AN IDP PROJECT REPORT

on

**“The Summer Semester and Backlog Management**

**Web Application”**

**Submitted**

**By**

221FA04067

G. Pranav

221FA04223

K. Sirisha

221FA04111

M.V.N.L. Sowmya

221FA04503

G. Anand

***Under the guidance of***

***Sd.Shareefunnisa***

*Associate Professor*



**SCHOOL OF COMPUTING & INFORMATICS**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

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

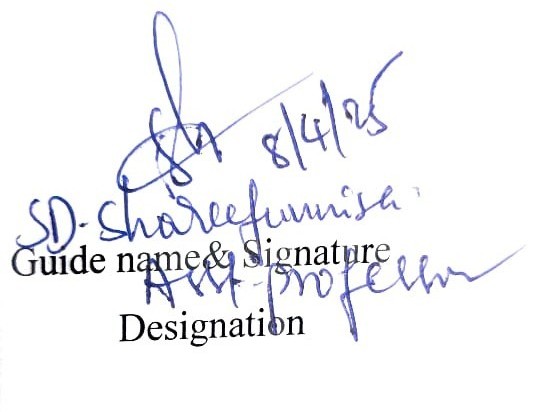
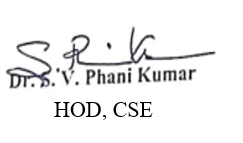
**Vadlamudi, Guntur.**

**ANDHRA PRADESH, INDIA, PIN-522213.**



**CERTIFICATE**

This is to certify that the IDP Project entitled **“The Summer Semester and Backlog Management Web Application”** that is being submitted by 221FA04067 (G.Pranav), 221FA04111(M.V.N.L.Sowmya), 221FA04223(K.Sirisha) and 221FA04503(G.Anand) for partial fulfilment of IDP Project is a bonafide work carried out under the supervision of Mrs. SD. Shareefunnisa, Assistant Professor, Department of CSE.



**DECLARATION**

We hereby declare that the IDP Project entitled “**The Summer Semester and Backlog Management Web Application”** that is being submitted by 221FA04067 (G.Pranav), 221FA04111(M.V.N.L.Sowmya), 221FA04223(K.Sirisha) and 221FA04503(G.Anand) in partial fulfilment of IDP 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 Mrs. SD. Shareefunnisa, Assistant Professor, Department of CSE.

By

**221FA04067 (G. Pranav),**

**221FA04111(M.V.N.L. Sowmya),**

**221FA04223(K. Sirisha),**

**221FA04503(G. Anand)**

**Abstract**

The increasing demand for efficient academic management has highlighted the challenges associated with manually handling summer semester and backlog courses. This project aims to develop a web-based platform that streamlines the enrollment, tracking, and completion of these courses for students, faculty, and administrators. The system provides an intuitive interface for students to enroll in summer and backlog courses, access study materials, track their progress, and communicate with faculty.

Faculty members can efficiently schedule lectures, evaluate student performance, and provide academic support, while administrators can oversee enrollments, monitor course completion rates, and generate insightful reports for institutional analysis. By automating these processes, the platform minimizes inefficiencies, reduces administrative burdens, and ensures a seamless academic progression for students**.**

**TABLE OF CONTENTS**

[LIST OF FIGURES 3](#_Toc194399561)

[ABBREIVATIONS 4](#_Toc194399562)

[1. INTRODUCTION 6](#_Toc194399563)

[1.1 Introduction 6](#_Toc194399564)

[1.2 Literature Survey 7](#_Toc194399565)

[1.2 Project Background 7](#_Toc194399566)

[1.3 Objective 7](#_Toc194399567)

[1.4 Project Description 8](#_Toc194399568)

[2. SOFTWARE REQUIREMENTS SPECIFICATION 10](#_Toc194399569)

[2.1 Requirement Analysis 10](#_Toc194399570)

[2.2 Problem Statement 10](#_Toc194399571)

[2.3 Software Requirement Specification 10](#_Toc194399572)

[2.4 Software Requirements 12](#_Toc194399573)

[2.5 Hardware Requirements 12](#_Toc194399574)

[2.6 Functional Requirements (Modules) 12](#_Toc194399575)

[2.7 Non-Functional Requirements: 13](#_Toc194399576)

[2.8 External Interface Requirements 14](#_Toc194399577)

[2.9 Feasibility study 14](#_Toc194399578)

[3. ANALYSIS & DESIGN 17](#_Toc194399579)

[3.1 Introduction 17](#_Toc194399580)

[3.2 System Overview 18](#_Toc194399581)

[3.3 System Architecture 18](#_Toc194399582)

[3.4 Data Design 19](#_Toc194399583)

[4. MODELING 23](#_Toc194399584)

[4.1 Design 23](#_Toc194399585)

[5. IMPLEMENTATION 30](#_Toc194399586)

[5.1 Sample Code 30](#_Toc194399587)

[5.2 Screen Captures 39](#_Toc194399588)

[6. TESTING 44](#_Toc194399589)

[6.1 Software Testing 44](#_Toc194399590)

[6.2 Black box Testing 44](#_Toc194399591)

[6.3 White box Testing 44](#_Toc194399592)

[6.4 Performance Testing 44](#_Toc194399593)

[6.5 Load Testing 44](#_Toc194399594)

[6.6 Manual Testing 44](#_Toc194399595)

[7. RESULTS AND CHALLENGES 46](#_Toc194399596)

[7.1 Results 46](#_Toc194399597)

[7.2 Challenges 46](#_Toc194399598)

[8. CONCLUSION 48](#_Toc194399599)

[8.1 Conclusions 48](#_Toc194399600)

[8.2 Scope for future work 48](#_Toc194399601)

**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 student module 39 |
| Figure 4-3 | Sequence Diagram for admin module 40 |
| Figure 4-4 | Activity Diagram 40 |
| Figure 4-5 | Class Diagram 41 |
| Figure 4-6 | Deployment Diagram of the System 42 |
| Figure 4-7 | ER Diagram 43 |

**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***

## INTRODUCTION

* 1. **Introduction**

The ongoing project is focused on developing a web-based platform to efficiently manage summer semester and backlog courses for students, faculty, and administrators. The main objective of this initiative is to streamline the enrollment, tracking, and completion process of summer and backlog courses, ensuring seamless academic progression for students. Managing backlog courses and summer semester enrollments manually often leads to inefficiencies, delays, and lack of proper tracking, which this system aims to resolve.

This online platform enables students to enroll in summer semester and backlog courses, access study materials, monitor progress, and communicate with faculty members. Faculty members can schedule lectures, assess student performance, and provide necessary guidance. Administrators can oversee enrollments, track course completion rates, and generate reports for institutional analysis.

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

#### students:

Students are the primary users of this system, utilizing it to enroll in courses, access academic resources, and monitor their progress. They can view detailed course structures, submit assignments, and receive feedback from faculty. Additionally, the system provides a communication interface that allows students to engage with instructors, ensuring clarity in course objectives and expectations.

#### Faculty:

Faculty members play a crucial role in managing course content, delivering lectures, and assessing student performance. They can schedule classes, upload study materials, and provide personalized guidance to students. Faculty members also have access to student records, allowing them to track individual progress and identify areas where additional support may be required.

#### 3.Admin:

Administrators oversee the entire system, ensuring smooth operation and efficient management of academic activities. Additionally, administrators generate analytical reports that help in evaluating student performance, faculty effectiveness, and overall academic progress. By leveraging data insights, administrators can make informed decisions to improve the efficiency and effectiveness of summer semester and backlog course management.

## Literature Survey

Conducting a literature survey is a crucial step in the development process. Before building the system, it is essential to evaluate factors like feasibility, cost, and implementation challenges. Selecting the appropriate technology stack is critical for the efficiency and scalability of the platform. Developers also require external support, which can be obtained from experienced programmers, reference materials, and online resources.

* Since the project follows the "**Agile Development Model**," relevant Agile methodologies have been studied and applied.
* Additional training on "**Version Control Systems**" has been undertaken to

ensure smooth collaboration and code management.

## Project Background

The current backlog and summer semester management system is inefficient

due to manual handling, leading to missed deadlines, confusion, and administrative

burdens. Implementing an automated, cloud-based solution will streamline course

registration, facilitate better communication between students and faculty, and ensure

that academic progress is tracked effectively. This transformation will significantly

improve the efficiency and reputation of academic management.

## Objective

The primary objective of this project is to automate all activities related to backlog and summer semester management, from student enrollment to academic tracking. The system should maintain comprehensive records of enrolled students, faculty schedules, course materials, and assessment outcomes. It should also provide an intuitive interface for generating reports on student performance and course completion. Additionally, the system should offer inquiry functionalities for students and faculty to track course progress, view schedules, and access feedback efficiently.

## Project Description

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

#### Student

* Enroll in backlog and summer semester courses.
* Access study materials, assignments, and course schedules.
* Track academic progress and receive structured feedback.
* Communicate with faculty regarding academic concerns.
* View and manage backlog course history.
  + 1. **Faculty**
  + Schedule and manage backlog and summer semester courses.
  + Upload study materials and conduct assessments.
  + Provide academic support and personalized feedback.
  + Track student performance and engagement.
  + Generate reports on student progression and course effectiveness.
    1. **Admin**
* Manage student enrollments and faculty assignments.
* Oversee course scheduling and monitor academic activities.
* Generate analytics and performance reports.
* Ensure the smooth functioning of the system.

***CHAPTER - 2***

***SOFTWARE REQUIREMENT SPECIFICATION***

# 2. SOFTWARE REQUIREMENTS SPECIFICATION

# 2.1 RequiremenNt Analysis

For the purpose of easy access and portability we propose to develop a web-based system with windows or ubuntu as a platform because they are widely used operating systems. As a part of this system, we are developing web-based software that allows data to 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 a diagrammatical view of the system what we are going to develop.

## Problem Statement

The problem statement focuses on managing and tracking students' Summer

semester registrations and backlog courses effectively. The challenge lies in efficiently

storing student records, tracking their academic progress, and ensuring seamless synchr-

onization of data between students, faculty, and administrators. The proposed system

will use a MySQL database for data storage and synchronization with the MySQL server.

* 1. **Functional Requirements**

## Software Requirement Specification

The project is developed using Java Programming Language with Eclipse.

We utilize Azure Extension, which includes various custom tools to facilitate the

deployment of web applications on the Azure cloud platform. On the server side,

Apache Tomcat Server is used. The backend database is managed using Net-Beans,

Java jdbc, servelets and maven, while the frontend is developed using Angular.

#### Purpose

The purpose of this document is to present a detailed description of **“Summer Semester and Backlog 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.

#### Scope of the project

The Summer Semester and Backlog Management System is designed to

streamline academic management for students, faculty, and administrators by

providing an efficient platform for tracking backlog courses, registering for summer

semesters, and monitoring academic progress.

#### Student:

The system allows students to register for summer semester courses, view

backlog course details, access academic records, and receive notifications about

registration deadlines. The platform ensures secure access to student records and

personalized academic planning.

#### Faculty:

Faculty members can manage student enrollments, update course records,

monitor student progress and provide feedback. The system enables seamless

communication between students and faculty regarding coursework, deadlines, and

academic assistance.

**3.Admin:**

Administrators can oversee student enrollments, generate reports on student

performance, and track the number of backlog courses managed within the system.

#### 2.3.4 Technologies Used

###### 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.

* ANGULAR

**Angular** is a TypeScript-based open-source web application framework and also Angular is a development platform that aims to make web development feel effortless, focused on developer productivity, speed and testability. Applications built with Angular can be deployed to mobile devices and desktops as websites and native applications.

* Java JDBC

**Java Database Connectivity** (JDBC) is an API that allows Java applications to interact with relational databases. JDBC provides a standard interface for accessing and manipulating database data using SQL queries. It is part of the Java Standard Edition (Java SE) and is widely used for database connectivity in Java applications.

* Servlets

**Java Servlets** are server-side Java programs that handle client requests and generate dynamic web content. They run on a web server or application server and extend the capabilities of web applications by interacting with client requests, processing data, and generating dynamic responses, typically in the form of HTML or JSON.

* MySQL

**MySQL** is one of the most popular relational database management systems on the web. MySQL is used for the internet applications as it provides good speed and is very secure. MySQL was developed to manage large volumes of data at very high speed to overcome the problems of existing solutions.

#### Overview

The application is designed for Windows and Ubuntu platforms and supports

offline data collection, allowing students to update their records without an active

internet connection. Once online, the system syncs the data with the server, ensuring

updated academic records.

## Software Requirements

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

* Frontend Framework - HTML, JavaScript
* Backend Framework - NetBeans
* Database - MySQL
* Authentication - JWT-based authentication
* State Management - Redux or Context API
* Styling - Tailwind CSS / Material UI
* Hosting - Vercel (Frontend), Render/Heroku (Backend)

## 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**

## Functional Requirements (Modules)

* **User Authentication & Role-Based Access:** Secure login for students, faculty, and administrators.
* **Student Registration:** Enable students to register for summer semester and backlog courses.
* **Course Management:** Faculty can update course details and academic schedules.
* **Academic Progress Tracking:** Students and faculty can track performance and progress.
* **Notification System:** Alerts for registration deadlines and academic events.
* **Admin Dashboard:** Manage student and faculty records, generate reports.
* **Data Security:** Ensure secure access and role-based permissions.

## Non-Functional Requirements:

#### 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.

#### Design Constraints:

The summer semester and backlog management shall be a stand-alone system running in a Windows environment. The system shall be developed using ANGULAR and connected to MySQL database.

#### Standards Compliance:

There shall be consistency in variable names within the system. The graphical user interface shall maintain a uniform design.

#### Availability:

The system shall be accessible 24/7 to allow students to register at their

convenience.

#### Portability:

The system shall be platform-independent and accessible via any web browser and be able to available to all users all the time while they have the internet.

#### Reliability:

The system shall ensure accurate record-keeping and backup mechanisms

to prevent data loss.

## External Interface Requirements

* + 1. **User Interface**

A critical aspect of this project was examining how the app would look and its

usability. The system features an intuitive and user-friendly interface designed using

HTML and JavaScript. The layout ensures easy navigation for students, faculty, and

administrators.

## Feasibility study

A key part of the preliminary investigation that reviews, 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.

#### Organisational Feasibility

The application aligns with institutional goals by improving the efficiency of academic management. It offers a scalable and flexible solution that can be upgraded to accommodate future requirements. However, it is up to the organization to upgrade or extend it.

#### 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.

#### 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 a cloud platform.

#### Behavioural Feasibility

The application is user-friendly, requiring no technical expertise. All modules are designed for ease of use, ensuring smooth execution of functionalities.

***CHAPTER - 3***

***ANALYSIS & DESIGN***

**3.** **ANALYSIS & DESIGN**

* 1. **Introduction**
     1. **Purpose**

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

* + - 1. **Document Purpose**

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

* + - 1. **Project Purpose**

The Summer Semester and Backlog Management Web Application is designed to facilitate the management of summer semester courses and backlog subjects for students in educational institutions. The system provides students with an efficient platform to register for summer courses, track backlog subjects, and access academic progress. Faculty members can manage course enrollments, approve student registrations, and monitor academic performance. Administrators oversee system operations, ensuring data security and streamlined management of summer semester and backlog records. The system aims to improve academic planning, ensure timely course completion, and enhance communication between students and faculty.

* + 1. **Scope**

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

* + - 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. The document outlines the system architecture, user interface design, and high-level module interactions. 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.
      1. **Project Scope**

The **Summer Semester and Backlog Management System** is designed to assist students in managing their backlog courses and summer semester enrollments efficiently. This system ensures a streamlined process for students who need to retake subjects or accelerate their academic progress through additional coursework. It consists of two main modules: **ADMIN** and **STUDENT**, each playing a crucial role in the overall functionality of the system.

1. **ADMIN:**

The **ADMIN** module manages summer semester courses and backlog records.

It allows the admin to add, update, or remove courses, approve backlog applications,

track student progress, and send notifications about deadlines, schedules, and exams.

1. **STUDENT:**

The **STUDENT** module enables students to view and register for summer semester

courses or backlog exams. Students can track application status, receive updates on

schedules and fees and sync their enrollment data with the university database when

online.

* 1. **System Overview**

Cloud deployment framework utilizes several development tools, including:

* + - 1. **Eclipse IDE**

Eclipse is a widely used Integrated Development Environment (IDE) that supports Java, JavaScript, and other programming languages. It is an open-source platform that provides an extensive set of tools for software development, including debugging, version control integration, and automated build tools. Eclipse is chosen for this project due to its robust support for enterprise-level application development and seamless integration with web frameworks and backend services.

* + - 1. **Java JDBC**

Java Database Connectivity (JDBC) is an API that allows Java applications to interact with relational databases. JDBC provides a standard interface for accessing and manipulating database data using SQL queries. It is part of the Java Standard Edition (Java SE) and is widely used for database connectivity in Java applications.

JDBC supports multiple databases, including MySQL, PostgreSQL, Oracle, SQL Server, and others, by using database-specific JDBC drivers. These drivers enable Java applications to communicate with databases seamlessly.

JDBC requires database-specific drivers, which are typically included as .jar files and managed using build tools like Maven. Dependencies for JDBC drivers are added in the pom.xml file when using Maven for project management.

* + - 1. **Maven**

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation.

Maven addresses two aspects of building software: how software is built, and its dependencies. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. Maven dynamically downloads Java libraries and Maven plug- ins from one or more repositories such as the Maven 2 Central Repository and stores them in a local cache. Maven projects are configured using a Project Object Model, which is stored in a pom.xml-file.

A screenshot of a cellphone

AI-generated content may be incorrect.



* 1. **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. The web application ensures seamless integration between these components, enabling students, faculty, and administrators to efficiently manage summer semester and backlog course enrollments while maintaining academic records securely.

**Attributes of a well-built web application architecture:**

* Solves business problems
* Supports visual aesthetic
* Enables A/B testing and analytics
* Ensures fast user experience
* Provides security
* Sustainable and self-regulating
* Scales out and logs errors in an easy way
* Guarantees a high level of automation

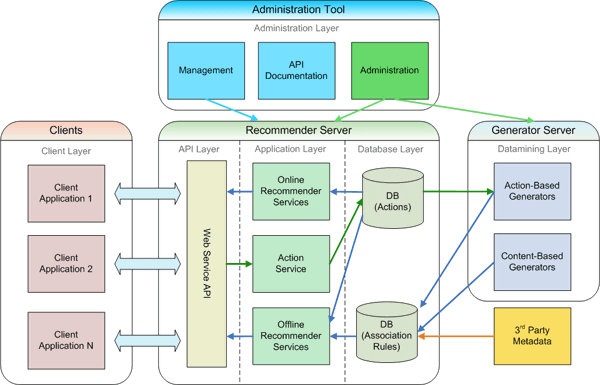
**Components of Web Application Architecture**

Web application architecture consists of application components, middleware systems, and databases. They can be divided into two groups:

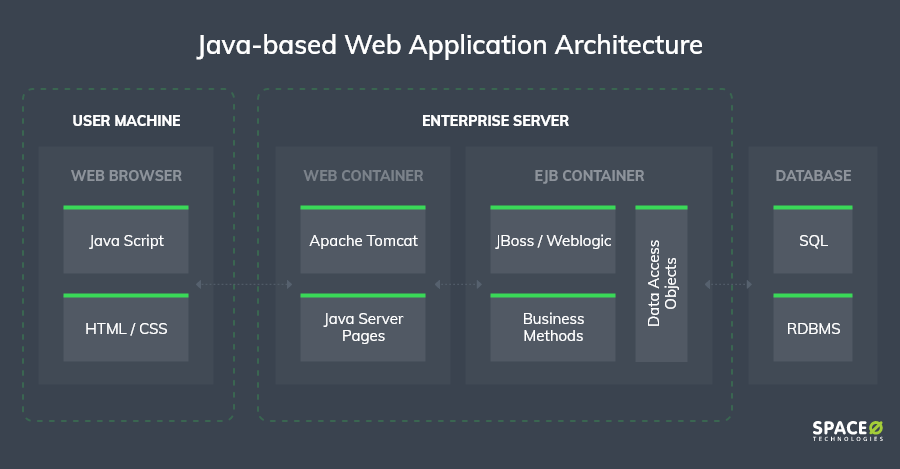
* UI/UX components
* Structural components

**UI/UX components** include dashboards, statistical data, notification elements, layouts, activity tracking, and other elements. These components create the visuals of a web page and lay the foundation for user experience.

Meanwhile, **structural components** include the web application server and the database server. knowledge of HTML, JavaScript, and CSS, as well as Python, PHP, Java, Ruby, .NET and Node.js are required to create them.



**Figure 3-2 Web Application Architecture Diagram**

**3.3.2 Overall Software Architecture**

**Figure 3-3 Java-based web application architecture**

The architecture shown in above figure is used in the sync operation where the data from web pages goes to web server (Apache Tomcat) to database server (MySQL). JSP is used here because of the interaction it can offer with the databases and it is easy to deploy on the XAMPP web server and here it sits in middle as shown in figure. SQL insert query is written in JSP script to insert this data into MySQL database server.

* 1. **Data Design**
     1. **Databases**

**MySQL RDBMS**

|  |
| --- |
| **Name** |
| Admin |

**Table 3.1 MySQL Database**

* + 1. **Tables**

|  |  |
| --- | --- |
| **Name** | **Description** |
| User | To store the user details |
| Student | To store student details |
| Admin | To store admin details |
| Course | To store course details |
| Enrollment | To enroll the course |

**Table 3.2 List of Database Tables**

**3.4.2.1 Database:** Table**:** user

**Description:** To store the user details

**Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Null** | **Default** | **Index** |
| user-id | Int | No | NULL | Primary Key Auto Increment |
| Name | Varchar (100) | No | NULL |  |
| Email | Varchar (100) | No | NULL | Unique Key |
| Password hash | Varchar (255) | No | NULL |  |
| Role | ENUM ('admin', 'student') | No | NULL |  |
| Created at | TIMESTAMP | Yes | Current Timestamp |  |

**Table 3.3 Structure of user table of MySQL Database**

**Definition**

CREATE TABLE IF NOT EXISTS users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

role ENUM('admin', 'student') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

* + - 1. **Database: Table:** student

**Description:** To store student details

**Columns:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Null** | **Default** | **Index** |
| student\_id | INT | NO | NULL | AUTO\_INCREMENT |
| user\_id | INT | NO | NULL |  |
| enrollment\_number | VARCHAR(50) | NO | NULL |  |
| course | VARCHAR(100) | YES | NULL |  |
| department | VARCHAR(100) | YES | NULL |  |
| semester | INT | YES | NULL |  |
| date\_of\_birth | DATE | YES | NULL | ON DELETE CASCADE |

Table 3.4 Structure of student table MYSQL Database

**Definition:**

CREATE TABLE IF NOT EXISTS students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL UNIQUE,

enrollment\_number VARCHAR(50) NOT NULL UNIQUE,

course VARCHAR(100),

department VARCHAR(100),

semester INT,

date\_of\_birth DATE,

contact VARCHAR(15),

address TEXT,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE

);

* + - 1. **Database: Table:** admin

**Description:** To store details of admin

**Columns:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Null** | **Default** | **Index** |
| admin\_id | INT | NO | NULL | AUTO\_INCREMENT |
| user\_id | INT | NO | NULL |  |
| designation | VARCHAR(100) | YES | NULL |  |
| contact | VARCHAR(15) | YES | NULL |  |

**Table 3.5 Structure of admin table of MySQL Database**

**Definition:**

CREATE TABLE IF NOT EXISTS admins (

admin\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL UNIQUE,

designation VARCHAR(100),

contact VARCHAR(15),

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE

);

**Database:** Table: courses

**Description:** To store course details

**Columns:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Null** | **Default** | **Index** |
| course\_id | INT | NO | NULL | AUTO\_INCREMENT |
| course\_name | VARCHAR(100) | NO | NULL |  |
| department | VARCHAR(100) | YES | NULL |  |
| semester | INT | NO | NULL |  |
| credits | INT | NO | NULL |  |

**Table 3.6 Structure of course table MYSQL Database**

**Definition**

CREATE TABLE IF NOT EXISTS courses (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL UNIQUE,

department VARCHAR(100),

semester INT NOT NULL,

credits INT NOT NULL

);

* + - 1. **Database: Table:** enrollment

**Description:** To enroll course

**Columns:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Null** | **Default** | **Index** |
| enrollment\_id | INT | NO | NULL | AUTO\_INCREMENT |
| student\_id | INT | NO | NULL |  |
| course\_id | INT | NO | NULL |  |
| enrollment\_date | TIMESTAMP | YES | CURRENT\_TIMESTAMP |  |

**Table 3.7 Structure of enrollment table MYSQL Database**

**Definition**

CREATE TABLE IF NOT EXISTS enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT NOT NULL,

course\_id INT NOT NULL,

enrollment\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (student\_id) REFERENCES students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (course\_id) REFERENCES courses(course\_id) ON DELETE CASCADE

);

***CHAPTER - 4***

***MODELING***

## 4. Modeling

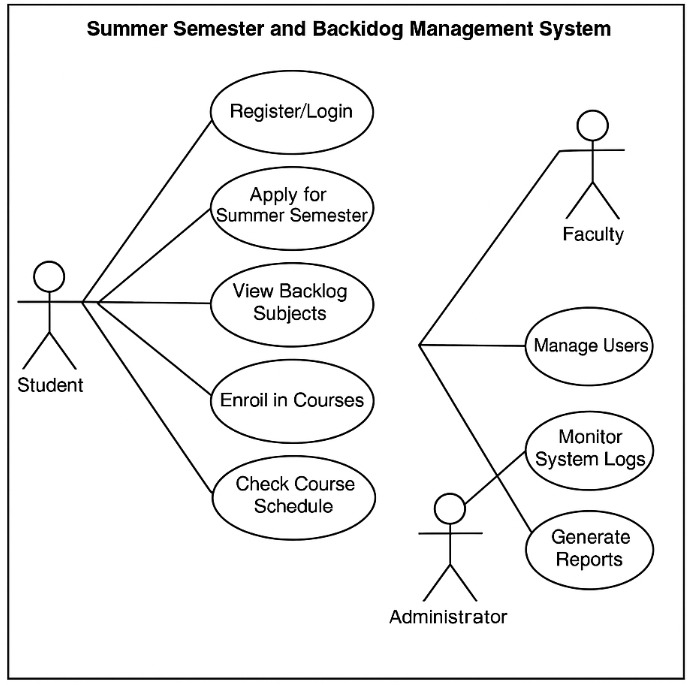
## Design

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 overview 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 number 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.

A diagram of a course

AI-generated content may be incorrect.

#### Figure 4-2 Sequence Diagram for student

#### 

#### 

#### 

#### Figure 4-3 Sequence Diagram for admin

#### 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 form 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.

A flowchart of a student

AI-generated content may be incorrect.

**Figure 4-4 Activity Diagram**

#### 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.

A diagram of a program

AI-generated content may be incorrect.

#### Figure 4-5 Class Diagram

#### 4.1.5 Deployment Diagram

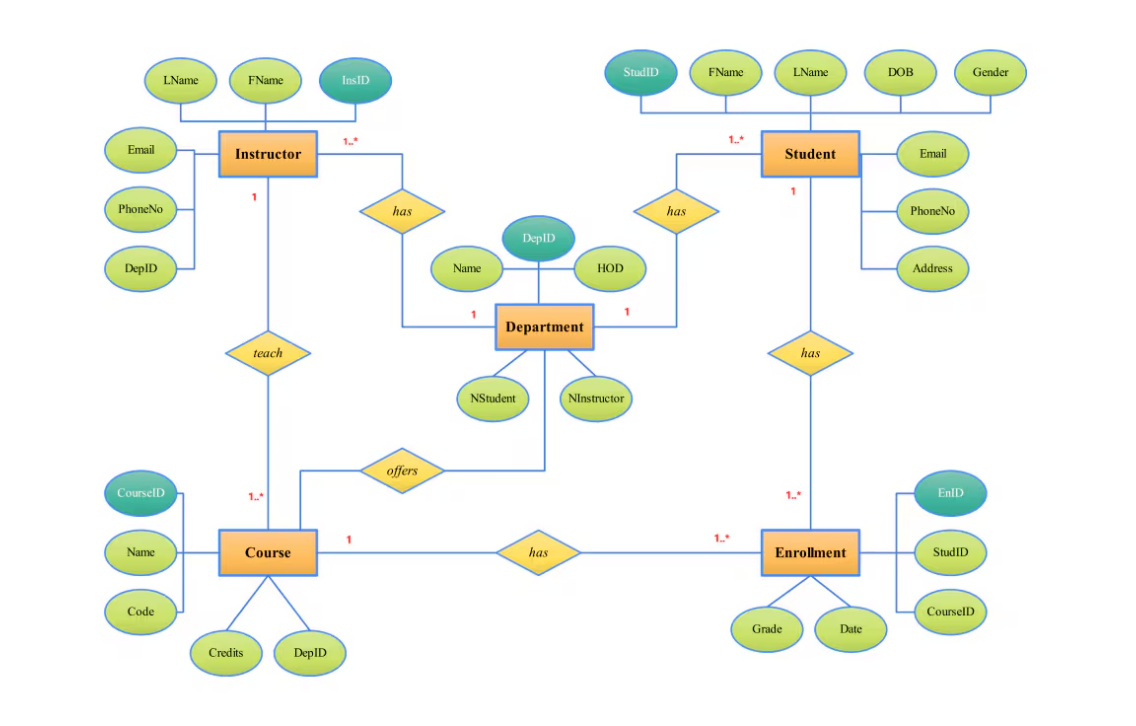
Deployment diagram shows execution architecture of systems that represent the assignment (deployment) of software artifacts to deployment targets (usually nodes). Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create 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.

**A diagram of a computer

AI-generated content may be incorrect.**

**Figure 4-6 Deployment Diagram**

#### 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.

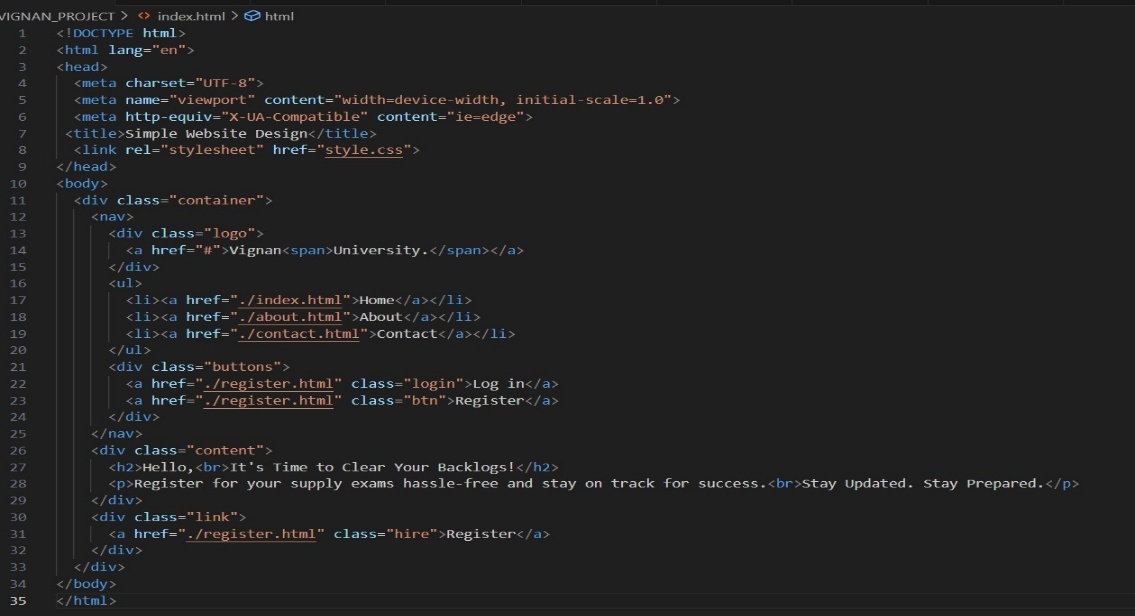
**Figure 4-7 ER Diagram**

***CHAPTER - 5***

***IMPLEMENTATION***

**5*.* IMPLEMENTATION**

* 1. **Sample Code**

******

**Index.html**

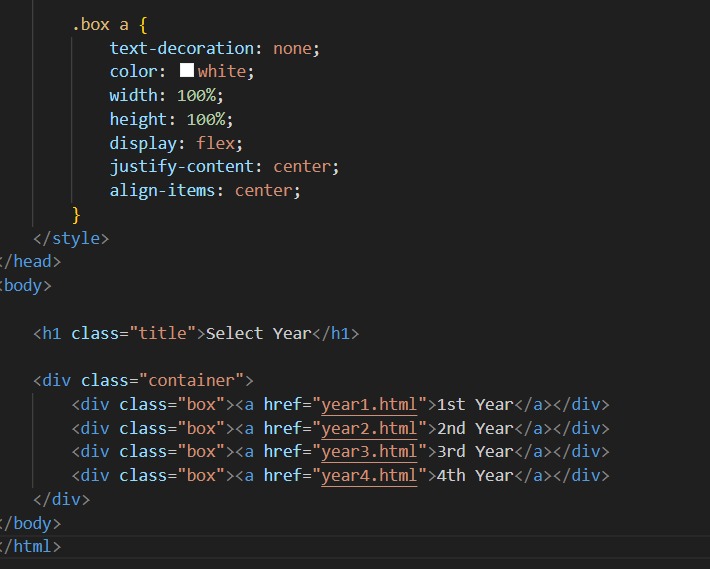
******

**Contact.html**

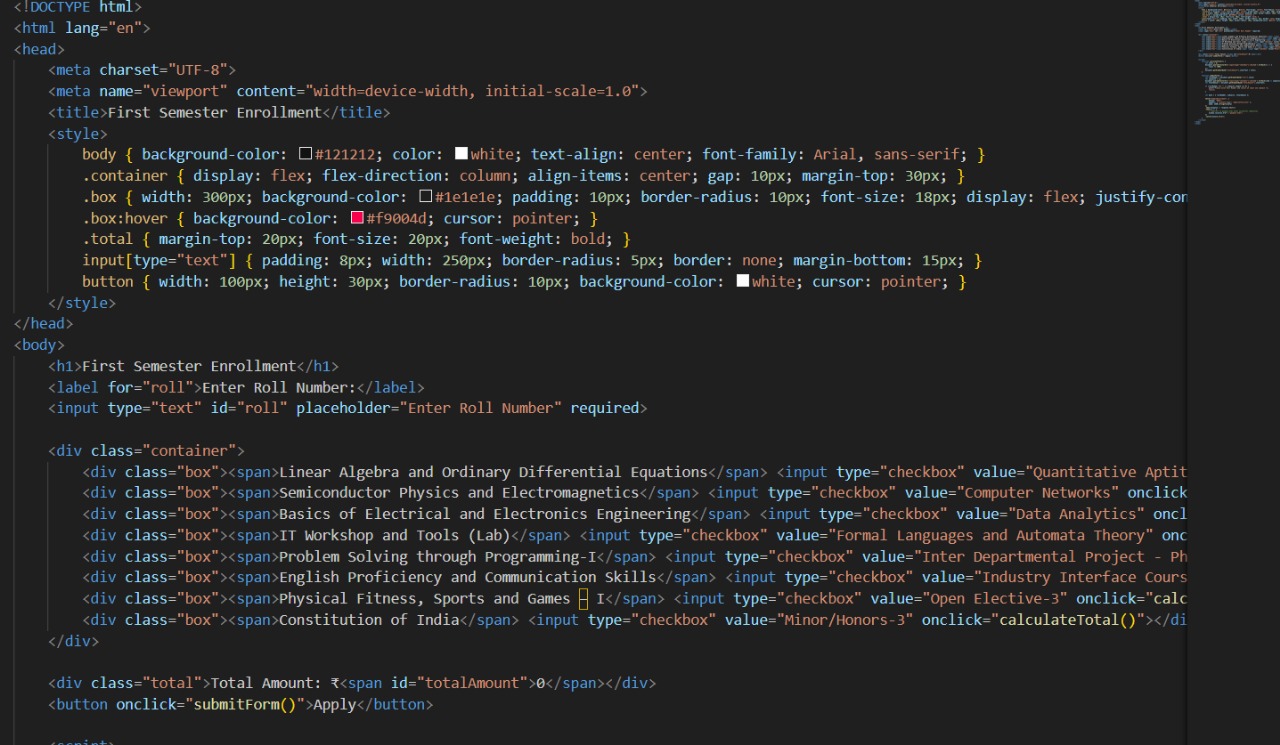
******

**Signup/login.html**

******

******

**Year selection.html**

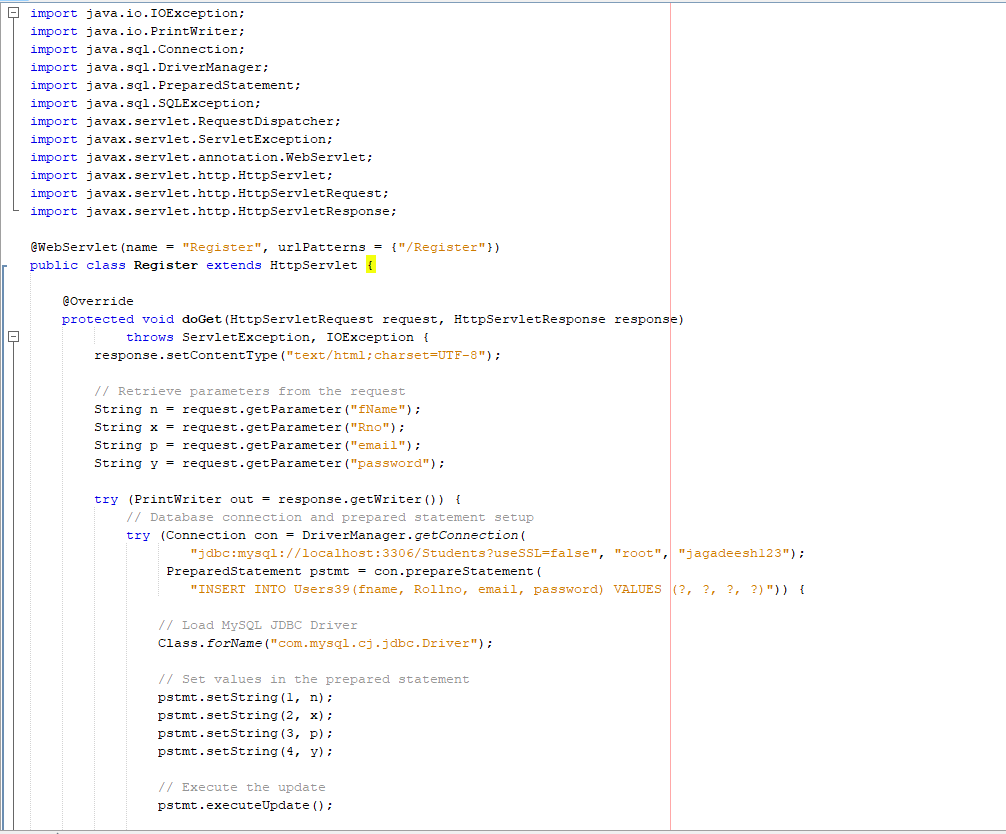
******

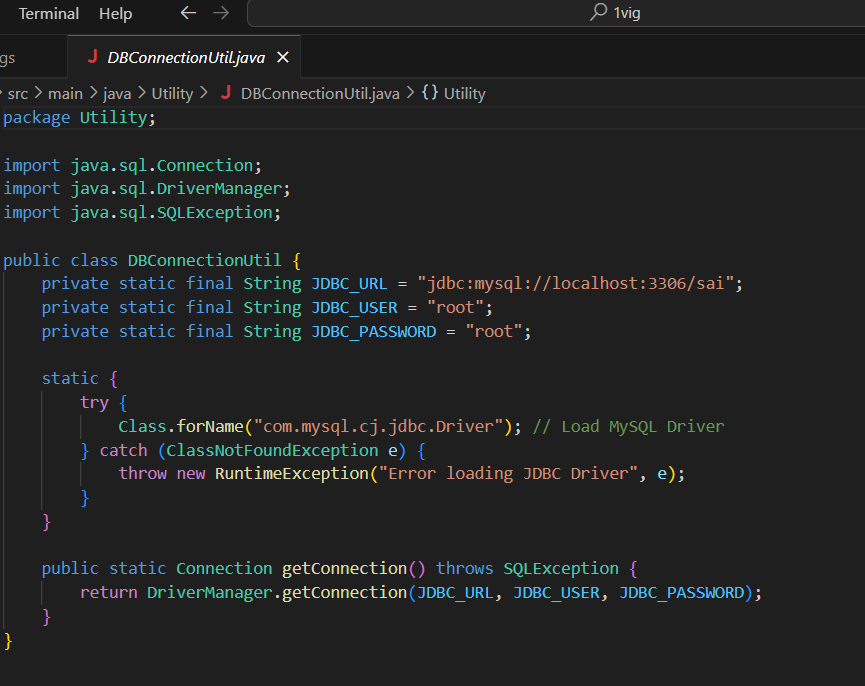
**Semester.html**

******

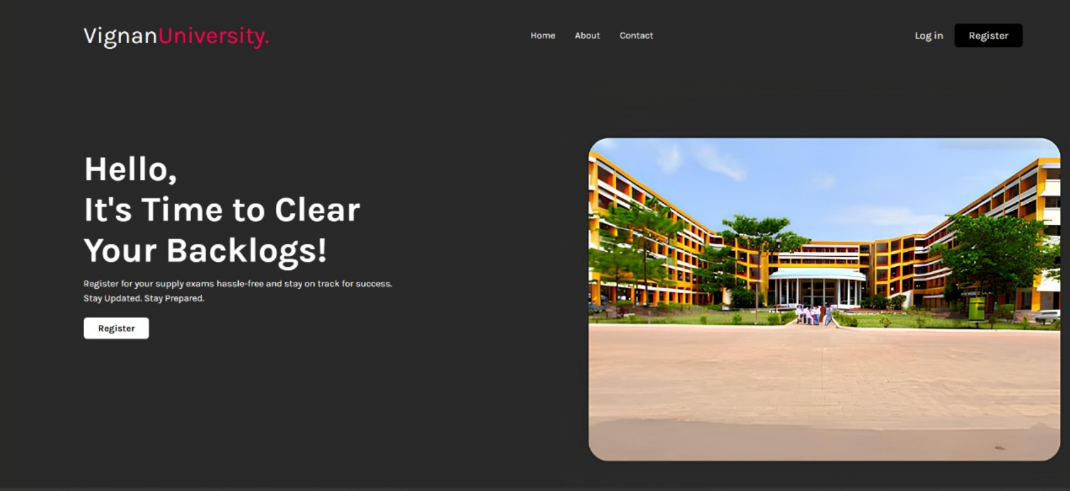
**Payment.html**

**Connection**

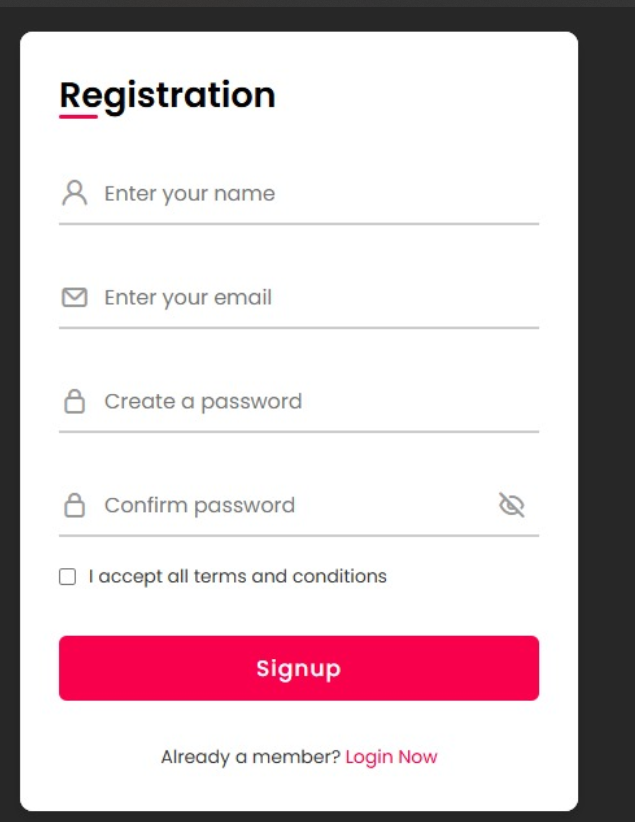
******

******

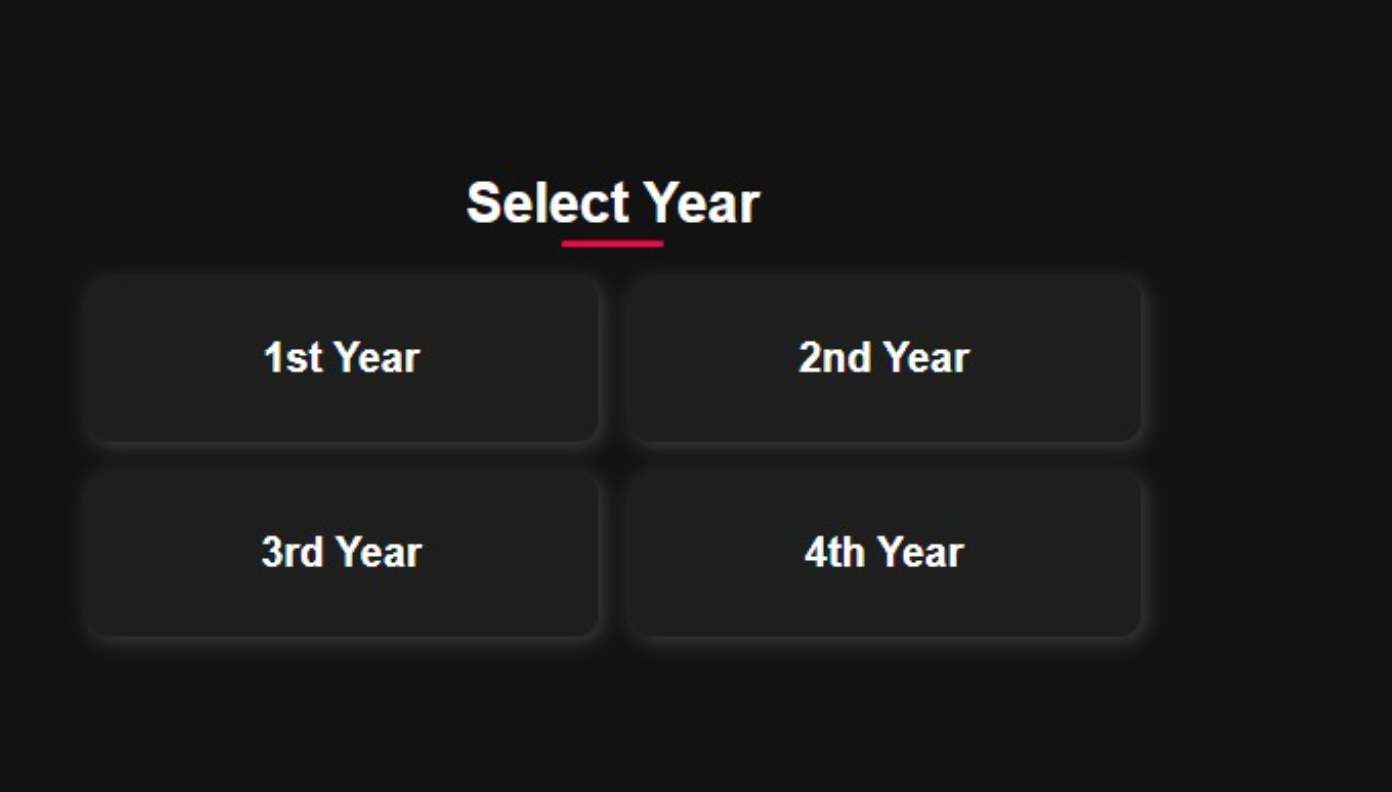
**Outputs**

******

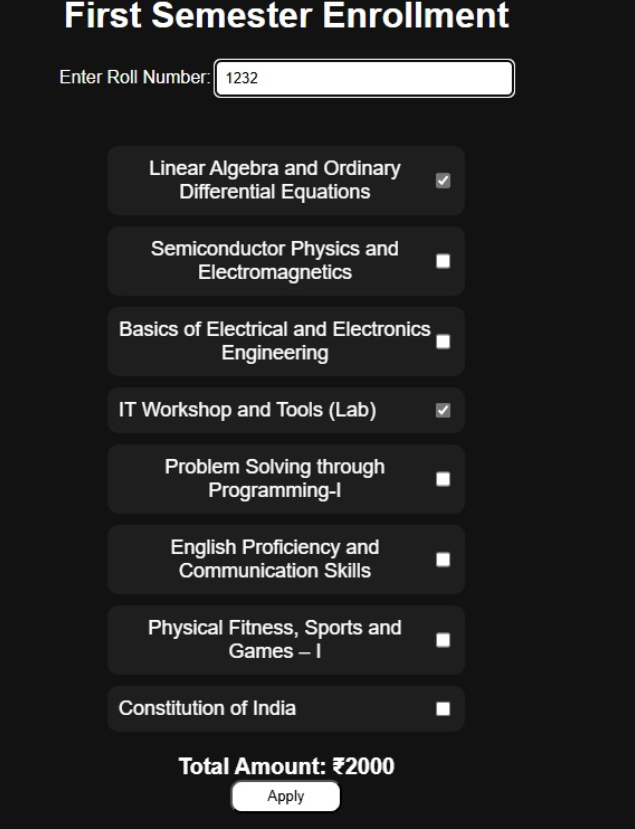
**Home page**

******

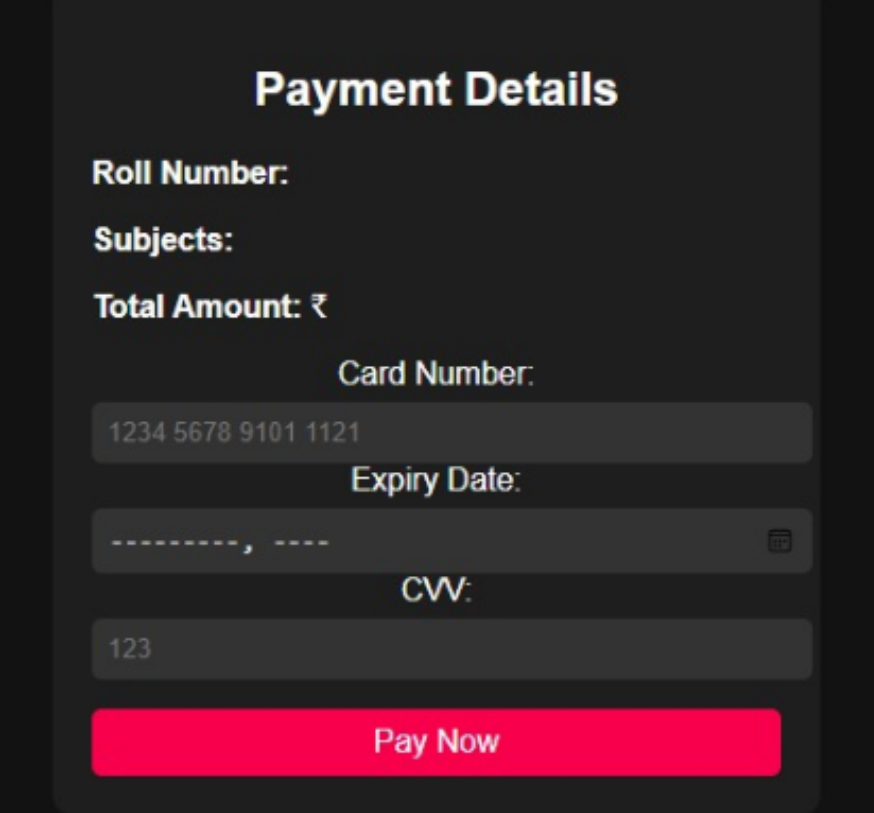
**Login/signup**

******

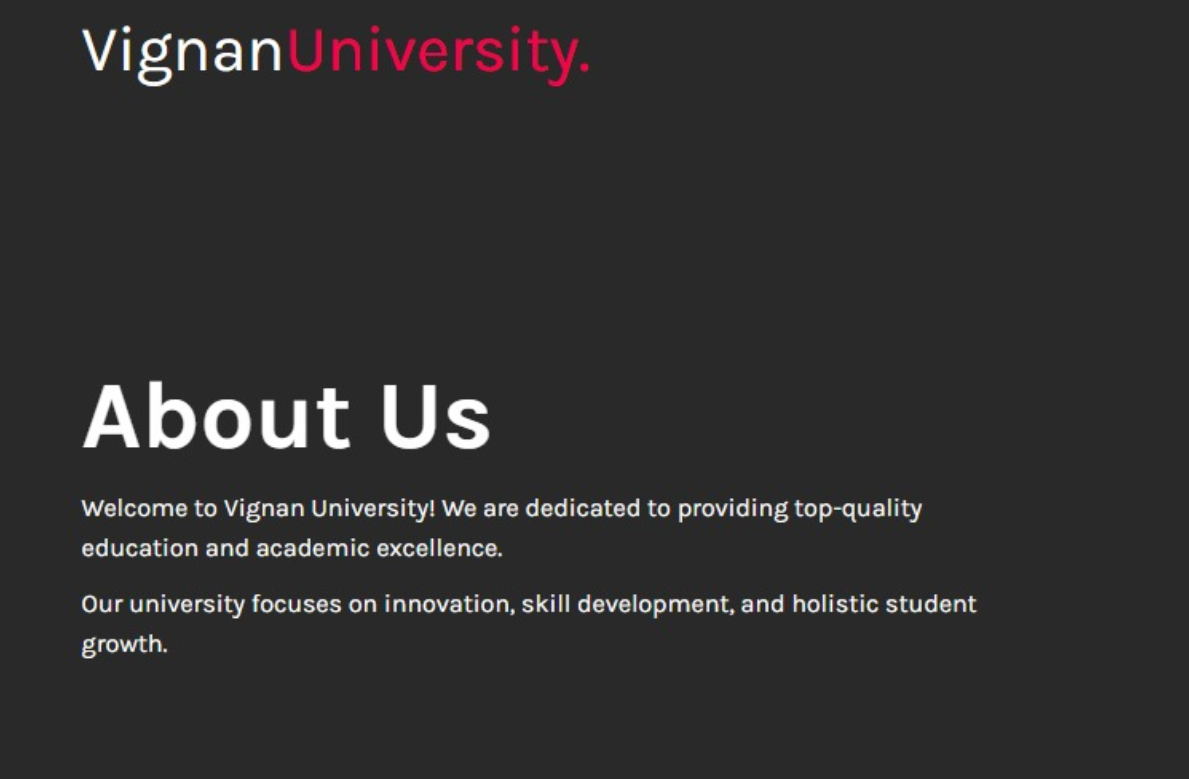
**Year selection**

******

**Year Enrollment**

******

**Payment details**

******

**About**

***CHAPTER - 6***

***TESTING***

*.*

* 1. **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.

## 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.

## 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.

## 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.

## 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.

## 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***

## RESULTS AND CHALLENGES

## 7.1 Results

* Successfully implemented a **centralized platform** for managing summer semester and backlog courses
* **Efficient scheduling system** allowed students to enroll in backlog courses seamlessly.
* **Role-based** **authentication** ensured secure access for students, faculty, and administrators.
* **Automated progress tracking** enabled faculty to monitor student performance effectively.
* Enhanced **user experience** with an intuitive interface and structured navigation.

## 7.2 Challenges

* Understanding the client requirements was one of the crucial tasks of the whole project.
* Designing an intuitive **User Interface (UI)** that caters to diverse user roles, including students, faculty, and administrators.
* Implementing real-time course updates and synchronization with the university's

database.

* Managing course conflicts and ensuring a smooth enrollment experience.
* Learning and integrating multiple technologies for efficient backend and frontend development.

***CHAPTER - 8***

***CONCLUSIONS & FUTURE WORK***

# CONCLUSION

## Conclusions

The application has been successfully designed to streamline the management of summer semester and backlog courses. The project was more complex than initially anticipated, requiring a deep understanding of academic policies, scheduling conflicts, and secure user access. However, overcoming these challenges made this project one of the most rewarding and insightful experiences in software development.

## Scope for future work

The application can further be modified in the following ways:

* **Mobile App Integration** for better accessibility.
* **Automated Notification System** for course registration deadlines and exam

schedules.

* **AI-Driven Course Recommendations** to assist students in selecting the right

Courses.

* **Multi-Language Support** to accommodate diverse student backgrounds.
* **Integration with University ERP Systems** for seamless academic record

management.

## Limitations

* The accuracy of course recommendations depend on the quality and completeness

of academic records.

* Managing real-time enrollment updates during peak registration periods may require

additional optimization.

* Ensuring data privacy and security remains a continuous priority to comply with

academic regulations and institutional policies.

# BIBLIOGRAPHY

1. Oracle corporation and its affiliates “MySQL open source database” with no author

[online] available at: https:/[/www.mysql.com/](http://www.mysql.com/)

1. Sonoo Jaiswal “An Overview on Java” [online] available at: https:/[/www.javatpoint.com/](http://www.javatpoint.com/)
2. JMockit “An automated testing toolkit for java” with no author [online] available at: <http://jmockit.github.io/tutorial/Introduction.html>
3. M. Vaqqas, September 23, 2014 “RESTful web Services: A Tutorial” [online] available at <http://www.drdobbs.com/web-development/tutorial>
4. The MIT License “Angular one framework” with no author [online] available at: https://angularjs.org/
5. DaolrevoLtd. Asim “Jasmine and Karma”

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

1. Software Freedom Conservancy “git local branching on the cheap” with no author [online] available at: https://git-scm.com/