

# **HOSTEL COMPLY MOBILE APPLICATION**

*A Major project report submitted to*

**Rajiv Gandhi University of Knowledge Technologies**

**SRIKAKULAM**

**In partial fulfillment of the requirements for the**

**Award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**4<sup>th</sup> year B. Tech 2<sup>nd</sup> semester**

**B.Lakshmi - S180216**

**T.Shyam - S180222**

**R.Lakshmisri - S181016**

**P.Naga Vinay -S180580**

**Under the Esteemed Guidance of**

**Mr.Ch.Satish Kumar,  
Assistant Professor.**



**Rajiv Gandhi University of Knowledge Technologies -IIT SRIKAKULAM**



## **CERTIFICATE**

This is to certify that the report entitled “**Hostel Comply Mobile Application**” was submitted by B.Lakshmi, bearing ID. No. S180487, T.Shyam, bearing ID. No. S180222, R.LakshmiSri bearing ID.No. S181016, P.Naga Vinay bearing ID. No. S180580 in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science is a Bonafede work carried out by them under my supervision and guidance. The report has not been submitted previously in part or in full to this or any other University or Institution to award any degree or diploma.

**Mr.Ch,SatishKumar,**

Project Guide,  
Department of CSE,  
RGUKT, SRIKAKULAM

**Ms. Ch.Lakshmi Bala,**

Head of the Department,  
Department of CSE,  
RGUKT, SRIKAKULAM

## DECLARATION

We declared that this thesis work titled “**Hostel Comply Mobile Application**”, by using Java modules and deployed on web during the year 2023-24 in partial fulfilment of the requirements for the Major Project in Computer Science and Engineering. Submitted by us under the guidance and supervision of CH.**Satish Kumar** is a bonfire work. We also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma. Furthermore, the technical details furnished in various chapters of this thesis are purely relevant to the above project and there is no deviation from the theoretical point of view for design, development and implementation.

Date : 11-03-2024,

**B.Lakshmi -S180487**

Place: Nuzvid

**T.Shyam - S180222**

**R.Lakshmi Sri -S1801016**

**P.Naga Vinay -S180580**

## ACKNOWLEDGEMENT

We would like to articulate my profound gratitude and indebtedness to my project guide Mr .Ch.Satish Kumar, Assistant Professor who has always been a constant motivation and guiding factor throughout the project time. It has been a great pleasure for us to get an opportunity to work under his guidance and complete the thesis work successfully.

We wish to extend my sincere thanks to our head of the department Ch Lakshmi Bala madam, for her constant encouragement throughout the project.

We are also grateful to other members of the department without their support my work would have been carried out so successfully.

We thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work.

Project Associate

<b>B.Lakshmi</b>	<b>S180487</b>
<b>T.Shyam</b>	<b>S180222</b>
<b>R.LakshmiSri</b>	<b>S181016</b>
<b>P.Naga Vinay</b>	<b>S180580</b>

## ABSTRACT

HostelComply is a mobile app facilitating complaint resolution in university hostels. It caters to three key roles: students, electricians, and plumbers. Students can easily report issues, specifying type and location. Electricians and plumbers receive real-time assignments and update complaint statuses upon resolution. The app boasts an intuitive interface for users and administrators alike. Admin features include user management and analytics. HostelComply aims to improve communication, reduce resolution times, and foster transparency. It offers interactive hostel layouts for efficient navigation. Real-time notifications ensure timely updates for all parties involved. The app emphasizes user feedback for continuous improvement.

**Keywords:** Mobile application, hostel layout, real-time notifications

## Table of Contents

<b>Title.....</b>	<b>i</b>	
<b>Certificate.....</b>		<b>ii</b>
<b>Declaration.....</b>		<b>iii</b>
<b>Acknowledgements.....</b>		<b>iv</b>
<b>Abstract.....</b>		<b>v</b>
<b>1.INTRODUCTION</b>		
1.1 Introduction		7
1.2 Motivation		7
1.3 Problem Statement		7
1.4 Objectives		7
1.5 Goal		8
1.6 Scope		8
1.7 Application		8
1.8 Limitations		9
<b>2. LITERATURE SURVEY</b>		
2.1 Study		10
2.2 Summary		11
<b>3. EXISTING SYSTEM</b>		
3.1 Existing System		12
3.2 Disadvantages		13
<b>4. PROPOSED SYSTEM</b>		
4.1 Proposed System		13
4.2 Advantages		13
4.3 System Requirements		14
<b>5. METHODS</b>		
5.1 Methods		10
<b>6. EXPERIMENT RESULTS</b>		11
<b>7.SOURCECODE</b>		33-40
<b>8.CONCLUSION</b>		40
<b>9.REFERENCES</b>		40

---

# CHAPTER-1

## INTRODUCTION

### 1.1 Introduction

Introducing HostelComply: a mobile app revolutionizing complaint resolution in university hostels. It offers seamless reporting for students and real-time assignments for electricians and plumbers. With interactive hostel layouts and efficient admin tools, it ensures swift issue resolution. HostelComply prioritizes user feedback and transparency, enhancing communication and fostering continuous improvement. Experience a streamlined and user-centric approach to hostel complaint management with HostelComply.

### 1.2 Motivation

The motivation for HostelComply arises from the inefficiencies of traditional hostel complaint systems, prompting a need for streamlined processes. Leveraging mobile technology, our aim is to provide students with an intuitive platform for prompt issue reporting. By offering real-time assignments and comprehensive hostel layouts, we empower service providers to resolve complaints efficiently. HostelComply is driven by a commitment to transparency and accountability, with a focus on continuous improvement through user feedback. Our ultimate goal is to revolutionize hostel complaint management, fostering a responsive and user-centric environment.

### 1.3 Problem Statement

HostelComply addresses the prevalent inefficiencies and communication barriers in university hostel complaint management systems. Current methods often result in delays in issue reporting and resolution, leading to frustration among students and service providers alike. Lack of transparency and coordination further exacerbates the problem, hindering timely resolution of hostel issues. HostelComply seeks to rectify these challenges by providing a user-friendly platform that facilitates seamless communication and efficient problem resolution processes.

### 1.4 Objectives

The primary objectives of the Email Assistant project include:

- Streamline complaint resolution by creating an intuitive mobile app interface for efficient reporting and resolution of hostel issues.
- Enhance communication and transparency through real-time notifications and interactive hostel layouts to facilitate seamless interaction between users and service providers.
- Foster continuous improvement by incorporating user feedback mechanisms to iteratively enhance HostelComply's features and functionalities for an optimized user experience.

### **1.5 Goal**

The goal of the project is to revolutionize hostel complaint management by introducing HostelComply, a mobile application. It aims to streamline the reporting and resolution process, enhance communication and transparency, and ultimately deliver a user-centric and efficient system for managing hostel issues.

### **1.6 Scope**

The scope of the project includes designing and developing the HostelComply mobile application. It encompasses creating user interfaces for students, electricians, plumbers, and administrators, along with integrating real-time notification systems and interactive hostel layouts. Additionally, the project involves implementing backend functionalities for complaint submission, assignment distribution, status tracking, and analytics.

### **1.7 Applications**

Key applications of the Email Assistant include:

- In university hostels, HostelComply streamlines complaint management, enhancing student living experiences.
- Residential complexes benefit from HostelComply's efficient handling of tenant complaints, improving overall facility maintenance.
- HostelComply facilitates seamless issue reporting and resolution in corporate offices, ensuring a conducive work environment.
- The hospitality industry leverages HostelComply to enhance guest satisfaction by promptly addressing room maintenance requests and other concerns.



## **1.8 Limitations**

Limitations of the Email Assistant include:

- HostelComply's functionality may be compromised in areas with poor network coverage or outdated technology infrastructure.
- Ensuring widespread adoption of HostelComply among users may require significant promotional efforts and user training.
- HostelComply must implement robust security measures to safeguard sensitive user data and mitigate privacy risks.
- The long-term sustainability of HostelComply depends on regular maintenance and updates to address bugs and remain compatible with evolving mobile platforms.
- Integration of HostelComply with existing hostel management systems may encounter compatibility issues, necessitating thorough planning and coordination.

## **CHAPTER-2**

### **LITERATURE SURVEY**

#### **2.1 Collect Information**

The project focuses on developing a voice-based system for desktop and mobile devices tailored for blind individuals, aiming to improve accessibility to email and multimedia functions. With approximately 60% of the global blind population residing in India, the need for such technology is significant. Current advancements include text-to-Braille systems, screen magnifiers, and screen readers, though efforts are ongoing to enhance internet accessibility. Prior attempts, such as IBM's text-to-speech interface, have been made, but designing complex interfaces remains a challenge. Simplified browsing solutions and tree structures have been explored, yet shortcomings persist in navigability and usability. Specific to India, existing solutions like Shruti Drishti and Web Browser for Blind integrate with Indian language ASR and TTS systems, but lack portability to mobile devices.

#### **2.2 Study**

Embarking on the development of a hostel compliance mobile application, this project aims to streamline hostel management processes, ensuring adherence to regulations and enhancing user experience. Through intuitive design and robust features, the application promises to optimize administrative tasks, improve communication, and foster a seamless living experience for hostel residents.

#### **2.3 Benefits**

- Enhances transparency and accountability by providing real-time access to hostel rules and regulations.
- Streamlines administrative tasks, reducing manual paperwork and saving time for hostel staff.
- Improves communication between hostel management and residents through instant notifications and updates.
- Enhances safety and security by implementing features like emergency alerts and incident reporting.
- Enhances the overall hostel experience by offering convenient access to essential services and amenities through a user-friendly mobile platform.

## 2.4 Summary

The literature highlights the following key points:

- **Mobile Application:** Develop a user-friendly mobile application for hostel management.
- **Rule Compliance:** Ensure adherence to hostel rules and regulations through the app.
- **Real-Time Updates:** Provide instant notifications and updates regarding hostel policies and events.
- **Administrative Efficiency:** Streamline administrative tasks such as room allocation, fee payment, and maintenance requests.
- **Communication Platform:** Serve as a centralized communication platform for hostel management and residents.
- **Emergency Response:** Implement features for emergency alerts and quick response to safety concerns.
- **Incident Reporting:** Enable residents to report incidents or issues directly through the app.
- **Amenity Access:** Facilitate convenient access to hostel amenities and services.
- **Feedback Mechanism:** Incorporate a feedback system for residents to share their experiences and suggestions.
- **Enhanced User Experience:** Ultimately, aim to enhance the overall hostel experience by leveraging technology to improve efficiency, communication, and safety.

## CHAPTER-3

### Existed System

#### 3.1 Existing System

- Manual hostel management processes relying on paper-based documentation.
- Lack of centralized communication channels between hostel management and residents.
- Inefficient administrative procedures for tasks like room allocation and fee collection.
- Limited accessibility to hostel rules and regulations, leading to compliance issues.
- Inadequate emergency response mechanisms and incident reporting systems.

#### 3.2 Disadvantages

- Potential user resistance due to technological unfamiliarity or preference for traditional methods.
- Initial investment required for software development, infrastructure setup, and staff training.
- Ongoing maintenance necessary for bug fixes, security updates, and compatibility with new mobile platforms.
- Concerns regarding data privacy and the need for stringent compliance with regulations.
- Digital divide issues arising from unequal access to smartphones and internet connectivity.
- Possible disruptions caused by technical issues such as connectivity problems or software glitches.
- Limited functionality may leave some aspects of hostel management reliant on manual processes.
- Security risks associated with cyberattacks targeting sensitive data stored within the application.
- Staff training requirements to ensure effective utilization of the application's features.
- Dependency on technology may lead to exclusion of users with limited access to digital devices or internet.

# 4. PROPOSED SYSTEM

## 4.1 Proposed System

The proposed hostel management mobile application aims to modernize and streamline traditional manual processes. Through an intuitive mobile interface, residents and management will access features like room allocation, fee payment, and maintenance requests. Real-time communication channels will facilitate instant notifications, updates, and rule enforcement, ensuring compliance with hostel regulations. An emergency response system will allow for quick reporting and coordination during safety incidents. Residents can provide feedback through the app, fostering continuous improvement of services. Access to amenities like laundry booking and meal plans will be simplified. The system will be scalable, allowing for future expansion and integration with other platforms. Additionally, data analytics tools will provide insights into user behavior and preferences. Customization options will enable tailoring features to specific hostel requirements. The application prioritizes user convenience, safety, and efficiency in hostel management.

## 4.2. Advantages

- **Efficiency:** Streamlines administrative tasks, reducing manual workload for hostel staff.
- **Convenience:** Provides residents with easy access to essential hostel services and information through their mobile devices.
- **Real-Time Communication:** Facilitates instant communication between hostel management and residents, improving responsiveness to queries and concerns.
- **Enhanced Compliance:** Ensures adherence to hostel rules and regulations through automated reminders and alerts, reducing instances of non-compliance.
- **Improved Safety:** Integrates emergency response features for swift reporting and coordination during safety incidents, enhancing overall hostel security.
- **Feedback Mechanism:** Allows residents to share feedback, enabling continuous improvement of hostel services and amenities.
- **Accessibility:** Accommodates residents with diverse needs and preferences, promoting inclusivity and accessibility.
- **Data Insights:** Enables data-driven decision-making through analytics tools, providing insights into user behavior and trends for service enhancements.

- Scalability: Designed to scale with the growth of the hostel, accommodating future expansion and integration needs.
- Competitive Edge: Sets the hostel apart by offering modern, tech-savvy amenities and services, attracting residents seeking convenience and efficiency.

### 4.3 System Requirements

- The proposed system requires the following components and resources:
  - Internet Connectivity: Access to email servers via internet connection for sending and receiving
- emails.
  - Compatible Hardware: A microphone for speech input and speakers or headphones for audio
- output.
  - Python Environment: The system is implemented in Python and requires a suitable Python runtime
- environment with necessary libraries such as `smtplib`, `imaplib`, `speech\_recognition`, and `pytsx3`.

### REQUIREMENT SPECIFICATION

Here's a requirement specification outline for your hostel management mobile application project:

#### 1. User Interface:

Intuitive and user-friendly interface accessible on both iOS and Android platforms.  
 Responsive design optimized for various screen sizes and devices.  
 Clear navigation and layout for easy accessibility of features.

#### 2. Authentication and Authorization:

Secure login system with options for residents, hostel staff, and administrators.  
 Role-based access control to ensure appropriate permissions for different user types.

#### 3. Functional Features:

Room Allocation: Ability to view available rooms, select preferences, and allocate rooms based on availability and preferences.  
 Fee Payment: Secure payment gateway integration for hostel fees, with options for one-time or recurring payments.  
 Maintenance Requests: Submission and tracking of maintenance requests with status updates and notifications.

- Rule Compliance: Automated reminders and notifications for hostel rules and regulations, with enforcement mechanisms for non-compliance.
- Communication Platform: Real-time messaging system for communication between residents, hostel staff, and management.
- Emergency Response: Dedicated feature for reporting emergencies, triggering alerts, and coordinating responses.
- Feedback Mechanism: User-friendly interface for residents to provide feedback, suggestions, and complaints.
- Amenity Access: Booking and management of hostel amenities such as laundry facilities, meal plans, and common areas.
- Data Analytics: Analytics dashboard for hostel management to track user activity, monitor trends, and make data-driven decisions.

#### 4. Technical Requirements:

- Cross-platform development using frameworks like React Native or Flutter for efficiency and compatibility.
- Secure data storage and transmission protocols to protect user privacy and sensitive information.
- Integration with third-party services for functionalities such as payment processing and communication.
- Regular updates and maintenance to address bugs, security vulnerabilities, and compatibility issues.

#### 5. Compliance and Regulations:

- Compliance with data protection regulations such as GDPR or CCPA to ensure the privacy and security of user data.
- Accessibility standards compliance to ensure inclusivity and usability for users with disabilities.

#### 6. Testing and Quality Assurance:

- Comprehensive testing procedures including unit testing, integration testing, and user acceptance testing to ensure functionality and reliability.
- Feedback collection from users during beta testing phase to identify and address usability issues and feature requests.

#### 7. Documentation and Support:

- Comprehensive user documentation including tutorials, FAQs, and troubleshooting guides.
- Customer support channels for resolving user queries and technical issues.

## 8. Scalability and Future Expansion:

- Design architecture that allows for scalability to accommodate future growth and additional features.
- Regular updates and feature enhancements based on user feedback and technological advancements.

## **Below are the functional requirements for the Hostel Comply Mobile Application:**

Functional requirements specify the system's behavior and capabilities. Here are the functional requirements for your hostel management mobile application project:

### 1. User Authentication:

- Residents, hostel staff, and administrators should be able to log in securely using credentials.
- Authentication methods may include username/password, biometric authentication, or social login options.

### 2. Room Management:

- Residents should be able to view available rooms, select preferences, and request room allocation.
- Hostel staff and administrators should have access to manage room allocation and update room statuses.

### 3. Fee Management:

- Residents should be able to view fee details, make payments securely, and receive payment receipts.
- Hostel staff and administrators should have access to manage fee details, generate invoices, and track payments.

### 4. Maintenance Requests:

- Residents should be able to submit maintenance requests for issues in their rooms or common areas.
- Hostel staff and administrators should receive notifications of maintenance requests, update request statuses, and track resolution.

### 5. Rule Compliance:

- Automated reminders and notifications should be sent to residents regarding hostel rules and regulations.
- Hostel staff and administrators should have access to manage rule notifications and enforce compliance measures.

### 6. Communication Platform:

- Residents should be able to communicate with hostel staff and other residents through messaging features.
- Hostel staff and administrators should have access to manage and respond to resident inquiries and messages.



#### 7. Emergency Response:

- Residents should have access to emergency contact information and be able to report safety incidents or emergencies.
- Hostel staff and administrators should receive immediate notifications of safety incidents and be able to coordinate response efforts.

#### 8. Feedback Mechanism:

- Residents should have the ability to provide feedback, suggestions, and complaints about hostel services and facilities.
- Hostel staff and administrators should have access to review and respond to resident feedback.

#### 9. Amenity Access:

- Residents should be able to book and manage access to hostel amenities such as laundry facilities, gym, or common areas.
- Hostel staff and administrators should have access to manage amenity bookings and availability.

#### 10. Data Analytics:

- Hostel staff and administrators should have access to analytics dashboards to track user activity, monitor trends, and make data-driven decisions.
- Analytics should include metrics such as room occupancy rates, fee collection trends, and maintenance request volumes.

#### 11. Localization and Internationalization:

- The application should support multiple languages and currencies to accommodate residents from diverse backgrounds.
- User interface elements should be adaptable to different cultural norms and preferences.

These functional requirements ensure that the hostel management mobile application effectively meets the needs of both residents and hostel staff, streamlining operations and enhancing the overall hostel experience.

## **Non-functional requirements for the Email Assistant project:**

Non-functional requirements specify the qualities or characteristics that describe how the system should behave or perform. Here are the non-functional requirements for your hostel management mobile application project:

### **1. Performance:**

- The application should load quickly and respond promptly to user interactions, with minimal latency.
- Response times for critical functions such as room allocation, fee payment, and emergency reporting should be within acceptable limits.

### **2. Scalability:**

- The application should be able to handle a growing number of users, rooms, and transactions without significant degradation in performance.
- It should support concurrent access by multiple users without compromising responsiveness or stability.

### **3. Reliability:**

- The application should be highly reliable, with minimal downtime and robust error handling mechanisms.
- It should gracefully handle unexpected errors, preventing data loss or system crashes.

### **4. Security:**

- The application should implement strong encryption protocols to protect sensitive user data and financial transactions.
- User authentication and authorization mechanisms should be secure and resistant to unauthorized access or tampering.

### **5. Privacy:**

- User privacy should be ensured through compliance with data protection regulations such as GDPR or CCPA.
- Personal information should be collected and stored securely, with clear policies for data usage and sharing.

### **6. Accessibility:**

- The application should be accessible to users with disabilities, complying with accessibility standards such as WCAG (Web Content Accessibility Guidelines).
- It should support assistive technologies such as screen readers and voice commands.

### **7. Usability:**

- The user interface should be intuitive and easy to navigate, with clear instructions and feedback for user actions.

- It should follow design principles for consistency, simplicity, and visual clarity.

#### 8. Compatibility:

- The application should be compatible with a wide range of mobile devices, operating systems, and screen sizes.
- It should support popular web browsers and maintain consistent functionality across different platforms.

#### 9. Maintainability:

- The codebase should be well-organized, modular, and documented to facilitate future maintenance and updates.
- Changes and enhancements to the application should be implemented efficiently without causing regression issues.

#### 10. Interoperability:

- The application should be able to integrate seamlessly with external systems and services such as payment gateways, messaging platforms, and analytics tools.
- It should support standard protocols and APIs for interoperability with other software components.

Addressing these non-functional requirements ensures that the hostel management mobile application delivers a reliable, secure, and user-friendly experience while maintaining high performance and scalability.

### **Hardware Requirements:**

The hardware requirements for your hostel management mobile application project depend on factors such as the development environment, deployment infrastructure, and expected user base. However, here are some general hardware requirements to consider:

#### 1. Development Environment:

- **Personal Computer (PC) or Laptop:** High-performance computer for software development tasks such as coding, testing, and debugging.
- **Development Tools:** Install development environments, Integrated Development Environments (IDEs), and software libraries suitable for mobile application development (e.g., Android Studio for Android development, Xcode for iOS development).

#### 2. Testing Devices:

- **Mobile Devices:** Android and iOS smartphones and tablets for testing the application on different platforms and screen sizes.
- **Emulators and Simulators:** Software-based tools for simulating mobile devices and testing the application in various scenarios without physical devices.

#### 3. Server Infrastructure:

- **Web Servers:** Servers to host the backend components of the application, including databases, APIs, and web services.

- Cloud Services: Consider using cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) for scalable and cost-effective hosting solutions.

#### 4. Networking Equipment:

- Internet Connection: Stable and high-speed internet connection for accessing development resources, downloading updates, and testing cloud-based services.
- Local Area Network (LAN): Local network infrastructure for communication between development devices and testing environments.

#### 5. Deployment Hardware:

- Deployment Servers: Dedicated servers or cloud instances for hosting the production version of the application.
- Load Balancers (if necessary): Hardware or software-based load balancers to distribute incoming traffic across multiple servers for scalability and reliability.

#### 6. Backup and Storage:

- Storage Devices: External hard drives or network-attached storage (NAS) devices for storing backups of application data and code repositories.
- Backup Systems: Automated backup solutions to protect against data loss and ensure business continuity in case of hardware failures or disasters.

#### 7. Security Hardware:

- Firewalls: Hardware firewalls to protect the network infrastructure from unauthorized access and cyber threats.
- Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS): Hardware appliances or software solutions for monitoring and protecting against suspicious network activities.

#### 8. Miscellaneous:

- Power Backup: Uninterruptible Power Supply (UPS) units to provide backup power in case of electrical outages and prevent data loss or corruption.
- Cooling Systems: Adequate cooling systems (e.g., air conditioning, fans) to maintain optimal operating temperatures for servers and networking equipment.

These hardware requirements provide a foundation for developing, testing, deploying, and maintaining your hostel management mobile application project. Adjustments may be necessary based on specific project needs, scalability requirements, and budget constraints.

## Software Requirements:

### 1. Development Environment:

- Integrated Development Environment (IDE): Android Studio for Android app development or Xcode for iOS app development.
- Software Development Kit (SDK): Android SDK or iOS SDK for compiling, debugging, and testing mobile applications.
- Programming Languages: Java or Kotlin for Android development, and Swift for iOS development.
- Version Control System: Git for managing source code and collaboration among developers.

### 2. Backend Infrastructure:

- Web Server: Apache, Nginx, or Microsoft IIS for hosting backend services.
- Database Management System (DBMS): MySQL, PostgreSQL, MongoDB, or SQLite for storing application data.
- Server-Side Scripting Language: PHP, Python, Ruby, or Node.js for server-side logic and API development.
- Frameworks and Libraries: Express.js, Django, Flask, or Laravel for building RESTful APIs.

### 3. Frontend Development:

- JavaScript Frameworks: React.js, Angular, or Vue.js for building interactive user interfaces.
- UI Frameworks: Material Design for Android apps, UIKit or SwiftUI for iOS apps.
- CSS Preprocessors: Sass or Less for writing structured CSS code.
- Responsive Design Tools: Bootstrap or Foundation for designing mobile-friendly layouts.

### 4. Communication and Collaboration:

- Communication Tools: Slack, Microsoft Teams, or Discord for team communication and collaboration.
- Project Management Tools: Jira, Trello, Asana, or Monday.com for managing tasks, deadlines, and project progress.
- Version Control Hosting: GitHub, Bitbucket, or GitLab for hosting source code repositories and managing version control.

### 5. Deployment and Hosting:

- Cloud Services: Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) for hosting backend services and databases.
- Containerization Tools: Docker for packaging applications into containers for deployment.
- Continuous Integration and Deployment (CI/CD) Tools: Jenkins, Travis CI, or CircleCI for automating build, test, and deployment pipelines.

### 6. Security and Monitoring:

- Security Tools: SSL/TLS certificates for securing data transmission, and security scanning tools for identifying vulnerabilities.

- Logging and Monitoring Tools: Elasticsearch, Logstash, and Kibana (ELK stack), or Prometheus and Grafana for monitoring application performance and logs.

#### 7. Documentation and Collaboration:

- Documentation Tools: Markdown, reStructuredText, or AsciiDoc for writing project documentation.
- Wiki Platforms: Confluence, DokuWiki, or MediaWiki for creating and sharing project documentation.

#### 8. Customer Support and Feedback:

- Customer Support Tools: Zendesk, Freshdesk, or Intercom for providing customer support and managing user inquiries.
- Feedback Collection Tools: UserVoice, SurveyMonkey, or Typeform for gathering user feedback and feature requests.

These software requirements provide the necessary tools and platforms for developing, deploying, and maintaining your hostel management mobile application project. Adjustments may be necessary based on specific project needs, technology preferences, and scalability requirements.

# System Design

### 4.1 Design of the System

The system design for your hostel management mobile application encompasses various components to ensure a seamless and efficient user experience. At its core, the application comprises a client-side mobile interface developed using platform-specific tools such as Android Studio for Android or Xcode for iOS. This interface provides residents with access to essential features including room allocation, fee payment, maintenance requests, and communication with hostel staff. The backend infrastructure consists of web servers hosting RESTful APIs developed using frameworks like Django or Express.js. These APIs handle requests from the mobile app, interacting with a database management system (DBMS) such as MySQL or MongoDB to store and retrieve hostel-related data, including room availability, resident details, and fee records.

Additionally, the backend system manages user authentication and authorization, ensuring secure access to application functionalities based on user roles and permissions. To facilitate real-time communication between hostel management and residents, WebSocket or push notification technologies are integrated to deliver instant updates, notifications, and alerts. The application employs industry-standard security measures, including encryption protocols for data transmission, user authentication mechanisms, and secure storage of sensitive information.

Deployment of the system involves hosting backend services on cloud platforms such as Amazon Web Services (AWS) or Microsoft Azure, ensuring scalability, reliability, and cost-effectiveness. Continuous integration and deployment (CI/CD) pipelines are implemented using tools like Jenkins or CircleCI to automate the build, test, and deployment processes, facilitating rapid iterations and updates. Monitoring and logging tools such as ELK stack (Elasticsearch, Logstash, Kibana) or Prometheus and Grafana are employed to track application performance, identify issues, and ensure system stability.

The system design also encompasses considerations for scalability, maintainability, and extensibility. The architecture is designed to accommodate a growing user base and evolving requirements, with modular components that can be easily extended or modified. Documentation and collaboration tools are utilized to facilitate knowledge sharing among development teams and ensure comprehensive documentation of system architecture, APIs, and deployment procedures. Overall, the system design prioritizes reliability, security, and usability to deliver a robust and user-friendly hostel management solution.

#### 4.1.1 Class Diagram

Class diagrams usually depict the process which is also called the workflow or the relation between the entities. It includes several sections like entities, attributes and relationships. The motive of a class diagram is to depict

the classes within a model. In object-oriented software, classes have attributes(member variables), operations (member capabilities) and relation.

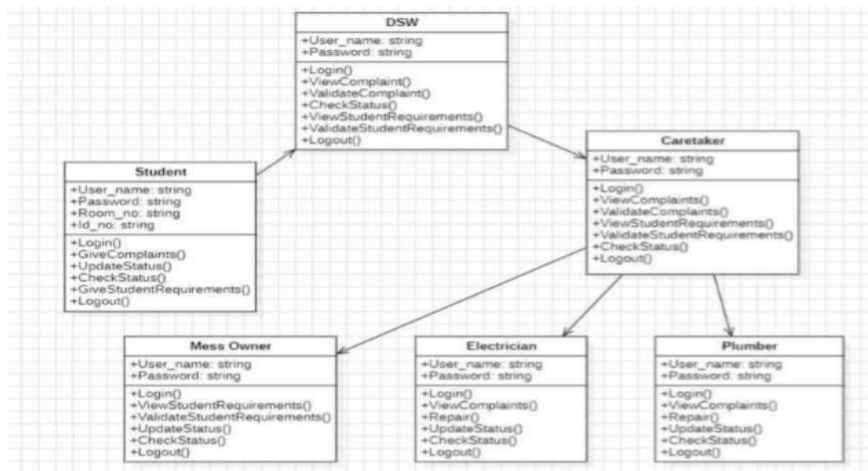
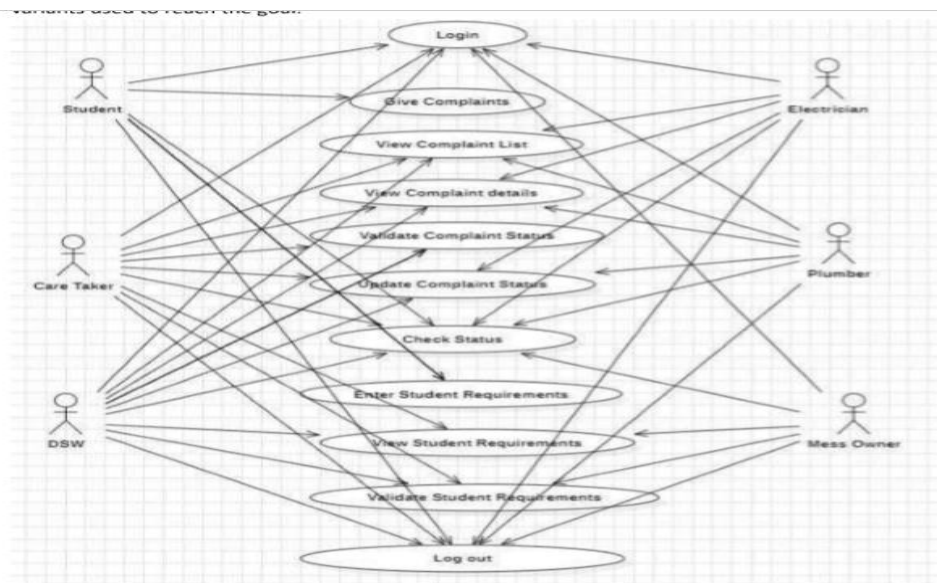


Table 3.1: Class Diagram

### 4.1.2 Use Case Diagram

A use case diagram is a type of Unified Modeling Language (UML) diagram that describes the functional requirements of a system and the actors that interact with it. The purpose of a use case diagram is to provide a high-level view of the system and its behavior from the perspective of end-users. The use case diagram consists of actors, use cases, and the relationships between them. An actor is an external entity that interacts with the system, such as a user or another system. A use case represents a set of actions or steps that a system performs in response to a request from an actor.





### 4.1.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is one variety of interaction diagram that suggests how methods operate with one another and in what order. Sequence diagrams are quite often referred to as event-hint diagrams, event situations, and timing diagrams. A sequence diagram suggests, as parallel vertical traces (lifelines), special systems or objects that are residing at the same time, and, as horizontal arrows, the messages exchanged between them, within the order the place they occur.

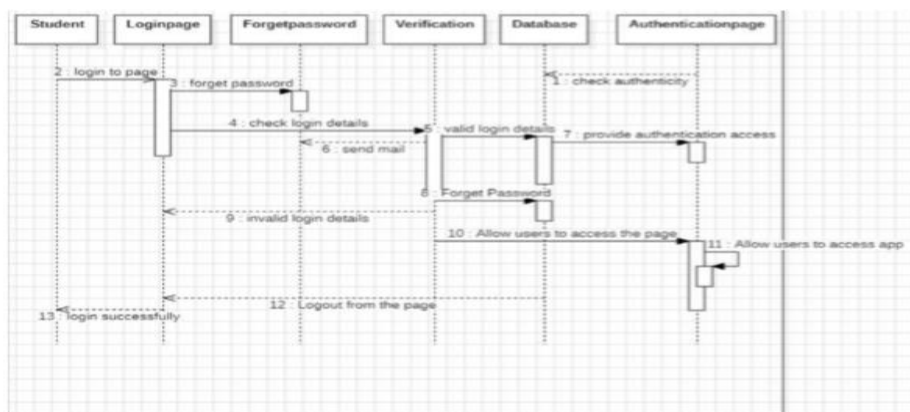


Table 3.3.1: Sequence Diagram

It is a visual representation data flows through a system. It is a popular modeling tool used in software engineering to describe the flow of data in a system, and to help identify areas where improvements can be made. A DFD consists of a set of processes, data stores, and data flows. A process represents a transformation or manipulation of data, such as calculations or data entry. Data stores represent where data is stored, such as a database or a file. Data flows represent the movement of data between processes and data stores. Symbols used in DFD

#### Description of Components:

##### 1. User Interface:

- Represents the interface through which users interact with the Email Assistant system.

Users input voice commands and receive auditory feedback through speech recognition and text-to-speech conversion.

- 

##### 2. Speech Recognition:

- Receives voice commands from the user interface and converts them into text.
- Passes the interpreted commands to the email management component for further processing.

### 3. Email Management:

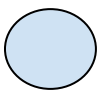
- Handles the processing of user commands related to email management tasks.
- Comprises functions for composing, sending, reading, forwarding, deleting, and searching emails

based on the interpreted user commands.

### 4. Email Server Integration:

- Interfaces with the email server (e.g., Gmail) to perform actions such as sending, receiving, and
- Executes the email-related tasks requested by the email management component and communicates
- the results back to the user interface for feedback.

This DFD provides an overview of the flow of data and processes involved in the Email Assistant system, highlighting the interaction between the user interface, speech recognition, email management, and email server integration components.



A circle represents a process.



A rectangle represents external entity



It defines a destination of the system data.



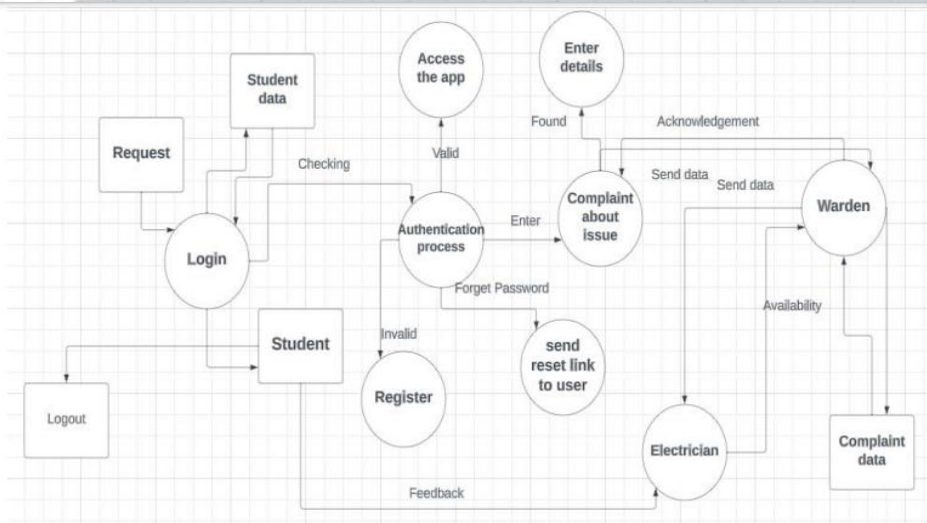
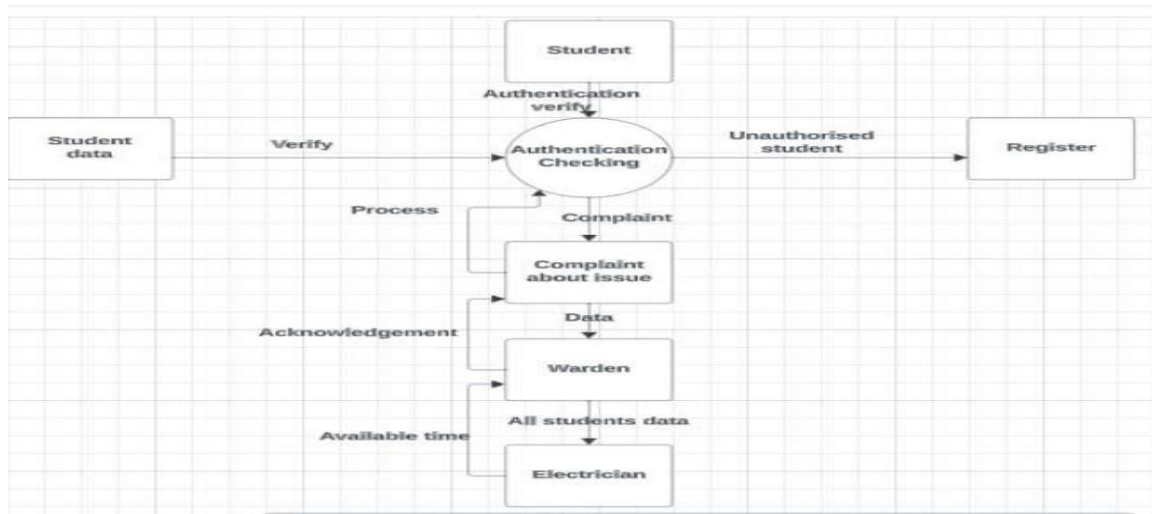
An arrow identifies dataflow.



Double line with one end closed indicates a data store.

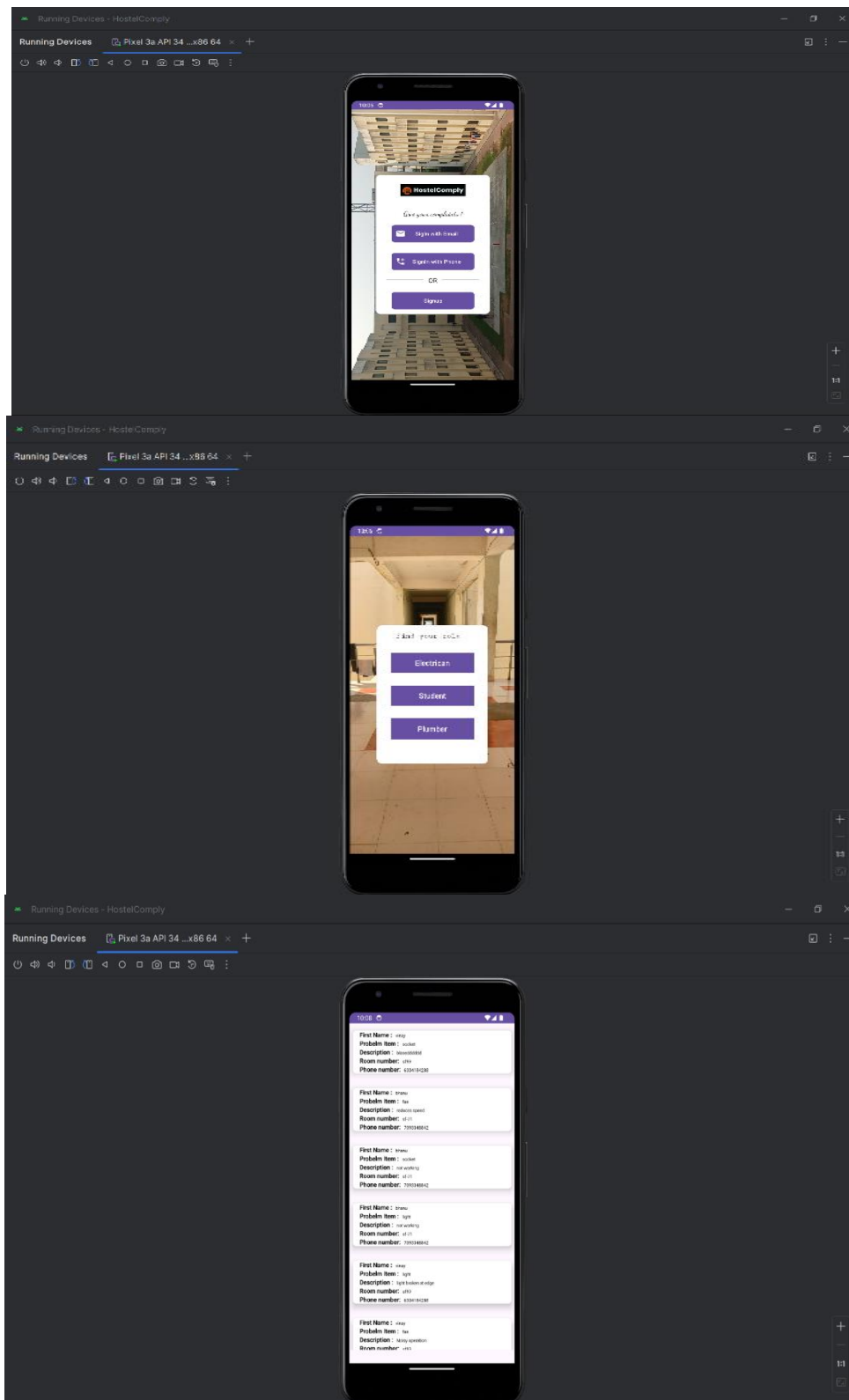


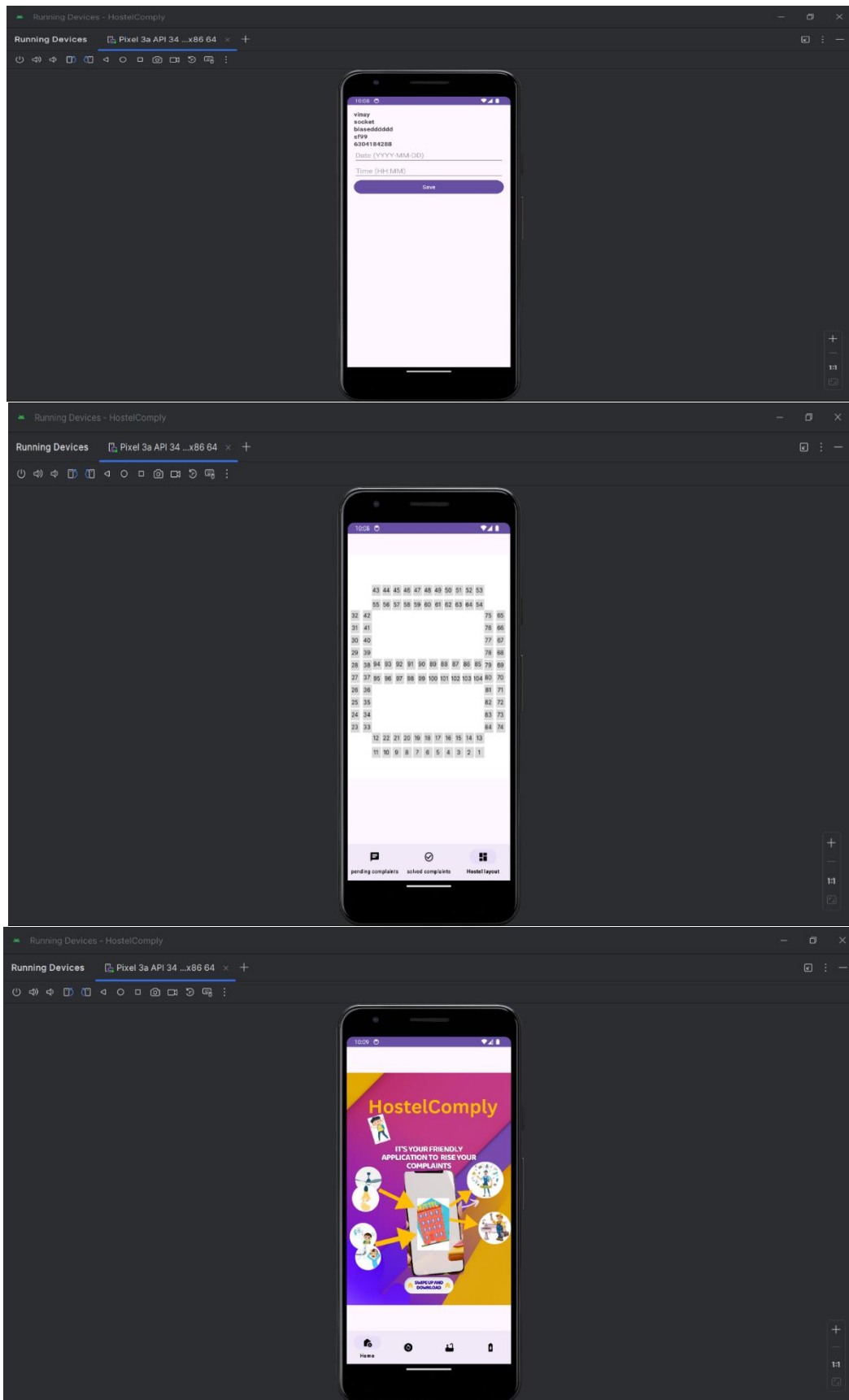
# DFD level-0:

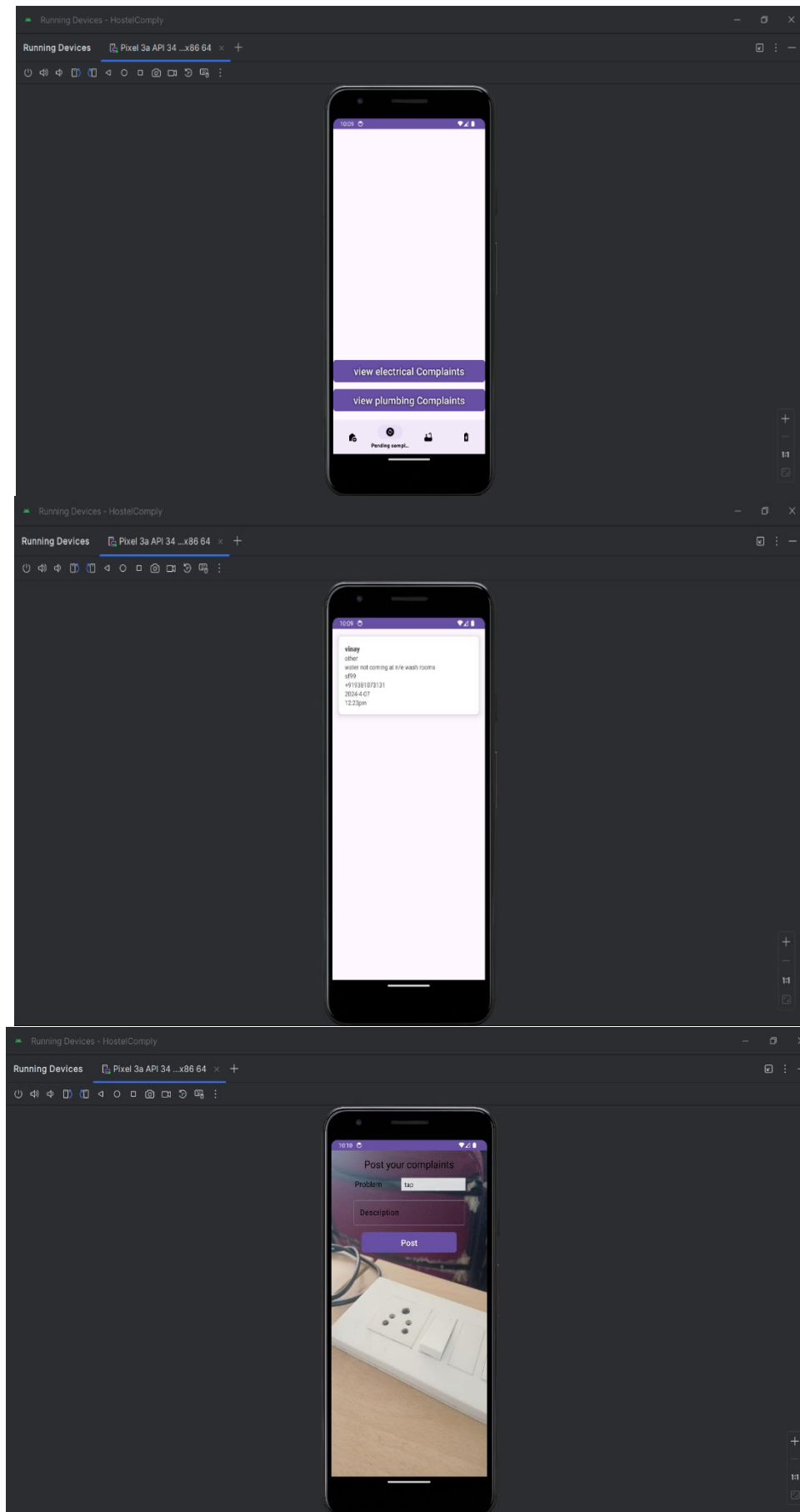


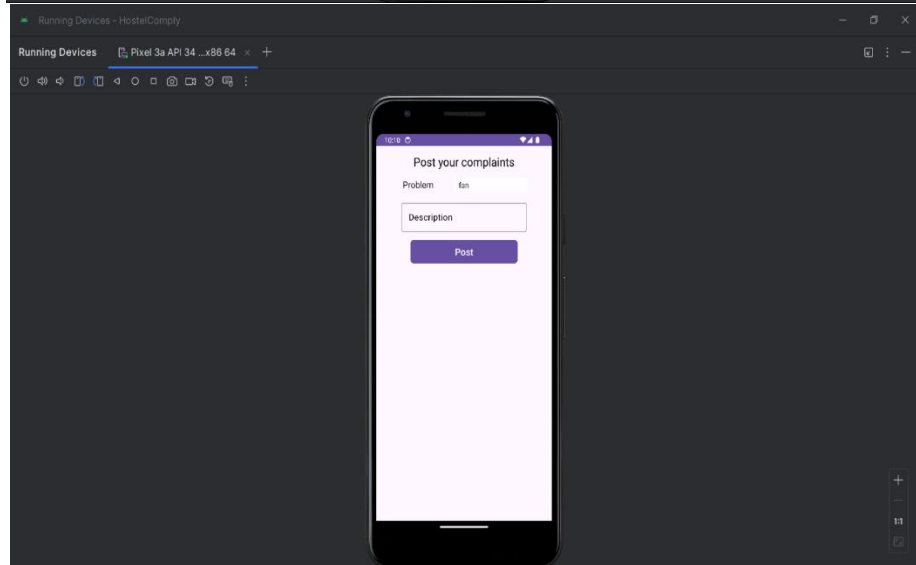
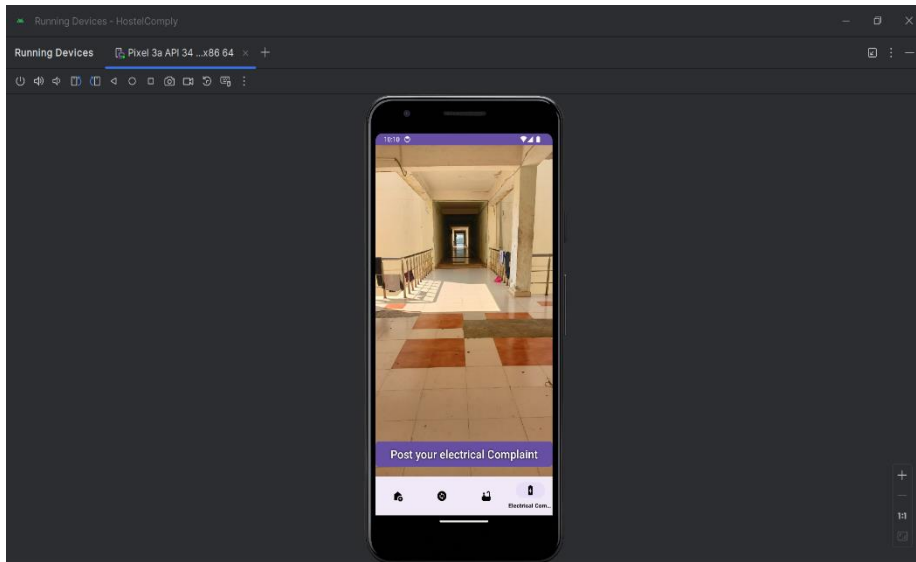
## CHAPTER-6

### EXPERIMENT RESULTS









### Source Code

#### 7.1 Unit Testing

##### 1. electricianPanel

###### **Electrical\_Hostel\_layout.java**

```
package my.hostelcomply.app.electricianPanel;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import my.hostelcomply.app.R;

public class Electrical_Hostel_layout extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        View v = inflater.inflate(R.layout.activity_electrical_hostel_layout,null);
        getActivity().setTitle("Electrician hostel layout");
        return v;
    }
}
```

###### **Electrician.java**

```
package my.hostelcomply.app.electricianPanel;

import java.util.HashMap;

public class Electrician {
    String Mobile_No,First_Name,Last_Name,EmailId>Password,Confirm_Password;
```



```

    public Electrician(String mobile_No, String first_Name, String last_Name, String emailId, String password,
String confirm_Password) {
        Mobile_No = mobile_No;
        First_Name = first_Name;
        Last_Name = last_Name;
        EmailId = emailId;
        Password = password;
        Confirm_Password = confirm_Password;
    }

    public String getMobile_No() {
        return Mobile_No;
    }

    public void setMobile_No(String mobile_No) {
        Mobile_No = mobile_No;
    }

    public String getFirst_Name() {
        return First_Name;
    }

    public void setFirst_Name(String first_Name) {
        First_Name = first_Name;
    }

    public String getLast_Name() {
        return Last_Name;
    }

    public void setLast_Name(String last_Name) {
        Last_Name = last_Name;
    }

    public String getEmailId() {
        return EmailId;
    }

    public void setEmailId(String emailId) {
        EmailId = emailId;
    }

    public String getPassword() {
        return Password;
    }

    public void setPassword(String password) {
        Password = password;
    }

```

```

    public String getConfirm_Password() {
        return Confirm_Password;
    }

    public void setConfirm_Password(String confirm_Password) {
        Confirm_Password = confirm_Password;
    }
}

```

### **ElectricianPendingComplaintsFragment.java**

```

package my.hostelcomply.app.electricianPanel;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

import my.hostelcomply.app.R;
import my.hostelcomply.app.Student;
import my.hostelcomply.app.studentPanel.StudentComplaintDetails;
import my.hostelcomply.app.studentPanel.Student_postComplaint;

public class ElectricianPendingComplaintsFragment extends Fragment{

    Button viewComplaint;

    @SuppressWarnings("MissingInflatedId")
    @Nullable

```

```

@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_electrician_pending_complaints,null);
    getActivity().setTitle("View complaints");

    viewComplaint=(Button) v.findViewById(R.id.view_complaint);
    viewComplaint.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v) {
            startActivity(new Intent(getContext(), StudelectricalComplaintDetailsList.class));
        }

    });
    return v;
}
}

```

### **ElectricianSolvedComplaintFragment.java**

```
package my.hostelcomply.app.electricianPanel;
```

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
```

```
import my.hostelcomply.app.R;
```

```
public class ElectricianSolvedComplaintFragment extends Fragment {
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {

        View v = inflater.inflate(R.layout.fragment_electrician_solved_complaints,null);
        getActivity().setTitle("Electrician solved");
        return v;
    }
}

```

### **OneComplaintDisplay.java**

```
package my.hostelcomply.app.electricianPanel;
```

```

import static android.opengl.ETC1.isValid;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.Date;

import my.hostelcomply.app.R;
import my.hostelcomply.app.Student;
import my.hostelcomply.app.studentPanel.StudentComplaintDetails;
import my.hostelcomply.app.studentPanel.Student_postComplaint;

public class OneComplaintDisplay extends AppCompatActivity {
    EditText dateEditText, timeEditText;
    Button saveButton;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference databaseReference;
    String dateStr, timeStr, status;
    StudentelectricalComplaintDetailswithdate sdw;

    FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one_complaint_display);
        dateEditText = findViewById(R.id.dateEditText);

```

```

timeEditText = findViewById(R.id.timeEditText);
saveButton = findViewById(R.id.saveButton);

firebaseDatabase = FirebaseDatabase.getInstance();
databaseReference = firebaseDatabase.getReference("StudentelectricalComplaintDetailswithdate");

// Initialize the FirebaseAuth instance
mAuth = FirebaseAuth.getInstance();

// Initialize the StudentelectricalComplaintDetailswithdate object
sdw = new StudentelectricalComplaintDetailswithdate();

// Retrieve data from intent
Bundle extras = getIntent().getExtras();
if (extras != null) {
    String firstname = extras.getString("firstname");
    String problemItem = extras.getString("problem_item");
    String description = extras.getString("description");
    String roomNumber = extras.getString("roomnumber");
    String phoneNumber = extras.getString("phonenumber");
    String studentId=extras.getString("studentId");
    String randomUid=extras.getString("randomUid");

    // Display data in TextViews
    TextView firstNameTextView = findViewById(R.id.firstNameTextView);
    TextView problemItemTextView = findViewById(R.id.problemItemTextView);
    TextView descriptionTextView = findViewById(R.id.descriptionTextView);
    TextView roomNumberTextView = findViewById(R.id.roomNumberTextView);
    TextView phoneNumberTextView = findViewById(R.id.phoneNumberTextView);

    firstNameTextView.setText(firstname);
    problemItemTextView.setText(problemItem);
    descriptionTextView.setText(description);
    roomNumberTextView.setText(roomNumber);
    phoneNumberTextView.setText(phoneNumber);

    sdw.setFirstname(firstname);
    sdw.setProblemIte(problemItem);
    sdw.setDescription(description);
    sdw.setRoomNumber(roomNumber);
    sdw.setPhoneNumber(phoneNumber);
    sdw.setStudentId(studentId);
    sdw.setRandomUid(randomUid);

}

status = "du";

```

```

saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dateStr = dateEditText.getText().toString();
        timeStr = timeEditText.getText().toString();

        if (TextUtils.isEmpty(dateStr) ||TextUtils.isEmpty(timeStr)) {
            Toast.makeText(OneComplaintDisplay.this, "Please enter date and time",
Toast.LENGTH_SHORT).show();
        } else {
            saveComplaint();
        }
    }
});
}

private void saveComplaint() {
    // Get the current user
    FirebaseUser currentUser = mAuth.getCurrentUser();

    if (currentUser != null) {

        String electricianPhoneNumber = currentUser.getPhoneNumber();
        String key = databaseReference.push().getKey();

        // Create a new complaint object
        StudentelectricalComplaintDetailswithdate complaintwd = new
StudentelectricalComplaintDetailswithdate(dateStr, timeStr, sdw.getFirstname(), sdw.getProblemIte(),
sdw.getDescription(), sdw.getRoomNumber(), sdw.getPhoneNumber(), status, sdw.getStudentId(),
electricianPhoneNumber,key,sdw.getRandomUid());

        // Generate a unique key for the complaint

        if (key != null) {
            // Save the complaint to Firebase
            databaseReference.child(key).setValue(complaintwd)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(OneComplaintDisplay.this, "Complaint posted successfully",
Toast.LENGTH_SHORT).show();
                            finish(); // Finish the activity after successful posting
                        } else {
                            Toast.makeText(OneComplaintDisplay.this, "Failed to post complaint",
Toast.LENGTH_SHORT).show();
                        }
                    }
                });
        }
    }
}

```

```

        }
    });
    } else {
        Toast.makeText(OneComplaintDisplay.this, "Failed to generate unique key",
            Toast.LENGTH_SHORT).show();
    }
    } else {
        Toast.makeText(OneComplaintDisplay.this, "Failed to get current user", Toast.LENGTH_SHORT).show();
    }
}
}
}

```

### **StundeelectricalComplaintDetails.java**

```
package my.hostelcomply.app.electricianPanel;
```

```
public class StundeelectricalComplaintDetails {
```

```

    private String desc;
    private String fname;
    private String phonenumber;
    private String problemitem;
    private String randomUid;
    private String roomnumber;
    private String studentId;
    private String status;

```

```

    public StundeelectricalComplaintDetails(String desc, String fname, String phonenumber, String problemitem,
        String randomUid, String roomnumber, String studentId,String status) {

```

```

        this.desc = desc;
        this.fname = fname;
        this.phonenumber = phonenumber;
        this.problemitem = problemitem;
        this.randomUid = randomUid;
        this.roomnumber = roomnumber;
        this.studentId = studentId;
        this.status=status;
    }

```

```

    public String getDesc() {
        return desc;
    }

```

```

    public void setDesc(String desc) {
        this.desc = desc;
    }

```

```
public String getFname() {  
    return fname;  
}  
  
public void setFname(String fname) {  
    this.fname = fname;  
}  
  
public String getPhonenumber() {  
    return phonenumber;  
}  
  
public void setPhonenumber(String phonenumber) {  
    this.phonenumber = phonenumber;  
}  
  
public String getProblemitem() {  
    return problemitem;  
}  
  
public void setProblemitem(String problemitem) {  
    this.problemitem = problemitem;  
}  
  
public String getRandomUid() {  
    return randomUid;  
}  
  
public void setRandomUid(String randomUid) {  
    this.randomUid = randomUid;  
}  
  
public String getRoomnumber() {  
    return roomnumber;  
}  
  
public void setRoomnumber(String roomnumber) {  
    this.roomnumber = roomnumber;  
}  
  
public String getStudentId() {  
    return studentId;  
}
```



```

    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}

```

### **StundeelectricalComplaintDetailsAdapter.java**

```

package my.hostelcomply.app.electricianPanel;

```

```

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

```

```

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

```

```

import java.util.ArrayList;

```

```

import my.hostelcomply.app.R;

```

```

public class StundeelectricalComplaintDetailsAdapter extends
RecyclerView.Adapter<StundeelectricalComplaintDetailsAdapter.MyViewHolder> {
    Context context;
    ArrayList<StundeelectricalComplaintDetails> list;

```

```

    public StundeelectricalComplaintDetailsAdapter(Context context, ArrayList<StundeelectricalComplaintDetails>
list) {
        this.context = context;
        this.list = list;
    }

```

```

    @NonNull

```

```

@Override
public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(context).inflate(R.layout.complaint_item, parent, false);
    return new MyViewHolder(v);
}

```

```

@Override
public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
    StudentelectricalComplaintDetails complaint = list.get(position);
    holder.firstname.setText(complaint.getFname());
    holder.problem_item.setText(complaint.getProblemitem());
    holder.description.setText(complaint.getDesc());
    holder.roomnumber.setText(complaint.getRoomnumber());
    holder.phonenumber.setText(complaint.getPhonenumber());
}

```

```

holder.cardView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(context, OneComplaintDisplay.class);
        intent.putExtra("firstname", complaint.getFname());
        intent.putExtra("problem_item", complaint.getProblemitem());
        intent.putExtra("description", complaint.getDesc());
        intent.putExtra("roomnumber", complaint.getRoomnumber());
        intent.putExtra("phonenumber", complaint.getPhonenumber());
        intent.putExtra("studentId", complaint.getStudentId());
        intent.putExtra("randomUid", complaint.getRandomUid());

        context.startActivity(intent);
    }
});
}

```

```

@Override
public int getItemCount() {
    return list.size();
}

```

```

public static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView firstname, problem_item, description, roomnumber, phonenumber;
    CardView cardView;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
    }
}

```

```

        firstname = itemView.findViewById(R.id.first_name);
        problem_item = itemView.findViewById(R.id.problem_item);
        description = itemView.findViewById(R.id.description);
        roomnumber = itemView.findViewById(R.id.roomnumber);
        phonenumber = itemView.findViewById(R.id.phonenumber);
        cardView = itemView.findViewById(R.id.cardView);
    }
}
}

```

### **StudentelectricalComplaintDetailsList.java**

```
package my.hostelcomply.app.electricianPanel;
```

```

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

```

```

import android.os.Bundle;
import android.util.Log;

```

```

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

```

```
import java.util.ArrayList;
```

```
import my.hostelcomply.app.R;
```

```

public class StudentelectricalComplaintDetailsList extends AppCompatActivity {
    RecyclerView recyclerView;
    DatabaseReference database;
    StudentelectricalComplaintDetailsAdapter studentelectricalComplaintDetailsAdapter;
    ArrayList<StudentelectricalComplaintDetails> list;

```

```
@Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_studentelectrical_complaint_details_list);

```

```

        recyclerView = findViewById(R.id.StudentelectricalComplaintDetailsList);
        database = FirebaseDatabase.getInstance().getReference("StudentelectricalComplaintDetails");
        recyclerView.setHasFixedSize(true);

```

```

recyclerView.setLayoutManager(new LinearLayoutManager(this));

list = new ArrayList<>();
studentelectricalComplaintDetailsAdapter = new StudentelectricalComplaintDetailsAdapter(this, list);
recyclerView.setAdapter(studentelectricalComplaintDetailsAdapter);

// Retrieve data from Firebase
database.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        list.clear(); // Clear the list before adding new data
        for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
            // Get data from each complaint entry
            String desc = dataSnapshot.child("desc").getValue(String.class);
            String fname = dataSnapshot.child("fname").getValue(String.class);
            String phonenumber = dataSnapshot.child("onenumber").getValue(String.class);
            String problemitem = dataSnapshot.child("problemitem").getValue(String.class);
            String randomUid = dataSnapshot.child("randomUid").getValue(String.class);
            String roomnumber = dataSnapshot.child("roomnumber").getValue(String.class);
            String studentId = dataSnapshot.child("studentId").getValue(String.class);
            String status = dataSnapshot.child("status").getValue(String.class);

            // Create StudentelectricalComplaintDetails object
            StudentelectricalComplaintDetails complaintDetails = new StudentelectricalComplaintDetails(desc,
            fname, phonenumber, problemitem, randomUid, roomnumber, studentId, status);
            if (complaintDetails.getStatus().equals("p")) {
                list.add(complaintDetails);
            }

            // Add the object to the list

        }
        // Notify adapter about data changes
        studentelectricalComplaintDetailsAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Handle database error
        Log.e("Firebase", "Error: " + error.getMessage());
    }
});
studentelectricalComplaintDetailsAdapter = new StudentelectricalComplaintDetailsAdapter(this, list);

```

```

        recyclerView.setAdapter(studentelectricalComplaintDetailsAdapter);
    }
}

```

### **StudentelectricalComplaintDetailswithdate.java**

```
package my.hostelcomply.app.electricianPanel;
```

```
import java.util.Date;
```

```
public class StudentelectricalComplaintDetailswithdate {
```

```
    private String date;
```

```
    private String time;
```

```
    private String firstname;
```

```
    private String problemIte;
```

```
    private String description;
```

```
    private String roomNumber;
```

```
    private String phoneNumber;
```

```
    private String status;
```

```
    private String studentId;
```

```
//    private String efirstname;
```

```
    private String emobile;
```

```
    private String key;
```

```
    private String randomUid;
```

```
    public StudentelectricalComplaintDetailswithdate() {
```

```
    }
```

```

    public StudentelectricalComplaintDetailswithdate(String date, String time, String firstname, String problemIte,
String description, String roomNumber, String phoneNumber,String status,String studentId,String emobile,String
key,String randomUid) {

```

```
        this.date = date;
```

```
        this.time = time;
```

```
        this.firstname = firstname;
```

```
        this.problemIte = problemIte;
```

```
        this.description = description;
```

```
        this.roomNumber = roomNumber;
```

```
        this.phoneNumber = phoneNumber;
```

```
        this.status=status;
```

```
        this.studentId=studentId;
```

```
//        this.efirstname=efirstname;
```

```
        this.emobile=emobile;
```

```
        this.key=key;
```

```
        this.randomUUID=randomUid;

    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getProblemIte() {
        return problemIte;
    }

    public void setProblemIte(String problemIte) {
        this.problemIte = problemIte;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

```

public String getRoomNumber() {
    return roomNumber;
}

public void setRoomNumber(String roomNumber) {
    this.roomNumber = roomNumber;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

public String getStudentId() {
    return studentId;
}

public void setStudentId(String studentId) {
    this.studentId = studentId;
}

// public String getEfirstname() {
//     return efirstname;
// }
//
// public void setEfirstname(String efirstname) {
//     this.efirstname = efirstname;
// }

public String getEmobile() {
    return emobile;
}

```

```

public void setEmobile(String emobile) {
    this.emobile = emobile;
}

public String getKey() {
    return key;
}

public String getRandomUid() {
    return randomUid;
}

public void setRandomUid(String randomUid) {
    this.randomUid = randomUid;
}

public void setKey(String key) {
    this.key = key;
}
}

```

## 2.plumberPanel

### OneComplaintDisplayp.java

```

package my.hostelcomply.app.plumberPanel;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import my.hostelcomply.app.R;

```



```

import my.hostelcomply.app.electricianPanel.OneComplaintDisplay;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailswithdate;

public class OneComplaintDisplay extends AppCompatActivity {

    EditText dateEditText, timeEditText;
    Button saveButton;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference databaseReference;
    String dateStr, timeStr, status;
    StudentplumbingComplaintDetailswithdate sdw;

    FirebaseAuth mAuth;

    @SuppressWarnings("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one_complaint_display);
        dateEditText = findViewById(R.id.dateEditText);
        timeEditText = findViewById(R.id.timeEditText);
        saveButton = findViewById(R.id.saveButton);

        firebaseDatabase = FirebaseDatabase.getInstance();
        databaseReference = firebaseDatabase.getReference("StudentplumbingComplaintDetailswithdate");

        // Initialize the FirebaseAuth instance
        mAuth = FirebaseAuth.getInstance();

        // Initialize the StudentelectricalComplaintDetailswithdate object
        sdw = new StudentplumbingComplaintDetailswithdate();

        // Retrieve data from intent
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            String firstname = extras.getString("firstname");
            String problemItem = extras.getString("problem_item");
            String description = extras.getString("description");
            String roomNumber = extras.getString("roomnumber");
            String phoneNumber = extras.getString("phonenummer");
            String studentId=extras.getString("studentId");

```

```

String randomUid=extras.getString("randomUid");

// Display data in TextViews
TextView firstNameTextView = findViewById(R.id.firstNameTextView);
TextView problemItemTextView = findViewById(R.id.problemItemTextView);
TextView descriptionTextView = findViewById(R.id.descriptionTextView);
TextView roomNumberTextView = findViewById(R.id.roomNumberTextView);
TextView phoneNumberTextView = findViewById(R.id.phoneNumberTextView);

firstNameTextView.setText(firstname);
problemItemTextView.setText(problemItem);
descriptionTextView.setText(description);
roomNumberTextView.setText(roomNumber);
phoneNumberTextView.setText(phoneNumber);

sdw.setFirstname(firstname);
sdw.setProblemItem(problemItem);
sdw.setDescription(description);
sdw.setRoomNumber(roomNumber);
sdw.setPhoneNumber(phoneNumber);
sdw.setStudentId(studentId);
sdw.setRandomUid(randomUid);

}

status = "du";

saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dateStr = dateEditText.getText().toString();
        timeStr = timeEditText.getText().toString();

        if (TextUtils.isEmpty(dateStr) || TextUtils.isEmpty(timeStr)) {
            Toast.makeText(OneComplaintDisplay.this, "Please enter date and time",
Toast.LENGTH_SHORT).show();
        } else {
            saveComplaint();
        }
    }
});
}

```

```

private void saveComplaint() {
    // Get the current user
    FirebaseUser currentUser = mAuth.getCurrentUser();

    if (currentUser != null) {

        String electricianPhoneNumber = currentUser.getPhoneNumber();
        String key = databaseReference.push().getKey();

        // Create a new complaint object
        StudentplumbingComplaintDetailswithdate complaintwd = new
        StudentplumbingComplaintDetailswithdate(dateStr, timeStr, sdw.getFirstname(), sdw.getProblemIte(),
        sdw.getDescription(), sdw.getRoomNumber(), sdw.getPhoneNumber(), status, sdw.getStudentId(),
        electricianPhoneNumber, key, sdw.getRandomUid());

        // Generate a unique key for the complaint

        if (key != null) {
            // Save the complaint to Firebase
            databaseReference.child(key).setValue(complaintwd)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(OneComplaintDisplay.this, "Complaint posted successfully",
                            Toast.LENGTH_SHORT).show();
                            finish(); // Finish the activity after successful posting
                        } else {
                            Toast.makeText(OneComplaintDisplay.this, "Failed to post complaint",
                            Toast.LENGTH_SHORT).show();
                        }
                    }
                });
        } else {
            Toast.makeText(OneComplaintDisplay.this, "Failed to generate unique key",
            Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(OneComplaintDisplay.this, "Failed to get current user", Toast.LENGTH_SHORT).show();
    }
}
}

```

### **PlumberPendingComplaintsFragment.java**

```
package my.hostelcomply.app.plumberPanel;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import my.hostelcomply.app.R;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailsList;

public class PlumberPendingComplaintsFragment extends Fragment {

    Button viewComplaint;

    @SuppressLint("MissingInflatedId")
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_electrician_pending_complaints,null);
        getActivity().setTitle("View complaints");

        viewComplaint=(Button) v.findViewById(R.id.view_complaint);
        viewComplaint.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getContext(), StudentelplumbingComplaintDetailsList.class));
            }

        });
        return v;
    }
}
```

```
}
```

### **PlumberSolvedComplaintFragment.java**

```
package my.hostelcomply.app.plumberPanel;
```

```
import androidx.fragment.app.Fragment;
```

```
public class PlumberSolvedComplaintFragment extends Fragment {  
}
```

### **Plumber\_hostel\_layout.java**

```
package my.hostelcomply.app.plumberPanel;
```

```
import static android.app.PendingIntent.getActivity;
```

```
import androidx.annotation.NonNull;
```

```
import androidx.annotation.Nullable;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.fragment.app.Fragment;
```

```
import android.os.Bundle;
```

```
import android.view.LayoutInflater;
```

```
import android.view.View;
```

```
import android.view.ViewGroup;
```

```
import my.hostelcomply.app.R;
```

```
public class Plumber_hostel_layout extends Fragment {
```

```
    @Nullable
```

```
    @Override
```

```
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle  
savedInstanceState) {
```

```
        View v = inflater.inflate(R.layout.activity_plumber_hostel_layout,null);
```

```
        getActivity().s
```

```
etTitle("Plumber hostel layout");
```

```
        return v;
```

```
    }
```

```
}
```

### **StudentPlumbingComplaintDetails.java**

```
package my.hostelcomply.app.plumberPanel;
```

```
public class StudentPlumbingComplaintDetails {
```

```

private String desc;
private String fname;
private String phonenumber;
private String problemitem;
private String randomUid;
private String roomnumber;
private String studentId;
private String status;

    public StudentPlumbingComplaintDetails(String desc, String fname, String phonenumber, String problemitem,
String randomUid, String roomnumber, String studentId,String status) {
        this.desc = desc;
        this.fname = fname;
        this.phonenumber = phonenumber;
        this.problemitem = problemitem;
        this.randomUid = randomUid;
        this.roomnumber = roomnumber;
        this.studentId = studentId;
        this.status=status;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }

    public String getFname() {
        return fname;
    }

    public void setFname(String fname) {
        this.fname = fname;
    }

    public String getPhonenumber() {
        return phonenumber;
    }

    public void setPhonenumber(String phonenumber) {
        this.phonenumber = phonenumber;
    }

```

```

public String getProblemitem() {
    return problemitem;
}

public void setProblemitem(String problemitem) {
    this.problemitem = problemitem;
}

public String getRandomUid() {
    return randomUid;
}

public void setRandomUid(String randomUid) {
    this.randomUid = randomUid;
}

public String getRoomnumber() {
    return roomnumber;
}

public void setRoomnumber(String roomnumber) {
    this.roomnumber = roomnumber;
}

public String getStudentId() {
    return studentId;
}

public void setStudentId(String studentId) {
    this.studentId = studentId;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}
}

```

**StudentelplumbingComplaintDetailsList.java**

```
package my.hostelcomply.app.plumberPanel;
```

```

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.util.Log;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

import my.hostelcomply.app.R;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetails;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailsAdapter;

public class StudentelplumbingComplaintDetailsList extends AppCompatActivity {

    RecyclerView recyclerView;
    DatabaseReference database;
    StudentplumbingComplaintDetailsAdapter studentplumbingComplaintDetailsAdapter;
    ArrayList<StudentPlumbingComplaintDetails> list;

    @SuppressWarnings("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_studentelplumbing_complaint_details_list);

        recyclerView = findViewById(R.id.StudentelplumbingComplaintDetailsList);
        database = FirebaseDatabase.getInstance().getReference("StudentPlumbingComplaintDetails");
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        list = new ArrayList<>();
        studentplumbingComplaintDetailsAdapter = new StudentplumbingComplaintDetailsAdapter(this, list);
        recyclerView.setAdapter(studentplumbingComplaintDetailsAdapter);
    }
}

```



```

// Retrieve data from Firebase
database.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        list.clear(); // Clear the list before adding new data
        for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
            // Get data from each complaint entry
            String desc = dataSnapshot.child("desc").getValue(String.class);
            String fname = dataSnapshot.child("fname").getValue(String.class);
            String phonenumber = dataSnapshot.child("phonenumber").getValue(String.class);
            String problemitem = dataSnapshot.child("problemitem").getValue(String.class);
            String randomUid = dataSnapshot.child("randomUid").getValue(String.class);
            String roomnumber = dataSnapshot.child("roomnumber").getValue(String.class);
            String studentId = dataSnapshot.child("studentId").getValue(String.class);
            String status = dataSnapshot.child("status").getValue(String.class);

            // Create StudentelectricalComplaintDetails object
            StudentPlumbingComplaintDetails complaintDetails = new StudentPlumbingComplaintDetails(desc,
            fname, phonenumber, problemitem, randomUid, roomnumber, studentId, status);
            if (complaintDetails.getStatus().equals("p")) {
                list.add(complaintDetails);
            }

            // Add the object to the list

        }
        // Notify adapter about data changes
        studentplumbingComplaintDetailsAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Handle database error
        Log.e("Firebase", "Error: " + error.getMessage());
    }
});
studentplumbingComplaintDetailsAdapter = new StudentplumbingComplaintDetailsAdapter(this, list);
recyclerView.setAdapter(studentplumbingComplaintDetailsAdapter);
}
}

```

**StudentplumbingComplaintDetailsAdapter.java**

```

package my.hostelcomply.app.plumberPanel;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

import my.hostelcomply.app.R;
import my.hostelcomply.app.electricianPanel.OneComplaintDisplay;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetails;

public class StudentplumbingComplaintDetailsAdapter extends
RecyclerView.Adapter<StudentplumbingComplaintDetailsAdapter.MyViewHolder> {
    Context context;
    ArrayList<StudentPlumbingComplaintDetails> list;

    public StudentplumbingComplaintDetailsAdapter(Context context,
ArrayList<StudentPlumbingComplaintDetails> list) {
        this.context = context;
        this.list = list;
    }

    @NonNull
    @Override
    public StudentplumbingComplaintDetailsAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View v = LayoutInflater.from(context).inflate(R.layout.complaint_item, parent, false);
        return new StudentplumbingComplaintDetailsAdapter.MyViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull StudentplumbingComplaintDetailsAdapter.MyViewHolder holder, int
position) {
        StudentPlumbingComplaintDetails complaint = list.get(position);
        holder.firstname.setText(complaint.getFname());
        holder.problem_item.setText(complaint.getProblemitem());
    }
}

```

```
holder.description.setText(complaint.getDesc());
holder.roomnumber.setText(complaint.getRoomnumber());
holder.phonenumber.setText(complaint.getPhonenumber());
```

```
holder.cardView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(context, OneComplaintDisplayp.class);
        intent.putExtra("firstname", complaint.getFname());
        intent.putExtra("problem_item", complaint.getProblemitem());
        intent.putExtra("description", complaint.getDesc());
        intent.putExtra("roomnumber", complaint.getRoomnumber());
        intent.putExtra("phonenumber", complaint.getPhonenumber());
        intent.putExtra("studentId", complaint.getStudentId());
        intent.putExtra("randomUid", complaint.getRandomUid());

        context.startActivity(intent);
    }
});
}
```

```
@Override
public int getItemCount() {
    return list.size();
}
```

```
public static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView firstname, problem_item, description, roomnumber, phonenumber;
    CardView cardView;
```

```
public MyViewHolder(@NonNull View itemView) {
    super(itemView);
    firstname = itemView.findViewById(R.id.first_name);
    problem_item = itemView.findViewById(R.id.problem_item);
    description = itemView.findViewById(R.id.description);
    roomnumber = itemView.findViewById(R.id.roomnumber);
    phonenumber = itemView.findViewById(R.id.phonenumber);
    cardView = itemView.findViewById(R.id.cardView);
}
```

```
}
```

### **StudentplumbingComplaintDetailswithdate.java**

```
package my.hostelcomply.app.plumberPanel;
```

```

public class StudentplumbingComplaintDetailswithdate {

    private String date;
    private String time;

    private String firstname;
    private String problemIte;
    private String description;
    private String roomNumber;
    private String phoneNumber;
    private String status;
    private String studentId;

    // private String efirstname;
    private String emobile;
    private String key;
    private String randomUid;

    public StudentplumbingComplaintDetailswithdate() {
    }

    public StudentplumbingComplaintDetailswithdate(String date, String time, String firstname, String problemIte,
String description, String roomNumber, String phoneNumber,String status,String studentId,String emobile,String
key,String randomUid) {
        this.date = date;
        this.time = time;
        this.firstname = firstname;
        this.problemIte = problemIte;
        this.description = description;
        this.roomNumber = roomNumber;
        this.phoneNumber = phoneNumber;
        this.status=status;
        this.studentId=studentId;
    //    this.efirstname=efirstname;
        this.emobile=emobile;
        this.key=key;
        this.randomUid=randomUid;

    }

    public String getDate() {
        return date;
    }
}

```

```
public void setDate(String date) {
    this.date = date;
}

public String getTime() {
    return time;
}

public void setTime(String time) {
    this.time = time;
}

public String getFirstname() {
    return firstname;
}

public void setFirstname(String firstname) {
    this.firstname = firstname;
}

public String getProblemIte() {
    return problemIte;
}

public void setProblemIte(String problemIte) {
    this.problemIte = problemIte;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public String getRoomNumber() {
    return roomNumber;
}

public void setRoomNumber(String roomNumber) {
    this.roomNumber = roomNumber;
}
```

```

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

public String getStudentId() {
    return studentId;
}

public void setStudentId(String studentId) {
    this.studentId = studentId;
}

// public String getEfirstname() {
//     return efirstname;
// }
//
// public void setEfirstname(String efirstname) {
//     this.efirstname = efirstname;
// }

public String getEmobile() {
    return emobile;
}

public void setEmobile(String emobile) {
    this.emobile = emobile;
}

public String getKey() {
    return key;
}

```

```

public String getRandomUid() {
    return randomUid;
}

public void setRandomUid(String randomUid) {
    this.randomUid = randomUid;
}

public void setKey(String key) {
    this.key = key;
}
}

```

### **3.studentPanel**

#### **MyelectricalpendingComplaintDetailsList.java**

```

package my.hostelcomply.app.studentPanel;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.util.Log;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

import my.hostelcomply.app.R;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetails;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailsAdapter;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailswithdate;

public class MyelectricalpendingComplaintDetailsList extends AppCompatActivity {

    RecyclerView recyclerView;
    DatabaseReference database;
    StudentEComplaintDetailsAdapter studentEComplaintDetailsAdapter;

```

```

ArrayList<StundelectricalComplaintDetailswithdate> list;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_myelectricalpending_complaint_details_list);

    recyclerView = findViewById(R.id.MyelectricalpendingComplaintDetailsList);
    database = FirebaseDatabase.getInstance().getReference("StundelectricalComplaintDetailswithdate");
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    list = new ArrayList<>();
    studentEComplaintDetailsAdapter = new StudentEComplaintDetailsAdapter(this, list);
    recyclerView.setAdapter(studentEComplaintDetailsAdapter);
    String userid = FirebaseAuth.getInstance().getCurrentUser().getUid();

    // Retrieve data from Firebase
    database.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            list.clear(); // Clear the list before adding new data
            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                // Get data from each complaint entry
                StundelectricalComplaintDetailswithdate complaint =
dataSnapshot.getValue(StundelectricalComplaintDetailswithdate.class);
                if (complaint != null && complaint.getStatus().equals("du")&&complaint.getStudentId().equals(userid))
            {
                list.add(complaint);
            }
        }
        // Notify adapter about data changes
        studentEComplaintDetailsAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Handle database error
        Log.e("Firebase", "Error: " + error.getMessage());
    }
    });
}
}

```

**MyplumbingpendingComplaintDetailsList.java**



```

package my.hostelcomply.app.studentPanel;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.util.Log;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

import my.hostelcomply.app.R;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailswithdate;
import my.hostelcomply.app.plumberPanel.StudentplumbingComplaintDetailswithdate;

public class MyplumbingpendingComplaintDetailsList extends AppCompatActivity {

    RecyclerView recyclerView;
    DatabaseReference database;
    StudentPComplaintDetailsAdapter studentpComplaintDetailsAdapter;
    ArrayList<StudentplumbingComplaintDetailswithdate> list;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_myplumbingpending_complaint_details_list);

        recyclerView = findViewById(R.id.MyplumbingpendingComplaintDetailsList);
        database = FirebaseDatabase.getInstance().getReference("StudentplumbingComplaintDetailswithdate");
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

```

```

list = new ArrayList<>();
studenttpComplaintDetailsAdapter = new StudentPComplaintDetailsAdapter(this, list);
recyclerView.setAdapter(studenttpComplaintDetailsAdapter);
String userid = FirebaseAuth.getInstance().getCurrentUser().getUid();

// Retrieve data from Firebase
database.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        list.clear(); // Clear the list before adding new data
        for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
            // Get data from each complaint entry
            StudentplumbingComplaintDetailwithdate complaint =
dataSnapshot.getValue(StudentplumbingComplaintDetailwithdate.class);
            if (complaint != null && complaint.getStatus().equals("du")&&complaint.getStudentId().equals(userid))
{
                list.add(complaint);
            }
        }
        // Notify adapter about data changes
        studenttpComplaintDetailsAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // Handle database error
        Log.e("Firebase", "Error: " + error.getMessage());
    }
});
}
}

```

### **StudentEComplaintDetailsAdapter.java**

```
package my.hostelcomply.app.studentPanel;
```

```

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

```

```

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

```

```

import java.util.ArrayList;

import my.hostelcomply.app.R;
import my.hostelcomply.app.electricianPanel.OneComplaintDisplay;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetails;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailsAdapter;
import my.hostelcomply.app.electricianPanel.StudentelectricalComplaintDetailswithdate;

public class StudentEComplaintDetailsAdapter extends
RecyclerView.Adapter<StudentEComplaintDetailsAdapter.MyViewHolder> {

    Context context;
    ArrayList<StudentelectricalComplaintDetailswithdate> list;

    public StudentEComplaintDetailsAdapter(Context context,
ArrayList<StudentelectricalComplaintDetailswithdate> list) {
        this.context = context;
        this.list = list;
    }

    @NonNull
    @Override
    public StudentEComplaintDetailsAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View v = LayoutInflater.from(context).inflate(R.layout.student_complaint_item, parent, false);
        return new MyViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull StudentEComplaintDetailsAdapter.MyViewHolder holder, int
position) {
        StudentelectricalComplaintDetailswithdate complaint = list.get(position);
        holder.firstname.setText(complaint.getFirstname());
        holder.problem_item.setText(complaint.getProblemIte());
        holder.description.setText(complaint.getDescription());
        holder.roomnumber.setText(complaint.getRoomNumber());
        holder.emobile.setText(complaint.getEmobile());
        holder.date.setText(complaint.getDate());
        holder.time.setText(complaint.getTime());

        holder.cardView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate to OneComplaintDisplay activity
                Intent intent = new Intent(context,OneComplaintStudentDisplay.class);
                intent.putExtra("firstname", complaint.getFirstname());
                intent.putExtra("problem_item", complaint.getProblemIte());
            }
        });
    }
}

```

```

        intent.putExtra("description", complaint.getDescription());
        intent.putExtra("roomnumber", complaint.getRoomNumber());
        intent.putExtra("phonenum", complaint.getEmobile());
        intent.putExtra("studentId", complaint.getStudentId());
        intent.putExtra("date", complaint.getDate());
        intent.putExtra("time", complaint.getTime());
        intent.putExtra("status", complaint.getStatus());

```

```

        intent.putExtra("key", complaint.getKey());
        intent.putExtra("randomUid", complaint.getRandomUid());

```

```

        context.startActivity(intent);
    }
});
}

```

```

@Override
public int getItemCount() {
    return list.size();
}

```

```

public static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView firstname, problem_item, description, roomnumber, emobile, date, time;
    CardView cardView;

```

```

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        firstname = itemView.findViewById(R.id.first_name);
        problem_item = itemView.findViewById(R.id.problem_item);
        description = itemView.findViewById(R.id.description);
        roomnumber = itemView.findViewById(R.id.roomnumber);
        emobile = itemView.findViewById(R.id.emobile);
        date = itemView.findViewById(R.id.date);
        time = itemView.findViewById(R.id.time);

```

```

        cardView = itemView.findViewById(R.id.cardView);

```

```

    }
}
}

```

### **StudentProfileFragment.java**

```

package my.hostelcomply.app.studentPanel;

```

```

import android.annotation.SuppressLint;

```

```

import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import org.checkerframework.common.subtyping.qual.Bottom;

import my.hostelcomply.app.R;

public class StudentProfileFragment extends Fragment {
    Button postComplaint;
    ConstraintLayout backing;

    @SuppressWarnings("MissingInflatedId")
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_student_profile,null);
        getActivity().setTitle("Post complaints");
        AnimationDrawable animationDrawable = new AnimationDrawable();

        animationDrawable.addFrame(getResources().getDrawable(R.drawable.bg2),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img2),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img3),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img4),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img5),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img6),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img7),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img8),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.bg3),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img9),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img10),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img11),3000);
        animationDrawable.addFrame(getResources().getDrawable(R.drawable.img11),3000);

        animationDrawable.setOneShot(false);
        animationDrawable.setEnterFadeDuration(850);

```

```

        animationDrawable.setExitFadeDuration(1600);
        backimg=v.findViewById(R.id.back1);
        backimg.setBackgroundDrawable(animationDrawable);
        animationDrawable.start();

        postComplaint=(Button) v.findViewById(R.id.post_complaint);
        postComplaint.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(),Student_postComplaint.class));
            }

        });
        return v;
    }
}

```

### **Student\_postComplaint.java**

```

package my.hostelcomply.app.studentPanel;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

import my.hostelcomply.app.R;
import my.hostelcomply.app.Student;

public class Student_postComplaint extends AppCompatActivity {

```

```

Button post_complaints;
DatabaseReference dataaa;
Spinner ProblemItems;
TextInputLayout desc;
String description, problems;
FirebaseStorage storage;
StorageReference storageReference;
FirebaseDatabase firebaseDatabase;
DatabaseReference databaseReference;
FirebaseAuth FirebaseAuth;
String StudentId;
String Roomnumber, Phonenummer, First_name,status;

@SuppressLint("MissingInflatedId")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_student_post_complaint);

    firebaseDatabase = FirebaseDatabase.getInstance();
    databaseReference = firebaseDatabase.getReference("StudentelectricalComplaintDetails");

    ProblemItems = findViewById(R.id.Pitems);
    desc = findViewById(R.id.description);
    post_complaints = findViewById(R.id.post);
    FirebaseAuth = FirebaseAuth.getInstance();

    try {
        String userid = FirebaseAuth.getInstance().getCurrentUser().getUid();
        dataaa = firebaseDatabase.getReference("Student").child(userid);
        dataaa.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Student studentc = dataSnapshot.getValue(Student.class);
                Roomnumber = studentc.getRoomnumber();
                Phonenummer = studentc.getMobile_No();
                First_name = studentc.getFirst_Name();
                status="p";

                post_complaints.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        problems = ProblemItems.getSelectedItem().toString().trim();
                        description = desc.getText().toString().trim();

                        if (isValid()) {
                            saveComplaint();
                        }
                    }
                })
            }
        })
    }
}

```

```

        });
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        Log.e("DatabaseError", databaseError.getMessage());
    }
});

} catch (Exception e) {
    Log.e("Errrrrr: ", e.getMessage());
}
}

private boolean isValid() {
    desc.setErrorEnabled(false);
    desc.setError("");

    boolean isValidDescription = !TextUtils.isEmpty(description);
    if (!isValidDescription) {
        desc.setError("Description is Required");
    }

    return isValidDescription;
}

private void saveComplaint() {
    String randomUid = String.valueOf(System.currentTimeMillis()); // Generating unique ID
    StudentId = FirebaseAuth.getInstance().getCurrentUser().getUid();

    StudentComplaintDetails complaint=new
StudentComplaintDetails(First_name,Roomnumber,Phonenumber,problems, description,randomUid,status,
StudentId);
    databaseReference.child(randomUid).setValue(complaint)
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(Student_postComplaint.this, "Complaint posted successfully",
Toast.LENGTH_SHORT).show();
                    desc.getEditText().setText(""); // Clearing the description field
                } else {
                    Toast.makeText(Student_postComplaint.this, "Failed to post complaint",
Toast.LENGTH_SHORT).show();
                }
            }
        });
}
}

```



```
}
```

### **ElectricianForgotPassword.java**

```
package my.hostelcomply.app;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class ElectricianForgotPassword extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_electrician_forgot_password);
    }
}
```

### **ElectricianRegistration.java**

```
package my.hostelcomply.app;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.Firebase;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.HashMap;

public class ElectricianRegistration extends AppCompatActivity{
    TextInputLayout Fname,Lname,Email,Pass,cpass,mobileno;
    Button signup, Email, Phone;
    FirebaseAuth FAuth;
    DatabaseReference databaseReference;
```

```

FirebaseDatabase firebaseDatabase;
String fname, lname, emailid, password, confirmPassword, mobile;
String role = "Electrician";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_electrician_registration);

    FName = (TextInputLayout)findViewById(R.id.Firstname);
    Lname = (TextInputLayout)findViewById(R.id.Lastname);
    Email = (TextInputLayout)findViewById(R.id.Email);
    Pass = (TextInputLayout)findViewById(R.id.Pwd);
    cpass = (TextInputLayout)findViewById(R.id.Cpass);
    mobileno = (TextInputLayout)findViewById(R.id.Mobileno);

    signup = (Button)findViewById(R.id.Signup);
    Email1 = (Button)findViewById(R.id.email);
    Phone = (Button)findViewById(R.id.phone);
    databaseReference = firebaseDatabase.getInstance().getReference("Electrician");
    FirebaseAuth.getInstance();
    signup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            fname = FName.getText().toString().trim();
            lname = Lname.getText().toString().trim();
            emailid = Email.getText().toString().trim();
            mobile = mobileno.getText().toString().trim();
            password = Pass.getText().toString().trim();
            confirmPassword = cpass.getText().toString().trim();

            if (isValid()){
                final ProgressDialog mDialog = new ProgressDialog(ElectricianRegistration.this);
                mDialog.setCancelable(false);
                mDialog.setCanceledOnTouchOutside(false);
                mDialog.setMessage("Registration in progress please wait.....");
                mDialog.show();

                FirebaseAuth.createUserWithEmailAndPassword(emailid, password).addOnCompleteListener(new
                OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {

                        if (task.isSuccessful()){
                            String useridd = FirebaseAuth.getInstance().getCurrentUser().getUid();
                            databaseReference = FirebaseDatabase.getInstance().getReference("User").child(useridd);
                            final HashMap<String, String> hashMap = new HashMap<>();
                            hashMap.put("Role", role);

```

```

        databaseReference.setValue(hashMap).addOnCompleteListener(new
OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {

        HashMap<String , String> hashMap1 = new HashMap<>();
        hashMap1.put("Mobile_No",mobile);
        hashMap1.put("First_Name",fname);
        hashMap1.put("Last_Name",lname);
        hashMap1.put("EmailId",emailid);
        hashMap1.put("Password",password);
        hashMap1.put("Confirm_Password",confpassword);

        firebaseDatabase.getInstance().getReference("Electrician")
        .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
        .setValue(hashMap1).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                mDialog.dismiss();

                FirebaseAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener(new
OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {

                        if(task.isSuccessful()){
                            AlertDialog.Builder builder = new
AlertDialog.Builder(ElectricianRegistration.this);
                            builder.setMessage("You Have Registered! Make Sure To Verify Your
Email");

                            builder.setCancelable(false);
                            builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {

                                    dialog.dismiss();
                                    String phonenumber = "+91"+mobile;
                                    Intent b = new Intent(ElectricianRegistration.this,
ElectricianVerifyPhone.class);

                                    b.putExtra("phonenumber", phonenumber);
                                    startActivity(b);

                                }
                            });
                            AlertDialog Alert = builder.create();
                            Alert.show();
                        }else{
                            mDialog.dismiss();

```



```

cpass.setErrorEnabled(false);
cpass.setError("");

boolean
isValid=false,isValidIname=false,isValidname=false,isValidemail=false,isValidpassword=false,isValidconfpassw
ord=false,isValidmobilenum=false;
if(TextUtils.isEmpty(fname)){
    Fname.setErrorEnabled(true);
    Fname.setError("Enter First Name");
}else{
    isValidname = true;
}
if(TextUtils.isEmpty(lname)){
    Lname.setErrorEnabled(true);
    Lname.setError("Enter Last Name");
}else{
    isValidIname = true;
}
if(TextUtils.isEmpty(emailid)){
    Email.setErrorEnabled(true);
    Email.setError("Email Is Required");
}else{
    if(emailid.matches(emailpattern)){
        isValidemail = true;
    }else{
        Email.setErrorEnabled(true);
        Email.setError("Enter a Valid Email Id");
    }
}
if(TextUtils.isEmpty(password)){
    Pass.setErrorEnabled(true);
    Pass.setError("Enter Password");
}else{
    if(password.length()<8){
        Pass.setErrorEnabled(true);
        Pass.setError("Password is Weak");
    }else{
        isValidpassword = true;
    }
}
if(TextUtils.isEmpty(confpassword)){
    cpass.setErrorEnabled(true);
    cpass.setError("Enter Password Again");
}else{
    if(!password.equals(confpassword)){
        cpass.setErrorEnabled(true);
        cpass.setError("Password Dosen't Match");
    }else{

```

```

        isValidconfpassword = true;
    }
}
if(TextUtils.isEmpty(mobile)){
    mobileno.setErrorEnabled(true);
    mobileno.setError("Mobile Number Is Required");
}else{
    if(mobile.length()<10){
        mobileno.setErrorEnabled(true);
        mobileno.setError("Invalid Mobile Number");
    }else{
        isValidmobilenum = true;
    }
}
}

```

```

    isValid = (isValidconfpassword && isValidpassword && isValidemail && isValidmobilenum &&
    isValidname && isValidlname) ? true : false;
    return isValid;

```

```

    }
}

```

### **Login.java**

```

package my.hostelcomply.app;

```

```

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

```

```

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

```

```

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

```

```

public class Login extends AppCompatActivity{

```

```

    TextInputLayout email,pass;
    Button Signin,Signinphone;

```

```

TextView Forgotpassword , signup;
FirebaseAuth Fauth;
String emailid,pwd;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    try{

        email = (TextInputLayout)findViewById(R.id.Lemail);
        pass = (TextInputLayout)findViewById(R.id.Lpassword);
        Signin = (Button)findViewById(R.id.button4);
        signup = (TextView) findViewById(R.id.textView3);
        Forgotpassword = (TextView)findViewById(R.id.forgotpass);
        Signinphone = (Button)findViewById(R.id.btnphone);

        Fauth = FirebaseAuth.getInstance();

        Signin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                emailid = email.getText().getText().toString().trim();
                pwd = pass.getText().getText().toString().trim();

                if(isValid()){

                    final ProgressDialog mDialog = new ProgressDialog(Login.this);
                    mDialog.setCanceledOnTouchOutside(false);
                    mDialog.setCancelable(false);
                    mDialog.setMessage("Sign In Please Wait.....");
                    mDialog.show();

                    Fauth.signInWithEmailAndPassword(emailid,pwd).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {

                            if(task.isSuccessful()){
                                mDialog.dismiss();

                                if(Fauth.getCurrentUser().isEmailVerified()){
                                    mDialog.dismiss();
                                    Toast.makeText(Login.this, "Congratulation! You Have Successfully Login",
Toast.LENGTH_SHORT).show();
                                    Intent Z = new Intent(Login.this,Panel_BottomNavigation.class);
                                    startActivity(Z);
                                }
                            }
                        }
                    });
                }
            }
        });
    }
}

```

```

        finish();

    }else{
        ReusableCodeForAll.ShowAlert(Login.this,"Verification Failed","You Have Not Verified
Your Email");

    }
    }else{
        mDialog.dismiss();
        ReusableCodeForAll.ShowAlert(Login.this,"Error",task.getException().getMessage());
    }
    }
    });
}
}
});
signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(Login.this,Registration.class));
        finish();
    }
});
Forgotpassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(Login.this,ForgotPassword.class));
        finish();
    }
});
Signinphone.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(Login.this,Loginphone.class));
        finish();
    }
});
}catch (Exception e){
    Toast.makeText(this,e.getMessage(),Toast.LENGTH_LONG).show();
}

}
String emailpattern = "[a-zA-Z0-9._-]+@[a-zA-Z0-9+\\\\.ac\\.in";

public boolean isValid(){

    email.setErrorEnabled(false);
    email.setError("");
    pass.setErrorEnabled(false);

```



```

pass.setError("");

boolean isvalid=false,isvalidemail=false,isvalidpassword=false;
if(TextUtils.isEmpty(emailid)){
    email.setErrorEnabled(true);
    email.setError("Email is required");
}else{
    if(emailid.matches(emailpattern)){
        isvalidemail=true;
    }else{
        email.setErrorEnabled(true);
        email.setError("Invalid Email Address");
    }
}
if(TextUtils.isEmpty(pwd)){

    pass.setErrorEnabled(true);
    pass.setError("Password is Required");
}else{
    isvalidpassword=true;
}
isvalid=(isvalidemail && isvalidpassword)?true:false;
return isvalid;
}
}

```

### **Plumber\_Loginphone.java**

```

package my.hostelcomply.app;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;

public class Plumber_Loginphone extends AppCompatActivity{
    EditText num;
    Button sendotp,signinemail;
    TextView txtsignup;
    FirebaseAuth FAuth;
    String numberr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```

setContentView(R.layout.activity_plumber_loginphone);

num=(EditText)findViewById(R.id.number);
sendotp=(Button)findViewById(R.id.otp);
signinemail=(Button)findViewById(R.id.btnEmail);
txtsignup=(TextView)findViewById(R.id.acsignup);


FAuth=FirebaseAuth.getInstance();

sendotp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        numberr=num.getText().toString().trim();
        String phonenumber= "+91" + numberr;
        Intent b=new Intent(Plumber_Loginphone.this,Plumbersendotp.class);
        b.putExtra("phonenumber",phonenumber);
        startActivity(b);
        finish();

    }
});

txtsignup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent a=new Intent(Plumber_Loginphone.this,Plumber_Registration.class);
        startActivity(a);
        finish();
    }
});

signinemail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent em=new Intent(Plumber_Loginphone.this, Plumber_Login.class);
        startActivity(em);
        finish();
    }
});

}
}

```

### **Student.java**

```

package my.hostelcomply.app;

public class Student {

```

```

private String First_Name, Last_Name, Roomnumber, EmailId, Password, Confirm_Password, Mobile_No;

public Student() {
}

public Student(String first_Name, String last_Name, String roomnumber, String emailId, String password,
String confirm_Password, String mobile_No) {
    First_Name = first_Name;
    Last_Name = last_Name;
    Roomnumber = roomnumber;
    EmailId = emailId;
    Password = password;
    Confirm_Password = confirm_Password;
    Mobile_No = mobile_No;
}

public String getFirst_Name() {
    return First_Name;
}

public void setFirst_Name(String first_Name) {
    First_Name = first_Name;
}

public String getLast_Name() {
    return Last_Name;
}

public void setLast_Name(String last_Name) {
    Last_Name = last_Name;
}

public String getRoomnumber() {
    return Roomnumber;
}

public void setRoomnumber(String roomnumber) {
    Roomnumber = roomnumber;
}

public String getEmailId() {
    return EmailId;
}

public void setEmailId(String emailId) {
    EmailId = emailId;
}

public String getPassword() {

```

```

        return Password;
    }

    public void setPassword(String password) {
        Password = password;
    }

    public String getConfirm_Password() {
        return Confirm_Password;
    }

    public void setConfirm_Password(String confirm_Password) {
        Confirm_Password = confirm_Password;
    }

    public String getMobile_No() {
        return Mobile_No;
    }

    public void setMobile_No(String mobile_No) {
        Mobile_No = mobile_No;
    }
}

```

### **VerifyPhone.java**

```

package my.hostelcomply.app;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;

import java.util.concurrent.TimeUnit;

```

```

public class VerifyPhone extends AppCompatActivity{
    String verificationId;
    Button Resend;
    TextView txt;
    String phonenumber;
    FirebaseAuth FAuth;
    Button verify;
    EditText entercode;

    @SuppressWarnings("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_verify_phone);
        phonenumber = getIntent().getStringExtra("onenumber").trim();

        sendverificationcode(onenumber);
        entercode = (EditText) findViewById(R.id.phoneno);
        txt = (TextView) findViewById(R.id.text);
        Resend = (Button) findViewById(R.id.Resendotp);
        Resend.setVisibility(View.INVISIBLE);
        txt.setVisibility(View.INVISIBLE);
        FAuth = FirebaseAuth.getInstance();
        verify = (Button) findViewById(R.id.Verify);
        verify.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String code = entercode.getText().toString().trim();
                Resend.setVisibility(View.INVISIBLE);

                if (code.isEmpty() && code.length() < 6) {
                    entercode.setError("Enter code");
                    entercode.requestFocus();
                    return;
                }
                verifyCode(code);
            }
        });

        new CountdownTimer(60000, 1000) {
            @Override
            public void onTick(long millisUntilFinished) {
                txt.setVisibility(View.VISIBLE);
                txt.setText("Resend Code within " + millisUntilFinished / 1000 + " Seconds");
            }
        }
    }
}

```

```

        @Override
        public void onFinish() {
            Resend.setVisibility(View.VISIBLE);
            txt.setVisibility(View.INVISIBLE);

        }
    }.start();

    Resend.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Resend.setVisibility(View.INVISIBLE);
            Resendotp(phonenummer);

            new CountdownTimer(60000, 1000) {
                @Override
                public void onTick(long millisUntilFinished) {
                    txt.setVisibility(View.VISIBLE);
                    txt.setText("Resend Code within " + millisUntilFinished / 1000 + " Seconds");
                }

                @Override
                public void onFinish() {
                    Resend.setVisibility(View.VISIBLE);
                    txt.setVisibility(View.INVISIBLE);

                }
            }.start();

        }
    });

}

private void Resendotp(String phonenummer) {

    sendverificationcode(phonenummer);
}

private void verifyCode(String code) {
    PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationId, code);
    linkCredential(credential);
}

private void linkCredential(PhoneAuthCredential credential) {

    FirebaseAuth.getCurrentUser().linkWithCredential(credential)

```

```

        .addOnCompleteListener(VerifyPhone.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {

                    Intent intent = new Intent(VerifyPhone.this, MainMenu.class);
                    startActivity(intent);
                    finish();

                } else {
                    ReusableCodeForAll.ShowAlert(VerifyPhone.this,"Error",task.getException().getMessage());
                }
            }
        });
    }
}

```

```

private void sendverificationcode(String number) {

```

```

//    PhoneAuthProvider.getInstance().verifyPhoneNumber(
//        number,
//        60,
//        TimeUnit.SECONDS,
//        TaskExecutors.MAIN_THREAD,
//        mCallBack
//    );
    PhoneAuthOptions options = PhoneAuthOptions.newBuilder(FirebaseAuth.getInstance())
        .setPhoneNumber(number)
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(this)
        .setCallbacks(mCallBack).build();
    PhoneAuthProvider.verifyPhoneNumber(options);
}

```

```

private PhoneAuthProvider.OnVerificationStateChangedCallbacks
    mCallBack = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
        @Override
        public void onCodeSent(String s, PhoneAuthProvider.ForceResendingToken forceResendingToken) {
            super.onCodeSent(s, forceResendingToken);

            verificationId = s;
            //ResendToken=forceResendingToken;

        }
    }

```

```

@Override
public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {

```

```

        String code = phoneAuthCredential.getSmsCode();
        if (code != null) {
            entercode.setText(code);
            verifyCode(code);

        }
    }

    @Override
    public void onVerificationFailed(FirebaseException e) {

        Toast.makeText(VerifyPhone.this, e.getMessage(), Toast.LENGTH_LONG).show();
    }
};
}

```

## 6.2 Integration Testing

Integration testing for our hostel management mobile application involves validating interactions between frontend, backend, database, and external services. Test cases cover scenarios like user authentication, room allocation, and communication features. Test data sets represent various scenarios and edge cases. Execution includes verifying data consistency, error handling, and performance. Regression testing ensures new changes don't break existing functionality, with documentation and reporting of test results for analysis and resolution.

## 6.3 White Box Testing

White box testing for your hostel management mobile app involves examining internal code structures, logic, and paths. Testers write test cases to cover all code branches and evaluate logical conditions. They assess code coverage to identify untested areas and improve overall test coverage. White box testing verifies boundary conditions, error handling, and security measures within the codebase. Techniques include unit tests, integration tests, and code reviews to ensure code quality and reliability.



## CHAPTER-7

### Conclusion and Future Work

#### 7.1 Conclusion

In conclusion, the hostel management mobile application presents an innovative solution for streamlining administrative tasks and enhancing communication between residents and management. With its intuitive interface, robust features, and emphasis on security and usability, the application promises to significantly improve the hostel living experience. Its successful implementation is poised to revolutionize hostel management practices, fostering efficiency, compliance, and satisfaction among users. As technology continues to evolve, the application stands as a testament to the potential of digital solutions in optimizing everyday processes and enriching user experiences.

#### 7.2 Future Work

Future work for the hostel management mobile application includes:

- Implementing additional features such as predictive maintenance scheduling and smart room allocation algorithms to further optimize operations.
- Enhancing security measures with biometric authentication options and advanced encryption protocols to safeguard sensitive data.
- Integrating IoT devices for real-time monitoring of hostel facilities and environmental conditions, enabling proactive maintenance and resource management.
- Expanding compatibility to support emerging platforms and technologies, ensuring continued relevance and accessibility for users.
- Exploring AI and machine learning algorithms for personalized recommendations, predictive analytics, and automated decision-making, further enhancing user experience and efficiency in hostel management.

### Bibliography

- <https://ieeexplore.ieee.org/>
- <https://dl.acm.org/>

