

AN IDP PROJECT REPORT

on

“Student Counselling Management System”

Submitted

By

221FA04234

M N Surya Charan

221FA04471

Mandadapu Jhansi

221FA04422

Srujana Rajavarapu

221FA04511

I. SHANMUKH

Under the guidance of

Mr. S.K. Sikinder

Associate Professor



SCHOOL OF COMPUTING & INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

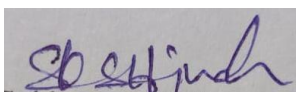
**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH Deemed
to be UNIVERSITY**

Vadlamudi, Guntur.

ANDHRA PRADESH, INDIA, PIN-522213.

CERTIFICATE

This is to certify that the Field Project entitled “Counselling Management System” that is being submitted by 221FA04234 (M N Surya Charan), 221FA04422 (Srujana Rajavarapu), 221FA04471 (Mandadapu Jhansi), and 221FA04511 (Inampudi Shanmukh) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of **Mr.Sk.Sikindar Shaik, Assistant Professor**, Department of CSE.



Sk. Sikindar Shaik
Assistant Professor, CSE



HOD,CSE



DECLARATION

We hereby declare that the Field Project entitled “**Counselling Management System**” that is being submitted by **221FA04234 (M N Surya Charan), 221FA04422 (Srujana Rajavarapu), 221FA04471 (Mandadapu Jhansi), and 221FA04511 (Inampudi Shanmukh)** in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of **Mr. Sk. Sikindar Shaik, Assistant Professor, Department of CSE.**

By

**221FA04234 (M N Surya Charan),
221FA04422 (Srujana Rajavarapu),
221FA04471 (Mandadapu Jhansi),
221FA04511 (Inampudi Shanmukh)**

ABSTRACT

In modern educational institutions, student counseling plays a vital role in academic and personal development. However, managing counseling records and student details manually poses several challenges for faculty members. The absence of a centralized system leads to inefficiencies in tracking counseling sessions, searching student details, approving updates, and maintaining attendance records.

This project addresses these challenges by providing a **web-based Counseling Management System** tailored for faculty and students. The system is designed to enhance efficiency, transparency, and accessibility in managing counseling processes. It offers faculty (counselors) a personalized dashboard where they can log in using their institutional credentials, view assigned students, search for student information, track attendance, and manage session records. Faculty can also approve or reject student requests for updating personal details.

The platform supports **real-time notifications** and ensures secure access using **JWT or OAuth-based authentication**, along with encrypted storage of sensitive data. With a responsive and user-friendly interface built using React.js and Bootstrap, this ensures usability across various devices. Students benefit from a streamlined process to request updates and access their information. Key features include role-based access control, performance support for up to 500 users, and scalability for future growth. The backend, powered by Node.js and Express.js, integrates with MongoDB for robust data handling.

By automating and digitizing the counseling process, this **empowers faculty with better management tools and provides students with a transparent and efficient platform**, ultimately fostering a more supportive academic environment.

TABLE OF CONTENTS

Contents

LIST OF FIGURES	3
ABBREIVATIONS	4
1. INTRODUCTION	6
1.1 Introduction	6
1.2 Literature Survey	7
1.2 Project Background.....	7
1.3 Objective	7
1.4 Project Description.....	8
2. SOFTWARE REQUIREMENTS SPECIFICATION	10
2.1 Requirement Analysis	10
2.2 Problem Statement	10
2.3 Software Requirement Specification	10
2.4 Software Requirements	12
2.5 Hardware Requirements	12
2.6 Functional Requirements (Modules)	12
2.7 Non-Functional Requirements:	13
2.8 External Interface Requirements.....	14
2.9 Feasibility study	14
3. ANALYSIS & DESIGN	17
3.1 Introduction	17
3.2 System Overview	18
3.3 System Architecture	18
3.4 Data Design	19
4. MODELING	23
4.1 Design 23	
5. IMPLEMENTATION.....	30
5.1 Sample Code.....	30
5.2 Screen Captures	39
6. TESTING.....	44
6.1 Software Testing	44
6.2 Black box Testing	44
6.3 White box Testing.....	44
6.4 Performance Testing	44

6.5 Load Testing.....	44
6.6 Manual Testing.....	44
7. RESULTS AND CHALLENGES	46
7.1 Results 46	
7.2 Challenges	46
8. CONCLUSION	48
8.1 Conclusions	48
8.2 Scope for future work.....	48
8.3 Limitations	48
BIBLIOGRAPHY	49

LIST OF FIGURES

Figure 3-1	A Directory structure for a java project auto generated by maven	28
Figure 3-2	Web Application Architecture Diagram...	30
Figure 3-3	Java Based web application architecture.....	32
Figure 4-1	Use Case Diagram for system.....	38
Figure 4-2	Sequence Diagram for Customers module.....	39
Figure 4-3	Sequence Diagram for Vendor module	40
Figure 4-4	Activity Diagram for CMS.....	40
Figure 4-5	Class Diagram for Offline CMS.....	41
Figure 4-6	Deployment Diagram of the System.....	42
Figure 4-7	ER Diagram.....	43
Figure 5-1	Home Screen Activity.....	54
Figure 5-2	Vendor Login Activity.....	55
Figure 5-3	Customer Login Activity	55
Figure 5-4	Vendor Dash board Activity	56
Figure 5-5	Vendor Details activity	56
Figure 5-6	List of pending orders Activity	56
Figure 5-7	Accepting pending orders	57
Figure 5-8	Rejecting pending orders	57
Figure 5-9	Customer dash board Activity.....	58
Figure 5-10	Placing the order Activity	58
Figure 5-11	Menu activity	59
Figure 5-12	Menu item activity.....	59
Figure 6-1	Test Case for Inserting new Record	62
Figure 6-2	Test Case for Updating order status	63
Figure 6-3	Coverage report for Junit and Jmockit Test Cases	63

ABBREIVATIONS

- ADS : ANDROID DATA SYNCHRONIZATION
- SDK : SOFTWARE DEVELOPMENT KIT
- ADT : ANDROID DEVELOPMENT KIT
- API : APPLICATION PROGRAMMING INTERFACE
- AOSP : ANDROID OPEN SOURCE PROJECT
- AVD : ANDROID VIRTUAL DEVICE
- CRM : CUSTOMER RELATIONSHIP MANAGEMENT
- UI : USER INTERFACE
- JSON : JAVA SCRIPT OBJECT NOTATION
- XAMPP : WINDOWS/LINUX APACHE MYSQL PERL PHP
- PHP : HYPERTEXT PREPROCESSOR
- IDE : INTEGRATED DEVELOPMENT ENVIRONMENT
- HTML : HYPER TEXT MARKUP LANGUAGE
- JDK : JAVA DEVELOPMENT KIT
- XML : EXTENSIBLE MARKUP LANGUAGE
- WIFI : WIRELESS FIDELITY
- AVD : ANDROID VIRTUAL DEVICES
- ADB : ANDROID DEBUG BRIDGE

CHAPTER - 1
INTRODUCTION

1. INTRODUCTION

1.1 Introduction

The ongoing project is centered on creating a user web platform to improve the organization and delivery of student counseling services efficiently and effectively. The main goal of this initiative is to simplify the counseling procedure, for students and counselors. In today's setting it's common for students to encounter difficulties in booking and attending counseling appointments resulting in delays, in accessing the necessary assistance. This initiative aims to tackle these challenges by offering a system that enables students to conveniently schedule appointments, avail resources and interact with counselors regardless of their location or the time of day.

This online platform allows students to discover counseling options and choose the counselor who suits them best. With scheduled appointments, in place counselors get. Ready to offer help. This initiative seeks to enhance the counseling journey by guaranteeing students access beneficial support.

In this project, we are going to create a website where there are three kinds of users namely Students, faculty and Admin.

1. Students:

Students are the primary users who utilize the system to book counseling appointments, track their academic and personal progress, and communicate with counselors through a secure messaging system. They can access session notes, receive feedback, and manage their counseling history, ensuring continuous support and guidance throughout their academic journey.

2. Faculty:

Counselors can access student profiles, schedule appointments, and provide structured feedback after each session. Additionally, the system enables real-time communication between counselors and students, ensuring that students receive timely advice and support when needed.

3.Admin:

Admins have access to system analytics, which help them monitor the number of counseling sessions conducted, track student engagement, and generate reports for institutional assessment.

1.2 Literature Survey

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

- As the project is under the "**Agile Development Model**," the "**Agile for Beginners WBT**" course has been done.
- Took the "**Prodigious Git FP**" course as extra learning to know about the Version control system.

1.2 Project Background

The student counseling services are currently facing significant challenges, primarily due to outdated management practices. To enhance the effectiveness and accessibility of these services, it is essential to modernize our approach. By automating administrative tasks and utilizing cloud-based solutions, we can streamline operations and improve the overall experience for students seeking support. This transformation will help restore the efficiency and reputation of our counseling services.

1.3 Objective

The objective of the system is to automate all the activities related to student counselling services, from scheduling appointments to tracking student progress. The system should maintain a detailed account of all counselling sessions conducted and the outcomes achieved. In addition to this, it should also manage the daily schedules and availability of counsellors. The system should provide an interface for generating reports on student progress and counsellor performance. Several inquiry facilities should also be provided to view student profiles, counselling history, feedback, and upcoming appointments.

1.4 Project Description

In this section all features in application are explained in brief.

1) Student

- **Managing student profiles** and reviewing student history to provide personalized guidance.
- **Scheduling and conducting counselling sessions** (online or offline).
- **Updating session notes** and tracking student progress.
- **Providing recommendations** on academic or personal challenges.
- **Engaging in real-time communication** with students for immediate support.
- **Generating reports** on student development and counselling effectiveness.

CHAPTER - 2

SOFTWARE REQUIREMENT

SPECIFICATION

2. SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Requirement Analysis

For the purpose of easy access and portability we proposed to develop a web based system with windows or ubuntu as a platform because they are mostly used operating systems on the market . As a part of this system we are going to develop web-based software in which data can be accessed and retrieved easily. The required documents for these processes are as follows.

1. Problem statement
2. Data flow diagrams
3. Use case diagram
4. Other UML diagrams.

The above mentioned documents gives us diagrammatical view of the system what we are going to develop.

2.2 Problem Statement

The problem statement concentrates on data storage in a MongoDB database and syncing the data to the MongoDB server.

2.3 Software Requirement Specification

The project is developed in Java Programming Language by using the visual studio. We use the Azure Extension which includes a variety of custom tools that help us to deploy web applications on the azure cloud platform. At the Server-side Apache Tomcat Server is used. Apache Web Server, MongoDB is used at the backend and angular is used for the front end.

2.3.1 Purpose

The purpose of this document is to present a detailed description of “**Student-Counselling Management System**” application. It will explain the purpose and features of the system that it will provide, constraints under which it must operate and how the system will react. The document also describes the non functional requirements of the system.

2.3.2 Scope of the project

The **Student-Counselling Management System** is designed to enhance the counseling experience for students, faculty, and administrators by providing an

efficient and structured platform for communication, appointment scheduling, and progress tracking. The system aims to improve student well-being by ensuring timely and effective counseling support.

1.Student:

The system allows students to register, schedule counseling appointments, communicate with counselors via real-time messaging, and track their progress. It ensures secure access to session histories and personalized guidance, helping students manage academic and personal challenges effectively.

2.Counselors:

Counselors can manage student profiles, schedule or modify appointments, track session histories, and provide feedback. The system offers tools for efficient student monitoring, seamless communication, and data-driven decision-making to improve counseling outcomes. history

2.3.3 Technologies Used

➤ REACT

React focuses on making web development efficient and intuitive, emphasizing component-based architecture, reusability, and a virtual DOM for performance optimization. Applications built with React can be rendered on the server or the client, and can be used to create both web and mobile applications through frameworks like React Native.

➤ HTML

Hypertext Markup Language is the main markup language for displaying web pages and other information that can be displayed in a web browser.

➤ JAVASCRIPT

JavaScript is a scripting language commonly implemented as part of a web browser in order to create enhanced user interfaces and dynamic websites.

➤ MongoDB

MongoDB is one of the most popular NoSQL database management systems on the web. MongoDB is used for internet applications as it provides flexibility and scalability. MongoDB was developed to manage large volumes of unstructured data efficiently, offering a schema-less design that overcomes the limitations of traditional relational databases..

2.3.4 Overview

The application is based on both windows and ubuntu platform an application

which communicates with the server allowing the collection of sales visit data offline and then syncs the data with that of the corporate server when there is access to the internet.

2.4 Software Requirements

The software interface is the operating system, and application programming interface used for the development of the software.

- Frontend Framework - React.js with Vite
- Backend Framework - Express.js with Node.js
- Database - MongoDB
- Authentication - JWT-based authentication
- State Management - Redux or Context API
- Styling - Tailwind CSS / Material UI
- Hosting - Vercel (Frontend), Render/Heroku (Backend)

2.5 Hardware Requirements

CLIENT			
OPERATING SYSTEM	SOFTWARE	DISK SPACE	RAM
Any operating system	Any Web Browser	Minimum 250 MB	256 MB

Table 2.1 Client Requirements

SERVER				
OPERATING SYSTEM	SOFTWARE	PROCESSOR	RAM	DISK SPACE
Ubuntu 12.04 LTS	Apache 2.2.22 MySQL 5.5.2	Intel Xeon I CPU E31220	256Mb	Minimum 250Mb

Table 2.2 Server Requirements

2.6 Functional Requirements (Modules)

1. **User Authentication & Role-Based Access:** Secure login for students, counselors, and administrators.
2. **Student Profile Management:** Maintain student records, counseling history, and progress tracking.

3. **Appointment Scheduling:** Allow students to book counseling sessions with available counselors.
4. **Counseling Session Management:** Enable counselors to conduct and document sessions.
5. **Messaging System:** Real-time chat functionality for student-counselor communication.
6. **Admin Control Panel:** Allow administrators to manage users and monitor the system.
7. **Notifications & Reminders:** Alerts for upcoming appointments and important messages.
8. **Data Security & Access Control:** Ensure privacy through encryption and role-based permissions.

2.7 Non-Functional Requirements:

2.7.1 Performance requirements:

Performance requirements define acceptable response times for system functionality. The load time for user interface screens shall take no longer than 10 seconds. The log in information shall be verified within 10 seconds. Queries shall return results within 10 seconds.

2.7.2 Design Constraints:

The online canteen management shall be a stand-alone system running in a Windows environment. The system shall be developed using ANGULAR and connected to Oracle database.

2.7.3 Standards Compliance:

There shall be consistency in variable names within the system. The graphical user interface shall have a consistent look and feel.

2.7.4 Availability:

The system shall be available during normal Canteen operating hours.

2.7.5 Portability:

The Canteen Management System shall run in any Microsoft Windows environment that contains browser and be able to available to all users all the time while they have the internet.

2.7.6 Reliability:

Since the application is being developed through java, the most famous, efficient and reliable language, so it is reliable in every aspect until and unless there is an error in the programming side. Thus the application can be a compatible and reliable one.

2.8 External Interface Requirements

2.8.1 User Interface

A critical aspect of this project was examining how the app would look and its usability. The layout of web applications is defined in a HTML and JSP files.

When the App is launched, it displays the First Screen- an activity that contains the layout displays buttons for Owner activity and Customer activity. The application has a user-friendly interface.

2.9 Feasibility study

A key part of the preliminary investigation that reviews anticipated costs and benefits and recommends a course of action based on operational, technical, economic, and time factors. The purpose of the study is to determine if the systems request should proceed further.

2.9.1 Organisational Feasibility

The application would contribute to the overall objectives of the organization. It would provide a quick, error free and cost effective solution to the current process CRM marketing. It would provide a solution to many issues in the current system. As the new system is flexible and scalable it can also be upgraded and extended to meet other complex requirements which may be raised in the future. However it is up to the organization to upgrade or extend it.

2.9.2 Economic Feasibility

The project is economically feasible as it only requires a computer with Any operating system. The application is free to download once released into market. The users should be able to connect to internet through computer and this would be the only cost incurred on the project.

2.9.3 Technical Feasibility

To develop this application, a high speed internet connection, a database server,

a web server and software are required. The current project is technically feasible as the application was successfully deployed on aws

2.9.4 Behavioural Feasibility

The application is behaviourally feasible since it requires no technical guidance, all the modules are user friendly and execute in a manner they were designed to.

CHAPTER - 3
ANALYSIS & DESIGN

3. ANALYSIS & DESIGN

3.1 Introduction

3.1.1 Purpose

In this section the purpose of the document and the project is described.

3.1.1.1 Document Purpose

An SDD is a representation of a software system that is used as a medium for communicating software design information.

3.1.1.2 Project Purpose

The **Student-Counselling Management System** aims to streamline and digitalize the counseling process in educational institutions. It provides students with an efficient platform to schedule appointments, communicate with counselors, and track their progress. Counselors can manage student records, monitor sessions, and offer personalized guidance, while administrators oversee system operations and ensure data security. The system enhances accessibility, improves communication, and ensures efficient management of counseling services, ultimately fostering student well-being and academic success.

3.1.2 Scope

In this section the scope of the document and the project is explained in brief.

3.1.2.1 Document Scope

This document contains a thorough description of the high level architecture that will be used in developing the system. Communicating at a purposefully high level, it will only form the basis for the Software Detailed Design and implementation. However, the SDD itself will not be in sufficient detail to implement the code. It will convey the overall system design of the system, the user interface design and higher level module design (including android development tools) and the architecture of the Linux kernel and the working of the Dalvik Virtual Machine. Design details that will not be included in the SDD are:

- Low level classes that will be used in the implementation. The full description of the implementation of each module is not needed, but the public modules that will be interfaced will be described.
- Exact detailed description of interactions within each module.

3.1.2.2 Project Scope

It enables secure authentication, appointment scheduling, real-time messaging, and progress tracking to ensure seamless interactions. The system enhances data management, improves communication, and ensures accessibility through a user-friendly web interface. Future expansions may include AI-driven analytics, mobile app integration, and academic performance tracking to further enhance its capabilities.

3.2 System Overview

Cloud deployment framework uses certain development tools which are as follows:

3.2.1.1 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your 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).

3.3 System Architecture

3.3.1 Architectural Design

Web application architecture is a framework connecting different elements to enable a web experience. It is the backbone of our daily internet browsing: typing in a URL and viewing and interacting with the website while the browser communicates with the server is one of the ways to describe what is web application architecture.

3.4 Data Design

Collection in DataBase :

- student
- faculty
- date
- due
- status

Definition

1. Faculty Collection

```
{  
  
  _id: ObjectId,  
  
  name: String,  
  
  email: String,  
  
  password: String,  
  
  role: String,  
  
  department: String,  
  
  students: [ObjectId], // References to Student documents  
  
  createdAt: Date,  
  
  updatedAt: Date  
  
}
```

2. Student Collection

```
{  
  
  _id: ObjectId,  
  
  name: String,  
  
  major: String,  
  
  year: String,
```

```
status: String, // 'On Track', 'At Risk', 'Critical'

gpa: Number,

attendance: Number,

flags: [String], // Array of flag strings

faculty: ObjectId, // Reference to Faculty document

createdAt: Date,

updatedAt: Date

}
```

3. Session Collection

```
{

  _id: ObjectId,

  date: String,

  title: String,

  notes: String,

  type: String, // 'In-Person', 'Virtual'

  faculty: ObjectId, // Reference to Faculty document

  createdAt: Date,

  updatedAt: Date

}
```

4. Task Collection

```
{

  _id: ObjectId,

  title: String,

  priority: String, // 'Low', 'Medium', 'High'

  due: String,

  faculty: ObjectId, // Reference to Faculty document
```



```
    createdAt: Date,  
    updatedAt: Date  
}
```

5. Message Collection

```
{  
  _id: ObjectId,  
  text: String,  
  date: String,  
  faculty: ObjectId, // Reference to Faculty document  
  createdAt: Date,  
  updatedAt: Date  
}
```

CHAPTER - 4
MODELING

4. MODELING

4.1 Design

Requirements gathering followed by careful analysis leads to a systematic Object Oriented Design (OOAD). Various activities have been identified and are represented using Unified Modeling Language (UML) diagrams. UML is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development.

4.1.1. Use Case Diagram

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical over view of the functionality of the system in terms of actors, which are persons, organizations or external system that plays a role in one or more interaction with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

In this application there is only actor – soldier and below is the use case diagram of this application.

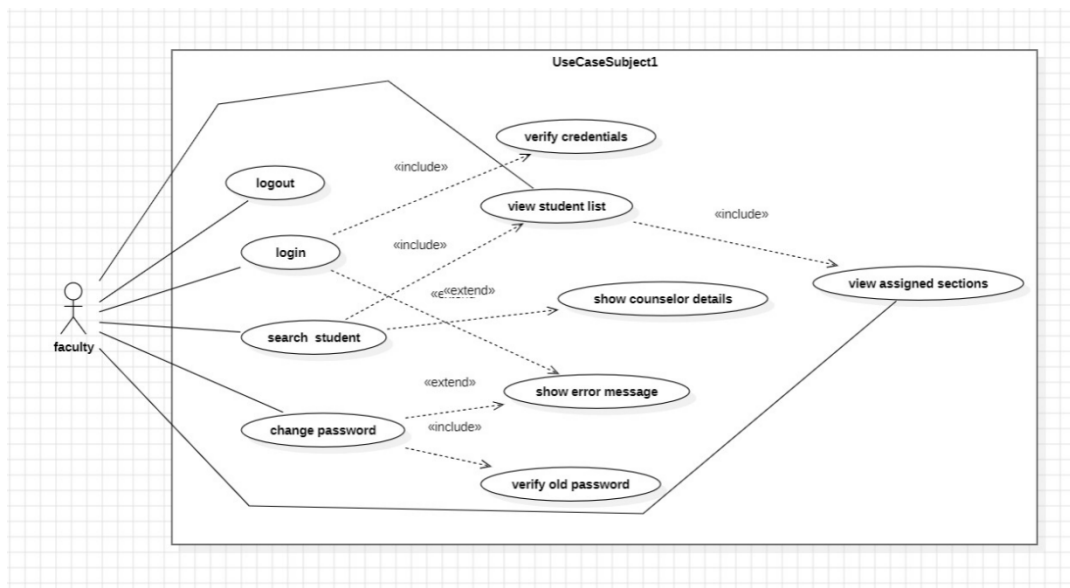


Figure 4-1 Use Case Diagram for System

4.1.2 Sequence Diagram

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom (i.e. Lower equals later).

A popular use for them is to document the dynamics in an object-oriented system. For each key, collaboration diagrams are created that show how objects interact in various representative scenarios for that collaboration.

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a numbers of lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, combined fragment, interaction use, state invariant, continuation, destruction occurrence.

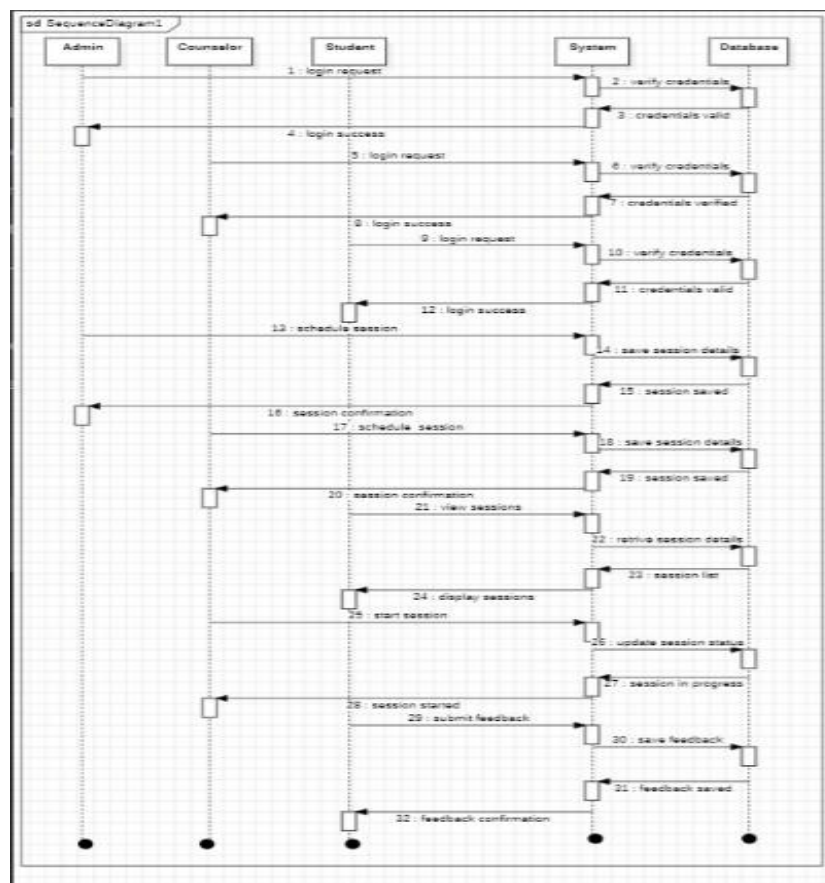


Figure 4-2 Sequence Diagram for Customers Module

4.1.3 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc. Activity is a particular operation of the system.

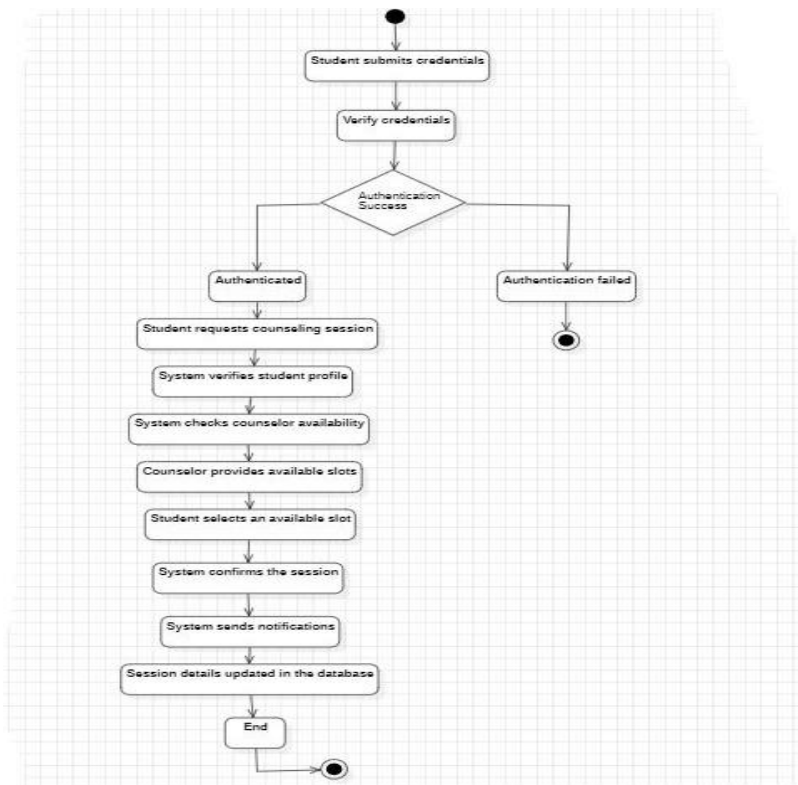


Figure 4-4 Activity Diagram for Counselling Management System

4.1.4 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

The class diagram is the main building block of object-oriented Modelling. It is used both for general conceptual modelling of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

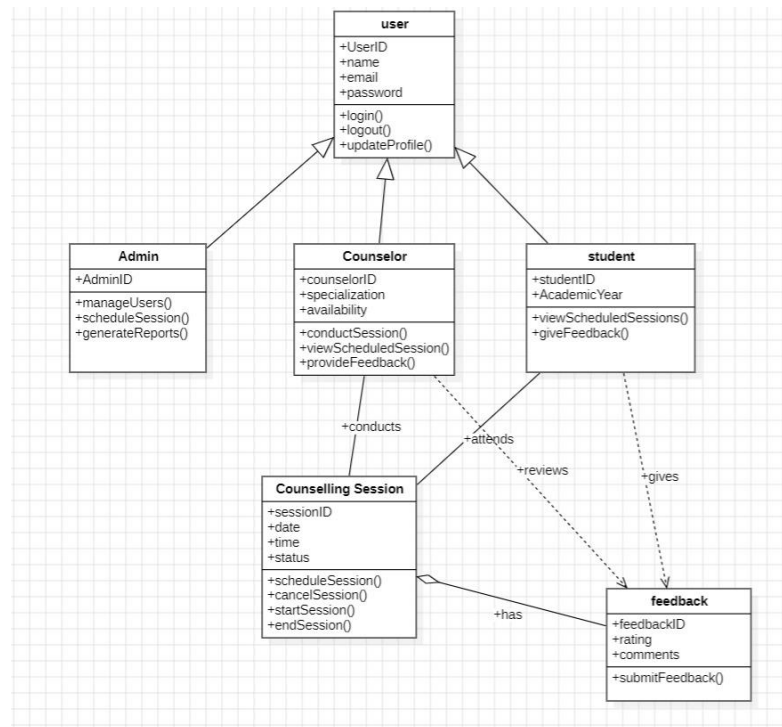


Figure 4-5 Class Diagram for CMS

4.1.5 Deployment Diagram

Deployment diagram shows execution architecture of systems that represent the assignment (deployment) of software artifacts to deployment targets (usually nodes). Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Artifacts represent concrete elements in the physical world that are the result of a development process and are deployed on nodes.

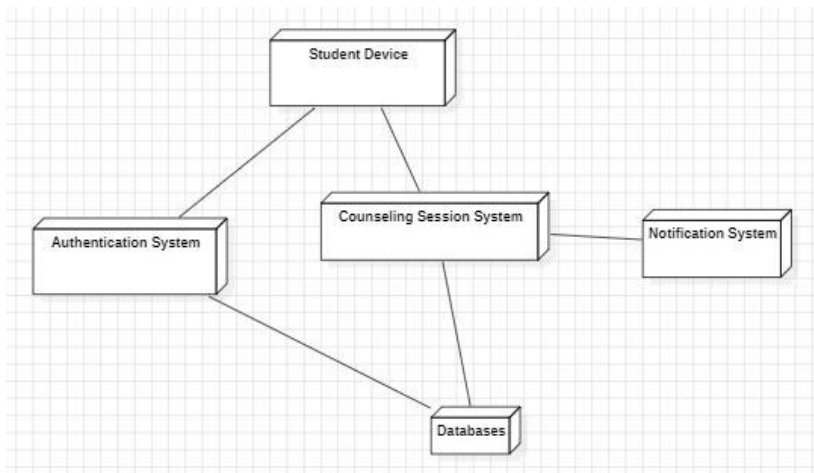


Figure 4-6 Deployment Diagram of the system

4.1.6 ER Diagram

An ER model is an abstract way to describe a database. Describing a database usually starts with a relational database, which stores data in tables. Some of the data in these tables point to data in other tables - for instance, your entry in the database could point to several entries for each of the phone numbers that are yours. The ER model would say that you are an entity, and each phone number is an entity, and the relationship between you and the phone numbers is 'has a phone number'. Diagrams created to design these entities and relationships are called entity–relationship diagrams or ER diagrams.

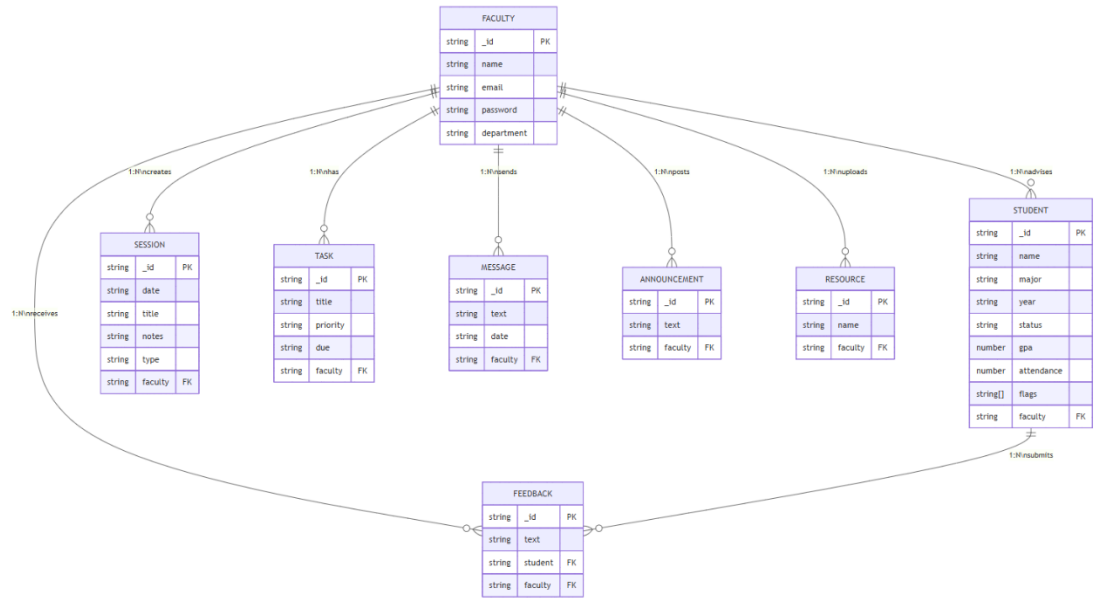


Figure 4-7 ER Diagram

CHAPTER - 5
IMPLEMENTATION

5. IMPLEMENTATION

5.1 Sample Code

5.1.1 Code for index page

```
<body>
  <img class="img1" src ="https://raw.githubusercontent.com/aparnaraghu
nath/practice1-
hexaware/master/logo.png?_sm_au_=iVVSsLNSRHM0Pb0J31QqjK0Jq2pjG" alt ="f
ood image error">

  <font color="red"><h1 class = "h1" style='margin-
left:10px'>THE STICKY FINGERS </h1></font>
  <hr>
  <button type="button" style='margin-
left:550px' > <a [routerLink]="['/cms-main',
  {outlets: {'data': ['vendor-
login']}]}">Vendor Login </a> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</button>

  <button type="button" style='margin-
left:100px'> <a [routerLink]="['/cms-main',
  {outlets: {'data': ['customer-
login']}]}">Customer Login</a> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</button>
  <hr/>
  <img class="center" size="100px" src ="https://serving.photos.photobox
.com/65822387ddb4203bf59f10ed865dec10030fb121f3695dcb64f92ac091563148f1
8e1bcd.jpg" alt ="food image error">
  <router-outlet name='data'></router-outlet>
</body>
<style>
  body{
background-color:black;
background-position: center;
background-size: cover;
}

  h1{
    text-align: center;
    color: black(189, 45, 69);
    font-size: 300%;
  }

  .center {
background-color: black;
display: block;
margin-left: auto;
margin-right: auto;
width:50%;
height: 80%;
}
```

```

.img1 {
  float:left;
  width: 10%;
  height: 80px;
}
.footer {
  position: fixed;
  left: 0;
  bottom: 0;
  height: 5%;
  width: 100%;
  background-color:black;
  color: white;
  text-align: center;
}
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 15px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}
</style>

```

5.1.2 Code for Student login

```

<body>
<form method="post" align = center>
  <font color="white"><p>VENDOR LOGIN</p></font>
  <font color="white">Phone Number :</font>
  <input type="text" name="PhoneNo" [(ngModel)]="PhoneNo" /> <br/><br />
  <font color="white">Password : </font>
  <input type="password" name="passWord" [(ngModel)]="passWord" /> <br/><br/>
  <input type="button" value="Login" (click)="validate()" />
</form>
<p *ngIf="count > 1">
  <b>Invalid Credentials...</b>
</body>
<style>
  body{
    background-color:black;
    background-position: auto;
    background-size: cover;
  }
</style>

```

5.1.3 Code for counsellor login

```
<body>
<form method="post" align = center>
  <font color="white"><p>CUSTOMER LOGIN</p></font>
  <font color="white">Phone Number :</font>
  <input type="text" name="PhoneNo" [(ngModel)]="PhoneNo" /> <br/><br/>
/>
  <font color="white">Password :</font>
  <input type="password" name="passWord" [(ngModel)]="passWord" /> <br/><br/>
  <button class="button button1" value="Login" (click)="validate()"><br/> LOGIN </button>
  <br>
  <br>
</form>
<p *ngIf="count > 1">
  <b>Invalid Credentials...</b>
</p>
</body>
<style>
  body{
    background-color :black;
    background-position: auto;
    background-size: cover;
  }
  .button {
    background-color: rgb(91, 209, 91);
    border: none;
    color: white;
    padding: 1px 5px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    transition-duration: 0.4s;
    cursor: pointer;
  }
  .button1 {
    background-color: white;
    color: black;
    border: 2px solid #4CAF50;
  }
  .button1:hover {
    background-color: #4CAF50;
    color: white;
  }
</style>
```

5.1.4 Code for Student dash board

[illegible]

5.1.5 Code for Counsellor dash board

[illegible]

```

text-align: center;
text-decoration: none;
display: inline-block;
}

.img1 {
float:right;
width: 10%;
height: 75px;
}
</style>

```

5.1.6 Code for pom.xml file

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.hexaware</groupId>
  <artifactId>MLP195</artifactId>
  <version>0.0.1</version>
  <packaging>war</packaging>
  <name>MLP195</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <jersey.version>2.25</jersey.version>
    <build.number>SNAPSHOT</build.number>
  </properties>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-enforcer-plugin</artifactId>
        <version>1.4.1</version>
        <executions>
          <execution>
            <id>coverage.check</id>
            <goals>
              <goal>enforce</goal>
            </goals>
            <phase>test</phase>
            <configuration>
              <rules>
                <requireFilesDontExist>
                  <files>
                    <file>target/coverage.check.failed</file>
                  </files>
                </requireFilesDontExist>
              </rules>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>

```

```

        </rules>
    </configuration>
</execution>
</executions>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.17</version>
    <executions>
        <execution>
            <phase>process-sources</phase>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <failOnError>true</failOnError>
        <includeTestSourceDirectory>true</includeTestSourceDirector
y>
        <configLocation>checkstyle.xml</configLocation>
    </configuration>
</plugin>
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>findbugs-maven-plugin</artifactId>
    <version>3.0.5</version>
    <configuration>
        <effort>Max</effort>
        <threshold>Low</threshold>
        <xmlOutput>true</xmlOutput>
    </configuration>
    <executions>
        <execution>
            <phase>process-sources</phase>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-pmd-plugin</artifactId>
    <version>3.8</version>
    <executions>
        <execution>

```



```

        <phase>process-sources</phase>
        <goals>
            <goal>check</goal>
            <goal>cpd-check</goal>
        </goals>
    </execution>
</executions>
</plugin>
</plugins>
<pluginManagement>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.20</version>
            <configuration>
                <systemPropertyVariables>
                    <coverage-output>html</coverage-output>
                    <coverage-metrics>all</coverage-metrics>
                    <coverage-
classes>com.hexaware.MLP195.factory.*,com.hexaware.MLP195.model.*</cove
rage-classes>
                    <coverage-check>85</coverage-check>
                </systemPropertyVariables>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.5.1</version>
            <configuration>
                <fork>true</fork>
                <executable>C:\Program Files\Java\jdk1.8.0_201\bin\javac.
exe</executable>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-checkstyle-plugin</artifactId>
            <version>2.17</version>
        </plugin>
    </plugins>
</pluginManagement>
</build>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.glassfish.jersey</groupId>

```

```

        <artifactId>jersey-bom</artifactId>
        <version>${jersey.version}</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>org.glassfish.jersey.containers</groupId>
        <artifactId>jersey-container-servlet-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jersey.media</groupId>
        <artifactId>jersey-media-json-jackson</artifactId>
    </dependency>
    <dependency>
        <groupId>org.jmockit</groupId>
        <artifactId>jmockit</artifactId>
        <version>1.33</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.jdbi</groupId>
        <artifactId>jdbi</artifactId>
        <version>2.73</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.19</version>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.16.12</version>
    </dependency>
</dependencies>
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>

```

```

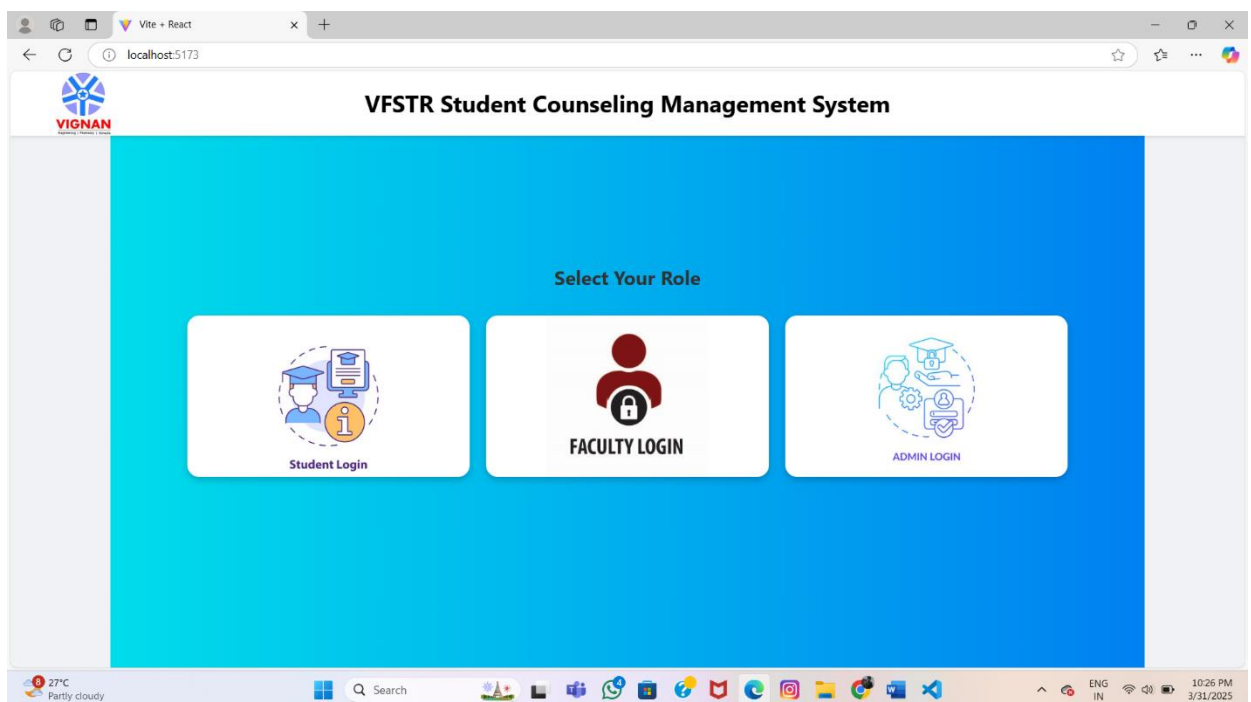
<artifactId>maven-checkstyle-plugin</artifactId>
<version>2.17</version>
<reportSets>
  <reportSet>
    <reports>
      <report>checkstyle</report>
    </reports>
  </reportSet>
</reportSets>
</plugin>
</plugins>
</reporting>
</project>

```

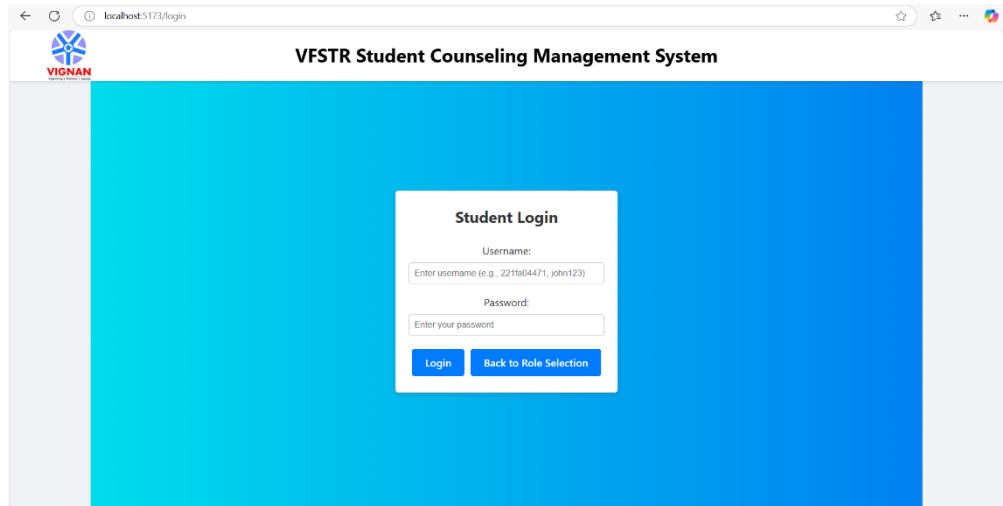
5.2 Screen Captures

5.2.1 home screen

Figure 5-1 home screen Activity



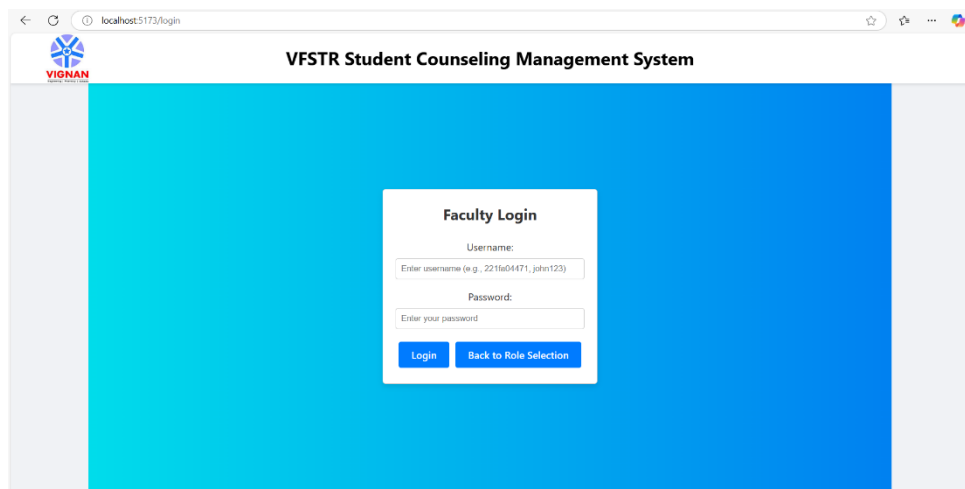
5.2.2 Student Login Screen



The screenshot shows a web browser window with the address bar displaying 'localhost:5173/login'. The page title is 'VFSTR Student Counseling Management System'. The header features the VIGNAN logo on the left. The main content area has a blue gradient background. In the center, there is a white box titled 'Student Login'. Inside this box, there are two input fields: 'Username:' with a placeholder 'Enter username (e.g., 221fa04471, john123)' and 'Password:' with a placeholder 'Enter your password'. Below the input fields are two buttons: 'Login' and 'Back to Role Selection'.

Figure 5-2 Student Login Activity

5.2.3 Counselor Login Screen



The screenshot shows a web browser window with the address bar displaying 'localhost:5173/login'. The page title is 'VFSTR Student Counseling Management System'. The header features the VIGNAN logo on the left. The main content area has a blue gradient background. In the center, there is a white box titled 'Faculty Login'. Inside this box, there are two input fields: 'Username:' with a placeholder 'Enter username (e.g., 221fa04471, john123)' and 'Password:' with a placeholder 'Enter your password'. Below the input fields are two buttons: 'Login' and 'Back to Role Selection'.

Figure 5-3 counselor Login Activity

5.2.4 Student dash board Screen

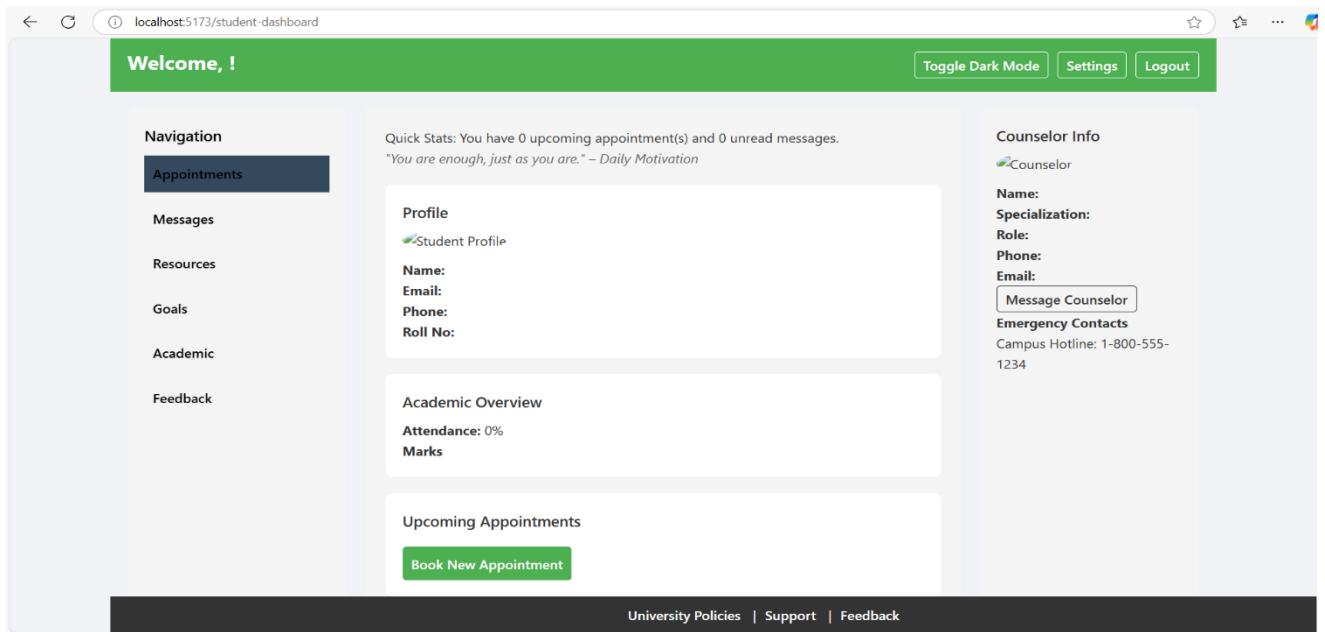


Figure 5-4 Student dash board Screen Activity

Counsellor Dash board

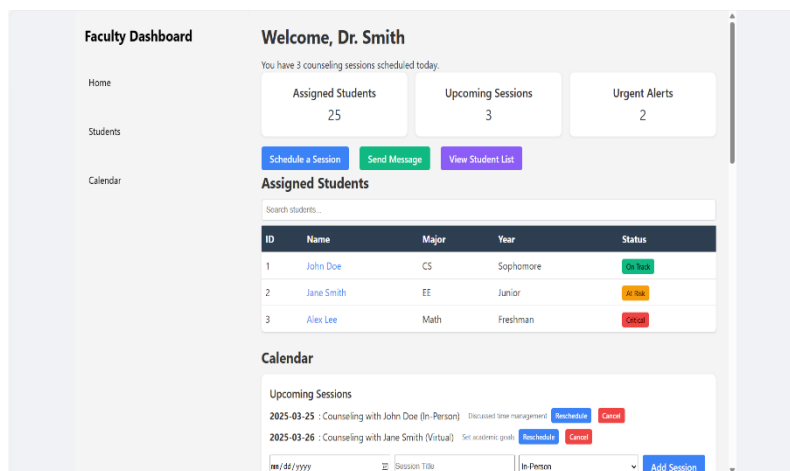


Figure 5-5 Counsellor dashboard Activity Flow

5.2.6 Admin Dashboard

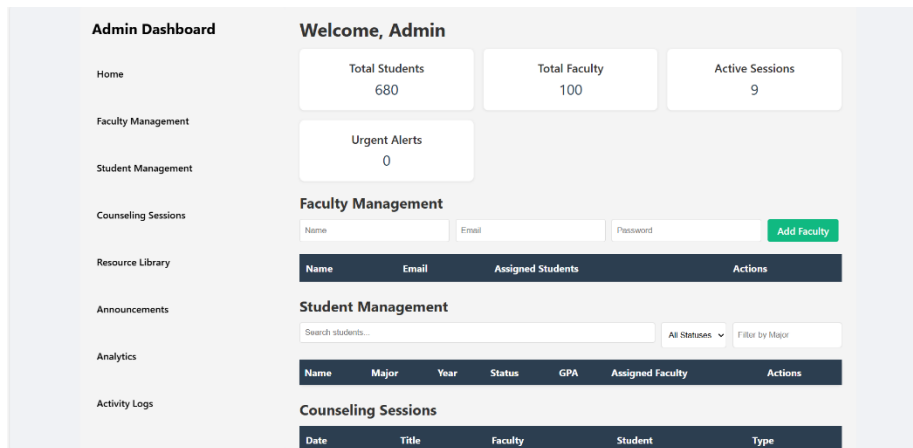


Figure 5-6 Admin DashBoard

CHAPTER - 6
TESTING

6. TESTING

6.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among these performance testing and load testing are the most important one for an android application and next sections deal with some of these types.

6.2 Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

6.3 White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

6.4 Performance Testing

Performance testing is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

6.5 Load Testing

Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

6.6 Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

CHAPTER - 7
RESULTS & CHALLENGES

7. RESULTS AND CHALLENGES

7.1 Results

- Successfully implemented a **centralized platform** for student counseling management.
- **Efficient appointment scheduling** and real-time messaging system improved communication.
- **Secure authentication** ensured role-based access control for students, counselors, and admins.
- **Automated session tracking** helped counselors monitor student progress effectively.
- Enhanced **user experience** with an intuitive interface and smooth navigation.

7.2 Challenges

- Understanding the client requirements was one of the crucial tasks of the whole project.
- Graphic User Interface (GUI) design was a difficult task as there are many types of Android devices with varying screen size and resolutions unlike iPhone.
- Implementing synchronization with server on Android was a challenging task.
- Learning different technologies and frameworks with little guidance.

CHAPTER - 8

CONCLUSIONS & FUTURE WORK

8. CONCLUSION

8.1 Conclusions

The application has been designed successfully to meet all the user requirements. I found this project to be far more difficult than I ever anticipated. Without doubt, this has been the most challenging and at the same time rewarding programming project I have undertaken since I started college.

8.2 Scope for future work

The application can further be modified in the following ways:

- **Mobile App Integration** for better accessibility.
- **Video Conferencing Support** for remote counselling.
- **Multi-Language Support** to cater to diverse users.
- **Academic Performance Integration** for holistic student guidance.

8.3 Limitations

The current application does have the **accuracy of counseling recommendations** depends on the quality of data input, which may vary. **Data security and privacy concerns** must be continuously addressed to ensure compliance with institutional and legal requirements

BIBLIOGRAPHY

[1] Oracle corporation and its affiliates “MySQL open source database”

with no author

[online] available at: <https://www.mysql.com/>

[2] Sonoo Jaiswal “An Overview on Java”

[online] available at: <https://www.javatpoint.com/>

[3] JMockit “An automated testing toolkit for java” with no author

[online] available at: <http://jmockit.github.io/tutorial/Introduction.html>

[4] M. Vaqqas, September 23, 2014 “RESTful web Services: A Tutorial”

[online] available at <http://www.drdobbs.com/web-development/tutorial>

[5] The MIT License “Angular one framework” with no author

[online] available at: <https://angularjs.org/>

[6] DaolrevoLtd. Asim “Jasmine and Karma”

[online] available at: <https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/>

[7] Software Freedom Conservancy “git local branching on the cheap” with no author

[online] available at: <https://git-scm.com/>