**OccupEye**

*Team 2B, Cohort 2*

Yash Yadav 1006199

Sarang Nambiar 1006181

Lim Jie Han 1006246

Ryan Pey Jun Hao 1006048

Rishika Banka 1006304

Siddharth Ganesh 1006407

Esan Natraj 1006002

Github: https://github.com/Sarang-Nambiar/OccupEye

**Background**

It has come to our attention that a significant proportion of students do not reserve the meeting rooms in the hostel, and a considerable number of them are unaware of the necessity to do so. Consequently, the utilization of study rooms operates on an unofficial, first-come-first-serve basis within the student population. This informal system, however, presents a new set of challenges for the resident community. Students who wish to utilize a meeting room must rely on the chance that the nearest one is available, which is typically not the case. Consequently, the unlucky student must search through every meeting room in the block to locate an unoccupied space. Hence, the matter at hand pertains to exploring viable strategies for assessing the occupancy capacity of meeting rooms to improve the convenience of our student population**.**

**Problem statement**

The current first-come-first-serve system for meeting rooms within the hostel has resulted in a lack of awareness amongst students regarding the necessity of reserving study rooms. As a result, the occupancy of meeting rooms remains unpredictable, causing inconvenience for students who wish to use them. The objective of this project is to explore and develop a reliable and efficient method for evaluating the occupancy capacity of meeting rooms, thereby providing greater convenience for students and improving the overall management of the hostel's resources.

**Our Solution**

OccupEye - The app that can display the current crowd levels of whichever room you need.

OccupEye uses advanced crowd intelligence technology to detect the occupancy capacity of any room listed in our app, updating you with real-time information and insights. It uses the Machine Learning Package YoloV5, to help detect the number of users present in each room through the cameras present. This information is then passed to our database, and finally retrieved by our app.

Our app denotes the occupancy levels of said rooms using color indicators:

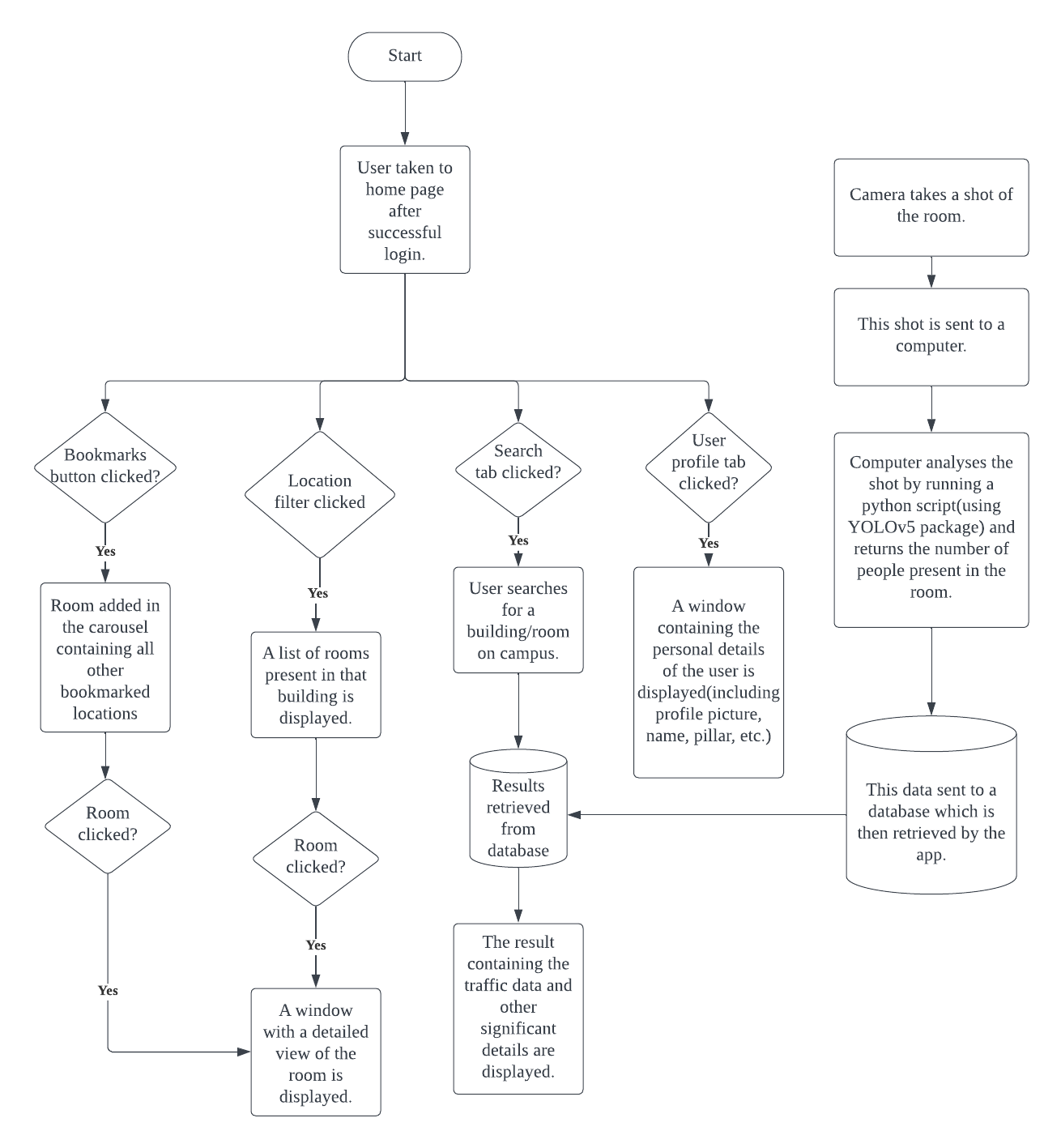
· Green – Vacant

· Amber – Relatively Occupied

· Red – Crowded

These colors are assigned according to the number of users in a given room as a percentage of the maximum occupancy of said room.

Additionally, our app has features such as search, filters, bookmarks, that help you navigate through the app and get details on your desired room faster. The login, register and forgot password pages of our app ensure user authentication and safe sign in. After this the user can access his/her bookmarks and customize his/her profile if needed.

**System Architecture Diagram**

**Design and Implementation**

**Technical Stack**

**Language :** Java

**Frontend :** Android UI

**Auth & DB :** Firebase (Auth, Realtime DB, Firestore)

**ML Integration (User Detection) :** YoloV5 (python)

**Java**

The project brief asked us to use Java as the main backend language for our app and hence we stuck to the requirements. It's also a programming language that is well structured with its OOP principles and easy to understand syntax rules, that helped us implement the functionalities of our app quite well.

**Firebase**

We had been introduced to Firebase in class and hence decided to use this as the database for our app. It is an easy to set up database that is easily integrated with the Android app, thus helping us to focus more on our app than on the database part.

*Firebase Authentication*

* We use this for login/register functionalities for our app.
* We also implemented the forgot password feature that sends a reset mail in case you forgot the password using the in-built features for firebase authentication.

*Firebase Realtime Database*

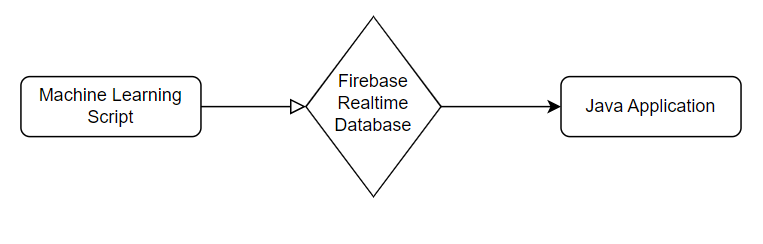
* Able to store and sync data in real-time, which is ideal for our app that updates the occupancy levels of the rooms real time as sensed by our camera.

*Firebase Storage*

* We use the firebase storage to store our user profiles, bookmarks and information. It is easy to implement and retract data from here.

**ML Integration (User Detection)**

A machine learning script is integrated as the start point in the chain of information. In our prototype, a camera provides a live feed into the script, which uses an object detection package (YoloV5) to detect the number of people present in the footage.

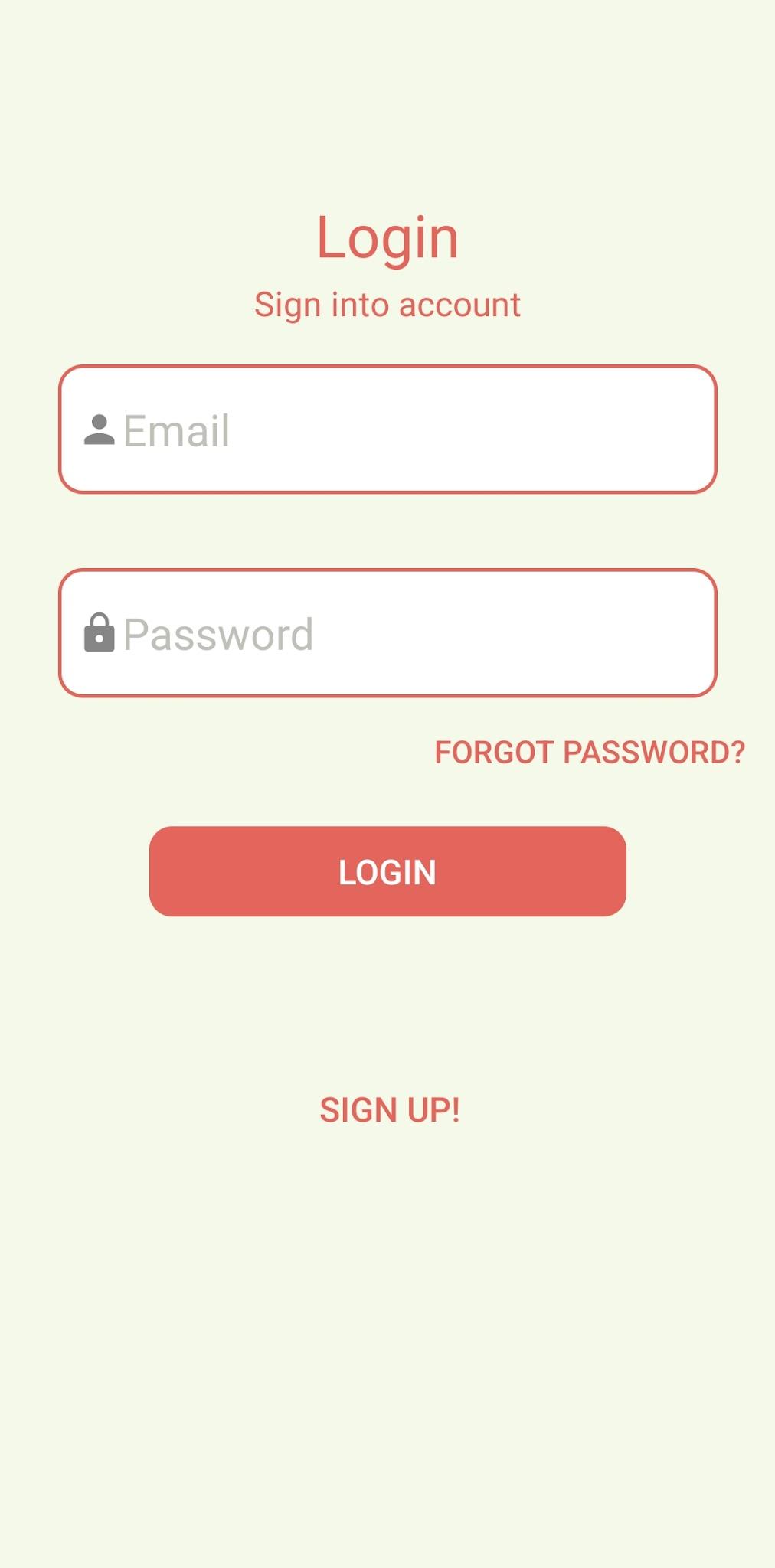


The script sends the number of users as well as the color grading of the room we want to display (calculated based on the number of people in the room) to the database. The data is then retrieved by the Java application and displayed on the app itself.

We used the OpenCV library in Python for this part of the project because we are familiar with the YoloV5 model and we found it easier to run the object detection algorithm using Python. Hence, we decided that tying the object detection data with the realtime database would simplify the workflow as we shift our focus onto developing the Android application itself in Java.

**Screenshots & Walkthrough**

**User Login Page:**

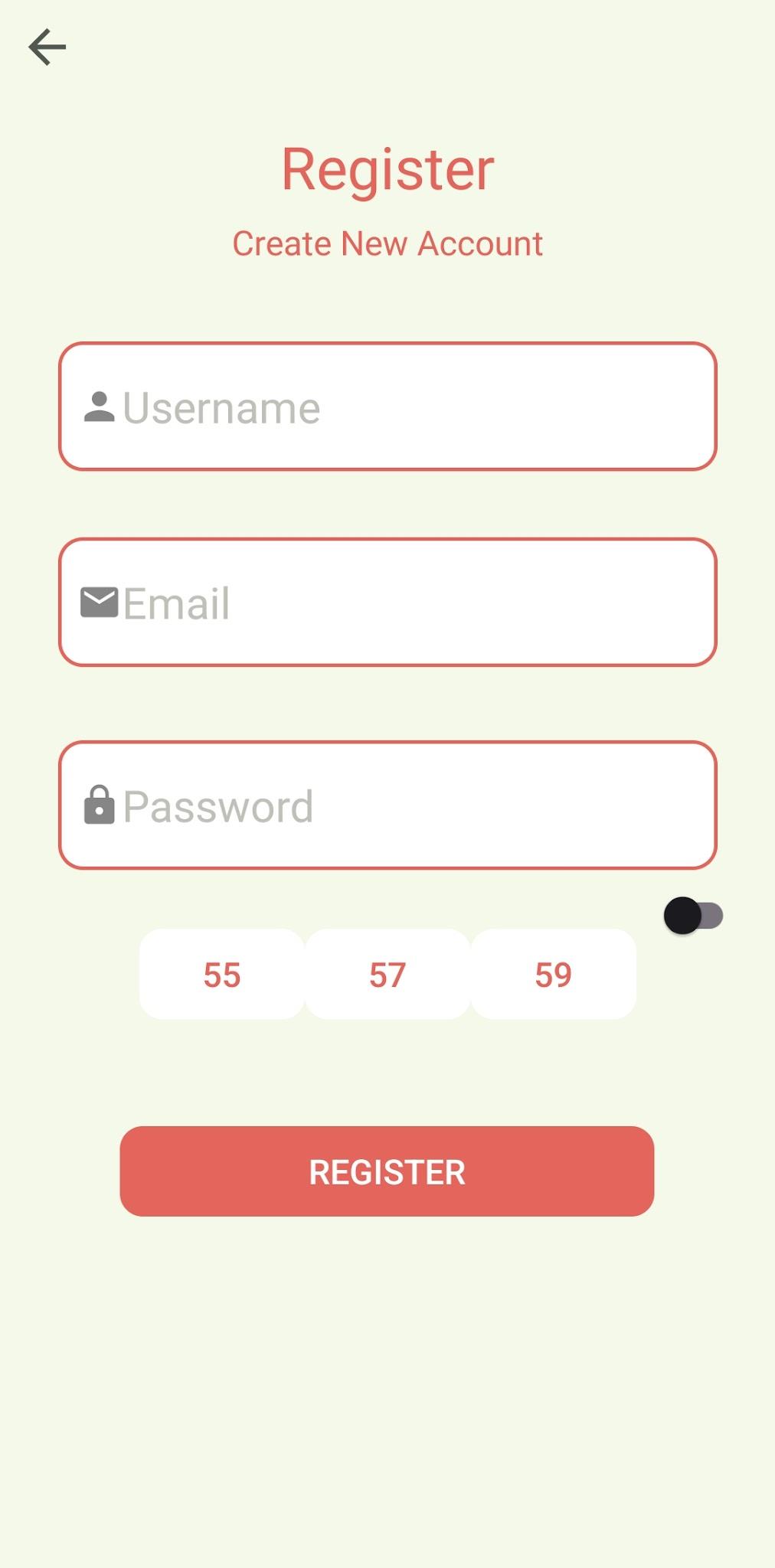


Users are greeted by this page when they first use the app or after they have logged out of the app. Users are to enter their email and password to log into the app with their registered user profile. The email and password entered are preliminarily validated in the app by checker functions that run when the user deselects the text fields. If the email and/or password does not satisfy the requirements such as minimum length and forbidden characters, a toast will appear to notify the user about the error. Users are only able to log in by pressing the Login button if the length requirements are satisfied, otherwise a toast message will tell the users that they have entered invalid entries.

Once the requirements are met and the user presses the Login button, the user’s email and password are stored in a User object instance, and the credentials are validated in the Firebase database. If there is a match, the user will be successfully logged into the app. Else, the user will be prompted to re-enter their email and password.

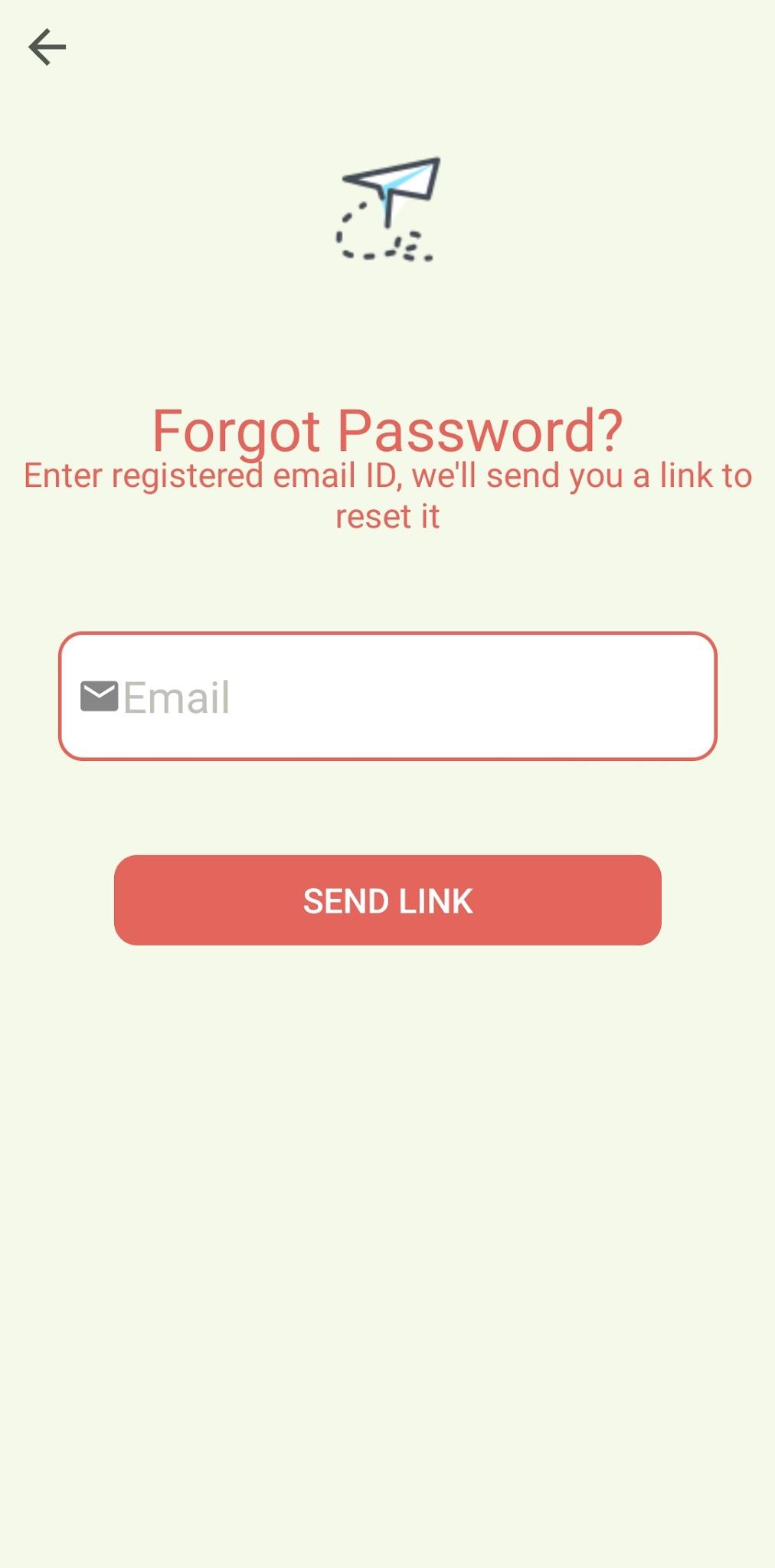
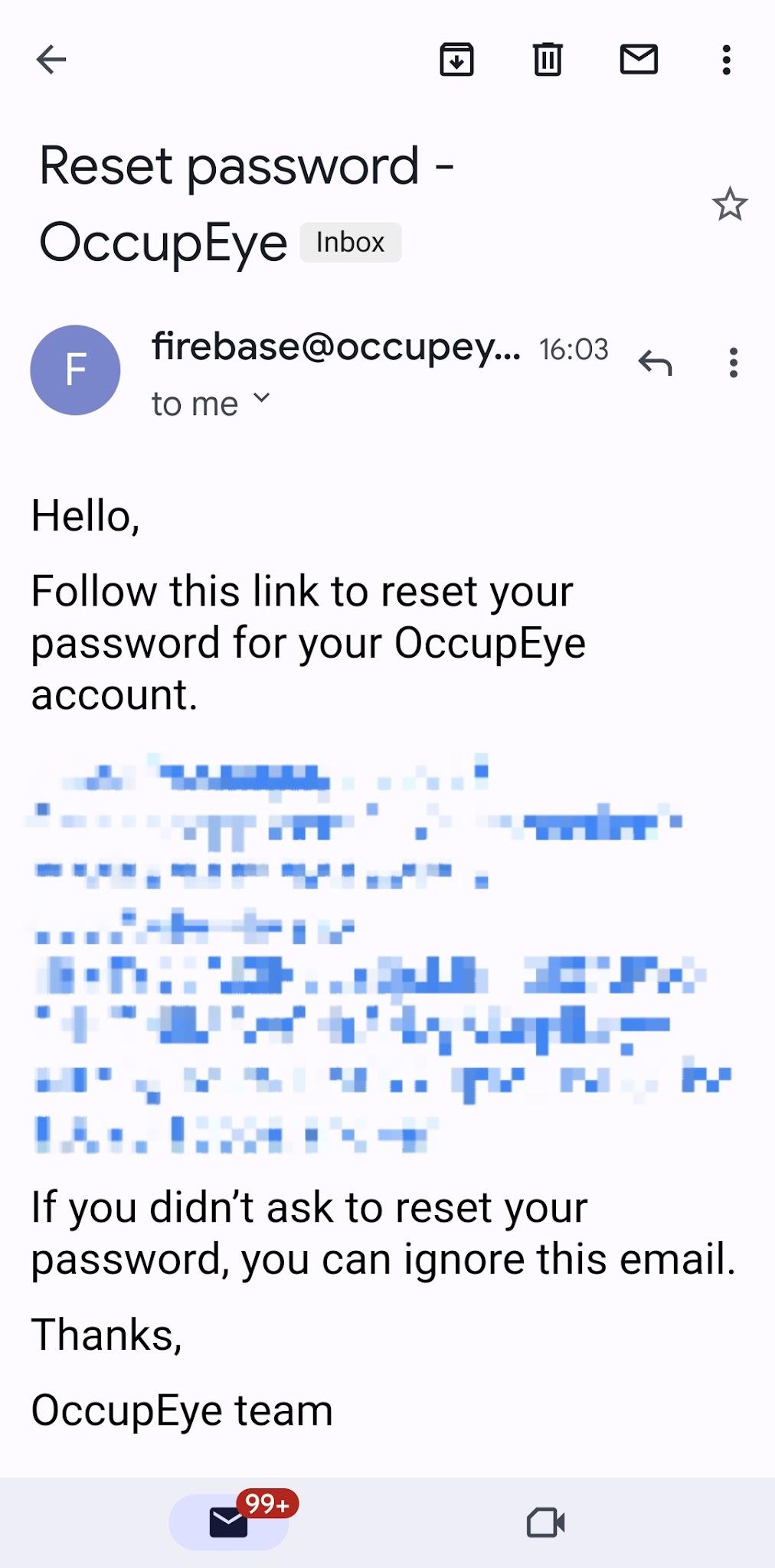
On this page, there are Forgot Password and Sign Up buttons for users to reset their password and create a new profile respectively.

**Sign Up / Register Page:**



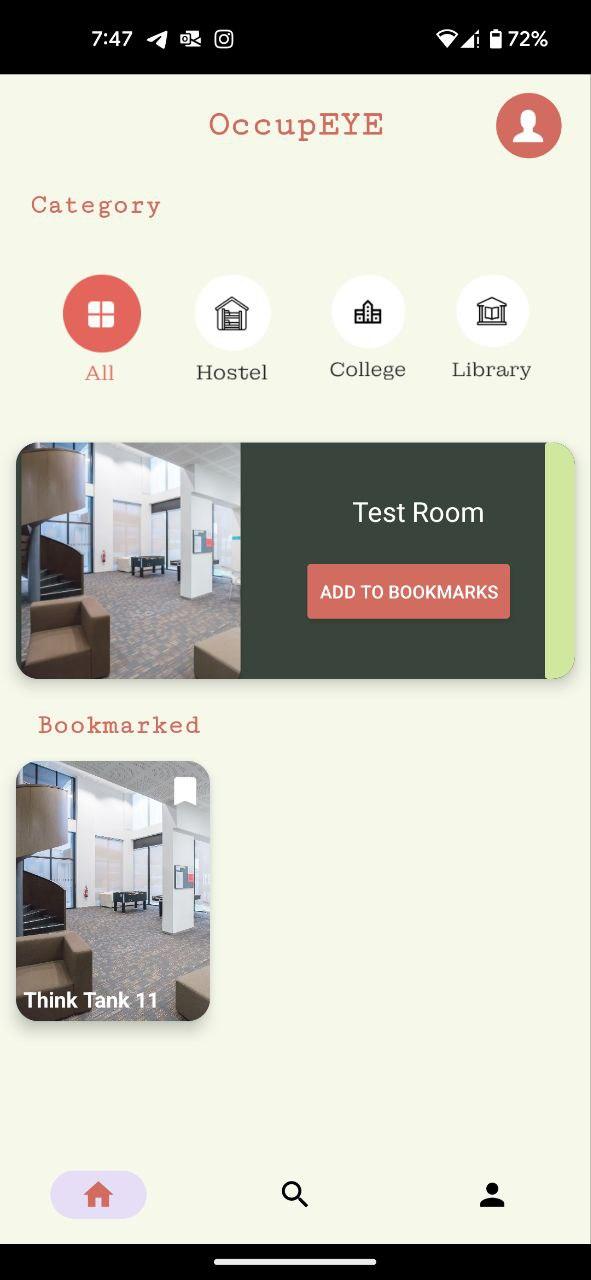
For new users, this page allows them to register a new profile so that they can store their preferences and bookmarks in the Firebase database. Users will have to enter their preferred username, email and password, as well as their hostel residence status and block number (if applicable). Once successfully registered, they will be directed to the app homepage.

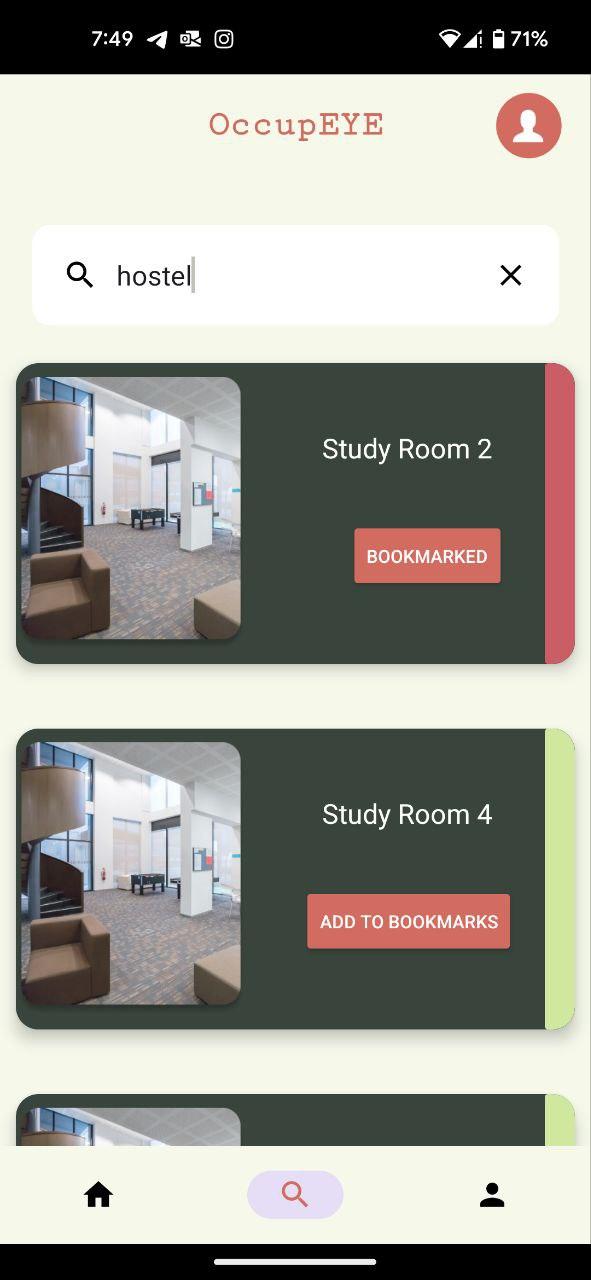
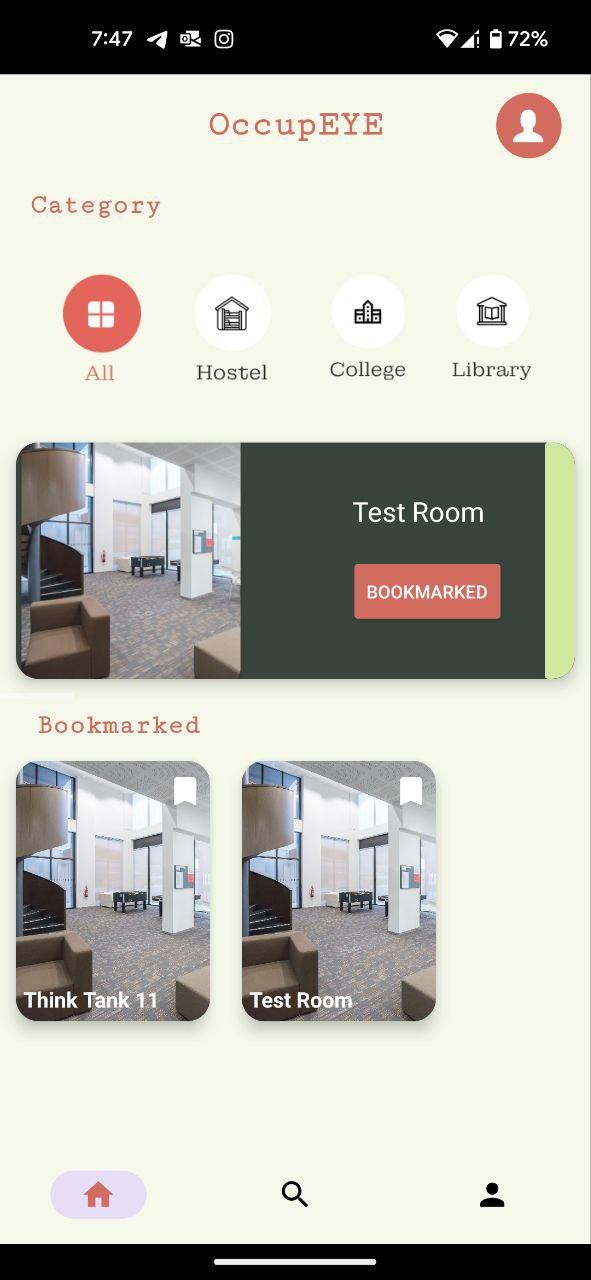
**Forgot Password Page:**

Users can reset their password on this page if they have previously registered a profile but have forgotten their password. If the entered email is in the Firebase database, a password reset email will be sent to the email address for the user to reset their password. Once completed, users will be able to login with their new password.

**Homepage:**

****

****

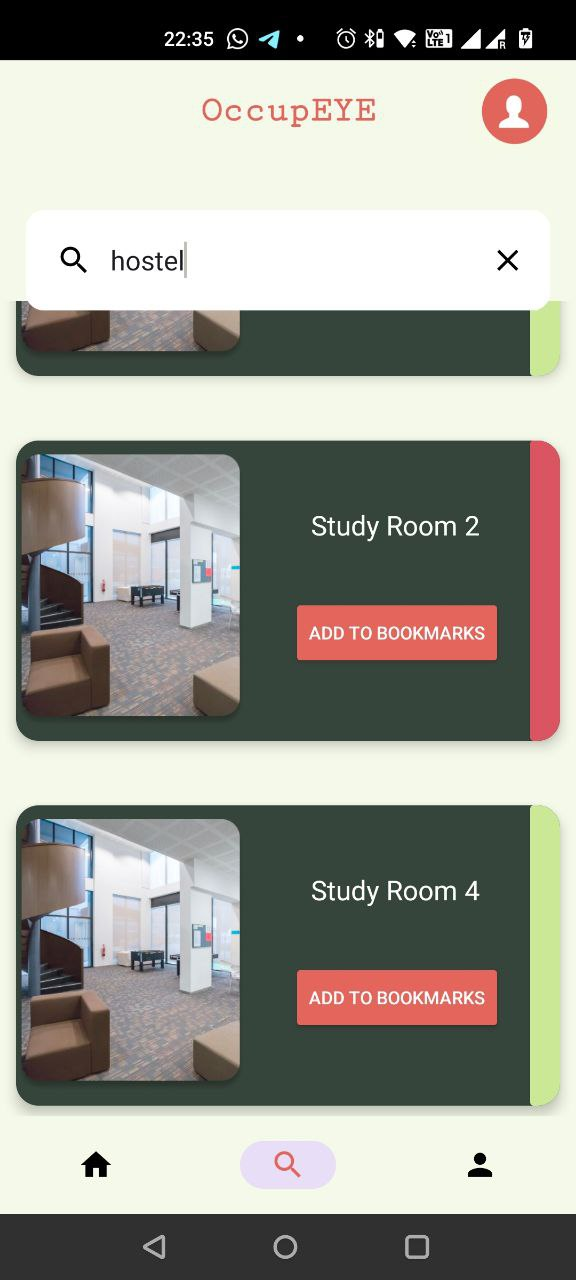
* Quicksearch feature:

The Quick Search feature provides an exceptional user experience by allowing individuals to explore rooms based on specific categories such as library, hostel, college, and others. The search results exhibit the top 10 matches of rooms available in the selected category. All rooms are presented in a recycler view that leverages the adapter to define an onClick method for the card, enabling the user to navigate effortlessly to the corresponding room page. Additionally, the "Add to Bookmarks" button feature empowers the user to save their preferred locations for future reference or quick review of the occupancy levels in that particular room. The Crowd Level Indicator is a particularly useful feature that varies based on the crowd percentage. The green indicator suggests that the room has plenty of seats available, while yellow indicates that the room is partially occupied. Finally, the red indicator warns that there are only a few seats left compared to the room capacity. Overall, this intuitive search feature is designed to enhance the user's experience and facilitate efficient navigation.

* Bookmarks feature:

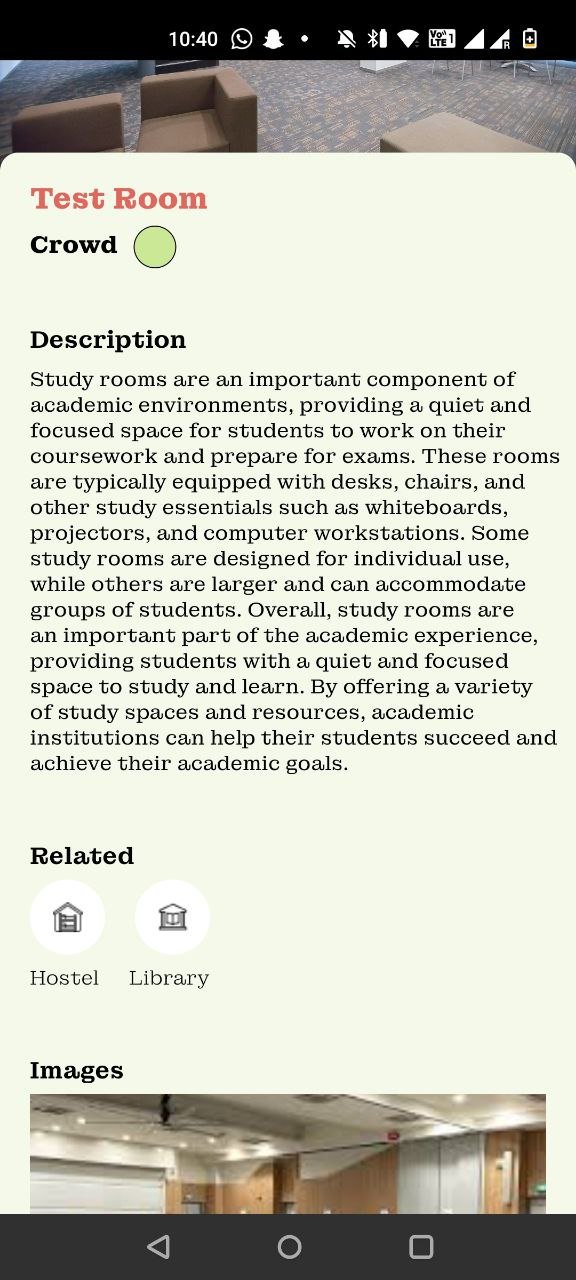
The feature known as "bookmarks" serves as a highly advantageous tool, empowering users to effortlessly preserve their preferred rooms for expedient access. This attribute can be conveniently manipulated through utilization of the bookmark icon, which can be located in the top right-hand corner of each individual card. The bookmarks are uniquely associated with each user's account, thus facilitating prompt and uncomplicated accessibility should they ever transition to an alternative Android device.The core objective of the bookmarks feature is to offer users a seamless experience in identifying the occupancy status of their favorite study spots or meeting rooms. By allowing users to easily mark and access their preferred rooms, this feature aims to streamline the overall user experience, rendering it more intuitive and efficient.

**Search Page:**



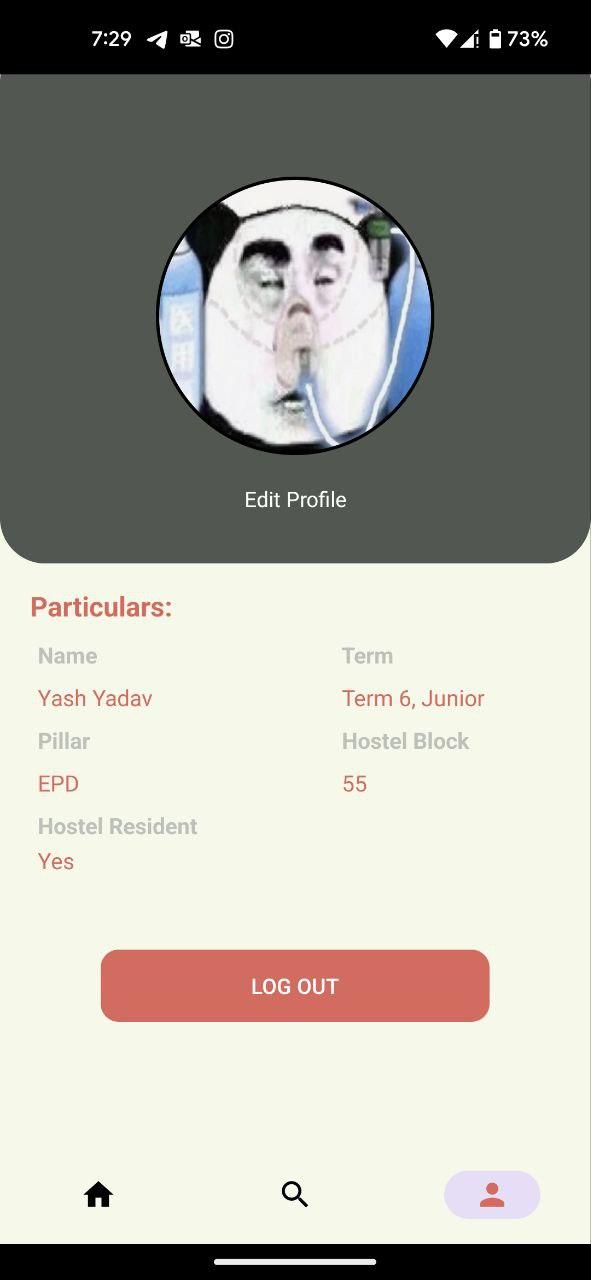
The search page allows users to find the rooms that they want specifically, giving them an option to go directly to the page that they want. The search bar also takes in generic locations such as hostel or library, in case the user needs to filter through places.

**Room Page:**



The room page gives you a detailed description of the room you are looking for, the color indicator just like the ones on the home page indicate the crowd levels of the room, as retrieved from the database. Each room also has a description and images in the form of a slider to give the user a better idea of what the room is like. You can navigate back to the home page using the buttons under the related section to view more rooms that match the current room you are looking at so that you can get the best possible room available that matches your needs.

**User Profile Page:**

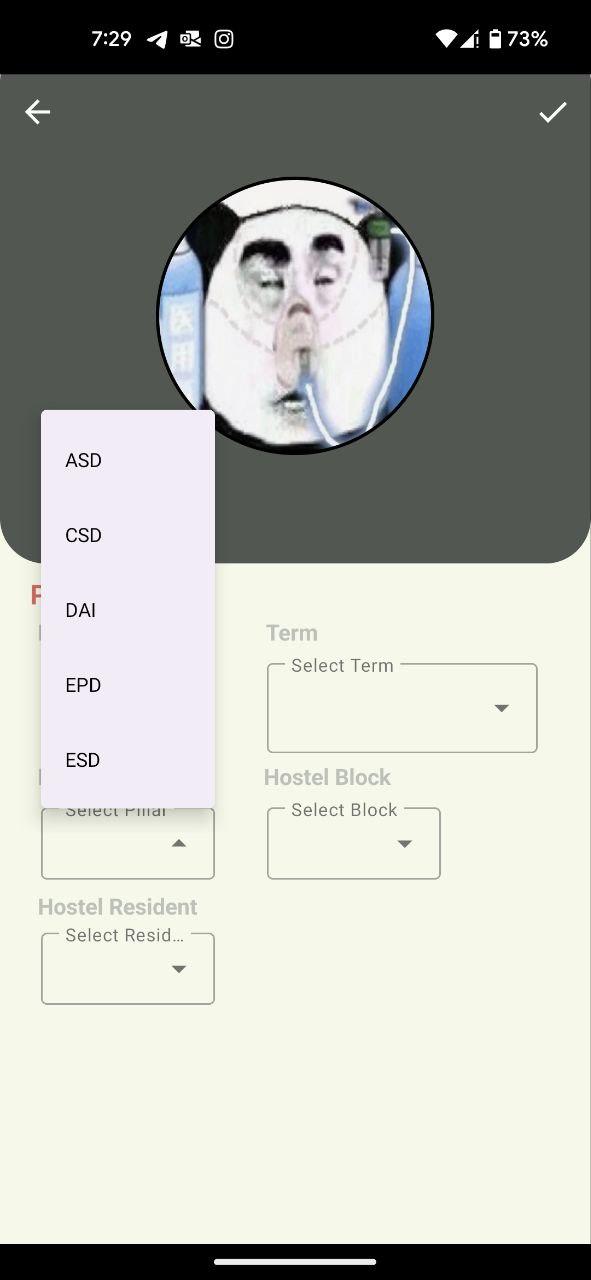
****

If the user profile tab is clicked on the bottom navigation bar, the user is navigated to the user profile page which is designed to give the user an overview of their account within the app. The user's page would contain the following information:

1. Profile picture: A visual representation of the user, typically a photograph or an avatar.
2. Name: The name of the student, which could include their first and last name, or just their first name.
3. Term: The term the student is currently in, such as the first semester, second semester, or summer term.
4. Pillar: The academic program the student is enrolled in, which could be a particular area of study or a major.
5. Block: The specific block or section of the campus where the student is living.
6. Hostel residency status: Information about whether the student is a resident of the campus hostel or not.

All of this information would be displayed in a clear and organized manner, making it easy for other users to quickly get a sense of who the student is and what their current status is at the institution. Furthermore, a "log out" button is provided for users who wish to conclude their session and sign out, or for those who prefer to access the platform using an alternate account.

**Edit Profile Page:**

****

If the “Edit Profile” button is clicked, the user is redirected to another activity where they are allowed to modify their personal information such as name, term, profile picture, etc.The purpose of this page is to give the user control over their profile and allow them to update their information as needed. The edit profile page typically displays the user's current information in text fields or dropdown menus, which the user can modify as desired. Once the user has made the desired changes, they typically have the option to save their updates or cancel and return to their user page without making any changes.

**Design Patterns**

We use the **MVC Design Principle** when we code the project. We wanted to keep our UI, View and Controller classes separate so that during bug fixes, it is easier for us to identify the issue and we can fix our code easily. You can see in our code how each java class in our code specifically focuses on only one functionality, be it the recycler view adapter, edit user, otp, login, register, bookmark, or any other class, every class has only one function that it carries out.

**Work Allocation**

We divided the work such that every group member was responsible for one page, starting with the UI, the xml code to replicate the UI, as well as the java code for that page so that each one gets some experience working in all parts of the project, and gets to learn equally.

Yash Yadav - Home Page

Sarang Nambiar - Users Page

Rishika Banka - Room Page

Ryan Pey Jun Hao + Lim Jie Han - ML Integration + Search Page

Esan Natraj + Siddharth Ganesh - Login-Registration-ForgotPassword + Firebase Setup

**Future Improvements**

The current version of the app enables individual users to easily bookmark their preferred meeting rooms and check their occupancy status. However, to expand this functionality, an innovative "Groups" feature can be implemented. This feature allows users to create groups with their regular study partners or project members, and the app can suggest meeting rooms that can comfortably accommodate the entire group.

Furthermore, an upgraded occupancy level indicator could be added to the app. Instead of displaying colors to indicate the occupancy level of a room, the app could show an estimated number of people currently occupying the room. This number being an estimate, derived from the data produced by the machine learning model, ensures that there are no privacy concerns related to the data collected.

To enhance the user's experience further, the app can also offer notification alerts that keep users informed about changes in the occupancy levels of bookmarked rooms. These alerts can notify the user when a room that was previously crowded becomes available. Users can set a preferred occupancy limit for their bookmarked rooms, ensuring that the app only recommends rooms with an occupancy level below the specified limit. For example, if a user sets a limit of 4 people, the app will only display rooms that can accommodate up to 3 people or fewer, as a room with 4 people would be deemed crowded for them.

**Conclusion**

In conclusion, this app represents a major step forward in improving the quality of life for students on campus. By reducing the challenges faced in finding an empty meeting room, students will be able to focus more on their studies and group projects with less time being lost in searching for viable locations. Additionally, by promoting the efficient use of University resources, we can help create a more sustainable and environmentally-friendly campus.

We believe that this app has the potential to make a significant impact on the student community, providing a valuable service that can help streamline the process of finding meeting rooms on campus. With its intuitive user interface, powerful features, and advanced machine learning algorithms, this app represents the cutting edge of innovation in the field of campus management and crowd intelligence.

We are excited to see how this app will evolve and improve over time, and we look forward to continuing to work with students, faculty, and staff to create a more vibrant and dynamic campus environment. Thank you for joining us on this journey, and we hope that this app will serve as a valuable tool for years to come.

**Citations**

1. Firebase Documentation

[Firebase Documentation (google.com)](https://firebase.google.com/docs/)

1. Android Documentation

<https://developer.android.com/docs>

1. XML Documentation

<https://www.w3schools.com/xml/xml_syntax.asp>