



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PASHCHIMANCHAL CAMPUS

SCHOOL MANAGEMENT SYTEM

Subject Code: EX 755

BY:

Apil Chand [PAS075BEI006]

Nitesh Kumar Chaurasia [PAS075BEI025]

Pradip Singh Saud [PAS075BEI027]

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE BACHELOR'S DEGREE IN ELECTRONICS,
COMMUNICATION & INFORMATION ENGINEERING

May, 2023



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PASHCHIMANCHAL CAMPUS
LAMACHOUR, POKHARA

[Subject Code: EX 755]

A MAJOR PROJECT REPORT ON
“SCHOOL MANAGEMENT SYSTEM”

SUBMITTED BY:

Apil Chand [PAS075BEI006]

Nitesh Kumar Chaurasia [PAS075BEI025]

Pradip Singh Saud [PAS075BEI027]

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

Pashchimanchal Campus

May, 2023

COPYRIGHT

The author has agreed that the library, Pashchimanchal Campus, may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the lecturers, who supervised the project works recorded herein or, in their absence, by the Head of Department wherein the project report was done. It is understood that the recognition will be given to the author of the report and to the Department of Computer and Electronics, Pashchimanchal Campus in any use of the material of this project report. Copying or publication or other use of this report for financial gain without approval of the Department and author's written permission is prohibited. Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head of Department

Department of Electronics and Computer Engineering

Pashchimanchal Campus, Institute of Engineering

Lamachaur, Pokhara-16

Nepal

ACKNOWLEDGEMENT

It gives us immense pleasure to express our deep sense of gratitude to our supervisor Asst. Prof. Laxmi Prasad Bastola for his invaluable guidance, motivation, constant inspiration and above all for his ever co-operating attitude that enabled us in bringing up this project in the present form. We solely take the responsibility of any possible mistakes that may have occurred in preparing this report and we would like to welcome comments and queries during the submission of this report.

Finally, we would like to thank all the staffs of Institute of Engineering Pashchimanchal Campus who helped us throughout our project.

ABSTRACT

A school management system developed using the Flutter framework and integrated with Firebase for data storage and authentication. The system addresses the challenges faced by educational institutions in managing student information, attendance, grades, and communication with parents. Traditional methods of managing school-related tasks are often cumbersome, time-consuming, and error-prone. Thus, there is a need for an efficient and user-friendly solution that can streamline these processes and enhance productivity.

The development of the school management system involved utilizing the Flutter framework, a cross-platform development tool, to create a mobile application as well as web application. Firebase, a comprehensive cloud-based platform, was integrated to handle data storage, authentication, and communication. The system focused on meeting the needs of school administrators, teachers, students, and parents. Key features included student enrollment, attendance tracking, grade management, and notice announcement functionalities. The implementation of this system resulted in significant benefits, such as easier student registration, efficient attendance tracking, streamlined grade management, improved communication, and enhanced engagement among administrators, teachers, students, and parents.

Keywords: *Flutter, Cross-platform, Firebase, authentication*

TABLE OF CONTENTS

COPYRIGHT.....	iii
ACKNOWLEDGEMENT	iv
ABSTRACT.....	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF ABBREVIATIONS.....	x
CHAPTER 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Problem Statement.....	1
1.3 Objectives	2
1.4 Scope of Project.....	2
1.5 Feasibility Analysis	3
1.5.1 Technical Feasibility.....	3
1.5.2 Operational Feasibility.....	5
1.5.3 Economical Feasibility	5
1.6 System Requirement.....	6
1.6.1 Software Requirement	6
1.6.2 Hardware Requirement.....	6
1.7 Organization of Report	6
CHAPTER 2: LITERATURE REVIEW	7
CHAPTER 3: METHODOLOGY	10
3.1 System Development Life Cycle.....	10
3.2 System Requirements	10
3.3 System Design	11
3.4 Technology Overview	12

3.4.1	Frontend	12
3.4.2	Backend	12
3.5	Flowchart.....	13
3.6	Analysis Model.....	15
3.6.1	Use Case Diagram	15
3.6.2	Sequence Diagram	16
3.6.3	Activity Diagram	18
3.6.4	Component Diagram.....	19
3.6.5	Data Flow Diagram.....	20
3.9	Software Testing	24
3.9.1	QA Testing Tools:	26
3.9.2	Test Cases	27
CHAPTER 4: FEATURE AND FUNCTIONALITIES		35
4.1	Features and Functionalities	35
4.2	User Manual	37
4.2.1	Login.....	37
4.2.2	User Entry	37
4.2.3	User Panel	38
4.2.4	Class.....	40
4.2.5	Grades and Notice.....	41
CHAPTER 5: EPILOGUE.....		42
5.1	Result and Discussion.....	42
5.2	Future Enhancements	42
5.3	Conclusions	44
REFERENCES		46
APPENDIX-A		48
Appendix-B.....		58

LIST OF FIGURES

Figure 3.1:Agile Development Model	10
Figure 3.2: General Workflow of System.....	14
Figure 3.3:Use Case Diagram of System.....	16
Figure 3.4: Sequence Diagram of System	17
Figure 3.5: Activity Diagram of System.....	18
Figure 3.6: Login Activity of System	19
Figure 3.7:Component Diagram of System	20
Figure 3.8: DFD Level 0 Diagram.....	20
Figure 3.9: DFD Level 1 Diagram.....	21
Figure 3.10: DFD Level 2 for Student.....	22
Figure 3.11: DFD Level 2 for Admin	23
Figure 3.12:DFD Level 2 for Student.....	24
Figure 4.1: Login Process	37
Figure 4.2:User Permission.....	38
Figure 4.3: User panel.....	39
Figure 4.4:Class of System	40
Figure 4.5: Notice and Grades of School.....	41

LIST OF TABLES

Table 3.1: Test cases	33
Table 3.2: Test cases analysis and result	34

LIST OF ABBREVIATIONS

AI - Artificial Intelligence

API - Application Programming Interface

CBD - Component-Based Development

CSS - Cascading Style Sheets

GB - Gigabyte

GH - GitHub

HTML - Hypertext Markup Language

ICT - Information and Communication Technology

IDE - Integrated Development Environment

MIS - Management Information System

OS - Operating System

RAM - Random Access Memory

REST - Representational State Transfer

SDK - Software Development Kit

SIS - Student Information System

SOA - Service-Oriented Architecture

UI - User Interface

UML - Unified Modeling Language

USB - Universal Serial Bus

VS Code - Visual Studio Code

CHAPTER 1: INTRODUCTION

1.1 Background

The speed of technology is transforming the way that we work, live and learn. The need for the adaptation of every sector with the development of technology and digitization becomes necessary. In this modern era, where automation is in almost every sector of development, the education sector, which is the most important part of any country, has been unable to cope with the pace of development. The educational sector is still far behind in the case of adapting digitalization in every aspect of its interest. The effect or loss of lacking digitalization in the educational sector or can say in educational institutions has been exposed during the pandemic time. The whole educational system collapsed during that time and it pushed our educational calendar 6 to 7 months back which directly affects each and every sector of the country's development. The motion of digitalization has started already but still the use of paper results in less productivity. The use of paper leads to less productivity, unnecessary time-consuming work, less efficiency, difficulty in managing, and lots of other negative aspects which finally drag us backward from the path of optimality. The proper analysis is also required to achieve optimality in every sector. For the development of students and also for the development of educational institutions the proper analysis of data is required. Tracking the performance of students is hard and inefficient without digitization and the same in the case of the whole education institution. All these problems need to be solved by developing and utilizing the digital environment in the educational sector to achieve the result everyone needs. The full digitalization of the country can only achieve by introducing the school management system in every institution.

1.2 Problem Statement

Inefficient and outdated school management systems pose significant challenges in effectively managing educational institutions. Manual and paper-based processes consume valuable time and resources, leading to inefficiencies and errors in tasks such as student enrollment, attendance tracking, grade management, and communication between

stakeholders. The lack of integration with modern information and communication technology (ICT) tools further exacerbates these issues, hindering the delivery of quality education. As a result, there is an urgent need to develop and implement a comprehensive school management system that integrates ICT solutions, streamlines administrative processes, and enhances communication and collaboration among teachers, administrators, students, and parents. This project aims to address these challenges by designing and implementing an efficient and user-friendly school management system that improves administrative efficiency, enhances data management, and promotes effective communication and engagement within educational institutions.

1.3 Objectives

1. Improve Efficiency:

The objective of the school management system is to enhance the overall efficiency of administrative processes. This includes streamlining tasks such as student enrollment, attendance tracking, grading, and generating reports.

2. Enhance Communication and Collaboration:

Another objective of the school management system is to improve communication and collaboration among various stakeholders, including teachers, students, parents, and school administrators. The system should provide a centralized platform for sharing important information, such as announcements, schedules, assignments, and grades, ensuring that everyone is on the same page.

1.4 Scope of Project

The scope of the School Management System project includes the development, implementation, and deployment of a comprehensive software solution to support the administrative and academic processes of the school.

1.5 Feasibility Analysis

After studying and analyzing the required functionalities of the systems, the next task is to do a feasibility study for the project. It is said that “All Projects are feasible given unlimited resources and times”. However, both resources and times are limited in reality. The project should adhere to the time and make efficient use of the available resources. The proposed solution should satisfy all the user requirements and should be flexible enough to allow for future changes based on new requirements.

The following areas are covered by the feasibility study: -

1. Technical Feasibility
2. Operational Feasibility
3. Economical Feasibility

1.5.1 Technical Feasibility

The technical feasibility of the School Management System project involves assessing whether the proposed solution can be effectively developed, implemented, and maintained within the available technological resources and constraints. The following factors contribute to the technical feasibility of the project:

1. Infrastructure: The availability of a suitable infrastructure, including hardware and network resources, is essential for the successful implementation of the system. This includes servers, databases, networking equipment, and devices (computers, laptops, tablets, etc.) for users to access the system.
2. Software Development: The project requires expertise in software development, including programming languages, frameworks, and tools. The development team should have the necessary skills and experience to build a robust and scalable system that meets the school's requirements.
3. Integration Capability: The School Management System may need to integrate with existing systems or third-party applications, such as accounting software or learning management systems. Assessing the compatibility and integration capabilities of these systems is crucial to ensure seamless data flow and functionality.

4. Data Management: The system will handle a significant amount of data, including student records, staff information, timetables, and financial data. Adequate database management systems and data security measures must be in place to handle data storage, retrieval, and protection.

5. Scalability: The system should be designed to accommodate future growth and expansion. It should be scalable to handle increasing data volumes, user loads, and additional features that may be required as the school evolves.

6. User Interface and User Experience: The system should have an intuitive and user-friendly interface that is easy to navigate and understand. User experience design principles should be considered to ensure efficient user interactions and minimize user training requirements.

7. Technical Support and Maintenance: A technical support team should be in place to provide ongoing maintenance, updates, and troubleshooting for the system. Regular backups, system monitoring, and bug fixes should be part of the technical support strategy.

8. Security: Data security and privacy are critical considerations for a School Management System. Robust security measures, such as encryption, access controls, and regular security audits, should be implemented to protect sensitive data from unauthorized access or breaches.

9. Compatibility: The system should be compatible with various devices and platforms, including desktops, laptops, tablets, and mobile devices. It should support popular web browsers to ensure accessibility for all users.

By evaluating these technical aspects, it can be determined whether the School Management System project is technically feasible within the available resources, infrastructure, and expertise. Any technical challenges or limitations can be addressed through proper planning, resource allocation, and technology selection to ensure the successful development and implementation of the system.

1.5.2 Operational Feasibility

The operational feasibility of the School Management System in the college project involves assessing whether the proposed system can be effectively integrated into the existing operational environment. Factors such as user acceptance, process alignment, training and support, resource allocation, change management, and scalability are considered to ensure the smooth implementation and utilization of the system.

1.5.3 Economical Feasibility

The economic feasibility of the School Management System project is evaluated by considering the following points:

1. **Cost Analysis:** A detailed assessment of the costs associated with developing, implementing, and maintaining the system, including hardware, software, training, and ongoing support.
2. **Cost Savings:** Identifying potential cost savings through automation of administrative tasks, improved resource allocation, and reduced manual efforts.
3. **Return on Investment (ROI):** Analyzing the expected financial benefits, both tangible and intangible, against the initial investment to determine the ROI of the project.
4. **Scalability and Future Growth:** Assessing the system's ability to accommodate future growth and increased student enrollment without incurring significant additional costs.
5. **Cost-Benefit Analysis:** Conducting a comprehensive analysis to compare the projected benefits of the system with the associated costs to determine its overall feasibility.
6. **Long-Term Financial Viability:** Considering the sustainability of funding for system maintenance, upgrades, and future enhancements to ensure its continued effectiveness over time.

By evaluating these economic factors, the feasibility of the School Management System project from an economic standpoint can be determined.

1.6 System Requirement

System Requirements are essential for the system to work efficiently.

1.6.1 Software Requirement

Software Requirements of the projects are as follows:

1. Operating System: Windows 7 SP1 or higher, Linux, or Mobile Phones
2. Browser: Any web browser like chrome, Microsoft edge , opera mini etc.
3. Flutter SDK, IDE, VS Code, Android Studio, Firebase SDK

1.6.2 Hardware Requirement

Hardware Requirements of the projects are: -

1. Processor: Multicore processor with a clock speed at least 2.3 GHz
2. Memory: Minimum of 8GB of RAM
3. Internet Access
4. Android/iOS Device
5. Storage
6. Graphic Card: Used to improve the performance of the Android Emulator but not a strict requirement
7. USB Cable: Connect your Android device to your computer

1.7 Organization of Report

The paper is organized into several sections. The introduction provides an overview of the School Management System project and its objectives. The literature review examines existing research and studies on school management systems, identifying gaps and limitations. The methodology section describes the research methodology and data collection methods employed in the project. The features, functionalities, and user manual section details the system's capabilities and provides instructions on its usage. The epilogue summarizes the project's outcomes, reflects on the implementation process, and discusses future developments. The references section lists all cited sources, and the appendix includes supplementary materials such as screenshots and technical documentation.

CHAPTER 2: LITERATURE REVIEW

Visscher and at all, bring together a series of studies from a range of countries that explore current features of computerized school information systems, their implementation in a range of schools, the outcomes of this implementation, and implications for the future in terms of further research. This text offers perhaps the widest view of ICT and school management from the perspective of MIS. However, it is clear from this literature that most concern currently is being focused on data entry and collation, rather than upon data transfer or analysis (and particularly so at the teacher level) [1].

The use of technology to update the academic facilities has a significant impact on every responsible stakeholders achievements of a particular institution. Therefore, the most important aspect of developing and implementing efficient information systems for any academic institution is to start with the genuine needs of administration, teachers, and students [2].

A study of Durnali, 2013, when comparing the data collection, processing, storage, accuracy and analysis, and dissemination of student data before and after an e-class management was implemented in their academic institutions, it has been shown through his studies that there are improvements in terms of data collection, processing, storage, accuracy, and analysis and dissemination of student data. If technology is employed for class management and development, it has an impact on how society reflects the socioeconomic, cultural, and technical change [3,4].

Research on creative information management in Taiwan illustrated the value of the class management method. Innovative information management makes students more inspired, enhances their learning effectively, and increases the sense of classes and academic institutions being defined [5].

A study in Turkey shows how teachers and the principal believe that the e-class management system is adequate in terms of administrative relations, student affairs, and student report card work time. A developed web-based information system for class management provides leverage for academic institutions that need the application to

facilitate learning, teaching, and administration quality and effectiveness. It is always necessary to have a modern class management and information system[6,7].

It is necessary to keep academic records and manage them properly because it is an important aspect of the institution's leverage in terms of keeping things in their rightful place to ensure quality processing and record-keeping. This will also help institution managers in their decision-making process and to also enhance the implementation of usable records in academic institutions that will lead to cost savings, transparency, easy accessibility, accountability, and retrieval of required information from their storage. The key areas that need to be present in an e-class information and management system are open standards, interoperability, transition, accessibility, cost efficiency, statutory-based innovations, and usability[8,9].

Digital Nepal, initiated in 2014, started as a campaign and has evolved into a transformative movement in the education sector of Nepal. It was triggered by the inefficiencies of the traditional school management system, which consumed significant time and human resources. Recognizing the need to bridge the gap between traditional and digital systems and to ensure quality education, Digital Nepal aimed to integrate information and communication technology (ICT) into the education sector. Within a short period, they successfully digitalized over 700 schools and colleges across the country, establishing credibility and trust among educational institutions. With their head office in Kathmandu and branches in Janakpur and Itahari, Digital Nepal's influence can be seen throughout the nation. Concurrently, Ingrails, founded in 2014, focused on delivering high-quality digital products and customer service[10].

Veda, was introduced to address the inefficiencies and lack of technology adoption in schools. Starting with St. Xavier's School in 2016, Veda expanded outside Kathmandu and reached 100 schools by 2018. By 2020, Veda became the most downloaded educational app in Nepal and continued to grow, reaching 500 schools in multiple cities. Currently, Veda is incorporating AI-based learning and has plans to expand internationally, targeting over 1200 schools. In 2022, Veda successfully implemented its school management software in its first international location, Brunei. These initiatives highlight the growing importance of digitalization and technology integration in the education sector, empowering schools and improving the quality of education in Nepal and beyond[11].

Information systems are created to provide solutions and feedback to encourage the effectiveness of learning, teaching, and administrative purposes. There are various information conveying systems and School management systems that are used by different educational institutions like PowerSchool SIS, Skyward Student Management Suite, and so on. Each system has its features, advantages, and disadvantages. These systems are considered for providing the necessary information, announcements of the particular institutions to their respective stakeholders but are failing to ensure the reach and engagement of the conveyed information and other managerial tasks for the institution.

Our School Management System(SMS) will be a better solution for providing effective communication and engagement among administration, teachers, and students, track academic progress, efficient staff and student management, organized learning materials, and proper record-keeping process.

CHAPTER 3: METHODOLOGY

3.1 System Development Life Cycle

The Agile model for a school management system involves defining the project vision and goals, creating a product backlog, and planning short development iterations known as sprints. During each sprint, the development team implements selected items from the backlog, collaborates with stakeholders for feedback, and conducts daily stand-up meetings to address challenges. Continuous integration and testing ensure system stability, while sprint review and retrospective meetings gather feedback and improve processes. The iterative development approach allows for ongoing refinement and expansion of the system based on changing requirements. During each iteration it goes through different steps which are plan, design, develop, test, deploy and review. Finally, the system is launched[12].

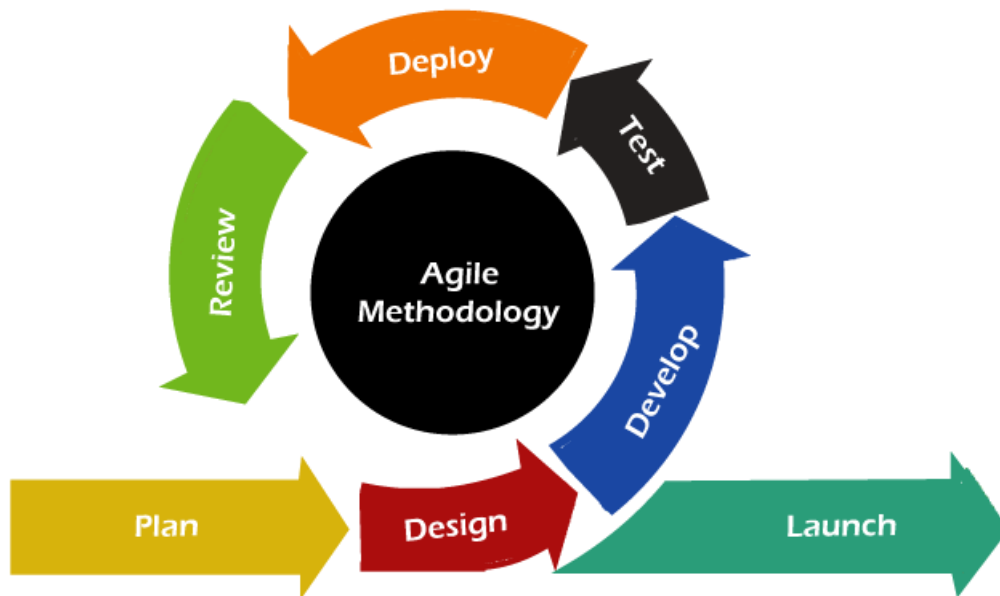


Figure 3.1: Agile Development Model[12]

3.2 System Requirements

The school management system needs to fulfill various requirements to ensure its effectiveness. From a hardware perspective, it should be compatible with different devices

like desktops, laptops, tablets, and smartphones, with specific hardware requirements depending on the device being used. In terms of software, it should be compatible with multiple operating systems such as Windows, macOS, iOS, and Android, with software requirements tailored to each operating system. The system must have network connectivity to access the Google Firebase backend, which depends on the specific network infrastructure in use. Security is paramount, requiring protection for sensitive data like student information and financial data. Firebase Authentication can manage user authentication and authorization, while Firebase Cloud Firestore provides secure data storage and retrieval. Performance requirements demand a fast and responsive system with minimal latency. Firebase Cloud Firestore ensures quick data operations, and Firebase Hosting enables rapid frontend deployment. Usability is vital, necessitating an intuitive and user-friendly interface achievable through Flutter and Dart. Finally, the system must be scalable, capable of accommodating a large number of users and data while offering the flexibility to scale up or down as needed. Firebase Cloud Firestore supports scalable data storage, while Firebase Functions implement serverless functionality that automatically scales.

3.3 System Design

The system's actual construction is the focus of the design phase. The network's design or configuration (hardware, operating system, programming, etc.) is included. plan of UIs(structures, reports, and so forth.), security concerns and the design of system interfaces (for communicating with other systems). Performance testing of the proposed design is critical to ensuring that it satisfies the requirements established during the analysis phase. To put it another way, the primary goal of this phase is to turn the previously established requirements into a comprehensive and in-depth set of specifications that will be used in the subsequent phase. During the design phase, there are a few things that need to be done:

1. Design the application
2. Design and coordinate the organization
3. Design and coordinate the information base
4. Make an alternate course of action
5. Begin a Support, Preparing and Tasks plan

6. Audit the plan
7. Articulate the business cycles and strategies
8. Establish a progress system
9. Convey the Framework Configuration Report
10. Audit last plan

An information base framework is basically just an electronic record keeping framework the data set itself can be viewed as sort of electronic file organizer. A database is a single collection of persistent data that is utilized by an institution's application system. In this context, the term "instituted" is merely a convenient generic term for any reasonable, independent science, technical, or other institution.

3.4 Technology Overview

3.4.1 Frontend

1. Flutter: Flutter is an open-source mobile application development framework created by Google. It allows developers to build high-performance, cross-platform mobile applications for iOS and Android using a single codebase. Flutter uses the Dart programming language, which is easy to learn and offers features such as hot reload, which allows developers to see changes in real-time..

3.4.2 Backend

1. Firebase: Firebase is a mobile and web application development platform that provides a range of services, including authentication, real-time database, cloud storage, and hosting. Firebase offers a scalable and secure backend infrastructure for mobile and web applications, allowing developers to focus on building the frontend of their applications.

2. Firebase Authentication: Firebase Authentication provides a simple way to add user authentication to mobile and web applications. It supports a range of authentication methods, including email and password, Google Sign-In, Facebook Login, and more.

3. Firebase Cloud Firestore: Firebase Cloud Firestore is a real-time NoSQL database that allows developers to store and sync data in real-time. It provides a flexible data model and

supports offline data access, making it easy to build real-time applications such as chat applications or collaborative tools.

4. **Firebase Cloud Functions:** Firebase Cloud Functions allows developers to run serverless functions in response to events triggered by Firebase services or HTTP requests. This allows developers to add custom logic to their applications without having to manage servers or infrastructure.

5. **Firebase Cloud Storage:** Firebase Cloud Storage provides a simple way to store and serve user-generated content, such as images, videos, and audio files. It supports secure uploads and downloads, and integrates with other Firebase services such as Firebase Authentication and Firebase Cloud Functions.

Overall, using Flutter and Firebase together provides a powerful and efficient platform for mobile application development, allowing developers to build high-performance, scalable, and secure mobile applications with ease. By leveraging the frontend capabilities of Flutter and the backend infrastructure of Firebase, developers can focus on building great user experiences without having to worry about managing servers or infrastructure.

3.5 Flowchart

The flowchart explains how the project work in real time and also be defined as graphical representation of a problem. When system is browse in website, it enter into new interface, then find it is new user or not. If it is new user or existing user and forget password then contact with admin else got to login page. In login page , validate user can enter into new interface else go to login page by showing notification error. After validation it is mandatory to know that validate user is admin, student, parent, teacher. After identification of user , each user has it own panel and permissions . User uses its feature and accomplish the work .

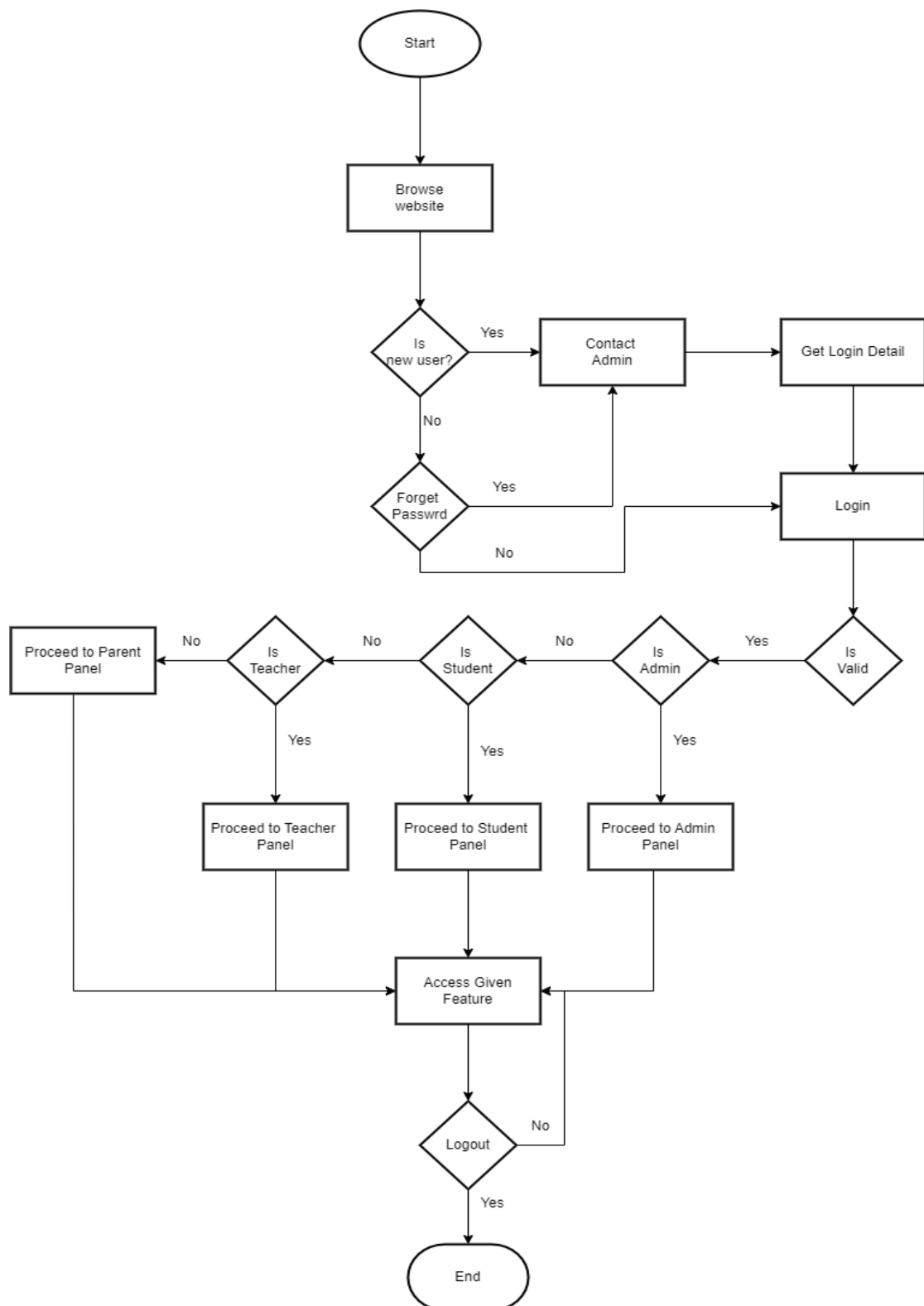


Figure 3.2: General Workflow of System

3.6 Analysis Model

To produce a model of the system which is correct, complete and consistent we need to construct the analysis model which focuses on structuring and formalizing the requirements of the system. Analysis model contains three models: functional, object and dynamic models. The functional model can be described by use case diagrams. Dynamic model can also be described in terms of sequence, activity diagrams. For the purpose of this project we have described the analysis model in terms of the functional model and dynamic models using use case and sequence diagrams.

3.6.1 Use Case Diagram

Use case diagrams provide a high-level overview of the system's functionalities and the actors involved. They help in understanding the system's requirements, identifying user interactions, and defining the scope of the system. Here are the commonly used symbols and their names in a use case diagram:

1. Actors: An actor represents a user, external system, or entity that interacts with the system. Actors are depicted as stick figures.
2. Use Cases: A use case represents a specific functionality or behavior of the system from the perspective of an actor. Use cases are depicted as ovals or ellipses.
3. Relationships: Relationships between actors and use cases are depicted using lines and arrows.

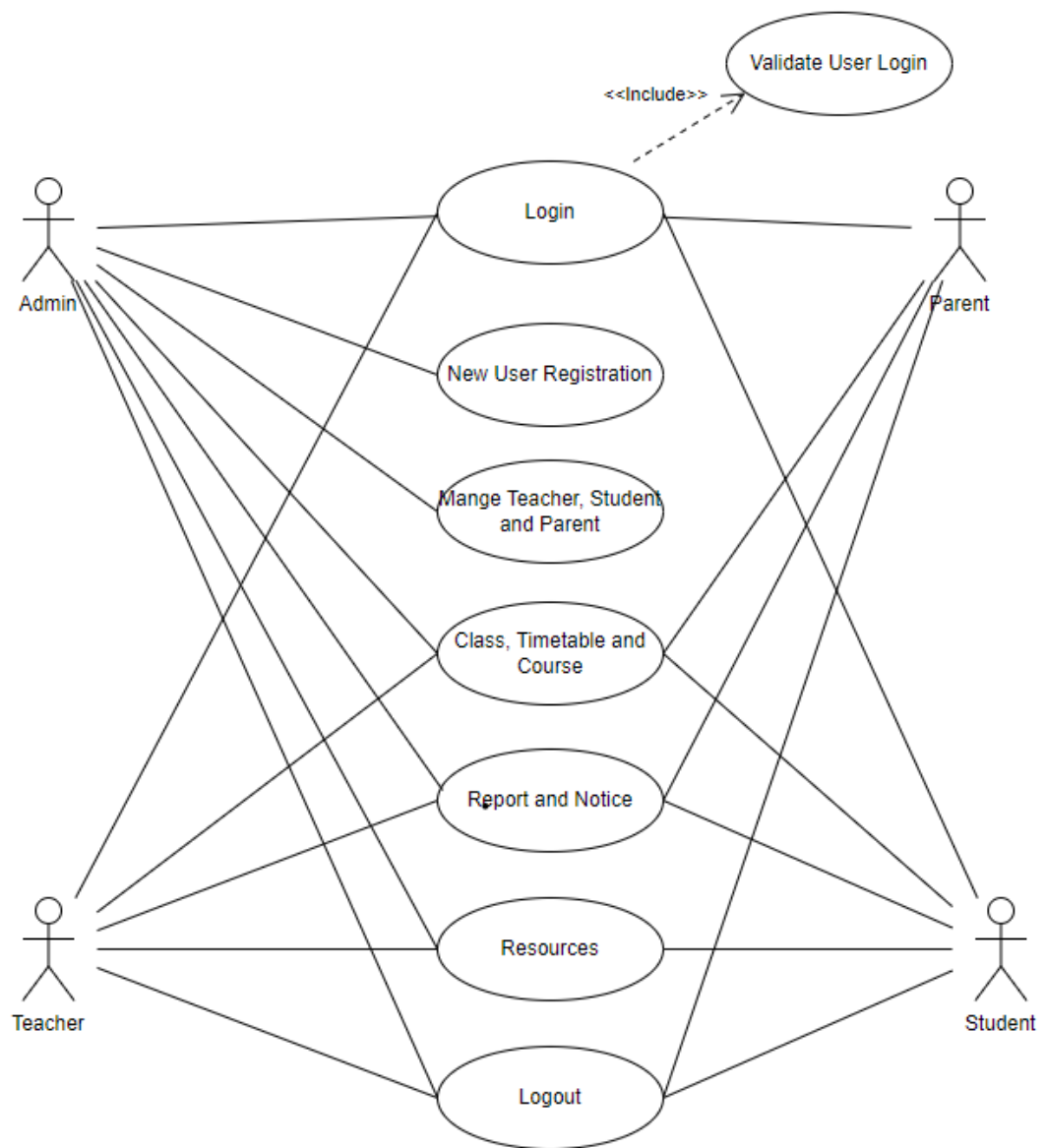


Figure 3.3: Use Case Diagram of System

3.6.2 Sequence Diagram

The designed sequence diagram illustrates the series of events that occurs in School Management System. In this illustration, the actors are represented by a stick man and the transactions or classes are represented by objects. It will give you clear explanation about

the behavior of a School Management System in terms of processing the flow of instructions.

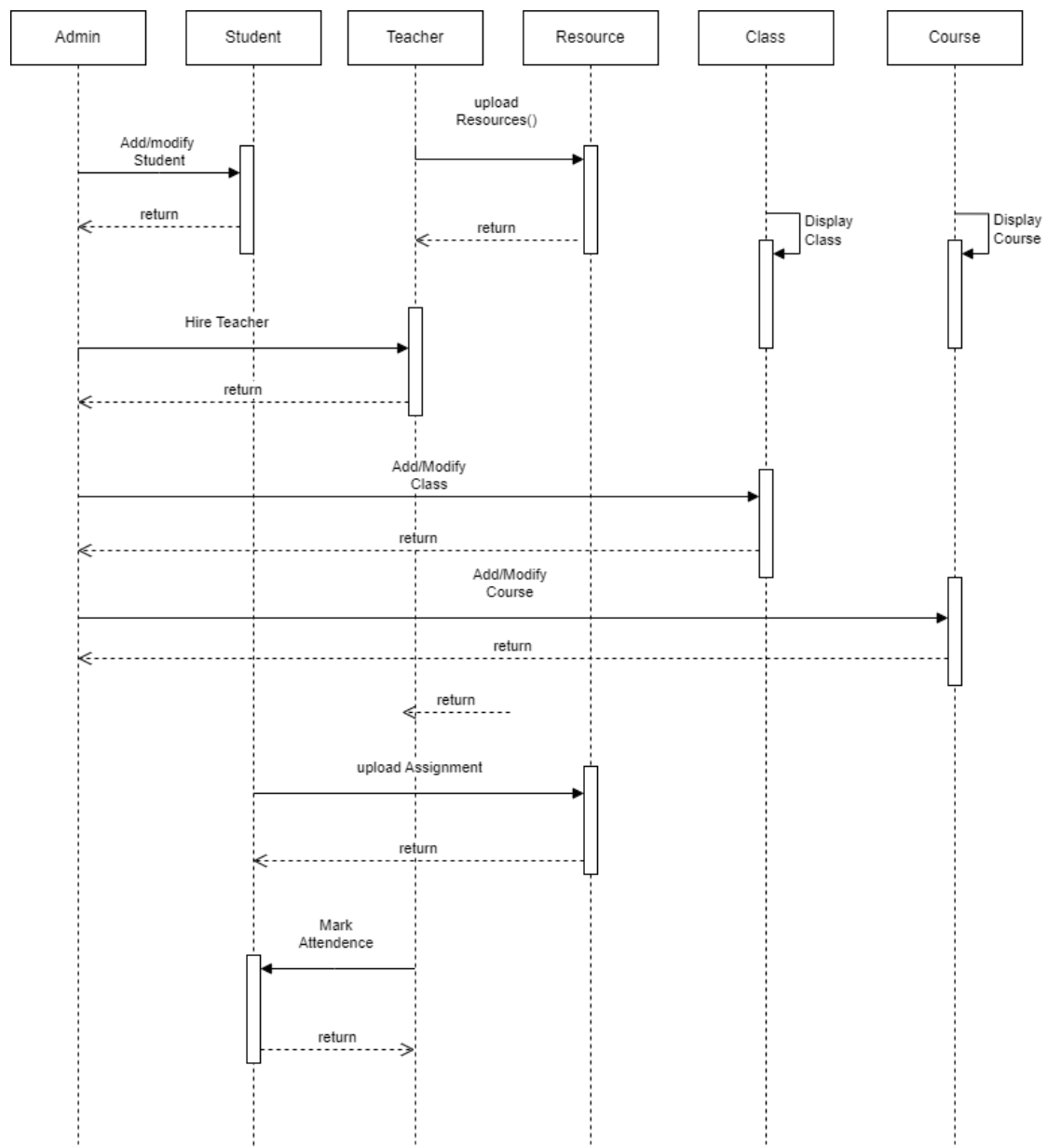


Figure 3.4: Sequence Diagram of System

3.6.3 Activity Diagram

This is the Activity UML diagram of School Management System which shows the flows between the activity of Teacher, Parent, Registration, School and Student. Every Activity in this UML Diagram of School Management System has specific permission set by Admin.

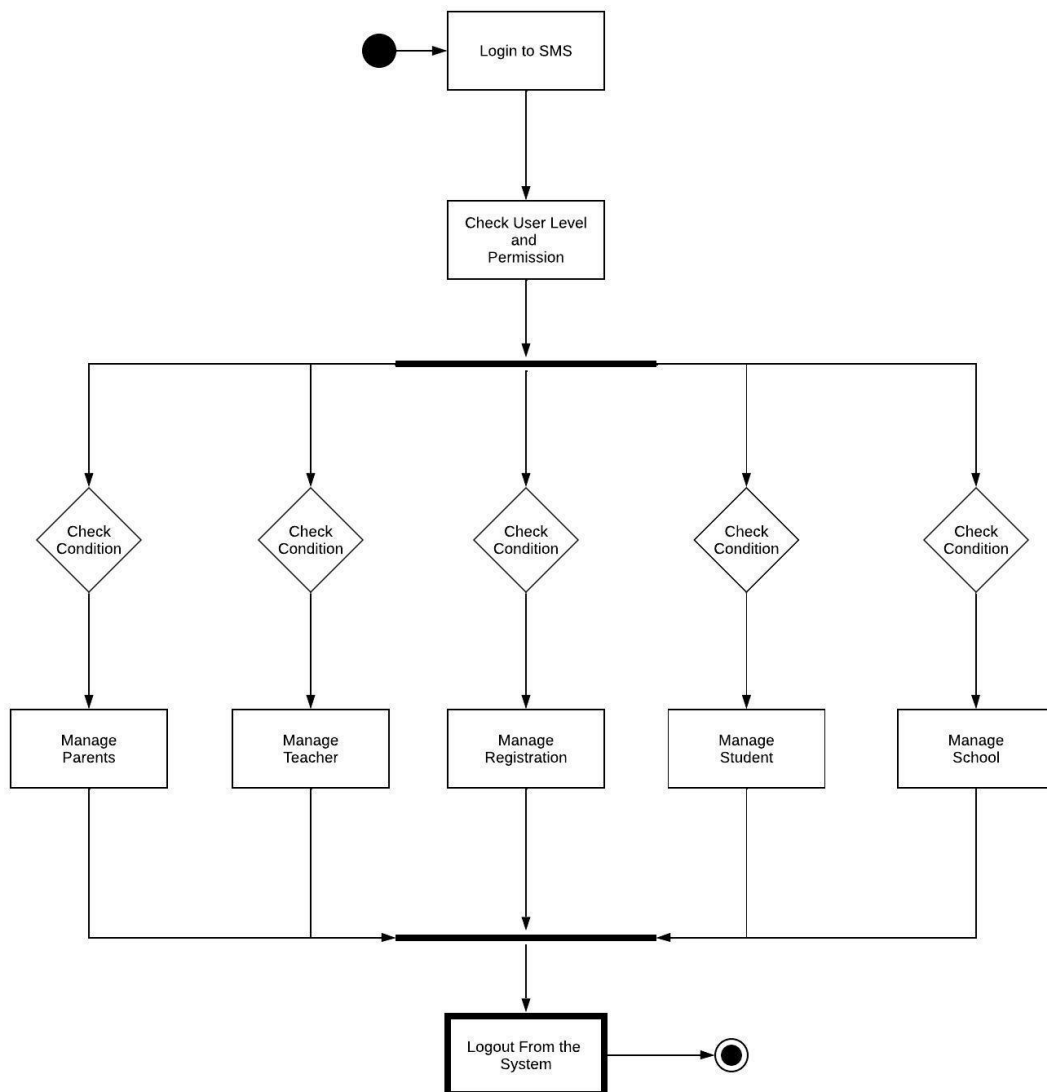


Figure 3.5: Activity Diagram of System

3.6.3.1 Login Activity

This is the Login Activity Diagram of School Management System, which shows the flows of Login Activity, where admin will be able to login using their username and password.

After login user can manage operations on Teacher, Parent, Student, Registration and other feature of School.

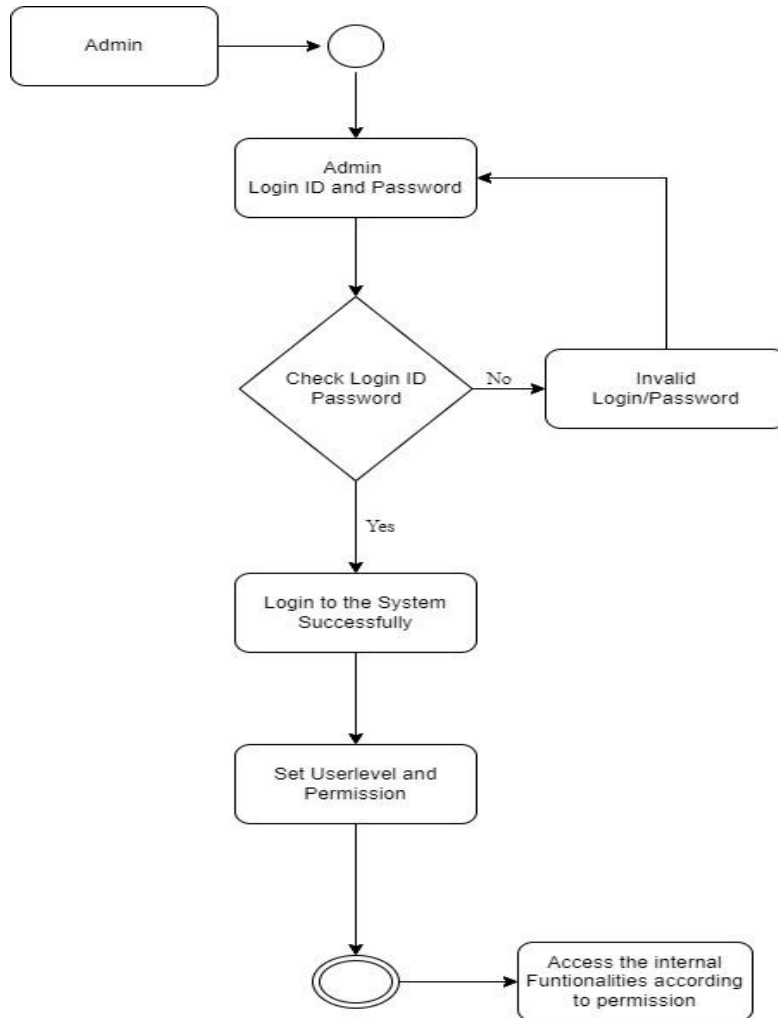


Figure 3.6: Login Activity of System

3.6.4 Component Diagram

This is a Component diagram of School Management System which shows components, provided and required interfaces, ports, and relationships between the Teacher, Course, Registration, School and Student. This type of diagram used in CBD to describe Service-Oriented Architecture (SOA). Component diagram describes the physical components in a system.

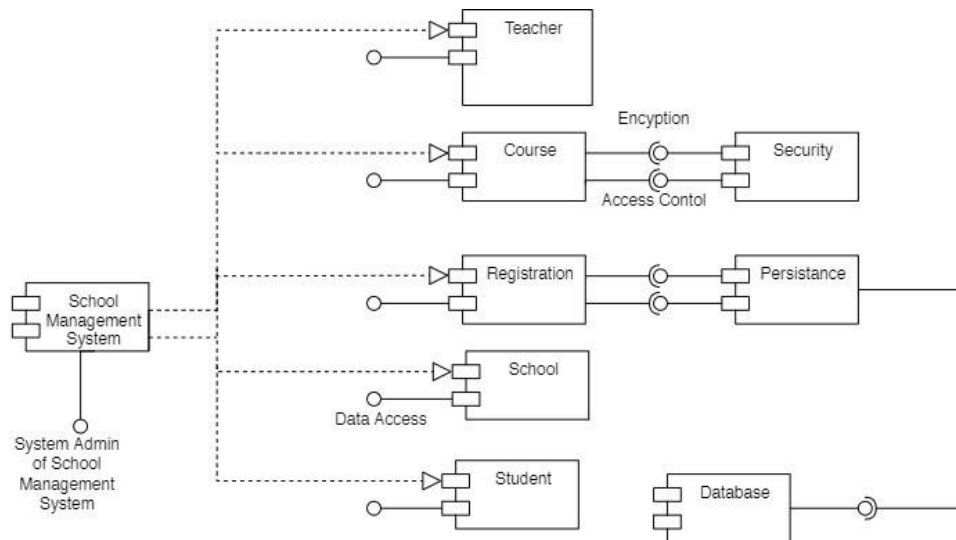


Figure 3.7: Component Diagram of System

3.6.5 Data Flow Diagram

3.6.5.1 DFD Level 0

The zero-level DFD would illustrate the high-level processes and data flows within the system. The School Management System interact with various actor including Admin, Student, Parent, Teacher and will manage all the activities such as Login, Course, Class and Timetable.

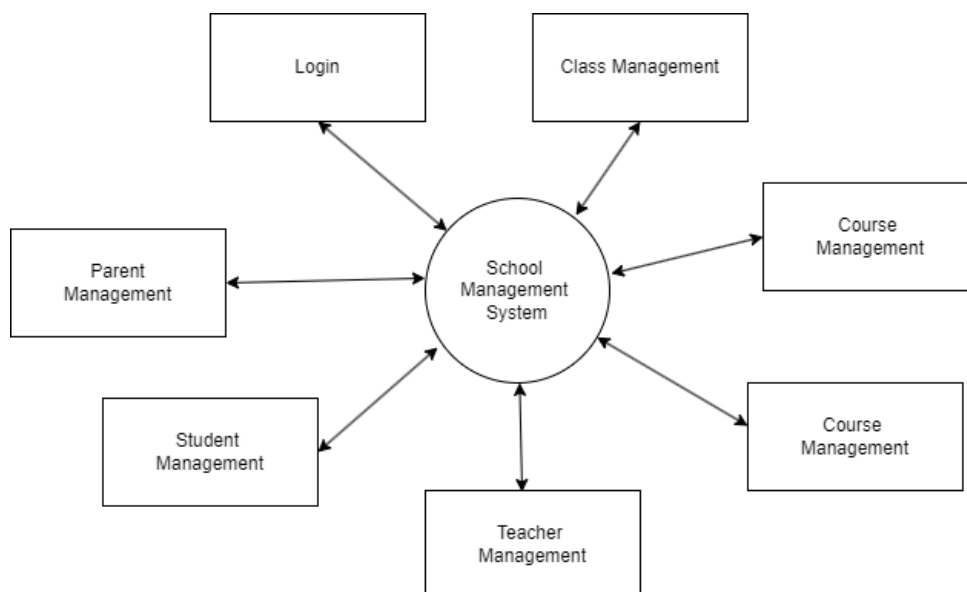


Figure 3.8: DFD Level 0 Diagram

3.6.5.2 DFD Level 1

DFD level 1 gives the more detail than information about processes and data flows within the system. The School Management System is still the central entity representing the entire system. The activities of DFD 0 are Login, Course management, Teacher Management and soon gives the detail view about check user login detail, Generate Course and Teacher data respectively.

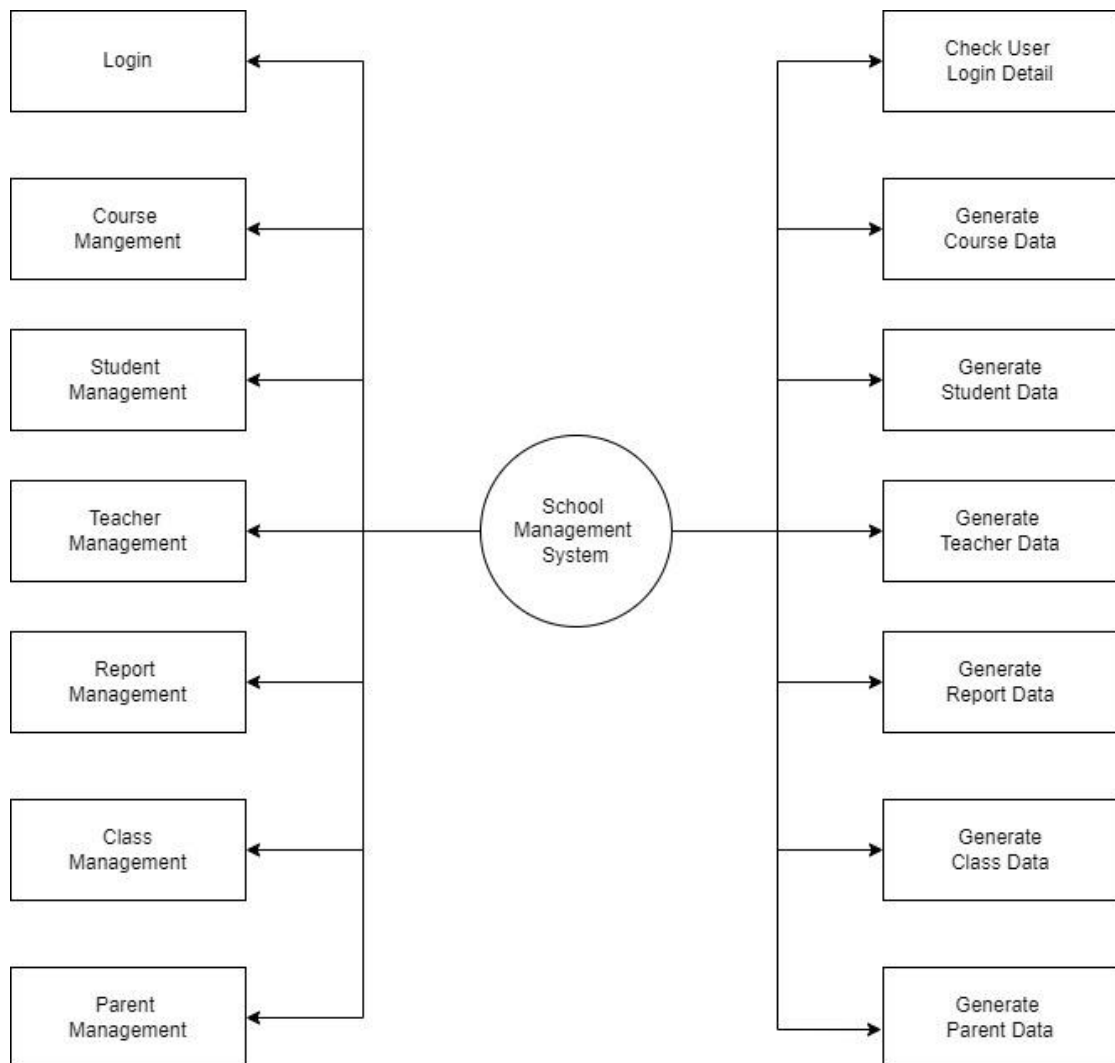


Figure 3.9: DFD Level 1 Diagram

3.6.5.3 DFD Level 2

DFD Level 2 goes one step deeper into parts of Level 1 of School Management System. First Level DFD of School Management System shows how the system is divided into sub-systems or processes. The Second Level DFD contain more details of Subjects, Attendances, Timetable, Teacher, Class, Students. DFD Level 2 diagram of Student , Admin and Teacher is shown below.

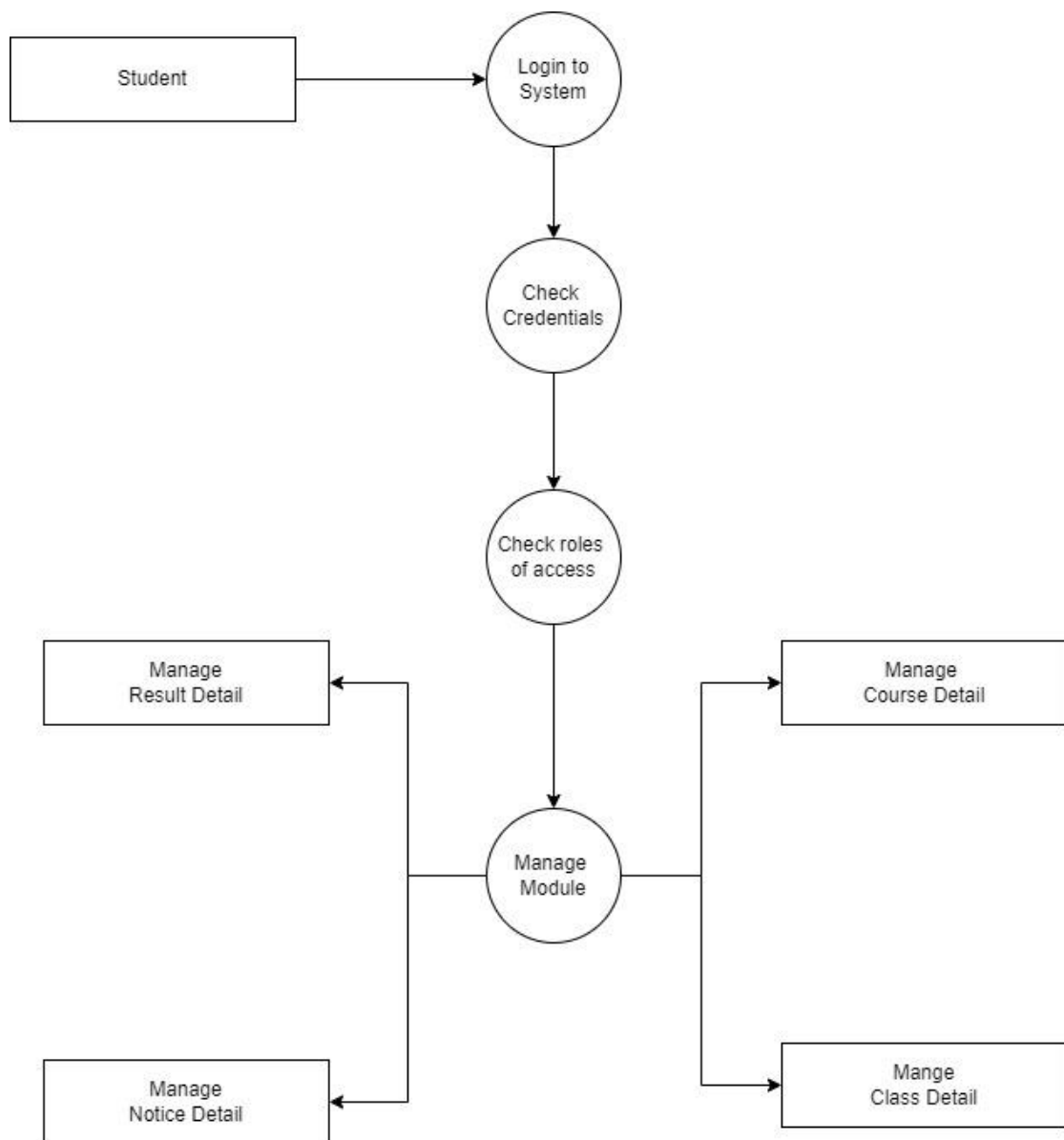


Figure 3.10: DFD Level 2 for Student

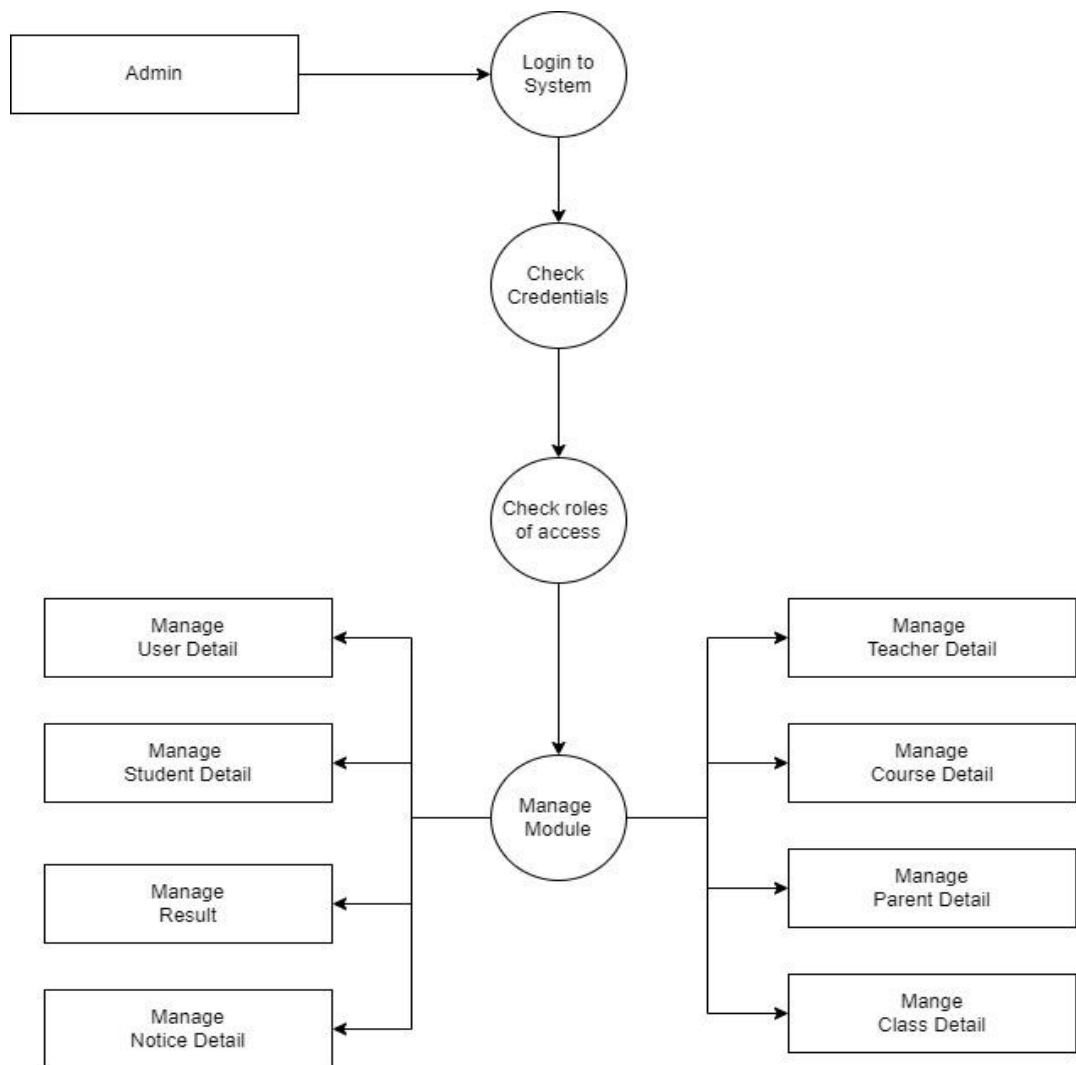


Figure 3.11: DFD Level 2 for Admin

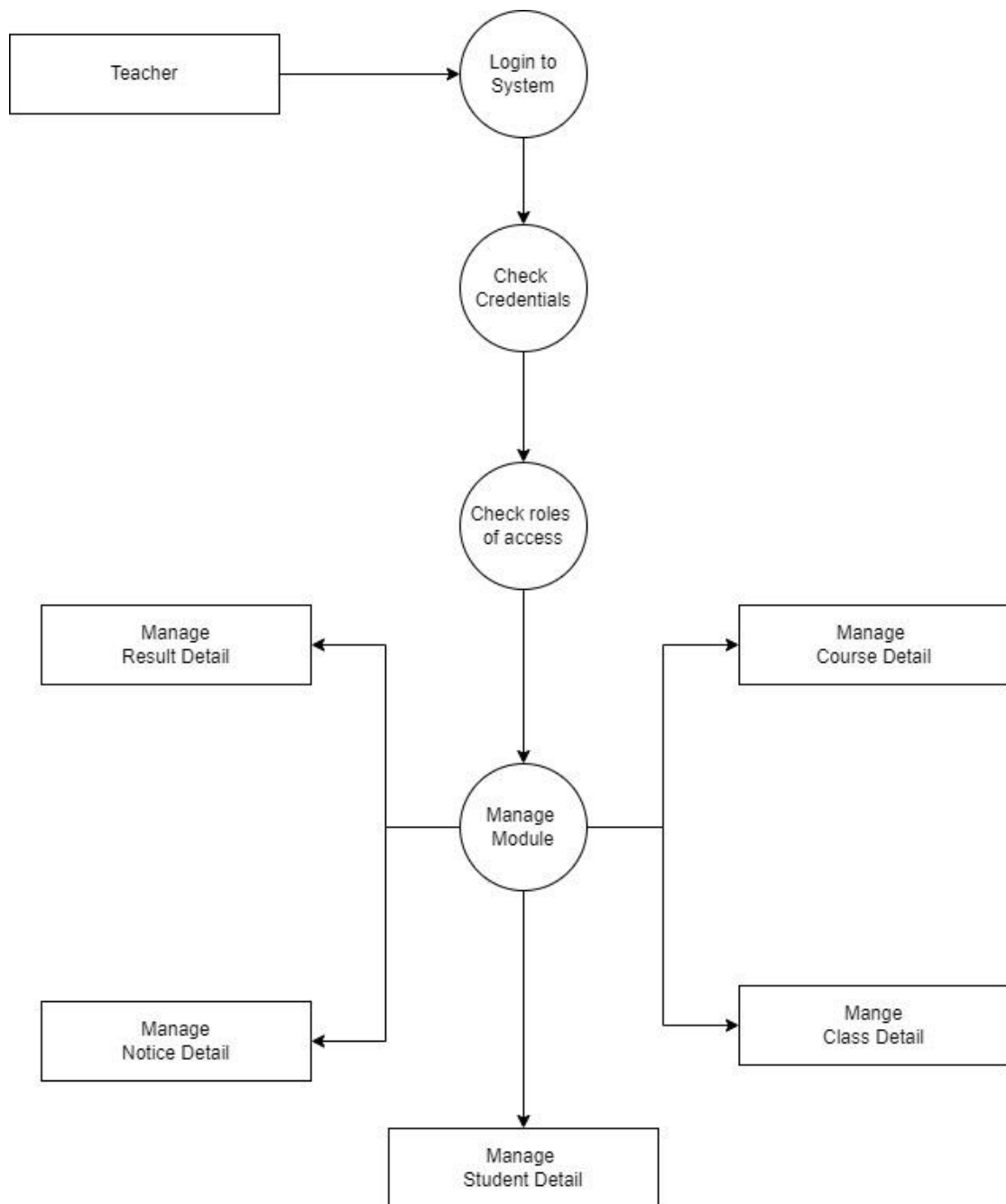


Figure 3.12:DFD Level 2 for Student

3.9 Software Testing

This project report aims to document the software QA testing process for a School Management System developed using Flutter and Firebase technologies. The report outlines the key aspects of QA testing, including requirement analysis, test planning, test

case development, and various types of testing. Additionally, it highlights the importance of QA testing in ensuring the functionality, reliability, and usability of the system. The School Management System is a web and mobile application developed using Flutter, a cross-platform framework, and Firebase, a cloud-based platform. The system aims to streamline school administration processes, enhance communication between stakeholders, and provide efficient management of student information, grades, attendance, and other essential functionalities[13].

Requirement Analysis:

Thorough requirement analysis was conducted to understand the system's objectives, features, and user expectations. The requirements were documented, including modules such as user registration, attendance tracking, grade management, timetable scheduling, report generation, and more. Clear and comprehensive documentation was prepared as a foundation for effective testing.

Test Planning:

A detailed test plan was created to define the scope, objectives, strategies, and timelines of the QA testing process. The plan outlined the different types of tests to be conducted, including functional testing, performance testing, security testing, usability testing, and compatibility testing. Testing environments, test data, and resources required were also specified in the plan.

Test Case Development:

Based on the requirements and test plan, test cases were developed to cover the system's functionalities comprehensively. Each test case represented a specific scenario or functionality to be tested, along with the expected results. Test cases were designed to cover all the modules and ensure accurate and effective testing.

Functional Testing:

Functional testing was performed to verify the system's behavior and functionality. Testers simulated various scenarios and user interactions to ensure that all features of the School Management System worked as intended. This testing phase covered user interfaces, data inputs and outputs, data validations, and the overall system behavior.

Performance Testing:

Performance testing assessed the system's response and stability under varying workloads. Load testing, stress testing, and scalability testing were conducted to evaluate the system's ability to handle a large number of simultaneous users, process requests efficiently, and maintain acceptable response times.

Security Testing:

Security testing focused on evaluating the system's security measures and ensuring the protection of sensitive data. Authentication mechanisms, data encryption, role-based access controls, and vulnerability assessments were performed to identify and address any vulnerabilities, breaches, or unauthorized access risks.

Usability Testing:

Usability testing aimed to evaluate the system's user-friendliness, intuitiveness, and overall user experience. Testers assessed how easily administrators, teachers, and students could navigate the system, perform tasks, and access relevant information. Feedback from real users was collected to gather insights and make necessary improvements.

Compatibility Testing:

Compatibility testing ensured that the School Management System was compatible with various devices, operating systems, browsers, and screen resolutions. This testing phase verified that users could access and use the system seamlessly, regardless of the device or platform they were using.

3.9.1 QA Testing Tools:

The following QA testing tools were utilized during the testing process


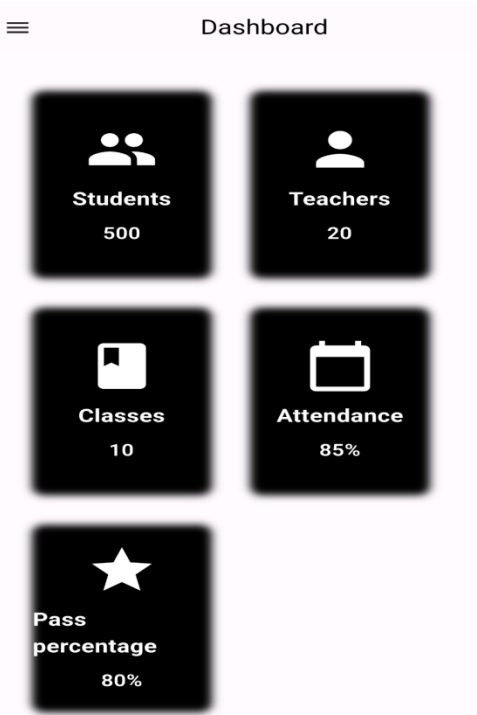
- Test automation: Flutter Driver
- Performance testing: JMeter
- Security testing: OWASP ZAP, Burp Suite
- Compatibility testing: BrowserStack

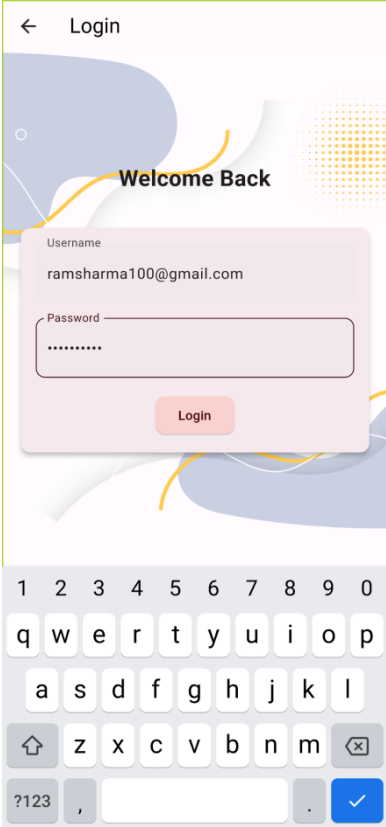
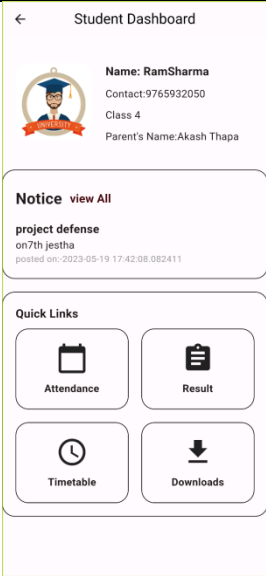
3.9.2 Test Cases

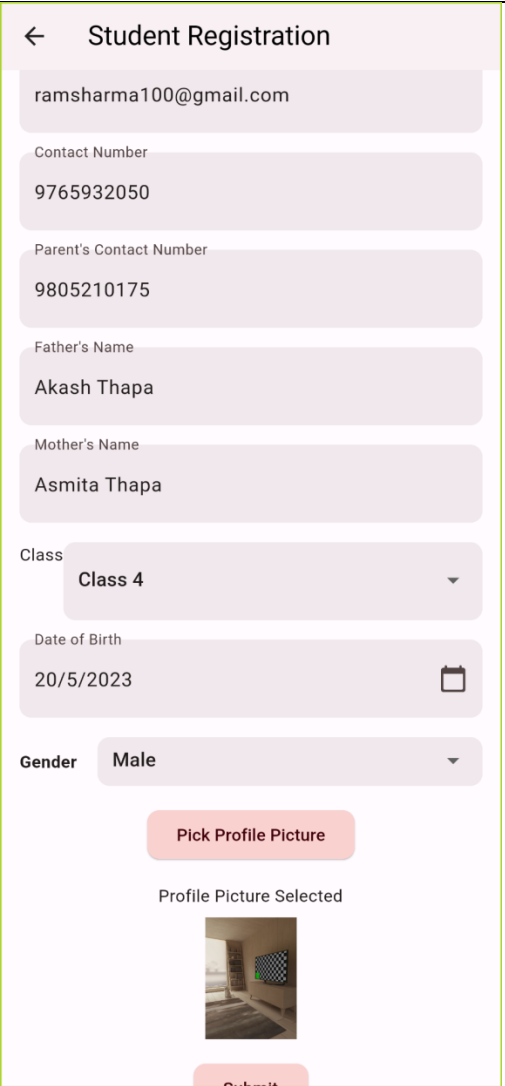
The Testing that is performed for this project is blackbox testing. The test cases performed are given below:

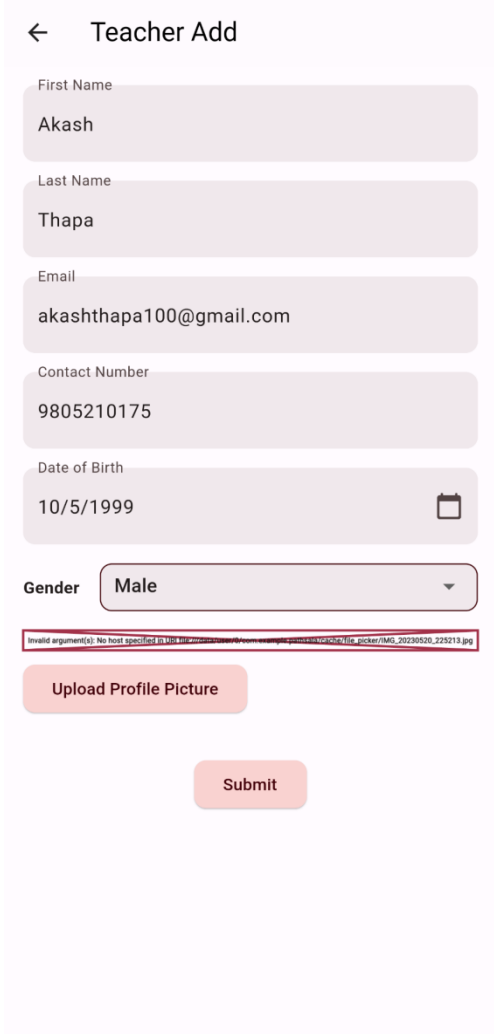
Test Case ID	Test Objectives
T1	To check the admin login
T2	To check the student registration
T3	To check the teacher registration
T4	To check the Teacher login
T5	To check notice announcement or post
T6	To check the student login
T7	To check the resource update

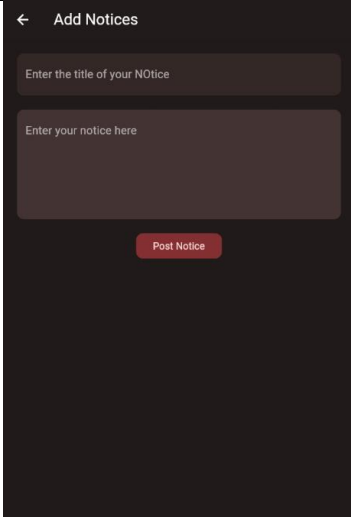
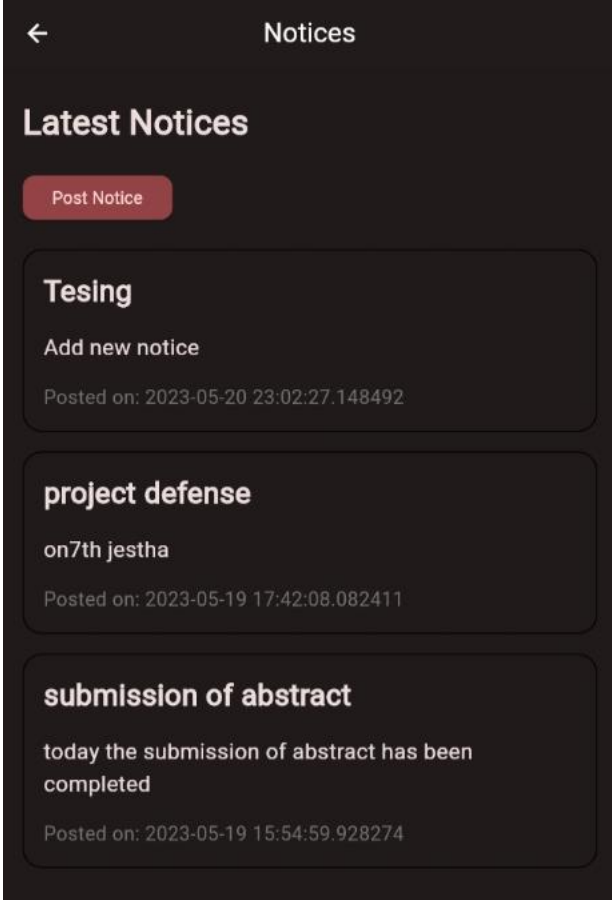
Table 3.1 Test Cases list

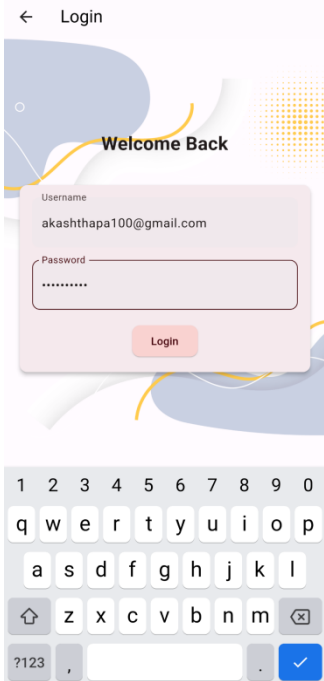
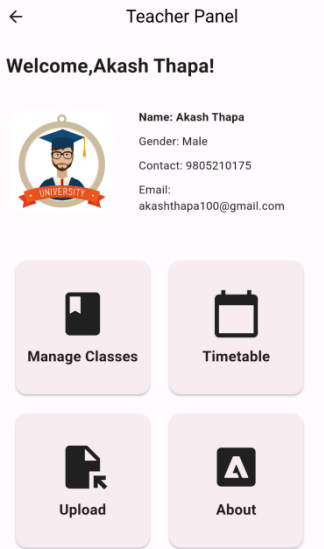
Test Objectives	To check the admin login
Test Data	
Expected Result	After the valid credentials are filled and tapped on login button, the useacer must be navigated to landing page .
Actual Result	<p>After the valid credentials are filled and tapped on login button, the user must be navigated to landing page.</p> 

Test Objective	To check the student login
Test Data	
Expected Output	After the valid credentials are filled and tapped on login button, the useaer must be navigated to landing page .
Actual Output	After the valid credentials are filled and tapped on login button, the useaer must be navigated to landing page .
Result Output	

Test Objectives	To check the student registration
Test Data	
Expected Result	After the form fields are filled and tapped on submit button, the student must register to database.
Actual Result	Student is registered to database

Test Objective	To check Teacher Registration
Test Data	
Expected Output	Teacher will be added to database
Actual Output	Teacher is added to database

Test Objectives	To check the notice announcement or post
Test Data	
Expected Result	After filling the form and submitting notice must be posted and shown in other notice showing section.
Actual Result	

Test Objectives	To check Teacher Login
Test Data	
Expected Result	After the valid credentials are filled and tapped on login button, the useaer must be navigated to landing page .
Actual Result	After the valid credentials are filled and tapped on login button, the useaer must be navigated to landing page .
Result Output	

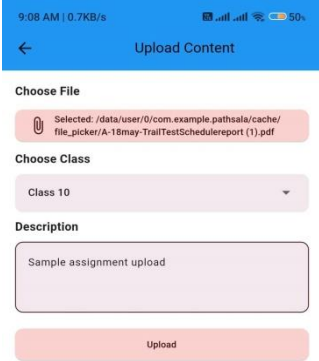
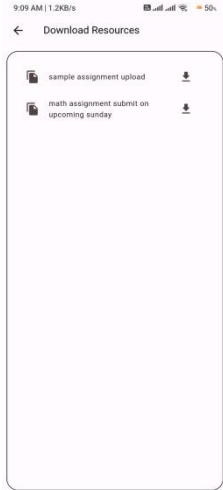
Test Objective	To check the resource update
Test Data	
Expected Output	After filling the form and pressing upload button it should upload it on database and show to the download resource section of student dashboard.
Actual Output	It is uploaded on database and shown to the download resource section of student dashboard.
Result Output	

Table 3.2: Test Cases analysis and result

CHAPTER 4: FEATURE AND FUNCTIONALITIES

4.1 Features and Functionalities

Features and Functionalities of a School Management System :

1. User Authentication and Access Control:

- Secure login and registration for administrators, teachers, students, and parents.
- Role-based access control to ensure appropriate access to system functionalities and data.

2. Student Management:

- Student enrollment and registration management.
- Class and section management.
- Student attendance tracking.
- Academic performance monitoring and grade management.
- Exam and assignment management.

3. Teacher Management:

- Teacher profile management.
- Class allocation and schedule management.
- Attendance recording for student in Application.
- Assignment and assessment management.

4. Parent Communication:

- Parent profiles and contact information management.
- Notification for important updates, events, and announcements.
- Children monitoring

5. Timetable and Schedule Management:

- Creation and management of school timetables.
- Class schedules and subject allocation.

6. Report Management:

- Generating various reports, such as attendance reports and progress reports.

7. Mobile App Support:

- Cross-platform mobile app development with Flutter, allowing access to the school management system on both Android and iOS devices.

4.2 User Manual

4.2.1 Login

After installing the software system user might treated as a public user. The user only can see homepage and Get Started option. When user click on Get Started option, A Login form appears into interface.

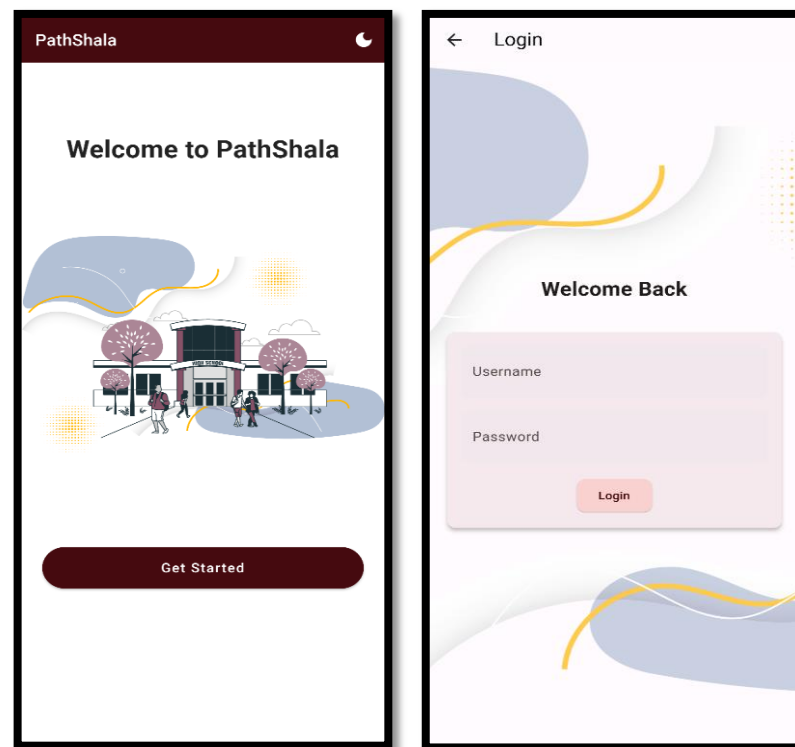


Figure 4.1: Login Process

4.2.2 User Entry

The Admin can add other user if necessary and set user permission for new user. For different role of user has different permission.

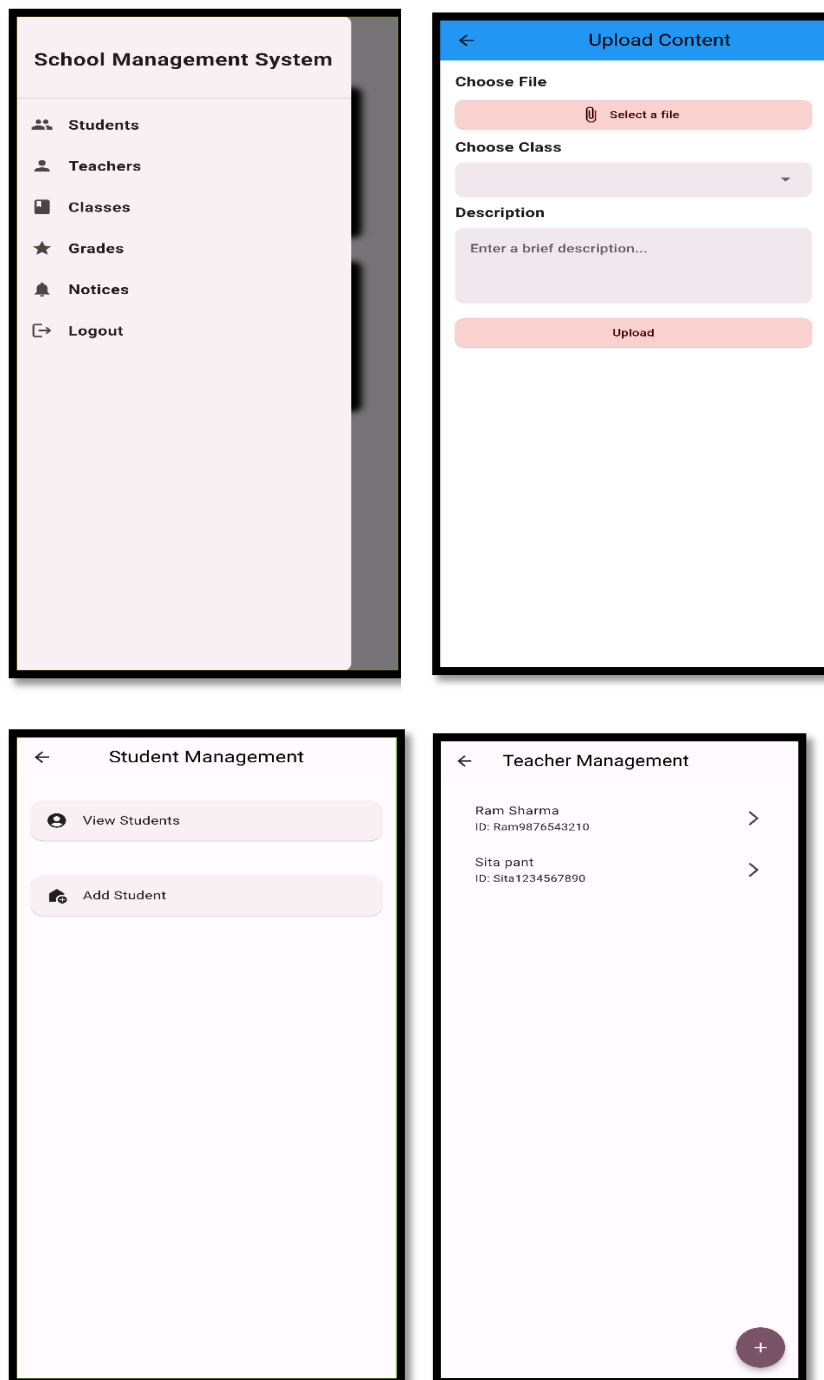


Figure 4.2:User Permission

4.2.3 User Panel

For each user there is different panel according to their role.

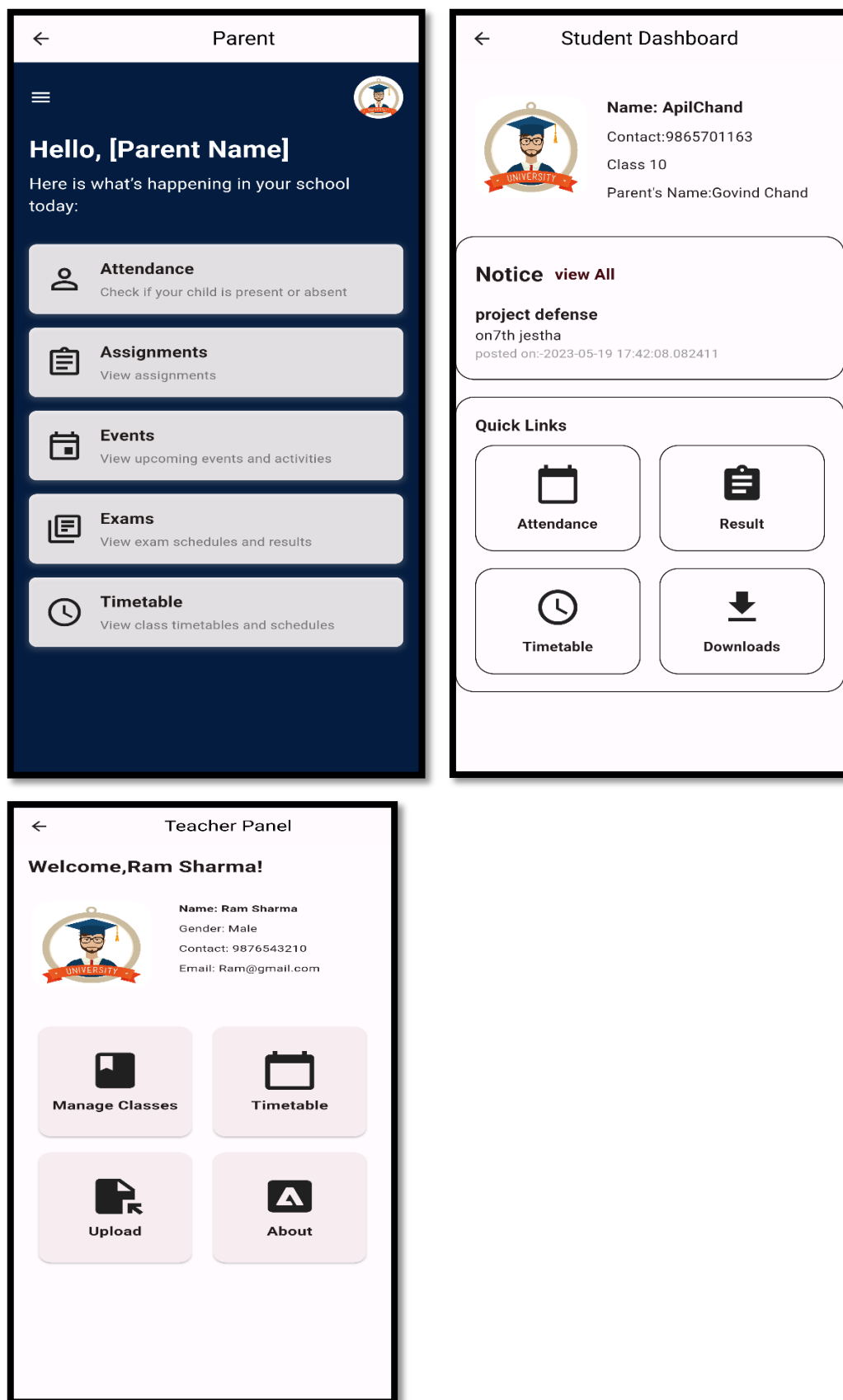


Figure 4.3: User panel

4.2.4 Class

This UI enter all existing class is an educational institute. It gives detail information about students in the class, attendance, notice, result and timetable.

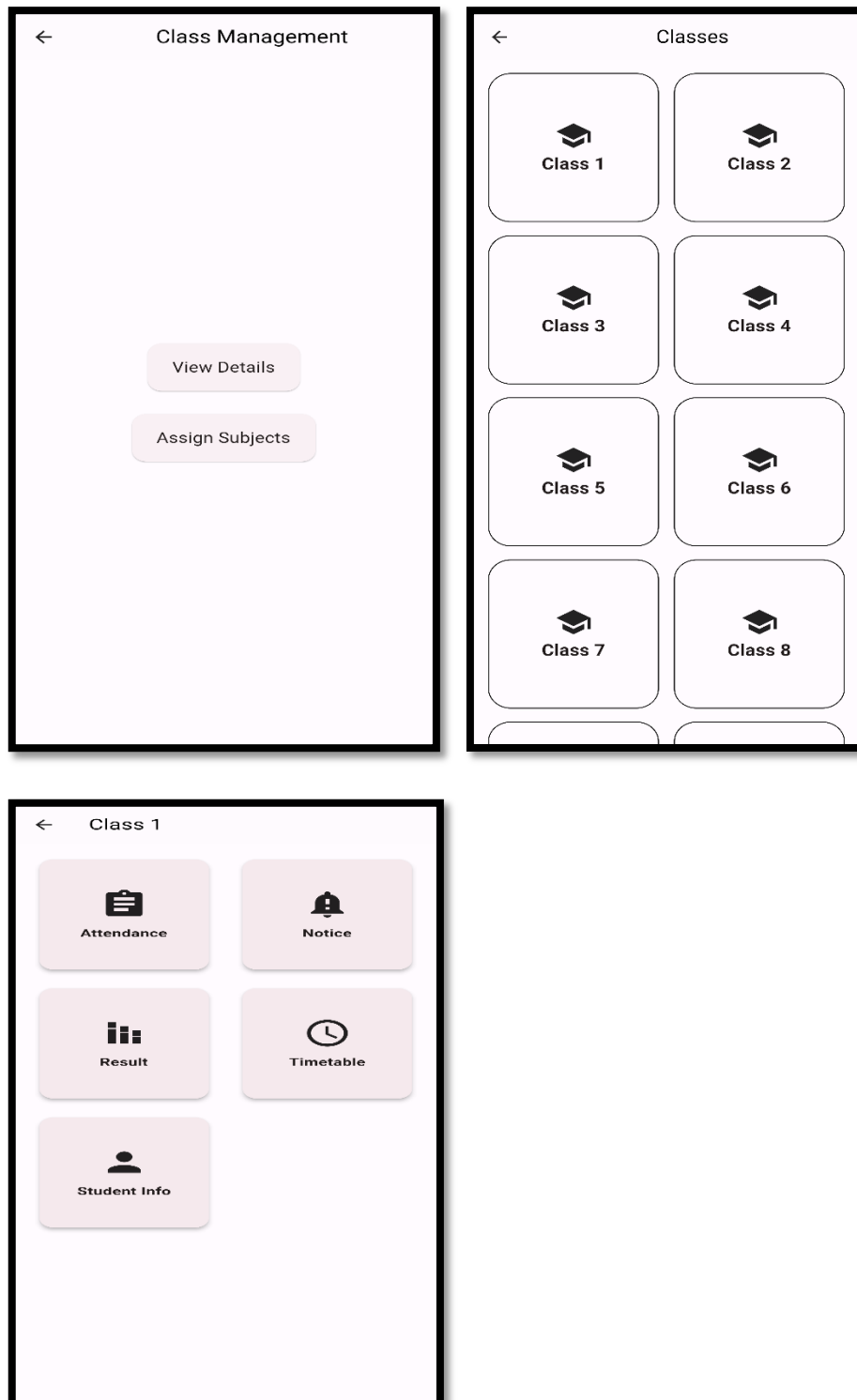


Figure 4.4: Class of System

4.2.5 Grades and Notice

It gives the detail of student Marks and notices of school or institute.

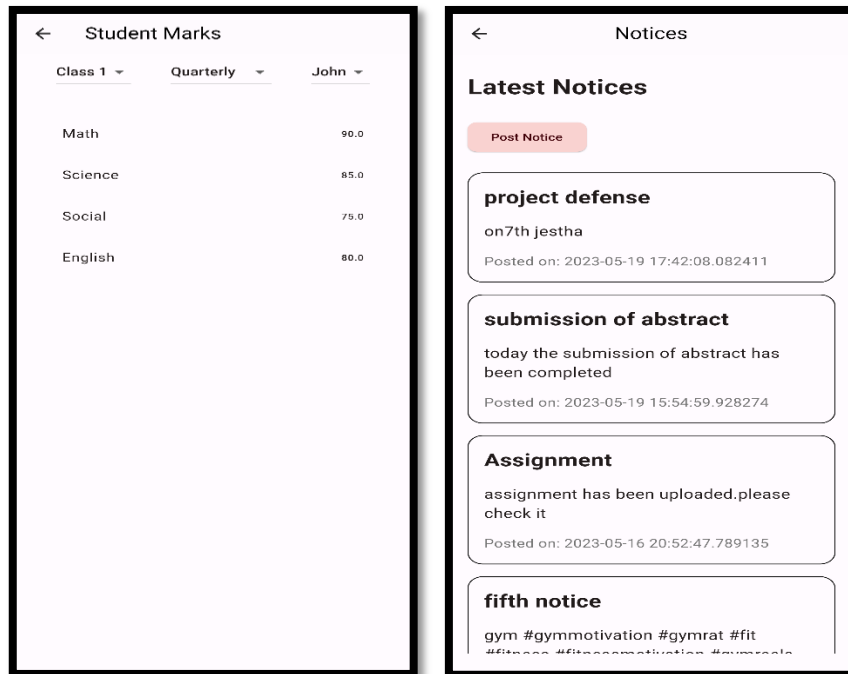


Figure 4.5: Notice and Grades of School

CHAPTER 5: EPILOGUE

5.1 Result and Discussion

The implementation of the school management system yielded positive results and significant improvements in school management processes. The system, which incorporated features such as student enrollment, attendance tracking, grade management, and communication channels, was successfully implemented with modifications tailored to meet the specific needs of educational institutions. It resulted in notable time savings in administrative tasks and increased efficiency. Qualitative feedback from teachers, administrators, and students emphasized the system's effectiveness in enhancing school management and communication. Despite encountering challenges during implementation, such as technical issues and user adoption, the overall impact of the digital system surpassed traditional methods in terms of time, resource utilization, and accuracy. The discussion highlighted the significance of the system's role in improving administrative efficiency, data management, and user satisfaction. Additionally, limitations concerning technical requirements, user training, and scalability were addressed. Overall, the successful implementation of the school management system demonstrated its positive influence on education institutions, validating its effectiveness in transforming school management processes and facilitating better communication among stakeholders.

5.2 Future Enhancements

Looking towards the future, we have identified several areas for further development and enhancements to maximize the potential of the School Management System. These recommendations aim to address emerging needs and provide an even more comprehensive and efficient solution for our school community:

Library management and Transportation management: In terms of future enhancements, the School Management System can benefit from further developments of library management and transportation management. For the library, implementing an online catalog and reservation system would allow users to conveniently search and reserve resources from anywhere. Additionally, incorporating a digital resource management

system would expand the library's collection to include e-books and digital publications, providing broader access to a variety of materials.

In transportation management, integrating GPS tracking would enable real-time updates on bus locations, ensuring student safety and allowing parents and administrators to track bus arrivals and departures. Implementing an algorithm-based system for optimal route planning would minimize travel time and fuel consumption, while a notification system would facilitate communication between the transportation department and parents, providing alerts on bus delays, route changes, or emergencies. Integration with the attendance system would automate attendance recording when students board or disembark from buses, streamlining attendance management processes.

Integration with Learning Management Systems: Explore opportunities for integration with popular Learning Management Systems (LMS) to streamline the process of sharing course materials, assignments, and grades. Seamless integration between the School Management System and LMS would provide a unified platform for both administrative and academic purposes.

Enhanced Communication Channels: Introduce additional communication channels within the system, such as in-app messaging or discussion forums, to facilitate effective communication between teachers, parents, and students. This would enable timely updates, clarifications, and collaboration, fostering stronger engagement and involvement in the educational process.

Advanced Data Analytics and Use of AI: Incorporate advanced data analytics and capabilities into the system to generate comprehensive reports and insights. This would allow administrators and teachers to identify patterns, trends, and areas for improvement in student performance, attendance, and overall school operations. Data-driven decision-making can support targeted interventions and personalized approaches to enhance student outcomes.

Integration with Financial Management Systems: Integrate the School Management System with the school's financial management software to streamline fee collection, expense tracking, and financial reporting. This would provide administrators with a holistic view of financial transactions and facilitate accurate and efficient financial management.

Continuous Training and Support: Establish a comprehensive training program for teachers and staff members to ensure they are proficient in utilizing the system's features. Ongoing support and regular workshops can help them explore advanced functionalities and make the most of the system's capabilities.

User Feedback and Iterative Improvements: Regularly solicit feedback from users, including teachers, parents, and students, to identify pain points and areas of improvement. Use this feedback to drive iterative updates and enhancements to the School Management System, ensuring that it remains responsive to the evolving needs of our school community.

By implementing these future developments and recommendations, it can further enhance the School Management System's effectiveness, usability, and overall impact. These enhancements will contribute to a more streamlined, data-driven, and collaborative educational environment, benefitting all stakeholders involved in school community.

5.3 Conclusions

In conclusion, the development of the school management system has been a transformative project for any institution. Through careful planning, collaboration, and implementation, a robust system that addresses the complex needs of any school has been successfully created.

The project has resulted in significant improvements in various areas of school management. With the system in place, streamlined administrative processes, reduced paperwork, and increased efficiency can be witnessed. Tasks such as student enrollment, attendance tracking, timetable scheduling, and resource management have become automated, saving valuable time and resources.

Moreover, the school management system has greatly enhanced communication and collaboration among stakeholders. Teachers, students, parents, and administrators can now easily access and share information through online portals and communication channels. This has fostered better engagement and involvement from all parties, leading to improved parent-teacher relationships and student support.

Furthermore, the system's data-driven features have provided us with valuable insights for informed decision-making. Real-time data on student performance, attendance trends, and resource utilization have allowed to identify areas of improvement and implement targeted interventions. This will positively impact student outcomes and overall school performance.

Overall, the development of the school management system has been a resounding success. It has revolutionized our school's operations, promoting efficiency, transparency, and collaboration. We are confident that this system will continue to evolve and adapt to meet the changing needs of our school community, ensuring a more effective and student-centered learning environment.

REFERENCES

- [1]J. Visscher, J. Wild and A. Fung, "Computerized school information systems: A review of the literature," in *IEEE Transactions on Education*, vol. 44, no. 2, pp.151-172, May 2001, doi: 10.1109/13.925383.
- [2]M. Breiter and J. Light, "The impact of technology on academic facilities," in *Proceedings of the 2006 IEEE International Conference on Information Technology: Research and Education*, 2006, pp. 1-4, doi: 10.1109/ITRE.2006.261301.
- [3]M. Durnali, "The impact of e-class management on student data management," in *2013 International Conference on Interactive Mobile and Computer Aided Learning (IMCL)*, 2013, pp. 1-4, doi: 10.1109/IMCL.2013.6640196.
- [4]S. Yıkıcı, M. Kılıç, and M. Durnali, "The impact of technology on society: Reflections on socioeconomic, cultural, and technical change," in *2019 International Conference on Computer Science and Engineering (UBMK)*, 2019, pp. 1-6, doi: 10.1109/UBMK.2019.8906829.
- [5]C. Chen, Y. Chen, and C. Chen, "The value of class management method in creative information management: A case study in Taiwan," in *2014 International Conference on Information Management, Innovation Management and Industrial Engineering*, 2014, pp. 1-4, doi: 10.1109/ICIII.2014.6972766.
- [6]M. Polat and I. Arabaci, "The adequacy of e-class management system in terms of administrative relations, student affairs and student report card work time: A study in Turkey," in *2013 International Conference on Interactive Mobile and Computer Aided Learning (IMCL)*, 2013, pp. 1-4, doi: 10.1109/IMCL.2013.6640197.
- [7]M. Pavlović, D. Randić, and P. Paović, "Web-based information system for class management," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014, pp. 1079-1084, doi: 10.1109/MIPRO.2014.6859729.

[8]O. Akinloye, E. Adu, and A. Ojo, "Academic records management: A key to institutional leverage," in *2017 International Conference on Computational Science and Computational Intelligence (C)*, 2017, pp. 1042-1047, doi: 10.1109/CSCI.2017.180.

[9]J. Strickley, "Key areas for e-class information and management systems," in *2011 International Conference on Information Science and Applications (ICISA)*, 2011, pp. 1-4, doi: 10.1109/ICISA.2011.5770009.

[10] eDigitalNepal. (n.d.). School Management System. Retrieved May 20, 3, from <https://edigitalnepal.com/p/school-management-system-1983>

[11] Veda. (n.d.). About Veda. Retrieved May 20, 2023, from <https://veda-app.com>

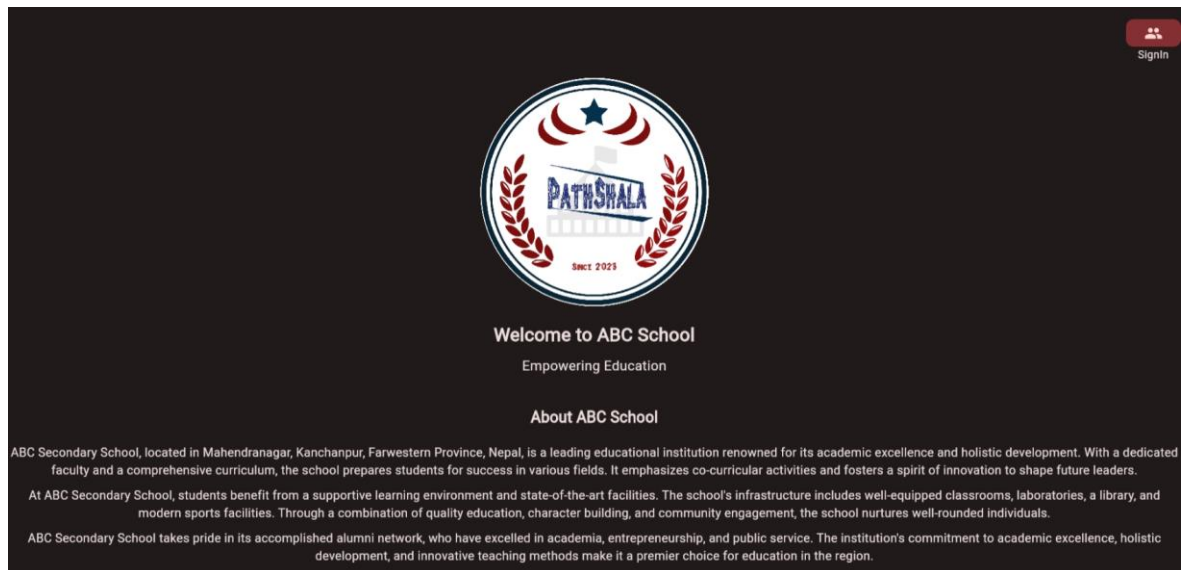
[12] JavaTpoint. (n.d.). Agile vs Waterfall Model. Retrieved May 20, 2023, from <https://www.javatpoint.com/agile-vs-waterfall-model>

[13] "Software Testing," IBM, [Online]. Available: <https://www.ibm.com/topics/software-testing#:~:text=Software%20testing%20is%20the%20process,development%20costs%20and%20improving%20performance>. [Accessed: May 20,2023].

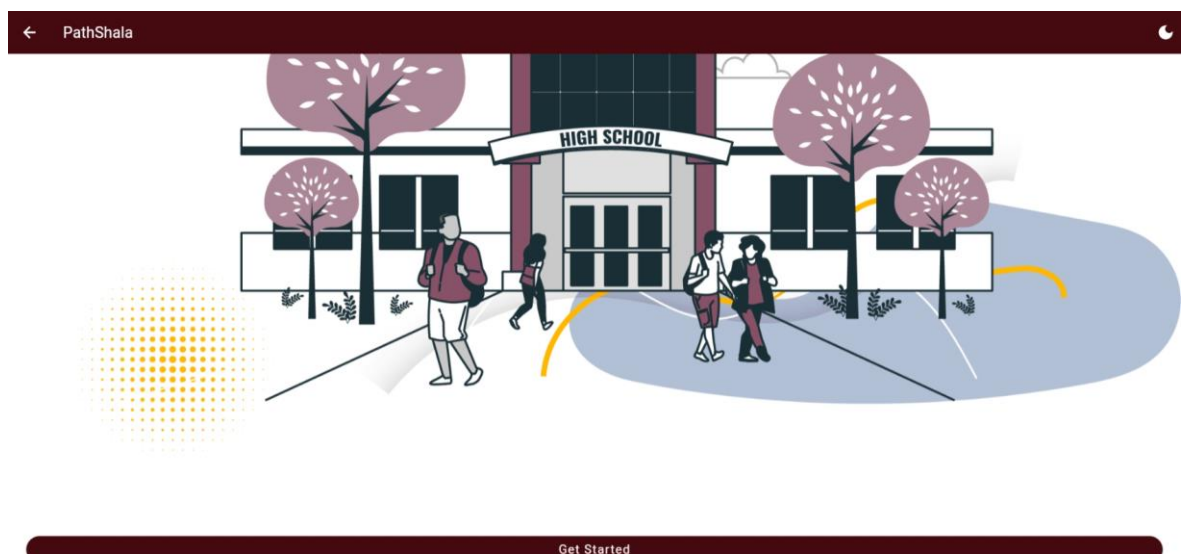
APPENDIX-A

Snapshots

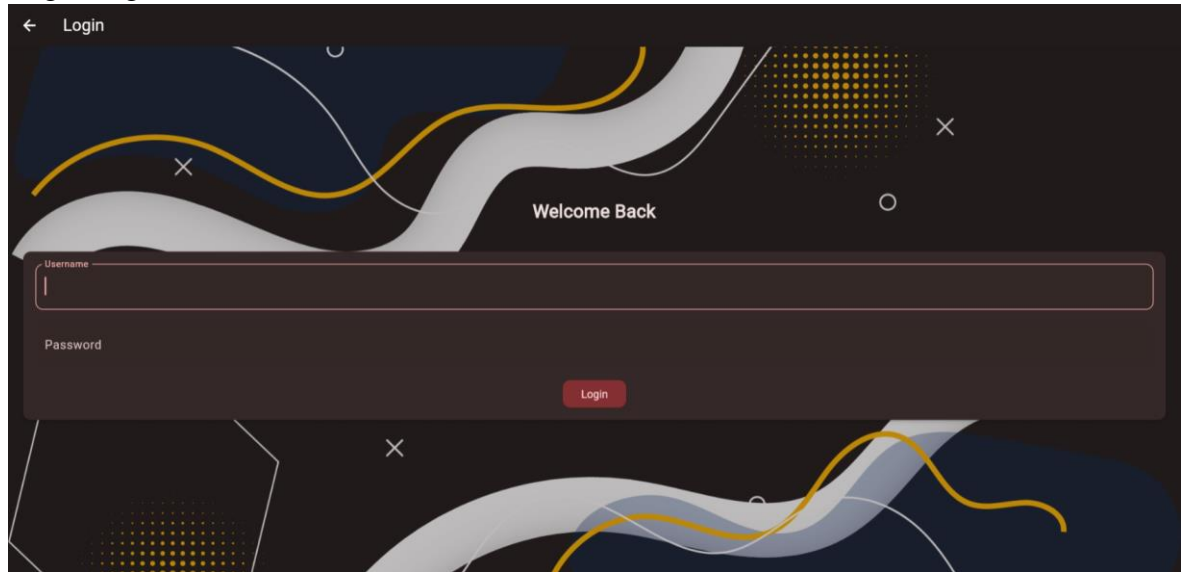
1. Web landing page



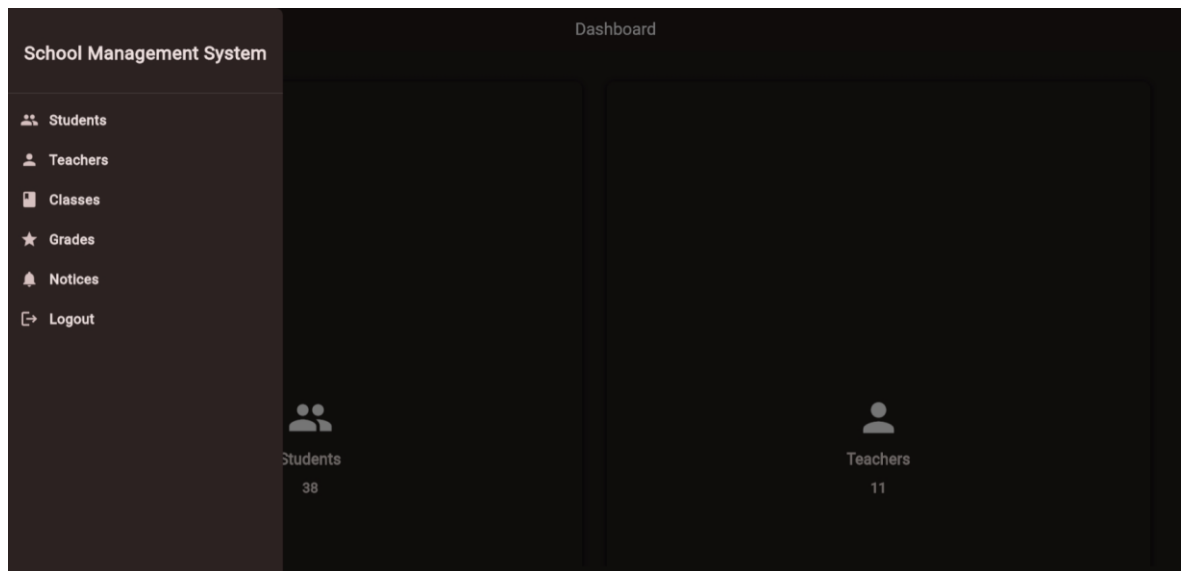
2. Homepage



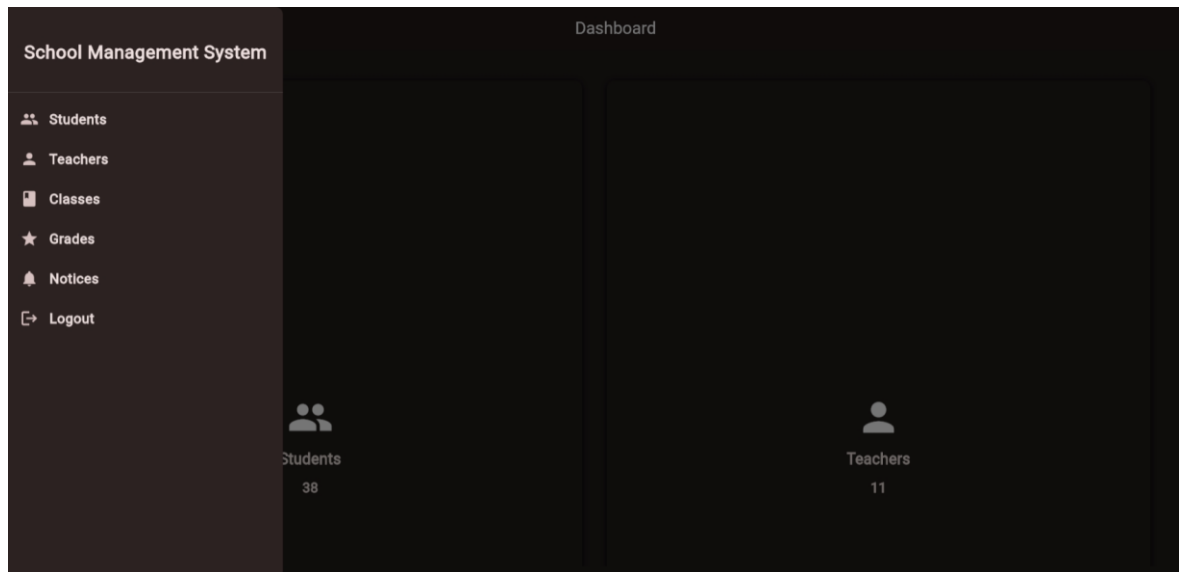
4. Login Page



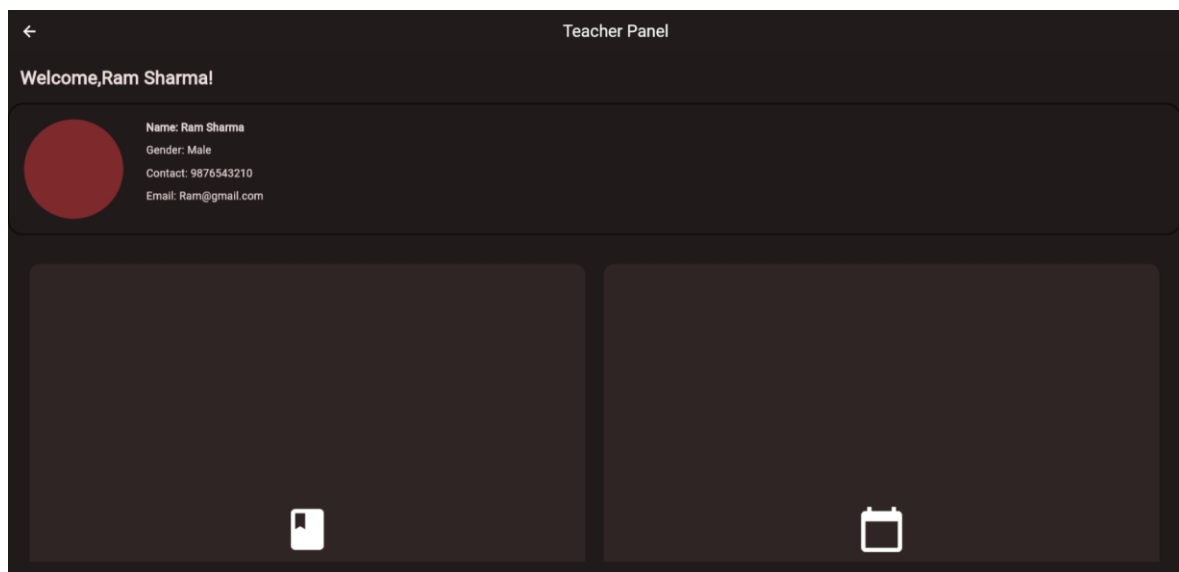
5. Admin Dashboard



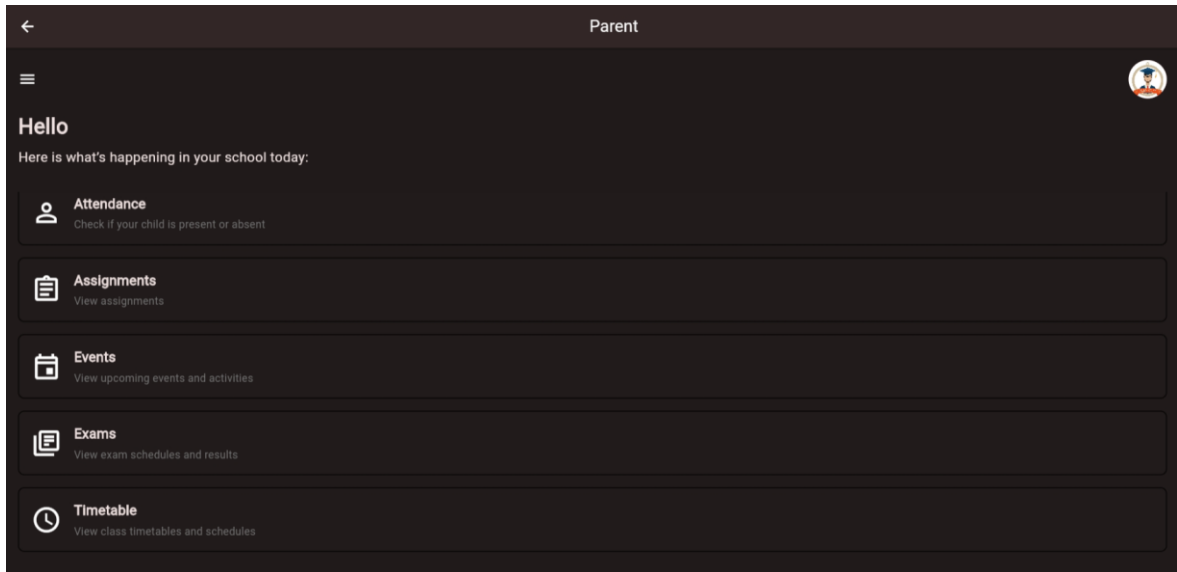
6. Student Dashboard



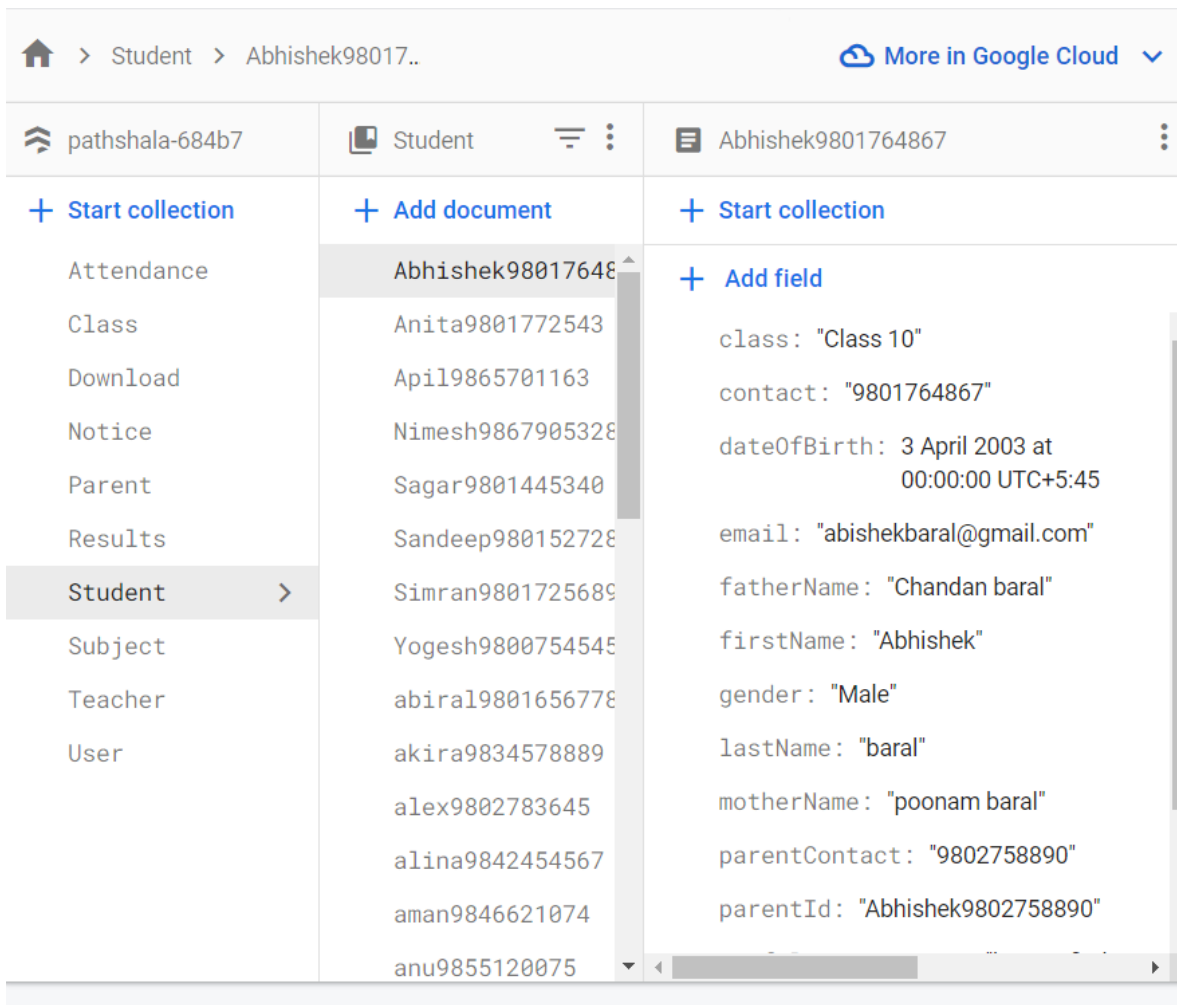
6. Teacher Dashboard



7. Parent Dashboard



8. Firebase Database

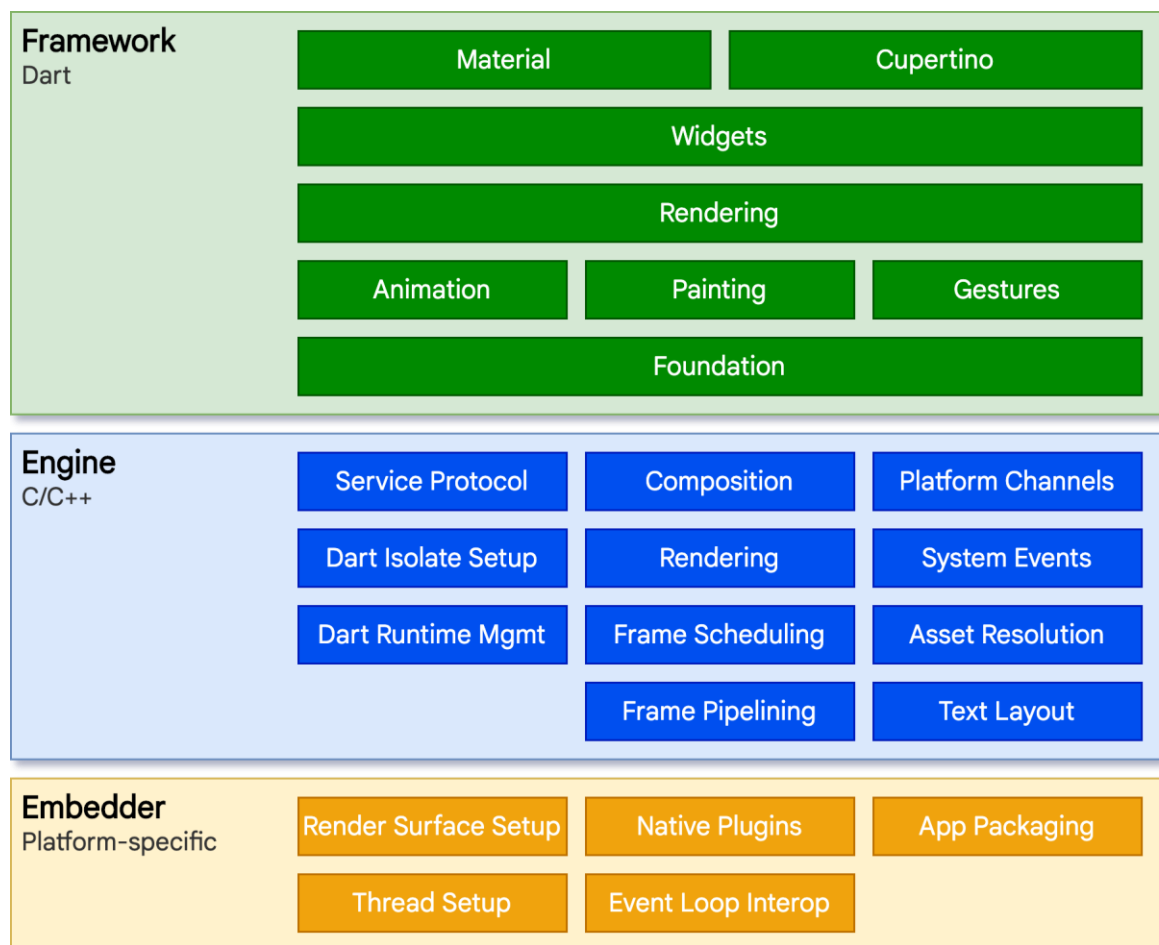


Development and Testing Tools

Flutter

Flutter is a cross-platform UI toolkit that allows developers to create apps for multiple operating systems, such as iOS and Android, while reusing code. It provides a way to interface with platform services and aims to deliver high-performance apps that feel native on each platform. During development, Flutter apps can be hot reloaded, meaning changes can be seen instantly without a full recompile. When released, Flutter apps are compiled to machine code or JavaScript for web deployment. The framework is open source and has a strong ecosystem of third-party packages. The overview covers the construction of Flutter, reactive user interfaces, widgets, the rendering process, platform embedders, integrating Flutter with other code, and support for the web.

Flutter Architecture



Flutter is designed as an extensible, layered system with independent libraries. It packages applications like native apps using platform-specific embedders. The Flutter engine, written in C++, supports core functionalities and is accessed through `dart:ui` in Dart. The Flutter framework, written in Dart, provides a modern, reactive framework with foundational, rendering, and widgets layers. Material and Cupertino libraries offer design-specific controls. Higher-level features are implemented as packages, both from the core libraries and the broader ecosystem. The overview covers UI development, widget composition, platform integration, and Flutter's web support.

Firebase

Firebase is a comprehensive platform provided by Google that offers a wide range of tools and services for building and managing web and mobile applications. It provides developers with backend services, such as real-time databases, authentication, cloud storage, and hosting, allowing them to focus on app development without the need for extensive infrastructure setup.

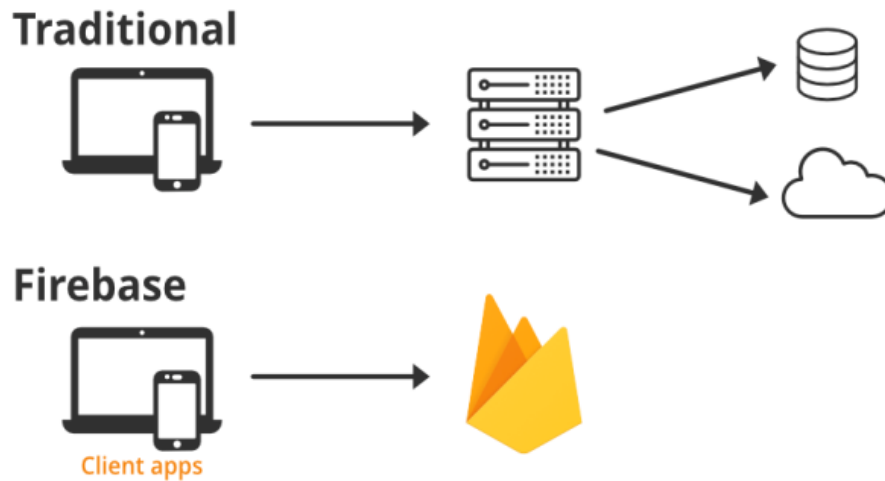
Firebase offers a NoSQL cloud database called Firestore, which allows developers to store and sync data in real-time across multiple clients. It also provides Firebase Authentication, which supports various authentication methods like email/password, social media logins, and phone number verification.

Additionally, Firebase offers features like cloud messaging for sending push notifications to users, cloud functions for serverless computing, remote configuration for managing app settings remotely, and crash reporting to monitor and analyze app crashes.

Firebase's suite of services also includes machine learning capabilities through Firebase ML, allowing developers to integrate machine learning models into their applications with ease. It supports features like image labeling, text recognition, and language translation.

Furthermore, Firebase provides robust analytics tools to track user engagement, app usage, and conversion rates, helping developers make data-driven decisions. It also offers A/B testing capabilities, allowing developers to test different variations of their app to optimize user experience and app performance.

Overall, Firebase is a powerful and user-friendly platform that simplifies backend development tasks, provides essential infrastructure, and enables developers to build high-quality web and mobile applications more efficiently.



QA Testing Tools

Flutter Driver

Flutter Driver is a testing framework and tool provided by Flutter for conducting automated integration tests on Flutter applications. It allows developers to simulate user interactions and verify the behavior and performance of their apps across different platforms.

Flutter Driver operates by running tests as separate processes that communicate with the Flutter app being tested. It provides a set of APIs that enable developers to interact with app elements, such as tapping buttons, scrolling lists, entering text, and verifying the state of the UI. This enables comprehensive testing of the app's functionality and user experience.

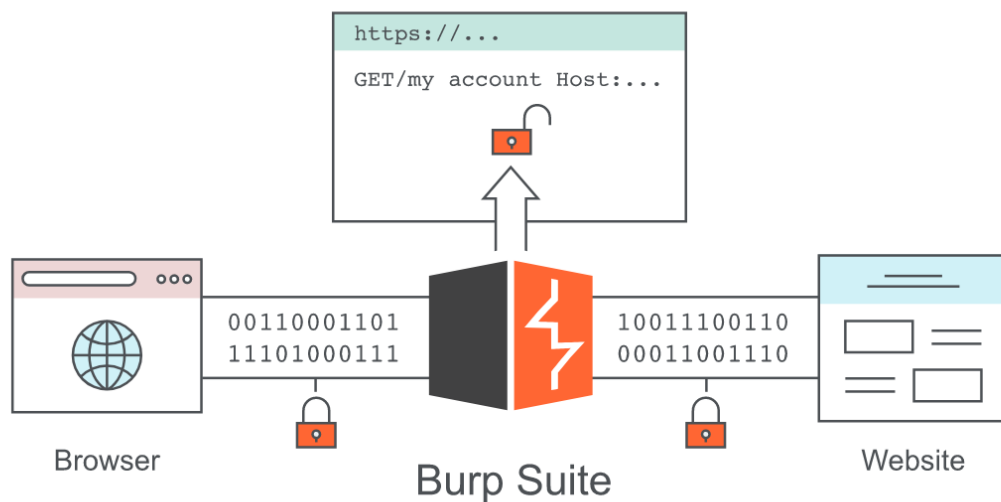
With Flutter Driver, developers can write test scripts in Dart using the Flutter testing framework. These tests can be executed on emulators, simulators, or physical devices. The

tool supports a wide range of test scenarios, including complex interactions, navigation flows, and data validation.

Flutter Driver provides features like synchronization and timeouts to ensure accurate testing results, as well as logging and error handling for debugging purposes. It also integrates with continuous integration (CI) systems, enabling automated testing as part of the development workflow.

By using Flutter Driver, developers can identify and fix issues early in the development cycle, ensure app stability across different devices and platforms, and maintain the quality and reliability of their Flutter applications.

BurpSuite and OWASP zap



Burp Suite and OWASP ZAP are both widely used web application security testing tools that assist developers and security professionals in identifying and mitigating vulnerabilities in web applications.

Burp Suite, developed by PortSwigger, is a comprehensive web application security testing platform. It offers a range of tools and features, including a proxy server that allows capturing and manipulating web traffic, a scanner for automated vulnerability detection, an intruder tool for performing targeted attacks, and a repeater for manual testing and

analysis. Burp Suite also provides various add-ons and extensions, making it highly customizable and adaptable to different testing scenarios.

OWASP ZAP (Zed Attack Proxy), developed by the Open Web Application Security Project (OWASP), is a free and open-source web application security scanner and testing tool. It is designed to help identify security vulnerabilities in web applications and APIs. OWASP ZAP offers similar features to Burp Suite, such as intercepting and modifying web traffic, scanning for common vulnerabilities, and performing manual security testing. It also provides an extensible architecture that allows users to create custom plugins and scripts for specialized testing requirements.

Both Burp Suite and OWASP ZAP support various security testing methodologies, including manual testing, automated scanning, and vulnerability assessment. They can identify common security issues like cross-site scripting (XSS), SQL injection, cross-site request forgery (CSRF), and more. These tools assist in securing web applications by uncovering vulnerabilities and providing recommendations for remediation.

Overall, Burp Suite and OWASP ZAP are powerful tools in the field of web application security testing, helping developers and security professionals identify and mitigate security risks, and ensuring the overall security and robustness of web applications.

BrowserStack

BrowserStack is a cloud-based web and mobile application testing platform that enables developers and testers to perform real-time testing of their applications across a wide range of browsers, devices, and operating systems. It provides a comprehensive testing infrastructure that allows users to validate the compatibility, functionality, and responsiveness of their applications on different browser and device combinations.

With BrowserStack, users can access a vast collection of virtual machines and physical devices hosted in the cloud, eliminating the need for setting up and maintaining an extensive device lab. It offers support for popular web browsers, including Chrome, Firefox, Safari, Internet Explorer, and Edge, on various operating systems such as Windows, macOS, iOS, and Android.

The platform provides interactive testing capabilities, allowing users to interact with applications in real-time and debug issues directly from their browsers. It also offers features like local testing, which enables testing on internal or development environments securely, and responsive testing, which helps ensure optimal performance across different screen sizes.

BrowserStack offers automated testing capabilities through integration with popular testing frameworks and tools like Selenium, Appium, and Cypress. This allows users to run automated tests on multiple browsers and devices simultaneously, speeding up the testing process and improving test coverage.

Additionally, BrowserStack provides a range of debugging and collaboration features, including live bug logging, network inspection, video recording, and sharing of test results with team members. It also integrates with popular development and project management tools, enabling seamless workflows and integration into existing development processes.

Appendix-B

Sample Code

Main.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flex_color_scheme/flex_color_scheme.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:pathshala/screens/webLanding.dart';
import 'firebase_options.dart';
import 'screens/Welcome.dart';
import 'package:firebase_core/firebase_core.dart';
```

```
Future main() async{
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(
  options: DefaultFirebaseOptions.currentPlatform,);

  runApp(MyApp());
}

// ignore: must_be_immutable
class MyApp extends StatelessWidget {
  FirebaseFirestore db = FirebaseFirestore.instance;

  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Pathshala',
      theme: FlexThemeData.light(
```

scheme: FlexScheme.purpleBrown,
surfaceMode: FlexSurfaceMode.highBackgroundLowScaffold,
blendLevel: 1,
appBarStyle: FlexAppBarStyle.background,
bottomAppBarElevation: 2.0,
subThemesData: const FlexSubThemesData(
 useTextTheme: true,
 useM2StyleDividerInM3: true,
 thinBorderWidth: 3.0,
 thickBorderWidth: 0.5,
 adaptiveRemoveElevationTint: FlexAdaptive.excludeWebAndroidFuchsia(),
 adaptiveElevationShadowsBack: FlexAdaptive.excludeWebAndroidFuchsia(),
 adaptiveAppBarScrollUnderOff: FlexAdaptive.excludeWebAndroidFuchsia(),
 defaultRadius: 10.0,
 defaultRadiusAdaptive: 16.0,
 adaptiveRadius: FlexAdaptive.excludeWebAndroidFuchsia(),
 elevatedButtonSchemeColor: SchemeColor.onPrimaryContainer,
 elevatedButtonSecondarySchemeColor: SchemeColor.primaryContainer,
 outlinedButtonOutlineSchemeColor: SchemeColor.primary,
 toggleButtonsBorderSchemeColor: SchemeColor.primary,
 segmentedButtonSchemeColor: SchemeColor.primary,
 segmentedButtonBorderSchemeColor: SchemeColor.primary,
 unselectedToggleIsColored: true,
 sliderValueTinted: true,
 inputDecoratorSchemeColor: SchemeColor.primary,
 inputDecoratorBackgroundAlpha: 19,
 inputDecoratorUnfocusedHasBorder: false,
 inputDecoratorFocusedBorderWidth: 1.0,
 inputDecoratorPrefixIconSchemeColor: SchemeColor.primary,
 fabUseShape: true,
 fabAlwaysCircular: true,

fabSchemeColor: SchemeColor.tertiary,
cardRadius: 14.0,
popupMenuRadius: 6.0,
popupMenuElevation: 3.0,
dialogRadius: 18.0,
datePickerDialogRadius: 18.0,
timePickerDialogRadius: 18.0,
appBarScrolledUnderElevation: 1.0,
drawerElevation: 1.0,
drawerIndicatorSchemeColor: SchemeColor.primary,
bottomSheetRadius: 18.0,
bottomSheetElevation: 2.0,
bottomSheetModalElevation: 4.0,
bottomNavigationBarMutedUnselectedLabel: false,
bottomNavigationBarMutedUnselectedIcon: false,
menuRadius: 6.0,
menuElevation: 3.0,
menuBarRadius: 0.0,
menuBarElevation: 1.0,
menuBarShadowColor: Color(0x00000000),
navigationBarSelectedLabelSchemeColor: SchemeColor.primary,
navigationBarMutedUnselectedLabel: false,
navigationBarSelectedIconSchemeColor: SchemeColor.onPrimary,
navigationBarMutedUnselectedIcon: false,
navigationBarIndicatorSchemeColor: SchemeColor.primary,
navigationBarIndicatorOpacity: 1.00,
navigationBarElevation: 1.0,
navigationRailSelectedLabelSchemeColor: SchemeColor.primary,
navigationRailMutedUnselectedLabel: false,
navigationRailSelectedIconSchemeColor: SchemeColor.onPrimary,
navigationRailMutedUnselectedIcon: false,

```

        navigationRailIndicatorSchemeColor: SchemeColor.primary,
        navigationRailIndicatorOpacity: 1.00,
        navigationRailBackgroundSchemeColor: SchemeColor.surface,
    ),
    keyColors: const FlexKeyColors(
        useSecondary: true,
        useTertiary: true,
        keepPrimary: true,
    ),
    visualDensity: FlexColorScheme.comfortablePlatformDensity,
    useMaterial3: true,

),
darkTheme: FlexThemeData.dark(
    scheme: FlexScheme.purpleBrown,
    surfaceMode: FlexSurfaceMode.highBackgroundLowScaffold,
    blendLevel: 2,
    appBarStyle: FlexAppBarStyle.background,
    bottomAppBarElevation: 2.0,
    subThemesData: const FlexSubThemesData(
        useTextTheme: true,
        useM2StyleDividerInM3: true,
        adaptiveElevationShadowsBack: FlexAdaptive.all(),
        adaptiveAppBarScrollUnderOff: FlexAdaptive.excludeWebAndroidFuchsia(),
        defaultRadius: 10.0,
        defaultRadiusAdaptive: 16.0,
        adaptiveRadius: FlexAdaptive.excludeWebAndroidFuchsia(),
        thinBorderWidth: 3.0,
        thickBorderWidth: 0.5,
        elevatedButtonSchemeColor: SchemeColor.onPrimaryContainer,
        elevatedButtonSecondarySchemeColor: SchemeColor.primaryContainer,

```

outlinedButtonOutlineSchemeColor: SchemeColor.primary,
toggleButtonsBorderSchemeColor: SchemeColor.primary,
segmentedButtonSchemeColor: SchemeColor.primary,
segmentedButtonBorderSchemeColor: SchemeColor.primary,
unselectedToggleIsColored: true,
sliderValueTinted: true,
inputDecoratorSchemeColor: SchemeColor.primary,
inputDecoratorBackgroundAlpha: 22,
inputDecoratorUnfocusedHasBorder: false,
inputDecoratorFocusedBorderWidth: 1.0,
inputDecoratorPrefixIconSchemeColor: SchemeColor.primary,
fabUseShape: true,
fabAlwaysCircular: true,
fabSchemeColor: SchemeColor.tertiary,
cardRadius: 14.0,
popupMenuRadius: 6.0,
popupMenuElevation: 3.0,
dialogRadius: 18.0,
datePickerDialogRadius: 18.0,
timePickerDialogRadius: 18.0,
appBarScrolledUnderElevation: 3.0,
drawerElevation: 1.0,
drawerIndicatorSchemeColor: SchemeColor.primary,
bottomSheetRadius: 18.0,
bottomSheetElevation: 2.0,
bottomSheetModalElevation: 4.0,
bottomNavigationBarMutedUnselectedLabel: false,
bottomNavigationBarMutedUnselectedIcon: false,
menuRadius: 6.0,
menuElevation: 3.0,
menuBarRadius: 0.0,


```

    menuBarElevation: 1.0,
    menuBarShadowColor: Color(0x00000000),
    navigationBarSelectedLabelSchemeColor: SchemeColor.primary,
    navigationBarMutedUnselectedLabel: false,
    navigationBarSelectedIconSchemeColor: SchemeColor.onPrimary,
    navigationBarMutedUnselectedIcon: false,
    navigationBarIndicatorSchemeColor: SchemeColor.primary,
    navigationBarIndicatorOpacity: 1.00,
    navigationBarElevation: 1.0,
    navigationRailSelectedLabelSchemeColor: SchemeColor.primary,
    navigationRailMutedUnselectedLabel: false,
    navigationRailSelectedIconSchemeColor: SchemeColor.onPrimary,
    navigationRailMutedUnselectedIcon: false,
    navigationRailIndicatorSchemeColor: SchemeColor.primary,
    navigationRailIndicatorOpacity: 1.00,
    navigationRailBackgroundSchemeColor: SchemeColor.surface,
  ),
  keyColors: const FlexKeyColors(
    useSecondary: true,
    useTertiary: true,
  ),
  visualDensity: FlexColorScheme.comfortablePlatformDensity,
  useMaterial3: true,

),
themeMode: ThemeMode.system,

    debugShowCheckedModeBanner: false,
    home: getHomePage()
  );
}

```

```

Widget getHomePage() {
  if (kIsWeb) {
    // Return the LandingPage for web
    return LandingPage();
  } else {
    // Return the WelcomeScreen for mobile
    return const WelcomeScreen();
  }
}

}

Get Started page
import 'package:flex_color_scheme/flex_color_scheme.dart';
import 'package:flutter/material.dart';
import 'package:pathsala/screens/login.dart';

class WelcomeScreen extends StatefulWidget {
  const WelcomeScreen({super.key});

  @override
  _WelcomeScreenState createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen> {
  bool _isDarkMode = false;

  void _toggleTheme() {
    setState(() {
      _isDarkMode = !_isDarkMode;

```

```
});
}
```

```
ThemeData _getThemeData(BuildContext context) {
  return _isDarkMode ? FlexThemeData.dark(scheme: FlexScheme.purpleBrown,) :
  FlexThemeData.light(scheme: FlexScheme.purpleBrown,);
}
```

```
@override
```

```
Widget build(BuildContext context) {
  final screenHeight = MediaQuery.of(context).size.height;
  return Theme(
    data: _getThemeData(context),
    child: Scaffold(
      appBar: AppBar(
        title: const Text('PathShala'),
        actions: [
          IconButton(
            icon: Icon(_isDarkMode ? Icons.light_mode : Icons.dark_mode),
            onPressed: _toggleTheme,
          ),
        ],
      ),
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              Container(
                alignment: Alignment.center,
```

```

child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    SizedBox(height: screenHeight*0.1),
    const Text(
      'Welcome to PathShala',
      style: TextStyle(
        fontSize: 30,
        fontWeight: FontWeight.bold,
      ),
    ),
    SizedBox(height: screenHeight*0.1),
    Image.asset(
      'images/background.png',
      fit: BoxFit.fill,
    ),
    SizedBox(height: screenHeight*0.1),
  ],
),

```

```

Container(
  padding: const EdgeInsets.all(24),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      SizedBox(
        width: double.infinity,
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const LoginPage(),
              ),
            );
          },
        ),
      ),
    ],
  ),
),

```

```

        ),
      );
    },
    style: ElevatedButton.styleFrom(

      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(32),
      ),
      padding: const EdgeInsets.symmetric(
        vertical: 16,
      ),
    ),
    child: const Text(
      'Get Started',
      style: TextStyle(fontSize: 16),
    ),
  ),
),
],
),
),
],
),
),
],
),
),
),
),
);
}

```

```
}
```

Login Page

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:pathsala/screens/Admin/Dashboard.dart';
import 'package:pathsala/screens/Parent/Pdashboard.dart';
import 'package:pathsala/screens/Student/Sdashboard.dart';
import 'package:pathsala/screens/Teacher/Tdashboard.dart';
```

```
class LoginPage extends StatefulWidget {
  const LoginPage({Key? key}) : super(key: key);

  @override
  _LoginPageState createState() => _LoginPageState();
}
```

```
class _LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Login'),
      ),
      body: Stack(
        children: [
```

```

Opacity(
  opacity: 0.7,
  child: Container(
    decoration: const BoxDecoration(
      image: DecorationImage(
        image: AssetImage('images/login background.png'),
        fit: BoxFit.cover,
      ),
    ),
  ),
),
Container(
  alignment: Alignment.center,
  child: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        const Text(
          'Welcome Back',
          style: TextStyle(
            fontSize: 24,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(height: 32.0),
        Card(
          elevation: 4.0,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10.0),

```

```

    ),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          TextField(
            controller: _usernameController,
            decoration: const InputDecoration(
              labelText: 'Username',
              border: OutlineInputBorder(),
            ),
          ),
          const SizedBox(height: 16.0),
          TextField(
            controller: _passwordController,
            decoration: const InputDecoration(
              labelText: 'Password',
              border: OutlineInputBorder(),
            ),
            obscureText: true,
          ),
          const SizedBox(height: 16.0),
          ElevatedButton(
            onPressed: _login,
            child: const Text('Login'),
          ),
        ],
      ),
    ),
  ),
],

```



```

        ),
    ),
),
],
),
);
}

```

```

Future<void> _login() async {
  final String enteredUsername = _usernameController.text.trim();
  final String enteredPassword = _passwordController.text.trim();

  try {
    final QuerySnapshot snapshot = await _firestore
      .collection('User')
      .where('username', isEqualTo: enteredUsername)
      .where('password', isEqualTo: enteredPassword)
      .limit(1)
      .get();

    if (snapshot.docs.isNotEmpty) {
      final DocumentSnapshot userSnapshot = snapshot.docs.first;
      final String role = userSnapshot['role'];
      final String userId = userSnapshot.id;

      switch (role) {
        case 'student':
          _navigateToStudentDashboard(userId);
          break;
        case 'parent':
          _navigateToParentDashboard(userId);

```

```

        break;
    case 'teacher':
        _navigateToTeacherDashboard(userId);
        break;
    case 'admin':
        _navigateToAdminDashboard(userId);
        break;
    default:
        _showAlertDialog('Invalid role');
    }
} else {
    _showAlertDialog('Invalid username or password');
}
} catch (e) {
    _showAlertDialog(e.toString());
}
}

void _navigateToStudentDashboard(String userId) {
    Navigator.pushReplacement(
        context,
        MaterialPageRoute(
            builder: (context) => StudentDashboard(studentId: userId),
        ),
    );
}

void _navigateToParentDashboard(String userId) {
    Navigator.pushReplacement(
        context,
        MaterialPageRoute(

```

```

        builder: (context) => ParentHomeScreen(parentId: userId,),
      ),
    );
  }

void _navigateToTeacherDashboard(String userId) {
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(
      builder: (context) => TeacherDashboardScreen(teacherId: userId),
    ),
  );
}

void _navigateToAdminDashboard(String userId) {
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(
      builder: (context) => const DashboardScreen(),
    ),
  );
}

void _showAlertDialog(String message) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Login Error'),
        content: Text(message),
        actions: [

```

```

        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('OK'),
        ),
      ],
    );
  },
);
}
}

```

Student Dashboard

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:pathsala/screens/Student/attendance.dart';
import 'package:pathsala/screens/Student/download.dart';
import 'package:pathsala/screens/Student/notices.dart';
import 'package:pathsala/screens/Student/result.dart';
import 'package:pathsala/screens/Student/timetable.dart';

```

```

FirebaseFirestore db = FirebaseFirestore.instance;

```

```

class StudentData {
  final String name;
  final String contact;
  final String parentsName;
  final String className;
  final String profile;

```

```

StudentData({
    required this.name,
    required this.contact,
    required this.parentsName,
    required this.className,
    required this.profile,
});
}

```

```

class Notice {
    final String title;
    final String message;
    final String postedOn;

```

```

Notice({
    required this.title,
    required this.message,
    required this.postedOn,
});
}

```

```

class StudentDashboard extends StatelessWidget {
    final String studentId;

```

```

StudentDashboard({Key? key, required this.studentId}) : super(key: key);

```

```

Future<StudentData> getStudentData(String studentId) async {
    DocumentSnapshot documentSnapshot =
        await db.collection('Student').doc(studentId).get();

```

```

if (!documentSnapshot.exists) {
    throw Exception('Document does not exist or data is null');
}

Map<String, dynamic>? data =
    documentSnapshot.data() as Map<String, dynamic>?;

if (data == null) {
    throw Exception('Data is null');
}

return StudentData(
    name: data['firstName'] + data['lastName'],
    contact: data['contact'],
    parentsName: data['fatherName'],
    className: data['class'],
    profile: data['profilePictureURL']);
}

Future<Notice> getLatestNotice() async {
    QuerySnapshot querySnapshot = await db
        .collection('Notice')
        .orderBy('postedOn', descending: true)
        .limit(1)
        .get();

    Map<String, dynamic> data =
        querySnapshot.docs.first.data() as Map<String, dynamic>;

    return Notice(
        title: data['title'],
        message: data['message'],

```

```

        postedOn: data['postedOn'],
    );
}

```

```

@override

```

```

Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Student Dashboard'),
      centerTitle: true,
    ),
    body: SafeArea(
      child: SingleChildScrollView(
        child: Column(
          children: [
            const SizedBox(height: 20),
            FutureBuilder<StudentData>(
              future: getStudentData(studentId),
              builder:
                (BuildContext context, AsyncSnapshot<StudentData> snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return const Center(
                      child: CircularProgressIndicator(),
                    );
                  }

                  if (snapshot.hasError) {
                    return Center(
                      child: Text('Error: ${snapshot.error}'),
                    );
                  }
                }
            )
          ],
        ),
      ),
    ),
  );
}

```

```

StudentData studentData = snapshot.data!;

return Container(
  padding: const EdgeInsets.all(20),
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(20),
  ),
  child: Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(20),
        ),
        width: 120,
        height: 120,
        child: CircleAvatar(
          radius: 50,
          backgroundImage: NetworkImage(studentData.profile),
        ),
      ),
      const SizedBox(width: 20),
      Expanded(
        child: SingleChildScrollView(
          scrollDirection: Axis.horizontal,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                children: [

```



```

Text(
  'Name: ${studentData.name}',
  style: const TextStyle(
    fontSize: 18,
    fontWeight: FontWeight.bold,
  ),
),
],
),
const SizedBox(height: 10),
Row(
  children: [
    const Text(
      'Contact:',
      style: TextStyle(fontSize: 16),
    ),
    Text(
      studentData.contact,
      style: const TextStyle(fontSize: 16),
    ),
  ],
),
const SizedBox(height: 10),
Text(
  studentData.className,
  style: const TextStyle(fontSize: 16),
),
const SizedBox(height: 10),
Row(
  children: [
    const Text(

```

```

        "Parent's Name:",
        style: TextStyle(fontSize: 16),
      ),
      Text(
        studentData.parentsName,
        style: const TextStyle(fontSize: 16),
      ),
    ],
  ),
],
),
),
),
],
),
);
},
),
const SizedBox(height: 20),
FutureBuilder<Notice>(
  future: getLatestNotice(),
  builder:
    (BuildContext context, AsyncSnapshot<Notice> snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }

    if (snapshot.hasError) {
      return Center(

```

```

        child: Text('Error: ${snapshot.error}'),
      );
    }

```

```

Notice notice = snapshot.data!;

```

```

return Container(
  padding: const EdgeInsets.all(20),
  decoration: BoxDecoration(
    border: Border.all(),
    borderRadius: BorderRadius.circular(20),
  ),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Row(
        children: [
          const Text(
            'Notice',
            style: TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
          TextButton(
            child: const Text(
              'view All',
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
              ),
            ),

```

```

    ),
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => NoticeScreen(),
        ),
      );
    },
  ),
],
),
const SizedBox(height: 10),
Text(
  notice.title,
  style: const TextStyle(
    fontSize: 18,
    fontWeight: FontWeight.bold,
  ),
),
Text(
  notice.message,
  style: const TextStyle(
    fontSize: 16,
  ),
),
Text(
  'posted on:-${notice.postedOn}',
  style: const TextStyle(
    color: Colors.grey,
    fontSize: 14,

```

```

        ),
      ),
    ],
  ),
);
},
),
const SizedBox(height: 20),
Container(
  padding: const EdgeInsets.all(20),
  decoration: BoxDecoration(
    border: Border.all(),
    borderRadius: BorderRadius.circular(20),
  ),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      const Text(
        'Quick Links',
        style: TextStyle(
          fontSize: 18, fontWeight: FontWeight.bold),
      ),
      const SizedBox(height: 10),
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Expanded(
            child: InkWell(
              onTap: () {
                Navigator.push(
                  context,

```

```

MaterialPageRoute(
  builder: (context) =>
    MonthlyAttendanceLogScreen(
      studentName: 'Apil Chand',
    ),
  ),
);
},
child: Container(
  padding: const EdgeInsets.all(20),
  decoration: BoxDecoration(
    border: Border.all(),
    borderRadius: BorderRadius.circular(20),
  ),
  child: const Column(
    children: [
      Icon(
        Icons.calendar_today,
        size: 50,
      ),
      SizedBox(height: 10),
      Text(
        'Attendance',
        style: TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.bold,
        ),
      ),
    ],
  ),
),
),

```

```

    ),
  ),
  const SizedBox(width: 20),
  Expanded(
    child: InkWell(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => const Examresult(),
          ),
        );
      },
      child: Container(
        padding: const EdgeInsets.all(20),
        decoration: BoxDecoration(
          border: Border.all(),
          borderRadius: BorderRadius.circular(20),
        ),
        child: const Column(
          children: [
            Icon(
              Icons.assignment,
              size: 50,
            ),
            SizedBox(height: 10),
            Text(
              'Result',
              style: TextStyle(
                fontSize: 16,
                fontWeight: FontWeight.bold,

```

```

        ),
        ),
    ],
    ),
    ),
    ),
    ),
    ],
),
const SizedBox(height: 20),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Expanded(
      child: InkWell(
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => TimetableScreen(),
            ),
          );
        },
        child: Container(
          padding: const EdgeInsets.all(20),
          decoration: BoxDecoration(
            border: Border.all(),
            borderRadius: BorderRadius.circular(20),
          ),
          child: const Column(
            children: [

```



```

        Icon(
          Icons.schedule,
          size: 50,
        ),
        SizedBox(height: 10),
        Text(
          'Timetable',
          style: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.bold,
          ),
        ),
      ],
    ),
  ),
),
const SizedBox(width: 20),
Expanded(
  child: InkWell(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) =>
            ResourceDownloadPage(),
        ),
      );
    },
    child: Container(
      padding: const EdgeInsets.all(20),

```

```
decoration: BoxDecoration(
  border: Border.all(),
  borderRadius: BorderRadius.circular(20),
),
child: const Column(
  children: [
    Icon(
      Icons.file_download,
      size: 50,
    ),
    SizedBox(height: 10),
    Text(
      'Downloads',
      style: TextStyle(
        fontSize: 16,
        fontWeight: FontWeight.bold,
      ),
    ),
  ],
),
],
),
],
),
],
),
],
),
```

```

    ),
  );
}
}

```

Teacher Dashboard

```

import 'package:flutter/material.dart';
import '../about.dart';
import 'Class_grid.dart';
import 'upload.dart';
import 'Teacher_Timetable.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

```

```

FirebaseFirestore db = FirebaseFirestore.instance;

```

```

class TeacherData{
  final String name;
  final String contact;
  final String email;
  final String gender;
  final String profile;
  TeacherData(
    {
      required this.name,
      required this.contact,
      required this.email,
      required this.gender,
      required this.profile
    }
  )
}

```

```

);
}

class TeacherDashboardScreen extends StatelessWidget {
  final String teacherId;

  const TeacherDashboardScreen({ Key? key, required this.teacherId }) : super(key: key);

  Future<TeacherData> getTeacherData(String teacherId) async {
    DocumentSnapshot snapshot =
      await db.collection('Teacher').doc(teacherId).get();
    Map<String, dynamic> data = snapshot.data() as Map<String, dynamic>;

    return TeacherData(
      name: data['firstName']+' '+data['lastName'],
      contact: data['contact'],
      email: data['email'],
      gender: data['gender'],
      profile: data['profilePictureURL']
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Center(child: Text('Teacher Panel')),
      ),

      body:

```

```

FutureBuilder<TeacherData>(
  future: getTeacherData(teacherId),
  builder: (BuildContext context,
    AsyncSnapshot<TeacherData> snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }

    if (snapshot.hasError) {
      return Center(
        child: Text('Error: ${snapshot.error}'),
      );
    }

    TeacherData teacherData = snapshot.data!;

    return Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        Padding(
          padding: const EdgeInsets.all(16.0),
          child: Text(
            'Welcome,${teacherData.name}!',
            style: const TextStyle(
              fontSize: 24.0,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
      ],
    ),
    Container(

```

```

padding: const EdgeInsets.all(20),
decoration: BoxDecoration(

  borderRadius: BorderRadius.circular(20),
  border: Border.all()
),
child: Row(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(20),

      ),
      width: 130,
      height: 130,
      child: CircleAvatar(
        radius: 50,
        backgroundImage: NetworkImage(teacherData.profile),
      ),
    ),
    const SizedBox(width: 30),
    Expanded(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Text(
            "Name: ${teacherData.name}",
            style: const TextStyle(fontWeight: FontWeight.bold),
          ),

```

```

        const SizedBox(height: 10),
        Text(
          "Gender: ${teacherData.gender}",
        ),
        const SizedBox(height: 10),
        Text(
          "Contact: ${teacherData.contact}",
        ),
        const SizedBox(height: 10),
        Text(
          "Email: ${teacherData.email}",
        ),

      ],
    ),
  ),
],
),
),
const SizedBox(
  height: 10,
),
Expanded(
  child: Padding(
    padding: const EdgeInsets.all(24.0),
    child: GridView.count(
      crossAxisCount: 2,
      crossAxisSpacing: 16.0,
      mainAxisSpacing: 16.0,
      shrinkWrap: true,

```

```

children: [
  _buildItem(
    context,
    'Manage Classes',
    Icons.class_,
    () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => ClassScreen(),
        ),
      );
    },
  ),
  _buildItem(
    context,
    'Timetable',
    Icons.calendar_today,
    () {
      {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => TeacherTimeTableScreen(
              teacherName: teacherData.name,
            ),
          ),
        );
      }
    },
  ),

```



```

        _buildItem(
          context,
          'Upload',
          Icons.file_open,
        ) {
          {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const UploadContentScreen(),
              ),
            );
          }
        },
      ),
      _buildItem(
        context,
        'About',
        Icons.adobe_outlined,
      ) {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => AboutScreen(),
          ),
        );
      },
    ),
  ],
),
),

```

```

        ),
      ],
    );
  }
)
);
}

```

```

Widget _buildItem(BuildContext context, String title, IconData iconData,
  VoidCallback onTap) {
  return InkWell(
    onTap: onTap,
    child: Card(
      elevation: 2.0,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(
            iconData,
            size: 64.0,
            //color: Colors.purple,
          ),
          const SizedBox(height: 8.0),
          Text(
            title,
            style: const TextStyle(fontSize: 18.0, fontWeight: FontWeight.bold),
            textAlign: TextAlign.center,
          ),
        ],
      ),
    ),
  );
}

```

```

    ),
  );
}
}

```

Parent Dashboard

```

import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

import '../Student/Attendance.dart';
import '../Student/Download.dart';
import '../Student/result.dart';
import '../Student/timetable.dart';

class ParentHomeScreen extends StatelessWidget {
  final String parentId;
  const ParentHomeScreen({ Key? key, required this.parentId }) : super(key: key);

  Future<String> fetchStudentName() async {
    final studentSnapshot = await FirebaseFirestore.instance
      .collection('Student')
      .where('parentId', isEqualTo: parentId)
      .limit(1)
      .get();

    final studentData = studentSnapshot.docs.first.data();
    return studentData['firstName']+' '+studentData['lastName'];
  }
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Center(child: Text('Parent')),
    ),

    body: SafeArea(
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Icon(
                  Icons.menu,

                ),
                CircleAvatar(
                  radius: 25.0,
                  backgroundImage: AssetImage('images/picture.jpg'),
                ),
              ],
            ),
            const SizedBox(height: 16.0),
            const Text(
              'Hello',
              style: TextStyle(

```

```

        fontSize: 28.0,
        fontWeight: FontWeight.bold,

    ),
),
const SizedBox(height: 8.0),
const Text(
    'Here is what's happening in your school today:',
    style: TextStyle(
        fontSize: 18.0,

    ),
),
const SizedBox(height: 32.0),
Expanded(
    child: SingleChildScrollView(
        child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
                FutureBuilder<String>(
                    future: fetchStudentName(),
                    builder: (context, snapshot) {
                        if (snapshot.connectionState == ConnectionState.waiting) {
                            return CircularProgressIndicator();
                        }
                        if (snapshot.hasError) {
                            return Text('Error: ${snapshot.error}');
                        }
                        final studentName = snapshot.data ?? 'Unknown';
                        return HomeScreenCard(
                            title: 'Attendance',

```

```

        subtitle: 'Check if your child is present or absent',
        icon: Icons.person_outline,
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => MonthlyAttendanceLogScreen(studentName:
studentName),
            ),
          );
        },
      );
    },
  ),
  const SizedBox(height: 16.0),
  HomeScreenCard(
    title: 'Assignments',
    subtitle: 'View assignments',
    icon: Icons.assignment_outlined,
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => const ResourceDownloadPage(),
        ),
      );
    },
  ),
  const SizedBox(height: 16.0),
  HomeScreenCard(
    title: 'Events',

```

```

        subtitle: 'View upcoming events and activities',
        icon: Icons.event_outlined,
        onPressed: () {
          // Handle events onPressed
        },
      ),
      const SizedBox(height: 16.0),
      HomeScreenCard(
        title: 'Exams',
        subtitle: 'View exam schedules and results',
        icon: Icons.library_books_outlined,
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => const Examresult(),
            ),
          );
        },
      ),
      const SizedBox(height: 16.0),
      HomeScreenCard(
        title: 'Timetable',
        subtitle: 'View class timetables and schedules',
        icon: Icons.access_time_outlined,
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => TimetableScreen(),
            ),
          );
        },
      ),
    ),
  ),
);

```

```

        );
    },
),
const SizedBox(height: 16.0),
],
),
),
),
],
),
),
),
);
}
}

```

```

class HomeScreenCard extends StatelessWidget {

```

```

    final String title;
    final String subtitle;
    final IconData icon;
    final VoidCallback onPressed;

```

```

    const HomeScreenCard({
        Key? key,
        required this.title,
        required this.subtitle,
        required this.icon,
        required this.onPressed,
    }) : super(key: key);

```

```

    @override

```



```

Widget build(BuildContext context) {
  return Material(
    borderRadius: BorderRadius.circular(8.0),
    child: InkWell(
      onTap: onPressed,
      child: Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(8.0),
          border: Border.all()

        ),
        padding: const EdgeInsets.all(16.0),
        child: Row(
          children: [
            Icon(
              icon,
              size: 40.0,
            ),
            const SizedBox(width: 16.0),
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    title,
                    style: const TextStyle(
                      fontSize: 18.0,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  const SizedBox(height: 4.0),

```

```

        Text(
          subtitle,
          style: TextStyle(
            fontSize: 14.0,
            color: Colors.grey[600],
          ),
        ),
      ],
    ),
  ],
),
),
),
);
}
}

```

~~~~~

### **Admin Dashboard**

```

import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:pathsala/screens/Admin/Student_management/student_management.dart';
import 'package:pathsala/screens/Admin/Teacher_management/teacher_management.dart';
import 'package:pathsala/screens/Admin/class_management/class_management.dart';
import 'package:pathsala/screens/Admin/result_management.dart';
import 'package:pathsala/screens/Teacher/Notice.dart';
import 'package:pathsala/screens/login.dart';

class DashboardScreen extends StatefulWidget {
  const DashboardScreen({Key? key}) : super(key: key);

```

```

@override
_DashboardScreenState createState() => _DashboardScreenState();
}

```

```

class _DashboardScreenState extends State<DashboardScreen> {
  int studentCount = 0;
  int teacherCount = 0;

```

```

@override
void initState() {
  super.initState();
  fetchCounts();
}

```

```

Future<void> fetchCounts() async {
  final QuerySnapshot studentSnapshot = await
  FirebaseFirestore.instance.collection('Student').get();

  final QuerySnapshot teacherSnapshot = await
  FirebaseFirestore.instance.collection('Teacher').get();

```

```

setState() {
  studentCount = studentSnapshot.size;
  teacherCount = teacherSnapshot.size;
});
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Center(child: Text('Dashboard')),

```

```

),
drawer: Drawer(
  child: ListView(
    children: [
      Container(
        height: 120.0,
        child: const DrawerHeader(
          decoration: BoxDecoration(),
          child: Center(
            child: Text(
              'School Management System',
              style: TextStyle(
                fontSize: 24.0,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
        ),
      ),
      ),
      ),
      ),
      ),
      ),
      ),
      DrawerItem(
        icon: Icons.people,
        title: 'Students',
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => StudentManagementScreen()),
          );
        },
      ),
      DrawerItem(
        icon: Icons.person,

```

```

        title: 'Teachers',
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => const TeacherManagementScreen()),
          );
        },
      ),
      DrawerItem(
        icon: Icons.class_,
        title: 'Classes',
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => Classmgmt()),
          );
        },
      ),
      DrawerItem(
        icon: Icons.grade,
        title: 'Grades',
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => const StudentMarks()),
          );
        },
      ),
      DrawerItem(
        icon: Icons.notifications,
        title: 'Notices',

```

```

onTap: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => const TeacherNoticeScreen()),
  );
},
),
DrawerItem(
  icon: Icons.logout,
  title: 'Logout',
  onTap: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => const LoginPage()),
    );
  },
),
],
),
),
body: Padding(
  padding: const EdgeInsets.all(16.0),
  child: GridView.count(
    crossAxisCount: 2,
    padding: const EdgeInsets.all(16.0),
    childAspectRatio: 0.75,
    mainAxisSpacing: 16.0,
    crossAxisSpacing: 16.0,
    children: [
      DashboardCard(
        title: 'Students',

```

```

        count: studentCount,
        icon: Icons.people,
    ),
    DashboardCard(
        title: 'Teachers',
        count: teacherCount,
        icon: Icons.person,
    ),
    DashboardCard(
        title: 'Classes',
        count: 10,
        icon: Icons.class_,
    ),
    DashboardCard(
        title: 'Attendance',
        count: 85,
        icon: Icons.calendar_today,
    ),
    DashboardCard(
        title: 'Pass percentage',
        count: 80,
        icon: Icons.grade,
    ),
  ],
),
);
}
}

```

```

class DrawerItem extends StatelessWidget {

```

```

const DrawerItem({
  Key? key,
  required this.icon,
  required this.title,
  required this.onTap,
}) : super(key: key);

final IconData icon;
final String title;
final VoidCallback onTap;

@override
Widget build(BuildContext context) {
  return ListTile(
    leading: Icon(
      icon,
    ),
    title: Text(
      title,
      style: const TextStyle(
        fontSize: 18.0,
        fontWeight: FontWeight.bold,
      ),
    ),
    onTap: onTap,
  );
}
}

class DashboardCard extends StatelessWidget {
  const DashboardCard({

```



```
Key? key,  
required this.title,  
required this.count,  
required this.icon,  
}) : super(key: key);
```

```
final String title;  
final int count;  
final IconData icon;
```

```
@override
```

```
Widget build(BuildContext context) {  
  return Container(  
    margin: const EdgeInsets.all(8.0),  
    decoration: BoxDecoration(  
      borderRadius: BorderRadius.circular(10.0),  
      boxShadow: const [  
        BoxShadow(  
          offset: Offset(0.0, 1.0),  
          blurRadius: 6.0,  
        )  
      ],  
      color: Theme.of(context).cardColor,  
    ),  
    child: Column(  
      mainAxisAlignment: MainAxisAlignment.center,  
      children: [  
        Icon(  
          icon,  
          size: 60.0,  
          color: Theme.of(context).iconTheme.color,
```

```

    ),
    const SizedBox(height: 10.0),
    Text(
      title,
      style: const TextStyle(
        fontSize: 20.0,
        fontWeight: FontWeight.bold,
      ),
    ),
    const SizedBox(height: 10.0),
    Text(
      count.toString(),
      style: const TextStyle(
        fontSize: 18.0,
        fontWeight: FontWeight.w600,
      ),
    ),
  ],
),
);
}
}

```