DON BOSCO COLLEGE OF ENGINEERING

FATORDA, MARGAO, GOA – 403 602.

DEPARTMENT OF COMPUTER ENGINEERING

2025 – 2026

CAMPUS CONNECT

By

Mr. Swapnil Naik

Mr. Pranav Gauns Dessai

Mr. Shivasharanappa Biradar

Under the Guidance of

Prof. Amey Tilve

Assistant Professor

BACHELOR OF ENGINEERING: GOA UNIVERSITY

DON BOSCO COLLEGE OF ENGINEERING

FATORDA, MARGAO, GOA- 403 602.

2024 – 2025

CERTIFICATE

The dissertation entitled

CAMPUS CONNECT

Submitted by

Mr. Swapnil Naik
Mr. Pranav Gauns Dessai
Mr. Shivasharanappa Biradar

In partial fulfillment of the requirements of the Bachelor's Degree in Computer Engineering of Goa University is evaluated and found satisfactory.

DATE: _____ **EXAMINER 1:**_____

PLACE: _____ **EXAMINER 2:** _____

ABSTRACT

The Campus Connect system is a web-based forum application designed to foster collaboration, discussion, and engagement within a college community. This project provides students and faculty with a secure and user-friendly platform to share ideas, post updates, ask questions, and participate in topic-based discussions—similar to modern social platforms like Reddit, but tailored to the academic environment.

Users can register and authenticate through a secure login process powered by JWT-based authentication, ensuring data privacy and controlled access. The platform allows users to create posts, comment on discussions, and upvote relevant content, promoting a dynamic and interactive campus ecosystem.

Developed using React for the frontend, Node.js and Express for the backend, and Supabase for data storage with Drizzle ORM for structured database interactions, Campus Connect emphasizes both performance and scalability. The responsive web interface ensures accessibility across devices, enabling seamless participation from anywhere.

.

ACKNOWLEDGEMENT

We would like to take this opportunity to express our sincere gratitude to all the stakeholders who have helped us directly or indirectly by providing us with the opportunity, the necessary expertise, and guidance, and motivating us constantly. Without their help, this project would have remained a dream.

We would like to thank our Director Rev. Fr. Wilfred Fernandes and our principal Dr. Neena S.P. Panandikar for providing us with the necessary facilities and the much-needed motivation throughout the project.

We would like to express our gratitude to our Head of the Department, Dr. Gaurang Patkar for showing confidence in us and for giving us timely suggestions and advice with his rich experience and expertise.

Guides are like beacons of light showing us the right path in the phase of uncertainty and confusion during the project. When things seem to be drifting away from the goal, they put us back on track with their constant monitoring and guidance, and expertise. We are indebted to our Internal Guide **Prof Amey Tilve** Assistant Professor, Department of Computer Engineering

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO THE PROJECT

The **Campus Connect** system is a web-based forum platform designed to enhance communication, collaboration, and knowledge sharing within a college environment. With the rise of online communities and digital learning ecosystems, there is a growing need for an academic social platform that connects students and faculty beyond the classroom.

This project aims to build a dedicated discussion platform similar to Reddit, where users can create posts, share insights, ask questions, and participate in discussions relevant to their academic departments or interests. The application integrates features such as user authentication, posting, commenting, and upvoting to encourage constructive engagement and interaction within the campus community.

1.2 PURPOSE OF THE PROJECT

The primary purpose of **Campus Connect** is to create a secure, interactive, and user-friendly digital space for students and faculty to exchange ideas and resources. Traditional notice boards, informal chat groups, or departmental emails often lead to fragmented communication. Campus Connect addresses this issue by offering a centralized and modern platform that supports real-time discussions and content sharing.

1.3 PROBLEM DEFINITION

In most colleges, communication among students, faculty, and departments is still handled through manual processes, scattered group chats, or static web portals. This leads to missed announcements, limited collaboration, and lack of centralized discussions.

The absence of an integrated academic forum also restricts students from sharing ideas across departments and limits informal peer learning. With the growing dependence on digital tools in education, there is a clear need for a unified, secure, and interactive communication platform.

1.3.1 EXISTING SYSTEM

Existing systems in educational institutions often rely on fragmented methods of communication, such as:

- Physical notice boards that are not accessible remotely.
- Department-specific chat groups with limited participation and poor organization.
- Institutional websites that lack discussion or interaction capabilities.
- No secure authentication or user-level access control for students and faculty.

These systems fail to promote campus-wide engagement and often lead to information gaps, low participation, and inefficiency in academic communication.

1.3.2 PROPOSED SYSTEM

The **Campus Connect** system aims to overcome these limitations by introducing an interactive, forum-based web platform. Key features include:

- **User Registration & Authentication:** Secure account creation and login using JWT to ensure verified access.
- **Post Creation & Commenting:** Users can create posts, reply through comments, and engage in topic-specific threads.
- **Upvoting System:** Promotes quality content through community-driven voting.
- **Department/Category-based Discussions:** Enables users to filter and participate in department-specific forums.
- **Responsive Web Interface:** Ensures accessibility across devices and browsers.

The proposed system not only enhances communication but also promotes a collaborative learning culture within the academic community.

1.4 SCOPE OF THE PROJECT

The **Campus Connect** system encompasses the following scope:

- **User Management:** Registration, login, and authentication for students and faculty.
- **Forum Management:** Post creation, commenting, and upvoting features for discussions.
- **Security:** Implementation of JWT-based authentication and secure data handling using Supabase.
- **Responsive Design:** A dynamic and intuitive web interface developed using React for seamless access.

- **Scalability:** The system can be extended to include features like media uploads, direct messaging, notifications, and event announcements.

Campus Connect serves as a foundation for building a comprehensive academic communication ecosystem.

1.5 REPORT ORGANISATION

The current introductory section provides a brief introduction about the chapter.

Chapter 1: Introduction

This section focuses on the purpose and scope of the proposed system of Online banking. It also highlights the limitations of the existing systems with regards to the proposed system.

Chapter 2: System requirement Specification

This chapter focuses on the specific requirement and aims of the project upon completion.

Chapter 3: System Design

This chapter delineated the software life cycle model used in the development of the product and reason(s) for opting the model by stating the advantages and disadvantages.

Chapter 4: Implementation

This chapter deals with the implementation of the project where the snapshots of execution steps are shown.

Chapter 5: Testing

This chapter deals with the testing of the system.

Chapter 6: Conclusion

This section deals with the conclusion that is derived after implementing the final system.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

The purpose of the literature survey is to review existing technologies, platforms, and research studies that are relevant to the development of Campus Connect — a college-based discussion forum. A literature survey helps to understand the current trends, identify gaps, and justify the need for a new system that addresses limitations in existing solutions.

Social platforms and online forums have become increasingly important in educational environments. Many studies highlight the effectiveness of digital forums in enhancing student engagement, knowledge sharing, and collaborative learning.

2.2 REVIEW OF EXISTING SYSTEMS

1. **Reddit**
   Reddit is a popular social platform where users can create posts, comment, and upvote content within topic-based communities. While it encourages community participation and content curation, it is too general-purpose for a college environment and lacks features tailored to academic needs such as department-based discussions and controlled access for verified students.

2. **Facebook Groups / WhatsApp Groups**
   Informal communication in college is often managed through Facebook or WhatsApp groups. While these provide instant messaging capabilities, they suffer from cluttered discussions, lack of structure, and difficulty in tracking posts or retrieving relevant information. Security and access control are also limited in these platforms.

2.3 RESEARCH AND TECHNOLOGY TRENDS

- **Integration of Web Technologies**
  Modern web frameworks like **React** and backend technologies like **Node.js** allow the development of responsive and scalable web applications. Real-time communication, secure authentication, and interactive UI components are achievable with these technologies.

- **Database Solutions for Academic Platforms**
  Cloud databases like **Supabase** and **PostgreSQL** provide secure storage, user management, and real-time data updates. Using ORMs like **Drizzle** simplifies data manipulation and enforces data integrity.

- **Security and Authentication**
  The use of **JWT-based authentication** has become standard for web applications, ensuring secure and session-based access control, which is particularly important for academic systems where data privacy is critical.

2.4 SUMMARY OF LITERATURE SURVEY

From the review, it is evident that:

- Existing social platforms do not fully address the need for structured, college-specific forums.
- Informal communication tools lack organization, tracking, and controlled access.
- There is an opportunity to develop a **secure, department-aware, and interactive web forum** tailored for colleges that integrates authentication, posting, commenting, and upvoting.
- Modern technologies such as **React, Node.js, Supabase, Drizzle ORM, and JWT** provide an ideal framework to implement such a system.

This survey justifies the development of **Campus Connect** as a platform that bridges the gap between traditional LMS and general social media, providing a secure, interactive, and academic-focused forum for students and faculty.

CHAPTER 3
SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

3.1.1 PURPOSE

The purpose of the **Campus Connect** system is to develop a secure, efficient, and user-friendly web application that serves as a centralized platform for academic discussions and community engagement within a college environment. The system allows users to register, authenticate, create posts, comment on discussions, and upvote relevant content — promoting collaboration, communication, and knowledge sharing among students and faculty members.

The key objectives of the system include:

- Providing a platform for students and faculty to share knowledge, discuss topics, and exchange ideas.
- Implementing secure authentication using **JWT** to protect user identities and maintain access control.
- Reducing communication fragmentation by replacing traditional notice boards and chat groups with a unified forum.
- Ensuring 24/7 accessibility through a responsive and device-independent web interface.
- Encouraging engagement and peer interaction through upvotes and discussion threads.

3.1.2 SCOPE

The **Campus Connect** system is a **web-based application** designed to connect members of an academic community through structured, topic-based discussions. It aims to modernize college communication by providing a digital platform for sharing knowledge, announcements, and discussions in a secure, scalable, and interactive environment.

Functional Scope:

- **User Registration & Login:** Secure registration and authentication using credentials and JWT tokens.
- **Post Creation:** Users can create posts on various academic and non-academic topics.
- **Commenting System:** Allows users to comment on posts, enabling threaded discussions.
- **Upvoting:** Users can upvote posts and comments to highlight relevant content.
- **Category/Department-based Forums:** Enables content filtering and organization based on subjects or departments.
- **Security & Data Protection:** Secure user data handling using Supabase and Drizzle ORM for reliable storage and structured querying.

- **Responsive Web Interface:** Accessible from any modern web browser on desktop, tablet, or mobile devices

3.1.3 OVERVIEW

The **Campus Connect** web application provides a digital forum tailored for the academic community to foster open communication, collaboration, and peer learning. Through an intuitive interface, users can post questions, share announcements, and participate in discussions across different topics or departments.

The system integrates **JWT-based authentication** for secure access control and leverages **Supabase** for database management, ensuring data consistency and reliability. Developed with **React**, **Node.js**, and **Express**, the system offers a smooth, responsive experience across devices.

By digitizing and centralizing communication, Campus Connect eliminates the inefficiencies of traditional communication methods such as notice boards or departmental emails. It enhances student engagement, promotes a culture of open discussion, and creates a digital archive of academic interactions.

Future improvements can include direct messaging, push notifications, media attachments, and integration with college management systems. Campus Connect serves as a scalable foundation for future digital campus initiatives.

3.2 OVERALL DESCRIPTION

3.2.1 PRODUCT PERSPECTIVE

User Interface: Web Application
Software Requirements: Any operating system (Windows, macOS, or Linux) with a modern web browser (Chrome, Firefox, Safari, Edge).
Backend Environment: Node.js and Express.js
Database: Supabase (PostgreSQL) managed through Drizzle ORM
Frontend Framework: React

3.2.2 SYSTEM FEATURES

The **Campus Connect** system provides a range of features designed to improve accessibility, security, and engagement across the college community. The major system features include:

1. User Registration & Authentication

- Secure registration and login using JWT authentication.
- Role-based user management for students, faculty, and administrators.
- Password encryption and validation for data protection.

2. Post Creation & Discussion Threads

- Users can create posts under specific topics or categories.
- Supports text-based posts for academic queries, announcements, and discussions.
- Departmental categorization to maintain organized discussions.

3. Commenting System

- Enables users to comment on posts, supporting nested/threaded discussions.
- Real-time updates on new comments for better interaction.
- Ability to reply to individual comments to maintain conversation flow.

4. Upvoting & Engagement

- Allows users to upvote posts and comments to prioritize valuable content.
- Ranking system based on user engagement and community feedback.
- Promotes quality discussions and helps highlight important threads.

5. Security & Data Protection

- **JWT-based authentication** for user verification.
- **Supabase** ensures secure data storage and authorization rules.
- Session timeout and token refresh mechanisms to prevent unauthorized access.
- Sanitization of user input to avoid SQL injection and XSS attacks.

6. Responsive Web Interface

- Developed using **React** for a fast and interactive experience.
- Fully responsive design compatible with mobile, tablet, and desktop browsers.
- Clean and minimal UI inspired by modern forum designs (e.g., Reddit).

CHAPTER 4

DESIGN

4.1 SOFTWARE DEVELOPMENT MODEL

Agile Software Development

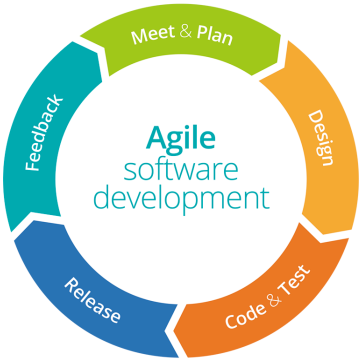

Fig 4.1 Agile Software Development

An agile methodology implies software that is incremental; it increases by regular additions. This approach offers a new version or approach in short intervals. This agile process is different from the other traditional approach to software development where the needs and requirements of the users are compiled, and then, in the end, the software is built all at once. Agile Manifesto is a declaration that distinctly states the key values and principles that software developers should follow to guide their work. It is the base of the agile movement. These are the four values and twelve principles as guidelines.

4.2 Agile Values of Agile Manifesto:

1. Team and Communication Chosen Over Procedure and Tools

The first value places more emphasis on teamwork and communication. As we all know that to build software, a team of people is required and not tools. An individual may have a sophisticated set of tools, but he needs to work with a team effectively with productive interactions to develop software. Tools and processes are definitely the keys to developing software, but they need to be in the hands of a qualified team to get the result and not the other way round.

2. Working Software Over Comprehensive Documentation

A lot of time was spent on the documentation of the product development, like the technical specification, technical requirements, test plans, design documents, and approval required for each. The kind of documentation created for these was very detailed, and many of which were not even referred to during the project process. The team wanted the finished product to be as per the specification, so the documents were focused on a lot. But the end product would still be different as the relevance was lost.

Agile streamlines these documents in such a form that gives the developer only what is needed to do the work without getting distracted by the minute details. Agile does not say that documentation is not necessary. All it says is that working software is preferred by customers rather than a document which perhaps will not be looked into also. Whenever a need for change arises, the agile team readily accepts it and makes continuous changes.

3. Communication With Client Preferred to Signed Agreements

Successful development teams work closely with their customers and communicate with them regularly. As it is only through listening to your customers that you get feedback, and you will understand what they exactly want from your product. It will be extremely beneficial if the legal relationship with a customer can be kept separate from a personal relationship. This will encourage communication, and knowing their thoughts, opinions, and preferences will result in a more satisfying product.

Communication also helps the client refine their vision and redefine their requirements if required during the course of the project. In traditional software development, once the requirements have been decided upon, the product can only be seen when it is ready. But agile defies this by allowing customer involvement in the entire procedure.

4. Readily Accepting Change Rather than Following a Strict Plan

It is believed that change is expensive and time-consuming and should be avoided at all costs. This is so because we give unnecessary focus to documentation and elaborate plans to deliver. We tend to stick to timelines and product specifications more than required. It is essential to realize that change is inevitable. It will be extremely beneficial to accept change and plan for it. There should always be room for change; otherwise, the plan will soon become obsolete.

According to agile, change is not an expense but necessary feedback that results in an improved project. A feedback-initiated change adds value to the project. Agile methodologies allow the agile team to change the process while in progress and make it fit for the team rather than the other way round. So, the first principle of communication with the entire team has to be referred to here. The developers of the team should be kept aware of the changes. The ultimate goal of your project should always be kept in mind, and if there is any sort of hindrance, the team should be made aware of it and reminded of these 4 values.

12 Principles of Agile Development:

These principles are a test to define whether you are agile:

- Satisfying Customers Through Timely and Constant Delivery of Valued Work: Customers are happier if they receive working software at regular intervals rather than waiting for long intervals between releases.
- Accept Change During the Entire Process: Whenever a requirement or feature needs to be changed, it should be done so readily.

- Release Effective Software Frequently: Since the team operates in software sprints, it ensures regular delivery of working software.
- Collaboration Between Business Stakeholders and Developers: Better decisions are made when the business and technical team work together.
- Motivate, Support, and Trust: Motivating the team is the key here. Whenever a project starts, total support to the team and encouraging ambiance and faith in the team will keep them going.
- One to One Discussions: The most important method of passing on any information to the entire team is by having one on one discussions.
- Software is Working: Progress can be measured only by software that is successfully working at that time.
- Agile Procedures Boost Continuous Development: The promoters, planners, and customers should all be able to progress.
- Importance to Technique: Right skills and good design ensure constant improvement of product, maintaining peace, and sustaining change.
- Keep it Simple: Developing just enough to get the job done, which is right now.
- Self-organizing Teams: Self-organizing Teams are where the best architecture, requirements, and designs emerge.
- Regular Reflections on How to Become More Effective: The team should keep working towards becoming more productive and adapt accordingly

4.2 DATA DESIGN

4.2.1 ENTITY RELATIONSHIP DIAGRAM

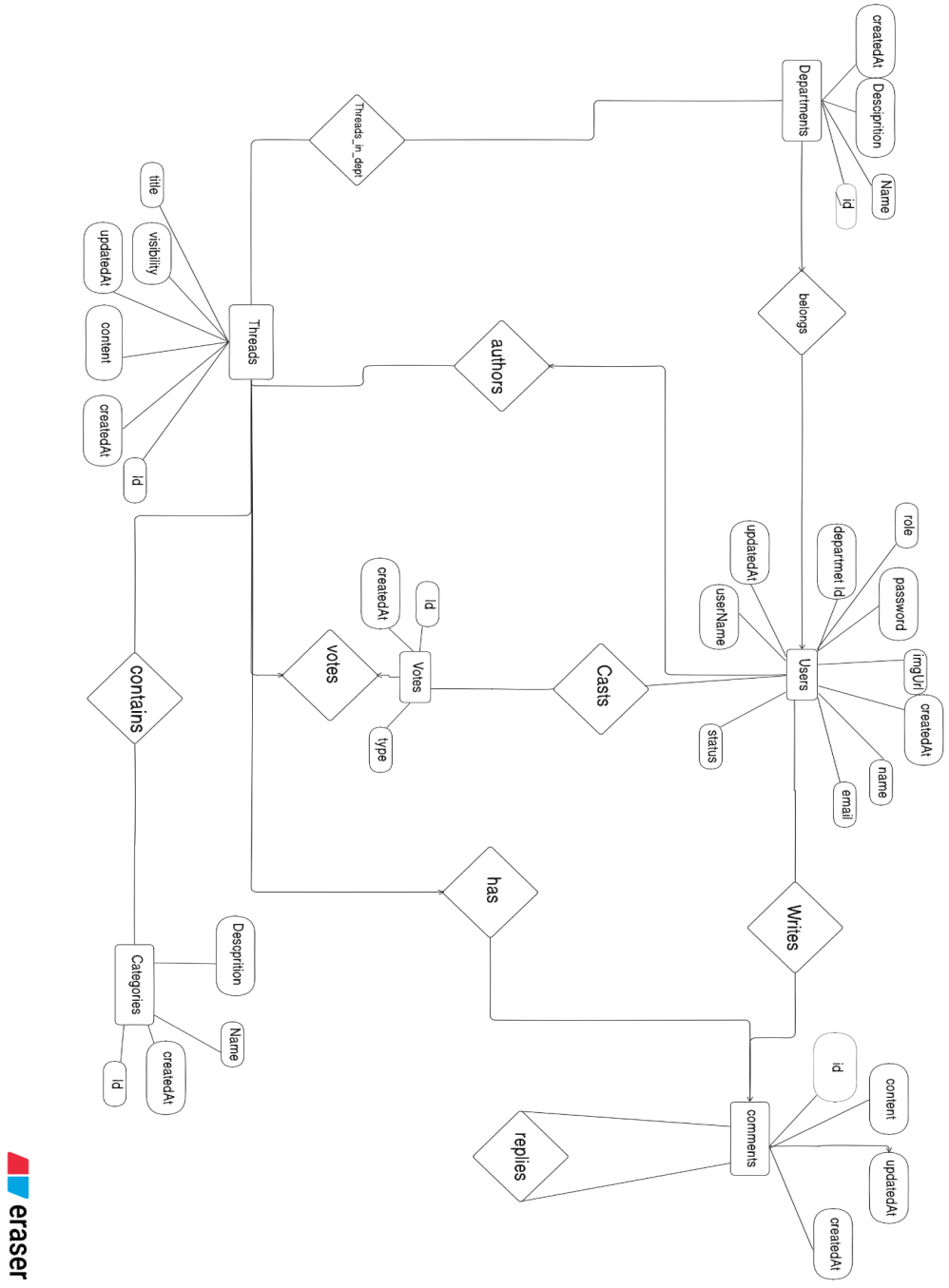

Fig 4.2

The ER (Entity-Relationship) Diagram represents the logical structure of the database. It shows the main entities — Users, Posts, Comments, and Votes — along with their attributes and relationships. This diagram helps in designing a normalized database for efficient data storage and retrieval.

4.3 FUNCTIONAL DESIGN DESCRIPTION

4.3.1 DATA FLOW DIAGRAMS

4.3.1.1 CONTEXT LEVEL DIAGRAM

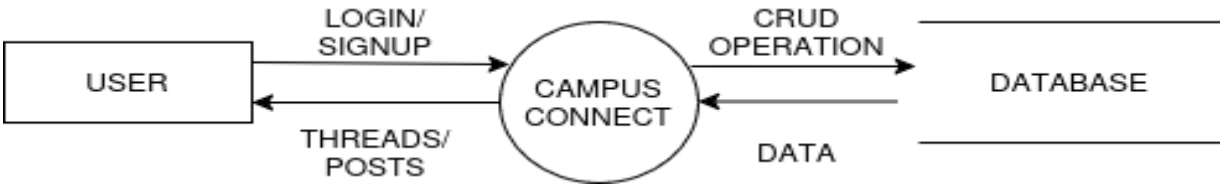

Fig 4.3

The Context Diagram provides a high-level overview of the system and its interactions with external entities such as Users and Database. It shows Campus Connect as a single process and depicts how users send requests and receive responses, establishing the system boundaries.

4.3.1.2 LEVEL 0 DIAGRAM

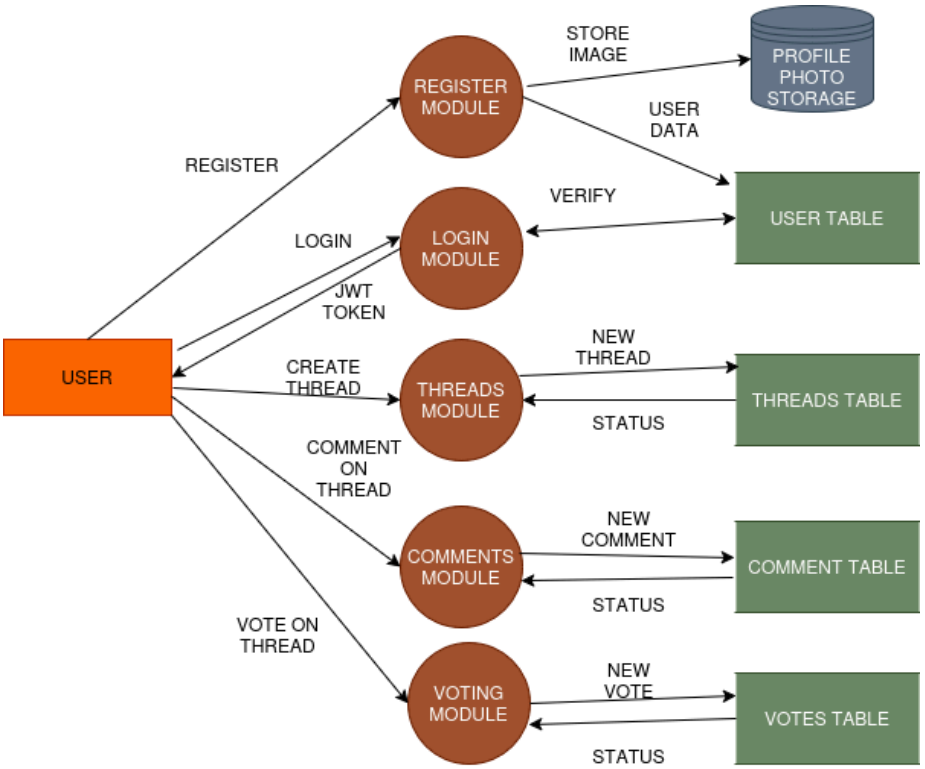

Fig 4.4

The Level 0 DFD illustrates the major processes of the system, data flows, and external entities. It shows how Users interact with core processes like Authentication, Post Management, and Comment Handling, with the database serving as the central data store.

4.3.1.3 LEVEL ONE DIAGRAM

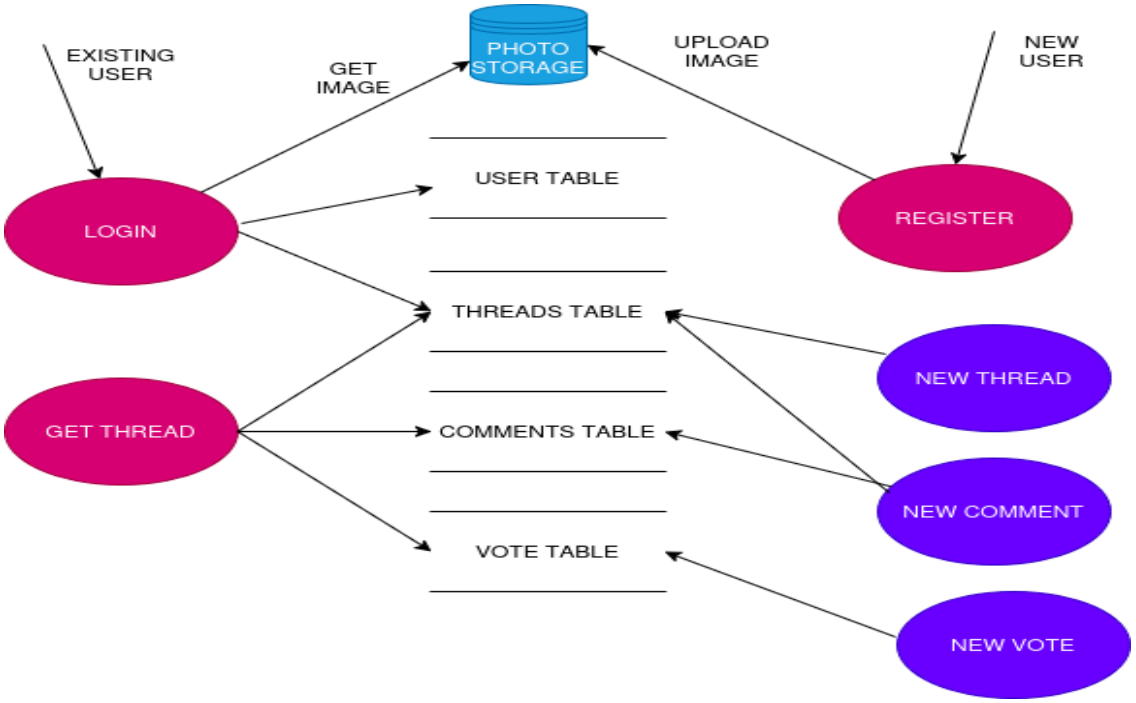

Fig 4.5

The Level 1 DFD decomposes Level 0 processes into more detailed subprocesses.

4.3.2 ACTIVITY DIAGRAM

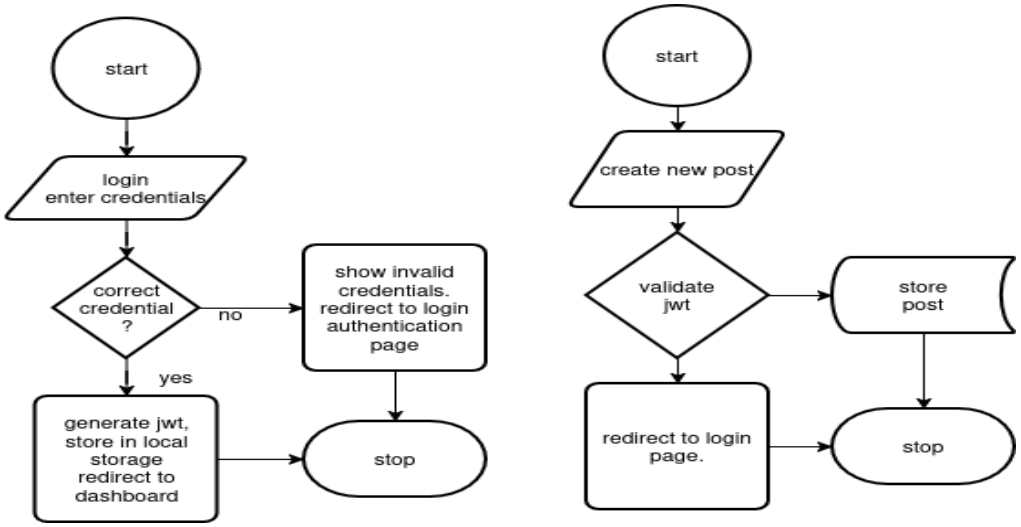

Fig 4.6

The Activity Diagram models the dynamic workflow of the system. It illustrates step-by-step user activities such as login and creating posts including decision points like invalid login, making the process flow easy to understand.

4.3.3 SEQUENCE DIAGRAM

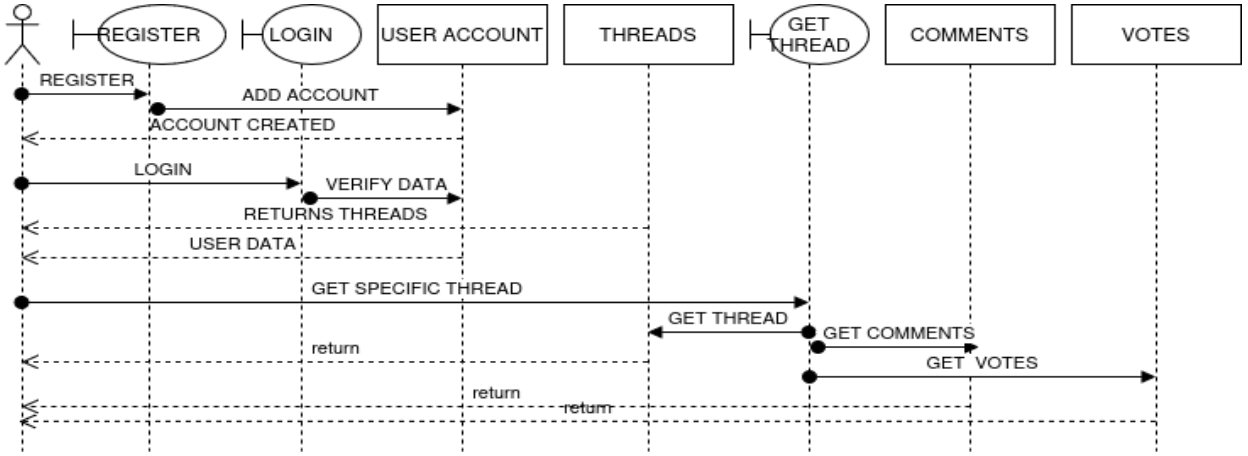

Fig 4.7

The Sequence Diagram models the interaction between objects over time for a specific functionality. For example, in the Post Creation sequence, it shows the User sending a request to the Frontend, which calls the Backend API, updates the Database, and returns a confirmation response to the User Interface. Sequence diagrams help visualize message flows and the order of operations within the system.

4.3.3.4 CLASS DIAGRAM

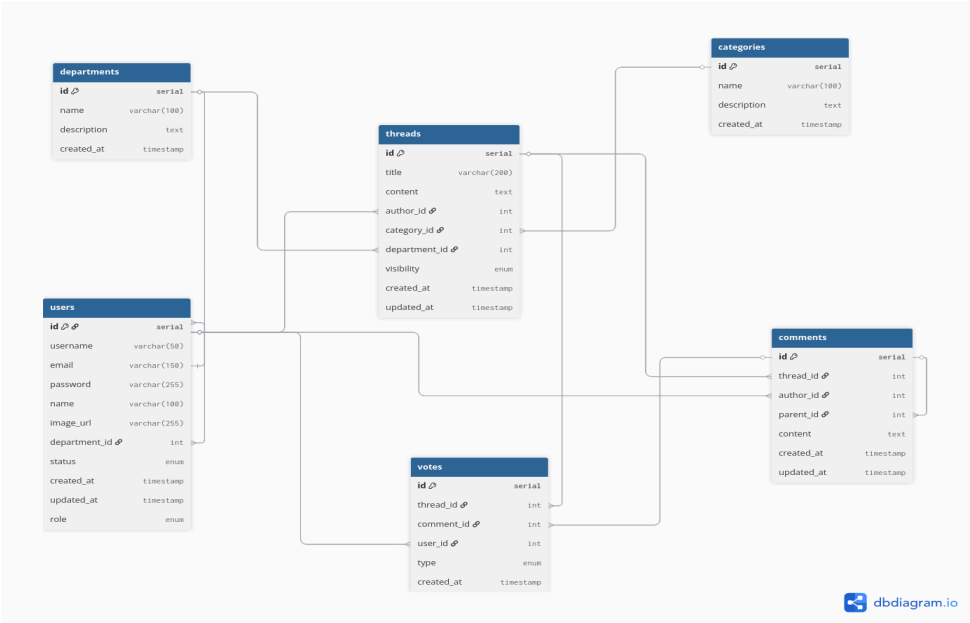

Fig 4.8

We will be using Object-Relational Mapping (ORM), which will reduce the need for separate modules. Most functionalities will be handled directly by the ORM, minimizing the use of separate classes or functions. Instead of distinct modules, API route points will be assigned designated functions to manage operations efficiently**.**

4.3.3.5 USE CASE

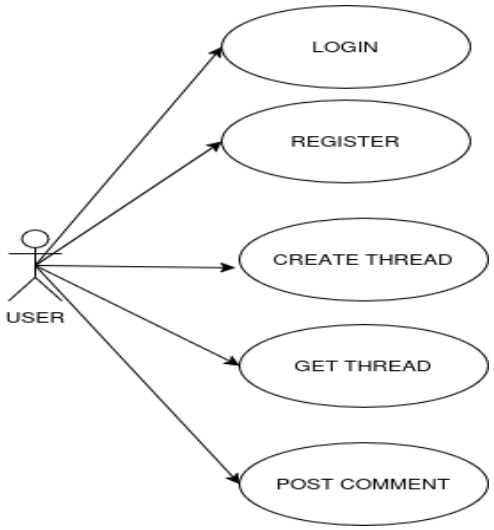

Fig 4.9

The Use Case Diagram represents the functional requirements of the Campus Connect system from a user perspective. This diagram helps in understanding what the system does without specifying how it is implemented.

CHAPTER 5

IMPLEMENTATION

5.1 OVERVIEW OF THE TECHNOLOGY USED

The Campus Connect system leverages modern web technologies to deliver a secure, efficient, and scalable platform for academic communication. The application follows a three-tier architecture comprising the frontend, backend, and database layers, each implemented using specialized tools and frameworks.

- React.js (Frontend):
  React is a JavaScript library used for building responsive and interactive user interfaces. It provides a component-based architecture that allows modular development and efficient rendering through its virtual DOM. React enables seamless navigation and a dynamic user experience within the Campus Connect platform.

- Node.js (Backend Runtime):
  Node.js is a powerful JavaScript runtime environment used to build the server-side of the application. It allows handling of concurrent client requests efficiently through its non-blocking I/O model. Node.js provides the foundation for executing backend logic and managing server operations.

- Express.js (Web Framework):
  Express is a lightweight and flexible web framework built on Node.js, used to design RESTful APIs for the system. It simplifies routing, middleware integration, and request handling, enabling the backend to efficiently manage authentication, post creation, commenting, and upvoting functionalities.

- Supabase (Database & Authentication):
  Supabase is an open-source backend-as-a-service platform built on PostgreSQL. It

provides a secure, reliable, and scalable database solution for storing user, post, and comment data. Supabase also supports built-in authentication and real-time data synchronization, enhancing system performance and integrity.

- Drizzle ORM (Object Relational Mapper):
  Drizzle ORM is used to interact with the Supabase PostgreSQL database in a structured and type-safe manner. It simplifies database schema definition, migrations, and queries, ensuring maintainable and efficient data management within the system.

- JWT (JSON Web Tokens):
  JWT is used to implement secure authentication and authorization mechanisms. It ensures that only verified users can access protected routes, post content, or interact within the forum. Tokens are generated upon successful login and validated on each request to maintain session security.

- Tailwind CSS (Styling Framework):
  Tailwind CSS is used for designing a clean, responsive, and modern user interface. It enables rapid UI development with utility-first **styling,** ensuring consistent design across devices.
  Together, these technologies enable Campus Connect to function as a robust and interactive web application that promotes collaboration and communication within a college ecosystem.

5.2 CODE SNIPPETS

```
import { db } from "../models/db.mjs";
import { voteEnum, votes } from "../models/schema.mjs";
import { eq, and } from "drizzle-orm";

export const createVote = async (req, res) => {
  try {
```

```
import MobileNav from "@/components/Dock.jsx";
import {Button} from "@/components/ui/button"
import {Card, CardContent} from "@/components/ui/card"
import {Avatar, AvatarFallback, AvatarImage} from
"@/components/ui/avatar"
import {
```

```
  const { threadId, commentId, type } = req.body;
  const userId = req.user.id;
  if (!threadId && !commentId) {
        return res.status(400).json({ error: "Either
threadId or commentId is required" });
  }
  if (threadId && commentId) {
        return res.status(400).json({ error: "Cannot vote
on both a thread and a comment simultaneously" });
  }
  const existingVote = await db.select().from(votes).where(
        and(
        eq(votes.userId, userId),
        threadId ? eq(votes.threadId, threadId) :
undefined,
        commentId ? eq(votes.commentId, commentId) :
undefined
        )
  );
  if (existingVote.length > 0) {
        return res.status(409).json({ error: "User has
already voted on this item" });
  }
  const newVote = await db.insert(votes).values({
        threadId,
        commentId,
        userId,
        type,
  }).returning();
  res.status(201).json({ message: "Vote recorded
successfully", vote: newVote[0] });
  } catch (e) {
  console.error("Vote creation error:", e);
  res.status(500).json({ error: "Internal server error" });
  }};
export const deleteVote = async (req, res) => {
  try {
  const { id } = req.params;
  const userId = req.user.id;
  const vote = await
db.select().from(votes).where(eq(votes.id, id));
```

```
  Tabs,
  TabsContent,
  TabsList,
  TabsTrigger,
} from "@/components/ui/tabs"
const BASE = 'https://campus-connect-98bf.onrender.com'
import Posts from "@/components/Posts.jsx";
import {useEffect, useState} from "react";
import {useNavigate} from "react-router-dom";
import Comments from "@/components/Comments.jsx";

export default function ProfilePage() {
  const navigate = useNavigate();
  const [token, setToken] = useState("");
  const [user, setUser] = useState({});
  const [userData, setUserData] = useState();
  async function getUserProfile() {
        if (token) {
        try {
        const res = await fetch(`${BASE}/profile`, {
              method: "GET",
              headers: {
              Authorization: `Bearer ${token}`,
              "Content-Type": "application/json"
              },
        })
        const data = await res.json()
        if (res.ok) {
              console.log(data);
              setUser(data);
        } else if (res.status === 401) {
              localStorage.clear()
              navigate("/auth")
        } else {
              console.error(data);
        }
        } catch (err) {
        console.error(err)
        }}}
  async function getUserProfileExtended() {
        if (token) {
```

```
  if (vote.length === 0) {
            return res.status(404).json({ error: "Vote not
found" });
  }
  if (vote[0].userId !== userId) {
            return res.status(403).json({ error: "You are not
authorized to delete this vote" });
  }
  await db.delete(votes).where(eq(votes.id, id));
  res.status(200).json({ message: "Vote deleted
successfully" });
  } catch (e) {
  console.error("Delete vote error:", e);
  res.status(500).json({ error: e});
  }
};
export const countVote = async (req,res) => {
  try{
  const { id } = req.params;
  const upvote = await
db.$count(votes,and(eq(votes.threadId,id),eq(votes.type,
"upvote")));
  const downvote = await db.$count(votes,
and(eq(votes.threadId,id),eq(votes.type,"downvote")));
  res.json({upvote: upvote, downvote: downvote});
  }catch (e){
  res.status(500).json({error: e.message})
  }}
```

```
            try {
            const res = await
fetch(`${BASE}/profileextend`, {
                        method: "GET",
                        headers: {
                        Authorization: `Bearer ${token}`,
                        "Content-Type": "application/json"
                        },
            })
            const data = await res.json()
            if (res.ok) {
                        console.log(data);
                        setUserData(data);
            } else if (res.status === 401) {
                        localStorage.clear()
                        navigate("/auth")
            } else {
                        console.error(data);
            }
            } catch (err) {
            console.error(err)
            }}}
  useEffect(() => {
            const storedToken =
localStorage.getItem("authToken");
            if (!storedToken) {
            navigate("/auth");
            } else {
            setToken(storedToken);
            }
  }, [navigate]);
```

5.3 User Interface

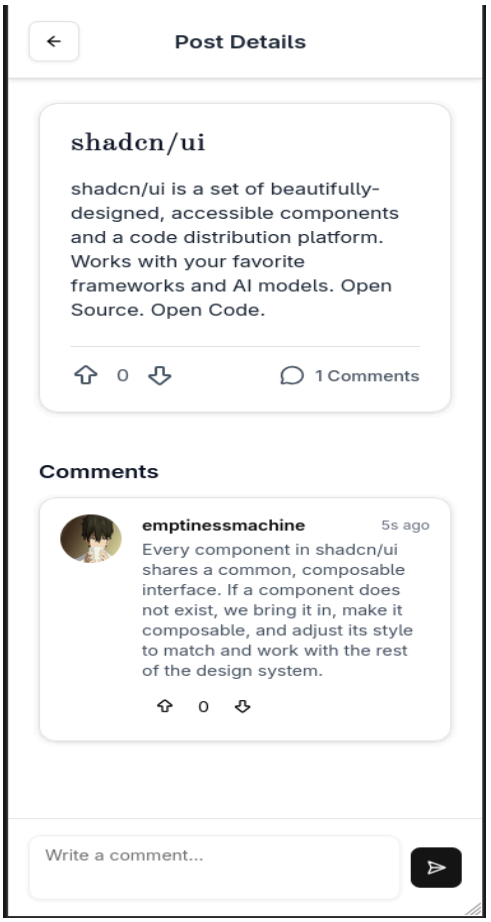

Fig 5.1 Thread UI

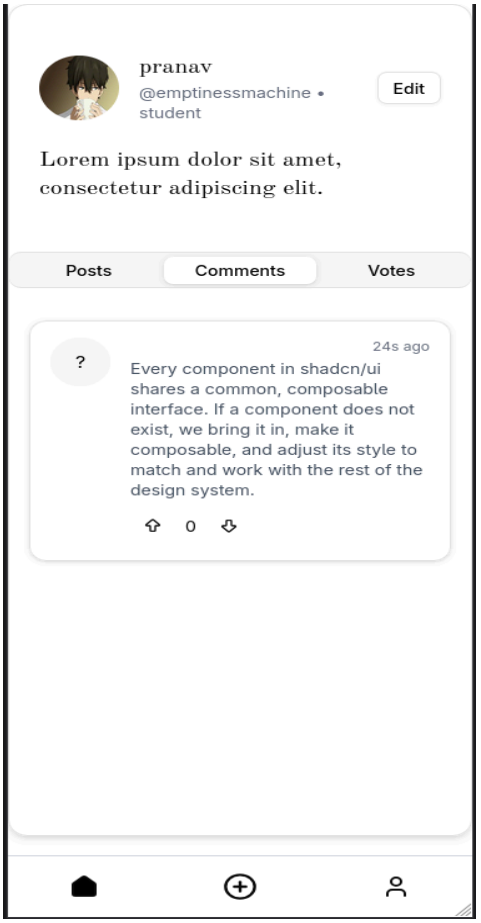

Fig 5.2 Profile UI

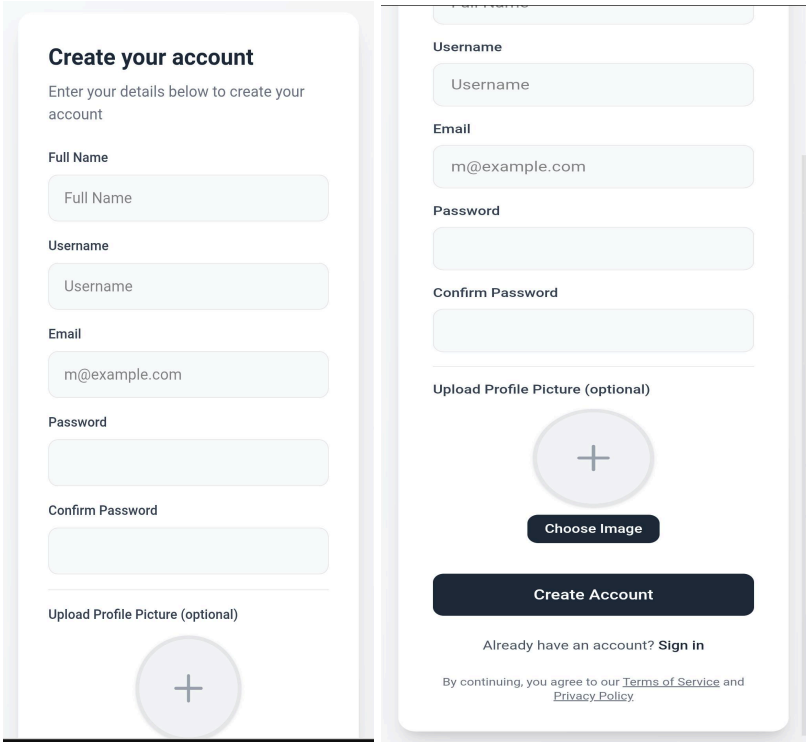

Fig 5.3 Sign Up

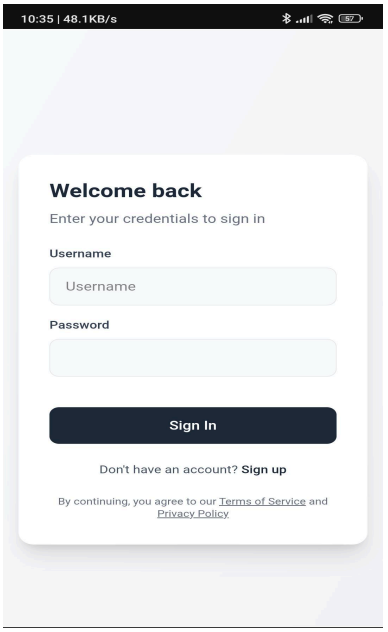

Fig 5.4 Login UI

CHAPTER 6

TESTING

6.1 TESTING INTRODUCTION

Testing is an essential phase of the software development process that ensures the application functions as intended and meets all specified requirements. The testing process for the Campus Connect web application was conducted to verify the accuracy, reliability, and performance of the system's features.

Different levels of testing such as unit testing, integration testing, and system testing were performed to ensure that all modules including authentication, post creation, commenting, and upvoting work together seamlessly. The testing aimed to identify and fix any bugs or inconsistencies before final deployment to ensure a smooth user experience.

6.2 TEST CASES

Test Case 1: User Registration

Description: Verify that a new user can register successfully.

Preconditions: The user does not already exist in the database.

Test Steps:

1. Navigate to the registration page.
2. Enter valid details such as name, email, and password.
3. Click on the "Register" button.

   Expected Result: The user account is created successfully and stored in the database.

   Actual Result: Pass

Test Case 2: User Login

Description: Verify that registered users can log in successfully.

Preconditions: The user must have a valid registered account.

Test Steps:

1. Navigate to the login page.
2. Enter valid email and password.
3. Click on the “Login” button.

   Expected Result: The user is authenticated, and a JWT token is generated for secure access.

   Actual Result: Pass

Test Case 3: Post Creation

Description: Verify that logged-in users can create a new post.

Preconditions: User must be logged in.

Test Steps:

1. Navigate to the “Create Post” section.
2. Enter a post title and description.
3. Click on “Submit.”

   Expected Result: Post is added to the database and displayed in the feed.

   Actual Result: Pass

Test Case 4: Comment on Post

Description: Verify that users can add comments to an existing post.

Preconditions: User must be logged in, and at least one post should exist.

Test Steps:

1. Open a post from the feed.
2. Enter a comment in the text field.

3. Click on “Submit Comment.”
   Expected Result: The comment appears under the post instantly.
   Actual Result: Pass

Test Case 5: Upvote Post

Description: Verify that users can upvote a post.

Preconditions: A valid post must exist in the feed.

Test Steps:

1. Log in to the system.
2. Navigate to a post and click the “Upvote” button.
   Expected Result: The upvote count increases by one and is reflected in the database.
   Actual Result: Pass

Test Case 6: Invalid Login

Description: Verify that login fails with incorrect credentials.

Preconditions: User exists in the database.

Test Steps:

1. Open Login page
2. Enter wrong username or password
3. Click “Login”
   Expected Result: System displays error “Invalid credentials” and denies access.
   Actual Result: Pass

Test Case 7: Empty Fields on Registration

Description: Verify that user cannot register without filling all required fields.

Preconditions: None

Test Steps:

1. Open Register page
2. Leave fields empty
3. Click “Register”

   Expected Result: System shows validation error messages.

   Actual Result: Pass

Test Case 8: Duplicate Username on Signup

Description: Verify that registration fails if username already exists.

Preconditions: Username exists in DB

Test Steps:

1. Enter already-used username
2. Click “Register”

   Expected Result: Show error "Username already exists".

   Actual Result: Pass

Test Case 9: Fetch User Profile

Description: Check if logged-in user profile loads correctly.

Preconditions: Valid authentication token

Test Steps:

1. Navigate to Profile page

   Expected Result: User info such as name, email, avatar loads correctly.

   Actual Result: Pass

Test Case 10: Create Post Without Title

Description: Verify system prevents empty title post creation.

Preconditions: User logged in

Test Steps:

1. Go to Create Post
2. Leave title empty
3. Enter content
4. Create post

   Expected Result: Error “Must Enter Content” or “Title required” appears.

   Actual Result: Pass

Test Case 11: Create Post Without Content

Description: Ensure post cannot be created without content.

Preconditions: User logged in

Test Steps:

1. Enter title
2. Leave editor empty
3. Click Submit

   Expected Result: Error "Must Enter Content"

   Actual Result: Pass

Test Case 12: Private Post Creation

Description: Verify selecting "Private" visibility sets correct type.

Preconditions: User logged in

Test Steps:

1. Select "Private" in visibility dropdown
2. Create post

   Expected Result: Post stored in DB with private flag.

   Actual Result: Pass

Test Case 13: Comment Without Login

Description: Verify users cannot comment without authentication.

Preconditions: User logged out

Test Steps:

1. Open post
2. Try typing comment
3. Click “Submit Comment”

   Expected Result: Redirect to Login page

   Actual Result: Pass

Test Case 14: Load Comments on Post

Description: Verify comments load correctly on post page.

Preconditions: At least one comment exists

Test Steps:

1. Open any post

   Expected Result: All associated comments are displayed.

   Actual Result: Pass

Test Case 15: Delete Own Post

Description: Verify user can delete their own post.

Preconditions: User is author of post

Test Steps:

1. Open profile
2. Select own post
3. Click delete

   Expected Result: Post removed from feed and DB

   Actual Result: Pass

Test Case 16: Unauthorized Post Deletion

Description: Verify user cannot delete another user's post.

Preconditions: User A logged in, User B's post

Test Steps:

1. Try deleting another user's post

   Expected Result: System shows “Unauthorized”

   Actual Result: Pass

Test Case 17: Expired JWT Token

Description: Verify expired token redirects to login.
Preconditions: Token expired
Test Steps:

1. Try accessing profile or create post
   Expected Result: Redirect to login page
   Actual Result: Pass

Test Case 18: Invalid Image Format

Description: Check if system rejects unsupported file types.

Preconditions: User logged in

Test Steps:

1. Upload non-image file (.exe, .txt)
   Expected Result: "Invalid file type" error
   Actual Result: Pass

Test Case 19: Upload Avatar

Description: Ensure profile picture upload works.

Preconditions: User logged in

Test Steps:

1. Go to Register or Edit Profile
2. Upload image
3. Submit
   Expected Result: Image preview visible and avatar saved on server
   Actual Result: Pass

Test Case 22: Fetch Feed (Homepage Posts Loading)

Description: Verify that the homepage successfully loads all public posts from the server.

Preconditions: At least one post exists in the database.

Test Steps:

1. Log in to the application (or stay logged out if public access is allowed).
2. Navigate to the homepage (Feed).
3. Wait for posts to load.

Expected Result:

- All public posts should be fetched from the backend.
- Posts should appear in the feed with title, author, and preview content.
- No errors should appear in console or UI.

Actual Result: Pass

CHAPTER 7

CONCLUSION

The Campus Connect web application developed in this project provides an efficient and interactive platform for fostering communication and collaboration within a college community. By integrating essential features such as user authentication, post creation, commenting, and upvoting, the system effectively replicates a campus-focused discussion forum similar to Reddit, promoting knowledge sharing and student engagement.

The use of React for the frontend, Node.js and Express for the backend, and Supabase with Drizzle ORM for data management ensures a modern, scalable, and responsive application architecture. This combination provides secure data handling, real-time interaction, and a seamless user experience across devices.

The successful implementation of Campus Connect demonstrates how technology can enhance campus communication and collaboration by digitizing student discussions and academic interactions. Future enhancements could include features such as direct messaging, media uploads, topic-based communities, and push notifications to further improve interactivity and user engagement.

REFERENCES

1. https://supabase.com/docs
2. https://react.dev/
3. https://ui.shadcn.com/