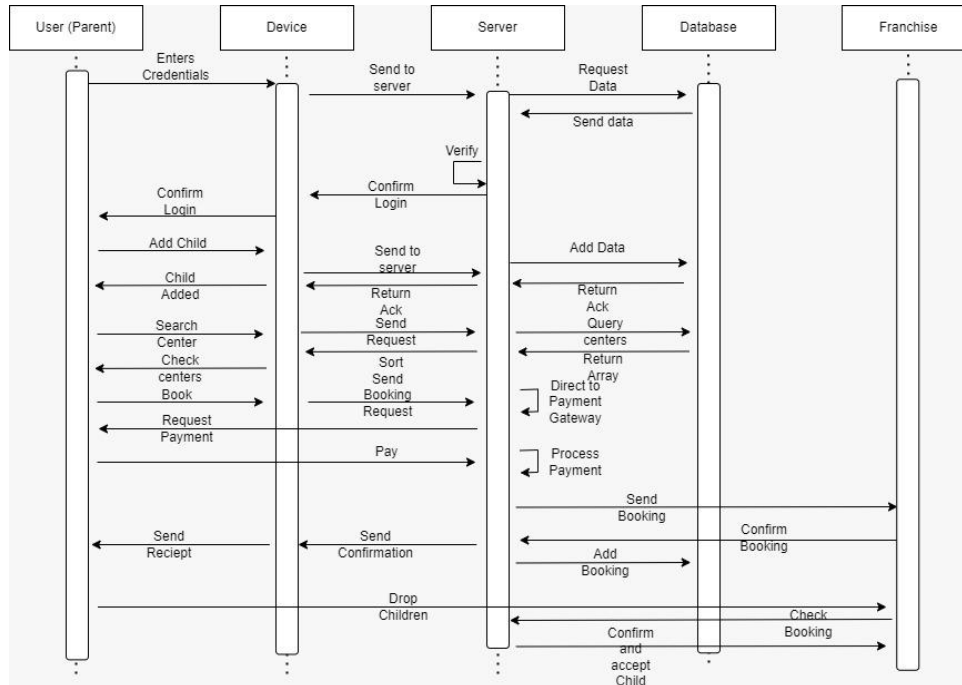
AuthenticationPage	<ul style="list-style-type: none"> <li>• Class holds the current status of authentication ie signup / signin by storing in a boolean variable.</li> <li>• It toggles between the signup and signin page by button click on buttons on button bar.</li> </ul>
SignInPage	<ul style="list-style-type: none"> <li>• It holds the email Id and password entered by the user and boolean variable showing customers agreement to terms and conditions.</li> <li>• It takes the email id of the user and password, and agreement to terms and conditions and then after clicking on login redirects the user to ProfilePage.</li> </ul>
SignUpPage	<ul style="list-style-type: none"> <li>• It holds the email Id and password entered by the user and boolean variable showing customers agreement to terms and conditions.</li> <li>• It takes the email id of the user and password, and agreement to terms and conditions and then after clicking on "Create An Account" button redirects the user to OTP Page.</li> </ul>
OTPPage	<ul style="list-style-type: none"> <li>• It takes the email ( passed by SignUp Page ) and stores the OTP required for verification and timer countdown.</li> <li>• On "Verify OTP" click it sends the OTP to the email of the user and then on successful validation redirects to PersonalDetailsPage.</li> <li>• "Request Another OTP" presents the new OTP to the account and resets the timer.</li> <li>• On timer expire it navigates back to SignUp page.</li> </ul>
PersonalDetailspage	<ul style="list-style-type: none"> <li>• It holds the name and phone number entered by the user and email id and password from OTPPage.</li> <li>• It takes the name and phone number input of the user, then after clicking on "SignUp" redirects the user to ProfilePage.</li> </ul>
MyApp	<ul style="list-style-type: none"> <li>• It stores the email, phone number, name and list of children of the logged in user.</li> <li>• It allows the user to edit the name and phone number via updateProfile() function which uses a put request at backend.</li> <li>• "Add child" button opens a static modal, and adds child information to the list of child class.</li> </ul>
AddChildPage	<ul style="list-style-type: none"> <li>• It stores the childName, Date Of Birth, gender of the child.</li> <li>• It calls an external method of MYApp class and appends the list of children.</li> <li>• "Cancel" button navigates back to MyApp Page.</li> </ul>
MyActivityModule	<ul style="list-style-type: none"> <li>• It stores the list of the added children, drop-off time of the activity, duration of activity, activity center.</li> <li>• It takes information for activity and calls bookActivity() method that stores data in the database.</li> <li>• "Confirm" button redirects to the PaymentPage.</li> </ul>

MyPartyModule	<ul style="list-style-type: none"> <li>It stores the number of adults and children participating, start time, duration and venue.</li> <li>It takes details for party and calls bookHall() method that stores data in the database which is accessed by the administrator.</li> <li>“Confirm” button redirects to the PaymentPage.</li> </ul>
PaymentPage	<ul style="list-style-type: none"> <li>Stores input email and Bill from BookingPage, stores the user preferred payment method(UPI / Net Banking)</li> <li>On “Confirm Payment” button click it redirects the user to the external webpage of concerned bank for making payments.</li> </ul>
BackedModule	<ul style="list-style-type: none"> <li>Schema for parent, child, activity, party, centers and payment page.</li> <li>It has api calls that manage interface with the backend and Flutter that fetch and update data.</li> </ul>





## Sequence Diagram(s)



## Design Rationale

Initially we had decided on using React native for coding our application, but when we were researching and testing it, we found 2 critical flaws that made us switch our project to flutter framework

1. We discovered that React native does not give a good enough performance over a wide range of devices for our application to be smooth and responsive. We found out that such performance bottlenecks are minimal in flutter
2. Due to our team not having as much development experience in creating mobile applications, we feel that it was necessary for our chosen framework to have a wider community support along with resources for debugging and testing. We believe that flutter has a better developer infrastructure and thus it was more suitable for our needs

For choosing our database and backend implementation, we were debating between using firebase or mongoDB as our central database for handling our backend API calls. Ultimately, we settled on using mongoDB for the following reasons

1. Our Team had much more extensive experience using mongoDB and creating applications for it. Thus we felt more comfortable using it as we also believed it was adequate for our purposes

2. MongoDB uses JSON Objects to send and receive data, while firebase uses HTTP API. We believed that using JSON objects would be much more convenient for our project, thus we decided to use MongoDB
3. We also discovered that using mongoDB is much more secure as compared to firebase
4. Initially, we had been leaning towards using firebase because of its ease of integration with our frontend, but we felt that the other factors outweigh this and we finally decided on using mongoDB

All throughout our app, We have decided and made the assumption that the primary identification for the user is the Email address. Thus the user is not provided an option for changing their registered email. Moreover, most of the external communication with the user is done through email interaction, for example sending a copy of the bill. We had contemplated the possibility of using the phone number for this purpose but we settled on using the email for the following reasons

1. It was simpler and easier for us to implement email identification as compared to phone number
2. We wanted to have communication with the user outside of the app(for example sending them OTP or sending a bill for the booking). We felt that such interactions would not be possible through SMS interaction and would be much more convenient in email form