Department of Computer Science

University of Gujrat

FINAL YEAR PROJECT

TITLE: Class Room Management System

Session : BSCS-19-B (2019-2023)

Project Advisor: Dr. Abdur Rehman

Submitted By

Bilal Saif	19011519-058
Muhammad Irfan Khan	19011519-069

STATEMENT OF SUBMISSION

This is certified that **Bilal Saif** Roll No. **19011519-058** and **Muhammad Irfan khan** Roll No. **19011519-069** has successfully completed the final year project named as **ClassRoom Management System** at the Department of Computer Science, University of Gujrat, to fulfill the requirement of the degree of **bachelor's in computer science**.

Project Supervisor

Project Coordination Office
Faculty of C&IT -UOG

Chairperson
Department of Computer Science

Acknowledgement

We truly acknowledge the cooperation and help make by Dr, Abdur Rehman, Chairman, Department of Computer Science, University of Gujrat. He has been a constant source of guidance throughout the course of this project. We would also like to thank for his help and guidance throughout this project. We are also thankful to our friends and families whose silent support led us to complete our project.

Date: 03 Sep 2023.

Abstract

Classroom Management System is a mobile application that aims to simplify the process of managing classrooms, schedules, and attendance for students, faculty, and administration. It provides a centralized platform for managing all the classroom-related data such as student and faculty information, class schedules, and room availability.

With the Classroom Management System, students can easily view their class schedules, access lecture notes, and stay informed about makeup lectures. Faculty members can manage their query about classrooms availability, while the administration can oversee the overall room management system and view free rooms. The application's primary focus is on room management, which helps students, faculty, and administration to find and book available rooms for various activities. This system ensures efficient use of resources and reduces the hassle of manual room scheduling. In summary, Classroom Management System is an efficient and user-friendly platform that streamlines classroom-related activities and enhances the overall learning experience for students and faculty alike.

Table of Contents

Chapter 1: Project Feasibility Report

1.1. Introduction

1.2. Project/Product Feasibility Report

Classroom Management System is a mobile application that aims to simplify the process of managing classrooms, schedules, and attendance for students, faculty, and administration. It provides a centralized platform for managing all the classroom-related data such as student and faculty information, class schedules, and room availability.

With the Classroom Management System, students can easily view their class schedules, access lecture notes, and stay informed about makeup lectures. Faculty members can manage their query about classrooms availability, while the administration can oversee the overall room management system and view free rooms.

The application's primary focus is on room management, which helps students, faculty, and administration to find and book available rooms for various activities. This system ensures efficient use of resources and reduces the hassle of manual room scheduling.

In summary, Classroom Management System is an efficient and user-friendly platform that streamlines classroom-related activities and enhances the overall learning experience for students and faculty alike.

1.2.1. Technical Feasibility

The technical feasibility of our project is an important aspect to consider. Here are some key points to address in the Technical Feasibility section of the project report:

1. **Technology Stack:** The technology stack are mentioned Flutter and Firebase, which are good choices for building a cross-platform application with real-time data capabilities.

2. **Development Tools:** The development tools we are using, such as IDEs (Integrated Development Environments), version control systems, and any additional libraries or packages that we'll incorporate into my Flutter application.

3. **Cross-Platform Capability:** Flutter's cross-platform capabilities will benefit in the project. |This will ensure a consistent user experience across different platforms (iOS and Android) and we are going to handle the platform-specific features.

4. **Firebase Integration:** The plan to integrate Firebase services into the application. the specific Firebase services we are using, such as Firebase Fire store for storing data, Firebase Authentication for user management.

5. **Real-Time Updates:** The Firebase's real-time capabilities will be leveraged in the application. In our case, we have mention that real-time updates to the timetable will help users see room availability in real-time.

6. **Database Design:**Provide a high-level overview of the database structure. The organize data related to classrooms, timetable, makeup lectures, and room availability. Er are using Firebase Firestore's database to store this structured data.

7. **User Interface:** we design the user interface of the application using Flutter. Consider including wireframes or mockups to visualize the different screens and user interactions.

8. **Authentication and Authorization:** The implement user authentication and authorization using Firebase Authentication.

9. **Scalability:** The system will handle a growing number of users and data. Firebase's scalable infrastructure can help manage increased user loads.

10. **Security:** Touch upon the security measures we are implementing to protect user data and ensure the application's integrity.

11. **Testing:** The testing strategy, including unit testing, integration testing, and potentially user acceptance testing. The importance of testing for ensuring a robust and bug-free application.

12. **Deployment:** Deploy of the application to both Android and iOS platforms. Work on both IOS and android. Considerations for app store guidelines and requirements.

1.2.2. Operational Feasibility

Operational feasibility is an important aspect of project feasibility that we need to consider. Operational feasibility assesses whether the proposed system can be smoothly integrated into the current operations and whether it will be convenient and effective for the users. Here are some points we could consider addressing in the Operational Feasibility section of our feasibility report:

1. **User Adoption and Acceptance:** the teachers, students, and administrative staff would be willing to adopt and use the new system. This could be influenced by factors such as user-friendliness, ease of learning, and the value the system brings to their tasks.

2. **Training and Support:** The training needs for users this help to interact with the system. There is a need for training sessions, user guides, or online tutorials to help users navigate and utilize the system effectively?

3. **Change Management:** The smoothly the transition from the current manual or existing system to the new system is managed. Any potential resistance to change through management.

4. **Integration with Existing Processes:** The proposed Classroom Management System will fit into the existing educational institution's processes. This seamlessly integrate with current scheduling and management procedures, or there be disruptions.

5. **Infrastructure and Technical Requirements:** The current technical infrastructure, such as devices and internet connectivity, supports the usage of the new system that the system's technical requirements are feasible within the institution's environment.

6. **Data Migration and Integration:** If applicable, data from the current system will be migrated to the new system. Any challenges related to data integration will be handled through planning.

7. **Resource Availability:** The institution has the necessary resources, both in terms of personnel and technology, to maintain and support the system after its deployment.

8. **Feedback and Iteration:** The plan for gathering user feedback after the system is deployed. We will incorporate user suggestions and address any issues that may arise.

9. **Scalability:** The system can handle an increase in users, classrooms, and other resources as the institution grows.

10. **Risk Mitigation:** The potential operational risks, such as system downtime, data loss, or user dissatisfaction, and the plan to mitigate these risks.

1.2.3. Economic Feasibility

Economic Feasibility refers to assessing whether the project is financially viable and whether the benefits outweigh the costs. Here's how we can approach this aspect of our project:

Costs:

1**. Development Costs**: The costs associated with developing our application. This includes expenses for software, hardware, third-party libraries, or plugins, and possibly hiring developers (if applicable).

2. **Maintenance Costs**: Then ongoing costs for maintaining servers, updating the app, and addressing any bugs or issues that may arise after deployment.

3. **Training Costs:** (if applicable). The need for training sessions to introduce users (student, teachers, administrators) to the new system, factor in those costs.

4. **Infrastructure Costs:** The expenses related to hosting the app, using Firebase services, and any other cloud services are plan to utilize.

Benefits:

1. **Time Savings:** The time savings that the system can provide to teachers and administrators by automating tasks like timetable management, makeup lecture scheduling, and room availability tracking.

2. **Efficiency Gains:** The system's features can lead to increased efficiency in managing classrooms, makeup lectures, and room bookings.

3. **Reduced Errors:** The system can reduce human errors that might occur during manual timetable and makeup lecture scheduling.

4. **Improved Resource Utilization:** The system can help optimize room utilization, leading to potential cost savings by better allocating resources.

5. **User Satisfaction:** The positive impact on user satisfaction due to the ease of use and features provided by the application.

Financial Metrics:

1. **Return on Investment (ROI**): The potential ROI by comparing the total benefits over a defined period with the total costs. This can help to determine whether the project is financially viable.

2. **Payback Period**: The time it will take for the project to generate enough benefits to cover its costs. This is payback period.

3. **Net Present Value (NPV):** The present value of future benefits minus the costs, the time value of money. A positive NPV indicates a potentially profitable project.

4. **Break-Even Analysis:** The point at which the benefits equal the costs, resulting in neither profit nor loss.

Risk Analysis:

1. **Identify Risks:** The potential risks that could impact the economic feasibility of the project, such as unexpected development delays, changes in technology trends, or low user adoption.

2. **Mitigation Strategies**: The plan to mitigate these risks to ensure the project's financial success. Is a mitigation strategy.

Conclusion:
Based on the analysis of costs, benefits, and financial metrics, conclude whether the project is economically feasible. If the benefits significantly outweigh the costs and the financial metrics indicate a positive outcome, it suggests that our Classroom Management System project holds economic viability.

Remember that this is a high-level guide, and we should tailor it to our specific project and situation. Additionally, economic feasibility is just one aspect of a comprehensive feasibility study. Make sure to consider other aspects like technical feasibility, operational feasibility, and legal/ethical considerations as well.

1.2.4. Schedule Feasibility

1. **Timeline and Milestones:** The project's timeline, including start and end dates, as well as important milestones. Break down the development process into phases, such as planning, UI design, backend development, integration, testing, and deployment.

2. **Scope and Complexity:** The complexity of the project. Consider the features we're planning to implement, the integration with Firebase, and the user interface design. Complex features might require more time and effort, potentially affecting the schedule feasibility.

3. **Resources:** The availability of resources, including our own time, team members (if applicable), and external resources.we have sufficient time to dedicate to the project, considering our academic commitments.,

4. **Learning Curve:** we mentioned using Flutter and Firebase, consider the learning curve associated with these technologies. If we or our team members are not familiar with these tools, it might take additional time to learn and implement them effectively.

5. **Unforeseen Challenges:** Projects often encounter unexpected challenges or roadblocks. Factor in some buffer time to address unforeseen issues, such as debugging, unexpected design changes, or Firebase integration complexities.

6. **Testing and Iteration:** The testing phase can sometimes reveal issues that need to be addressed before deployment. Account for time needed for testing, bug fixes, and iteration.

7. **Documentation and Presentation:** To allocate time for preparing documentation and presentations, which are essential components of a final year project.

8. **Parallel Tasks:** Tasks that can be worked on simultaneously. For example, while the UI design is being finalized, backend development can be started. Areas where parallel work can be beneficial to expedite the project.

9. **Time Management:** Time management skills and well we are balance our academic commitments with the project work. Avoid overcommitting and ensure a realistic allocation of time for both.

10. **Gantt Chart:** A Gantt chart is a visual representation of the project schedule. It helps in understanding task dependencies and timelines. Consider creating a Gantt chart to depict our project's schedule.

It's important to be realistic in our assessment and to build some flexibility into the schedule to accommodate unexpected challenges. Remember that project schedules can be dynamic and might need adjustments as the project progresses.

1.2.5. Specification Feasibility

The Specification Feasibility section of the Project/Product Feasibility Report is crucial as it outlines the technical and functional requirements of our project. Let's break down this section further:

1. **User Requirements:** Clearly our users (students, teachers, administrators) expect from the application. For instance, users should be able to view the timetable, schedule makeup lectures, check room availability, and differentiate between occupied and empty rooms.

2. **Functional Requirements:** List the specific functionalities our application will offer, such as:
   - Creating and managing makeup lectures.
   - Displaying the timetable.
   - Indicating room availability through color-coded markers.

3. **Technical Requirements:** Specify the technical aspects of our application, such as:

   - Programming languages and frameworks (Flutter, Firebase).
   - Database design (Firebase Firestore or Realtime Database) for storing timetables, makeup lectures, room statuses, etc.
   - User authentication and authorization mechanisms.
   - Integration of date and time picker widgets for selecting makeup lecture date and time.
   - UI/UX design considerations for a user-friendly interface.

4. **Data Flow:** The data will flow within the application. For instance:
   - When a user creates a makeup lecture request, it will be stored in the database.
   - The timetable page will retrieve data from the database to display room statuses.

5. **UI/UX Design:** The user interface are designed to facilitate easy navigation, clear presentation of information, and effective user interaction. color schemes and design elements that help users distinguish between room statuses.

6. **Constraints:** The potential limitations we might face, such as time constraints, device compatibility, or resource limitations (e.g., Firebase usage costs).

7. **Performance Considerations:** The plan to optimize the application's performance, especially if it involves real-time data updates and synchronization.

8. **Scalability:** The system will scale as the number of users and data increases. Firebase generally offers good scalability options, but it's important to plan for potential bottlenecks.

9. **Security and Privacy:** The user data will be stored securely and we are handling user authentication and authorization.

10. **Testing Strategy:** The testing plan, including unit testing, integration testing, and user acceptance testing. Our app functions as expected under various scenarios.

11. **Development Timeline:** A rough timeline for different phases of development, including research, design, implementation, testing, and documentation.

Remember that the Specification Feasibility section should provide a clear and comprehensive understanding of help our application will function and meet user requirements. It's a critical part of our feasibility report that guides the development process.

1.2.6. Information Feasibility

"Information Feasibility" section of our Project/Product Feasibility Report. This section focuses on determining whether the information required for our project is accessible, available, and accurate. Here's how we can structure this part of our report:

Data Sources and Availability

In this section, The sources from which we'l be gathering the necessary data for our Classroom Management System. Since our project involves creating and managing timetables, we need to ensure that the required data is available and accessible. In our case, Firebase can act as our data source.

1. **Firebase Realtime Database:** The Firebase will be used to store timetable information. its real-time synchronization capabilities and it can be accessed by the application to display accurate data to users.

Data Accuracy and Consistency
For a classroom management system, accurate and consistent data is crucial to provide reliable information to users.

1. **Data Validation:** The methods we'll implement to ensure that the data entered into the system is accurate and valid. For example, validation checks on dates, subjects, and room availability.

2. **Error Handling:** The errors and inconsistencies are handled. This might involve providing informative error messages to users when they try to create conflicting schedules or when data fails to be saved in the database.

Data Visualization
The ability to visualize data effectively is important to convey information clearly to users.

1. **Room Availability Visualization:** we'll visually represent room availability on the timetable. The color coding (red for empty, green for fully booked) will be implemented and how users will interpret this information.

2. **User Interface Design:** The user interface are designed to present the timetable and room availability information in a user-friendly and intuitive manner.

Data Security and Privacy
The measures we are take to ensure the security and privacy of the data stored in our application.

1. **Authentication and Authorization**: we are implementing user authentication and authorization mechanisms to ensure that only authorized users can access and modify timetable data.

2. **Data Encryption:** If applicable, mention sensitive data (such as user credentials) will be encrypted to prevent unauthorized access.

Conclusion
Summarize the feasibility of acquiring and using the necessary information for our project.

To provide references to any resources, libraries, or tools we are using to implement these features in our Flutter and Firebase-based application. By addressing the above points in our Information Feasibility section, we will demonstrate a thorough understanding of how our project will effectively handle and present data to users.

1.2.7. Motivational Feasibility

Motivational feasibility is an important aspect to consider, as it helps determine whether the project is worth pursuing from a motivational perspective. Here are some points to consider for the motivational feasibility of our project:

1. **Interest and Enthusiasm:** Our own interest and enthusiasm for the project. Are we passionate about classroom management, app development, and the technologies we are using (Flutter and Firebase)? our own motivation and excitement will play a crucial role in driving the project forward.

2. **Relevance**: The relevance of the project to our academic and career goals. This project aligns with our interests and the skills we want to develop. it be a valuable addition to our portfolio when we are applying for jobs in the future

3**. Learning Opportunities:** Reflect on the knowledge and skills we are gain from working on this project. It provides us with opportunities to learn new things, enhance our coding abilities, and deepen our understanding of app development and database management?

4. **Impact and Value:** About the potential impact of our project. A Classroom Management System benefits educational institutions. It solves real problems faced by teachers and students. Knowing that our work can make a positive impact can be highly motivating.

5. **Challenges and Growth:** The challenges we might face during the development process. Projects with technical challenges can be incredibly motivating as they push we to learn and grow. Are we excited about overcoming obstacles and expanding our problem-solving skills?

6. **Long-Term Vision:** The result of our project. It feels like seeing our app being used in a classroom environment. Visualizing the end goal can provide a motivational boost during challenging phases of development.

7. **Recognition and Acknowledgment:** The successful completion of this project could be acknowledged. Our university recognizes our work. This project led to other opportunities or collaborations.

8. **Collaboration:** we're working with a team, the camaraderie and teamwork can be highly motivating. Sharing the journey with peers who are equally invested can create a sense of community.

9. **Personal Satisfaction:** The sense of accomplishment and personal satisfaction that comes from completing a project can be a strong motivational factor. Imagine the feeling of pride when we showcase our fully functional Classroom Management System.

Remember, motivation can be an ever-changing aspect. There might be times when we encounter difficulties or doubts, but focusing on the reasons why we started the project and the positive outcomes it can bring can help reignite our motivation.

1.2.8. Legal & Ethical Feasibility

When it comes to the "**Legal & Ethical Feasibility**" aspect of our project, there are several considerations we should keep in mind, especially since we are dealing with an application that involves data management and potentially user information. Here are some points we should address in our feasibility report:

Legal Considerations:

1. **Data Privacy Laws:** Ensure that we are compliant with data privacy laws such as the General Data Protection Regulation (GDPR) if our application is used by individuals in the European Union. Even if not targeting EU users, it's a good practice to follow similar data protection principles.

2. **User Consent**: user data will be collected, stored, and used. Obtain explicit consent from users for collecting and processing their data.

3. **Terms of Service and Privacy Policy**: Create and display clear terms of service and a privacy policy that users must agree to before using our application. These documents should explain how their data will be used, who has access to it, and how they can exercise their rights.

4. **Intellectual Property:** If we're using any third-party libraries, frameworks, or components in our project, make sure we are following their respective licensing agreements and giving proper credit.

5. **Copyright and Ownership**: Clearly define the ownership of the application's code, content, and any other intellectual property. If we're collaborating with others, outline how ownership and credit will be attributed.

Ethical Considerations:

1. **Fair Treatment**: Ensure that the application treats all users fairly and does not discriminate based on factors like gender, race, ethnicity, etc. Avoid perpetuating bias in the allocation of resources.

2. **Transparency:** Be transparent about room availability and bookings are displayed. Users should understand how the system works and should not be misled by inaccurate information.

3. **Data Security**: Implement proper security measures to protect user data from unauthorized access, breaches, or leaks.

4. **Accessibility:** Design the application to be accessible to users with disabilities. This includes considerations for visual impairment, motor disabilities, etc.

5. **User Empowerment:** Provide users with control over their data. They should be able to edit or delete their information if they choose to do so.

6. **Accountability:** If there are any issues with the application, ensure that there is a clear mechanism for users to report problems and for our team to address them promptly.

Remember, the legal and ethical considerations are crucial to ensure the success and reputation of our application. It's advisable to consult with our academic advisor, legal experts, and ethical guidelines in our university or field to ensure we cover all necessary aspects in our feasibility report.

1.3. Project/Product Scope

Classroom Management System is an application-based software that aims to provide an effective and efficient way to manage classroom activities. This system is designed to help educators and administrators streamline their tasks such as managing student attendance, scheduling classes, organizing makeup lectures, and empty rooms. The system also allows students to access their class schedules, view their makeup lactures, and communicate with their teachers and peers.

The Classroom Management System will be developed using modern application technologies such as html, Java and PHP. The system will have a user-friendly interface that will be accessible from any device with an internet connection. The system will also be scalable and flexible, allowing it to be adapted to the needs of different educational institutions by making a few changes.

The primary goal of this project is to create a system that will simplify the management of classroom activities for teachers and administrators. The system will automate routine tasks such as attendance, and scheduling makeups, freeing up time for teachers to focus on more important tasks such as creating lesson plans and interacting with students. The system will also improve communication between teachers, administrators and parents, leading to better collaboration and student support. Overall, this system will streamline and streamline the educational process, benefiting both educators and students.

1.4. Task Dependency Table

Creating a Task Dependency Table is a crucial step in project planning and documentation. This table helps us outline the tasks involved in our project and establish their dependencies, ensuring a smooth flow of development. Here's how we can start creating our Task Dependency Table:

1. **Identify High-Level Tasks:**

Start by listing the major tasks involved in our project. Based on our description, some tasks might include:

- UI Design and Lawet

- Firebase Integration

- Creating Makeup Lectures Page

- Generating Timetable

- Room Occupancy Logic

- Color Coding Implementation

2. **Define Dependencies:**

Determine the order in which tasks need to be completed. Some tasks might depend on others being finished before they can start. For example:

- Firebase Integration depends on UI Design (to fetch and display data)

- Creating Makeup Lectures Page might depend on UI Design (lawet and navigation)

- Generating Timetable depends on Makeup Lectures Page (to gather data)

- Room Occupancy Logic might depend on Generating Timetable (to analyze data)

- Color Coding Implementation might depend on Room Occupancy Logic (to apply colors)

3. **Create a Table:**

Create a table with columns for:

- Task Name

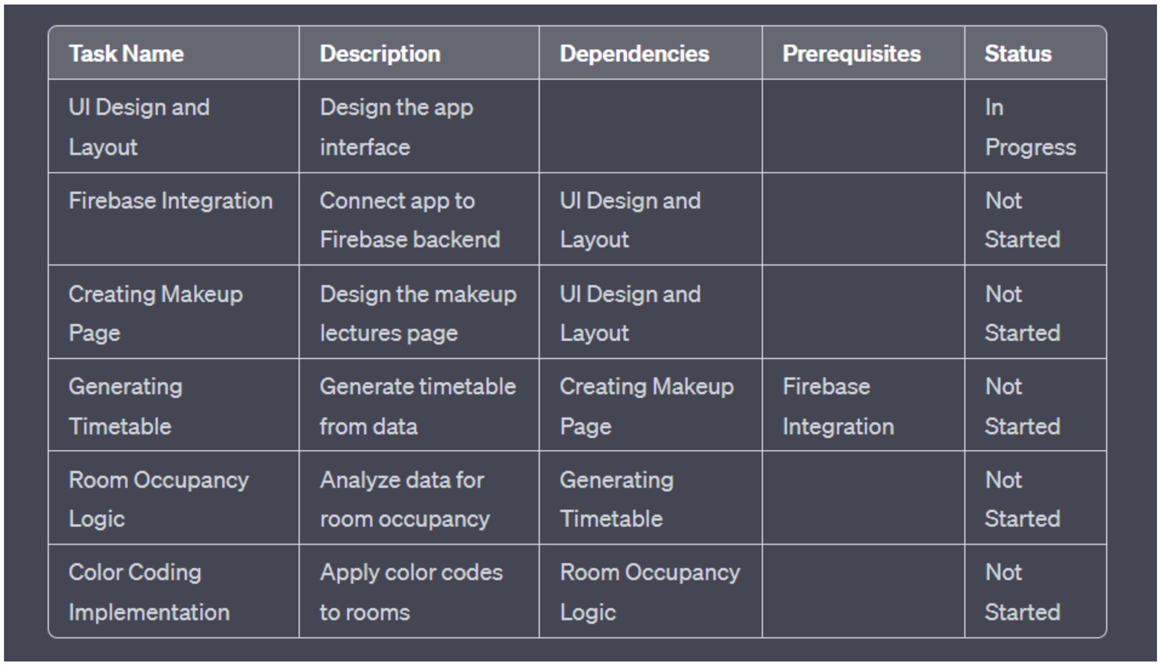
- Description

- Dependencies (Tasks that need to be completed before this task)

- Prerequisites (Tasks that require completion of this task)

- Status (Not Started, In Progress, Completed)

Here's the lawet:



Task Name	Description	Dependencies	Prerequisites	Status
UI Design and Layout	Design the app interface			In Progress
Firebase Integration	Connect app to Firebase backend	UI Design and Layout		Not Started
Creating Makeup Page	Design the makeup lectures page	UI Design and Layout		Not Started
Generating Timetable	Generate timetable from data	Creating Makeup Page	Firebase Integration	Not Started
Room Occupancy Logic	Analyze data for room occupancy	Generating Timetable		Not Started
Color Coding Implementation	Apply color codes to rooms	Room Occupancy Logic		Not Started

4. **Update and Track Progress:**

As we work on our project, update the status of each task in our table. This will help we keep track of our progress and identify any bottlenecks or delays.

5. **Iterate as Needed:**

Project plans can change as we delve deeper into development. Don't hesitate to update our Task Dependency Table as we gain more insights into the project's requirements and challenges.

Remember that this is just a basic structure to help we get started. We can customize it based on our project's specific needs and the tools we're using. In “ClassRoom Management System”

1.5. CPM - Critical Path Method

The critical path method (CPM) can be a useful technique to manage the project's timeline and ensure that we're able to complete it efficiently. Let's break down how we can apply the critical path method to our project documentation:

1. **Identify Tasks and Dependencies**:
Make a list of all the tasks required to complete our project. This could include tasks like designing the UI, setting up Firebase integration, creating the makeup lecture functionality, generating the timetable, etc. Determine which tasks are dependent on others. For example, we'll need to set up Firebase before we can start integrating it into our app.

2. **Estimate Durations:**
Estimate how long each task will take to complete. It's important to be realistic and account for potential challenges and setbacks. This will give we an idea of the time needed for each task.

3. **Create a Network Diagram:**
Create a visual representation of the tasks and their dependencies. This is typically done using a network diagram or a flowchart. This will help we understand the order in which tasks need to be completed and which tasks are dependent on others.

4. **Calculate Early Start (ES) and Early Finish (EF) Times:**
Determine the earliest possible start and finish times for each task. Start with tasks that have no dependencies (usually the initial tasks), and then move on to tasks that depend on those. The early finish time of a task is the early start time plus its estimated duration.

5. **Identify the Critical Path:**
The critical path consists of the sequence of tasks with the longest total duration. These are the tasks that must be completed on time to avoid delaying the entire project. Calculate the late start (LS) and late finish (LF) times for each task. The critical path tasks will have LS and LF times that are equal to their ES and EF times.

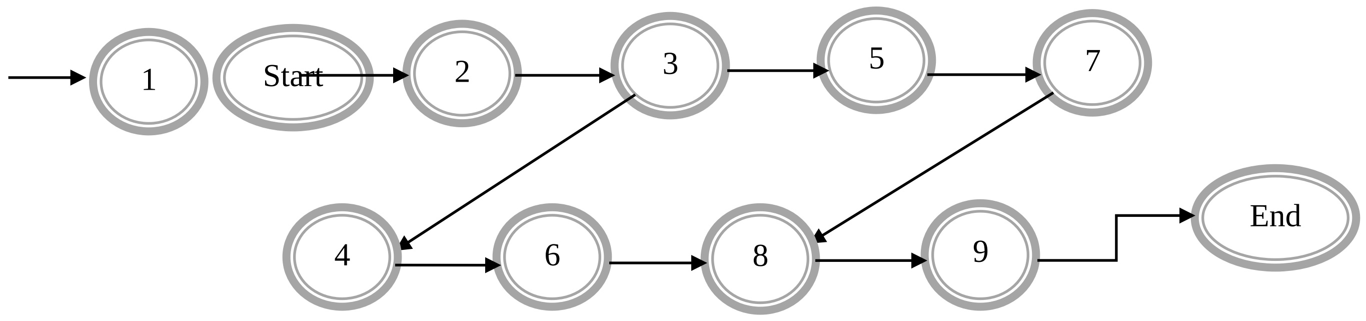
6. **Schedule Management:**
Monitor the progress of tasks regularly. If a task on the critical path is delayed, it could potentially delay the entire project. Focus on managing the critical tasks to ensure the project stays on track.

7. **Resource Allocation:**
As we're working with Flutter and Firebase, consider the resources (such as development time, coding expertise, testing, etc.) needed for each task. Ensure that we allocate resources appropriately to avoid bottlenecks.

8. **Document and Communicate:**
Document our critical path analysis in our project documentation. This will help we and our team understand the timeline and dependencies. If we're working with a team, communicate the critical path and its importance to everyone involved.

Remember, the critical path method is a dynamic tool that can help we adapt and manage our project as it progresses. It can also highlight potential areas of risk that might impact our project's completion date. Make sure to adjust our plan as needed and communicate any changes to our team or stakeholders.

ACTIVITY	**Duration**	**Immediate Predecessors**
1. System Analysis and Requirements	1 week	
2. Design and Development	2 weeks	1
3. Database Design and Development	3 weeks	2
4. Back-end Development	2 weeks	3
5. Front-end Development	1 week	3
6. Database Integration	2 weeks	4
7. User Interface Integration	2 weeks	5
8. System Testing and Debugging	3 weeks	6, 7
9. Implementation and Deployment	2 weeks	



Network Diagram for the above-mentioned activities

1.6. Gantt chart

Creating a Gantt chart can help we visualize the project timeline and tasks. Here's how we could create a Gantt chart for our project.

1. **Identify Project Tasks:**

Start by listing all the tasks involved in our project. These could include tasks like:

Requirement analysis and documentation

User interface design

Database design and Firebase integration

Creating the timetable feature

Creating the makeup lecture feature

Designing the room occupancy visualization

Testing and debugging

User testing and feedback

Finalizing documentation

2. **Determine Task Durations:**

Estimate the time it will take to complete each task. This could be in terms of days, weeks, or even months, depending on the complexity of the task and the resources available.

3. **Sequence Tasks:**

Tasks are dependent on others and need to be completed in a specific order. For example, we'll need to design the user interface before implementing the Firebase integration.

4. **Create Gantt chart:**

We can use various tools to create a Gantt chart. Tools like Microsoft Excel, Google Sheets, or dedicated project management software like Microsoft Project or Trello can be useful. Here's a simplified example of how our Gantt chart might look:

Task	Start Date	Duration
Requirement analysis	2023-06-01	1 week
UI design	2023-06-14	2 week
Database design and integration	2023-06-20	1 week
Timetable feature	2023-06-28	1 week
Makeup lecture feature	2023-07-10	1 week
Room occupancy visualization	2023-07-17	1 week
Testing and debugging	2023-06-30	2 week
User testing and feedback	2023-08-14	2 week
Documentation finalization	2023-08-30	3 week

1.7. Allocation of Members to Activities

it sounds like we're working on an interesting project related to classroom management using Flutter and Firebase. The part we need help with involves the "Allocation of Members to Activities." This seems to be about creating makeup lectures for missed classes and managing classroom availability. I'll provide we with a step-by-step approach to implement this feature in our application:

1. **Data Modeling:**
Before we start coding, it's important to design the data structure that will hold information about makeup lectures, classroom availability, and other relevant details. In Firebase, we might use Fire store collections and documents to represent this data. Create collections for makeup lectures and classroom availability.

2. **User Interface:**
Create the user interface for the "Create Makeup" page. This should include input fields for the makeup lecture date and subject selection. Also, provide a "Submit" button to initiate the process.

3. **Processing the Data:**
When the user submits the makeup lecture details, we need to process this data. On the backend, we will create a new document in the makeup lectures collection with the chosen date, subject, and other necessary information.

4. **Classroom Availability:**
To manage classroom availability, we need to track which rooms are occupied and which are available at a given time. We might have a collection in Firebase that stores the availability status of each room for each date. We could also use a color code like red for occupied and green for available.

5. **Displaying Classroom Availability:**
On the "Makeup Lecture" page, after submitting the details, we can query the database to get the classroom availability status for the selected date. Based on this data, we can display the rooms with appropriate colors (red for occupied, green for available).

6. **Handling Room Selection:**
Allow users to select an available room for the makeup lecture. This could be done by clicking on an available room, which then marks it as occupied and updates the database accordingly.

7. **Updating Availability Data:**
When a room is selected for a makeup lecture, update the availability data in the database to reflect that the selected room is now occupied for that date and time.

8. **Error Handling:**
Implement proper error handling to deal with scenarios such as selecting an already occupied room, handling connectivity issues with Firebase, etc.

9. **User Feedback:**
Provide appropriate feedback to users after they've successfully selected a room for the makeup lecture. This could be a confirmation message or a toast notification.

10. **Testing and Refinement:**
Thoroughly test our feature to make sure it works as expected. Make any necessary refinements based on user testing and feedback.

Remember to handle the authentication and security aspects of our application as well. Firebase provides authentication mechanisms that we can integrate into our Flutter app to ensure that only authorized users can access and modify the data.

Task	Duration (days)	Dependencies
T1	8	
T2	15	
T3	8	T1(M1)
T4	8	
T5	10	T2, T4(M2)
T6	5	T1, T2 (M3)
T7	8	T1 (M1)
T8	14	T4 (M5)
T9	14	T3, T6 (M4)
T10	18	T5, T7 (M7)

1.8. Tools and Technology with reasoning

The technology choices we've made are quite appropriate for building a mobile application with real-time functionality. Here's a breakdown of our chosen tools and technologies along with reasoning

1. **Flutter:**

Flutter is a popular open-source UI software development toolkit created by Google. It's known for its fast development, expressive UI components, and ability to create high-performance applications for various platforms from a single codebase. Choosing Flutter for our project is a good decision because:

- It allows we to create a consistent user interface across both Android and iOS platforms.
- The "hot reload" feature makes development faster and more iterative.
- Flutter's rich widget library helps in building interactive and visually appealing interfaces.
- Dart, the programming language for Flutter, is easy to learn and work with.

2. **Firebase:**

Firebase is Google's mobile and web application development platform that provides a wide range of services, including real-time databases, authentication, hosting, storage, and more. Using Firebase in our Classroom Management System project has several advantages:

1. **Real-time Database:** Firebase's real-time database allows we to update and synchronize data in real time, which is crucial for a classroom management system where schedules and room availability change frequently.

2. **Authentication:** We can use Firebase Authentication to manage user logins and permissions, ensuring secure access to the application.

3. **Cloud Firestore:** Firestore is another database option in Firebase that offers powerful querying capabilities and seamless integration with mobile applications.

4. **Hosting:** Firebase Hosting provides a simple way to deploy and host our application without the need for complex server setups.

3. **Reasoning for Flutter and Firebase:**

1. **Cross-Platform Development:** Flutter allows us to create a single codebase for both Android and iOS platforms, reducing development time and maintenance efforts.

2. **Real-Time Functionality:** Firebase's real-time capabilities are well-suited for applications that require instant updates, like our Classroom Management System. Changes to the timetable, room availability, and makeup classes can be reflected in real time.

3. **Scalability:** Firebase can handle many users and real-time updates without significant backend infrastructure management.

4. **User-Friendly Interface**: Flutter's widgets and design options can help create an intuitive and visually appealing user interface, essential for user adoption.

5. **Ease of Integration**: Flutter and Firebase have good integration support, allowing us to focus on building features rather than dealing with complex integrations.

Remember to thoroughly document our technological choices and reasoning in our project documentation. This will showcase our understanding of the technologies and our ability to make informed decisions based on the project's requirements.

1.9. Vision Document

1. **Introduction**

The Classroom Management System is an application designed to facilitate efficient management of classroom schedules, particularly focused on handling makeup lectures. The system aims to streamline the process of scheduling makeup lectures, managing classroom availability, and providing an intuitive interface for students and instructors to access and interact with the timetable. The application is developed using the Flutter framework for the user interface and Firebase for backend services.

2. **Purpose and Scope**

The primary purpose of the Classroom Management System is to provide a user-friendly platform that assists in managing makeup lectures effectively. It addresses the following key objectives:

i. **Makeup Lecture Scheduling**: The system enables students and instructors to schedule makeup lectures for missed classes. Users can specify the date and subject for the makeup lecture.

ii. **Timetable Visualization:** The application offers a visual representation of the classroom timetable for the selected date. It highlights available and occupied rooms using distinct colors, allowing users to easily identify room availability.

iii. **Efficient Room Management:** By indicating room availability through color-coding, the system optimizes room allocation, minimizing conflicts and maximizing room utilization.

iv. **User-Friendly Interface:** The user interface is designed to be intuitive and easy to navigate, ensuring a seamless experience for both students and instructors.

3. **Features**

The Classroom Management System includes the following key features:

1. **Login and Authentication:** Users (students, instructors and Admin) can log in securely using their credentials. Firebase authentication ensures data security.
2. **Makeup Lecture Creation:** Users can schedule makeup lectures by selecting the date and subject. This information is stored in the Firebase database.
3. **Timetable Display**: The system generates a timetable for the selected date, highlighting available and occupied rooms using color-coding.
4. **Room Availability Indicator**: Empty rooms are indicated with a red color, and fully booked rooms are indicated with a green color, making it easy for users to identify room availability at a glance.
5. **User Profiles:** Students and instructors can manage their profiles, view their schedules, and track makeup lectures.

4. **Technology Stack**

The Classroom Management System is built using the following technologies:

i. **Frontend:** Flutter framework is used for developing the user interface, providing a consistent experience across multiple platforms.

ii. **Backend:** Firebase serves as the backend platform, providing real-time database capabilities, authentication services, and hosting for the application.

5**. Future Enhancements**

In the future, the Classroom Management System could be expanded with the following enhancements:

i. **Automated Scheduling**: Implement an algorithm to suggest optimal makeup lecture schedules based on room availability and instructor/student preferences.

ii. **Notifications**: Integrate push notifications to remind users of upcoming makeup lectures and changes in their schedules.

iii. **Analytics and Reporting:** Provide insights into room utilization, makeup lecture frequency, and attendance trends through visual analytics.

iv. **Integration with Learning Management Systems**: Connect the system with existing learning management platforms to synchronize course information and assignments.

6. **Conclusion**

The Classroom Management System aims to simplify the process of scheduling makeup lectures and managing classroom availability. By providing a user-friendly interface and leveraging modern technologies, the application contributes to efficient classroom resource utilization and improved student-instructor and admins interaction. Through continuous development and future enhancements, the system has the potential to become an integral tool in educational institutions, enhancing the overall learning experience.

1.10. Product Features/ Product Decomposition

A Classroom Management System can be quite useful for educational institutions. Let's break down the product features and product decomposition for our project.

Product Features:

1. **User Authentication:**
   - Allow users (teachers and administrators) to register and log in to the system using their credentials.

2. **Dashboard:**
   - Provide a user-friendly dashboard where users can see an overview of their classes, schedules, and notifications.

3. **Timetable Management:**
   - Allow administrators to create and manage the main class timetable.
   - Teachers can view their assigned classes and schedules.

4. **Makeup Class Creation:**
   - Teachers can request makeup classes by selecting a date and subject.
   - The system can automatically find available rooms based on the selected date and subject.

5. **Room Availability Display:**
   - Display the availability of classrooms on a color-coded interface (red for occupied, green for available).
   - Teachers can quickly see which rooms are available for makeup classes.

6. **Classroom Booking:**
   - Teachers can book available rooms for makeup classes directly from the interface.

7. **Notifications:**
   - Send notifications to teachers and administrators regarding class changes, makeup class approvals, etc.

8. **Admin Controls:**
   - Admins can approve or reject makeup class requests.
   - Admins can manage user accounts, subjects, and other system settings.

9. **Data Management:**
   - Use Firebase to store and manage user data, class schedules, room availability, etc.

Product Decomposition:

1. **User Authentication:**
   - Registration page with user details input.
   - Login page for authentication.

2. **Dashboard:**
   - Overview of upcoming classes and notifications.

3. **Timetable Management:**
   - Admin interface to create and edit the main class timetable.
   - Teacher interface to view assigned classes and schedules.

4. **Makeup Class Creation:**
   - Makeup class request form with date and subject selection.

5. **Room Availability Display:**
   - Interface to display available classrooms for the selected date and subject.

- Color-coded system to differentiate between occupied and available rooms.

6. **Classroom Booking:**
   - Interface to book available rooms for makeup classes.

7. **Notifications:**
   - System to send automatic notifications to users.

8. **Admin Controls:**
   - Admin dashboard to manage makeup class requests and approvals.
   - User and subject management interfaces.

9. **Data Management (Firebase):**
   - Set up Firebase database to store user data, class schedules, room availability, etc.

Remember to break down each feature into smaller tasks, define the user interfaces clearly, and plan out the interactions between different parts of our system. Good documentation and clear organization are key to the successful development of our project. Good luck!

Chapter 2: Software Requirement Specification (For Object Oriented Approach)

2.1 Introduction:

The "Classroom Management System" is a comprehensive software solution designed to streamline and enhance the management of classroom schedules, makeup lectures, and room occupancy within an educational institution. This project aims to address the challenges faced in effectively managing classroom assignments, makeup lectures, and room availability, thereby improving the overall educational experience for both students and faculty members.

Background:

In traditional educational settings, managing classroom schedules, makeup lectures, and room occupancy can often be a time-consuming and error-prone task. Academic institutions frequently encounter scenarios where regular classes are missed due to unforeseen circumstances, such as faculty unavailability or student events. Consequently, there arises a need for a systematic approach to handle such situations efficiently.

Project Scope:

The "Classroom Management System" is a mobile application developed using the Flutter framework and integrated with Firebase for backend support. This project focuses on providing a user-friendly interface to manage makeup lectures and visualize room occupancy status. Users, including faculty members and administrators, can access the application to create makeup lectures, select dates, specify subjects, and view room availability on designated dates.

Objectives:

The primary objectives of the "Classroom Management System" are as follows:

1. **Efficient Makeup Lecture Creation**: The application offers a simplified process for faculty members to schedule makeup lectures for missed classes. Users can choose the date and subject for which the makeup lecture needs to be conducted.

2. **Real-time Room Availability**: The system provides real-time information about room occupancy status. Empty rooms are highlighted with red indicators, while fully booked rooms are indicated in green. This visualization aids in selecting appropriate rooms for makeup lectures.

3. **User-Friendly Interface:** The application is designed with an intuitive user interface that allows users to easily navigate through the various features, select options, and submit requests.

4. **Data Integrity:** By leveraging Firebase as the backend database, the system ensures data integrity and reliability. This reduces the risk of data loss and maintains the accuracy of scheduling information.

5. **Enhanced Decision-Making:** Faculty members and administrators can make informed decisions regarding makeup lectures based on real-time room availability, optimizing the allocation of resources.

6**. Streamlined Communication:** The application facilitates better communication between faculty members and administrators by providing a centralized platform to manage makeup lectures and room assignments.

Target Audience:

The "Classroom Management System" primarily caters to the needs of educational institutions, including colleges and universities. The system benefits faculty members responsible for conducting classes and makeup lectures, as well as administrators in charge of managing classroom schedules and resources.

Conclusion:

The "Classroom Management System" seeks to revolutionize the way educational institutions handle makeup lectures and room occupancy management. By leveraging the capabilities of Flutter and Firebase, this project aims to provide a seamless and efficient solution to address the challenges associated with classroom scheduling and utilization. Through its user-friendly interface and real-time room availability visualization, the system aims to enhance the overall educational experience and streamline administrative processes.

2.2 Systems Specifications

A Classroom Management System that involves a timetable and makeup lectures using Flutter and Firebase sounds quite practical and useful. I help us with the Systems Specifications section of our Software Requirement Specification document. This section is crucial as it outlines the overall functionality and behavior of our system.

Here's a structure we can follow for the Systems Specifications section:

1. **Introduction:**

A brief overview of the Classroom Management System, its purpose, and the problem it aims to solve. Mention the importance of efficient classroom scheduling and management for educational institutions.

2. **System Architecture:**

High-level architecture of our system, outlining the components and their interactions. We can mention the use of Flutter for the frontend user interface and Firebase for the backend database and authentication.

3. **Functional Requirements:**

The functional requirements that our system must fulfill. These requirements should be specific, measurable, and directly tied to the system's core functionality. Here are some examples:

I. Timetable Management:

a. Allow users to view the existing timetable.
b. Provide the ability to create makeup lectures.
c. Users should be able to specify the date and subject for makeup lectures.
d. The system must update the timetable with makeup lecture information.

II. Room Availability Display:

a. Display the timetable for a selected date, highlighting occupied and vacant rooms.
b. Use color-coding to distinguish between occupied and vacant rooms (e.g., red for vacant, green for occupied).

III. **Authentication and User Access:**

a. Implement user authentication using Firebase.
b. Differentiate between student, teacher and Admin accounts.
c. Only authorized teacher users should be able to request for creating makeup lectures and view room availability.
d. These request are accepted by the Admin only.

4. **Non-Functional Requirements:**

Outline the non-functional requirements that define the quality attributes of our system. These may include:

I. **Usability:**

a. The user interface should be intuitive and easy to navigate.
b. Response times for actions (e.g., viewing timetable, submitting makeup lectures) should be within acceptable limits.

II. **Scalability:**

   a. The system should be able to handle a growing number of users and timetable entries.

III. **Security:**

   a. User authentication and data transmission should be secure.
   b. Only authorized users should have access to sensitive actions.

5. **User Interface:**

A description of the user interface elements and Users will interact with the system. We can include wireframes, mockups, or diagrams to illustrate the interface.

6. **Use Cases:**

Various scenarios of how users will interact with the system to achieve specific tasks. For instance:

- **Use Case 1**: Viewing Timetable
- **Use Case 2**: Creating Makeup Lecture
- **Use Case 3**: Checking Room Availability

7. **Constraints and Assumptions:**

Systems Specifications:

1. **User Roles:**

- **Admin:** Responsible for approving makeup lectures and managing user accounts.
- **Professors/Teachers:** Allowed to create and manage makeup lectures.
- **Students:** Allowed to view the timetable and receive notifications about upcoming makeup lectures.

2. **Functional Requirements**

a) **User Authentication:** Users (admin, professors, students) must log in with valid credentials to access the system.

b) **Admin Functions:**
   i) Approve or reject makeup lecture requests.
   ii) Manage user accounts (add, edit, delete).

c) **Professor/Teacher Functions:**

i. Create makeup lecture requests (date, time, classroom).
ii. Edit or cancel makeup lecture requests.

d) **Student Functions:**
   i. View the classroom timetable.
   ii. Receive notifications about approved makeup lectures.

2. **Non-Functional Requirements:**
   i. **Security:** User data must be stored securely, and only authorized users should have access to sensitive information.

   ii. **Usability:** The system should have an intuitive user interface for easy navigation.

   iii. **Performance:** The application should load quickly and handle a reasonable number of concurrent users.

   iv. **Scalability:** The system should be able to accommodate a growing number of users and data.

   v. **Reliability:** The system should be available for use during normal working hours without frequent downtime.

4. **Constraints and Assumptions:**
   I. **Constraints:**
      a. **Limited Budget**: The project has a limited budget, which may affect the choice of technologies and resources available.
      b. **Technology Stack**: The project may be constrained to using specific technologies mandated by the university or department.
      c. **Timeframe**: The project must be completed within the specified academic semester, which may limit the scope and features.

   I. **Assumptions:**
      a. **Availability of Internet**: It is assumed that users will have access to the internet to log in and use the application.
      b. **User Compliance**: Users are expected to follow the guidelines and use the system as intended.
      c. **Hardware**: Users will have access to compatible devices (computers or smartphones) to access the application.

These constraints and assumptions should be considered while designing and developing our classroom management system. They provide a framework for understanding the limitations and expectations of our project.

8**. Future Enhancements:**

Any potential future enhancements or features that could be added to the system, such as integration with other educational tools, advanced analytics, or mobile app versions.

2.2.1. Identifying External Entities:

Flutter and Firebase, and we're currently focusing on the Software Requirement Specification (SRS) document, specifically the section about identifying external entities. External entities refer to the components, systems, or actors that interact with our application. In this case, we've mentioned a few key entities like makeup lectures, rooms, and the timetable.

Here's how we can approach identifying and describing these external entities in our SRS:

1. **Makeup Lectures:**
   a. What makeup lectures are in the context of our application. These are likely sessions that allow students to catch up on missed lectures.
   b. Users (students or administrators) initiate the creation of makeup lectures.
   c. The purpose of makeup lectures within our application's workflow.
   d. The attributes associated with makeup lectures, such as date, subject, etc.

2. **Rooms:**
   a. The concept of rooms within our application. These could be physical classrooms where lectures take place.
   b. Rooms are utilized in the makeup lecture creation process.
   c. The application tracks the availability and occupancy status of rooms.
   d. Any attributes associated with rooms, such as room numbers, capacity, etc.

3**. Timetable:**
   a. The role of the timetable within our application. This could be a schedule that displays lecture timings, subjects, and rooms.
   b. The timetable is accessed by users and where it fits in the application's flow.
   c. Makeup lectures and regular lectures are reflected in the timetable.
   d. The attributes of the timetable, such as date, time slots, subjects, and room availability indicators.

4. **Users (Actors):**

a. The different types of users who will interact with our application, such as students and administrators.
b. The roles and responsibilities of each user type.
c. Each user type interacts with the makeup lecture creation, room occupancy, and timetable features.

5. **Color Indicators:**

a. The use of color indicators (red and green) to represent room occupancy status.
b. These indicators help users quickly understand which rooms are available and which are fully booked.

Additionally, ensure that our SRS covers functional requirements (what the system should do), non-functional requirements (performance, security, etc.), constraints, and assumptions. This will create a comprehensive guide for our project's development.

2.2.2. Context Level Data Flow Diagram:

Creating a Context Level Data Flow Diagram (DFD) is a great way to visualize the flow of data and interactions in our system. Here's a step-by-step guide on how to create the Context Level DFD for our Classroom Management System:

1. **Identify External Entities:**

Start by identifying the external entities that interact with our system. In our case, the external entities could include students, teachers, and administrators. These entities interact with our system to perform various actions.

2. **Identify Processes:**

Next, identify the main processes that our system will perform. Based on our project description, processes could include actions like "Create Makeup Lecture," "Generate Timetable," and "Display Room Availability."

3. **Draw the Context Level DFD**:

Create a DFD diagram using shapes to represent external entities, processes, and data flow. Here's a basic breakdown:

- External Entities:

Represented as rectangles. Label them as "Student," "Teacher," and "Administrator."

a. **Processes:** Represented as circles or ovals. Label them as "Create Makeup Lecture," "Generate Timetable," and "Display Room Availability."
b. **Data Flow:** Represented as arrows connecting external entities and processes. These arrows indicate the flow of data between entities and processes.

c. **Data Stores**: If needed, represent data stores as open-ended rectangles. In our case, Firebase can be considered a data store.

4. **Connect External Entities and Processes:**
Draw arrows (data flows) to connect external entities to the processes they interact with. For example, we might have an arrow from "Student" to "Create Makeup Lecture" to show that students initiate the process of creating makeup lectures.

5. **Connect Processes and Data Store:**
If our system interacts with a data store (Firebase in our case), draw arrows from processes to the data store to indicate data retrieval or storage.

6. **Describe Data Flows:**
Add labels to the arrows to describe what kind of data is being exchanged. For instance, data like "Makeup Lecture Details," "Timetable Request," or "Room Availability Information" should be labeled on the respective arrows.

7. **Add Process Descriptions:**
Beside each process, add a brief description of what the process does. For example, beside the "Generate Timetable" process, we could write "Processes requests and compiles room availability and lecture information to generate a timetable."

8. **Review and Refine:**
Review our diagram for accuracy and completeness. Make sure that all interactions between entities and processes are clearly represented.

2.2.3. Capture "shall" Statements:
Currently working on the Software Requirement Specification (SRS) and need assistance with capturing "shall" statements for the Systems Specifications section. I'll help we outline some "shall" statements based on our description:

1. **Creating Makeup Lectures:**
   - The system shall provide a button to access the makeup lecture creation page.
   - The system shall allow users to select a specific date for creating makeup lectures.
   - The system shall provide an option to choose the subject for the makeup lecture.
   - The system shall validate the selected date and subject before proceeding.

2. **Displaying Timetable:**
   - The system shall display the timetable for the selected makeup lecture date.
   - The timetable shall visually differentiate between empty and occupied rooms.
   - Empty rooms shall be highlighted using a red color code.
   - Occupied rooms shall be highlighted using a green color code.
   - The system shall ensure accurate room status representation based on real-time data.

3. **Room Availability:**
- The system shall retrieve and display a list of available rooms for the selected date.
- The system shall mark rooms as available or occupied based on the timetable.
- The system shall update room availability status in real-time as new makeup lectures are scheduled.

4. **User Interaction:**
- The system shall provide interactive buttons for users to submit the makeup lecture details.
- The system shall allow users to view room details by clicking on specific rooms in the timetable.
- The system shall display relevant information when a room is clicked, including its occupancy status and makeup lecture details.

5. **Error Handling:**
- The system shall handle errors gracefully if the selected date is invalid or unavailable.
- The system shall display appropriate error messages when required fields are not filled.
- The system shall prevent scheduling makeup lectures in already occupied rooms.

6. **Database Integration:**
- The system shall store makeup lecture information in the Firebase database.
- The system shall retrieve timetable and room availability data from Firebase.
- The system shall ensure secure and authenticated access to the Firebase database.

7. **Visual Feedback:**
- The system shall provide visual feedback to the user upon successful submission of makeup lecture details.
- The system shall display loading indicators during data retrieval from the database.

2.2.4 Allocate Requirements:

"Software Requirement Specification" section of our project documentation, specifically focusing on "Allocate Requirements." This section typically outlines the specific functionalities and features of our application.

To do with the "Allocate Requirements" section, I'll break down the information we've provided and provide we with a structured outline that we can expand upon in our documentation.

Makeup Lecture Creation

1. **Create Makeup Lecture Page**
   - Develop a dedicated page within the application for creating makeup lectures.
   - This page should be accessible via a button or specific navigation path.

2. **Input Date for Makeup Lecture**
   - Provide a user interface element (such as a date picker) for users to select the date on which they want to create a makeup lecture.

3. **Select Subject for Makeup Lecture**
   - Implement a selection mechanism (dropdown, autocomplete, etc.) for users to choose the subject for which they are scheduling the makeup lecture.

4. **Submit Button**
   - Include a "Submit" button to initiate the process of creating the makeup lecture based on the chosen date and subject.

Makeup Lecture Timetable Display

1. **Display Timetable**
   - After submitting the makeup lecture details, present users with a timetable view of the chosen date.
   - This timetable should include information about the availability of classrooms and their occupancy status.

2. **Color-Coded Representation**
   - Use a color-coded scheme (e.g., red and green) to visually distinguish between empty and occupied classrooms.
   - Red indicates that a classroom is available and can be booked for the makeup lecture.
   - Green indicates that a classroom is fully booked and not available for scheduling.

3. **Interactive Interface**
   - Make the timetable interactive, allowing users to click on available classrooms to proceed with the booking process.

4. **Booking Confirmation**
   - Provide a confirmation mechanism for users to finalize the booking of a classroom for the makeup lecture.
   - Display relevant details (date, subject, and classroom) before confirming.

2.2.5. Prioritize Requirements (If Any):

The software requirement specification (SRS) is an essential document that outlines what our project will entail and how it will function. Prioritizing requirements helps we determine which features are critical for the initial version of our project. Based on our description, I'll help we identify and prioritize requirements for our Classroom Management System:

Priority 1 - Must-Have Features:

1. **User Authentication:**
   - Users should be able to sign up, log in, and log out securely.
   - Different user roles (students, teachers, administrators) should have appropriate access levels.

2. **Timetable Display:**
   - Display the overall timetable with classes and events for each day.
   - Highlight occupied and available rooms using color codes.
   - Rooms' availability status should be updated in real-time.

3. **Makeup Class Creation:**
   - Teachers can create makeup classes.
   - Select the date and subject for the makeup class.
   - Submitting a request updates the timetable with the makeup class.

Priority 2 - Important Features:

1. **Real-Time Updates:**
   - Any changes to the timetable (e.g., makeup class creation) should reflect in real-time for all users.

2. **User Profiles:**
   - Users can view and edit their profiles.
   - Teachers' profiles can include subjects they teach.

3. **Room Booking Management:**
   - Admins can manage room bookings and availability manually.
   - Mark rooms as occupied or available as needed.

Priority 3 - Nice-to-Have Features:

1. **Notifications:**
   - Users receive notifications about makeup class updates, room changes, etc.

2. **Attendance Tracking:**
   - Teachers can mark attendance for makeup classes.
   - Students can view their attendance records.

3. **Reporting:**
   - Generate reports on makeup classes, attendance, and room occupancy.

2.3. Existing Systems / Literature Review:

Developing a Classroom Management System using Flutter and Firebase is a great choice. Let's proceed with the literature review or existing systems section of our project. This part of the documentation is crucial to understand what similar systems or applications are already out there, how they function, and what features they offer. This will help we identify gaps in the existing solutions that our project could address.

Here's a step-by-step guide to help we with our literature review:

1. **Identify Relevant Keywords:** Begin by identifying keywords related to our project. These could include terms like "classroom management system," "timetable management," "Flutter apps," "Firebase integration," etc.

2. **Search Academic Databases:** Look for academic databases like IEEE Xplore, ACM Digital Library, Google Scholar, and others. Use our identified keywords to search for research papers, articles, and conference papers related to classroom management systems or similar topics.

3. **Analyze Existing Systems:** Once we find relevant papers, analyze the existing systems mentioned in those papers. Pay attention to the features they offer, the technologies they use, their strengths, and their limitations.

4. **Note Down Key Points:** While reviewing each system, take notes on key points such as the main functionalities, user interfaces, technologies used, user experiences, and any unique features that stand out.

5. **Compare and Contrast**: Compare the different existing systems we've found. What are the common features among them? What are the differences? How do they handle classroom scheduling, makeup classes, room occupancy, and user interactions?

6. **Identify Gaps:** Based on our analysis, identify any gaps or limitations in the existing systems. These gaps could be areas where our project could excel or provide unique solutions.

7. **Document the Review:** In our documentation, present a summary of each existing system we've analyzed. Provide a comparison table or a chart that highlights the similarities and differences among these systems. Mention the pros and cons of each system.

8. **Highlight Our Approach:** After reviewing existing systems, we can highlight how our proposed Classroom Management System using Flutter and Firebase will differ from or improve upon these existing solutions. What unique features will our system offer? How will it address the identified gaps?

9. **Cite Our Sources:** Make sure to properly cite the sources of the existing systems we've reviewed. This is important for academic integrity and to give credit to the authors of the original work.

2.3.1. Existing System:

. A Classroom Management System with a focus on timetable management using Flutter and Firebase is a great choice! As we're working on the "Software Requirement Specification" part, we'll need to review existing systems or literature related to classroom management systems. Here's how we can approach the Existing Systems / Literature Review section:

Existing Systems / Literature Review:

1. **Identify Existing Systems:** Research and identify any existing classroom management systems or scheduling tools that are currently being used in educational institutions. Look for both commercial products and open-source solutions.

2. **Review Features**: Analyze the features and functionalities offered by these existing systems. Pay attention to how they handle timetable creation, makeup lectures, room availability, and user interactions.

3. **Compare Technologies:** Since we're using Flutter and Firebase, focus on systems that are built using similar technologies. Look for case studies, blog posts, or research papers that discuss the use of Flutter and Firebase in educational or scheduling applications.

4. **User Reviews and Feedback:** If possible, gather information about user reviews and feedback for these existing systems. This can give we insights into the strengths and weaknesses of different solutions from the perspective of end-users.

5. **Limitations and Gaps**: Identify any limitations or gaps in the existing systems that our project could address. Perhaps there are features missing in current solutions that we can implement in our system.

6. **Innovative Ideas**: Look for innovative ideas that we can incorporate into our project. This could include unique ways of visualizing room availability, optimizing timetable creation, or improving user experience.

7. **Academic Papers:** Search for any research papers or academic articles related to classroom management systems, timetable scheduling, or educational technology. These papers can provide theoretical background and insights into industry trends.

8. **Open-Source Projects**: Explore open-source projects related to scheduling or classroom management on platforms like GitHub. These projects can provide we with valuable code examples and implementation ideas.

9. **Industry Reports**: If there are any industry reports or surveys related to education technology and classroom management, these can provide we with a broader understanding of the challenges and needs in this domain.

10. **Citations and References:** Make sure to properly cite and reference the sources we use in our literature review. This adds credibility to our project and acknowledges the work of others in the field.

2.4. Usecase Diagram of Our Project:

A Use Case Diagram is a great way to visually represent the interactions between users and the system. Here's a basic outline of how our Use Case Diagram might look for our project:

Use Case Diagram: Classroom Management System

1. **Actors:**
   - Student
   - Teacher
   - Admin

2. **Use Cases:**

   Student:
   - View Timetable
   - Get Notification
   - View Makeup Lecture Schedule

Teacher:
- View Timetable
- Schedule Makeup Lecture
- View Makeup Lecture Schedule
- Check Room Availability

Admin:
- Manage Timetable
- Manage Room Availability
- Accept the request for makeup lecture.
- Manage Users

3. **Use Case Descriptions:**

View Timetable:
- Actors: Student, Teacher
- Description: Users can view the regular class timetable.

Request Makeup Lecture:
- Actors: Student
- Description: Students can request a makeup lecture for a specific date and subject.

Schedule Makeup Lecture:
- Actors: Teacher
- Description: Teachers can schedule makeup lectures based on student requests.

View Makeup Lecture Schedule:
- Actors: Student, Teacher
- Description: Users can view the schedule of makeup lectures.

Check Room Availability:
- Actors: Student, Teacher
- Description: Users can check the availability of rooms for makeup lectures.

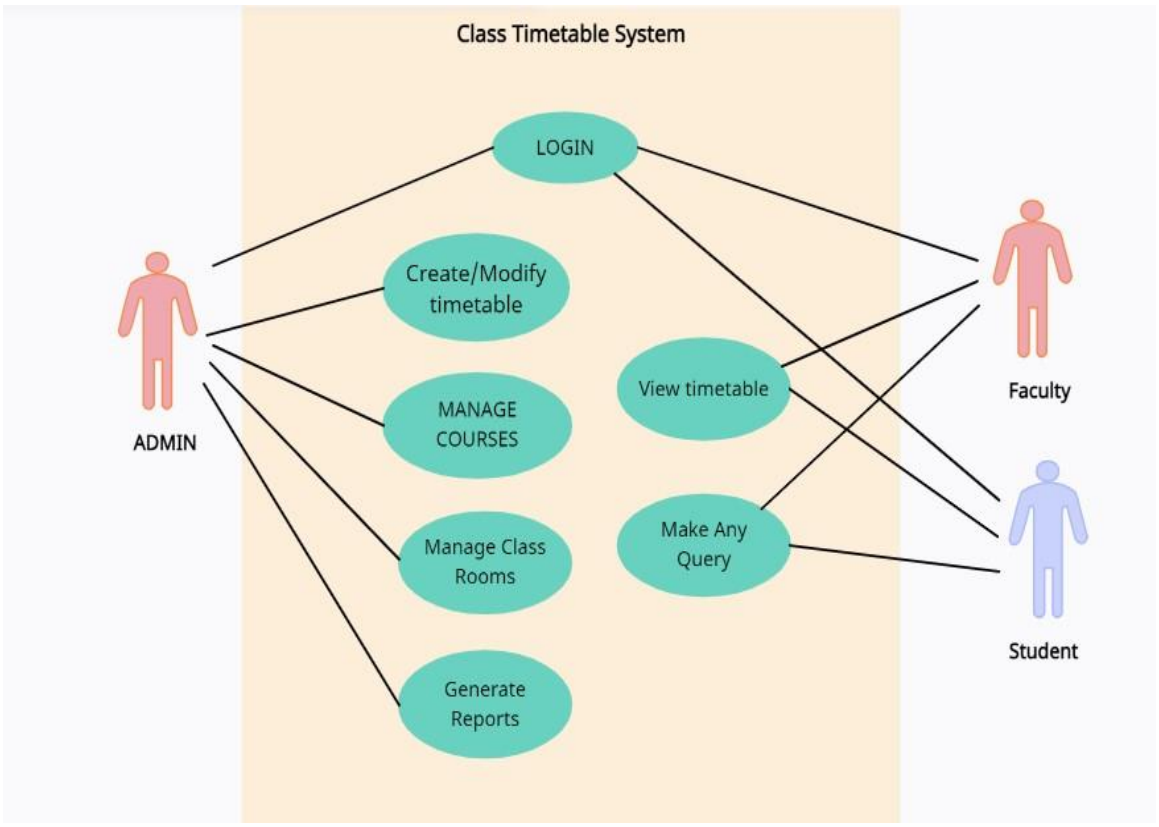
Manage Timetable:
- Actors: Admin
- Description: Admin can manage the regular class timetable, including adding, editing, and deleting classes.

Manage Room Availability:
- Actors: Admin
- Description: Admin can manage room availability, mark rooms as occupied or empty.

Manage Users:
- Actors: Admin
- Description: Admin can manage user accounts, including adding and deleting users.

4. **Relationships:**
- Association between actors and use cases to show who interacts with which functionality.

Remember that this is a simplified outline, and our actual Use Case Diagram might have more details and specific interactions. Also, since we're building the project with Flutter and Firebase, consider including interactions related to authentication, database operations, and real-time updates.



Use case Diagram of Our Project:

2.4.1. Use case Description.

Use case description is a crucial step in project documentation. I'll help we outline the use case description for our "Classroom Management System" project.

Use Case Description: Classroom Management System

Use Case 1: Create Makeup Lecture

- **Actor:** Faculty Member

- **Description**: This use case allows a faculty member to schedule a makeup lecture for a missed class.

- **Flow of Events:**
  1. The faculty member logs into the application using their credentials.
  2. They navigate to the main dashboard.
  3. On the dashboard, they click on the "Create Makeup Lecture" button.
  4. The application presents the "Create Makeup Lecture" page.
  5. The faculty member selects the date for the makeup lecture using a date picker.
  6. They choose the subject for which the makeup lecture needs to be scheduled from a dropdown list.
  7. Upon clicking the "Submit" button, the application validates the inputs.
  8. If the inputs are valid, the application schedules the makeup lecture and updates the timetable.
  9. The timetable is then refreshed, and available rooms are displayed.
  10. Empty rooms are highlighted in red, and fully booked rooms are highlighted in green.

- **Alternative Flow:**
  - If the inputs are not valid (e.g., date is not available, subject is not selected), appropriate error messages are displayed, and the faculty member is prompted to correct the inputs.

- **Post-conditions:** The makeup lecture is added to the timetable, and room availability status is updated.

Use Case 2: View Timetable

- **Actor:** Faculty Member

- **Description:** This use case allows a faculty member to view the timetable and room availability for a specific date.

- **Flow of Events:**

1. The faculty member logs into the application using their credentials.
2. They navigate to the main dashboard.
3. On the dashboard, they select the "View Timetable" option.
4. The application presents a calendar view where the faculty member can choose a date to view the timetable.
5. After selecting a date, the application fetches and displays the timetable for that date.
6. Empty rooms are highlighted in red, and fully booked rooms are highlighted in green.

- **Post-conditions:** The faculty member gains insight into room availability for the selected date.

Chapter 3: Design Document (For Object Oriented Approach)

3.1. Introduction:

Classroom scheduling and management is a complex process for educational institutions. Creating timetables, allocating classrooms, managing makeup lectures etc. is typically handled manually by administrative staff. This project aims to automate and streamline some of these classroom management processes via a mobile-based application.

The Classroom Management System provides key features to manage lecture timetables and booking of makeup lectures. The system allows professors to request and book makeup lectures as per their requirements. The application checks for classroom availability on the requested date/time and books the makeup lecture by updating the timetable. It also notifies students automatically about any changes in lecture schedule.

The main objectives of this project are:

- **Automate** lecture timetable generation/edit and management
- **Allow** professors to easily request and book makeup lectures
- **Check** classroom availability before booking makeup lectures
- **Update** timetable with booked makeup lectures
- **Notify** students automatically about makeup lecture schedule

The system focuses mainly on makeup lecture booking use case for professors. Key entities include Professor, Student, Course, Classroom, Timetable and Makeup Lecture. Core relationships modelled are - A professor teaches Courses, A Course has enrolled Students, Timetable contains scheduled Lectures, Lectures occur in Classrooms, and Professor can request Makeup Lecture for a Course to be booked in an available Timeslot in a Classroom.

This project aims to develop a useful classroom management system by leveraging these capabilities for automating lecture timetables and makeup bookings. The system would make scheduling of classes more efficient for educational institutes.

Now we discuss these artifacts one by one as follows:

3.2. Domain Model

Here is a domain model for the Classroom Management System focused on makeup lecture booking:

Domain Model

Actors:
- Professor
- Student
- Admin

Key Entities:
- Professor
- Student
- Course
- Classroom
- Timetable
- Makeup Lecture

Relationships:
- Professor teaches Courses
- Courses have Students enrolled
- Timetable contains scheduled Lectures
- Lectures are held in Classrooms
- Professor can request Makeup Lecture
- Makeup Lecture is for a specific Course
- Makeup Lecture booked into available Timeslot in Classroom
- Students notified of Makeup Lecture schedule

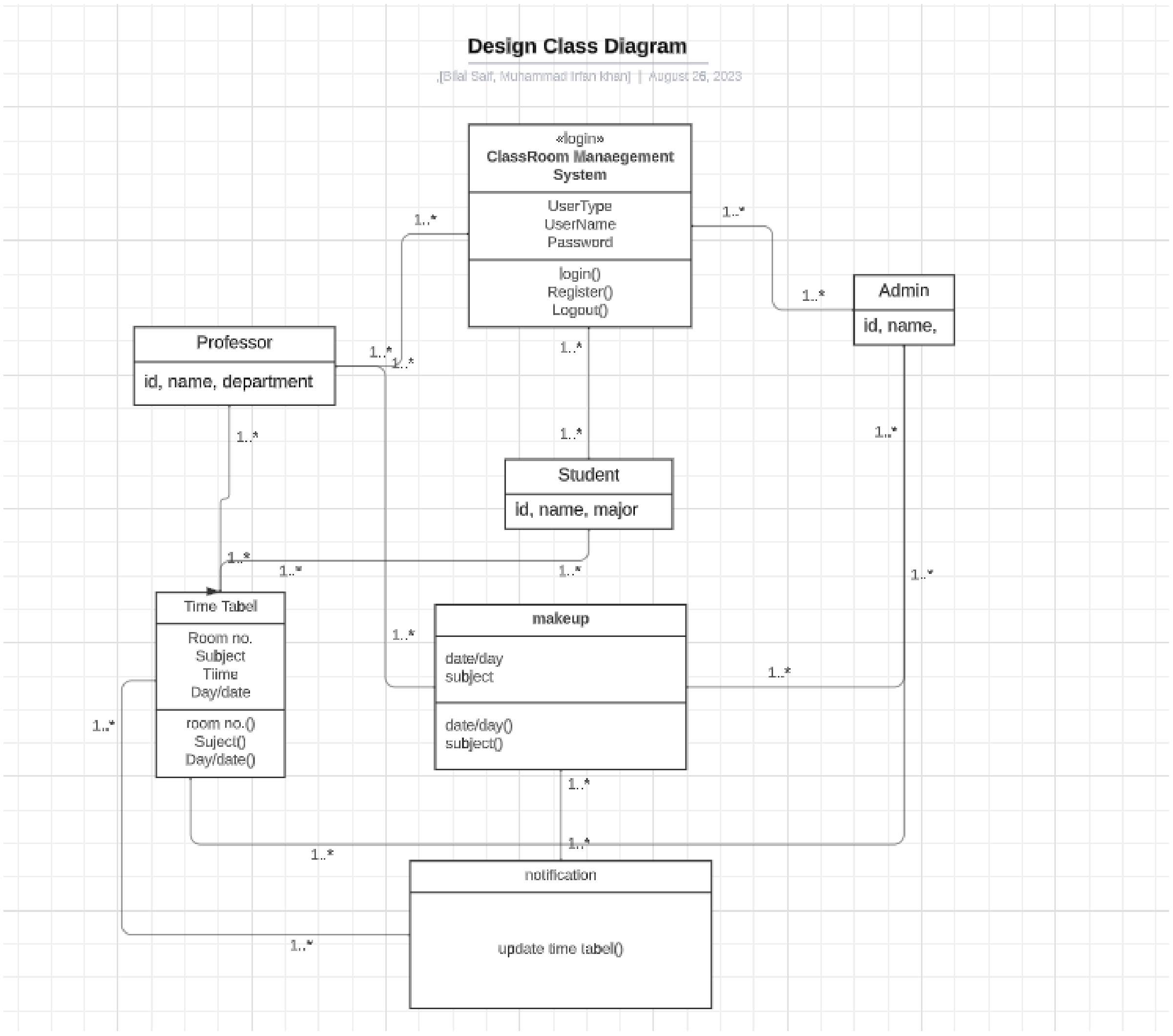
Key Attributes:
- Professor: id, name, department
- Student: id, name, major
- Course: id, name, code
- Classroom: id, number, capacity
- Timetable: id, date, timeslot
- Makeup Lecture: id, date, time, course, classroom

Use Cases:
- Professor requests makeup lecture
- System checks classroom availability
- System books makeup lecture into timetable
- System notifies students

3.3. Design Class Diagram

The key classes in the diagram are:

- Professor: Stores professor details like id, name, department etc.
- Student: Stores student details like id, name, major etc.
- Course: Stores course details like id, name, and code.
- Classroom: Stores classroom details like id, number, and capacity.
- Timetable: Stores timetable slots with date, time, course etc.
- MakeupLecture: Request and booked makeup lectures linked to Course, Classroom and Timetable.



3.4. Sequence Diagram

The key steps are:

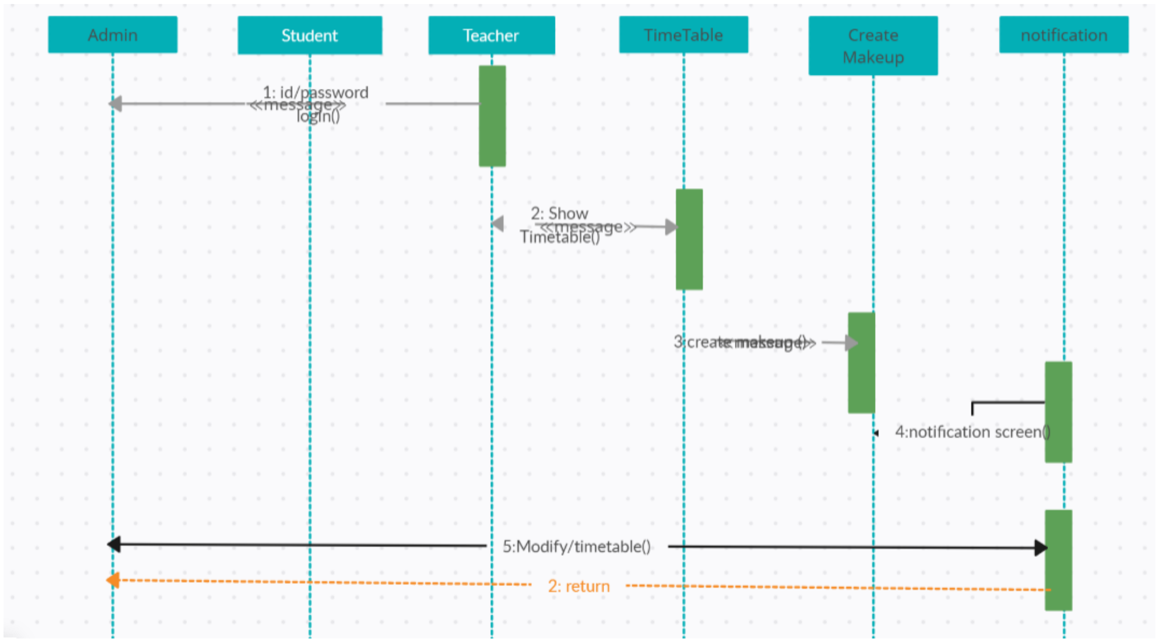
1. Professor requests makeup lecture for a course by providing date/time
2. MakeupLecture object is created with the request details
3. MakeupLecture checks classroom availability by interacting with Classroom object
4. If classroom available, MakeupLecture books by updating Timetable
5. MakeupLecture sends notification to enrolled Students of the course
6. Professor receives confirmation that lecture is booked

The main objects involved are:

- Professor - Requests makeup lecture
- MakeupLecture - Encapsulates booking logic
- Classroom - Checks availability
- Timetable - Gets updated with booking
- Student - Receives notification

Some key methods used:

- requestMakeupLecture()
- checkAvailability()
- bookTimetable()
- sendNotification()



Sequence Diagram

3.4.1. Distributing Control Flow in Sequence Diagrams

Centralized control of a flow of events or part of the flow of events means that a few objects steer the flow by sending messages to, and receiving messages from other objects. These controlling objects decide the order in which other objects will be activated in the use case. Interaction among the rest of the objects is very minor or does not exist.

The control flow is distributed among multiple objects rather than centralized to one or two objects.

The key steps are:

1. Professor requests makeup lecture through MakeupLectureController
2. MakeupLectureController creates MakeupLecture object with request details
3. MakeupLecture checks availability by interacting with Classroom
4. Classroom checks in Timetable if available
5. If available, MakeupLecture books by updating Timetable
6. Timetable updates and saves booking
7. MakeupLecture requests NotificationManager to send notifications
8. NotificationManager sends notification to all enrolled Students
9. MakeupLectureController returns confirmation to Professor

Here the control flow is distributed among multiple classes:

- MakeupLectureController - gets request and returns confirmation
- MakeupLecture - encapsulates booking logic
- Classroom - checks availability
- Timetable - handles booking data
- NotificationManager - handles sending notifications

This allows separation of responsibilities and modular code.

3.5. State chart diagram

The key states of a MakeupLecture are:

- Created - Initial state when request is received
- AvailabilityChecked - After checking classroom availability
- Booked - Once booked into timetable
- Notified - When students are notified
- Cancelled - If lecture is cancelled

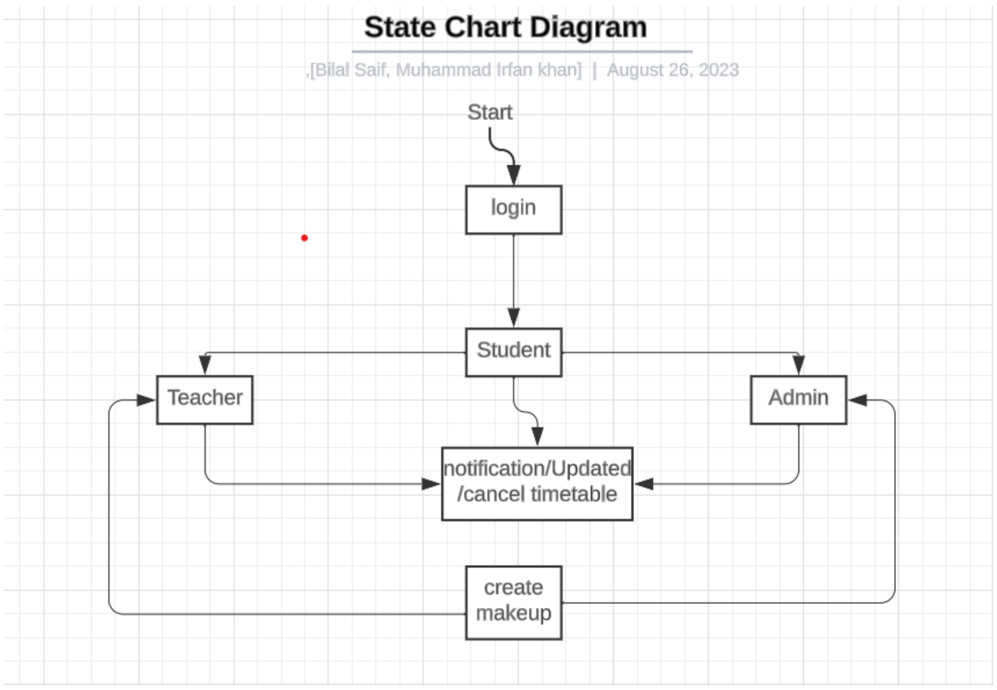
The transitions between states are triggered by events like:

- checkAvailability()
- bookTimetable()
- sendNotification()
- cancelMakeupLecture()

Some behaviors based on state:

- Cannot book unless availability is checked
- Cannot notify unless lecture is booked
- Cannot cancel unless in booked state

The current state determines what operations are valid. This helps model the lecture lifecycle.


State Chart Diagram
,[Bilal Saif, Muhammad Irfan khan] | August 26, 2023
Start
login
Student
Teacher
Admin
notification/Updated /cancel timetable
create makeup

The key objects involved are:

- Professor - Requests makeup lecture
- MakeupLecture - Encapsulates booking logic
- Classroom - Checks availability
- Timetable - Gets updated with booking
- Student - Receives notification

The key interactions are:

1. Professor requests MakeupLecture
2. MakeupLecture checks availability with Classroom
3. Classroom checks in Timetable
4. Timetable updates booking
5. MakeupLecture sends notification to Students
6. Professor receives confirmation

This collaboration diagram shows:

- The objects involved in the use case
- The relationships between the objects
- The sequence of messages exchanged

It provides a visual representation of how the objects collaborate to realize the use case.

A collaboration diagram that describes part of the flow of events of the use case Receive Deposit Item in the Recycling-Machine Sys

Chapter 4: User Interface Design

4.1. Introduction

Developing a user-friendly interface for such a system can greatly enhance its usability and effectiveness. Let's break down our introduction and see how we can elaborate on each point:

Introduce the concept of our "Class Room Management System" and its purpose. Highlight the importance of efficient classroom scheduling and management in educational institutions. We can mention how manual scheduling can be error-prone and time-consuming, and that our application aims to streamline this process.

User Interface Design:

1. **Overview:** The significance of a well-designed user interface (UI) in software applications. A user-friendly UI can improve user adoption, ease of use, and overall user satisfaction. That our project focuses on creating an intuitive and visually appealing UI for the "Class Room Management System."

2. **Flutter Framework**: Introduction to the Flutter framework. Explain that Flutter is a popular open-source UI software development kit (SDK) developed by Google, known for building natively compiled applications for mobile, web, and desktop from a single codebase. Highlight its advantages such as fast development, expressive UI components, and platform consistency.

3. **Firebase Integration:** Firebase as a cloud-based platform that offers various backend services to support app development. Mention that Firebase can be used for real-time databases, authentication, hosting, and more. Our project leverages Firebase for features like data storage, user authentication, and potentially real-time updates for the classroom availability status.

4. **Project Scope**: Overview of the specific features our project will focus on. The primary goal is to create an interface that allows users to efficiently manage classroom schedules and make-up lectures. The intuitive process of creating makeup lectures, selecting dates, and viewing room availability.

Next Steps:

We can continue our introduction by briefly outlining the subsequent sections of our documentation. These sections could include:

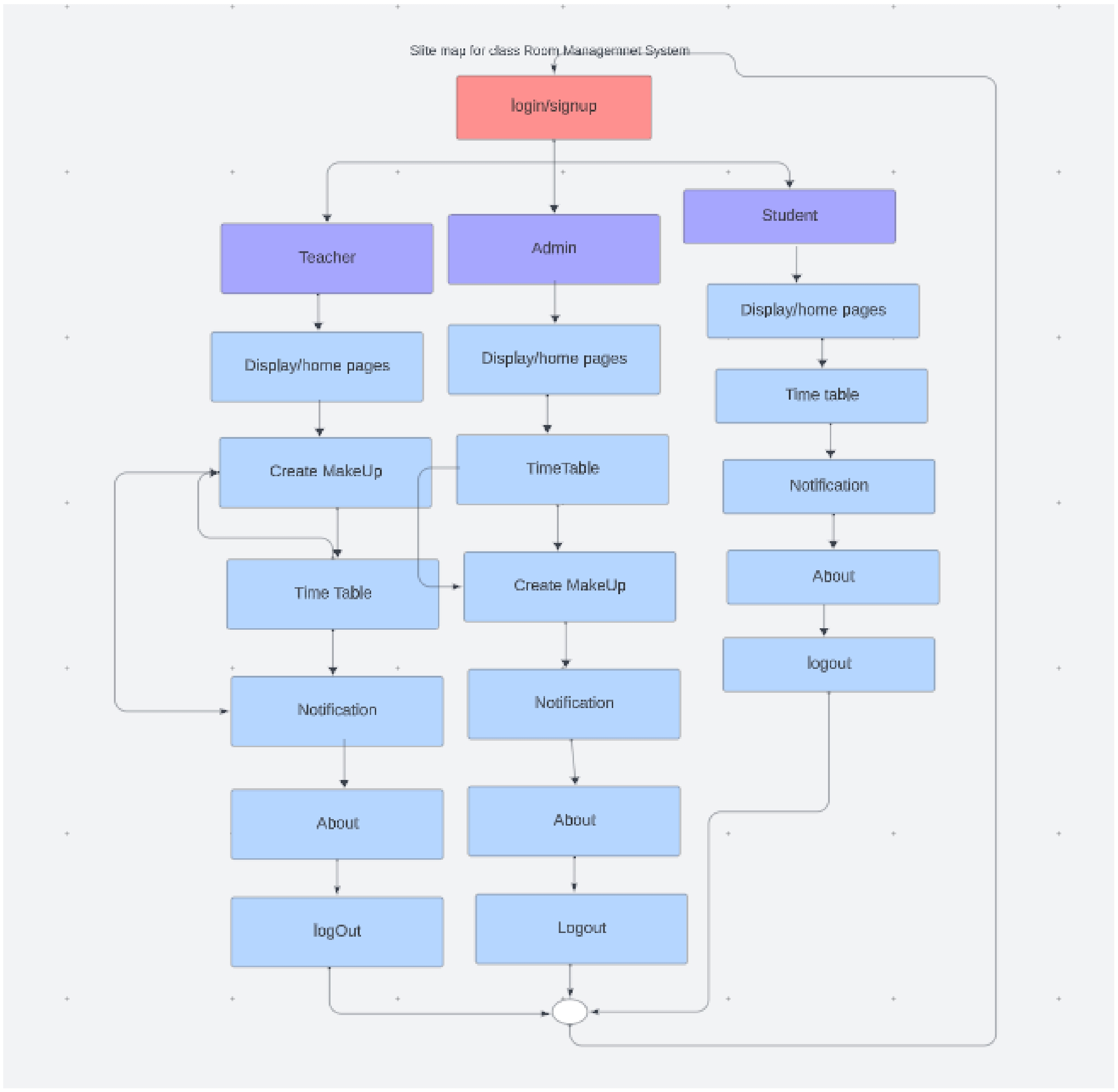
- **System Architecture**: Explain the overall architecture of our application, including the front-end built with Flutter and the backend integrated with Firebase services.

- **User Workflows:** The step-by-step user workflows for creating makeup lectures, selecting dates, and viewing room availability. Provide screenshots or wireframes to visualize these workflows.

- **UI Design Principles**: The design principles and guidelines we are following to create a visually appealing and user-friendly interface. Highlight concepts such as color coding for room availability, intuitive navigation, and responsive design.

- **Technology Stack**: A comprehensive list of technologies we are using, including specific Flutter packages for UI design, Firebase services for backend functionality, and any other tools or libraries.

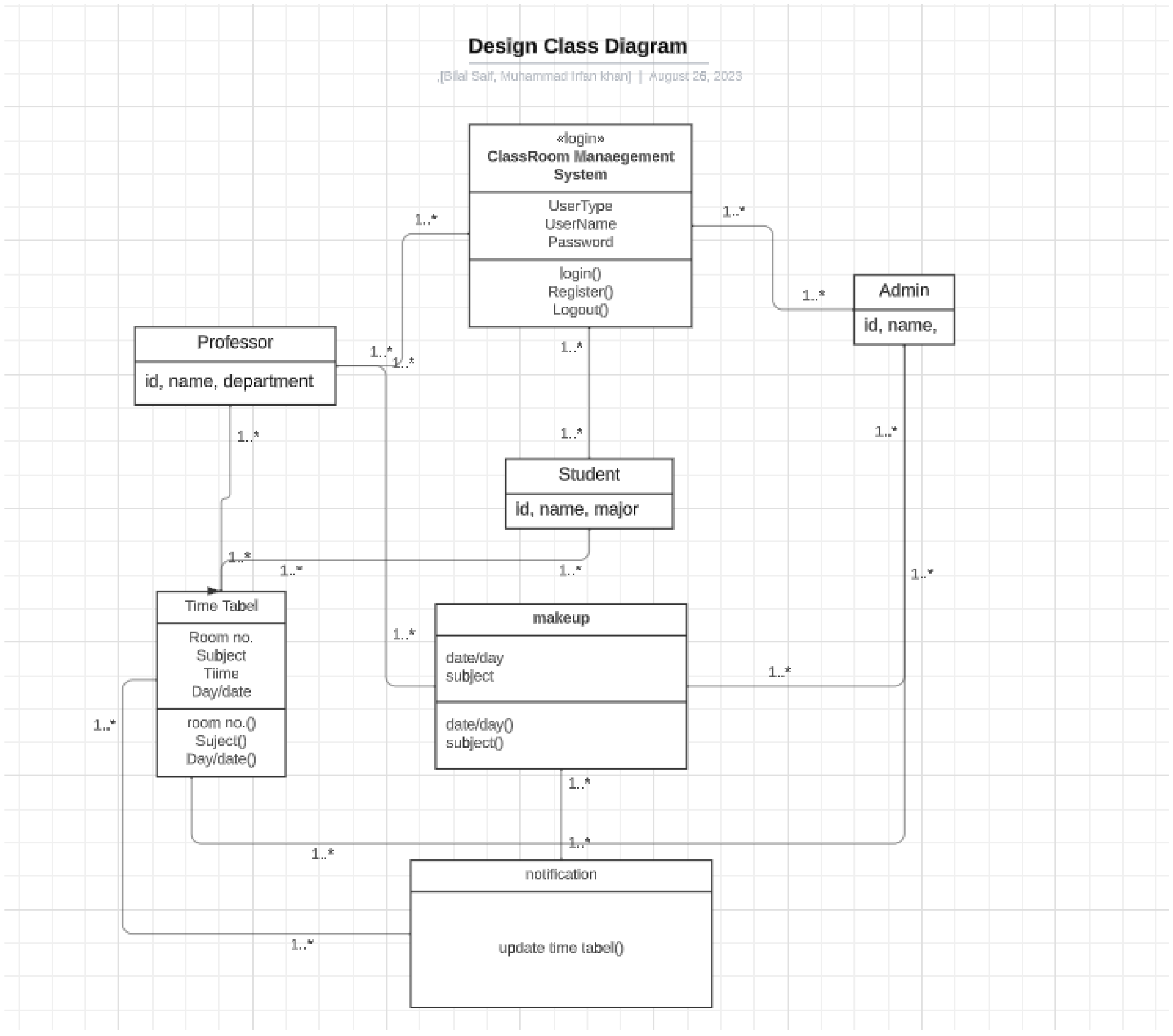
.4.2. Site Maps



Site map for Class Room management system

4.3. Story boards:

Project related to classroom management using Flutter and Firebase. Creating a user-friendly interface for our application is crucial, and storyboards can help we visualize the flow and interactions within our app. Here's a step-by-step guide on how to proceed with our User Interface Design using storyboards for our "Classroom Management System" project:



1. **Identify User Flows:**
The various user flows and scenarios that our application will handle. This could include actions like logging in, viewing the timetable, creating makeup lectures, checking room availability, and more.

2. **Create Storyboards:**
For each identified user flow, create storyboards that outline the sequence of screens and interactions. Use sketches or digital tools to design these storyboard frames. Each frame should represent a different screen or state of our application.

3. **Design Screens:**
Translate the storyboard frames into actual screens. Design the UI elements using Flutter widgets based on our sketches. Ensure consistency in design, color scheme, and typography across screens to provide a cohesive user experience.

4. **Implement Navigation:**
The navigation paths between screens. In Flutter, we can use the `Navigator` class to manage the stack of screens. Make sure the navigation flow is intuitive and allows users to move smoothly between different sections of the app.

5. **Dynamic Data Handling:**
Since our app interacts with Firebase to display timetable data, we'll need to integrate Firebase into our Flutter app. Fetch data from Firebase and populate our screens with dynamic content based on user inputs.

6. **Create Makeup Page:**
Implement the "Create Makeup" page where users can select the date and subject for the makeup lecture. Design the UI elements for date selection and subject input. Upon submitting, the app should fetch and display the timetable for the selected date.

7. **Room Availability Visualization:**
The timetable view with color-coded rooms. Use red for empty rooms and green for occupied ones. Implement this view based on the data retrieved from Firebase. We might use Flutter's `GridView` or `Table` widgets for this purpose.

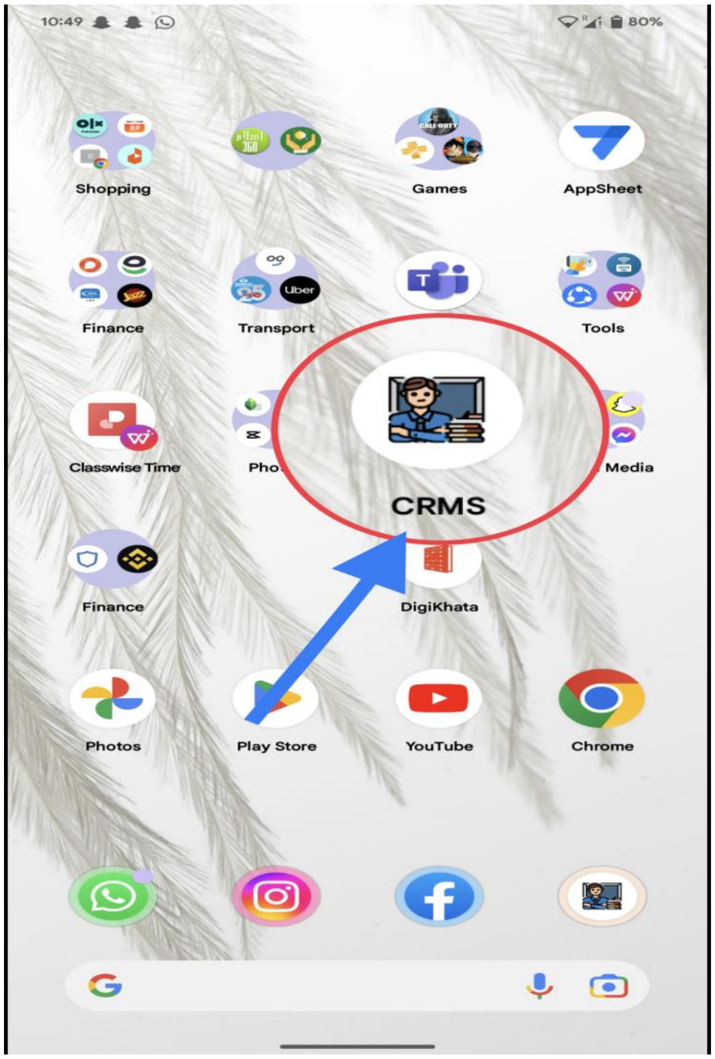
8. **Interaction and Feedback:**
Ensure that buttons, forms, and other interactive elements provide visual feedback to users when tapped. Consider using animations or changing button colors to indicate interactions.

9. **Testing:**
Regularly test our app on different devices and screen sizes to ensure responsive design. Also, test different scenarios to ensure the app behaves as expected, fetching data accurately from Firebase.

10. **Iterative Design:**
Design is an iterative process. Gather feedback from peers or potential users and refine our UI based on their suggestions. Pay attention to user experience and make necessary adjustments.

Remember that documentation should also include explanations of each screen, interactions, and the rationale behind design decisions. The documentation will help reviewers understand our thought process and the functionality of our application.

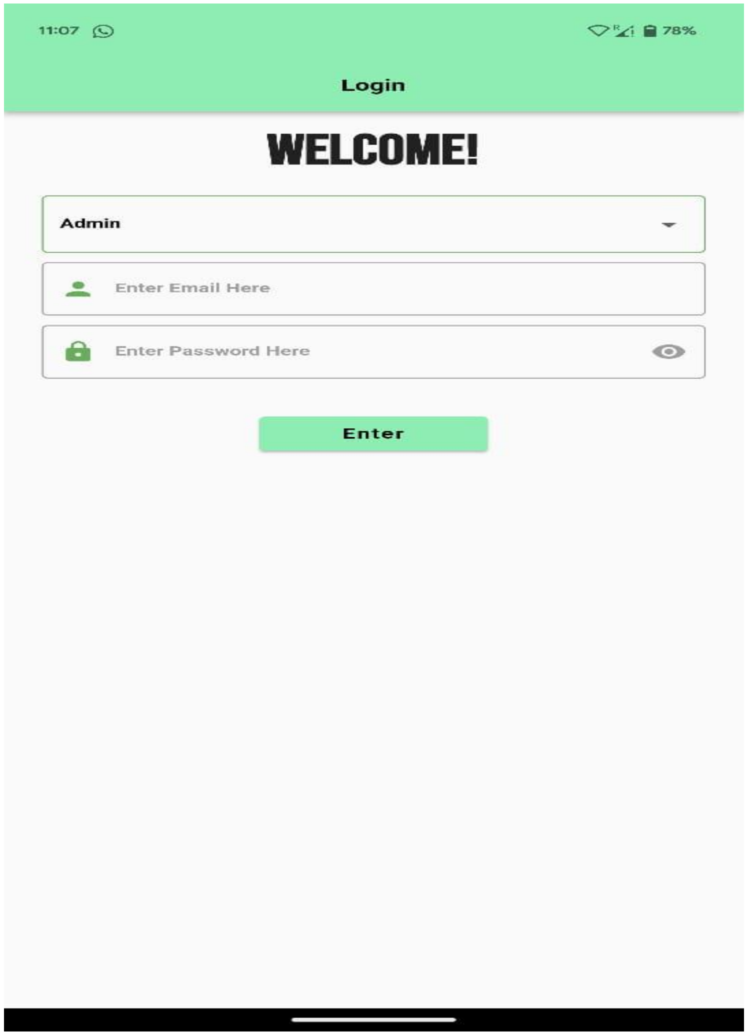
4.3. Story boards:



Story boards

Storyboards for our classroom management system project can help visualize the user interactions and flow of the application. Start with a login page for students, teachers, and admin. Upon login, teachers can access a dashboard to create makeup lectures with details like date, time, and classroom. Admin will have an approval dashboard to review and confirm makeup lectures. Students can view their respective timetables and receive notifications for upcoming makeup lectures. Each user role should have their own set of functionalities and user interfaces. Storyboards will provide a clear visual representation of the application's functionality and user journeys.

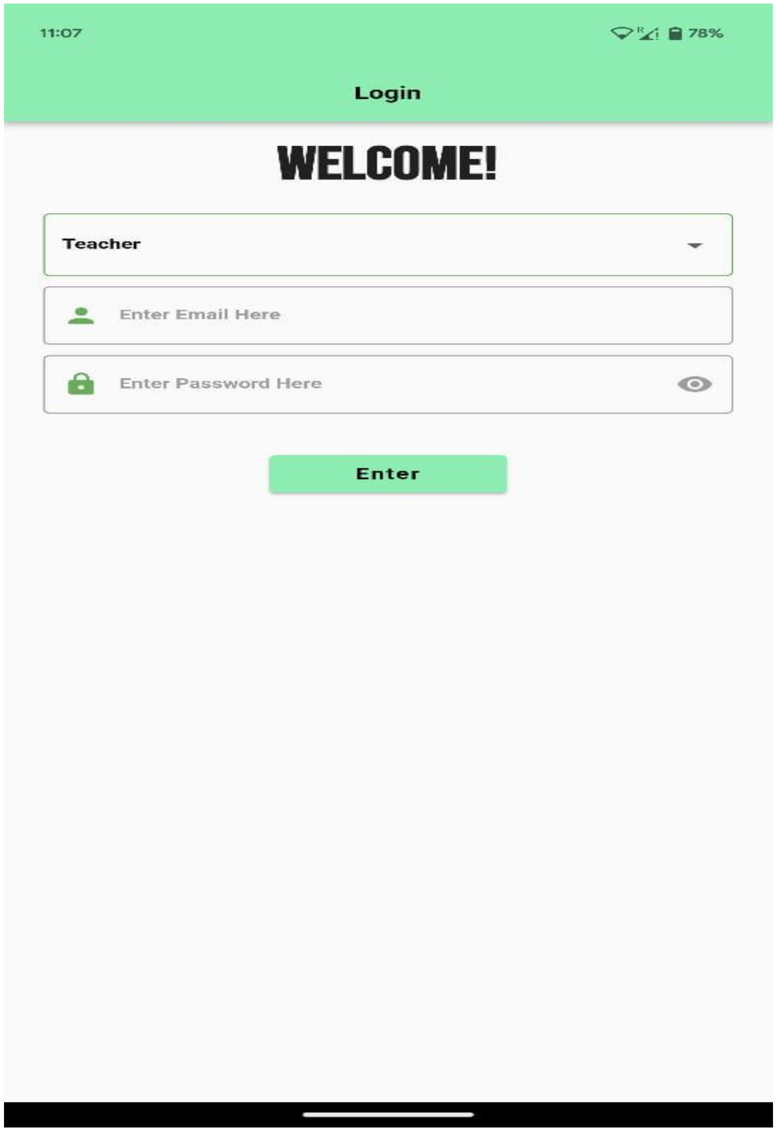
Admin Dashboard:



Admin Dashboard

The Admin Dashboard is a crucial component of the Classroom Management System application. It serves as the central control hub for administrators to oversee and manage makeup lectures. In this section of the project documentation, we will outline the key functionalities of the Admin Dashboard. Admins will have the ability to review and approve makeup lecture requests submitted by professors, ensuring their suitability and alignment with the schedule. They can also view and edit the overall timetable for classrooms, adjusting as necessary. Additionally, the Admin Dashboard will provide a comprehensive overview of upcoming makeup lectures, allowing administrators to stay informed and organized in managing classroom resources efficiently.

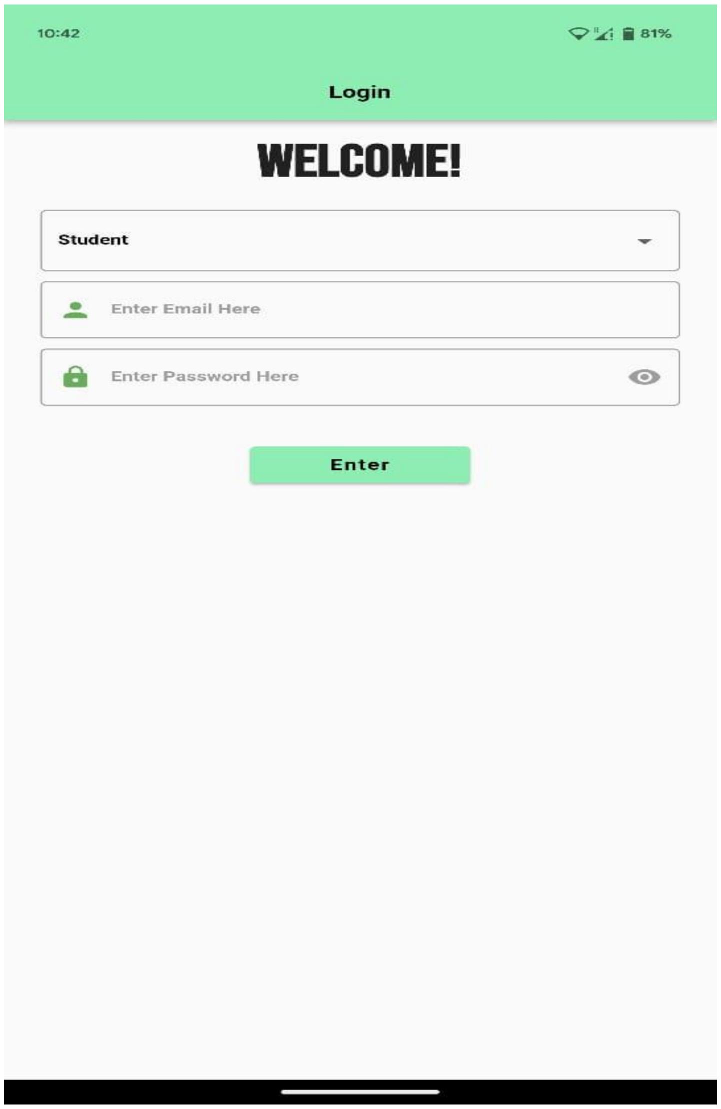
Teacher Dashboard:



Teacher Dashboard

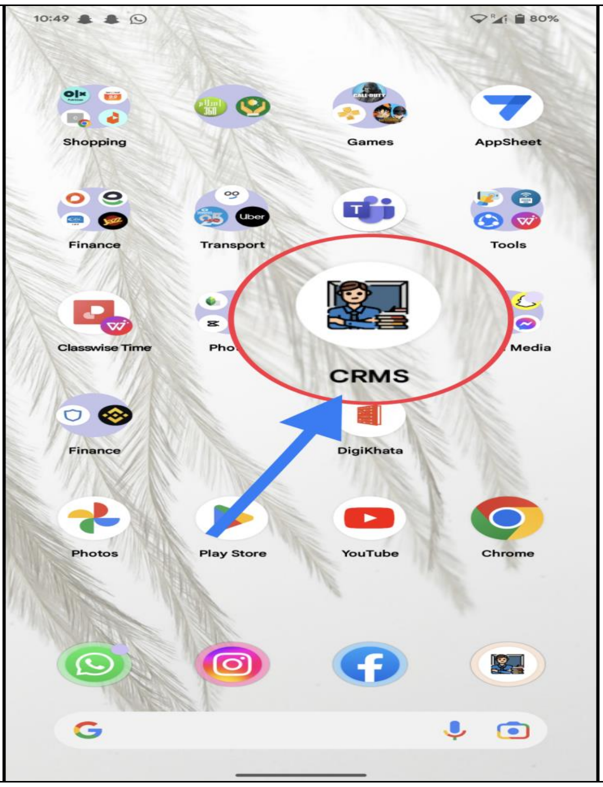
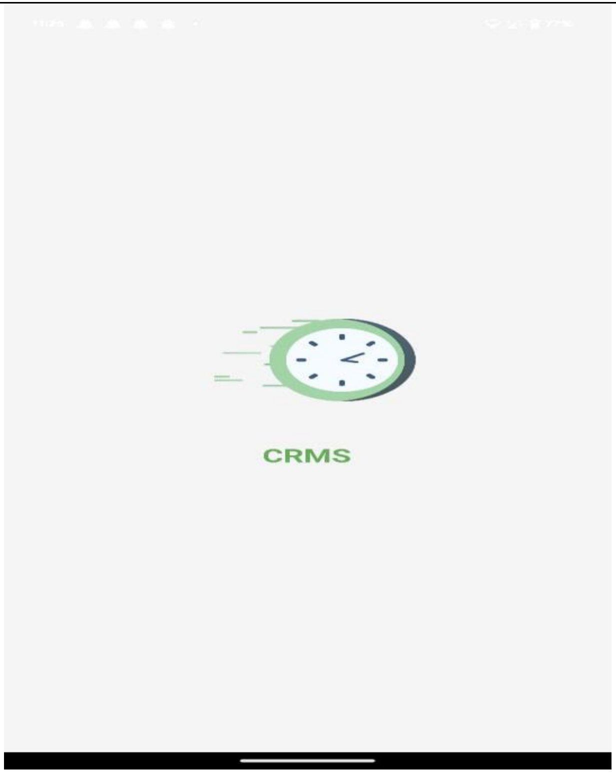
The Teacher Dashboard in our Classroom Management System application will provide professors with the ability to create and schedule makeup lectures. Professors will have a user-friendly interface where they can input the date, time, and classroom for the makeup lecture. They will also be able to specify the reason for the makeup lecture and provide any additional information. Once the makeup lecture is created, it will be sent to the admin for approval. The Teacher Dashboard will also allow professors to view their regular class schedules and any upcoming makeup lectures they have scheduled. It will streamline the process of arranging makeup lectures and ensure efficient communication between professors, admin, and students.

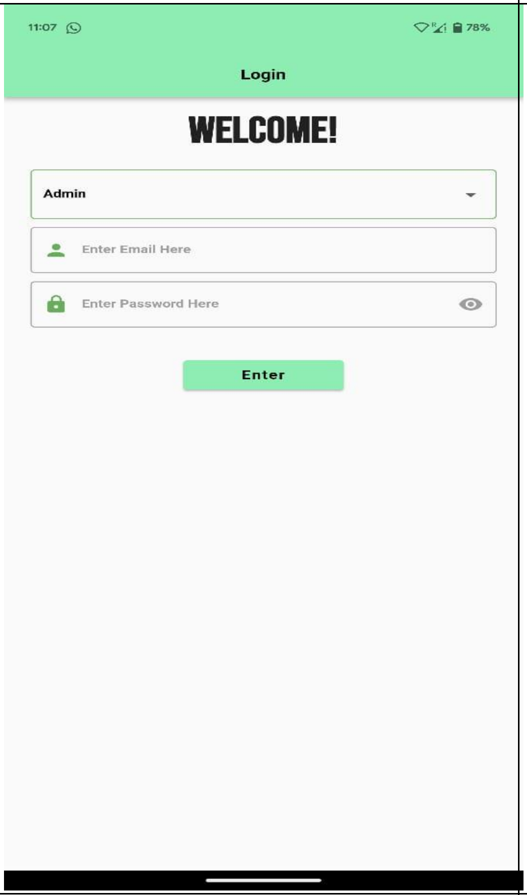
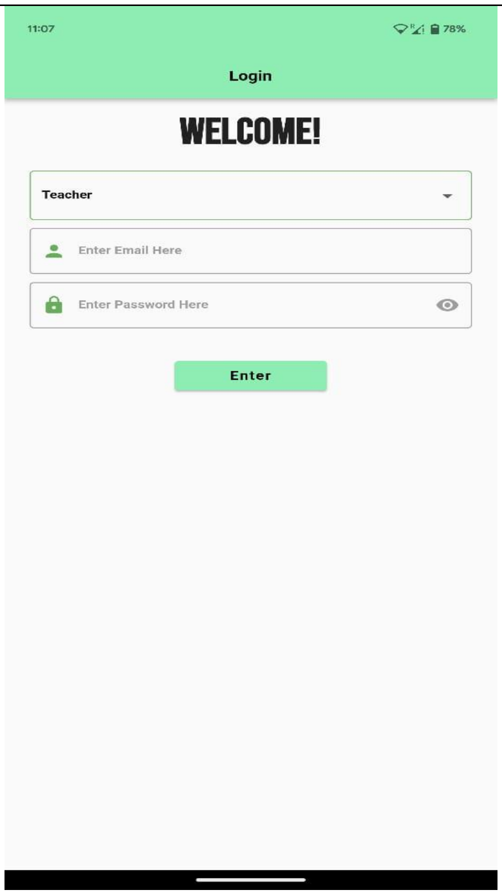
Student Dashboard:

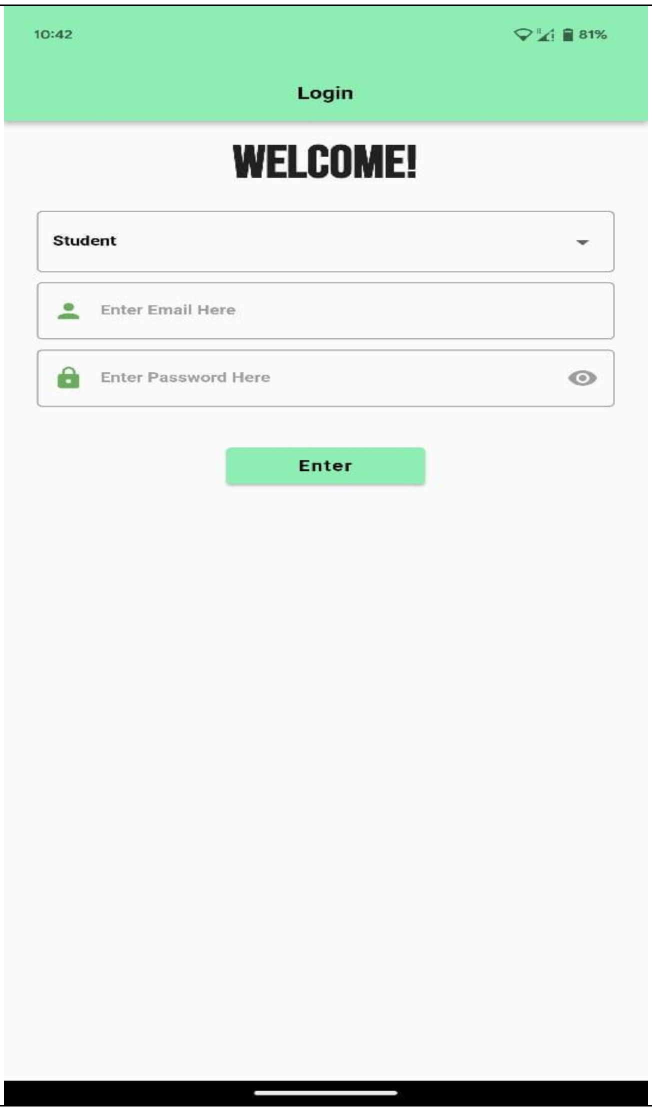


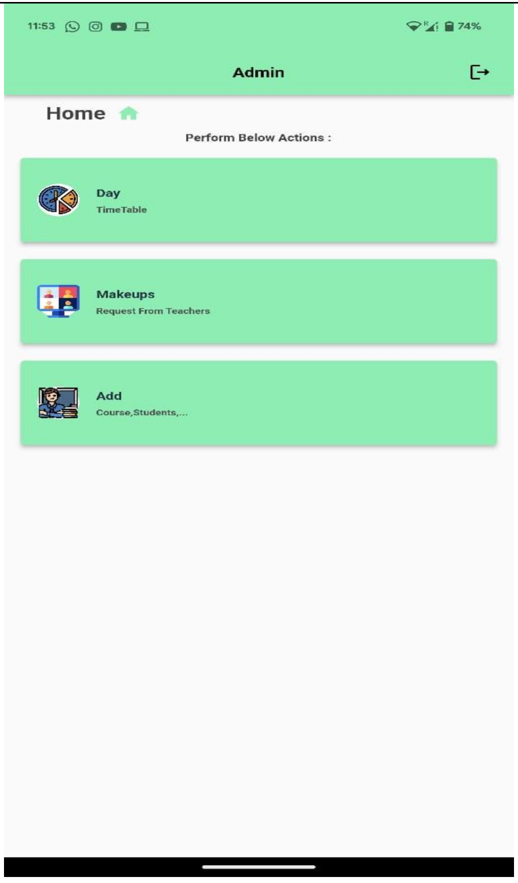
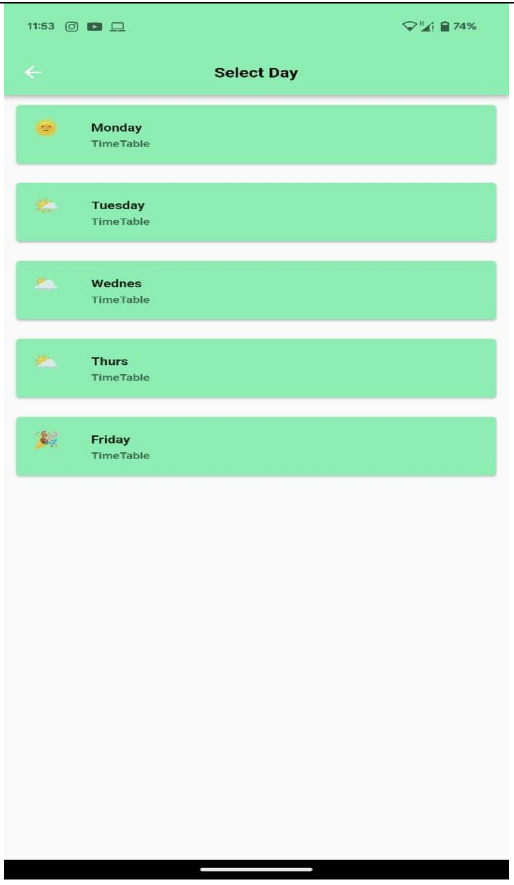
Student Dashboard

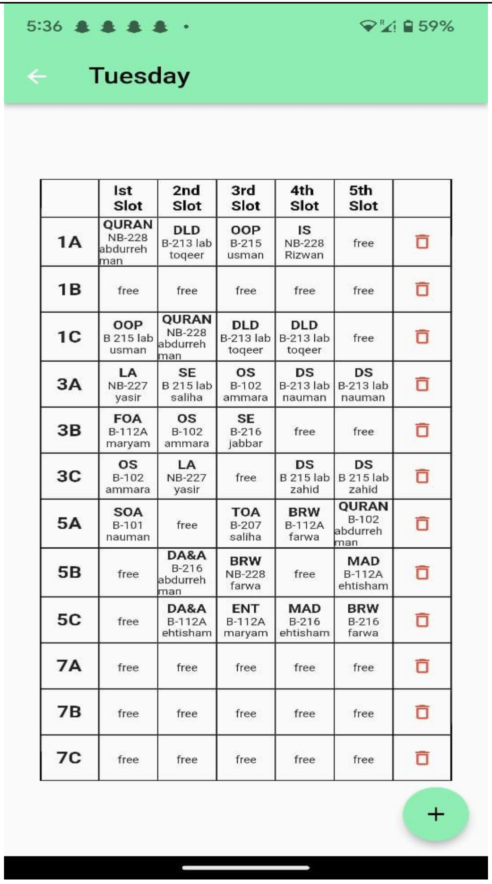
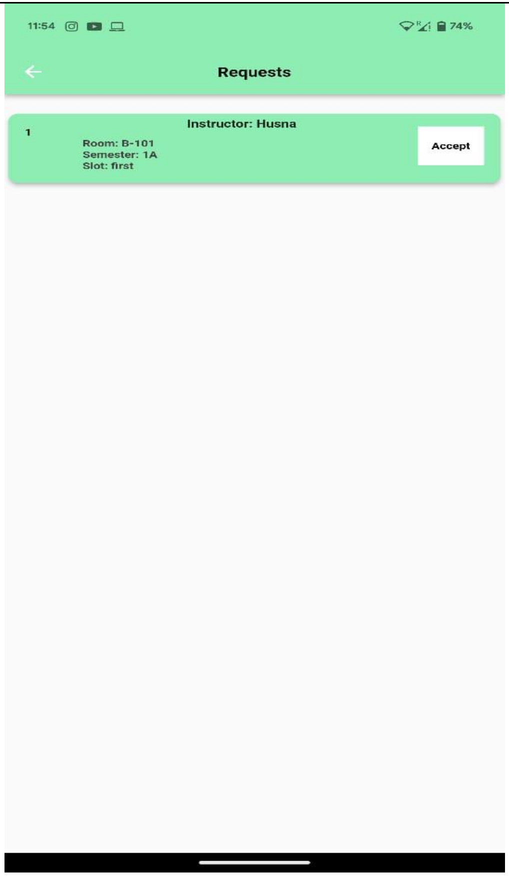
The Student Dashboard in our Classroom Management System application serves as a user-friendly interface for students to access essential information regarding their class schedules and makeup lectures. Students can log in to view their personalized timetables, receive notifications about upcoming makeup lectures, and stay updated on any changes in their class schedule. This dashboard provides a convenient and organized way for students to manage their academic commitments and ensures they are well-informed about any makeup lectures arranged by the professors and approved by the admin. It prioritizes user experience and accessibility for students, enhancing their overall educational experience**.**

UI 1	1	2
Screen image	 Shopping Games AppSheet Finance Transport Tools Classwise Time Media CRMS Finance DigiKhata Photos Play Store YouTube Chrome	
Visual Cues	This is the application icon named as" **CRMS."** If the user clicks on it than the splash Screen open and after that login screen	This is the Splash Screen. Clicking on the Splash screen nothing happened
Tactile Cues	Application Icon	Login

UI ID	**3**	**4**
welcome Screen/ Sign In	 11:07 78% Login WELCOME! Admin Enter Email Here Enter Password Here Enter	 11:07 78% Login WELCOME! Teacher Enter Email Here Enter Password Here Enter
Visual Cues	This is the SignIn screen. the admin, sign form here	This is the sign In screen, he will have to create the sign up form admin sign In if the teacher is new
Tactile Cues	Sign In for admin	Sign In for teacher
User input	The admin have to enter the email and password in press the button	The Teacher have to enter the email and password in press the button
Machine output	After email and password the admin have to hit the enter button	After email and password the Teacher have to hit the enter button

UI ID	**5**
welcome Screen/ Sign In	 10:42 81% Login WELCOME! Student Enter Email Here Enter Password Here Enter
Visual Cues	This is the Sign in screen. the student, sign form here
Tactile Cues	Sign In for Student
User input	The student has to enter the email and password in press the button
Machine output	After email and password, the student have to hit the enter button

UI Admin	**6**	**7**
Admin display	 11:53 Admin Home Perform Below Actions : Day TimeTable Makeups Request From Teachers Add Course,Students,...	 11:53 Select Day Monday TimeTable Tuesday TimeTable Wednes TimeTable Thurs TimeTable Friday TimeTable
Visual Cues	This is the admin display and their many options. 1.Day 2.makeups 3.Add	On clicking in the days Section this is the screen in which there are name od days 1.Monday 2.Tuesday 3.wednesday 4.Thursday 5.Friday
Tactile Cues	Click on Days	Click on Monday
User input	Nothing	Nothing
Machine output	This is admin display and form here he or she can control all the application	In this there are the timetable od each semester and their section

UI Admin	8	9
Admin display / Timetable of Monday		
Visual Cues	On clicking on the Tuesday the next UI is the time table of Monday their many options 1.add	On clicking in the Makeup, the next Is the request of all the makeups from the teachers
Tactile Cues	Click on back	Click on Accept
User input	Nothing	Nothing
Machine output	This is the display off Monday timetable and this timetable is vires form different semester and their section	In this there are the request of makeup lecture arrange by the teachers and the admin have the access to accept the request or reject.

5:36 59%

Tuesday

	Ist Slot	2nd Slot	3rd Slot	4th Slot	5th Slot	
1A	**QURAN** NB-228 abdurrehman	**DLD** B-213 lab toqeer	**OOP** B-215 usman	**IS** NB-228 Rizwan	free	🗑
1B	free	free	free	free	free	🗑
1C	**OOP** B 215 lab usman	**QURAN** NB-228 abdurrehman	**DLD** B-213 lab toqeer	**DLD** B-213 lab toqeer	free	🗑
3A	**LA** NB-227 yasir	**SE** B 215 lab saliha	**OS** B-102 ammara	**DS** B-213 lab nauman	**DS** B-213 lab nauman	🗑
3B	**FOA** B-112A maryam	**OS** B-102 ammara	**SE** B-216 jabbar	free	free	🗑
3C	**OS** B-102 ammara	**LA** NB-227 yasir	free	**DS** B 215 lab zahid	**DS** B 215 lab zahid	🗑
5A	**SOA** B-101 nauman	free	**TOA** B-207 saliha	**BRW** B-112A farwa	**QURAN** B-102 abdurrehman	🗑
5B	free	**DA&A** B-216 abdurrehman	**BRW** NB-228 farwa	free	**MAD** B-112A ehtisham	🗑
5C	free	**DA&A** B-112A ehtisham	**ENT** B-112A maryam	**MAD** B-216 ehtisham	**BRW** B-216 farwa	🗑
7A	free	free	free	free	free	🗑
7B	free	free	free	free	free	🗑
7C	free	free	free	free	free	🗑

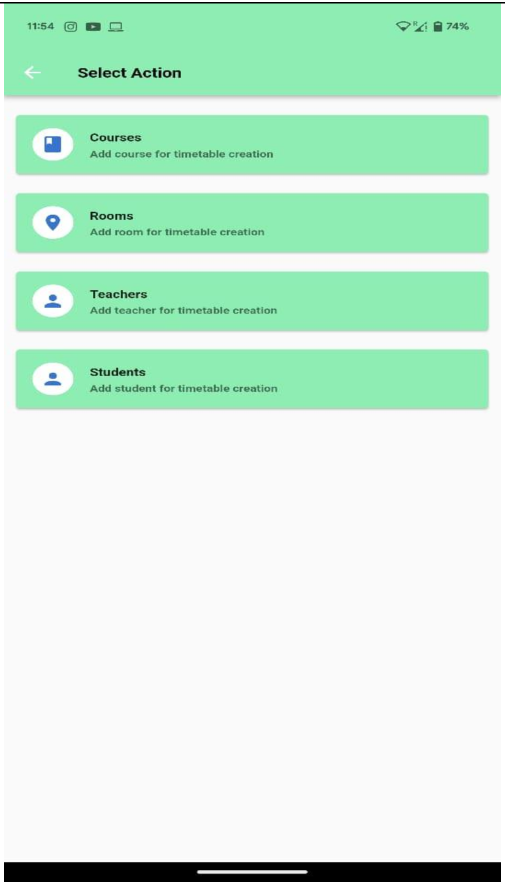
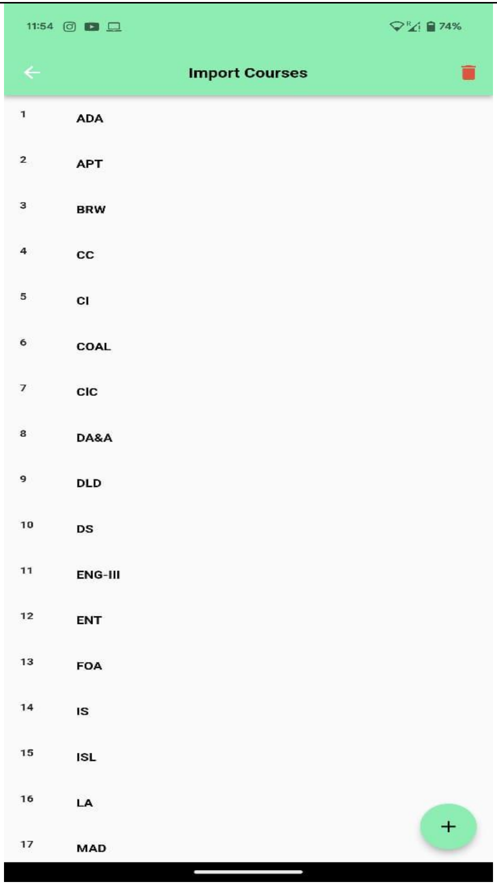
+

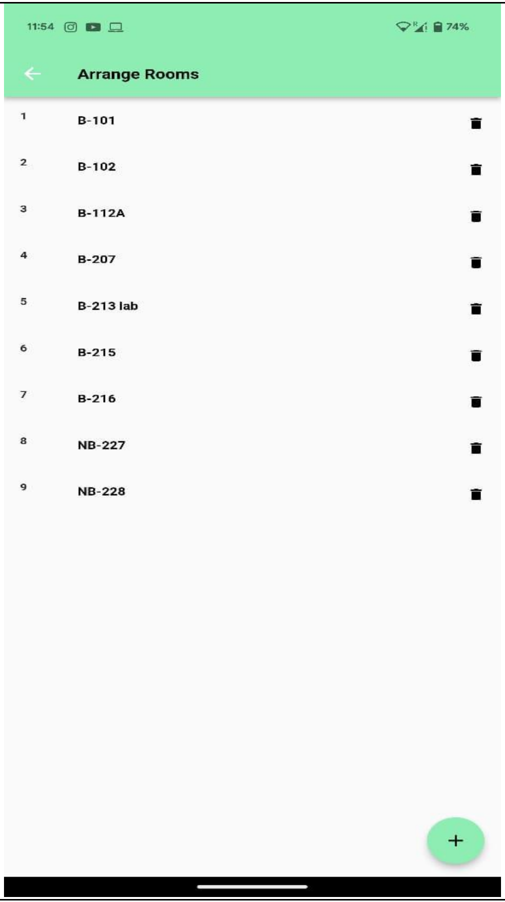
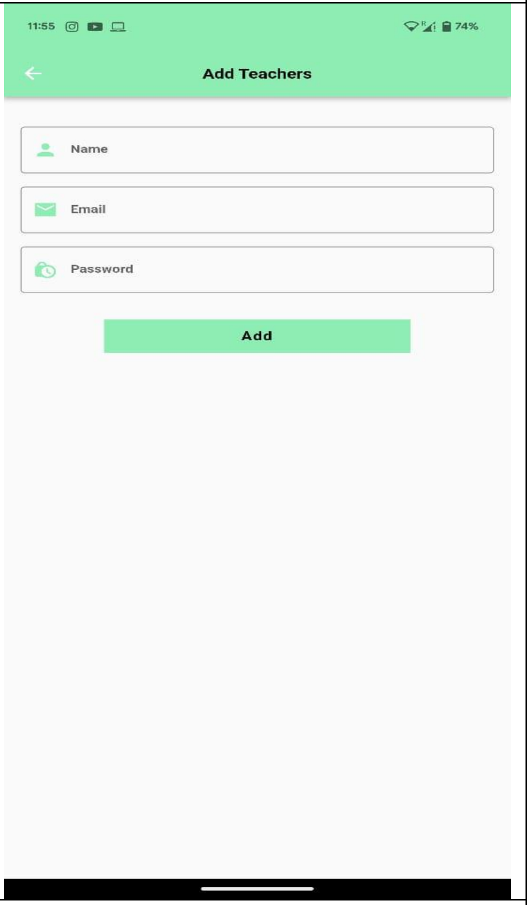
11:54 74%

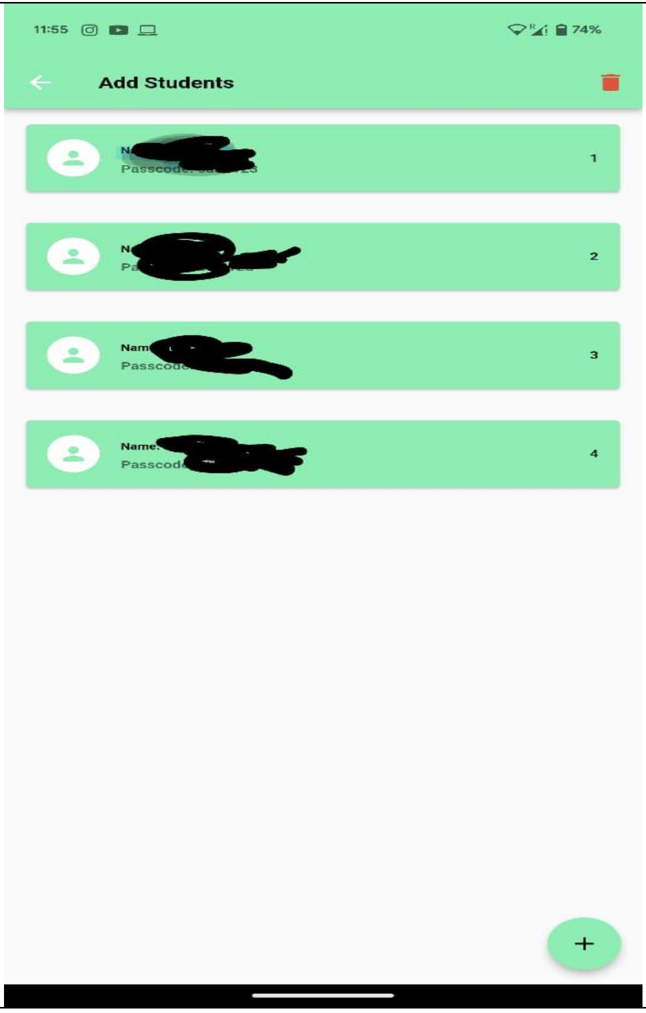
Requests

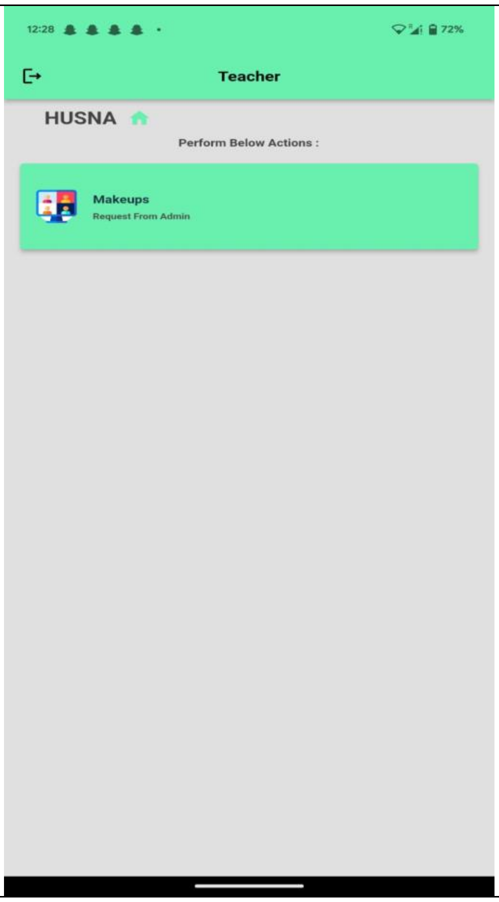
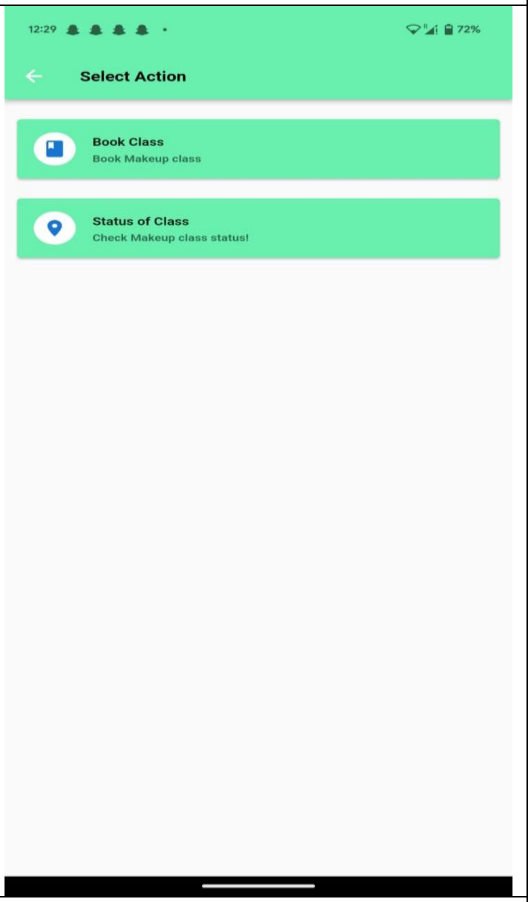
1 Instructor: Husna
Room: B-101
Semester: 1A
Slot: first

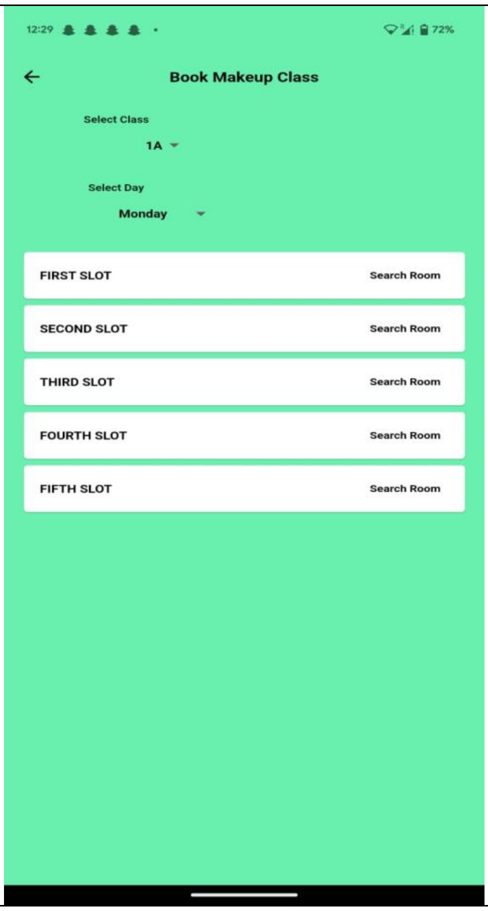
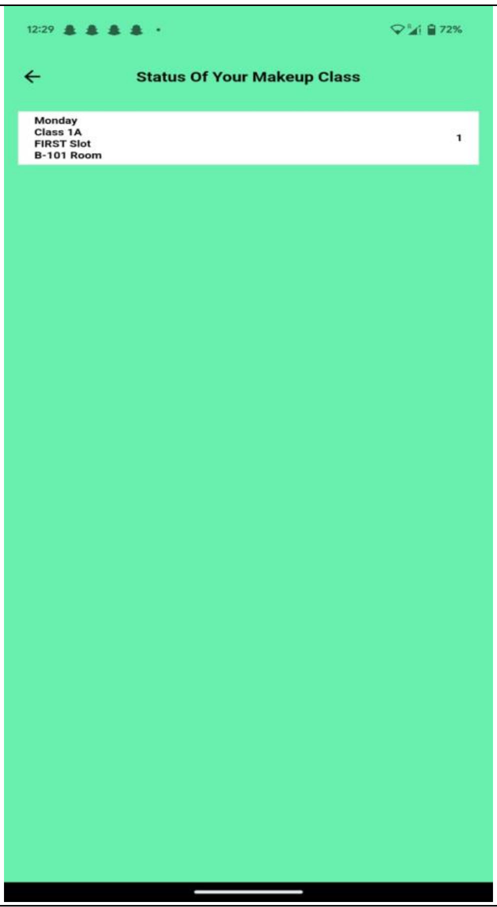
Accept

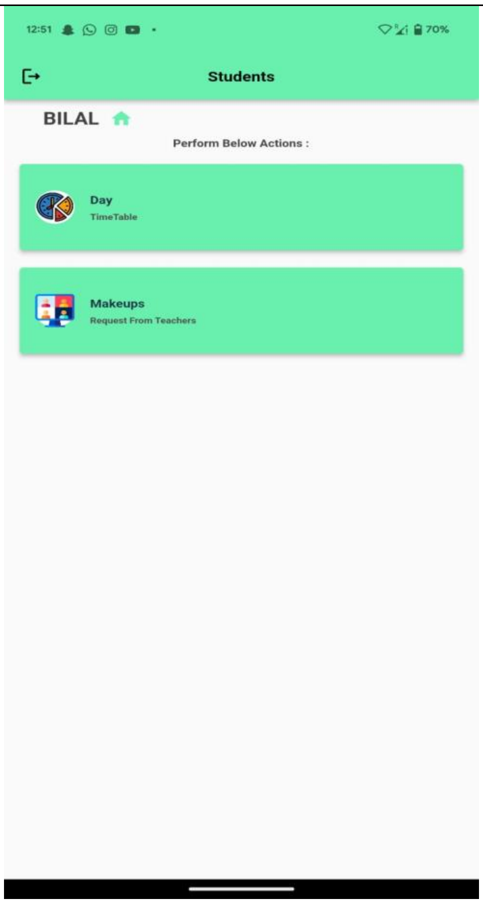
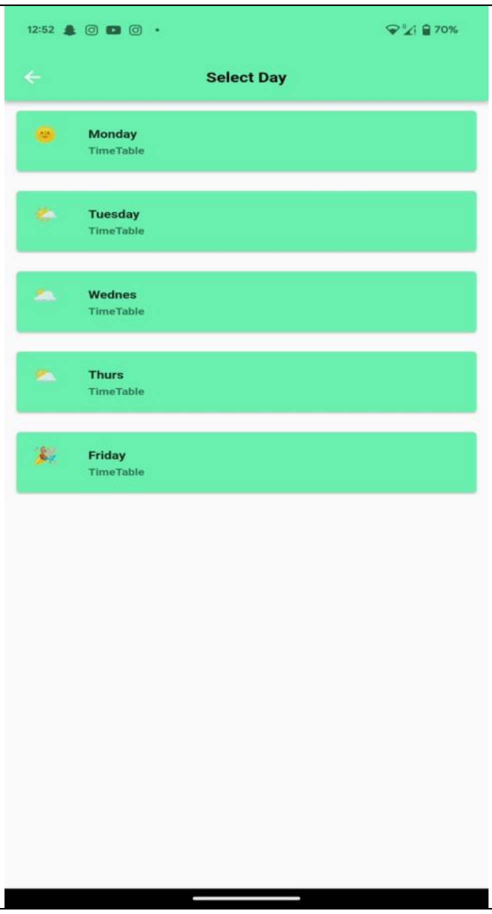
UI Admin	**10**	**11**
Admin display / Select Action	 11:54 74% Select Action Courses – Add course for timetable creation Rooms – Add room for timetable creation Teachers – Add teacher for timetable creation Students – Add student for timetable creation	 11:54 74% Import Courses 1 ADA 2 APT 3 BRW 4 CC 5 CI 6 COAL 7 CIC 8 DA&A 9 DLD 10 DS 11 ENG-III 12 ENT 13 FOA 14 IS 15 ISL 16 LA 17 MAD
Visual Cues	On clicking on the Add these are the options visible 1.courses 2.room 3.teacher 4.student	On clicking in the courses these are the courses that have be add in the application
Tactile Cues	Click on courses	Click on back
User input	Nothing	Nothing
Machine output	This is the display the courses I have to click on the Courses	In this there are many courses added in and if we have add more courses than we have to click on the add button

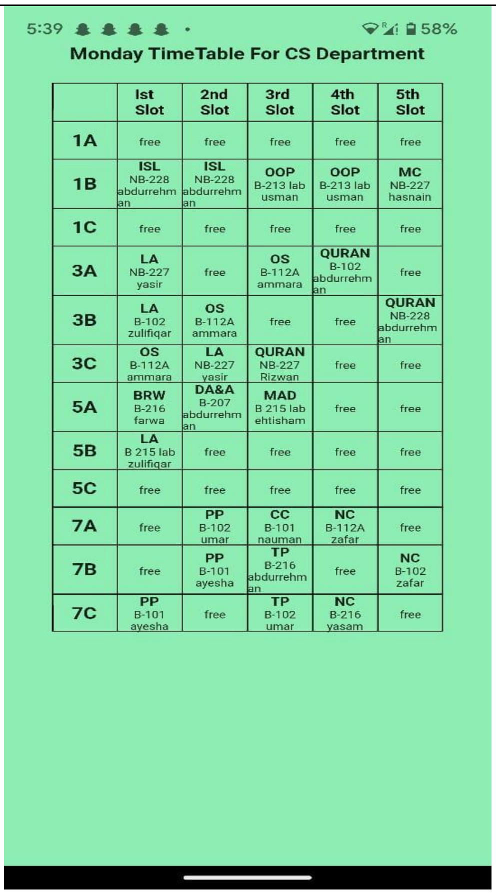
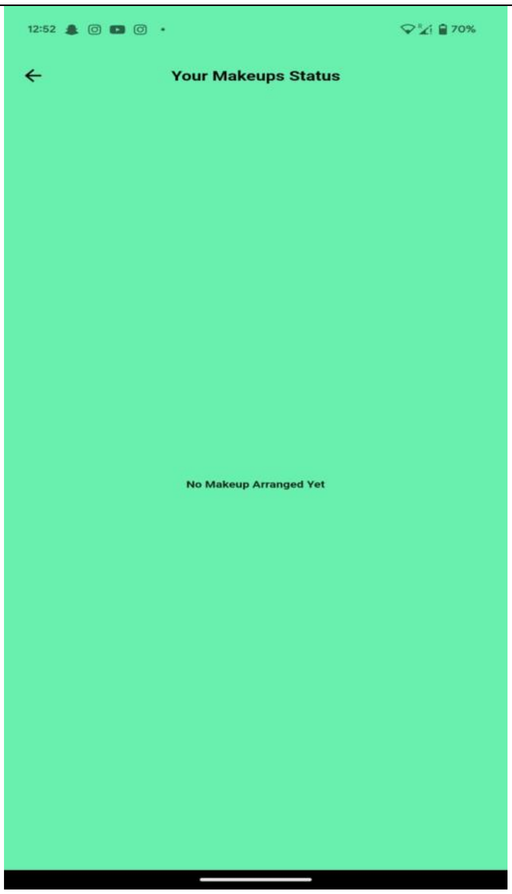
UI Admin	**12**	**13**
Admin display / Select Action	 11:54 74% Arrange Rooms 1 B-101 2 B-102 3 B-112A 4 B-207 5 B-213 lab 6 B-215 7 B-216 8 NB-227 9 NB-228 +	 11:55 74% Add Teachers Name Email Password Add
Visual Cues	On clicking on Room there are the list of room and there is a option 1.add	On clicking on the teacher there is a form 1.add
Tactile Cues	Click on back	Click on add
User input	Nothing	Tacher name, email of teacher , and password
Machine output	This is the display all the room are visible and form and we can add the add by pressing on the add button	This is the form for new teacher and from here admin can create the login for teacher

UI Admin	**14**
Admin display / Add student	 11:55 74% Add Students [redacted] 1 [redacted] 2 [redacted] 3 [redacted] 4 +
Visual Cues	On clicking on add student these are the student login created till now and one button 1.add
Tactile Cues	Click on add
User input	Nothing
Machine output	This is the display all the student are display there and there login and password and we can add the new student and make their sign up

UI Teacher	**15**	**16**
Teacher display / Makeups	 12:28 72% Teacher HUSNA Perform Below Actions : Makeups Request From Admin	 12:29 72% Select Action Book Class Book Makeup class Status of Class Check Makeup class status!
Visual Cues	Now then we login with teacher id then we have the create makeup option 1.makeup	When we create makeup then there are two options 1.book class 2.status of class.
Tactile Cues	Click on makeup	Click on Book Class
User input	Nothing	Nothing
Machine output	This is the teacher login and there is a option for makeup form their we can request for makeup lectures	Form makeup UI there are two button one is book class and other is status of class

UI Teacher	**17**	**18**
Teacher display / Book Makeup room/status of the class	 12:29 72% Book Makeup Class Select Class 1A Select Day Monday FIRST SLOT Search Room SECOND SLOT Search Room THIRD SLOT Search Room FOURTH SLOT Search Room FIFTH SLOT Search Room	 12:29 72% Status Of Your Makeup Class Monday Class 1A FIRST Slot B-101 Room 1
Visual Cues	Now then we book the makeup class the we have to tell the semester and section and the second thing is day And after that select the slot with sortable time	When we create the makeup lecture the this is the status of the class this shoe that ours makeup request have be accepted
Tactile Cues	Click on slot and then room	Click on back
User input	Semester, section and after that select the day and slot	Nothing
Machine output	In this the teacher making the makeup leacture by selecting the semester and than section and after that slot and room	This is the status of makeup lecture and this shoe that our makeup room have be booked by the admin

UI Teacher	**19**	**20**
Student display / Day /Makeup	 12:51 Students BILAL Perform Below Actions : Day TimeTable Makeups Request From Teachers	 12:52 Select Day Monday TimeTable Tuesday TimeTable Wednes TimeTable Thurs TimeTable Friday TimeTable
Visual Cues	Now then login form student id and there are few option visible 1.day 2.makeup	When we open the day option there are days name 1.Monday 2.Tuesday 3.Wednesday 4.Thursday 5.Friday
Tactile Cues	Click on Day	Click on Monday
User input	Nothing	Nothing
Machine output	When we login form student id than this will show we the day wise time table of the lectures	This will show we're the time table of ours subject on day wise

UI Teacher	**21**	**22**
Student display / Day wise timetable /Makeup		
Visual Cues	Now then login form student id and there we view the timetable on the specific day	When we open the makeup lecture then this UI page is open now their is no makeup lecture is arrange
Tactile Cues	Click on back	Click on back
User input	Nothing	Nothing
Machine output	When we login form student id than this will show the time table day wise	This will show we the makeup lectures and now there is no makeup lectures

5:39 58%

Monday TimeTable For CS Department

	Ist Slot	2nd Slot	3rd Slot	4th Slot	5th Slot
1A	free	free	free	free	free
1B	ISL NB-228 abdurrehman	ISL NB-228 abdurrehman	OOP B-213 lab usman	OOP B-213 lab usman	MC NB-227 hasnain
1C	free	free	free	free	free
3A	LA NB-227 yasir	free	OS B-112A ammara	QURAN B-102 abdurrehman	free
3B	LA B-102 zulifiqar	OS B-112A ammara	free	free	QURAN NB-228 abdurrehman
3C	OS B-112A ammara	LA NB-227 yasir	QURAN NB-227 Rizwan	free	free
5A	BRW B-216 farwa	DA&A B-207 abdurrehman	MAD B 215 lab ehtisham	free	free
5B	LA B 215 lab zulifiqar	free	free	free	free
5C	free	free	free	free	free
7A	free	PP B-102 umar	CC B-101 nauman	NC B-112A zafar	free
7B	free	PP B-101 ayesha	TP B-216 abdurrehman	free	NC B-102 zafar
7C	PP B-101 ayesha	free	TP B-102 umar	NC B-216 yasam	free

12:52 70%

Your Makeups Status

No Makeup Arranged Yet

Chapter 5: Software Testing

5.1 Introduction:

This deliverable is based on the IEEE standard of software testing i.e., IEEE SOFTWARE TEST DOCUMENTATION Std 829-1998. This standard describes a set of basic test documents that are associated with the dynamic aspects of software testing (i.e., the execution of procedures and code). The standard defines the purpose, outline, and content of each basic document. While the documents described in the standard focus on dynamic testing, several of them may be applicable to other testing activities (e.g., the test plan and test incident report may be used for design and code reviews). This standard may be applied to commercial, scientific, or military software that runs on any digital computer. Applicability is not restricted by the size, complexity, or criticality of the software. However, the standard does not specify any class of software to which it must be applied. The standard addresses the documentation of both initial development testing and the testing of subsequent software releases. For a particular software release, it may be applied to all phases of testing from module testing through user acceptance. However, since all the basic test documents may not be useful in each test phase, the particular documents to be used in a phase are not specified. Each organization using the standard will need to specify the classes of software to which it applies, and the specific documents required for a particular test phase.

The standard does not call for specific testing methodologies, approaches, techniques, facilities, or tools, and does not specify the documentation of their use. Additional test documentation may be required (e.g., code inspection checklists and reports). The standard also does not imply or impose specific methodologies for documentation control, configuration management, or quality assurance. Additional documentation (e.g., a quality assurance plan) may be needed depending on the methodologies used.

Following are standard artifacts, which must be included in this deliverable:

1. Test Plan
2. Test Design Specification
3. Test Case Specification
4. Test Procedure Specification
5. Test Item Transmittal Report
6. Test Log
7. Test Incident Report
8. Test Summary Report

5.2. Test plan

Certainly, A Test Plan for our Classroom Management System project. A Test Plan outlines the strategy and approach for testing our software to ensure its quality and reliability. Here's a basic outline for our Test Plan:

Test Plan for Classroom Management System

1. I**ntroduction**
   a. Purpose of the Test Plan
   b. Scope of testing (what is included and excluded)
   c. Objectives of testing
   d. Roles and responsibilities of the testing team

2. **Test Objectives**
   b. The specific goals and objectives of testing for our project.
   c. Define what we want to achieve through testing.

1. **Test Items**
   d. All the components and functionalities of our Classroom Management System that will be tested.

2. **Features to be Tested.**
   a. The features or functionalities of our application that will be tested.
   b. Include features related to student access, teacher access, admin approval, and notification systems.

3. **Features not to be Tested.**
   a. Specify any features or functionalities that are explicitly excluded from testing.

4. **Test Environment**
   a. The hardware, software, and network configurations that will be used for testing.
   b. The browsers and devices that will be supported.

5. **Test Data**
   a. The test data and test scenarios that will be used during testing.
   b. Include sample data for makeup lectures, student profiles, and teacher profiles.

6. **Test Cases**
   a. Create detailed test cases for each feature or functionality to be tested.
   b. Include test case ID, description, preconditions, steps, expected results, and actual results columns.

7. **Test Execution Schedule**
    a. Create a timeline for test execution, specifying when each test case will be executed.
    b. Allocate resources and personnel accordingly.

8. **Test Deliverables**
    a. List the documents and reports that will be produced as a result of testing.
    b. This may include defect reports, test summary reports, etc.

9. **Defect Management**
    a. The process for reporting, tracking, and prioritizing defects.
    b. Specify who will be responsible for defect resolution.

10. **Risks and Mitigations**
    a. Identify potential risks related to testing and how they will be mitigated.
    b. Include risks associated with software, resources, and timelines.

11. **Sign-off Criteria**
    a. The criteria that must be met for the testing phase to be considered complete.
    b. Specify who will provide the sign-off for testing completion.

12. **Approvals**
    a. List the stakeholders who need to approve this Test Plan.

13. **Appendix**
    a. Include any additional information, templates, or references that may be relevant to testing.

5.3. Test design specification

Test Design Specification for Classroom Management System

1. Introduction

The Classroom Management System (CMS) is an application designed for managing classroom schedules, including makeup lectures. This document outlines the test design specification for the CMS, focusing on various aspects of software testing. The CMS has three main user roles: students, teachers, and admin. This specification will provide a comprehensive plan for testing the functionality of the system.

2. Objectives of Testing

The primary objectives of testing the CMS are as follows:
a) To ensure that the application meets the requirements and functionalities as described in the project documentation.
b) To identify and rectify any defects, bugs, or issues in the application.
c) To Athe system is user-friendly and intuitive for all three user roles (students, teachers, and admin).
d) To validate that the makeup lecture scheduling process is accurate and efficient.

3. Test Design

3.1 Test Levels

The testing process will involve the following levels:
a) Unit Testing: Testing individual components and functions.
b) Integration Testing: Testing the interaction between various modules.
c) System Testing: Testing the complete application.
d) User Acceptance Testing (UAT): Testing by end-users to ensure it meets their requirements.

3.2 Test Types

The following test types will be employed during the testing process:
a) Functional Testing: Ensuring that all functions and features work as intended.
b) Usability Testing: Assessing the user-friendliness of the system.
c) Security Testing: Evaluating the system's security measures.
d) Performance Testing: Assessing system responsiveness and stability.
e) Regression Testing: Ensuring that new changes do not affect existing functionality.
f) User Acceptance Testing: Validating that the system meets user expectations.

3.3 Test Environments

Testing will be conducted in multiple environments:
a) Development Environment: Testing during the development phase.
b) Staging Environment: Testing on a replica of the production environment.
c) Production Environment: Live testing with real users after deployment.

3.4 Test Data

Test data will be generated to simulate different scenarios, including student, teacher, and admin roles. This data will cover various aspects such as classroom schedules, makeup lectures, and notifications.

4. Test Cases

4.1 Functional Test Cases

Functional test cases will be designed to verify the correctness of various features within the CRMS:

a) **Student Login:**
- The students can log in with valid credentials.
- The students cannot access admin or teacher functionalities.

b) **Teacher Login:**
- Ensure teachers can log in with valid credentials.
- The teachers can schedule makeup lectures.

c) **Admin Login**:
- The admin can log in with valid credentials.
- The admin can approve makeup lecture requests.

d) **Makeup Lecture Scheduling:**
- The accuracy of makeup lecture scheduling.
- The makeup lectures do not conflict with existing schedules.

e) **Notification System:**
- Confirm that students receive notifications for upcoming makeup lectures.
- Ensure notifications are timely and accurate.

4.2 Usability Test Cases

Usability test cases will evaluate the user-friendliness of the CMS:
a) **User Interface:**
- Assess the overall lawet and design for user-friendliness.
- The ease of navigation for students, teachers, and admin.

b) **User Guidance:**
- Check for helpful tooltips and guidance throughout the application.

4.3 Security Test Cases

Security test cases will focus on ensuring the CMS's protection against potential threats:
a) **Authentication:**
- The only authorized users can access the system.

b) **Data Protection:**
- The sensitive user data is securely stored and transmitted.

4.4 Performance Test Cases

Performance test cases will assess the CMS's responsiveness and scalability:
a) **Load Testing:**
- Evaluate system performance under various levels of user load.

b) **Stress Testing:**
- Determine the system's stability under extreme conditions.

5.4. Test Case Specification

Test Case Specification for Classroom Management System

2. **Test Environment:**

- Specify the hardware and software requirements for testing.

- Mention any specific configurations needed for the test environment.

3. **Test Case Design:**

- List the test cases for each major functionality of the application.

3.1. **User Authentication:**

- TC001: A valid teacher can log in successfully.

- TC002: A valid admin can log in successfully.

- TC003: A invalid user cannot log in.

3.2. **Makeup Lecture Creation:**

- TC004: A teacher can create a makeup lecture.

- TC005: Verify tha a teacher cannot create a makeup lecture without required details.

- TC006: An admin can approve makeup lectures.

3.3. **Time Table Viewing:**

- TC007: A students can view the classroom timetable.

- TC008: A students receive notifications about upcoming makeup lectures.

3.4. **User Roles:**

- TC009: A Teachers can only access makeup lecture creation.

- TC010: A admins can approve makeup lectures.

- TC011: A students have read-only access to the timetable.

4. **Test Data:**

- Define the test data required for each test case, including valid and invalid inputs.

5. **Test Execution:**

- Specify the steps to execute each test case.
- Include any preconditions or setups required.

6. **Expected Results:**

- The expected outcomes for each test case.
- Identify pass/fail criteria.

7. **Test Reporting:**

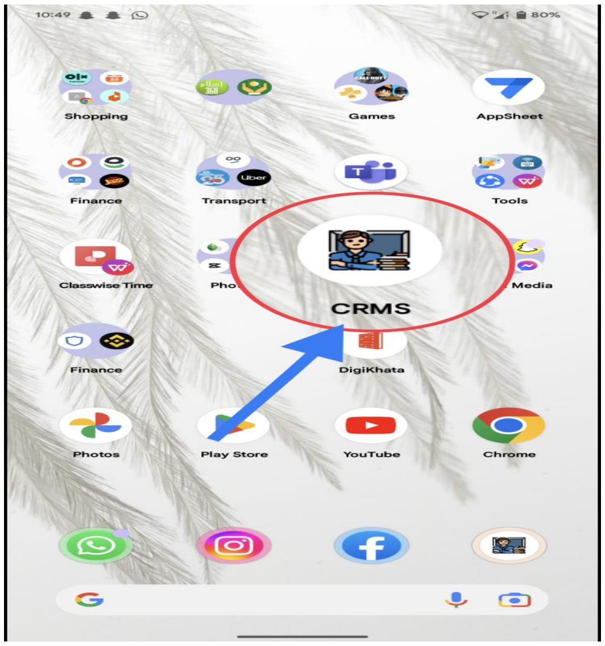
- Explain how test results will be documented and reported.
- Include a format for recording test outcomes.

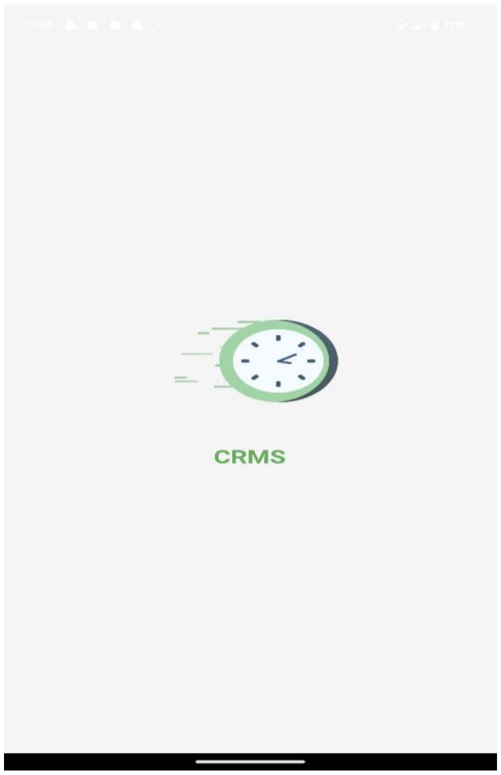
8. **Defect Tracking:**

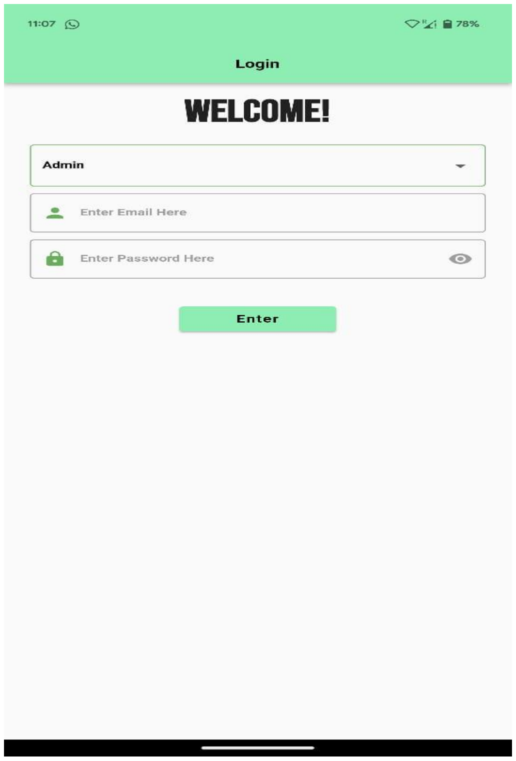
- The process for tracking and managing defects found during testing.
- Include a defect reporting template.

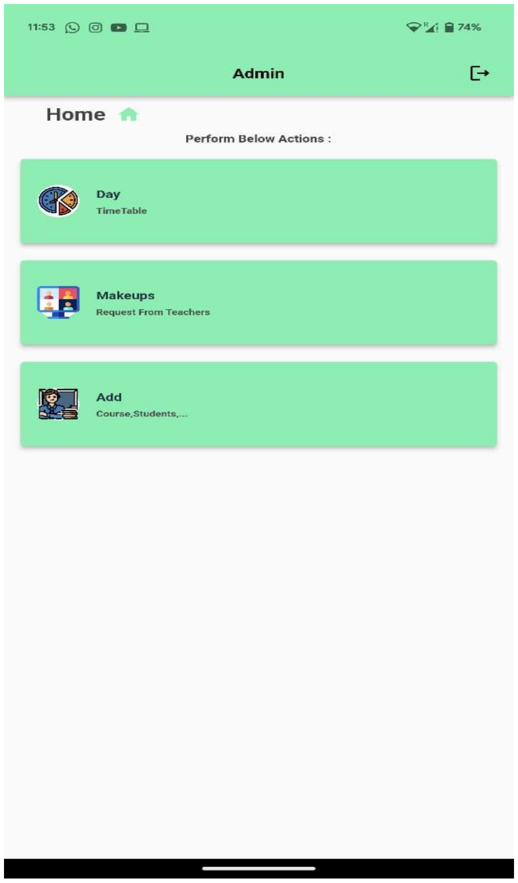
5.5. Test procedure specification

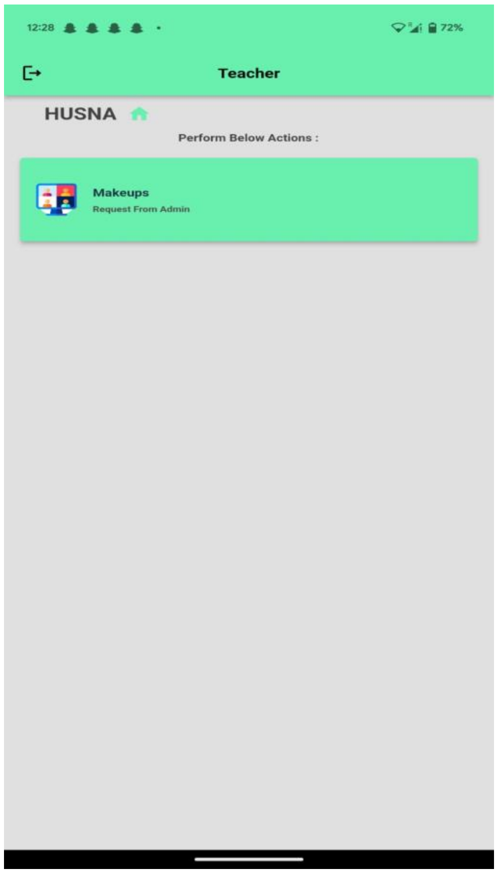
5.5.2.4.2. Set up:




CRMS


11:07
78%
Login
WELCOME!
Admin
Enter Email Here
Enter Password Here
Enter


11:53
74%
Admin
Home
Perform Below Actions :
Day
TimeTable
Makeups
Request From Teachers
Add
Course,Students,...


12:28
72%
Teacher
HUSNA
Perform Below Actions :
Makeups
Request From Admin

5.9. Test summary report

Summary Report for the Classroom Management System project:

Test Summary Report

1. **Introduction**
   - Briefly explain the purpose of the Test Summary Report.
   - Provide an overview of the Classroom Management System project.

2. **Test Objectives**
   - Clearly define the objectives of the software testing phase.
   - Describe what we aimed to achieve through testing.

3. **Scope of Testing**
   - Detail the components and functionalities of the Classroom Management System that were tested.
   - Specify what aspects were not covered in the testing phase.

4. **Testing Approach**
   - Explain the testing methodologies and techniques used.
   - Discuss the test levels (unit, integration, system, and acceptance) applied.
   - Mention any automation tools used for testing.

5. **Test Environment**
   - Describe the hardware and software configurations used for testing.
   - Specify any special testing tools or platforms utilized.

6. **Test Cases and Scenarios**
   - Provide an overview of the test cases and scenarios created.
   - Include details on the number of test cases, their purpose, and coverage.

7. **Test Execution**
   - Discuss the actual execution of test cases.
   - Mention any issues encountered during testing.
   - Provide information on the pass/fail status of test cases.

8. **Defect Tracking**
   - Explain the process of defect tracking and management.
   - Provide statistics on the number of defects found, their severity, and status.

9. **Test Results**
   - Present the overall test results, including test coverage and pass/fail ratios.
   - Use tables and graphs to illustrate the data effectively.

10. **Risk Assessment**
- Identify any risks associated with the testing phase.
- Describe how these risks were managed or mitigated.

11. **Lessons Learned**
- Reflect on the challenges faced during testing.
- Share any valuable insights or lessons learned from the testing process.

12. **Recommendations**
- Suggest any improvements or changes based on the testing experience.
- Provide recommendations for future testing efforts.

13. **Conclusion**
- Summarize the key findings and outcomes of the testing phase.
- Reiterate the importance of testing in ensuring software quality.

14. **Appendices**
- Include any additional documents or artifacts related to testing (e.g., test plans, test scripts).

15. **References**
- Cite any references or resources used during the testing process.

16. **Acknowledgments**
- Thank individuals or teams involved in the testing phase.

17. **Author Information**
- Provide information about the author of the Test Summary Report.