STUDENT STAY HUB

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
(Computer Science and Engineering)

Submitted By:

Harsh Rathi (2104115)
Harsharan Singh (2104116)
Jaskaran Singh (2104125)

Submitted To:

Mrs. Lakhvir Kaur Grewal
Assistant professor

Department of Computer Science and Engineering
Guru Nanak Dev Engineering College
LUDHIANA,141006

November, 2024

ABSTRACT

This is a design and implementation of an online Mess Management System “STUDENT STAY HUB” software developed for managing various activities in the hostel. The number of educational institutions has been increasing rapidly for the past few years. Thereby the number of hostels is also increasing for the accommodation of the students studying in this institution. Hence there is a lot of strain on the person who is running the hostel and software is not usually used in this context. This particular project deals with the problems of managing a hostel and avoids the problems that occur when carried out manually. Identifying the drawbacks of the existing system led to designing a computerized system that will be compatible with the existing system with the system which is more user friendly and GUI oriented. We can improve the efficiency of the system, thus overcoming the drawbacks of the existing system. In this, the mess system is digitalized. The students can mark their attendance in online mode. The students can add their respective plans according to the given menu. They can also add their mobile number against their names. The name of the customer or student is displayed on the home screen of the application. The plan is also displayed on the home page of the application. The firebase is used to store the email and password of the customer or students. The language used to make this application is Kotlin. This language is easier than Java. That’s why about 9% of the developers are moving towards Kotlin because of its simple code. This application of mess system will help students to properly cooperate with the college system.

ACKNOWLEDGEMENT

We are highly grateful to Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the minor project work at Guru Nanak Dev Engineering College (GNDEC).

The constant guidance and encouragement received from Dr. Kiran Jyoti H.O.D. CSE Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Mrs. Lakhvir Kaur Grewal, without her wise counsel and able guidance, it would have been impossible to complete the project in this manner.

We express gratitude to other faculty members of the computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, we are indebted to all whosoever have contributed in this report work.

Harsharan Singh

Harsh Rathi

Jaskaran Singh

List of Figures

Table of Contents

Chapter 1: INTRODUCTION

1.1 Introduction to project

The Student Stay Hub is a comprehensive application designed to streamline and automate various aspects of hostel administration. It includes features such as student registration, room allocation, fee management, attendance tracking, and communication tools. This system enhances efficiency, reduces manual work, and provides real-time data for better decision-making in hostel management. It would provide better maintenance for the hostelers. Students can post complaints about the application itself. The students can register themselves and their data would be saved in the backend database. In our current era of automated systems with either software or hardware, it's not advisable to be using manual systems. Hostels without a management system are usually done manually. Registration forms verification and other data saving processes are done manually and most at times, they are written on paper. Thus, a lot of repetitions can be avoided with an automated system. The drawbacks of existing systems lead to the design of a computerized system that will help reduce a lot of manual input. With this system in place, we can improve the efficiency of the system, thus overcoming the drawbacks of the existing manual system. This system is designed in favor of the hostel management which helps them to save the records of the students about their rooms and other things. Mess is also an integral part of the hostel system. Nowadays, everything is becoming digital. So, we had added a new module in this application as "Mess Master". In this we had various modules or sections like adding customers and their specific plans. The plan is added in accordance with the time of the day like for Breakfast – B, Lunch – L and Dinner – D. The user can even update or delete the specific plan.

1.2 Project Category

This project basically falls in the application development category. A hostel management system falls squarely within the realm of application development. It involves creating software to solve specific problems related to managing hostel facilities, including tasks such as room allocation, student registration, fee management, inventory tracking, and possibly integration with other systems like attendance tracking or security systems. Since it's an application developed to address real-world needs and functions as a standalone system or service, it fits well within the category of application development. Specifically, it would be categorized as a specialized application tailored for hostel management.

- **Education:** Hostel management systems are commonly used in educational institutions like schools, colleges, and universities to manage accommodation for students. So, it makes sense under education.
- **Hospitality:** Since hostels provide accommodation services, you could categorize it under hospitality, although hostels are typically considered more budget-friendly accommodations compared to hotels.
- **Management Tools:** Hostel management systems are essentially management tools designed to streamline processes such as room allocation, fee management, attendance tracking, etc. Therefore, it could be categorized under management tools or utilities.
- **Software Development:** This category is more generic but encompasses any software project developed using Kotlin.

1.3 Objectives

1.3.1 To create an attendance system for effective management of the mess system.

1.3.2 To design a Polling system for mess menu modification and daily feedback.

1.3.3 To generate reports forecasting attendance records and menu provided on a monthly basis.

1.4 Problem Formulation

The Mess Management System is developed for automating the activities of mess. The software will be a great relief to the employees. This software will help users in case of reporting, registration and searching the information about residents and rooms. The aim of the Hostel Management System is to carry out the activities of Hostel in an efficient way. It will take the operations of Hostel to an upper level by providing faster access to data and allowing additional upgrade, modification, and deletion of data in a very systematic and reliable manners. The existing system is highly manual, involving a lot of paperwork and calculation and therefore, may be erroneous. This has led to inconsistency and inaccuracy in the maintenance of data. The data, which is stored on the paper only, may be lost, stolen or destroyed due to natural calamity like fire and water. The existing system is sluggish and consumes a lot of time causing inconvenience to students and the employees. Due to manual nature, it is difficult to update, delete, add or view the data. Since the number of students has drastically increased therefore maintaining and retrieving detailed records of every student is extremely difficult. Mess is also an integral part of the hostel system. Nowadays, everything is becoming digital. So, we had added a new module in this application as “Mess Master”. In this we had various modules or sections like adding customers and their specific plans. The plan is added in accordance with the time of the day like for Breakfast – B, Lunch – L and Dinner – D. The user can even update or delete the specific plan. The mess system being offline had different problems. Now the attendance of students will be easily kept on the record.

1.5 Recognition of Need

- All the work is done manually. Different copies of the student information are kept for different departments. Room is allotted according to the room requirements and other special facilities demanded by the student. Room categories: Single, Double, Air-Conditioned and Corner. Payment modes: Cash, Cheque and Draft. Hostel facilities and charges and other information are all kept in a booklet. Student information, staff information, fee records, student check-in and check-out, room status, staff's salary, all this information is kept in registers. All calculations relating to students' fees, staff salary, fines and penalties, hostel funds are done manually.
- The existing system is highly manual, involving a lot of paperwork and calculation and therefore may be erroneous. This has led to inconsistency and inaccuracy in the maintenance of data. The data, which is stored on the paper only, may be lost, stolen or destroyed due to natural calamity like fire and water. The existing system is sluggish and consumes a lot of time causing inconvenience to students and the employees.
- Due to manual nature, it is difficult to update, delete, add or view the data. Since the number of students has drastically increased, maintaining and retrieving detailed records of every student is extremely difficult. So, taking the above drawbacks, there was a need to propose an application for the hostel management system. The proposed system would help the students as well as the warden.

1.6 Existing System

- All the work is done manually. Different copies of the student information are kept for different departments. Room is allotted according to the room requirements and other special facilities demanded by the student. Room categories: Single, Double, Air-Conditioned and Corner. Payment modes: Cash, Cheque and Draft. Hostel facilities and charges and other information are all kept in a booklet. Student information, staff information, fee records, student check-in and check-out, room status, staff's salary, all this information is kept in registers. All calculations relating to students' fees, staff salary, fines and penalties, hostel funds are done manually.
- The existing system is highly manual, involving a lot of paperwork and calculation and therefore may be erroneous. This has led to inconsistency and inaccuracy in the maintenance of data. The data, which is stored on the paper only, may be lost, stolen or destroyed due to natural calamity like fire and water. The existing system is sluggish and consumes a lot of time causing inconvenience to students and the employees.
- Due to manual nature, it is difficult to update, delete, add or view the data. Since the number of students has drastically increased therefore maintaining and retrieving detailed records of every student is extremely difficult. So, taking the above drawbacks, there was a need to propose an application for the hostel management system. The proposed system would help the students as well as the warden.

1.7 Proposed System

- **Long-term storage of records:** Long-term storage of records in an online hostel management system involves designing a robust data storage and management strategy to ensure that data is securely stored, easily accessible, and reliably preserved for an extended period. Here are some key considerations:
- **Database Design:** Choose a suitable database management system (DBMS) for storing hostel records. Common choices include relational databases like MySQL, PostgreSQL or SQLite, or NoSQL databases like MongoDB or Cassandra, depending on your specific requirements. Design an efficient database scheme that organizes data into tables with appropriate relationships and indexes for quick retrieval.
- **Data Security:** Implement security measures to protect sensitive hostel records from unauthorized access, modification, or deletion. This may include role-based access control (RBAC), encryption of sensitive data, regular security audits, and compliance with relevant data protection regulations such as GDPR or HIPAA.
- **Backup and Disaster Recovery:** Set up regular backups of hostel records to prevent data loss in case of hardware failure, human error, or cyberattacks. Implement a robust disaster recovery plan to restore data quickly in the event of a catastrophic failure.
- **High accuracy in calculations:** Achieving high accuracy in calculations is crucial for a hostel management system to ensure that financial transactions, room allocations, and other numerical operations are carried out correctly. Here are some strategies to achieve high accuracy in calculations
- **Use Decimal Data Types:** Avoid using floating-point data types like floats or doubles for financial calculations, as they can introduce rounding errors due to their limited

precision. Instead, use decimal data types that provide fixed-point arithmetic with precise decimal representation, ensuring accurate calculations.

- **Implement Error-Checking Mechanisms:** Incorporate error-checking mechanisms into the calculation algorithms to detect and handle exceptions such as division by zero, overflow, or underflow. Implement robust validation checks to ensure that input data is within acceptable ranges and formats before performing calculations.

1.7Unique features of the proposed system

The proposed enhancements to the hostel management system aim to introduce unique features that differentiate it from conventional systems and elevate its functionality and presentation.

- **Smart Room Allocation:** Utilize algorithms or artificial intelligence to optimize room allocation based on factors such as student preferences, room availability, proximity to amenities, and compatibility among roommates. This feature ensures fair and efficient room assignments, leading to improved resident satisfaction and resource utilization.
- **Automated Maintenance Requests:** Implement automated maintenance request systems that allow residents to submit maintenance requests through the hostel management system or mobile app. Use sensors or IoT devices to detect maintenance issues automatically and generate service tickets for prompt resolution, enhancing resident satisfaction and operational efficiency.
- **Plan Management:**

  In this, you can see the name of the plan along with their price and the validity of days to which it will expire.
- **Customer Management:**

  In this, you will be shown the different customers who had recently added them along with their mobile numbers.
- **Front Page:**

  In this, you have different options like the candidate who had recently added their name and mobile number. At the bottom of the page, you have buttons for home, users manage and settings.

- **Main Page:**

  The main page serves as the entry point for users accessing mess system information. It has Login option for the registered users and sign up for the new candidates. You can fill your credentials to login in to your account and access the application or sign up with your unique email address as if in future you forget your password, you can easily change it.

Chapter 2: Requirement Analysis and System Specification

2.1 Feasibility Study

Feasibility is the measure of how beneficial / practical an information system will be to an organization. A feasibility study looks at the viability of an idea with an emphasis on identifying potential problems and attempts to answer one main question: Will the idea work and should you proceed with it. Feasibility studies are often conducted to assess the potential risks, benefits, and challenges associated with implementing the project. Ultimately, a project is considered feasible if it can be implemented successfully while meeting its objectives within the available resources and constraints. When assessing the feasibility of a project the following criteria can be identified:

Technical Feasibility

- Evaluate technical requirements, such as hardware, software & networking
- Infrastructure
- Assess compatibility with existing systems and technologies.
- Ensure the availability of skilled personnel for development and maintenance.

Economic Feasibility

- Estimate development costs, including software development and hardware.
- Licensing, and any training expenses.
- Analyze potential cost savings and benefits in the long run.
- Determine the return on investment (ROI) and payback period.

Operational Feasibility

- Identify how the system will integrate with current mess management process.
- Assess the impact on daily operations and workflow.
- Consider potential disruptions during the implementation phase.

2.2 Software Requirement Specifications

2.2.1 Data Requirement

Designing a hostel management system requires careful consideration of various data requirements to ensure that all aspects of hostel operations are efficiently managed. Here are some essential data requirements for a hostel management system application:

Student Information:

- Name
- Roll number or student ID
- Contact information (phone number, email)
- Address
- Gender
- Course/Program
- Year of study
- Emergency contact details

Room Information:

- Room number
- Room type (single, double, triple, etc.)
- Capacity (number of beds)
- Availability status (occupied or vacant)
- Facilities available in the room (e.g., attached bathroom, AC, etc.)

Hostel Building Information:

- Building name or block
- Address/location
- Floor plans

- Room allocation rules and regulations

Complaints and Suggestions:

- Complaint registration and tracking
- Resolution status
- Suggestions and feedback collection
- Plan Management
- Customer Management
- Settings

2.2.2 Functional Requirement:

User Authentication and Authorization:

- The system should allow users to authenticate themselves (e.g., login) using credentials such as username and password.
- Different user roles should be defined (e.g., administrator, warden, student) with specific access permissions.

Student Management:

- The system should allow administrators to add, edit, and delete student records.
- Students should be able to view and update their personal information.
- Student records should include details such as name, roll number, contact information, room allocation, etc.

Room Management:

- The system should manage room allocations, including assigning students to rooms and updating room availability.

- Warden or administrative users should be able to view room details, such as room type, capacity, occupancy status, etc.
- Room allocation rules should be configurable, such as maximum occupancy per room, gender restrictions, etc.

Maintenance and Housekeeping:

- The system should allow users to report maintenance issues and track their resolution status.
- Housekeeping schedules should be managed within the system, and staff should be able to update cleaning logs.
- Warden or administrative users should be able to assign maintenance tasks and track their progress.

Complaint Management:

- The system should allow students and staff to register complaints and track their resolution status.
- Warden or administrative users should be able to assign complaints to appropriate staff members for resolution.
- Complaints should be categorized and prioritized based on severity.

Plan Management:

- In this, you can see the name of the plan along with their price and the validity of days to which it will expire.

Customer Management:

- In this, you will be shown the different customers who had recently added them along with their mobile numbers.

Front Page

- In this, you have different options like the candidate who had recently added their name and mobile number.
- At the bottom of the page, you have buttons for home, users manage and settings.

Main Page:

- The main page serves as the entry point for users accessing mess system information.
- It has Login option for the registered users and sign up for the new candidates.
- You can fill your credentials to login in to your account and access the application or sign up with your unique email address as if in future you forget your password, you can easily change it.

Plan Options:

- In this, you can see three basic options like add plan, edit plan and remove plan etc. Likewise, you can use these three options to do the specific task.

Settings:

- In this, you see the name and account credentials of the registered user. Along with the App settings and account settings.
- You can use the notifications and data saver, change password and even reset application. There is option of “Logout” too.

2.2.3 Performance Requirement

Performance requirements in the Software Requirements Specification (SRS) document for a Hostel Management System (HMS) ensure that the system operates efficiently and meets specific performance criteria. Here are some performance requirements you might include:

Response Time:

- The system should respond to user interactions (e.g., login, room allocation) within X seconds under normal load conditions.
- Web pages should load within X seconds, with a maximum acceptable delay of Y seconds for peak load situations.

Throughput:

- The system should support a minimum of X simultaneous user connections without experiencing performance degradation.
- The system should process a minimum of Y transactions per minute during peak usage periods.

Scalability:

- The system architecture should be scalable to accommodate an increase in the number of users and data volume.
- The system should be capable of handling Z percent growth in user base over the next N years without significant performance degradation.

Resource Utilization:

- The system should utilize CPU, memory, and disk resources efficiently to ensure optimal performance.

- Memory usage should not exceed X percent of available system memory, and CPU usage should remain below Y percent under normal operating conditions.

Network Performance:

- The system should deliver acceptable performance over a range of network conditions, including low-bandwidth and high-latency connections.
- File uploads and downloads should be optimized to minimize transfer times and bandwidth consumption.

2.2.4 Dependability Requirement:

Dependability requirements in the Software Requirements Specification (SRS) document for a Mess Management System (MMS) outline the system's reliability, availability, maintainability, and security aspects. Here are some dependability requirements you might include:

Reliability:

- The system should operate reliably under normal and peak load conditions, ensuring consistent performance without unexpected failures.
- The mean time between failures (MTBF) should exceed X hours, where a failure is defined as any event that causes the system to become unavailable or perform incorrectly.
- The system should implement error handling mechanisms to detect and recover from faults gracefully, minimizing service disruptions and data loss.

Availability:

- The system should have a high level of availability, with downtime not exceeding X percent of total operating hours.
- The system should be designed with fault-tolerant components and redundancy to ensure continuous operation in the event of hardware or software failures.
- Scheduled maintenance activities should be performed during off-peak hours to minimize service interruptions for users.

Maintainability:

- The system should be designed for ease of maintenance and updates, with modular components that can be modified or replaced without impacting system functionality.
- Code should be well-documented and structured to facilitate understanding, debugging, and future enhancements by developers.
- System administrators should have access to tools and utilities for monitoring system health, diagnosing issues, and applying patches or updates as needed.

Security:

- The system should enforce access control mechanisms to prevent unauthorized access to sensitive data and functionality.
- User authentication and authorization should be implemented using strong encryption and industry-standard protocols to protect against unauthorized login attempts.

- Data transmission over networks should be encrypted using secure protocols (e.g., SSL/TLS) to prevent interception or tampering by malicious actors.
- The system should maintain audit logs of user activities and security events for monitoring, analysis, and forensic purpose

2.2.5 Maintainability Requirement:

Maintainability requirements in the Software Requirements Specification (SRS) document for a Hostel Management System (HMS) outline the system's ease of maintenance, updates, and modifications. Here are some maintainability requirements you might include:

Modularity:

- The system should be modularly structured, with distinct and cohesive components that can be modified or replaced independently without affecting the overall functionality.
- Changes to one module should not require extensive modifications to other parts of the system, promoting code reusability and reducing maintenance efforts.

Documentation:

- The system should be thoroughly documented, including design documents, code comments, user manuals, and maintenance guides.
- Code documentation should adhere to industry standards and best practices, providing clear explanations of algorithms, data structures, and implementation details.

- User manuals should provide comprehensive instructions on system usage, administration, troubleshooting, and maintenance tasks.

2.2.6 Security Requirement:

Security requirements in the Software Requirements Specification (SRS) document for a Hostel Management System (HMS) are essential to ensure confidentiality, integrity, and availability of sensitive data and system resources. Here are some security requirements you might include:

Authentication and Authorization:

- Users should be required to authenticate themselves using secure mechanisms such as passwords, biometrics, or multi-factor authentication (MFA).
- The system should implement role-based access control (RBAC) to restrict user access to authorized functionalities and data based on their roles and privileges.
- Passwords should be securely hashed and stored using industry-standard encryption algorithms to prevent unauthorized access in case of data breaches.

Data Confidentiality:

- Confidential information such as personal details, financial records, and administrative data should be encrypted during transmission and storage to prevent unauthorized access or interception.

- Access to sensitive data should be restricted to authorized users only, and data encryption keys should be securely managed and rotated regularly.

2.2.7 Look and Feel Requirement:

The "Look and Feel" requirements in the Software Requirements Specification (SRS) document for a Hostel Management System (HMS) outline the visual and interactive aspects of the user interface (UI) and user experience (UX). These requirements ensure that the HMS is intuitive, user-friendly, and aesthetically pleasing. Here are some examples of "Look and Feel" requirements for an HMS:

User Interface Design:

- The user interface should have a modern and clean design with intuitive navigation and visually appealing components.
- Consistent design elements, such as colors, fonts, icons, and layouts, should be used across all pages and modules of the application.
- The UI should be responsive and adaptable to different screen sizes and devices, including desktops, laptops, tablets, and smartphones.

Navigation and Menus:

- Navigation menus should be well-organized and easy to access, allowing users to quickly find and navigate to different sections and functionalities of the HMS.

- Drop-down menus, sidebars, and breadcrumbs can be used to provide hierarchical navigation and indicate the user's current location within the application.

2.3 SDLC Model Used

Agile model is chosen for the development of STUDENT STAY HUB due to its iterative and incremental approach, which aligns well with the project's dynamic nature and the need for continuous feedback and adaptation. The Agile model emphasizes collaboration between cross-functional teams, customer involvement throughout the development process, and the delivery of working software in short iterations known as sprints.

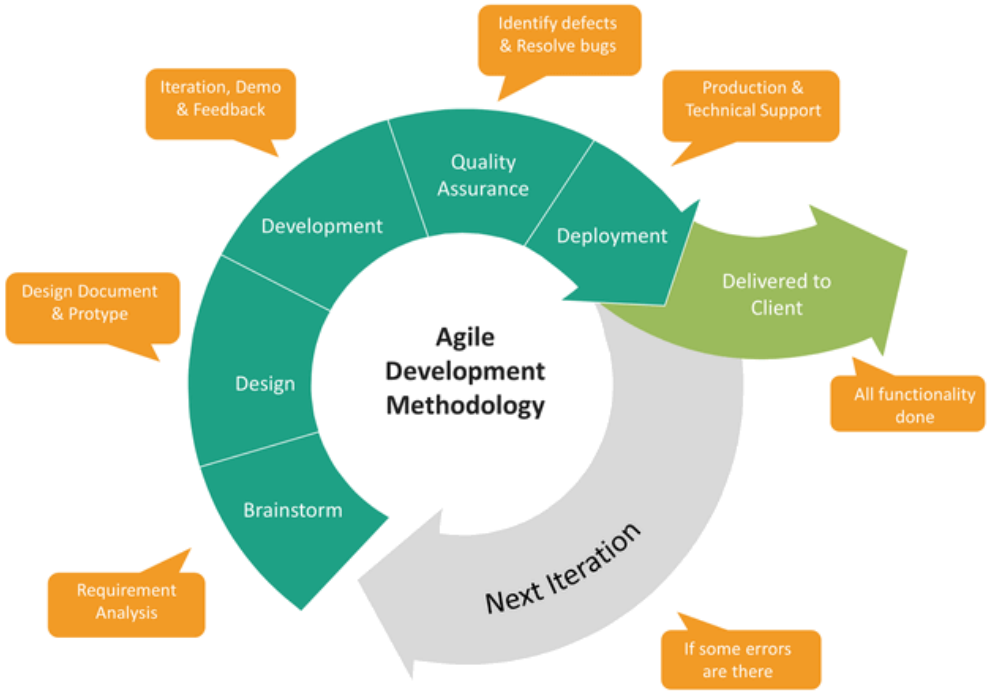


Figure 2.1 Agile SDLC

2.3.1 Key characteristics

- **Iterative Development:** The development process is divided into small iterations or sprints, typically lasting 1-4 weeks. Each sprint focuses on delivering a set of prioritized features or user stories, allowing for continuous improvement and adaptation based on feedback.

- **Flexibility and Adaptability:** Agile allows for changes to be made throughout the development process in response to evolving requirements or stakeholder feedback. This flexibility enables the project team to quickly respond to changing needs and priorities.
- **Customer Collaboration:** Customers or stakeholders are actively involved in the development process, providing feedback and guidance to ensure that the final product meets their expectations. Regular reviews and demonstrations during sprint reviews help to gather feedback and validate progress.
- **Cross-Functional Teams:** Agile promotes collaboration between cross-functional teams, including developers, designers, testers, and product owners. This collaboration fosters communication, transparency, and a shared understanding of project goals and priorities.
- **Continuous Integration and Testing:** Continuous integration practices ensure that new code is integrated into the main codebase frequently, allowing for early detection and resolution of integration issues. Automated testing is also emphasized to maintain software quality and reliability.

2.3.2 Phases:

- **Planning:** The project team collaborates with stakeholders to define project goals, scope, and priorities. The product backlog is created, consisting of user stories or features to be implemented. Sprint planning meetings are held to select user stories for the upcoming sprint.
- **Execution:** Development work begins with the implementation of selected user stories. Daily stand-up meetings are held to discuss progress,

identify obstacles, and adjust plans as needed. Continuous integration and testing are performed throughout the sprint to ensure quality and reliability.

- **Review and Retrospective:** At the end of each sprint, a sprint review meeting is held to demonstrate completed work to stakeholders and gather feedback. A sprint retrospective meeting follows to reflect on the sprint process, identify areas for improvement, and make adjustments for future sprints.
- **Delivery:** At the end of the project, the final product is delivered to the customer or stakeholders. The delivery phase may include additional testing, documentation, and training to ensure a smooth transition to the operational environment.

2.3.3 Benefits:

- **Flexibility:** Agile allows for changes to be made throughout the development process, accommodating evolving requirements and priorities.
- **Transparency:** Regular communication and collaboration foster transparency between the project team and stakeholders, ensuring alignment and shared understanding.
- **Quality:** Continuous integration and testing practices promote software quality and reliability, reducing the risk of defects and issues in the final product.
- **Customer Satisfaction:** Active involvement of customers or stakeholders throughout the development process ensures that the final product meets their expectations and requirements.

The Agile model provides a flexible and adaptive approach to software development, making it well-suited for the dynamic and evolving nature of the OSTLER project. By embracing Agile principles and practices, the project team can effectively manage project complexity, respond to changing requirements, and deliver a high-quality product that meets the needs of stakeholders and users. Agile models are based on iterative software development. An independent working module is built after the completion of iteration. Iteration should not consume more than two weeks to complete a code. Agile methodologies invite the developers to get involved in testing, rather than a separate quality assurance team. Agile methodologies are suitable in changing environments because of new practices and principles that enable a team to develop a product in short duration. The Agile Model was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project. Also, anything that is a waste of time and effort is avoided. The Agile Model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves.

Chapter 3: System Design

3.1 Design Approach

Designing a hostel management system in Kotlin involves several key considerations, including system architecture, database design, user interface, and integration of necessary functionalities. Here's a high-level design approach for building a hostel management system:

System Architecture:

- **Layered Architecture:** Adopt a layered architecture pattern such as Model-View-Controller (MVC) or Model-View-View Model (MVVM) to separate concerns and improve maintainability. The layers may include presentation layer (UI), business logic layer, and data access layer.
- **Modular Design:** Decompose the system into modular components to promote reusability and scalability. Identify core modules such as student management, room allocation, billing, inventory management, etc.
- **Microservices (Optional):** Consider implementing certain functionalities such as microservices if scalability and independent deployment are critical requirements.

Database Design:

- **Entity-Relationship (ER) Modeling**: Identify entities such as students, rooms, hostel staff, inventory items, etc. Define relationships between entities (e.g., a student may be allocated to a room).
- **Database Schema:** Design an appropriate relational database schema based on the ER model. Use Kotlin's support for ORMs (Object-Relational Mapping) like Room or Exposed for database interactions.

User Interface:

- **Responsive Design:** Ensure the user interface is responsive and accessible across different devices and screen sizes. Utilize frameworks like Android Jetpack Compose for building dynamic and modern UIs in Kotlin.
- **Intuitive Navigation:** Design an intuitive navigation flow to facilitate easy access to different functionalities. Use navigation components to manage navigation within the application.

Functionality Implementation:

- **Student Management:** Implement features for adding, editing, and deleting student records. Include functionalities for managing student details, such as personal information, contact details, etc.
- **Room Allocation:** Develop functionalities for allocating rooms to students based on availability and preferences. Implement features to manage room occupancy, check-in/check-out processes, etc.

3.2 System Design

In STUDENT STAY HUB, the frontend interface is designed to provide users with a seamless and intuitive experience. It incorporates modern web technologies such as Kotlin to create responsive and visually appealing user interfaces. Through the frontend, users can access various features of the application, including student, warden and complaint module.

Designing a hostel management system application in Kotlin involves considering various aspects, including system architecture, database design, user interface, and integration of functionalities. A well-designed system ensures scalability, maintainability, and reliability while meeting the specific requirements of hostel administrators and residents.

Data Flow Diagrams:

- The DFD illustrates the flow of data between various components of the system, including user interface, server, and database.
- It showcases how user inputs from the interface are transmitted to the server for processing and how the server retrieves relevant data from the database.
- Additionally, the DFD depicts the flow of information back to the user interface, presenting the results of the requested queries or actions.
- Data Flow Diagram (DFD) represents the flow of data within information systems.
- Data Flow Diagrams (DFD) provide a graphical representation of the data flow of a system that can be understood by both technical and non-technical users.
- The models enable software engineers, customers, and users to work together effectively during the analysis and specification of requirements.

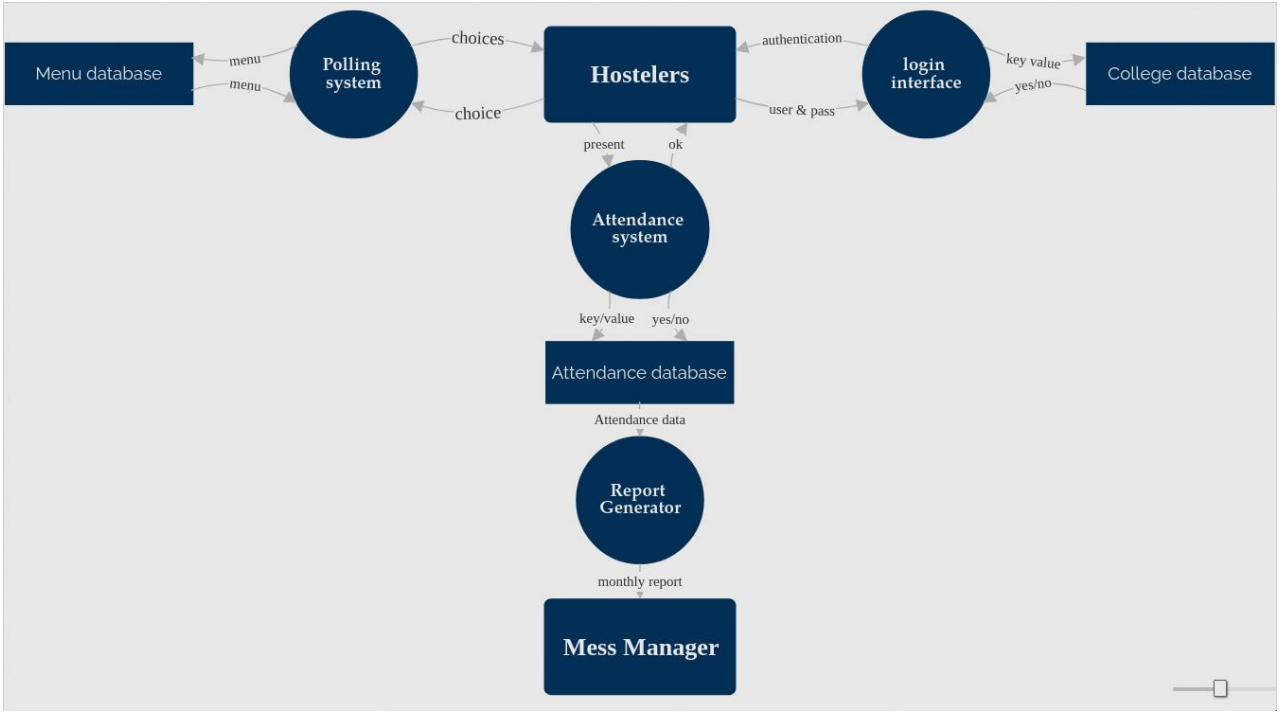


Figure 3.1 DFD

Process: Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence

Data Flow: Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.

Warehouse (Data Store): The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the

warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updating.

Terminator (External Entity): The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

Flowchart

The flowcharts are simple visual tools that help us understand and represent processes very easily. They use shapes like arrows, rectangles, and diamonds to show steps and decisions clearly. If someone is making a project or explaining a complex task, flowcharts can make complex ideas easier to understand. Flowcharts are the visual representations of an algorithm or a process. Flowcharts use symbols/shapes like arrows, rectangles, and diamonds to properly explain the sequence of steps involved in the algorithm or process. Flowcharts have their use cases in various fields such as software development, business process modeling, and engineering. Flowcharts are used due to the numerous amounts of benefits they provide. Below are some of the important reasons to use flowcharts. They provide clarity and simplification to the complex processes and algorithms, which in turn helps other people to understand them easily. Flowcharts provide a universal visual language that can be understood by anyone across different teams and helps reduce miscommunications. They are an optimal solution for documenting standard operating procedures, workflows, or business processes. This makes it easier to train new

employees. Flowcharts help in increasing the visualization of the problem being solved which enables more informed and data-driven choices.

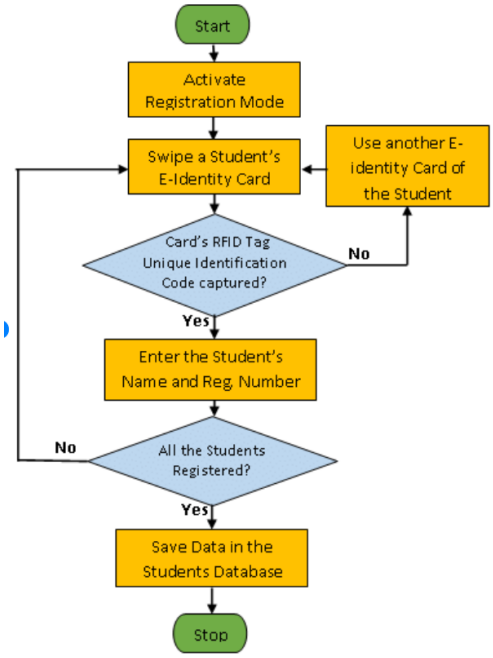


Figure 3.2 Flowchart

E-R Diagram

- An **Entity-Relationship Diagram (E-R Diagram)** is a visual representation used to model the structure of a database.
- It shows the relationships between different entities (objects) within the system and how these entities interact with each other.
- It was introduced by Peter Chen in 1976 as a way to visually describe the logical structure of databases.
- A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course.

- In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.
- An Entity Relationship Diagram is a diagram that represents relationships among entities in a database.
- It is commonly known as an ER Diagram. An ER Diagram in DBMS plays a crucial role in designing the database.
- Today's business world previews all the requirements demanded by the users in the form of an ER Diagram.
- Later, it's forwarded to the database administrators to design the database.

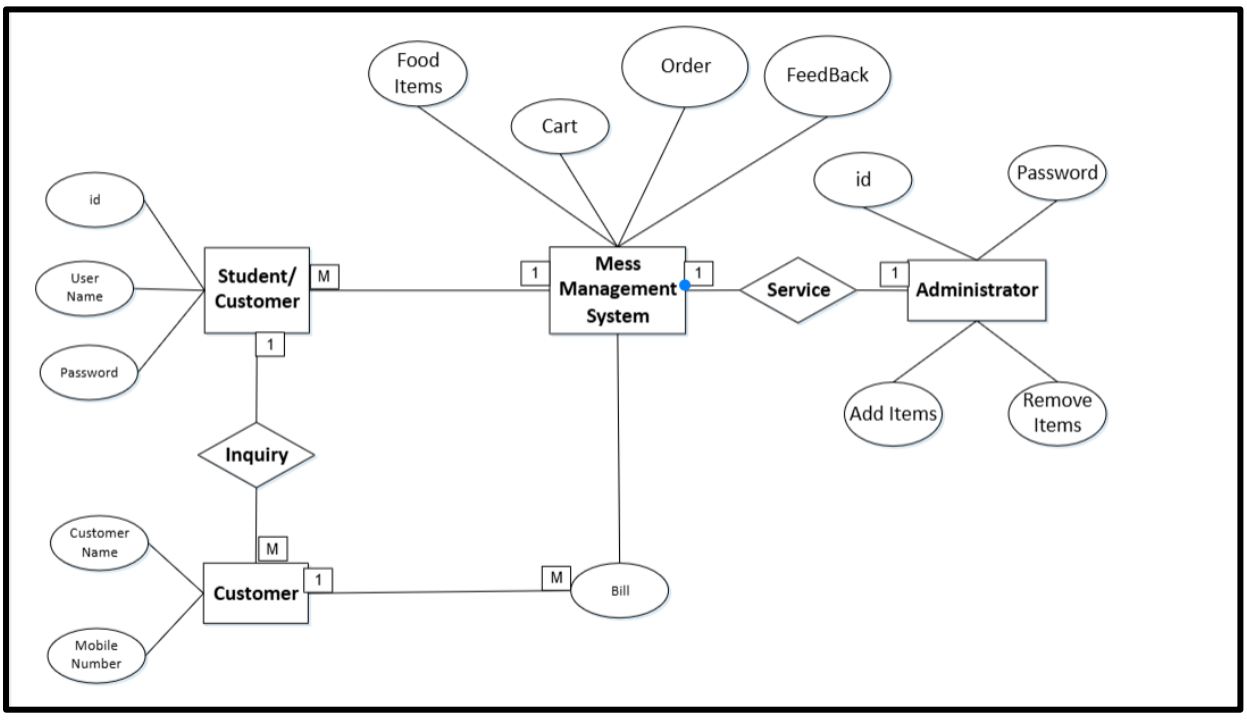


Figure 3.3 E-R Diagram

3.3 Methodology

Developing a mess management system in Kotlin requires a systematic approach to ensure efficiency, quality, and alignment with user requirements. Here's a methodology tailored for building such a system:

Requirements Gathering:

- **Stakeholder Interviews:** Engage with hostel administrators, staff, and residents to understand their needs and pain points.
- **Use Case Analysis:** Identify key use cases such as student registration, room allocation, billing, inventory management, etc.

System Design:

- **Architecture Design:** Define the system architecture, including layers, components, and technologies.
- **Database Design:** Design the database schema and entity relationships using Kotlin-compatible ORMs like Room or Exposed.
- **User Interface Design:** Create wireframes and UI mockups to visualize the application's layout and navigation flow.

Development:

- **Iterative Development:** Adopt an iterative development approach, breaking down the project into manageable tasks and implementing them incrementally.
- **Kotlin Coding Standards:** Follow Kotlin coding standards and best practices to maintain code consistency and readability.

- **Version Control:** Use version control systems like Git for collaborative development and tracking changes.

Testing:

- **Unit Testing:** Write unit tests for individual components and functionalities to ensure their correctness and reliability.
- **Integration Testing:** Conduct integration tests to verify the interaction between different modules and components.
- **User Acceptance Testing (UAT):** Involve stakeholders in UAT sessions to validate the system against user requirements.

Deployment and Release:

- **Continuous Integration (CI):** Set up automated build and test pipelines using CI/CD tools to ensure code quality and stability.
- **Deployment Automation:** Automate the deployment process to streamline the release of updates and new features.
- **Versioning:** Use semantic versioning to track releases and communicate changes to users and stakeholders.

Chapter 4: Implementation, Testing and Maintenance

4.1 Introduction to Languages, IDE's, Tools Used

4.1.1 Kotlin

Kotlin is an object-oriented language which also has a lot of functional programming elements. From the object-oriented side, it supports nominal subtyping with bounded parametric polymorphism (akin to generics) and mixed-site variance.

History:

Kotlin is a statically typed programming language developed by JetBrains, the company behind popular IDEs like IntelliJ IDEA, PyCharm. It was announced in 2011 and open-sourced in 2012 under the Apache 2.0 license. Kotlin was designed to address the limitations and challenges of existing programming languages, particularly Java, while also being fully interoperable with Java.

Syntax:

Kotlin's syntax is concise, expressive, and similar to other modern programming languages like Java, C#, and Scala. It draws inspiration from various programming paradigms, including object-oriented, functional, and procedural programming. Here's a brief overview of Kotlin's syntax:

Variable Declaration:

```
Val immutable Variable: Int = 10 // Immutable variable
var mutable Variable: String = "Hello" // Mutable variable
```

Functions:

```
fun greet (name: String): String {
   return "Hello, $name!"
```

```
}
```

Class Declaration:

```
class Person (Val name: String, var age: Int) {
   // Class members and methods
}
```

Null safety

```
var nullable String: String? = null // Nullable variable
```

Extension Functions

```
fun String.addExclamation(): String {
   return "$this!"
}
```

Elements: Variables and Constants: Kotlin supports both mutable and immutable variables using var and Val keywords, respectively. Functions: Functions are first-class citizens in Kotlin and can be passed as arguments, returned from other functions, or stored in variables.

Classes and Objects: Kotlin supports object-oriented programming with classes, objects, inheritance, and interfaces.

- **Null Safety:** Kotlin's type system distinguishes between nullable and non-nullable types, reducing the risk of null pointer exceptions.
- **Lambda Expressions:** Kotlin provides concise syntax for defining anonymous functions or lambda expressions.

- **Coroutines:** Kotlin introduces coroutines for asynchronous programming, allowing developers to write asynchronous code sequentially.

Attributes:

- **Interoperability:** Kotlin is fully interoperable with Java, allowing seamless integration with existing Java codebases and libraries.
- **Conciseness:** Kotlin's concise syntax reduces boilerplate code and improves readability, leading to increased developer productivity.
- **Null Safety**: Kotlin's null safety features help prevent null pointer exceptions, a common source of bugs in other programming languages.
- **Coroutines:** Kotlin's support for coroutines simplifies asynchronous programming and makes it easier to write concurrent and non-blocking code.
- **Tooling Support:** Kotlin is developed by JetBrains, which provides excellent tooling support in IntelliJ IDEA and Android Studio.

Advantages:

- **Concise Syntax:** Kotlin's concise syntax reduces boilerplate code, making codebases more readable and maintainable.
- **Null Safety:** Kotlin's type system prevents null pointer exceptions, improving code reliability and robustness.
- **Interoperability:** Kotlin is interoperable with Java, allowing gradual adoption and seamless integration with existing Java projects.
- **Coroutines:** Kotlin's coroutines simplify asynchronous programming, making it easier to write asynchronous and non-blocking code.

- **Tooling Support:** Kotlin is supported by JetBrains, offering excellent IDE support in IntelliJ IDEA and Android Studio.

Disadvantages:

- **Learning Curve:** Although Kotlin's syntax is similar to other modern programming languages, there may be a learning curve for developers unfamiliar with its features and idioms. Community and Ecosystem: While Kotlin's community is growing rapidly, it may not have as extensive a library ecosystem as more established languages like Java or Python.
- **Compilation Time:** Kotlin's compilation time may be slower compared to Java, particularly for larger codebases, although improvements are continuously made in this area.

Despite these disadvantages, Kotlin's advantages in terms of conciseness, null safety, interoperability, and tooling support make it a compelling choice for many developers, particularly those working on Android, backend, and multiplatform projects.

4.1.2 IDE’s

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development. It's based on JetBrains' IntelliJ IDEA software and tailored specifically for Android development. Android Studio provides a comprehensive set of tools and features to streamline the app development process.

Features:

- **Code Editor:** Android Studio offers a feature-rich code editor with syntax highlighting, code completion, and code refactoring capabilities.
- **Gradle Build System:** Android Studio utilizes the Gradle build system for building, testing, and packaging Android apps.
- **Emulator:** Android Studio includes an emulator that simulates Android devices for testing and debugging purposes.
- **Debugging Tools:** Android Studio provides robust debugging tools for identifying and fixing bugs in Android apps.
- **Performance Profiler:** Android Studio includes a performance profiler for analyzing app performance metrics such as CPU usage, memory allocation, and network activity.
- **Version Control Integration:** Android Studio seamlessly integrates with version control systems like Git for managing source code repositories.
- **Templates and Wizards:** Android Studio offers templates and wizards for quickly creating new Android projects, activities, fragments, and other components.

- **Localization Tools:** Android Studio includes tools for managing app translations and supporting multiple languages.
- **Google Play Integration:** Android Studio integrates with Google Play services for publishing apps to the Google Play Store.

Advantages:

- **Rich Feature Set:** Android Studio offers a comprehensive set of tools and features tailored specifically for Android development, streamlining the entire development lifecycle.
- **Performance:** Android Studio is optimized for performance, providing fast build times, responsive code editing, and efficient resource management.
- **Community Support:** Android Studio benefits from a large and active developer community, with extensive documentation, tutorials, and online resources available to help developers troubleshoot issues and learn new skills.
- **Integration with Google Services:** Android Studio seamlessly integrates with various Google services and APIs, making it easy to incorporate features like Google Maps, Firebase, and Google Cloud Platform into Android apps.
- **Constant Updates:** Android Studio receives regular updates and enhancements, ensuring compatibility with the latest Android SDK versions, platform updates, and development tools.
- **Cross-Platform Development:** Android Studio supports cross-platform development with Kotlin Multiplatform, enabling developers to share code between Android, iOS, and other platforms.

4.2 Testing Techniques

Testing is a crucial process in software development that involves evaluating a system or application to identify defects, errors, or discrepancies between expected and actual behavior. Testing ensures that the software meets quality standards, functions correctly, and satisfies user requirements. For testing, black-box testing is done. Black- box testing means examining the functionality of the system without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit testing, integration system and acceptance. The black box is a powerful technique to check the system under test from the user's perspective. The testing team does not cover the inside details such as code, server logic, and development method.

So, the testing involves the following steps:

- **Unit testing**: The individual UI of each screen was separately tested.
- **Integration Testing**: Two or more screens were then combined and examined to find out whether they are working functionally correct when combined with each other.
- **System Testing**: Different modules together combined and checked to find out if there is any bug in its working. After this, the whole software was tested as a whole.
- **Acceptance Testing**: Software was then given to the third party for testing.

4.3 Test Cases

- **Valid Plan Added:**

  Test Case: User will add their plan.

  Expected Result: The system will add the plan and give an alert “plan added”. Breakfast as B, Lunch – L and for Dinner – D. The plan will be added as: BL – Parantha, D – Dessert etc.

- **Invalid Plan Added:**

  Test Case: We will check if they had added the plan in the correct way or not.

  Expected Result: The system would check if the students had added the plan in the right way.

- **Invalid Login**

  Test Case: The student had tried to login with another account.

  Expected Result: The system would check for the account. If the student had used another account to log in then it would come up with the message “Invalid Login”.

- **Valid Login:**

  Test Case: The user can login with their credentials.

  Expected Result: The system would check for the account. If the student had used the correct credentials to log in then they can successfully access their account.

Chapter 5: Results and Discussions

5.1 User Interface Representation

In STUDENT STAY HUB, the user interface (UI) is designed to provide seamless and intuitive experience for users interacting with the hostel management system. Here's a representation of the UI components: Overall, the UI design of Ostler prioritizes user convenience, accessibility, and clarity, ensuring that users can efficiently access the desired hostel services with ease.

5.1.1 Description of Modules

- **Main Page:**

  The main page serves as the entry point for users accessing mess system information. It has a Login option for the registered users and signs up for the new candidates. You can fill in your credentials to login into your account and access the application or sign up with your unique email address as if in future you forget your password, you can easily change it.

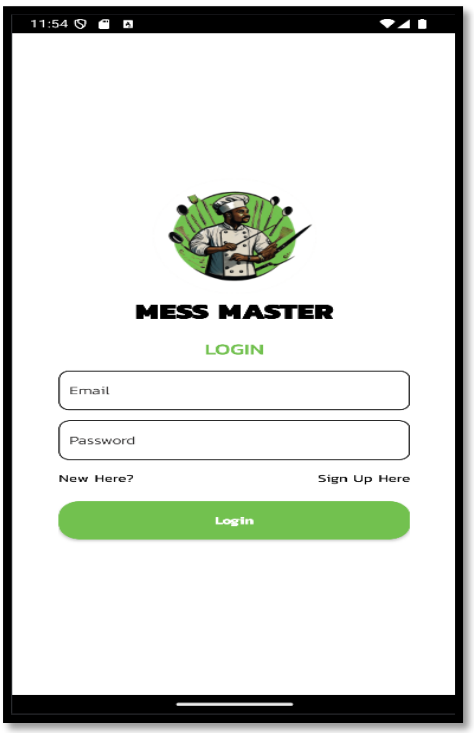


Figure 5.1 Main Page

- **Front Page**

  In this, you have different options like the candidate who has recently added their name and mobile number. At the bottom of the page, you have buttons for home, users manage and settings.

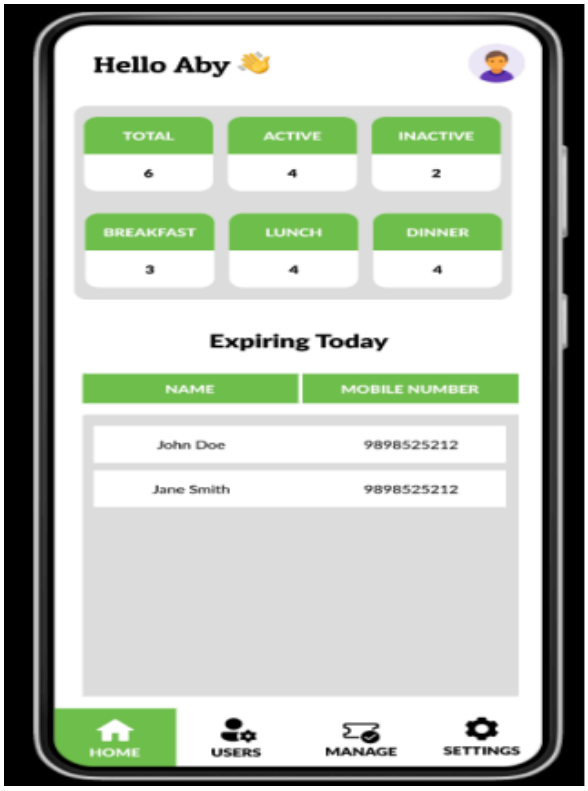


Figure 5.2 Front Page

- **Customer Management**

  In this, you will be shown the different customers who have recently added them along with their mobile numbers.

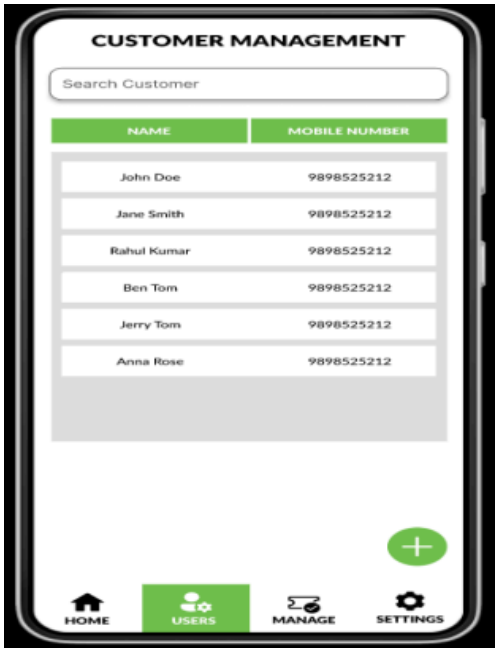


Figure 5.3 Customer Management

- **Plan Management**

  In this, you can see the name of the plan along with their price and the validity of days to which it will expire.

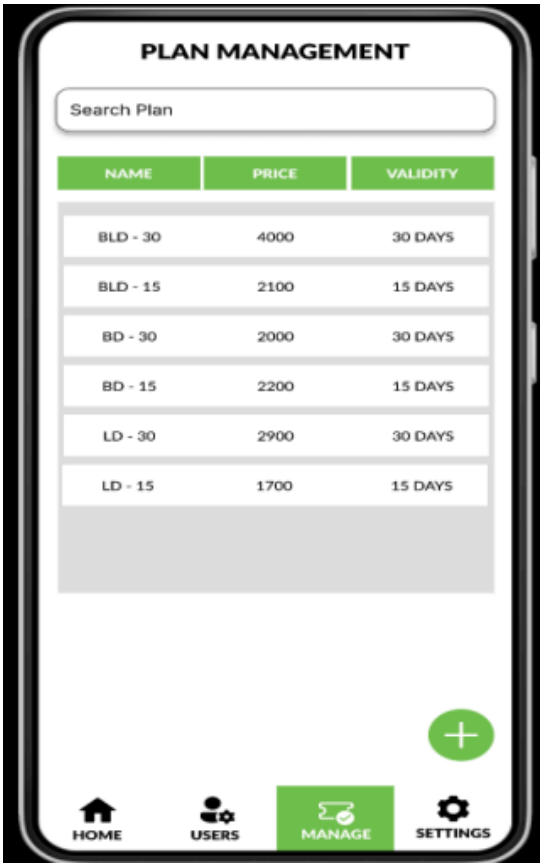


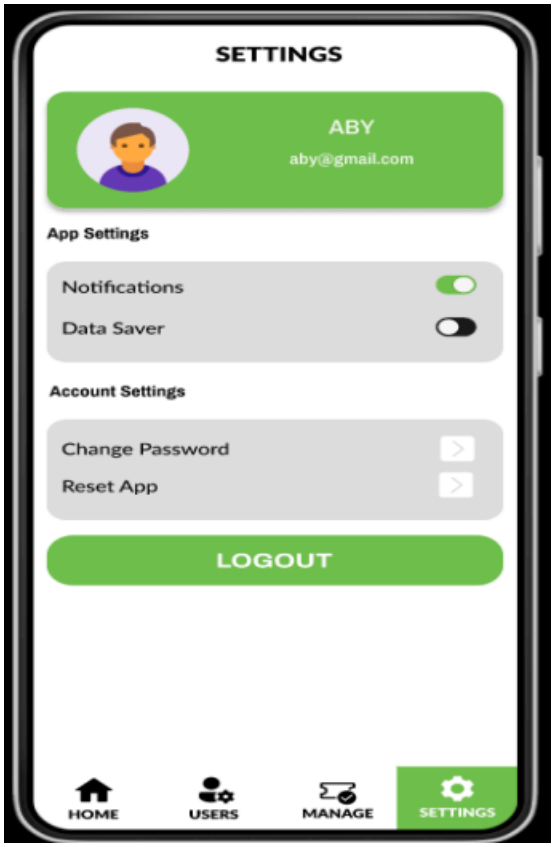
Figure 5.4 Plan Management

- **Plan Options**

  In this, you can see three basic options like add plan, edit plan and remove plan etc. Likewise, you can use these three options to do the specific task.

- **Settings**

  In this, you see the name and account credentials of the registered user. Along with the App settings and account settings. You can use the notifications and data saver, change password and even reset application. There is option of "**Logout**" too.



Fiure 5.5 Settings

5.2 Back End Representation

Kotlin is preferred by many developers for its conciseness, safety features, and interoperability with Java. Coupling it with Firebase, a comprehensive suite of mobile development tools, allows developers to achieve more with less code.

To use Firebase with Kotlin, first, ensure that you have Android Studio installed (version 3.0 or later recommended). Start a new project and choose Kotlin as the programming language.

Next, integrate Firebase into your project. Here are the steps:

In Android Studio, click on Tools -> Firebase.

- You'll see the assistant panel on the right. Select the Firebase service you want to add (e.g., Authentication, Fire store, Analytics, etc.).
- Click on the provided link to connect your app to Firebase, then click "Add Firebase SDK" to finish the setup.

Firebase Authentication with Kotlin

Firebase Authentication is a crucial service that provides backend services, ready-made UI libraries, and easy-to-use SDKs to authenticate users to your app. With Firebase's powerful features and Kotlin's robustness, Android app development becomes a more streamlined and enjoyable process. This guide covers only the basics of using Kotlin with Firebase, but the possibilities are endless. Whether it's implementing push notifications with Firebase Cloud Messaging or analyzing user behavior with Google Analytics, Kotlin and Firebase provide all the tools you need to build high-quality Android apps.

Remember, as with any tool or technology, the key to mastery lies in consistent practice and exploration. Keep building, keep iterating, and soon, you'll be adept at using Kotlin and Firebase for your Android app development journey.

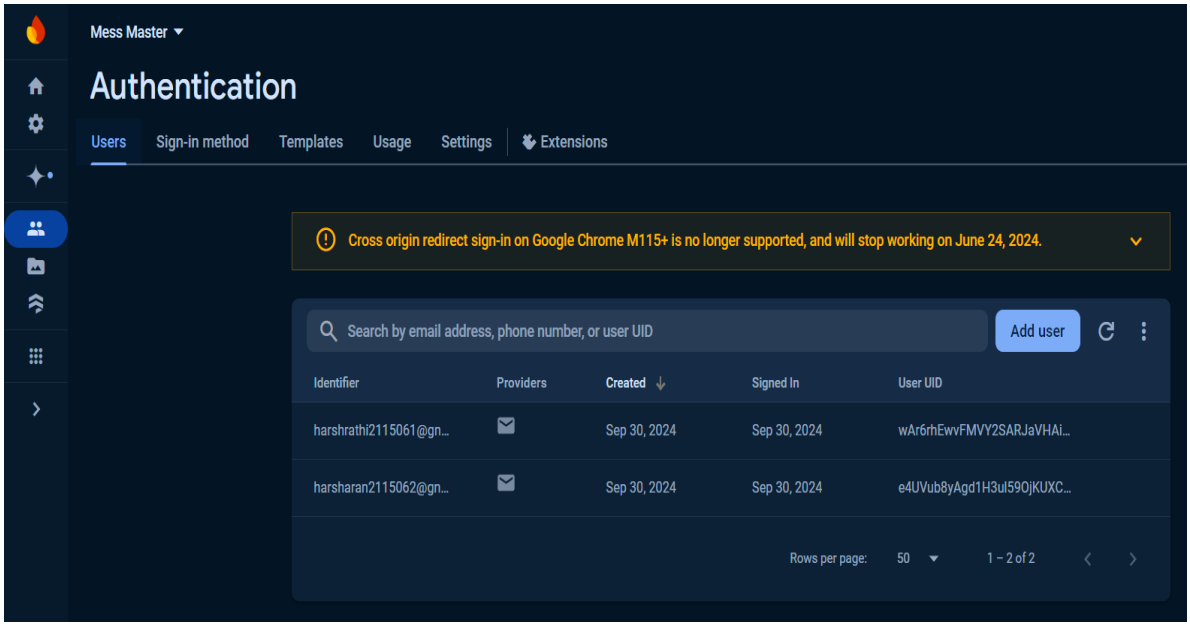


Figure 5.6 Firebase

Chapter 6: Conclusion and Future Scope

6.1 Conclusion

In conclusion, developing a hostel management system application in Kotlin offers a robust and scalable solution for efficiently managing student accommodation in educational institutions. By leveraging Kotlin's versatile features, including its concise syntax, null safety, interoperability with Java, and strong tooling support, developers can build a modern and reliable application tailored to the specific needs of hostel administrators and residents. The application's future scope is promising, with opportunities for integration with emerging technologies such as IoT, mobile platforms, data analytics, and cloud computing. Additionally, the customizable and scalable nature of Kotlin allows for the implementation of advanced features, such as automation, customization, and integration with educational ERP systems, to enhance operational efficiency and user experience. As educational institutions continue to embrace digital transformation, hostel management systems play a vital role in streamlining administrative processes, optimizing resource utilization, and improving the overall accommodation experience for students. By choosing Kotlin as the development language, institutions can benefit from its modern features, extensive community support, and compatibility with industry-standard tools and frameworks, ensuring the long-term success and sustainability of the hostel management system application.

6.2 Future Scope

The future scope of a hostel management system application made in Kotlin is promising, considering the ongoing digital transformation in various sectors, including education and accommodation management. Here are some potential future developments and opportunities for such an application. Adoption in Educational Institutions: Hostel management systems are increasingly becoming essential tools for educational institutions to efficiently manage student accommodation. As more schools, colleges, and universities recognize the benefits of digitizing their hostel management processes, the demand for robust and scalable solutions like the one developed in Kotlin will continue to rise. Integration with IoT and Smart Devices: Future hostel management systems may integrate with IoT (Internet of Things) devices and smart sensors to automate tasks such as room access control, energy management, and facility monitoring. Kotlin's versatility and compatibility with emerging technologies make it suitable for implementing such integrations. Enhanced Mobile Experience: With the growing reliance on mobile devices, hostel management systems are likely to evolve to provide enhanced mobile experiences for both administrators and residents. Kotlin's support for Android app development ensures that the application can offer seamless performance and user-friendly interfaces on mobile platforms. Data Analytics and Insights Hostel management systems can leverage data analytics and machine learning techniques to generate insights into occupancy trends.

REFERENCES

[1] Mr. A.Aswar, Mr. A.Ganesan, Dr. V.Kavitha, Mr. V.Karthicksabri PG & Research Department of Computer Applications, on Hostel Automation system.

[2] Development of an Automated Hostel Facility Management System Journal of Science and Engineering

[3] SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE) – volume 3 Issue 4–April 2016 ISSN: 2348 – 8387 www.internationaljournalssrg.org

[4] Online Hostel Management International Journal of Advanced Engineering & Science Research (IJAES) Volume 5, Issue 1, March 2017

[5] SOFTWARE FOR HOSTEL MANAGEMENT SYSTEM Master's Degree student, 2) PhD., Lecturer

[6] HOSTEL MANAGEMENT SYSTEM USING SERVICE NOW European Journal of Molecular & Clinical Medicine, 2020, Volume 7, Issue 4, Pages 1078-1084

[7] Mastering the Management System by Robert S. Kaplan and David P. Norton

[8] Lore: a database management system for semi structured data Publication
SIGMOD Record

[9] HOTEL FRONT OFFICE MANAGEMENT THIRD EDITION James A. Bardi, Ed.D, CHA The Pennsylvania State University

[10] Measuring Hotel Guest Satisfaction by Using an Online Quality Management System Kesh Prasad, Philip W. Wirtz & Larry Yu