



CENTRALIZED DATABASE SYSTEM TO MANAGE STUDENT INFORMATION



A MINI PROJECT REPORT

Submitted by

SHAHUL HAMEED MANSOOR K	1921044
SIVA SUBRAMANIAN S	1921045
GOWTHAM S	2021T303

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING, SALEM

(An Autonomous Institution Affiliated to Anna University Chennai, NAAC Accredited)

ANNA UNIVERSITY: CHENNAI - 600 025

JUNE 2022

GOVERNMENT COLLEGE OF ENGINEERING, SALEM

(An Autonomous Institution Affiliated to Anna University Chennai, NAAC Accredited)

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this mini project report “**CENTRALIZED DATABASE SYSTEM TO MANAGE STUDENT INFORMATION**” is the bonafide work of “**SHAHUL HAMEED MANSOOR K (1921044), SIVA SUBRAMANIAN S (1921045), GOWTHAM S (2021T303)**” who carried out the mini project work under my supervision during the academic year 2021-2022.

SIGNATURE

**Dr. A. M. KALPANA M.E., Ph.D.
PROFESSOR &
HEAD OF THE DEPARTMENT**

Computer Science and Engineering,
Government College of Engineering,
Salem-11.

SIGNATURE

**Tmt. P. THARANI M.E.,
ASSISTANT PROFESSOR &
SUPERVISOR**

Computer Science and Engineering
Government College of Engineering,
Salem-11.

Submitted for the mini project viva-voice examination held at the Government College of Engineering, Salem-11 on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First of all we extend our heart-felt gratitude to the almighty for providing us with enough strength, courage and ideas for successful completion of the mini project. Behind every achievement lies an unfathomable sea of gratitude to those who are behind it.

We convey our heartfelt gratitude to the honorable and respected principal **Dr. R. MALAYALAMURTHI M.E., Ph.D.** for his encouragement and support for the successful completion of the project.

We are ineffably indebted to the Head of the Department, Computer Science and Engineering, **Dr. A. M. KALPANA, M. E., Ph. D.** who took keen interest till the completion of our mini project work by providing all the necessary information for developing a good system.

We are thankful to our mini project guide and motivative coordinator **Tmt. P.THARANI, M.E.,** Assistant Professor, Computer Science and Engineering for her remarkable guidance and the incessant help in all possible ways from the beginning to accomplish the project successfully.

We are highly grateful to our Faculty Members of our department for their valuable suggestions and immediate counseling throughout the completion of the mini project. It would not have been possible without the kind support and help of many individuals. We also extend our gratitude to the non-teaching faculty for their timely support.

We also acknowledge with a deep sense of reverence and gratitude towards our parents, family members and friends who supported us for the successful completion of the mini project.

ABSTRACT

In education Institutions, most of the work details are done by web applications. But for the students, information is collected through hard copy. Collecting separate files and being maintained by class advisors and putting documents safe is inconvenient. To handle this process, there is a way to collect, store and access details through an online database system.

This project “CENTRALIZED DATABASE SYSTEM TO MANAGE STUDENT INFORMATION” is necessary to record all student’s data for finding potential gaps for efficient decision making, carrying out daily activities, thus saving a great deal of time. It also minimizes the chances of human error. The best thing is that it can be accessed 24/7. This web service will offer better communication, faster attendance management and streamlined activities to the respective classrooms.

This service is designed to make the management of student data such as storage, tracking, and monitoring information simple and easy. It also enables college authorities to share relevant information with students, teachers, and parents. It makes the process of student enrollment quick, systematic, and error-free. As the entire data is saved at a central location, typically in the cloud, and role-based access to data is granted to each faculty member, the student data remains secure.

It provides the teaching faculty with summaries on time and information as regards regular latecomers, absentees, etc. The events calendar keeps the parents informed as regards the various activities and events that the department organizes. The best aspect of employing cloud computing technology for the management of student databases is that it ensures the safety and security of college and student data.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF SYMBOLS	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 Centralized Database System to Manage Student Information	1
2	SYSTEM ANALYSIS	2
	2.1 EXISTING SYSTEM	2
	2.2 PROPOSED SYSTEM	2
	2.3 FEASIBILITY STUDY	3
	2.3.1 Economical feasibility	3
	2.3.2 Technical feasibility	4
	2.3.2 Operational feasibility	4
3	SYSTEM SPECIFICATION	5
	3.1 HARDWARE REQUIREMENTS	5
	3.2 SOFTWARE REQUIREMENTS	5
4	SOFTWARE DESCRIPTION	6
	4.1 FRONT END	6
	4.1.1 Front end definition	6
	4.1.2 HTML	6
	4.1.3 CSS	7

	4.1.4 Javascript	7
	4.2 FRAMEWORK	8
	4.2.1 React	8
	4.3 BACKEND	8
	4.3.1 Firebase (Database)	8
	4.4 PLATFORM	9
	4.4.1 Visual studio code	9
5	SYSTEM DESIGN	10
	5.1 DEFINITON	10
	5.2 ARCHITECTURE	10
	5.3 USE CASE DIAGRAM	13
	5.3.1 Use case – student	14
	5.3.2 Use case – faculty	14
	5.3.3 Use case – admin	16
	5.4 DATA FLOW DIAGRAM	16
	5.4.1 Context level – DFD	17
	5.4.2 Student – DFD	18
	5.4.3 Admin – DFD	19
	5.4.4 Faculty – DFD	20
6	MODULES	21
	6.1 HOME PAGE	21
	6.2 LOGIN MODULE	22
	6.3 ABOUT MODULE	23
	6.4 NEWS MODULE	23
	6.5 STUDENT DASHBOARD	23
	6.6 FACULTY MODULE	24
	6.7 ATTENDANCE MODULE	25
	6.8 ADMIN DASHBOARD	25

7	SYSTEM TESTING	27
	7.1 INTRODUCTION	27
	7.2 TEST SCENARIOS	27
	7.3 APPROPRIATE TESTS	28
	7.3.1 Load Testing	28
	7.3.2 Functional Testing	28
	7.3.3 Usability Testing	29
	7.4 TEST SCENARIOS ON MODULES	29
8	8.1 CONCLUSION	32
	8.2 FUTURE ENHANCEMENT	33
	APPENDIX -1 SOURCE CODE	34
	APPENDIX -2 SCREENSHOTS	56
	REFERENCES	61


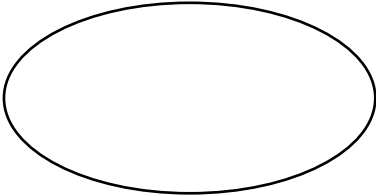
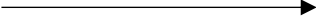

LIST OF TABLES

TABLE NO	TITLE	PAGE NO.
6.1	Login module	22
6.2	News page	23
6.3	Biodata module	24
6.4	Faculty module	24
6.5	Attendance module	25

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO.
4.1	HTML, CSS, JAVASCRIPT	7
4.2	Firebase in backend	8
4.3	VS code context view	9
5.1	Overview of Architecture	11
5.2	Detailed Architecture	12
5.3	Use case diagram	15
5.4	Data flow diagram	17
5.5	Student - DFD	18
5.6	Admin - DFD	19
5.7	Faculty - DFD	20
6.1	Modules Overview	21
6.2	Database Schema	26
7.1	Modules	29
7.2	Integration testing	29
7.3	End-to-end testing	30

LIST OF SYMBOLS

SYMBOLS	DESCRIPTION
	External Entity
	Process
	Data flow
	Data store

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheets
DFD	Data Flow Diagram
DOM	Document Object Model
HDD	Hard Disk Drive
HTML	Hyper Text Markup Language
LCD	Liquid Crystal Display
MVC	Model View Controller
OS	Operating System
RAM	Random Access Memory
SDK	Software Development Toolkit
UI	User Interface
VS CODE	Visual Studio Code
WWW	World Wide Web

CHAPTER 1

INTRODUCTION

1.1 Centralized Database System to Manage Student Information

Centralized Database System to Manage Student Information is a web-based project. The aim is to design a website for managing student information and manipulating attendance. This project will make an efficient communication between faculty and student.

In education Institutions, collecting separate files and being maintained by class advisors and putting documents safe is inconvenient. To handle this process, there is a way to collect, store and access details through an online database system. This project is used to store, update, retrieve, process the data and also used to collect and update student's attendance. In this system there is an effective communication between faculty and students.

This project is going to help

- To access student information through dynamic web application
- To maintain student data through database management.
- To update student's attendance information.
- To generate attendance report.
- To share student information to the need.

Faculty can see all the information of students and students can see their own information. In this project there is an admin role to control over all online website and data likewise in an institution admin controlling physical files. This project saves entire data at a central location, typically in cloud and has role-based access to data.

CHAPTER 2

SYSTEM ANALYSIS

Systems analysis is "the process of studying a procedure or business to identify project goal and purposes and create systems and procedures that will efficiently achieve them".

2.1 EXISTING SYSTEM

- In existing system there is shown only student's profile related details in websites.
- The existing technology provides a similar user interface but is not effective to store, retrieve, and edit/update information available in the database on an on-demand basis.
- There is as far rare communication between students and teachers.
- There is no proper admin Dashboard.
- The existing system doesn't give access to different type users.
- There is separate existing system for each module, there is no combined system for all modules.

2.2 PROPOSED SYSTEM

Our proposed system will offer better communication, faster attendance management and streamlined activities to the respective classrooms. There will be an effective communication between student's and staffs.

The proposed system combines all modules into one for different types of users. The proposed gives report generation for attendance. The development of his new system can give access to see store and manage student information in online.

The system has different access levels for users from security point of view. Creation of report will be efficient. Staffs Can intimate students about events, exams and etc...

Admin has proper dashboard. Through that admin dashboard, the admin can add/delete student profile, classes for Staffs. With this system staff store their subject attendance in database. They manage that stored attendance details on any time. The students can intimate teachers before they going to absent.

2.3 FEASIBILITY STUDY

Feasibility is the determination of whether or not a project is worth doing. The processor that is followed in making this determination is called a Feasibility Study. Feasibility study is the test of system proposal according to its workability impact on the organization ability to meet user's needs, and effective use of resources. The result of feasibility is a formal proposal. This is simply a report a formal document detailing the nature and scope of the proposed solution.

It describes a preliminary study undertaken to determine and document a project's viability. The results of this analysis are used in making the decision whether to proceed with the project or not. Three key considerations are involved in the feasibility analysis,

- Economic feasibility
- Technical feasibility
- Operational feasibility

2.3.1 Economical Feasibility

In the economic side, it is generally the bottom-line consideration of the project. It will increase the efficiency and decrease man-hour to achieve the result. Economic analysis could also be referred to as benefit analysis. In economic analysis the procedure is to determine the benefits and savings that are expected from a

candidate system and compare them with cost. It will provide timely and up to date information to the administrative and individual departments. Since all the information is available in a few seconds the system performance will be substantially increased.

2.3.2 Technical Feasibility

In the technical side, it is the most difficult area to access because objectives, functions performance are somewhat hazy; anything seems to be possible if right assumptions are made. A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The considerations that are normally associated with technical include development risk, technology and resource availability.

2.3.3 Operational Feasibility

Operational feasibility is to gain an understanding of whether the proposed system will likely solve the business problems, or take advantage of the opportunities or not. Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis in the phase of system development. Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the feasibilities to gauge. In order to determine these feasibilities, it is important to understand the management commitment to the proposed project. It is important to understand how the new systems will fit into the current day-to-day operations of the organization.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE REQUIREMENTS

The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and requirements are also important.

- Processor: Intel i3 8th Gen processor
- RAM: 4GB or 8GB
- Hard Disk: 500 GB HDD
- Input Device: Standard keyboard
- Output Device: LCD Monitor

3.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7 or above
- Front End : HTML, CSS, JAVASCRIPT
- Framework : REACT
- Back End : Firebase
- Platform : Visual Studio Code

CHAPTER 4

SOFTWARE DESCRIPTION

4.1 FRONT END

4.1.1 Front End Definition

The layer above the back end is the front end and it includes all software or hardware that is part of a user interface. Human or digital users interact directly with various aspects of the front end of a program, including user-entered data, buttons, programs, websites and other features.

4.1.2 HTML

HTML (Hypertext Markup Language) is used to create documents on the World Wide Web. It is a platform independent language that can be used on any platform such as windows, Linux, Macintosh, and so on. To display a document on the web it is essential to mark-up the different elements (headings, paragraph, tables, and so on) of the document with the HTML tags.

To view a mark-up document, the user has to open the document in a browser. A browser understands and interprets the HTML tags, identifies the structure of the document (which parts are which) and makes decisions about presentation(how the parts look) of the document. HTML also provides tags to make the document look attractive using graphics, font size and colors. Users can make a link to the other document or the different section of the same document by creating Hypertext Links also known as Hyperlinks.

- To create, save and view a HTML document.
- To describe ordered and unordered lists.
- To explain graphics in HTML document.
- To describe hypertext links and making text/image links.

4.1.3 CSS

Cascading Style Sheets is a stylesheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

- The layout of a web page is better controlled.
- Style (CSS) kept separate from structure (HTML), meaning smaller file size.
- Reduced file size means reduced bandwidth, which means faster loading time.



Figure 4.1 HTML, CSS, JAVASCRIPT

4.1.4 JAVASCRIPT

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.

JavaScript runs on the client side of the web, which can be used to design program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behavior.

4.2 FRAMEWORK

4.2.1 REACT

React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM, offering a simpler programming model and better performance.

4.3 BACKEND

4.3.1 Firebase (database)

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data.

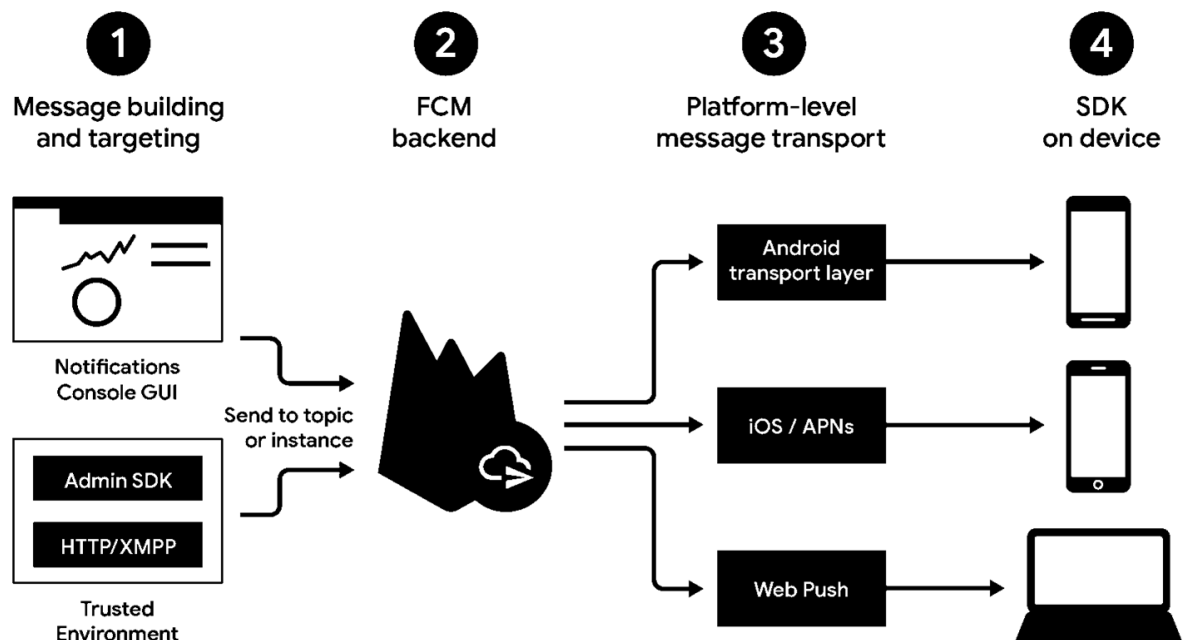


Figure 4.2 Firebase in backend

Features of Firebase: This feature mainly includes backend services that help developers to build and manage their applications in a better way. Services included under this feature are :

- **Cloud Firestore:** The cloud Firestore is a NoSQL document database that provides services like store, sync, and query through the application on a global scale. It stores data in the form of objects also known as Documents
- **Authentication:** Firebase Authentication service provides easy to use UI libraries and SDKs to authenticate users to the app. It reduces the manpower and effort required to develop and maintain the user authentication service.

4.4 PLATFORM

4.4.1 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on the desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity). VS Code extensions let us add languages, debuggers, and tools for installation to support web development workflow.

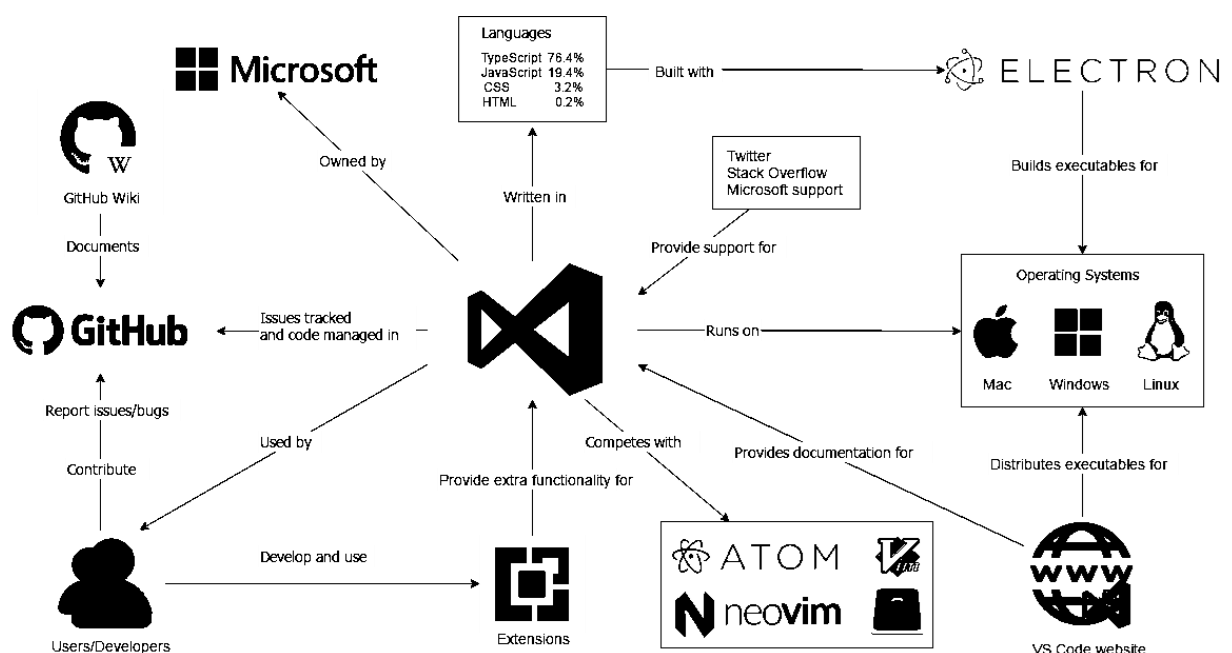


Figure 4.3 Visual studio code context view

CHAPTER 5

SYSTEM DESIGN

5.1 DEFINITION

System design is the process of defining the elements of a system such as architecture, modules and components, the different interfaces of those components and the data that goes through that system. A computer system consists of hardware components that have been carefully chosen so that they work well together and software components or programs that run in the computer.

5.2 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components. An architecture diagram is a visual representation of all the elements that make up part, or all, of a system. It helps the engineers, designers, stakeholders and anyone else involved in the project to understand a system or app's layout.

The diagram as being a bit like a blueprint to a building: can see the thing as a whole, as well as different kinds of interior views, and things like pipes, walls, floorplans, and so on. Architectural diagrams do the same: they show the whole system, as well as its various components.

The benefits of using software architecture diagrams are:

- Diagrams make it easier to take in information. They also help with comprehension and recall.

- **Aid understanding:** Diagrams offer a top-down view of a process or system, which makes it easier to get the gist of something at a glance. It also shows how each component interacts both with other components and as part of a bigger system of something that makes it easier for teams to collaborate more effectively. It also makes it easier for teams to see how the introduction of new features or jobs will affect other elements of the system.
- **Improve collaboration:** Being able to understand how processes and features interact means it's easier to spot weak points, bottlenecks, or other issues when everyone is working together efficiently, collaboration becomes a joint effort.

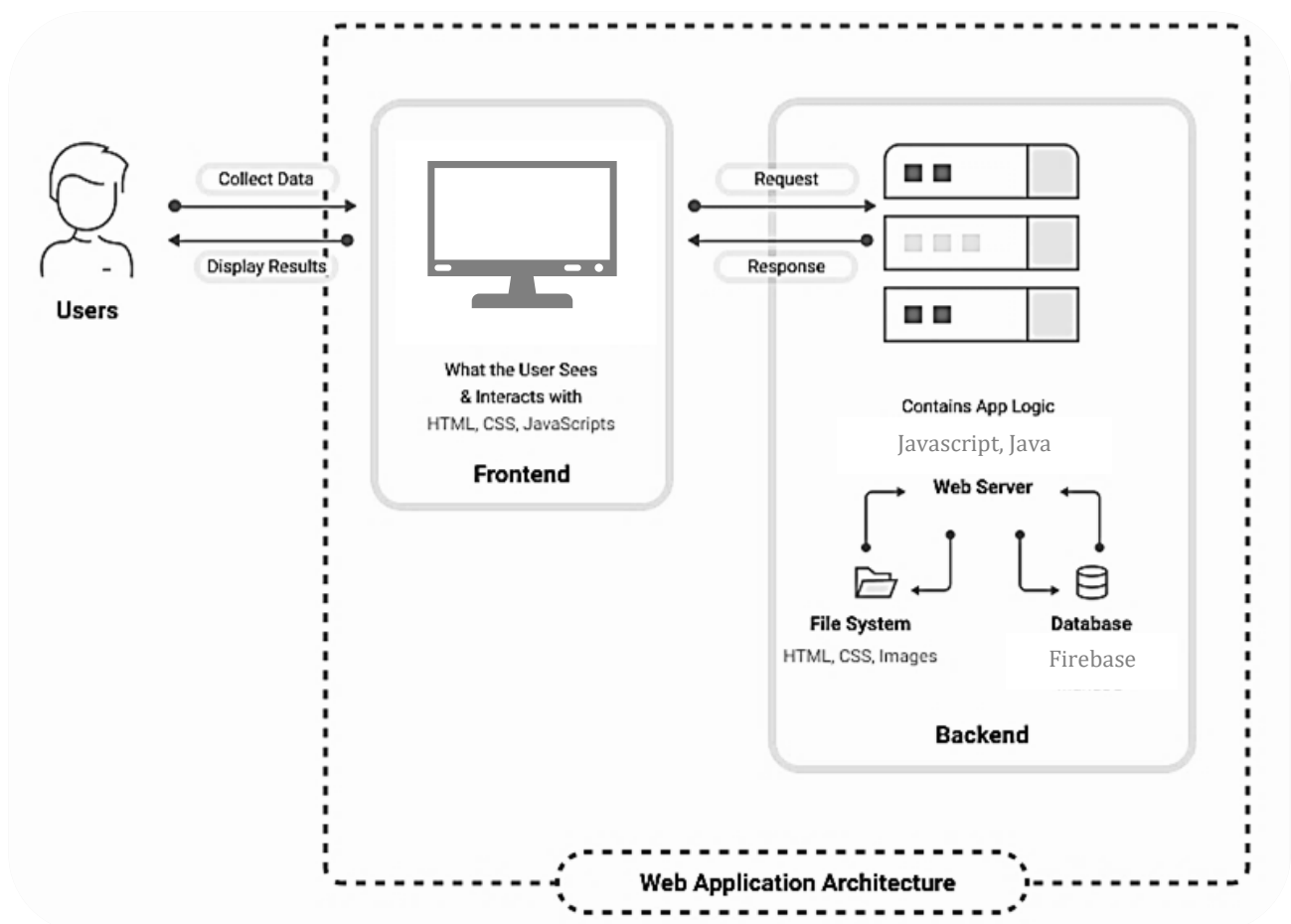


Figure 5.1 Overview of architecture

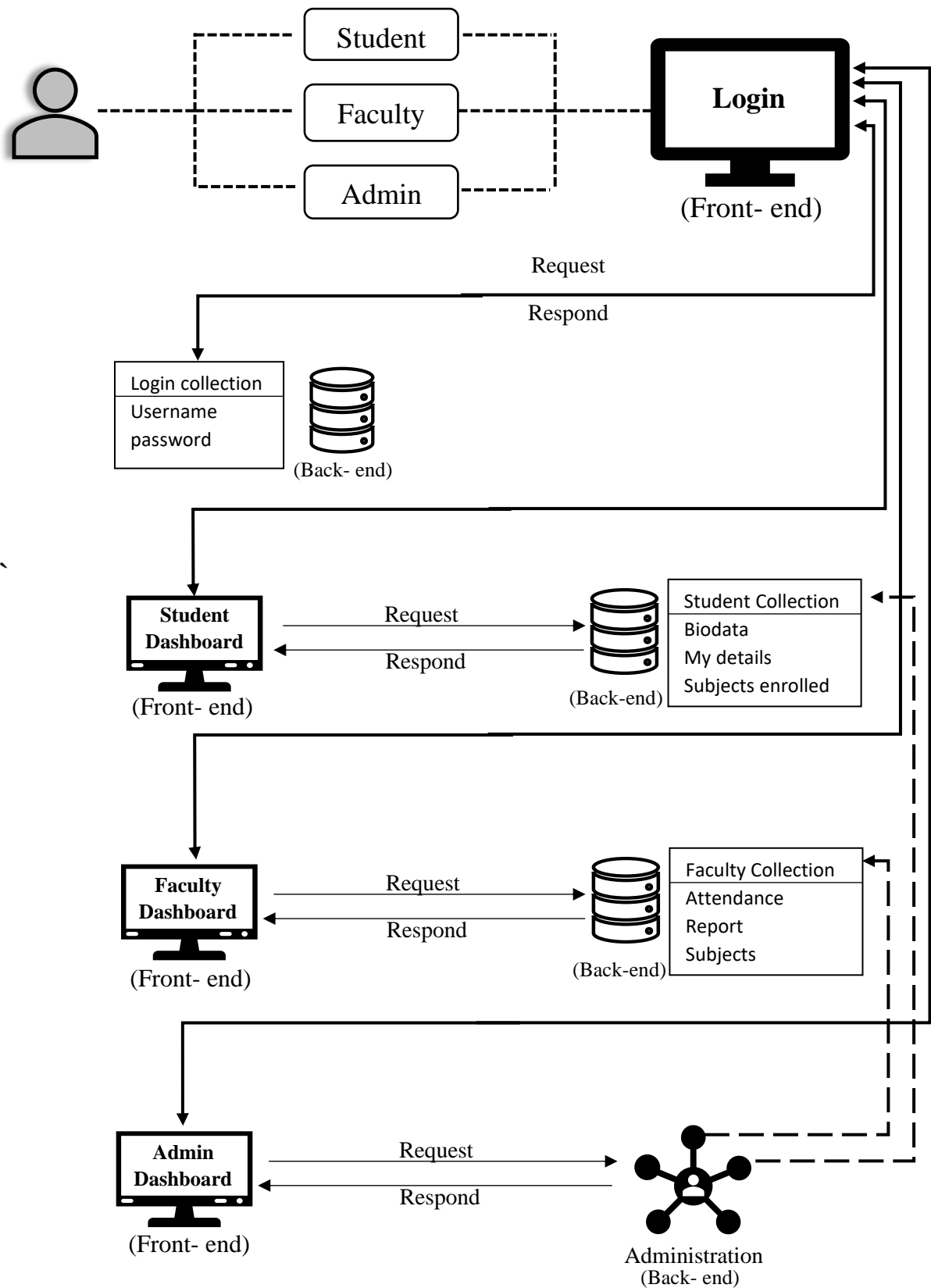


Figure 5.2 Architecture

5.3 USE CASE DIAGRAM

Use Case Diagram is a group of actors. It is a methodology used in system analysis to identify, clarify and organize system requirements. It is made up of a set of possible sequences of interaction between system and users in a particular environment and related to a particular goal.

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. Use case diagrams will specify the events in a system and how those events flow, however, use case diagram does not describe how those events are implemented.

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems. There are a number of benefits with having a use case diagram over similar diagrams such as flowcharts.

The reasons why an organization would want to use case diagrams include:

- Represent the goals of systems and users.
- Specify the context a system should be viewed in.
- Specify system requirements.
- Provide a model for the flow of events when it comes to user interactions.
- Provide an outside view of a system.
- Show's external and internal influences on a system.
- How use case diagrams work.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference or performing a consumer-service-oriented task.

The boundary, which defines the system of interest in relation to the world around it. The actors, usually individuals involved with the system defined according to their roles. The use cases, which are the specific roles played by the actors within and around the system. The relationships between and among the actors and the use cases.

5.3.1 USE CASE - STUDENT

- Each Student can login by using their roll.no and password details.
- After login, student can enter into student dashboard. There student can able to enter their bio data and view their details.
- Student can easily able to view their person details and records through their account given by admin.
- Student can able to enter the details one time only and optional for some changeable details.
- Student can also able to change their account password.

5.3.2 USE CASE – FACULTY

- Each Faculty can login by using their name and password and enter into faculty dashboard page.
- After login, faculty can enter into faculty dashboard page. There is given the subject list details, by using this list faculty can enter into their handling subjects.
- After entering into their handling subject, there is an attendance section. Faculty can enter only absentees' details by entering count and also enter name or roll no of the students.
- Faculty can put attendance period wise only. And also, can check previous days attendance reports.
- Faculty can also able to change their account password.

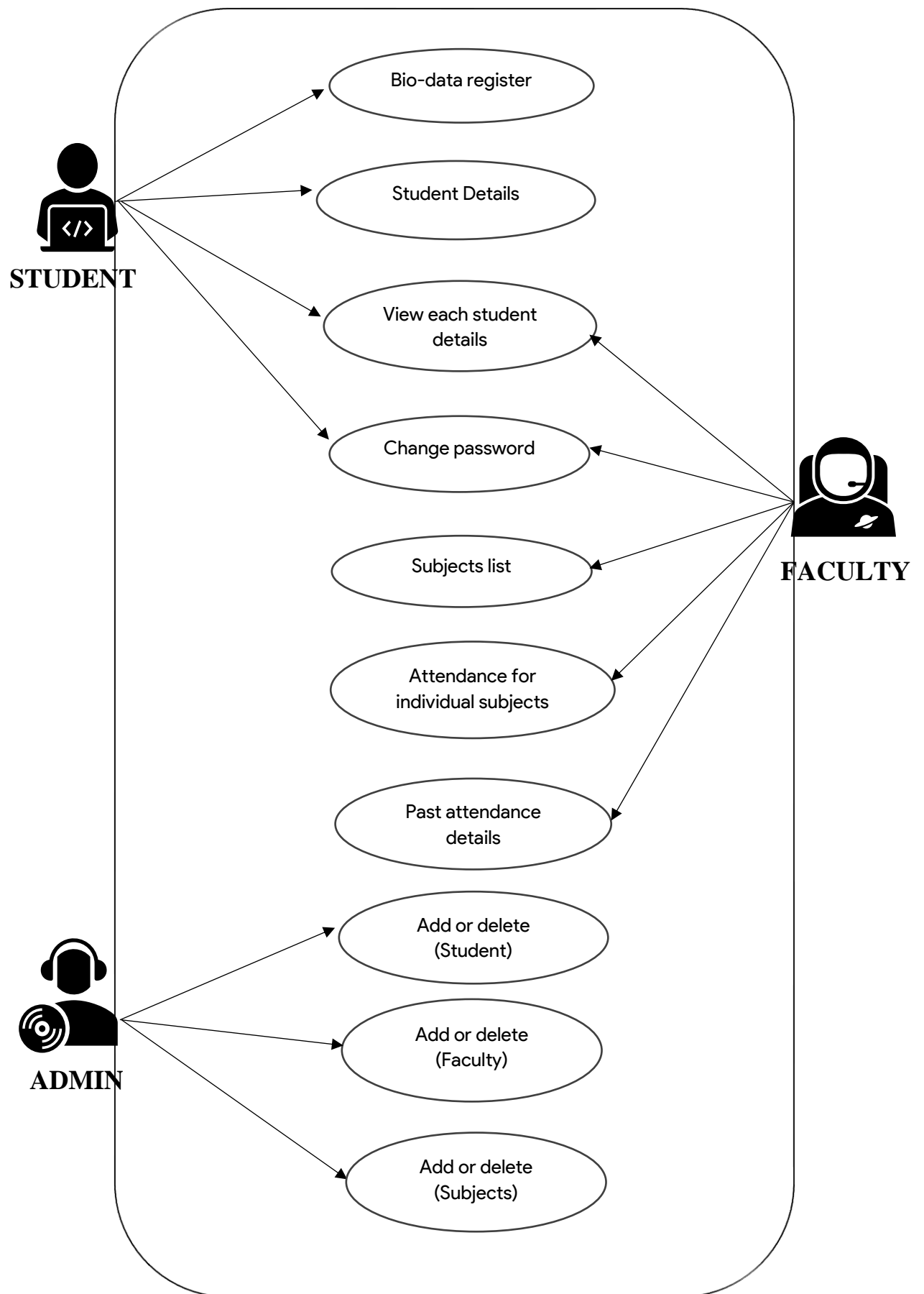


Figure 5.3 Use case diagram

5.3.3 USE CASE – ADMIN

- Admin can login by entering their name and password.
- After login, admin can able to edit student, faculty, and subject related sections.
- By clicking subject add or delete label, admin can able to edit subject details.
- By clicking Faculty add or delete label, admin can able to edit faculty account details.
- By clicking Student add or delete label, admin can able to edit student account details.

5.4 DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. Individuals seeking to draft a data flow diagram must identify external input and output, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

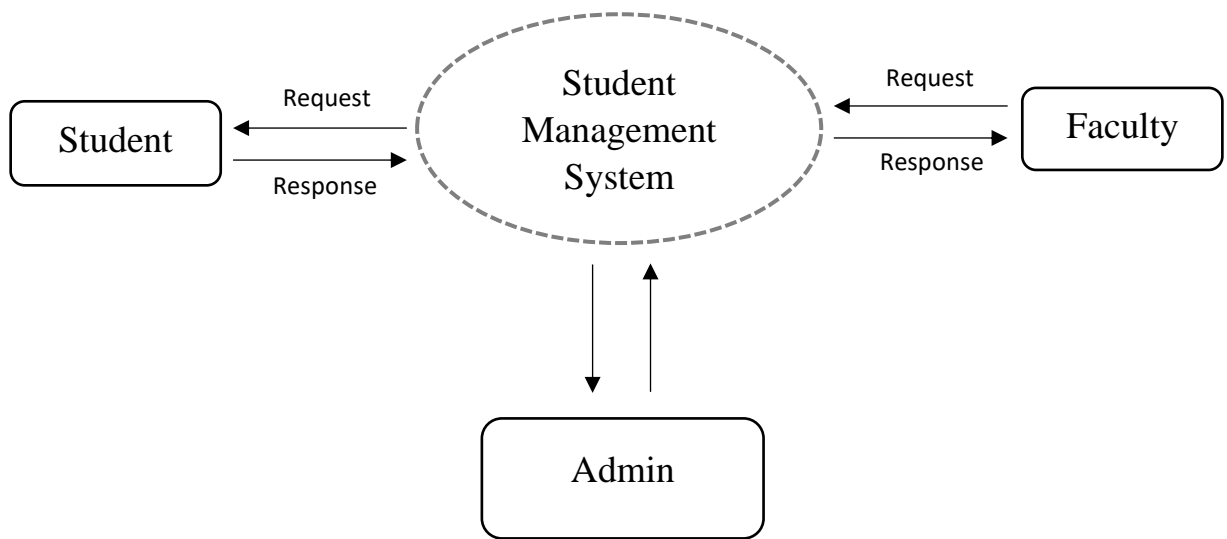


Figure 5.4 Data flow diagram

5.4.1 CONTEXT LEVEL - DATA FLOW DIAGRAM

While User enter in to home page of the website, there is a logon dashboard. By clicking this user can enter into another page, there is three cards called Student, Faculty and Admin. By clicking student card user can enter into login page. By entering roll no and password, user can move to another page called student dashboard. User can also click faculty card. By clicking this card user enter into login page. By entering name and password user can move to another page called Faculty dashboard. By clicking admin card user enter into login page, by entering name and password user can move to another page called admin dashboard.

5.4.2 STUDENT DATA FLOW DIAGRAM

When Student login into web page, the login data will be sent to database and check whether the user have entered the correct credentials or not. If it is correct then the student dashboard will open. There is a menu bar. Here there are few options to do. Bio data, Documents, Subject list, change password &logout. When we enter into bio data section, there student can able to register one time and can change with some specific editable options. When student can register their details once, the data can be stored into student collection in database. Likewise, the data can be stored in each other collections in database.

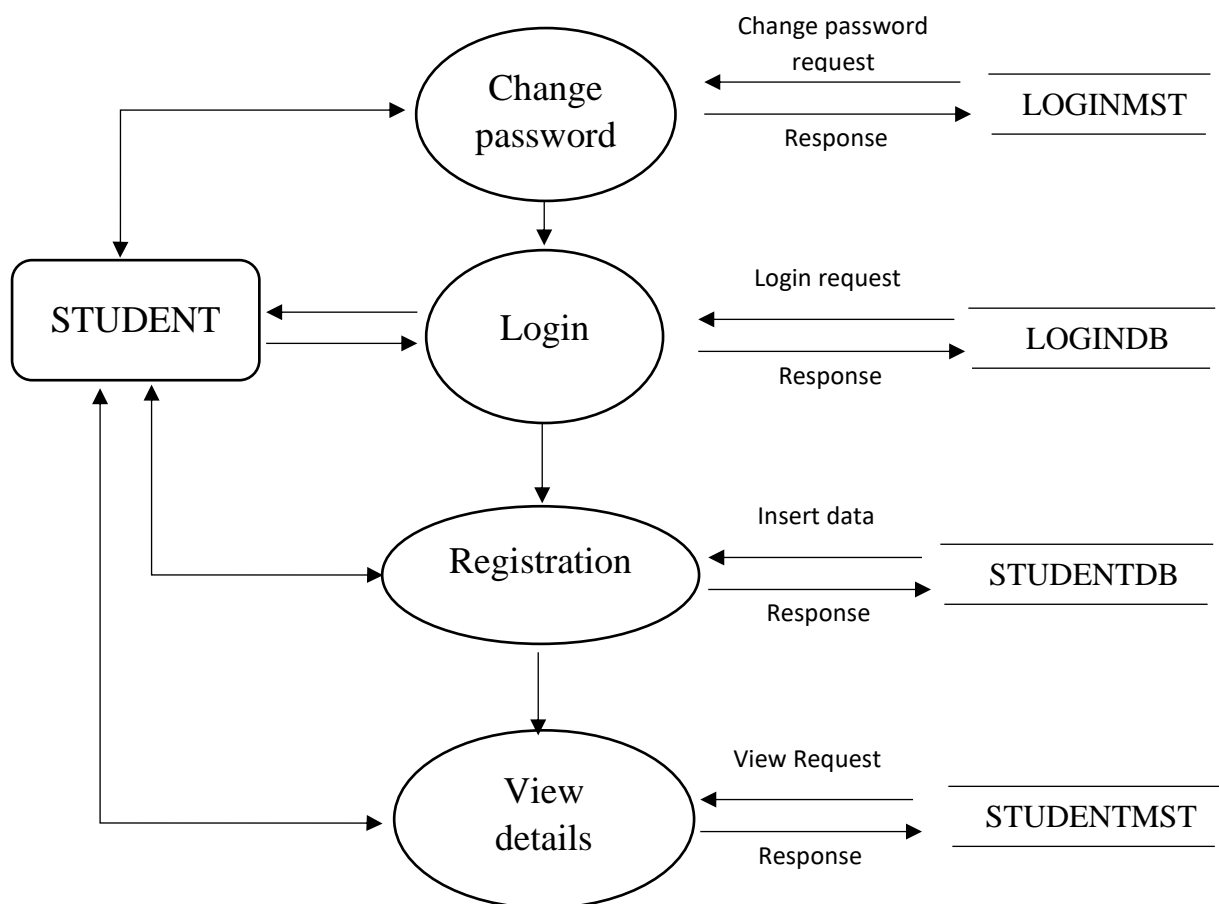


Figure 5.5 Student – DFD

5.4.3 ADMIN DATA FLOW DIAGRAM

When an admin login to the page, the user details can send to database and check for authentication. After the positive result the admin can enter into the admin dashboard. There is a navigation bar. Here is a list of (add or delete) subjects, faculty and students. When admin click these buttons to add or delete, the data changes request can be sent to database and changes response can sent to user.

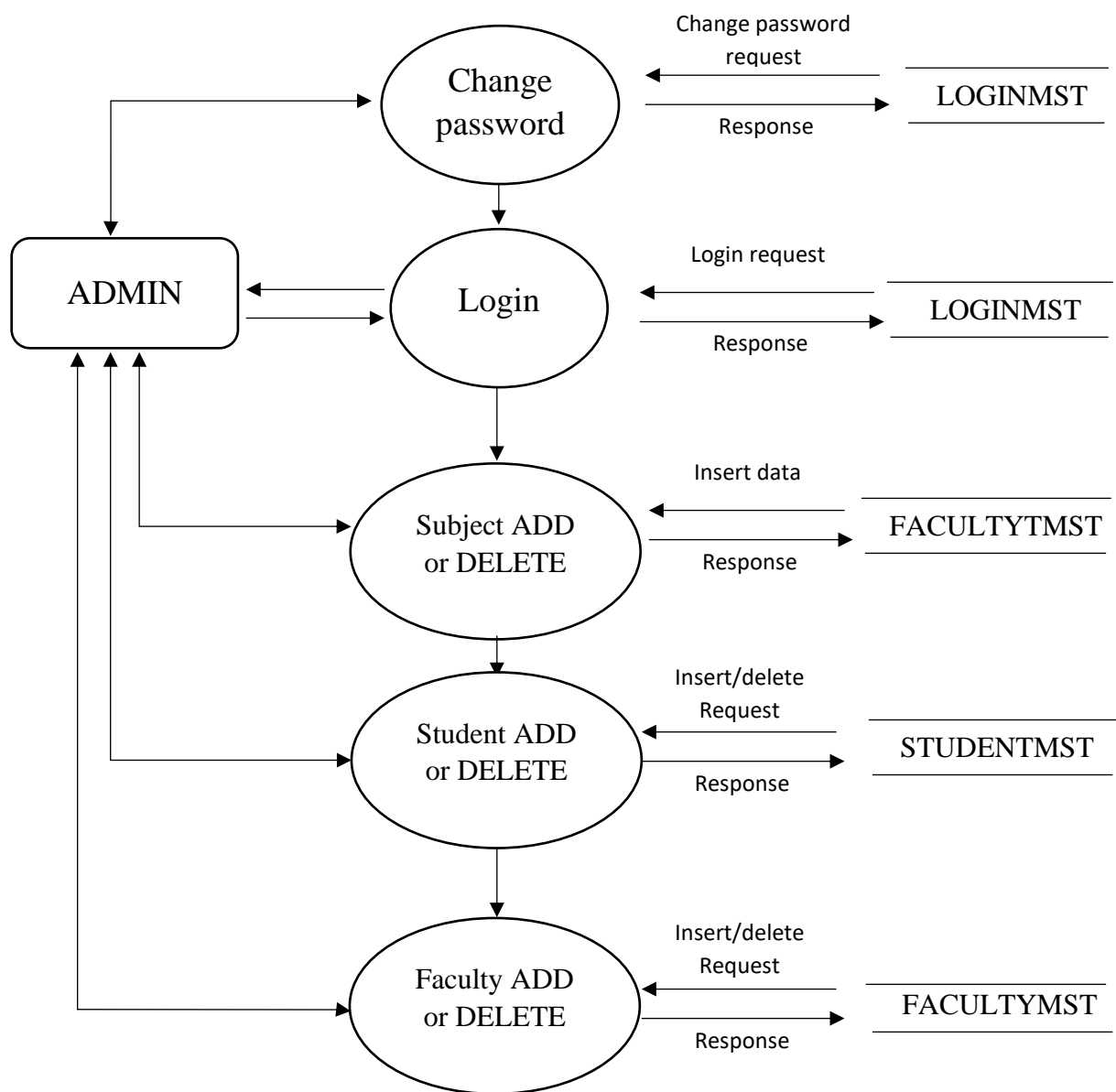


Figure 5.6 Admin - DFD

5.4.4 FACULTY DATA FLOW DIAGRAM

When faculty login to the web page, the entered user details can be sent to database and check for authentication. After checking, the user can be move to another page called faculty dashboard. Here there is the navigation bar. Here some list are shown. Subject list, attendance per period, past attendance and change password. When user click the subject wise attendance, data request can be sent to database and sends responds.

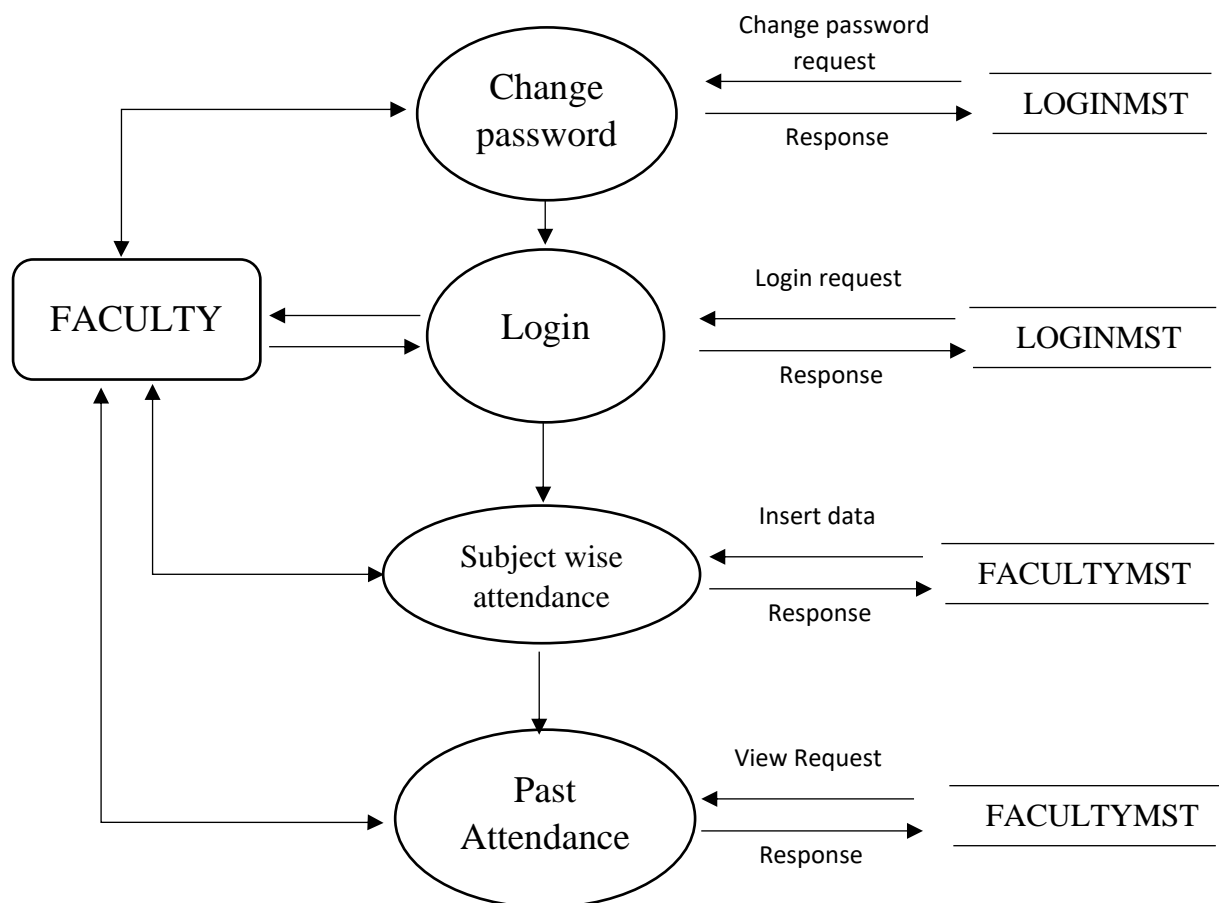


Figure 5.7 Faculty - DFD

CHAPTER 6

MODULES

MODULE DESCRIPTION

6.1 HOME PAGE

A home page is the main page of a website. The Home page of our website contains navbar that navbar contains the main elements of website like

- LOG IN
- NEWS
- ABOUT
- HOME
- LOGOUT

This Elements help to navigate Whole web page.

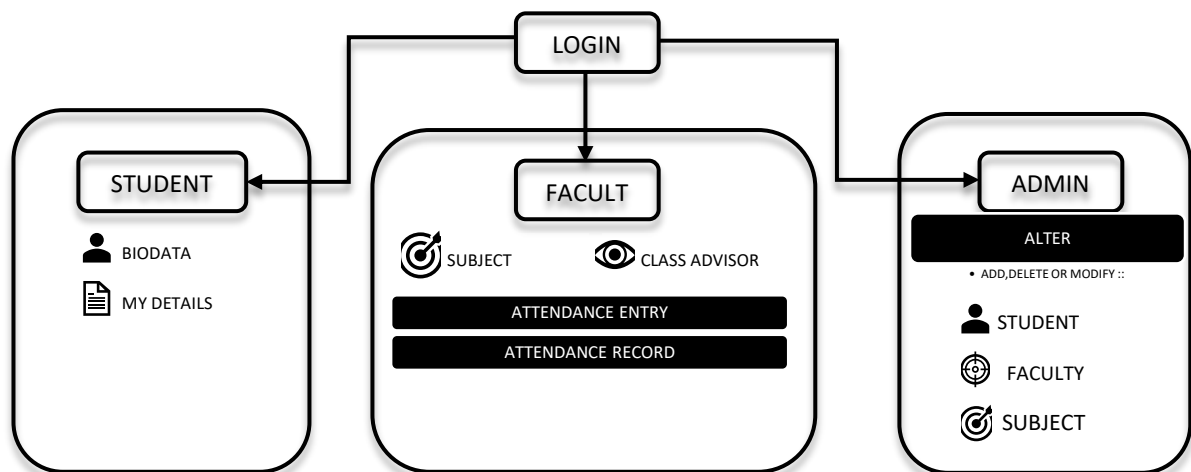


Figure 6.1 Modules Overview

6.2 LOG IN MODULE

The users can access the website only after they logged in, so this module is for that purpose. The login module has two sub modules that first one is for, to help the user to navigate to their login page. The next one is the real login page where they enter their credentials to access the web page.

Our webpage going to have three types of users

1. Student
2. Faculty
3. Admin

The first sub module only helps the different type users to navigate to their respective login page (the second sub module). So that the different types of users can enter their different types of credentials

Table 6.1 Login module

Users	Attribute	Type	Constraint
Student	Regno	Number	Only regno
	Password	String	Longer than 7 characters
Faculty	Faculty name	String	Predefined
	Password	String	Longer than 7 characters
Admin	Username	String	Longer than 5 characters
	Password	String	Longer than 7 characters

6.3 ABOUT MODULE

The About page is the section of a website where users go to find out about the website, they're on. This about page is contains the Members who were worked in this project, motto of the project and introduction of the project which is short Summary of this project

6.4 NEWS MODULE

This web page going to intimate the students about events, test and etc. This admin feed the news on basis.

Table 6.2 News page

Attribute	type	constraint
Year	String	One to four
Description	String	-
Date	Date	-

6.5 STUDENT DASHBOARD

In this dashboard students have to first submit their all bio-data and can view their details. This dashboard will only be visible for students. particularly for that only logged student. In this web page student can

- see their information
- change their password
- easily log out, they need not to go to home page for logout

Table 6.3 Biodata module

Attribute	Type
Name	String
Roll no	Number
Gender	String
Caste	String
Address	String
Mobile no	Number

6.6 FACULTY MODULE

Using previous login module, this dashboard will show particular Classes for the Faculty. Faculty can put attendance for that their subject class. this module will help that faculty user to changing password, direct log out. Faculty will be able to view student details through Firestore database. This dashboard will alter for each faculty user (i.e. class advisor and subject faculty).

Table 6.4 Faculty module

Attribute	Type
Faculty Name	String
Subjects	String

6.7 ATTENDANCE MODULE

This module for faculty users. This module was navigated from faculty dashboard. Using this module faculty can put attendance, and also for seeing attendance history. This module contains report generation for that respective subject. The module contains form that have input of period, year and number of absentees. Using this form faculty can submit their attendance.

Table 6.5 Attendance module

Attribute	Type	Constraint
Date	Date	Date format
Absentees	Number	Less than enrolment value
Period	Number	Less than 8
Year	Number	1 to 4
Subject	String	Predefined

6.8 ADMIN DASHBOARD

This dashboard is for Admin. This module give access to the admin for adding and deleting student profiles and faculty's subjects. In this module admin can change their credential and logout using dashboard. The module has navigation bar for admin convenience. This dashboard has counts for the number of students, faculty and courses.

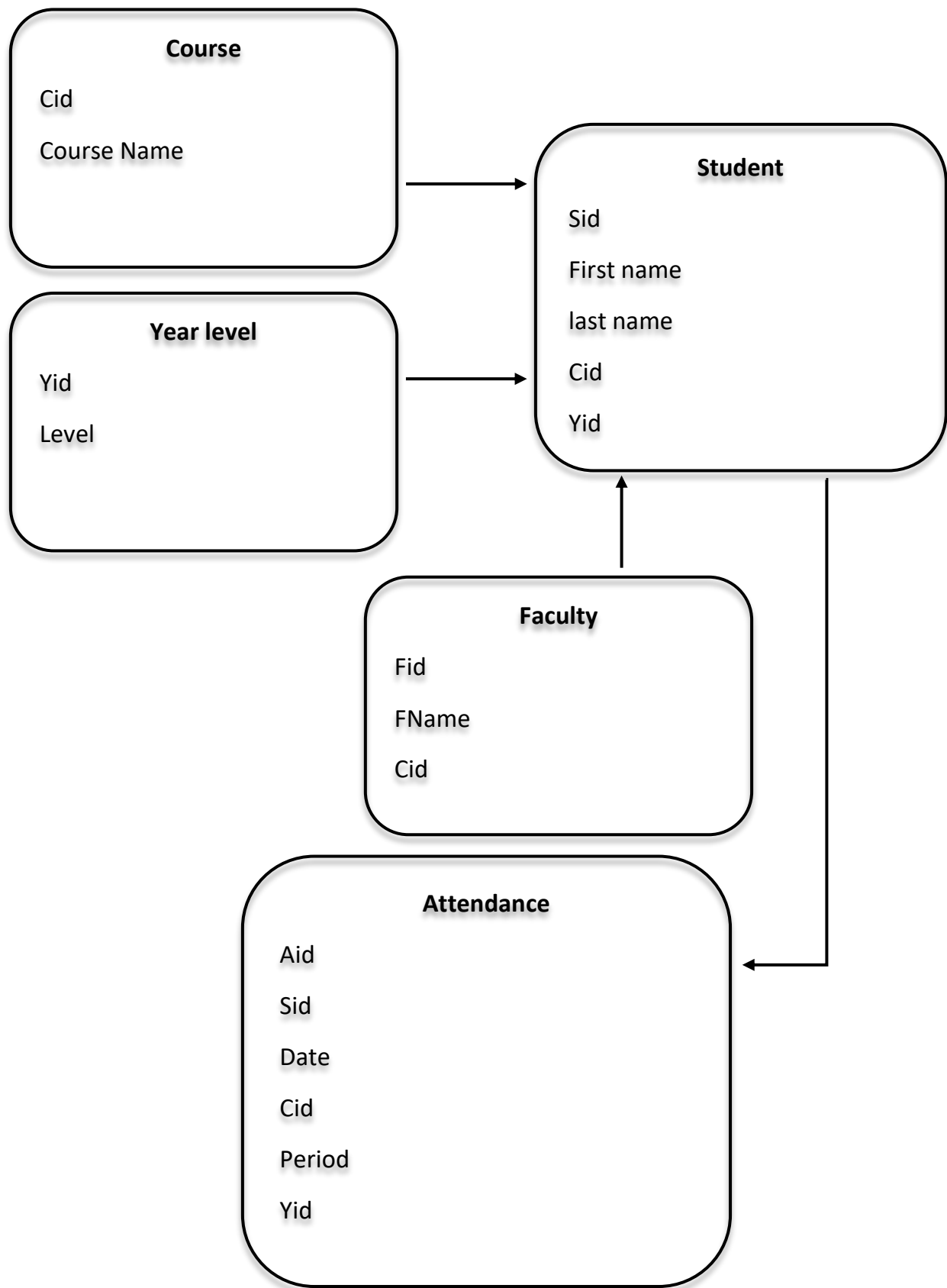


Figure 6.2 Database Schema

CHAPTER 7

SYSTEM TESTING

7.1 INTRODUCTION

System Testing includes testing of a fully integrated software system. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements. To check the end-to-end flow of an application or the software as a user is known as **System testing**.

In this, all the necessary modules of an application are navigated and checked if the end features or the end product works fine, and test the product as a whole system. It is **end-to-end testing** where the testing environment is similar to the production environment.

7.2 TEST SCENARIOS

The various system test scenarios are as follows:

7.2.1 Scenario 1

The admin surfs the website, opens it and logs into the admin dashboard using his credentials. Under the students edit tab, the admin will add the student unique identity number who have registered for the current semester. Under the subjects edit tab, the admin will add the subject name and code with the year number. Under the teachers edit tab, the admin will add the teachers. The teachers will be registered along with their subjects to be handled in the current semester.

7.2.2 Scenario 2

All the students who have been registered have to log in to their dashboard and fill up all the details which are listed in the side menu bar for the student's personal reference. The student can change his password for security reasons. Once all the details are filled, the student can't alter except the one's that changes every year such as income certificate.

7.2.3 Scenario 3

The admin will be notified about the student's details submission.

7.2.4 Scenario 4

The teachers can enter their dashboard and manage attendance periodically per day for each of their subjects undertaken and obtain the attendance count list daily and monthly basis. There is a class advisor section where the teachers can handle all the subjects list of the particular class along with their attendance management.

7.3 APPROPRIATE TESTS

7.3.1 Load Testing

Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.

7.3.2 Functional Testing

Functional testing of a system is performed to find if there's any missing function in the system. Tester makes a list of vital functions that should be in the system and can be added during functional testing and should improve quality of the system.

7.3.3 Usability Testing

The purpose of this testing is to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.

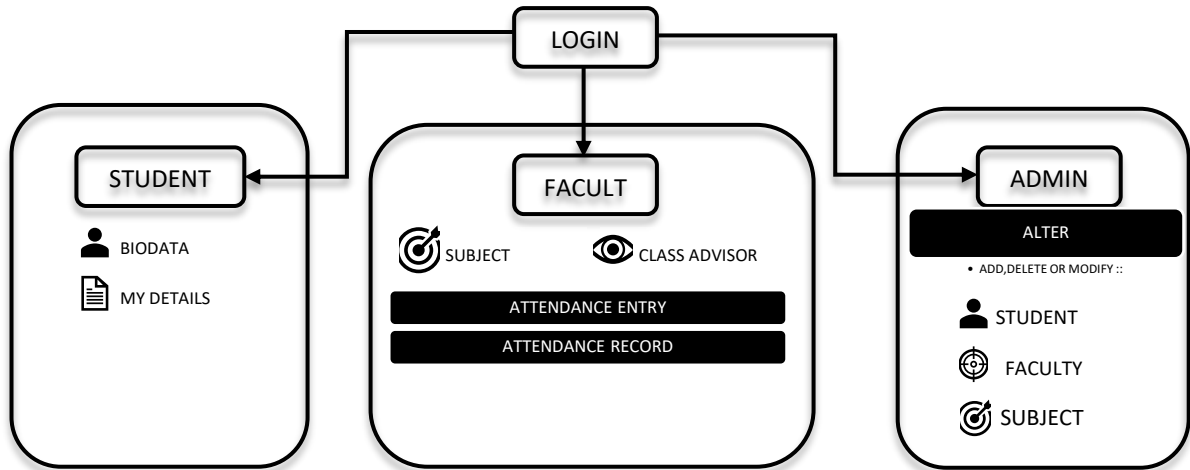


Figure 7.1 Modules- Student ,Faculty ,Admin

7.4 TEST SCENARIOS ON MODULES

There are three different major modules like student, faculty and admin:

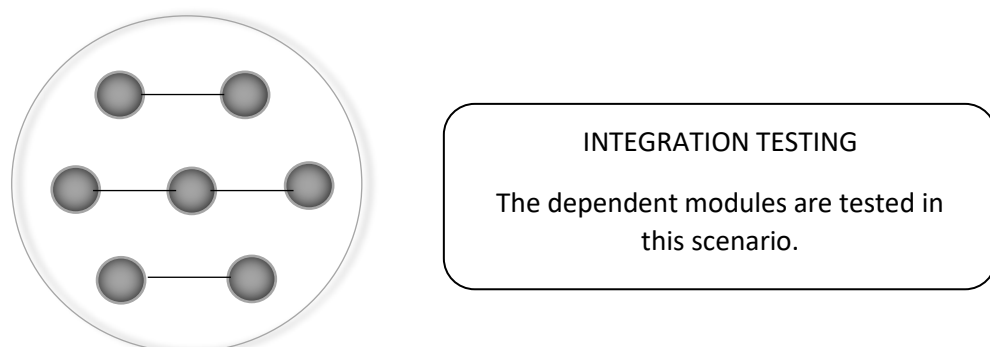


Figure 7.2 Integration testing

7.4.1 Scenario 1

This test is carried out to ensure that the database is linked properly through admin, faculty and student.

- First logged in as admin, let us say A and registered with the subjects of current semester. Then the associated technical faculty are allotted their subjects and the students are registered with their register numbers. Now, Admin logged out.
- After that, logged in as a faculty and the associated subjects are being configured whether they are linked or not and logout.
- Now using the default credentials of a student, logged in as a student.



Figure 7.3 End-to-end testing

7.4.2 Scenario 2

This test is carried out to ensure that the edits done by admin is reflected back in other modules too.

- In case of a student or faculty leaves the college, it is required to delete their credentials from the website. So, logged in as admin, under the faculty-edit tab, the particular faculty's name is entered and deleted. Also in the similar way, under the student-edit tab, the particular student's name is deleted.

- Now, an attempt is made to log in as student with the deleted credentials and it fails. In similar manner, an attempt is made to log in as faculty with the deleted credentials and it again fails.

7.4.3 Scenario 3

This test is performed to test data security.

- A student can only enter into his dashboard only when he enters correct credentials otherwise fails. An attempt is made to directly seek into a student's dashboard by any means and it returns back to login page.
- Likewise, the student's credential is used to log into the admin dashboard and faculty dashboard and that too fails.

7.4.4 Scenario 4

This test is carried out to check the attendance is stored properly or not.

- Logged in as faculty and a subject is chosen say S then the attendance window pops up. Now, the absentee's registered numbers are entered and submitted.
- The attendance is now checked in the attendee tab of the subject S.

The following are performed under system testing:

- Verification of input functions of the application to test whether it is producing the expected output or not.
- Testing of the whole system for End-to-End testing.
- Behaviour testing of the application via a user's experience.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION

Student Management System can be used by educational institutions to maintain their student records easily. Achieving this objective is difficult using the manual system as the information is scattered, can be redundant, and collecting relevant information may be very time-consuming. All these problems are solved by this project.

This system helps in maintaining the information of pupils of the organization. It can be easily accessed by the persons who and kept safe for a long period of time without any changes. Student management systems make faculty jobs more accessible by giving them an easy place to find and sort information. This system allows teachers and student managers to follow with their student engagement.

The idea is to create a scenario that makes the works of administration and teachers simpler. This project is intended to serve the easiest way to handle student database in a centralized manner which can be accessed by any person with the credentials.

To conclude, this project works like a component which can access all the databases. It overcomes the many limitations incorporated in the attendance.

- Easy implementation environment
- Generate report flexibly

8.2 FUTURE ENHANCEMENT

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner. The following are the future scope for the project.

- Discontinue of particular student eliminate potential attendance.
- Bar code Reader based attendance system
- Individual Attendance system with photo using Student login

APPENDIX-1

SOURCE CODE

App.js

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Home from "../components/home/home";
import Logincard from "../components/logincard/Logincard.js";
import Staff from "../components/staff/staff.js";
import News from "../components/card/news.js";
import About from "../components/about/about.js";
import StudentLogin from "../components/login/student_login.js";
import StaffLogin from "../components/login/staff_login";
import AdminLogin from "../components/login/admin_login";
import Attendance from "../components/attendance/attendance.js";
import Admin from "../components/adminmodule/admin/admin";
import Studentedit from "../components/adminmodule/studentsedit/studentedit";
import Staffedit from "../components/adminmodule/staffedit/staffedit";
import Subjectedit from "../components/adminmodule/subjectsedit/subjectedit";
import ChangePWD from "../components/adminmodule/changePWD/changepwd";
import { AuthProvider } from "../components/contexts/AuthContext";
import PrivateRoute from "../components/contexts/privateroutes";
import Studentdash from "../components/studentdashboard/studentdash";
import Biodoc from "../components/studentdashboard/Biodoc";
import Mydetailsdisplay from
"../components/studentdashboard/mydetailsdisplay";
import ChangePWDstudent from
"../components/studentdashboard/changePWD/changepwdstudent";
import ChangeStaffPWD from "../components/staff/changeStaffPwd";
import Studentbio_data from "../components/studentdashboard/studentbio_data";
import SearchStudent from "../components/staff/searchstud_info";
import Report from "../components/Report/report";

function App() {
  return (
    <Router>
      <AuthProvider>

        <Routes>
```

```

    <Route path="/" exact element={<Home />} />
    <Route path="/Login_card" element={<Logincard />} />
    <Route path="/login-Student" element={<StudentLogin />} />
    <Route path="/login-Staff" element={<StaffLogin />} />
    <Route path="/login-Admin" element={<AdminLogin />} />
    <Route path="/News" element={<News />} />
    <Route path="/about" element={<About />} />
    <Route element={<PrivateRoute/>}>
      <Route path="/student-dashboard" element={<Studentdash />} />
      <Route path="/bio-data" element={<Studentbio_data />} />
      <Route path="/change-password" element={<ChangePWD />} />
      <Route path="/admin-subjects-edit" element={<Subjectedit />} />
      <Route path="/Classes" element={<Staff />} />
      <Route path="/admin-staff-edit" element={<Staffedit />} />
      <Route path="/Admin-dashboard" element={<Admin />} />
      <Route path="/Attendance" element={<Attendance />} />
      <Route path="/Admin-student-edit" element={<Studentedit />} />
      <Route path="/student-dashboard/details"
element={<Mydetailsdisplay/>}/>
      <Route path="/student-dashboard/change-password"
element={<ChangePWDstudent/>}/>
      <Route path="/change-fac-password" element={<ChangeStaffPWD />}
/>

      <Route path="/report" element={<Report/>}/>
      <Route path="/search-student" element={<SearchStudent/>}/>
    </Route>
  </Routes>
</AuthProvider>
</Router>

  );
}

```

```

export default App;

```

firebaseConfig.js

```
import { initializeApp } from "firebase/app";
import { getFirestore } from "@firebase/firestore";
import { getStorage } from "firebase/storage";

const firebaseConfig = {
  apiKey: "AIzaSyCugSzgF_J0poowdvJsbutd1ptd67a_QmM",
  authDomain: "attendance-a6e82.firebaseio.com",
  databaseURL: "https://attendance-a6e82-default-rtdb.firebaseio.com",
  projectId: "attendance-a6e82",
  storageBucket: "attendance-a6e82.appspot.com",
  messagingSenderId: "181145177933",
  appId: "1:181145177933:web:9d7c218144292648d43104",
  measurementId: "G-QM1B2DK2F0"
};

const app=initializeApp(firebaseConfig);
const storage = getStorage()
export const db=getFirestore(app);
export {storage};
```

AuthContext.js

```
import React, { useContext, useState, useEffect } from "react"
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  sendPasswordResetEmail,
  onAuthStateChanged,
  signOut,
} from "firebase/auth";
import { auth } from "../../firebase";

const AuthContext = React.createContext({})

export function useAuth() {
  return useContext(AuthContext)
}
```

```

export function AuthProvider({ children }) {
  const [currentUser, setCurrentUser] = useState()
  const [loading, setLoading] = useState(true)

  function signup(email, password) {
    return createUserWithEmailAndPassword(auth, email, password)
  }

  function login(email, password) {
    return signInWithEmailAndPassword(auth, email, password)
  }

  function logout() {
    return auth.signOut()
  }

  function resetPassword(email) {
    return sendPasswordResetEmail(email)
  }

  function updateEmail(email) {
    return currentUser.updateEmail(email)
  }

  function updatePassword(password) {
    return currentUser.updatePassword(password)
  }

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged(user => {
      setCurrentUser(user)
      setLoading(false)
    })

    return unsubscribe
  }, [])

  const value = {
    auth,
    currentUser,
    login,
    signup,
    logout,
    resetPassword,
    updateEmail,
    updatePassword
  }

```



```

    }

    return (
      <AuthContext.Provider value={value}>
        {!loading && children}
      </AuthContext.Provider>
    )
  }
}

```

privateroutes.js

```

import React from "react"
import { Navigate, Outlet, useLocation } from "react-router-dom"
import { useAuth } from "../AuthContext"

export default function PrivateRoute() {
  const { currentUser } = useAuth()
  // const {auth }= useAuth();
  const location = useLocation();

  return (
    currentUser ? <Outlet /> : <Navigate to="/login_card" state ={{from:
location}} replace />
  )
}

```

subject.services.js

```

import { db } from "../../firebaseConfig";
import { setDoc } from "firebase/firestore";
import {
  collection,
  getDocs,
  getDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  doc,
} from "firebase/firestore";

```

```

const subjectCollectionRef = collection(db, "subject");
const studentCollectionRef = collection(db, "student");
const AbsenteesCollectionRef = collection(db, "absentees");
class subjectDataService {
  addSubjects = (newSubject) => {
    return addDoc(subjectCollectionRef, newSubject);
  };
  addStudent = (rollNoImg, newSubject) => {
    console.log(newSubject);
    return setDoc(doc(studentCollectionRef, rollNoImg), newSubject);
  };

  updateStudent = (rollNoImg, updatedSubject) => {
    console.log(updatedSubject);
    const subjectDoc = doc(studentCollectionRef, rollNoImg);
    return updateDoc(subjectDoc, updatedSubject);
  };

  updateSubject = (id, updatedSubject) => {
    const subjectDoc = doc(db, "subject", id);
    return updateDoc(subjectDoc, updatedSubject);
  };

  deleteSubject = (id) => {
    const subjectDoc = doc(db, "subject", id);
    return deleteDoc(subjectDoc);
  };

  getAllSubjects = () => {
    return getDocs(subjectCollectionRef);
  };

  getSubject = (id) => {
    const subjectDoc = doc(db, "subject", id);
    return getDoc(subjectDoc);
  };
  getReport=()=>{
    return getDocs(AbsenteesCollectionRef);
  }
  getStudent=()=>{
    return getDocs(studentCollectionRef)
  }
  getstudoc = (id) => {
    console.log(id)
    const studoc = doc(db, "student", id);

```

```

        return getDoc(studoc);
    };
}

export default new subjectDataService();

```

facultytosubject.services.js

```

import { db } from "../../../firebaseConfig";

import {
    collection,
    getDoc,
    doc,
} from "firebase/firestore";

const faculties = collection(db, "staff");

class facultytosubServices {

    getSubject = (id) => {
        const subjectDoc = doc(db, "subject", id);
        return getDoc(subjectDoc);
    };
}

export default new facultytosubServices();

```

adminlogin.js

```

import React, { useRef, useState } from "react"
import { useAuth } from "../contexts/AuthContext"
import { Link, useNavigate } from "react-router-dom"
import "../loginstyle.css";

function AdminLogin() {
    const regno = useRef();
    const password = useRef();
    const { login } = useAuth()
    const [error, setError] = useState("")
    const [loading, setLoading] = useState(false)
    const navigate = useNavigate()

```

```

async function handleSubmit(e) {

    e.preventDefault();
    try {
        setError("");
        if (regno.current.value=== "admin") {
            setLoading(true)
            await login(regno.current.value+"@gmail.com", password.current.value);
            navigate("/Admin-dashboard");
        }else{
            setError("Failed to log in");
        }
    } catch {
        setError("Failed to log in");
    }
    setLoading(false)
}

return (

    <div className="Login">
        <div className="login_user">
            <div className="login_container">
                <div className="login_wrapper">
                    <div className="title"><span>Admin Login</span></div>
                    <form onSubmit={handleSubmit} className="login_form">
                        {error && <h2>{error}</h2>}
                        <div className="row">
                            <i className="fas fa-user"></i>
                            <input type="text" ref={regno} placeholder="Admin"
minLength={5} required />
                        </div>
                        <div className="row">
                            <i className="fas fa-lock"></i>
                            <input type="password" ref={password} placeholder="Password"
minlenght={8} required />
                        </div>
                        <div className="row button">
                            <input type="submit" disabled={loading} value="Login" />
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
)

```

```

    </div>

    );
}
export default AdminLogin;

```

report.js

```

import React, { useEffect, useRef, useState } from 'react'
import rcss from "../report.module.css"
import { Table, Button } from "react-bootstrap";
import SubjectDataService from '../adminmodule/services/subject.services';
import { useLocation } from 'react-router-dom';

export default function Report() {
    const Dateref = useRef();
    const [Report, setReport] = useState([]);
    const [Student, setStudent] = useState([]);

    const location = useLocation();
    const [percentageform, setPercentageform] = useState(true);
    const [Absentform, setAbsenteesform] = useState(false);
    const [Loading, setLoading] = useState(false);
    const [Dates, setDates] = useState("0000-00-00")

    useEffect(() => {
        getReport();
        getStudent();
    }, []);
    const getReport = async () => {
        const data = await SubjectDataService.getReport();

        setReport(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })));
    };
    const getStudent = async () => {
        const data = await SubjectDataService.getStudent();

        setStudent(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })));
    };
    const showabsentform = () => {
        setAbsenteesform(true)
        setPercentageform(false)
    }
}

```

```

const showpercantageForm = () => {
    setAbsenteesform(false)
    setPercentageform(true)
}
var result = [];

var totsub = 0, periods = 0;
let p = 1;
Report.map((item) => {
    if (item.subject === location.state.sub) {
        totsub++;
    }
    if (Dates === item.date) {
        periods++;
        result = [
            ...result,
            {
                id: p++,
                noabs: item.NoAbs,
                regno: item.regno,
                period:item.period,
                name: [],
                whole:[]
            }
        ]
    }
})

var wholeStudent = []
let i = 1;
Student.map((item) => {
    wholeStudent = [
        ...wholeStudent,
        {
            id: i++,
            name: item.name,
            regno: item.regno,
            percentage: totsub
        }
    ]
})
wholeStudent.map((item) => {
    Report.map((absentees) => {
        if (absentees.subject === location.state.sub) {
            for (let i = 0; i < absentees.NoAbs; i++) {

```

```

        if (item.regno === absentees.regno[i]) {
            item.percentage--;
        }
    }
}
}))
var result2 = []
result.map((item) => {
    for (let y = 0; y < item.regno.length; y++) {
        Student.map((items) => {
            if (item.regno[y] === items.regno) {
                // item.name[y]=items.name
                item.whole=[...item.whole,items]
            }
        })
    }
})
}
)
totsub = 100 / totdsub;

const dateModification = () => {
    setLoading(true)
    setDates(Dateref.current.value)
}

return (
    <div className={rcss.report}>
        <header>
            <nav>
                <h1 className={rcss.subject}>{location.state.sub}</h1>
                <h3
className={rcss.faculty}>{location.state.facname}</h3>
            </nav>
        </header>
        <div className={rcss.btn}>
            <button className={percentageform ? rcss.toggle_btn1 :
rcss.toggle_btn2} onClick={showpercentagForm}><h4>Attendance
Percentage</h4></button>
            <button className={Absentform ? rcss.toggle_btn1 :
rcss.toggle_btn2} onClick={showabsentform}><h4>Absentees</h4></button>
        </div>
        <div className={rcss.reportcontent}>
            {percentageform && <section>

```

```

<div className={rcss.percentage}>
  <Table striped bordered hover size="sm" >
    <thead>
      <tr>
        <th>#</th>
        <th>Regno</th>
        <th>Name</th>
        <th>Percentage</th>
      </tr>
    </thead>
    <tbody>
      {wholeStudent.map((doc, index) => {
        return (
          <tr key={doc.id}>
            <td>{index + 1}</td>
            <td>{doc.regno}</td>
            <td>{doc.name}</td>
            <td>{doc.percentage * totsub}</td>
          </tr>
        );
      })}
    </tbody>
  </Table>
</div>
</section>
<section>
  {Absentform&&
    <div >
      <div className={rcss.absent}>
        <form name="form">
          <input type="date" id={rcss.input}
className="auto-submit" ref={Dateref} onChange={dateModification} />
        </form>
      </div>
      <h2 className={rcss.tot__period}>Total Period
<span>{periods}</span></h2>
      {result.map((docs) => {
        return (
          <div>
            <div className={rcss.details}>
              <h4>Period
<span>{docs.period}</span></h4>
              <h5>No of Absentees
<span>{docs.noabs}</span></h5>
            </div>

```



```

        <div className={rcss.result}>
            <Table striped bordered hover size="sm" >
                <thead>
                    <tr>
                        <th>#</th>
                        <th>Regno</th>
                        <th>Name</th>
                    </tr>
                </thead>
                <tbody>
                    {docs.whole.map((doc, index) => {
                        return (
                            <tr key={doc.id}>
                                <td>{index + 1}</td>
                                <td>{doc.regno}</td>
                                <td>{doc.name}</td>
                            </tr>
                        );
                    })}
                </tbody>
            </Table>
        </div>
    </div>
    );
    })}
</div>
</section>
</div>
</div>
)
}

```

changepwdstudent.js

```

import { Link } from 'react-router-dom';
import React, { useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { useAuth } from "../../contexts/AuthContext";
import "./chagePwd.css"

function ChangePWDstudent() {

    const Navigate = useNavigate();
    const [error, setError] = useState("")

```

```

const { logout,updatePassword } = useAuth()
const password1 = useRef();
const password2 = useRef();
const [ok, setOk] = useState(false)
const [loading, setLoading] = useState(false)
async function handleLogout() {
  setError("")
  try {
    await logout()
    Navigate("/");
  } catch {
    setError("Failed to log out")
  }
}
async function changingPassword(e) {
  e.preventDefault();
  // setError("")
  try {
    setLoading(true)
    console.log(password1.current.value)
    console.log(password2.current.value)
    const pwd = password1.current.value;
    if (password1.current.value === password2.current.value) {
      console.log("inside if");
      await updatePassword(pwd)
      setOk(true)
      password1.current.value = ""
      password2.current.value = ""
      setError("Password Changed Successfully")
    } else {
      setError("Different Password")
    }
  } catch {
    setError("Failed to change password or try Login again")
  }
  setLoading(false)
}
return(
  <div className="changepassword">
    <div id="admin_header"><h1>STUDENT DASHBOARD</h1></div>
    <div className="admin-wrapper">
      <input type="checkbox" id="btn" hidden />
      <label htmlFor="btn" className="admin-menu-btn">
        <i className="fas fa-bars"></i>
        <i className="fas fa-times"></i>

```

```

</label>
<nav id="sidebar">
  <div className="admin-title">
    Side Menu
  </div>
  <ul className="list-items_stud">
    <li><Link to="/student-dashboard"><i className="fas fa-home"></i>Home</Link></li>
    <li><Link to="/bio-data"><i className="fas fa-sliders-h"></i>Update Details</Link></li>
    <li><Link to="/student-dashboard/details"><i className="fas fa-address-book"></i>Review Details </Link></li>
    <li><Link to="/student-dashboard/change-password"><i className="fas fa-stream"></i>Change Password</Link></li>
    <li><button className="logout-btn"
onClick={handleLogout}><i className="fas fa-user"></i>&ensp;&ensp;Log
out</button></li>
  </ul>
</nav>

</div>

<div className="admin-wrapper1">
  <div className="admin-container">
    <div className="admin-simple-cards">
      {error} && <div>
        <h2>{error}</h2>
        {ok} && <button id="input_sub_add" onClick={e => {
Navigate("/student-dashboard") }}>ok</button>
      </div>
    <div className="admin-items">
      <h4>NEW PASSWORD</h4>
      <div className="admin-cards-content">
        <form onSubmit={changingPassword}>
          <input id="input_reg" type="text" placeholder="Enter
Password" ref={password1} required onFocus={e=>{setError('')}} />
          <input id="input_reg" type="text" placeholder="Re-Enter
Password" ref={password2} required onFocus={e=>{setError('')}}
style={{marginTop: "2.5em"}} />
          <input id="input_sub_add" type="submit" value="CHANGE" />
        </form>
      </div>
    </div>
  </div>
</div>

```

```

        </div>
    </div>
    </div>
)
}

export default ChangePWDstudent;

```

addSubject.js

```

import React, { useState, useEffect } from "react";
import { Form, Alert, InputGroup, Button, ButtonGroup } from "react-
bootstrap";
import subjectDataService from "../services/subject.services.js";

const AddSubject = ({ id, setSubjectId }) => {
    const [name, setName] = useState("");
    const [code, setCode] = useState("");
    const [year, setYear] = useState("III");
    const [abb, setAbb] = useState("");
    const [flag, setFlag] = useState(true);
    const [message, setMessage] = useState({ error: false, msg: "" });

    const handleSubmit = async (e) => {
        e.preventDefault();
        setMessage("");
        if (name === "" || code === "" || abb === "" ) {
            setMessage({ error: true, msg: "All fields are mandatory!" });
            return;
        }
        const newSubject = {
            name,
            code,
            abb,
            year,
        };

        try {
            if (id !== undefined && id !== "") {
                await subjectDataService.updateSubject(id, newSubject);
                setSubjectId("");
                setMessage({ error: false, msg: "Updated successfully!" });
            } else {

```

```

        await subjectDataService.addSubjects(newSubject);
        setMessage({ error: false, msg: "New Subject added successfully!" });
    }
} catch (err) {
    setMessage({ error: true, msg: err.message });
}

setName("");
setCode("");
setAbb("");
};

const editHandler = async () => {
    setMessage("");
    try {
        const docSnap = await subjectDataService.getSubject(id);
        // console.log("the record is :", docSnap.data());
        setName(docSnap.data().name);
        setCode(docSnap.data().code);
        setYear(docSnap.data().year);
        setAbb(docSnap.data().abb);
    } catch (err) {
        setMessage({ error: true, msg: err.message });
    }
};

useEffect(() => {
    if (id !== undefined && id !== "") {
        editHandler();
    }
}, [id]);
return (
    <>
    <div className="p-4 box">
        {message?.msg && (
            <Alert
                variant={message?.error ? "danger" : "success"}
                dismissible
                onClose={() => setMessage("")}
            >
                {message?.msg}
            </Alert>
        )}

        <Form onSubmit={handleSubmit}>

```

```

<Form.Group className="mb-3" controlId="formSubjectTitle">
  <InputGroup>
    <InputGroup.Text id="formSubjectTitle">Name</InputGroup.Text>
    <Form.Control
      type="text"
      placeholder="Subject Name"
      value={name}
      onChange={(e) => setName(e.target.value)}
    />
  </InputGroup>
</Form.Group>

<Form.Group className="mb-3" controlId="formSubjectAuthor">
  <InputGroup>
    <InputGroup.Text id="formSubjectAuthor">Code</InputGroup.Text>
    <Form.Control
      type="text"
      placeholder="Subject code"
      value={code}
      onChange={(e) => setCode(e.target.value)}
    />
  </InputGroup>
</Form.Group>

<Form.Group className="mb-3" controlId="formSubjectAuthor">
  <InputGroup>
    <InputGroup.Text id="formSubjectAuthor">ABB</InputGroup.Text>
    <Form.Control
      type="text"
      placeholder="Subject ABBREVIATION"
      value={abb}
      onChange={(e) => setAbb(e.target.value)}
    />
  </InputGroup>
</Form.Group>

<ButtonGroup aria-label="Basic example" className="mb-3">
  <Button
    disabled={!flag}
    variant="success"
    onClick={(e) => {
      setYear("I");
      setFlag(true);
    }}
  >

```

```

        I
      </Button>
      <Button
        variant="danger"
        disabled={!flag}
        onClick={(e) => {
          setYear("II");
          setFlag(true);
        }}
      >
        II
      </Button>
      <Button
        disabled={!flag}
        variant="info"
        onClick={(e) => {
          setYear("III");
          setFlag(true);
        }}
      >
        III
      </Button>
      <Button
        variant="warning"
        disabled={!flag}
        onClick={(e) => {
          setYear("IV");
          setFlag(true);
        }}
      >
        IV
      </Button>
    </ButtonGroup>
    <div className="d-grid gap-2">
      <Button variant="primary" type="Submit">
        Add/ Update
      </Button>
    </div>
  </Form>
</div>
</>
);
};

export default AddSubject;

```

studentsedit.js

```
import React, { useRef, useState } from "react"
import { useAuth } from "../../contexts/AuthContext"

import { Link, useNavigate } from "react-router-dom"
import "./studentedit.css";

function Studentedit() {

  const regno = useRef()
  const regno2 = useRef()
  const passwordRef = "gcecse123"
  const { signup, logout } = useAuth()
  const [error, setError] = useState("")
  const [loading, setLoading] = useState(false)
  const [update, setUpdate] = useState(true)
  const history = useNavigate()

  async function adminsignup(e) {
    e.preventDefault()

    try {
      setError("")
      setLoading(true)
      await signup(regno.current.value+"@gmail.com", passwordRef)
      setError("Student "+regno.current.value +" added Successfully")
      regno.current.value=""
    } catch {
      setError("Failed to add student")
    }

    setLoading(false)
  }

  async function adminidel(e) {
    e.preventDefault()
    setUpdate(false)
    try {
      setError("")
      setLoading(true)
      await signup(regno2.current.value+"@gmail.com", "student")
      setError("Student "+regno2.current.value +" deleted Successfully")
    }
  }
}
```



```

        regno.current.value=""
    } catch {
        setError("Failed to delete student")
    }

    setLoading(false)
}

async function handleLogout() {
    setError("")

    try {
        await logout()
        history("/");
    } catch {
        setError("Failed to log out")
    }
}

return(
    <div className="studentedit">
        <div id="admin_header"><h1>ADMIN DASHBOARD</h1></div>
    <div className="admin-wrapper">
        <input type="checkbox" id="btn" hidden />
        <label htmlFor="btn" className="admin-menu-btn">
            <i className="fas fa-bars"></i>
            <i className="fas fa-times"></i>
        </label>
        <nav id="sidebar">
            <div className="admin-title">
                Side Menu
            </div>
            <ul className="admin-list-items">
                <li><Link to="/admin-dashboard"><i className="fas fa-home"></i>Home</Link></li>
                <li><Link to="/admin-student-edit"><i className="fas fa-sliders-h"></i>Students edit</Link></li>
                <li><Link to="/admin-staff-edit"><i className="fas fa-address-book"></i>Staffs edit </Link></li>
                <li><Link to="/admin-subjects-edit"><i className="fas fa-cog"></i>Subjects edit</Link></li>
                <li><Link to="/change-password"><i className="fas fa-stream"></i>Change Password</Link></li>
                <li><button className="logout-btn" onClick={handleLogout}><i className="fas fa-user"></i>&ensp;&ensp;Log out</button></li>

```

```

        </ul>
    </nav>
</div>
{error}&& <h3 style={{position:"absolute" ,
    top: "150px",
    left:update?"10%":"63%",
    textAlign:"center",
    color:update?"blue":"red"
}}>{error}</h3>
<div className="admin-wrapper1">
    <div className="admin-container">
        <div className="admin-simple-cards">

            <div className="admin-items">
                <h4>ADD STUDENT</h4>
                <div className="admin-cards-content">

                    <form onSubmit={adminsSignup}>
                        <input id="input_reg" type="text" placeholder="Registered
No." ref={regno} onFocus={e=>{setError('')}}/>
                        <input id="input_sub_add" type="submit" value="ADD" />
                    </form>
                </div>
            </div>

            <div className="admin-items">

                <h4>DELETE STUDENT</h4>
                <div className="admin-cards-content">
                    <form onSubmit={adminDel}>
                        <input id="input_reg" type="text"
placeholder="Registered No." ref={regno2} onFocus={e=>{setError('')}}/>
                        <input id="input_sub_del" type="submit" value="DELETE"
/>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
);
}
export default Studentedit;

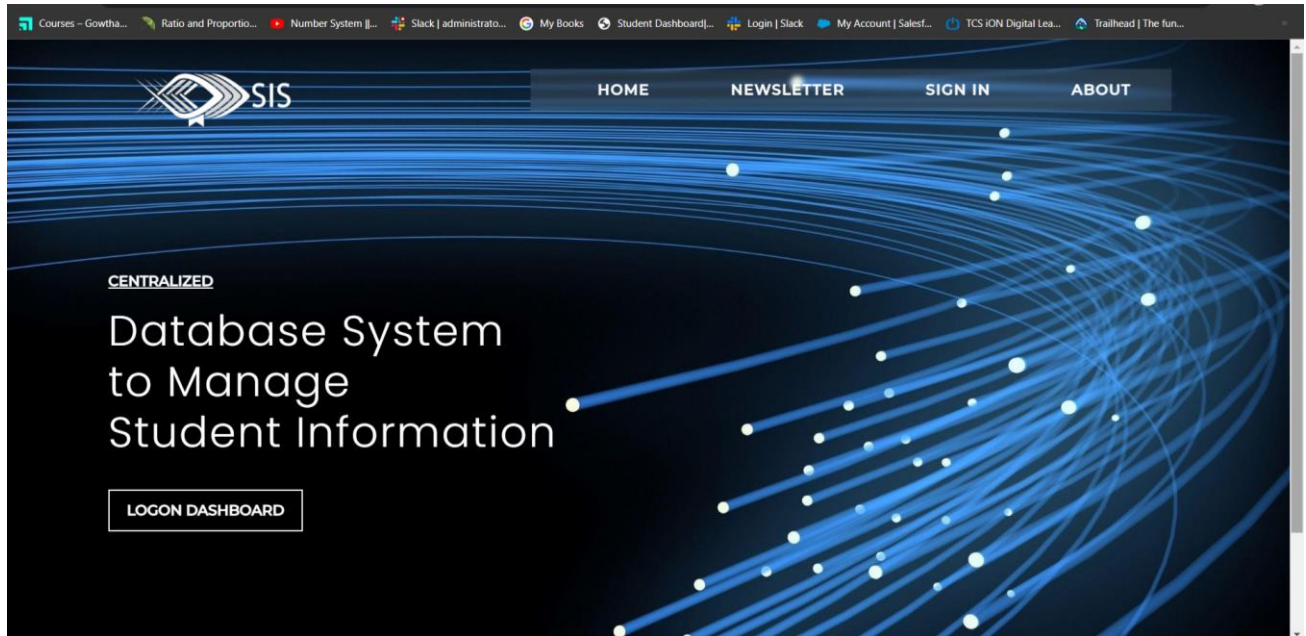
```

APPENDIX 2

SCREENSHOTS

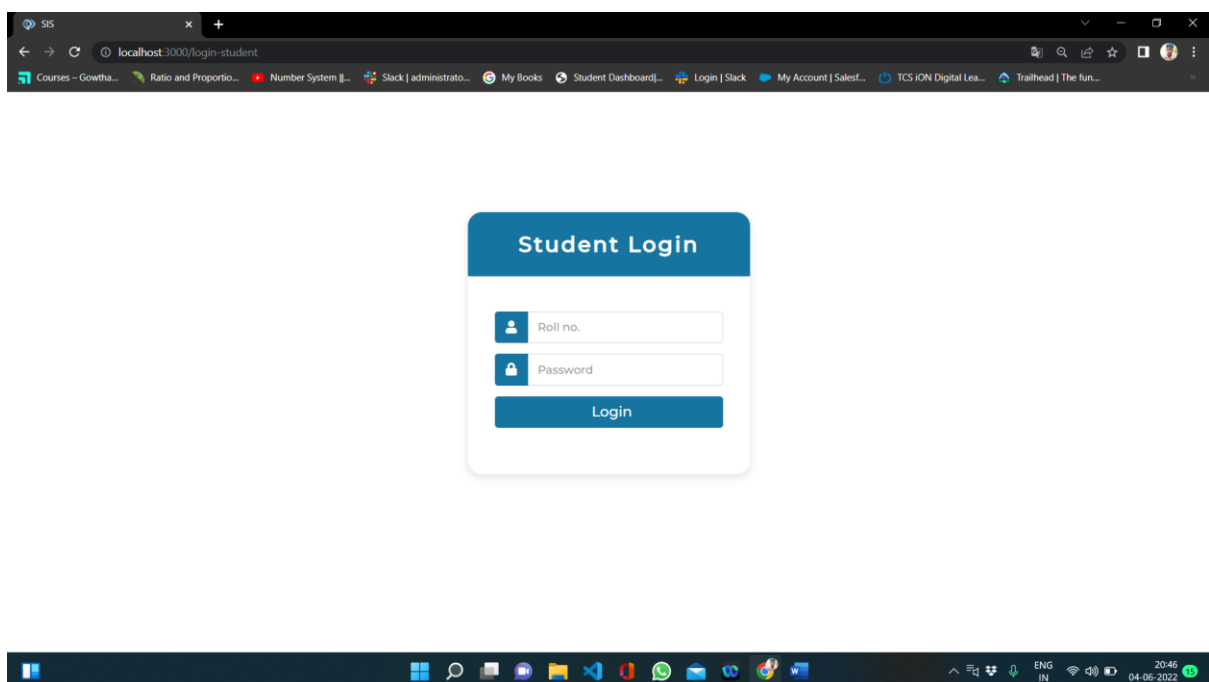
HOME PAGE

The home page is the first user interface that pops up. It contains navigation tab to newsletter, logon dashboard and about web page.



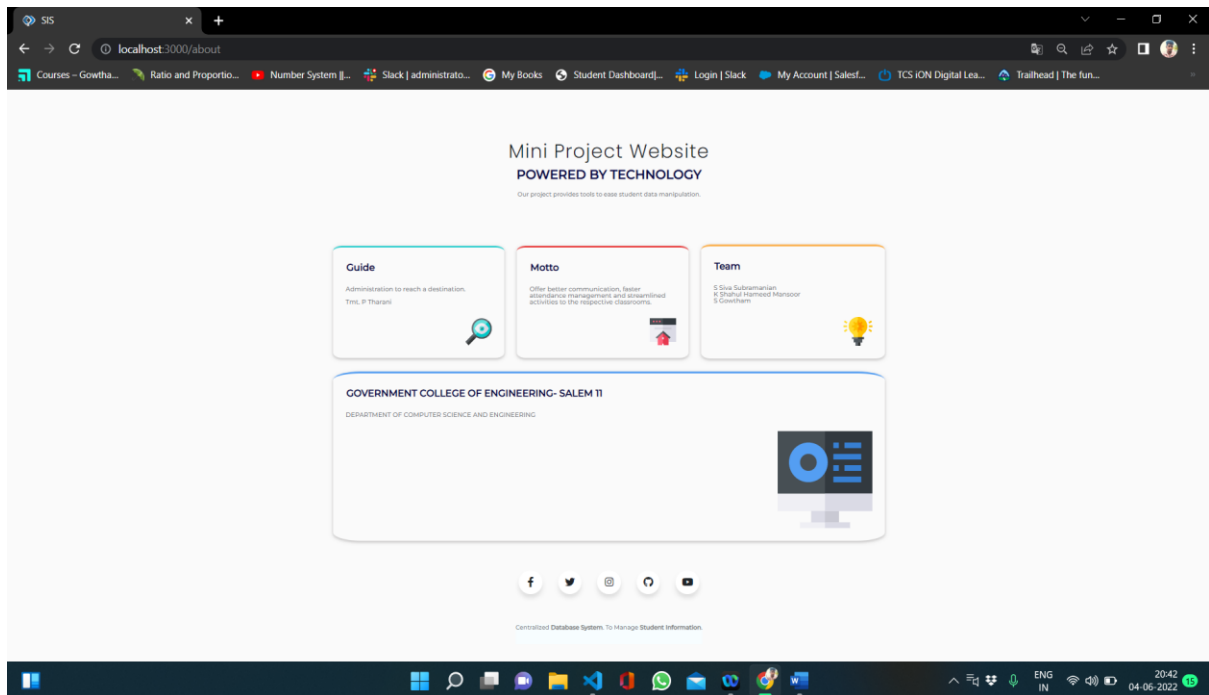
LOGIN PAGE

The login page contains navigation to student, faculty and admin login.



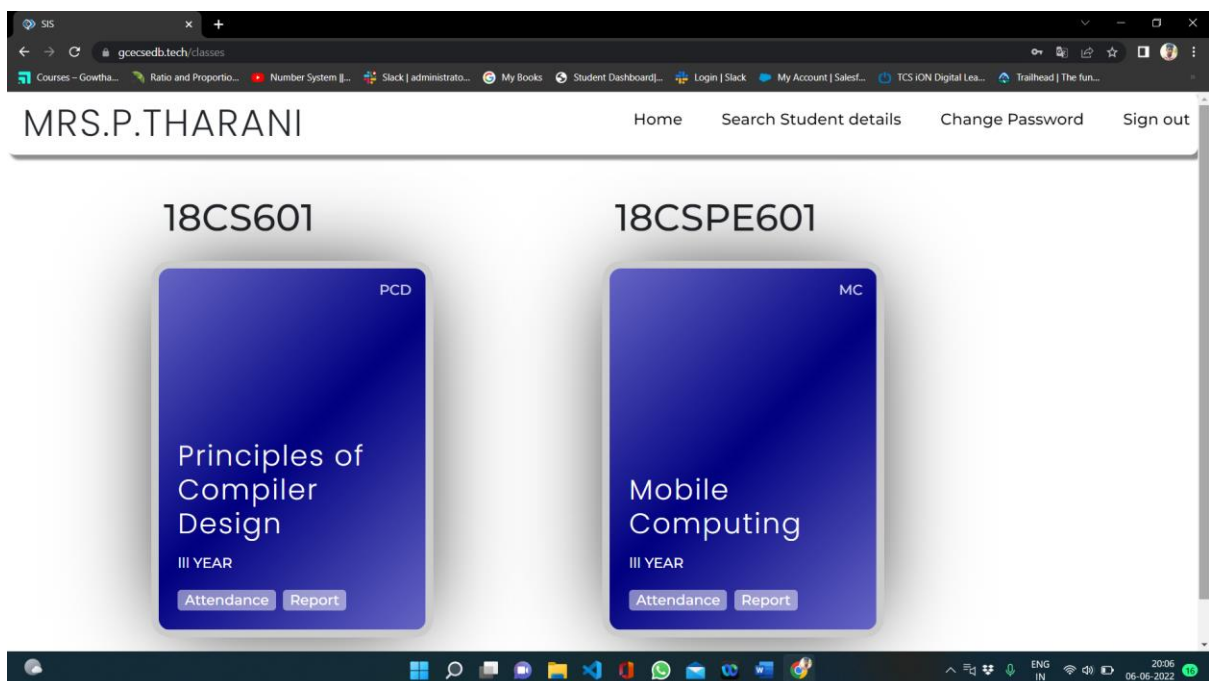
ABOUT

The about page contains description about the mini project.



FACULTY DASHBOARD

The faculty dashboard contains navigation tab to search student details, register subject wise student attendance and view attendance report.



SEARCH STUDENT DETAILS

This page can be referred by faculty to search student information.

SEARCH STUDENT DETAILS

1921000 Search

STUDENT - BIO DATA INFO

Personal Details

Full Name: SIVASUBRAMANIAN S

Date of Birth: 2022-06-05

Email: sivasubramanian29631@gmail.com

Mobile Number: +918220378233

Gender: Male

Parent phone no.: 09706422258

Identity Details

Roll no.: W

Nationality: W

Profile Image: [Placeholder]

Community: [Placeholder]

ADMIN DASHBOARD

The admin dashboard contains navigation to edit students, staff, subject details and display each of their count.

ADMIN DASHBOARD

Side Menu

- Home
- Students edit
- Faculty edit
- Subjects edit
- Change Password
- Log out

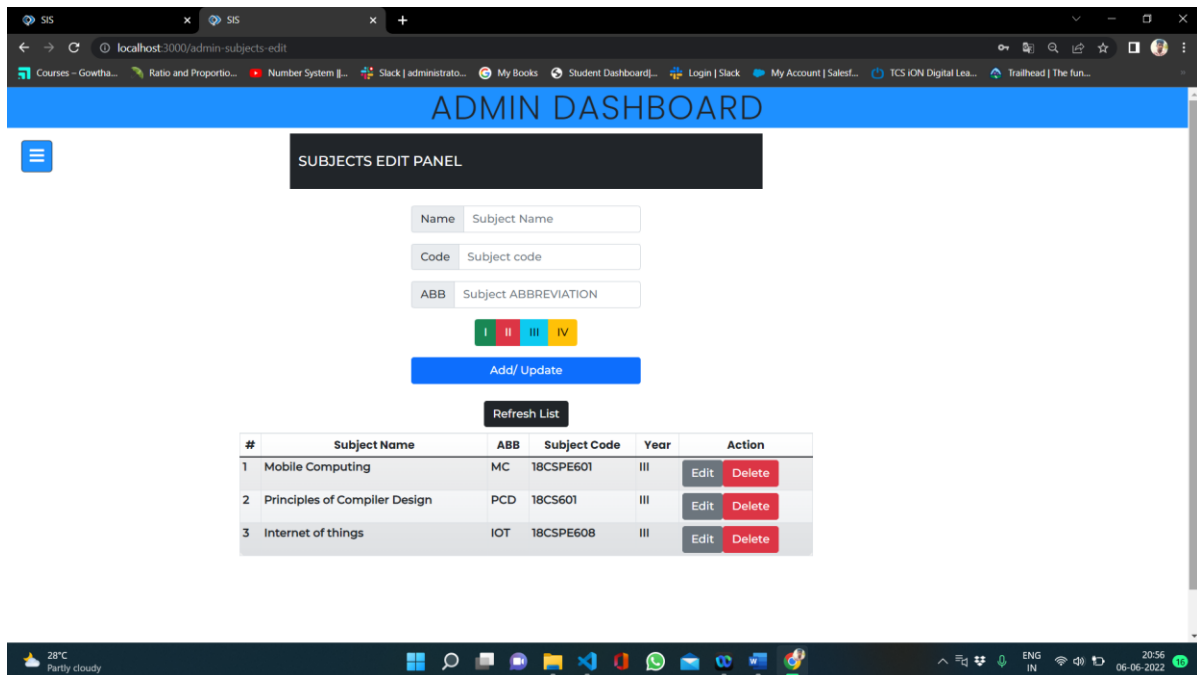
STUDENTS COUNT

FACULTY COUNT

SUBJECTS COUNT

SUBJECTS EDIT

The subject edit page contains the options to add, delete or update the course details.

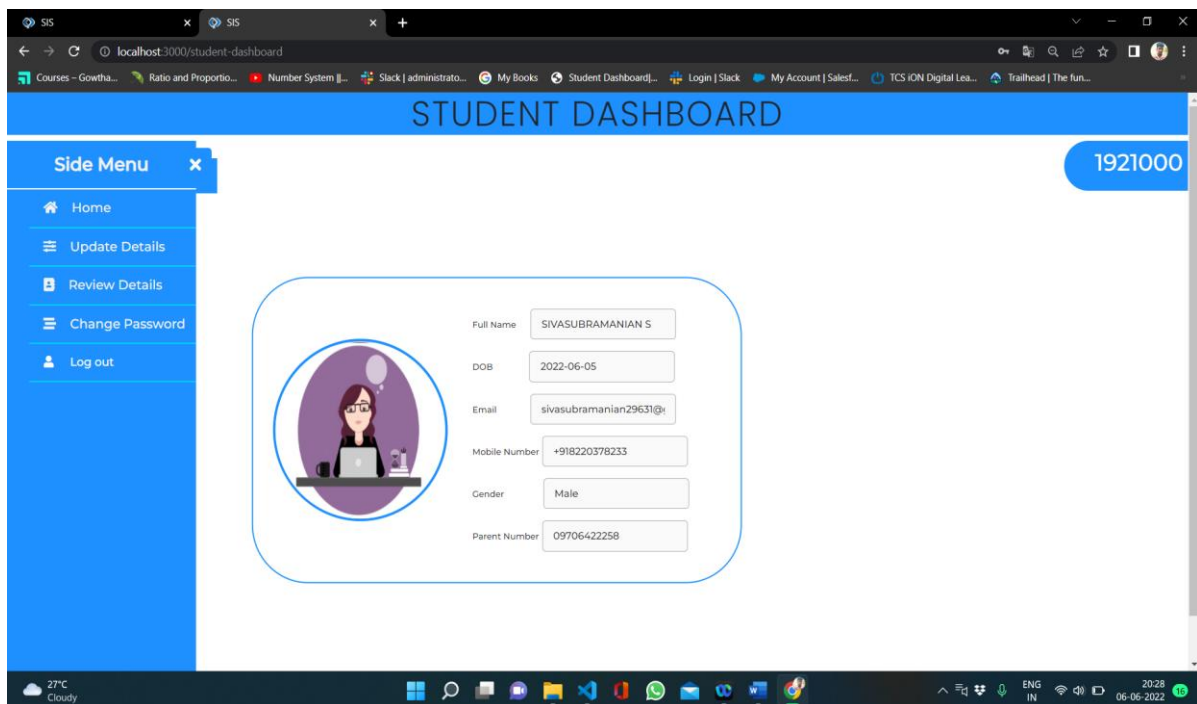


The screenshot shows the 'ADMIN DASHBOARD' with a 'SUBJECTS EDIT PANEL'. It includes input fields for 'Name' (Subject Name), 'Code' (Subject code), and 'ABB' (Subject ABBREVIATION). Below these are four colored buttons (I, II, III, IV) and an 'Add/ Update' button. A 'Refresh List' button is also present. A table lists existing subjects with columns for #, Subject Name, ABB, Subject Code, Year, and Action (Edit, Delete).

#	Subject Name	ABB	Subject Code	Year	Action
1	Mobile Computing	MC	18CSPE601	III	Edit Delete
2	Principles of Compiler Design	PCD	18CS601	III	Edit Delete
3	Internet of things	IOT	18CSPE608	III	Edit Delete

STUDENT DASHBOARD

The student dashboard contains navigation to update bio data and review their registered details.

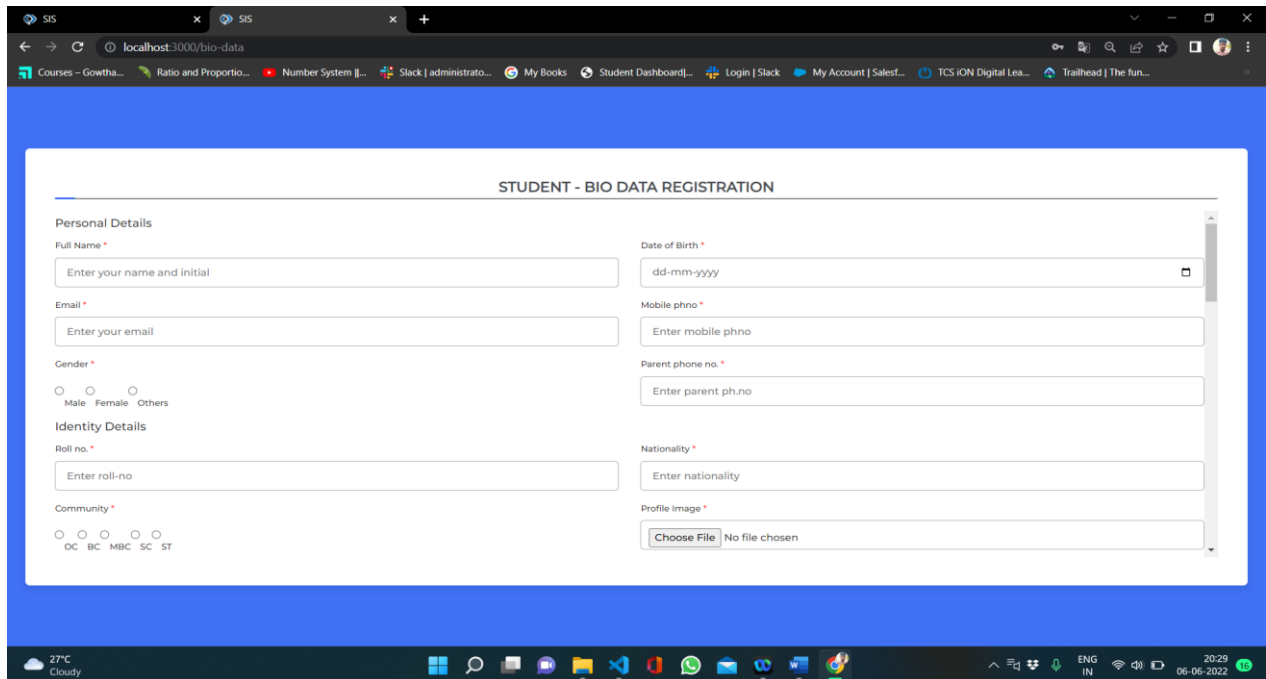


The screenshot shows the 'STUDENT DASHBOARD' for user 1921000. It features a 'Side Menu' with options: Home, Update Details, Review Details, Change Password, and Log out. The main area displays a profile card with a user avatar and a form for updating details.

Field	Value
Full Name	SIVASUBRAMANIAN S
DOB	2022-06-05
Email	sivasubramanian29631@x
Mobile Number	+918220378233
Gender	Male
Parent Number	09706422258

UPDATE DETAILS BY STUDENT

This page includes student registration form to fill up their details about personal information, address and family details.



The screenshot shows a web browser window with the URL `localhost:3000/bio-data`. The page title is "STUDENT - BIO DATA REGISTRATION". The form is divided into two main sections: "Personal Details" and "Identity Details".

Personal Details:

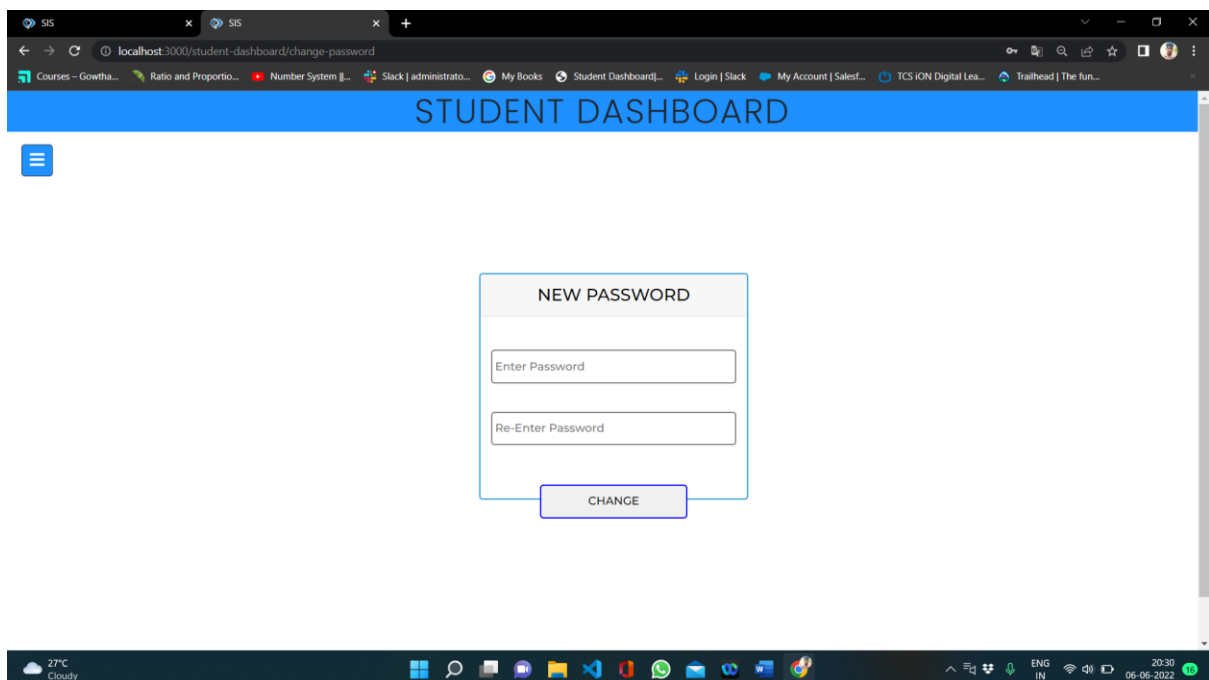
- Full Name: A text input field with a placeholder "Enter your name and initial".
- Date of Birth: A date picker field with a placeholder "dd-mm-yyyy".
- Email: A text input field with a placeholder "Enter your email".
- Mobile phno: A text input field with a placeholder "Enter mobile phno".
- Gender: Three radio buttons labeled "Male", "Female", and "Others".
- Parent phone no.: A text input field with a placeholder "Enter parent ph.no".

Identity Details:

- Roll no.: A text input field with a placeholder "Enter roll-no".
- Nationality: A text input field with a placeholder "Enter nationality".
- Community: Four radio buttons labeled "OC", "BC", "MBC", and "SC".
- Profile image: A file upload field with a "Choose File" button and the text "No file chosen".

CHANGE PASSWORD

This page is used to change student login credentials.



The screenshot shows a web browser window with the URL `localhost:3000/student-dashboard/change-password`. The page title is "STUDENT DASHBOARD". The form is titled "NEW PASSWORD" and contains two text input fields: "Enter Password" and "Re-Enter Password". Below these fields is a "CHANGE" button.

REFERENCES

1. Firebase Documentation - fundamentals

<https://firebase.google.com/docs/web/setup>

2. Firestore database - demo video

https://www.youtube.com/watch?v=QcsAb2RR52c&list=PLl-K7zZEsYlmOF_07IayrTntevxtbUxDL

3. Firebase - password authentication

<https://firebase.google.com/docs/auth/web/password-auth>

4. Node js - modules installation

<https://docs.npmjs.com/creating-node-js-modules>

5. Bootstrap - layout documentation

<https://getbootstrap.com/docs/4.1/layout/overview/>

6. React - Handling events documentation

<https://reactjs.org/docs/handling-events.html>

7. HTML CSS document

https://www.w3schools.com/html/html_css.asp