MYMATE

A PROJECT REPORT
for
Mini Project (KCA 353) Session

(2024-25)

Submitted by

Abhinav Kajla
(2300290140004)
Ashutosh
(2300290140043)
Divyanshi Bhadauria
(2300290140058)
Divyansh Chauhan
(2300290140056)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Mr. Shish Pal Jatav

(Assistant Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(May 2024)

DECLARATION

We hereby declare that the work presented in this report entitled **“Mymate”**, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma from any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, and results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported int he report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, We shall be fully responsible and answerable.

Name –

Abhinav Kajla (2300290140004)
Ashutosh (2300290140043)
Divyanshi Bhadauria (2300290140058)
Divyansh Chauhan (2300290140056)

CERTIFICATE

It is to certify that **Abhinav Kajla (2300290140004), Ashutosh (2300290140043), Divyanshi Bhadauria (2300290140058), Divyansh Chauhan (2300290140056)** has carried out the project work having "**Mymate** (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Mr. Shish Pal Jatav	**Dr. Arun Kumar Tripathi**
Assistant Professor	**Head**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

ABSTRACT

The project that is proposed aims to create a website platform similar to LinkedIn but designed specifically focusing on the enhanced interaction between junior and senior students within a college. We do understand the importance of the peer-to-peer support, mentorship, and knowledge sharing in academic environments, our tis project would be enacting as a platform seeks to bridge the gap between different year groups, enabling smooth communication and collaboration.

By this we can create an ecosystem where the students can learn from one other and help each other to achieve everyone's goal, professional growth. The platform will allow users to create user's personal profiles that highlight their academic achievements, skills, extracurricular activities, and ongoing projects. It would be acting as an opportunity for the seniors to showcase their experience and skills and offer mentorship. It would help the seniors to develop the leadership skills, networking skills along the capabilities to work along with the juniors. On the other hand, for the juniors, it would be possible for them to view and surf around for different working profiles and work cultures, cope with the academic challenges and career-related questions.

This interaction aims to create a sense of community and foster collaboration, enabling a seamless transfer of knowledge from experienced students to those just starting their academic journey.

Other functionalities we would be aiming would be to provide the real-time messaging, group discussion based on the fields of internet and update the students for the upcoming organized events.

Other features of the platform will include real-time messaging, group discussions based on fields of interest, and activity feeds that display updates about relevant events, achievements, or discussions.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. Shish Pal Jatav** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, the Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Abhinav Kajla
(2300290140004)
Ashutosh
(2300290140043)
Divyanshi Bhadauria
(2300290140058)
Divyansh Chauhan
(2300290140056)

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

Welcome to MyMate, a platform where connections are forged, experiences are shared, and opportunities are discovered. MyMate is a professional networking web application that bridges the gap between juniors and seniors, empowering users to engage, learn, and grow together.

Unlike traditional professional platforms, MyMate emphasizes the value of mentorship and collaboration, creating a unique ecosystem where knowledge sharing and mutual support thrive. Whether you're a seasoned professional sharing your journey, a student seeking guidance, or someone building meaningful networks, MyMate is your space to connect and grow.

Through its carefully designed features, MyMate fosters a collaborative environment tailored to the needs of individuals aiming to enhance their professional and personal growth.

Objective

The primary objective of MyMate is to provide a platform for students and professionals to:

Connect with peers and mentors.

Share valuable resources and experiences.

Build a network to foster personal and professional growth.

MyMate serves as a bridge between individuals at different stages of their career journeys, creating opportunities for mentorship, collaboration, and knowledge sharing.

By focusing on intuitive design, robust functionalities, and seamless user interaction, MyMate aims to establish itself as a trusted space for professional networking and development.

Need for Blog Website

In a highly competitive and interconnected world, the need for professional networking platforms has never been more critical. Platforms like MyMate fill the gap by offering:

Mentorship Opportunities: Direct connections between juniors and seniors enable the sharing of advice, insights, and guidance.

Resource Sharing: A centralized place for users to access and share learning resources, articles, and opportunities.

Professional Development: A space for users to showcase skills, share achievements, and receive constructive feedback.

Community Building: A platform for like-minded individuals to engage in meaningful interactions, fostering a sense of community and belonging.

Unlike traditional platforms, MyMate caters specifically to students and early-career professionals, providing them with the tools they need to succeed in their respective fields.

Modules

The MyMate project consists of key modules designed to enhance user experience and foster meaningful connections. These include User Registration and Login for secure access, a dynamic Home Page showcasing curated content and updates, and Post Management allowing users to create, edit, like, and share posts. The Chat System enables real-time communication, while Profile Management lets users customize and update their personal and professional details. Additionally, the Follow System promotes networking by allowing users to connect and stay updated with others, supported by a Notification System that keeps them informed of activities like likes, comments, and new followers. Together, these modules create an interactive and engaging platform tailored for mentorship and professional growth.

- User Registration and Login
  - Secure user authentication using JWT and bcrypt for password hashing.
  - Email and password-based signup and login system.

- Home Page
  - Displays curated content such as recent posts, trending topics, and updates from followed users.

- Post Management
  - Users can create, like, share, save, and manage posts.
  - Includes functionalities for editing and deleting posts.

- User Interaction
  - A follow system allows users to connect with others and stay updated.
  - Chat functionality supports real-time communication.

- Profile Management
  - Users can view and update their profiles, including personal and professional details.

- Notifications
  - Alerts for likes, comments, shares, and new connections.
  -

1.1 Functionalities

User Registration and Login

- Account creation with validation and secure password storage.
- Login using JWT for authentication and cookies for session management.

Home Page

- Displays a feed of posts from followed users.
- Highlights trending topics and featured posts.

Post Creation and Interaction

- Users can create, edit, and delete posts.
- Options to like, share, and save posts for future reference.

Chat System

- Real-time messaging powered by WebSocket (e.g., Socket.io).
- User-friendly chat interface for seamless communication.

Profile Management

- Update user details such as bio, profile picture, and professional information.
- View other users’ profiles for networking purposes.

Follow System

- Users can follow/unfollow others to build their network.
- Notifications for new followers.

Security Features

- HTTPS for secure communication.
- Password encryption using bcrypt.
- Authentication tokens with JWT for secure user sessions.

CHAPTER 2

LITERATURE REVIEW

The concept of professional networking platforms, such as LinkedIn, has been extensively explored in research and practice. Studies have highlighted the critical role such platforms play in fostering mentorship, collaboration, and knowledge sharing among individuals from diverse professional backgrounds. MyMate, inspired by these platforms, introduces a unique focus on bridging the gap between juniors and seniors, emphasizing peer-to-peer learning and experience sharing in a community-driven environment.

Related Work and Existing Solutions

LinkedIn and Other Professional Networks:

LinkedIn remains the leading platform for professional networking, offering features like job postings, skill endorsements, and group discussions. However, its primary focus on corporate environments often overlooks the specific needs of students and early-career professionals seeking mentorship and guidance. MyMate addresses this gap by providing a tailored platform for these users.

Social Media Platforms (e.g., Facebook, Instagram):

While social media platforms support networking and content sharing, they are not designed for professional interactions. The lack of structured mentorship mechanisms and professional focus differentiates MyMate, which includes targeted features like resource sharing, experience posts, and following systems to facilitate career growth.

Slack and Discord Communities:

Real-time communication tools like Slack and Discord support collaboration and networking within specific communities. However, these platforms are often limited to isolated groups and lack public visibility for posts, making them less effective for broader networking and showcasing personal achievements. MyMate combines real-time chat with a public, searchable post system for maximum reach and impact.

Key Technologies in Similar Platforms

Authentication and Security:

Platforms like LinkedIn and Facebook employ robust authentication systems, often integrating OAuth for social logins. However, MyMate opts for traditional email-based login, secured with JWT and bcrypt, to ensure simplicity and data security.

Scalable Databases:

MongoDB, widely used in modern platforms, offers flexibility and scalability, making it an ideal choice for handling diverse data models like users, posts, and chats. MyMate leverages MongoDB's schema design capabilities to optimize performance and support features like user interaction tracking.

User Interaction Design:

Effective UI/UX is central to user retention in networking platforms. Research shows that intuitive layouts, responsive designs, and accessibility features significantly improve user engagement. MyMate incorporates these principles with its clean interface, easy navigation, and responsive components for posts, chats, and profiles.

Gaps Addressed by MyMate

Existing solutions cater to professional networking or general social interactions but rarely focus on mentorship and peer-driven learning for juniors and seniors.

Experience Sharing:

Allowing users to post and discuss personal experiences, fostering knowledge exchange.

Mentorship-Oriented Design:

A follow system designed for networking with peers and mentors.

Community-Driven Features:

Features like likes, comments, and chat for collaborative interactions.

CHAPTER 3

FEASIBILITY STUDY

A feasibility study is a detailed analysis that considers all of the critical aspects of a proposedproject in order to determine the likelihood of it succeeding.
Success in business may be defined primarily by return on return on investment, meaning that the project will generate enough profit to justify the investment. However, many other importantfactors may be identified on the plus or minus side, such as community reaction and environmental impact.

A feasibility study is an important step in any project, including an emotion detection project. It helps to determine the technical, economic, operational, and legal feasibility of the project. Here are some key aspects to consider in a feasibility study for an emotion detection project.

Based on the results of the feasibility study, the project team can make informed decisions about the viability and scope of the emotion detection project. If the feasibility study indicates that the project is viable and has potential benefits, the team can proceed with the project planning and implementation. If the study indicates that the project is not feasible or has significant risks and limitations, the team can consider alternative approaches or abandon the project altogether.

Before starting the project, feasibility study is carried out to measure the viable of the system. Feasibility is necessary to determine is creating a new or improved system is friendly with the cost, benefits, operation, technology and time.

Feasibility studies are important for a communications service provider to determine whether your broadband project will succeed or not. It should be the first action taken when wanting to begin a new project. It is one, if not the most important factor in determining whether the project can and should move forward. Also, if you are applying for broadband loans and grants, a feasibility study is normally required.

The feasibility study evaluates the technical, economic, and operational aspects of the proposed MyMate project to determine its viability. This analysis ensures the project aligns with the goals of bridging the gap between juniors and seniors through mentorship and community-building features while addressing the required resources, cost-effectiveness, and operational challenges.

3.1 Technical Feasibility

The technical feasibility of the MyMate project assesses the availability of resources, technology compatibility, scalability, and security considerations to ensure smooth development and deployment.

Resource Availability:

Availability of skilled developers proficient in the MERN stack (MongoDB, Express.js, React.js, Node.js).

Infrastructure readiness for hosting servers, cloud services, and databases.

Access to devices and browsers for testing and debugging across platforms.

Technology Requirements:

Tech Stack: MongoDB for the database, Express.js for backend, React.js for the frontend, and Node.js for server-side scripting.

APIs and services for features like real-time chat and notifications (e.g., Socket.IO, Twilio).

Secure communication protocols (HTTPS) for user data and interactions.

Compatibility:

Ensure responsiveness and compatibility across browsers (Chrome, Firefox, Safari) and devices (mobiles, tablets, desktops).

Testing compliance with modern web standards and accessibility guidelines.

Scalability and Performance:

Design a scalable architecture to support a growing user base and increasing data load.

Optimize backend queries and API interactions for high-speed performance.

Conduct load and stress testing to simulate concurrent user activity.

Security:

Implement robust JWT-based authentication and bcrypt for password hashing.

Encryption for sensitive user data and secure data storage practices.

Vulnerability assessments to prevent unauthorized access or data breaches.

Testing and Quality Assurance:

Rigorous testing across modules (registration, login, posts, chat, etc.) for functionality, usability, and compatibility.

Conduct penetration testing and security audits to ensure the platform's robustness against attacks.

Documentation and Training:

Prepare technical documentation for system architecture and deployment processes.

Provide user manuals and support guides for smooth onboarding of new users.

3.2 Economic Feasibility

Economic feasibility evaluates the financial viability of the project by analyzing costs against expected benefits and returns.

Cost-Benefit Analysis:

Initial investment for development, including salaries for developers and costs of hosting services (e.g., AWS, Azure).

Benefits include long-term growth in user base, monetization through ads, premium memberships, and sponsored content.

Return on Investment (ROI):

Expected ROI based on projected user subscriptions and premium feature adoption.

Anticipate revenue from partnerships with academic institutions or career services.

Market Demand:

Analyze the demand for mentorship platforms targeting students and early-career professionals.

Examine competitor platforms to identify revenue opportunities and market gaps.

Scalability:

Ensure scalability in terms of infrastructure costs for future user growth without excessive financial burden.

Cost Reduction:

Streamlining development with open-source tools and frameworks.

Efficient resource allocation and usage of third-party APIs for cost savings.

Revenue Generation:

Explore diverse revenue streams, including advertisements, premium features, and affiliate partnerships.

Risk Assessment:

Assess potential risks, such as insufficient user adoption or operational costs exceeding revenue, and develop mitigation strategies.

3.3 Operational Feasibility

Operational feasibility focuses on the practicality of implementing the system and its alignment with organizational goals.

Alignment with Objectives:

The MyMate platform directly supports the goal of fostering mentorship, collaboration, and career guidance, making it a strategic initiative.

Compatibility with Processes:

Integration of user-friendly interfaces and workflows to ensure seamless operation for users of all technical expertise levels.

Resource Availability:

Availability of technical and non-technical staff for development, testing, and post-launch maintenance.

Skills and Training Requirements:

Minimal training needed for end users due to the intuitive design and self-explanatory functionalities.

Technical training for the development team to handle advanced features like real-time chat and analytics.

Acceptance by Stakeholders:

Engage users, mentors, and sponsors during the design phase to ensure alignment with expectations.

Offer demo sessions and early access to garner support and feedback.

Risk Assessment:

Address potential resistance to adoption by incorporating user-friendly tutorials and live support.

Ensure compliance with legal and privacy regulations (e.g., GDPR for data protection).

Scalability and Flexibility:

Design the system to scale efficiently with increasing user registrations and content contributions.

Incorporate flexibility to add new features and adjust to changing user demands.

Impact on Operations:

Improve communication and collaboration within the user community.

Reduce the reliance on external platforms by providing integrated mentorship and networking features.

CHAPTER 4

E-R DIAGRAM/FLOWCHART

E-R DIAGRAM

The User model defines a schema for users in the application with fields such as name, email, password, bio, and profilePic, along with support for OAuth authentication through oauthProvider and oauthId. It includes relationships with other entities such as posts, likedPosts, savedPosts, followers, and following, enabling features like user interactions and post engagement. The schema also incorporates arrays for education, projects, and weblinks, allowing users to showcase their profiles. Role-based access is handled via the role field, which defaults to "user" but can be expanded to include "admin." Additionally, password reset functionality is supported with resetPasswordToken and resetPasswordExpire. Timestamps (createdAt and updatedAt) and virtual fields (followerCount and followingCount) enhance data tracking and user insights.

The Post model represents user-created posts, with attributes like content, media, and relationships to author, likes, comments, and shares. Users can like, share, comment on, and save posts. Comments include nested details such as the commenting user and text, and both schemas support timestamps for easy activity tracking. Together, the User and Post models form the core structure for user interaction and content sharing within the application.

Entities:

Attributes for Each Entity:

1. **User**
    - name
    - email
    - password
    - oauthProvider
    - oauthId
    - bio
    - profilePic
    - profilePicPublicId
    - notes
    - weblinks
    - education (Relationship with Education)
    - projects (Relationship with Project)
    - followers (Relationship with other Users)

- following (Relationship with other Users)
- posts (Relationship with Posts)
- likedPosts (Relationship with Posts)
- savedPosts (Relationship with Posts)
- role
- resetPasswordToken
- resetPasswordExpire
- Timestamps: createdAt, updatedAt

2. **Post**
    - content
    - media
    - mediaPublicId
    - author (Relationship with User)
    - likes (Relationship with Users)
    - shares (Relationship with Users)
    - comments (Relationship with Comment Entity)
    - savedBy (Relationship with Users)
    - Timestamps: createdAt, updatedAt

3. **Comment (Nested inside Post)**
    - user (Relationship with User)
    - text
    - createdAt

4. **Education**
    - Attributes can include degree, institution, year, etc. (Not fully provided but referenced in User schema)

5. **Project**
    - Attributes can include title, description, link, etc. (Not fully provided but referenced in User schema)

Relationships:

- **User to Post**: One-to-Many (A user can create multiple posts)
- **User to User**: Many-to-Many (Followers and Following)
- **User to Comment**: One-to-Many (A user can create multiple comments)
- **Post to Comment**: One-to-Many (A post can have multiple comments)
- **User to Education/Project**: One-to-Many

4.1 E-R Diagram

```
Table users {
  id integer [primary key] // Unique identifier for the user
  name varchar [note: 'Full name of the user']
  email varchar [unique, note: 'Email address of the user']
  password varchar [note: 'Hashed password']
  bio text [note: 'User bio']
  profile_pic varchar [note: 'Profile picture URL']
  role varchar [note: 'User role, e.g., user/admin']
  created_at timestamp [note: 'Timestamp of user creation']
  updated_at timestamp [note: 'Timestamp of last update']
}

Table posts {
  id integer [primary key] // Unique identifier for the post
  content text [note: 'Content of the post']
  user_id integer [note: 'ID of the user who created the post']
  created_at timestamp [note: 'Timestamp of post creation']
  updated_at timestamp [note: 'Timestamp of last update']
}

Table comments {
  id integer [primary key] // Unique identifier for the comment
  user_id integer [note: 'ID of the user who wrote the comment']
  post_id integer [note: 'ID of the post the comment is associated with']
  text text [note: 'Comment text']
  created_at timestamp [note: 'Timestamp of comment creation']
}

Table likes {
  id integer [primary key] // Unique identifier for the like
  user_id integer [note: 'ID of the user who liked the post']
  post_id integer [note: 'ID of the post that was liked']
  created_at timestamp [note: 'Timestamp of the like']
}

Table follows {
  id integer [primary key] // Unique identifier for the follow record
  following_user_id integer [note: 'ID of the user who is following']
  followed_user_id integer [note: 'ID of the user being followed']
  created_at timestamp [note: 'Timestamp of the follow action']
}

Ref: posts.user_id > users.id // A post is created by a user (many-to-one)
Ref: comments.user_id > users.id // A comment is written by a user (many-to-one)
Ref: comments.post_id > posts.id // A comment is associated with a post (many-to-one)
Ref: likes.user_id > users.id // A like is given by a user (many-to-one)
Ref: likes.post_id > posts.id // A like is associated with a post (many-to-one)
Ref: follows.following_user_id > users.id // A user follows another user (self-
```

referencing)
Ref: follows.followed_user_id > users.id // A user is followed by another user (self-referencing)

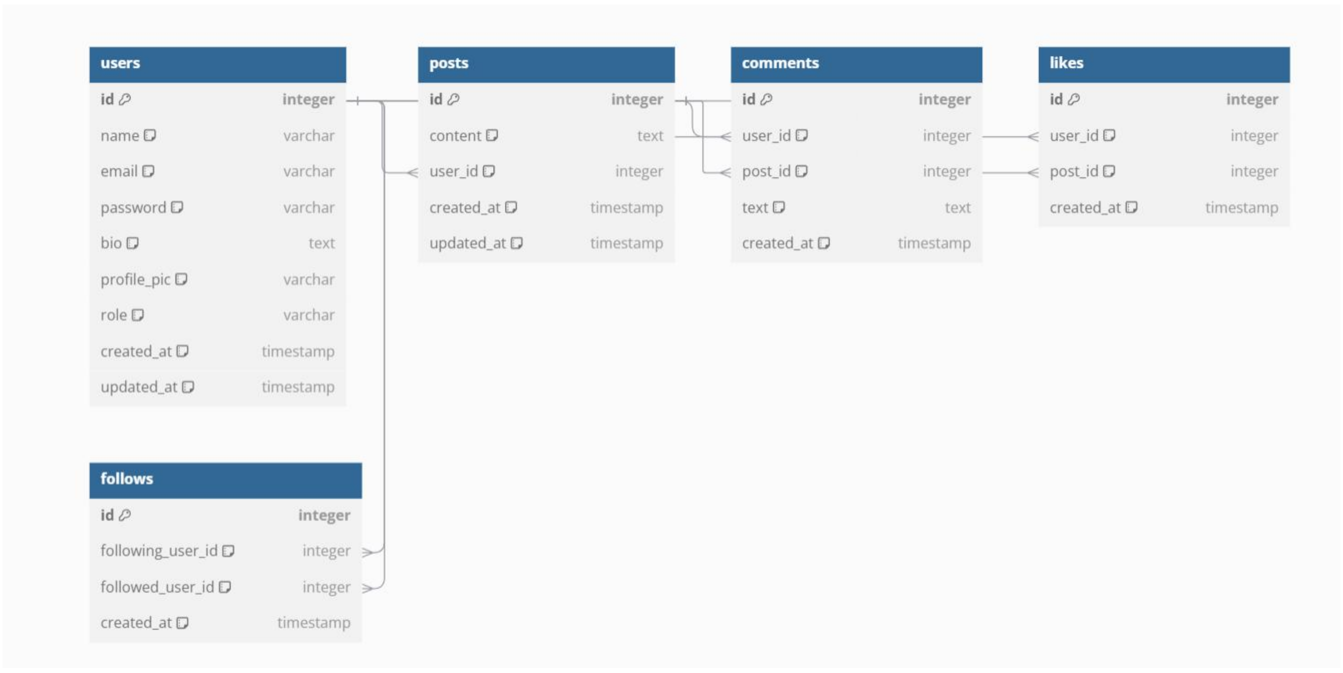


Fig 4.1 Diagram for E-R Diagram

FLOWCHARTS

Flowcharts are graphical representations of processes or systems, consisting of various symbols connected by arrows to illustrate the flow of steps or actions.
They are visual tools that represent the steps in a process or system using symbols such as ovals, rectangles, diamonds, and arrows. They are designed to simplify complex processes, making them easier to understand and communicate.
Here's more content about flowcharts:
Visual Representation: Flowcharts are diagrams that visually represent processes or workflows using symbols and arrows.

Symbolic Language: They use standardized symbols like rectangles for processes, diamonds for decisions, and arrows for flow direction.

Process Mapping: Flowcharts map out the steps of a process from start to finish, showing the sequence of actions.

Decision Points: They highlight decision points where the flow can take different paths based on conditions.

Easy Understanding: Flowcharts simplify complex processes, making them easier to understand and follow.

Problem-Solving Tool: They help identify bottlenecks, redundancies, and inefficiencies in a process, aiding in problem-solving and optimization.

Applications: Used in various fields including business, software development, project management, education, and engineering.

Standardization: Flowcharts provide a standardized way to document and analyze processes, ensuring clarity and consistency.

Software Tools: There are many software tools available for creating flowcharts, offering templates and drag-and-drop functionality.

Communication: Flowcharts facilitate communication by providing a visual representation of processes, making it easier to convey information to others.

User Authentication

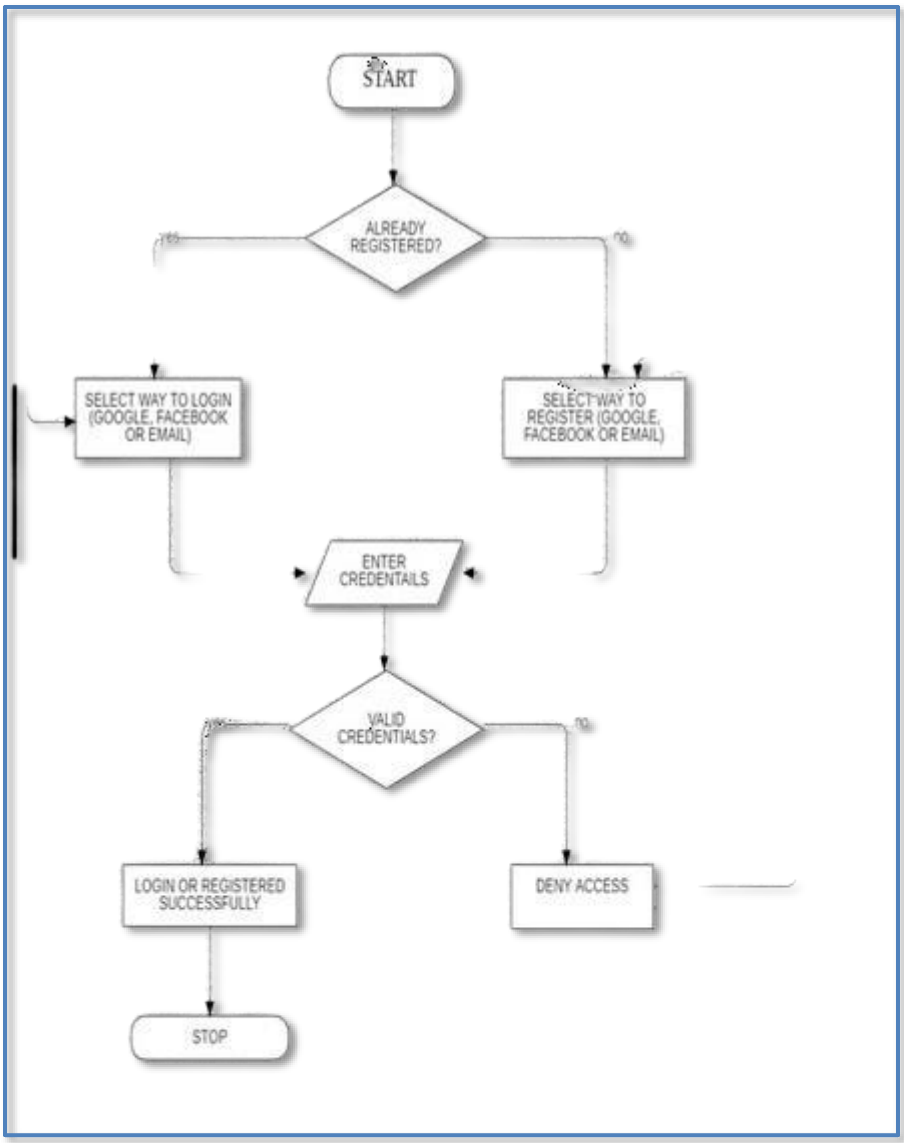

START
ALREADY REGISTERED?
yes
no
SELECT WAY TO LOGIN (GOOGLE, FACEBOOK OR EMAIL)
SELECT WAY TO REGISTER (GOOGLE, FACEBOOK OR EMAIL)
ENTER CREDENTAILS
VALID CREDENTIALS?
yes
no
LOGIN OR REGISTERED SUCCESSFULLY
DENY ACCESS
STOP

Fig 4.2 : Flowchart of User Authentication Module

Content Creation

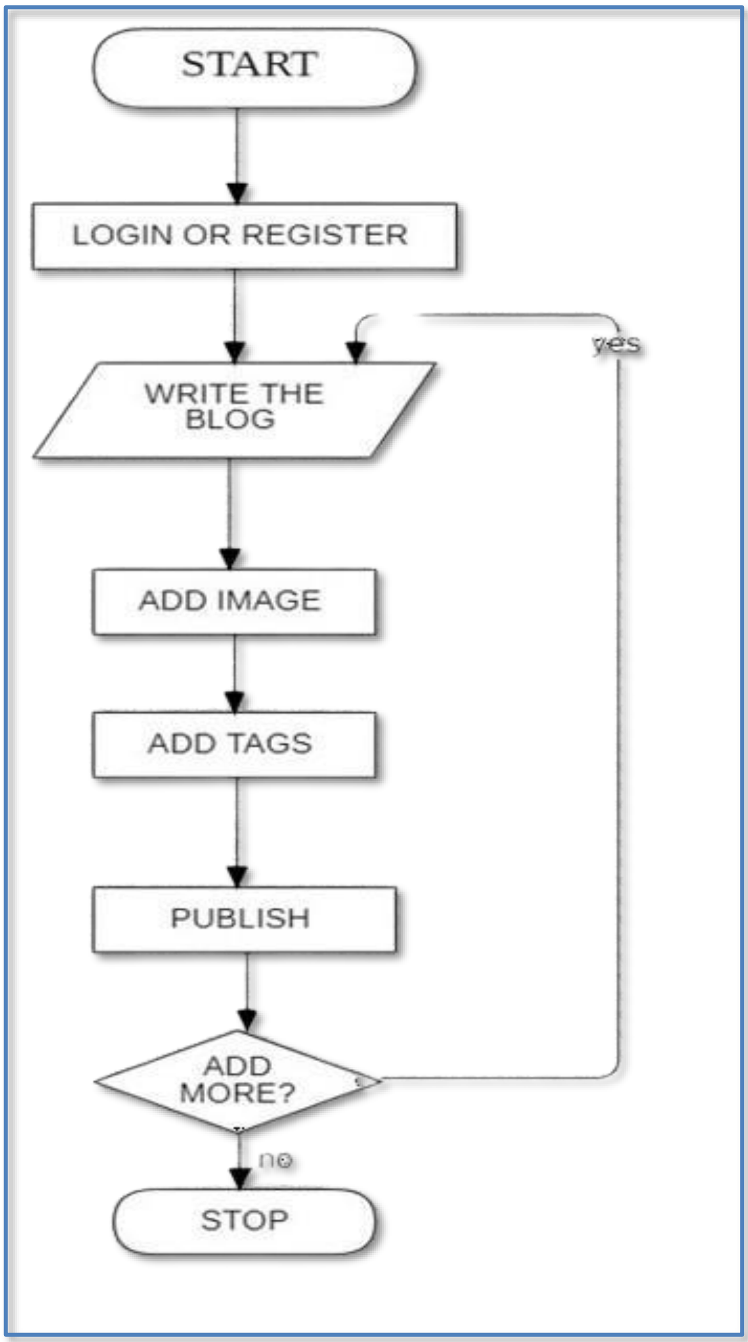


Fig 4.3: Flowchart of Content Creation Module

CHAPTER 5

TECHNOLOGY USED AND SETUP

HARDWARE REQUIREMENTS

Hardware requirements for a project refer to the specific physical components or devices needed to support the project's objectives. These requirements can vary significantly depending on the nature of the project. These requirements include computing power, storage, network connectivity, memory etc.

The hardware requirements for accessing “Mymate” is enlisted in the table below:

Table 5.1: Hardware Requirements

S. No.	Description of system requirement
1	5 GB or more Hard disk.
2	8 GB RAM.
3	Core i5 7th gen or above processor.
4	50 Mbps or more internet connectivity

GB or more Hard disk:

This specifies the storage requirement for the PC. It should have a hard disk with a capacity of
5 gigabytes (GB) or more. This is where you store your operating system, software applications, and data. A PC with a 5 GB or larger hard disk provides ample storage for

the operating system, software applications, and user data. This ensures smooth performance, accommodates growing storage needs, and allows for data backups and future expansion.

8 GB RAM:

This sets the minimum random-access memory (RAM) requirement for the PC. It should have at least 8 gigabytes of RAM. RAM is essential for running applications and the operating system efficiently. Having 8 GB of RAM ensures smooth performance for running multiple applications simultaneously, faster operation of the operating system, seamless multitasking, and readiness for resource-intensive tasks like gaming or video editing.

Core i5 7th gen or above processor:

This specifies the processor requirement for the PC. It should have an Intel Core i3 processor or a more powerful one. The processor is a crucial component that determines the computer's overall speed and performance. A Core i5 7th gen or higher processor ensures strong performance and responsiveness for the PC. This Intel processor provides ample computing power for everyday tasks, multitasking, and even some demanding applications like photo or video editing. It offers a balance of speed, efficiency, and reliability, making it suitable for most users' needs.

50 Mbps or more internet connectivity:

A 50 Mbps or higher internet connection ensures fast and reliable access to online resources, data, and collaborative tools required for the project. With this speed, users can quickly download and upload files, stream multimedia content, participate in video conferences, and collaborate with team members in real-time. It provides a seamless online experience, reducing wait times and enhancing productivity, particularly for projects that rely heavily on cloud-based services, remote collaboration, or data-intensive tasks.

SOFTWARE REQUIREMENTS

Software requirements for a project outline the specific programs, platforms, and functionalities needed to achieve project goals. They detail essential software components such as operating systems, development tools, databases, and any specialized software necessary for project execution. These requirements serve as a roadmap for software selection, development, integration, and testing throughout the project lifecycle.

The software requirements of “Collablog” are enlisted in the following software requirements table:

Table 5.2 Software Requirements

S. No.	Description	Type
1	Operating System	Windows 10 or 11
2	Front End	Vite, React JS, JS, Tailwind CSS
3	Back End	Node JS
4	Database and Storage	MongoDB
5	IDE	VS Code
6	Browser	Chrome, Firefox, Edge

Operating System

The specified operating system requirement for the project development environment is either Windows 10 or 11, with the option to use a newer version if available. This ensures compatibility with the latest software development tools, libraries, and frameworks required for the project. Additionally, it provides a consistent and stable platform for developers to create, test, and deploy the project's software components. By standardizing the operating system environment, it facilitates collaboration among team members and simplifies software configuration management, version control, and troubleshooting processes throughout the project lifecycle.

Front End

The frontend of a project is what users interact with directly, including the interface, design, and user experience. The goal is to create an intuitive, visually appealing, and responsive interface that guides users seamlessly through the application.

React JS

React.js is a JavaScript library for building user interfaces. It simplifies UI development by breaking it down into reusable components and managing updates efficiently with a virtual DOM. Developers use JSX to write UI components, enabling a declarative approach to defining UIs based on application state

Tailwind CSS

Tailwind CSS is a utility-first CSS framework that streamlines frontend development by providing a set of pre-defined utility classes for styling HTML elements. It prioritizes simplicity, responsiveness, and customization, making it easy for developers to create modern and responsive user interfaces efficiently.

Vite

Vite is a fast build tool for modern web development, specifically designed to optimize the development experience for frontend projects. It leverages native ES modules in modern browsers to deliver instant server startup and rapid hot module replacement (HMR). With Vite, developers can build and serve frontend applications quickly, enhancing productivity and facilitating a smooth development workflow.

Back End

The backend of a project comprises server-side code, databases, and APIs that handle data processing, business logic, authentication, security, and communication with clients. It powers the functionality of the application, manages data storage, and ensures that users can interact with the system securely and efficiently.

Node JS

Node.js is a runtime environment that allows you to use JavaScript for server-side development. It efficiently handles many connections simultaneously due to its eventdriven and asynchronous nature. This makes it ideal for building scalable applications like real-time chat apps, APIs, and microservices. Using the same language for both frontend and backend simplifies development, and the npm ecosystem provides a vast array of modules to speed up the development process. Node.js is built on the fast V8 JavaScript engine, ensuring high performance.

Database and Storage

The database in a project provides essential data management capabilities, allowing for the storage, retrieval, and organization of data. Integrating a database involves installing the necessary drivers, connecting to the database, defining schemas and models, and using these models in the application logic.

MongoDB

MongoDB is a NoSQL, document-oriented database designed for managing large volumes of data across distributed systems. It features a flexible, schema-less data model that allows data to be stored in JSON-like documents with dynamic schemas. This flexibility enables developers to store complex data structures and adapt quickly to changing requirements without needing to define a rigid schema upfront. MongoDB supports horizontal scaling through sharding, distributing data across multiple servers, and is optimized for high- performance read and write operations, making it suitable for high-throughput applications. Its rich query language supports ad hoc queries, indexing, and real-time aggregation, while replication ensures high availability and redundancy through replica sets.

IDE

An Integrated Development Environment (IDE) is a software tool that combines various features to facilitate programming tasks. It typically includes a code editor, compiler/interpreter, debugger, build automation tools, version control integration, and project management capabilities. IDEs increase developer productivity by providing a centralized environment for coding, debugging, and managing projects, ultimately leading to more efficient software development.

Visual Studio Code

Visual Studio Code (VS Code) is a highly popular and versatile integrated development environment (IDE) developed by Microsoft. It's favoured by developers across various platforms and programming languages due to its extensive features and flexibility. Visual Studio Code (VS Code) is the preferred integrated development environment for coding.

Browser

Accessing a project via a web browser allows users to interact with web-based applications or websites. It provides accessibility, cross-platform compatibility, user-friendly interfaces, security features, scalability, and easy updates, making it a crucial aspect of modern computing. Mozilla Firefox, Google Chrome, Microsoft Edge, any of the browsers can be used to access the software.

INSTALLATION OF SOFTWARE REQUIREMENTS

Visual Studio Code (VS Code)

To install Visual Studio Code (VS Code):
Download: Go to the official VS Code website (https://code.visualstudio.com) and download the installer for your operating system (Windows, macOS, or Linux).
Run Installer: Once the download is complete, run the installer executable file. After the installation completes, launch Visual Studio Code from your system's applications menu or desktop shortcut.

Node JS

To install Node.js:

Download: Visit the official Node.js website (https://nodejs.org) and download the appropriate installer for your operating system (Windows, macOS, or Linux).

Run Installer: Open the downloaded installer file. Finish the installation process. The installer will automatically add Node.js and npm (Node Package Manager) to your system PATH.

Verify Installation: Open a terminal or command prompt and run the following commands to verify the installation:

```
node -v
npm -v
```

Vite

To install Vite, execute the following lines on the terminal

```
vite build
npm create vite@latest .
```

React JS

To install React JS, execute the following lines on the terminal

```
npm create vite@latest my-project   template react
cd my-project
```

To install various react components used in the project

```
npm install moment
npm install react-quill
npm install react-tagsinput
npm install react-share

npm install react-router-dom react-icons
npm install --save react-tag-input
```

Tailwind CSS

To install Tailwind CSS, execute the following lines on the terminal

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

MongoDB Download:

Go to the MongoDB Download Center and download the latest version of MongoDB for Windows.

Install:

Run the downloaded .msi installer.
Follow the setup instructions. Make sure to select "Complete" setup. During the setup, check "Install MongoDB as a Service".
Set Up Environment:
After installation, add the MongoDB bin directory (e.g., C:\Program Files\MongoDB\Server\{version}\bin) to the PATH environment variable.

Run MongoDB:

Open Command Prompt and type mongo to start the MongoDB shell.

CHAPTER 6
TESTING

Unit Testing :

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but it could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist in testing a module in isolation. Unit tests are typically written and run by software developers to ensure that codemeets its design and behaves as intended.

Benefits of Unit Testing

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it offers several benefits.

Find Problems Early:

Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build.

The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

Facilitates Change:

Unit testing allows the programmer to refactor code or upgrade system libraries later, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes that may break a design contract.

Documentation:

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in the development unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify the functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, with success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. Some different types of integration testing are big-bang, top- down, and bottom-up, mixed (sandwich) and risky- hardest. Other Integration Patterns are collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing because it expects to have few problems with the individual components.

The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.
To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

Top-Down and Bottom-Up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready.

This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing
where the top integrated modules are tested, and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top-down testing with bottom-up testing.

Black-Box Testing

Black box testing is a technique used to test the functionality of a software application without having knowledge of its internal structure or implementation details. It focuses on the inputs and outputs of the system and verifies if the expected outputs match the desired results. Here are some examples of black box testing techniques that can be applied to the KIET Event Management App:

Equivalence Partitioning:

Identify different categories of inputs for the app, such as valid and invalid inputs, and divide them into equivalence classes.
Test representative values from each equivalence class to ensure the app behaves consistently within each class.

Boundary Value Analysis:

Identify the boundaries or limits for inputs in the app, such as minimum and maximum values, and test values at those boundaries.
Test values just above and below the boundaries to verify the app's behaviour at critical points.

Decision Table Testing:

Identify the different conditions and rules that govern the behaviour of the app.

Create a decision table with combinations of conditions and corresponding expected results.

Test different combinations of conditions to validate the app's decision-making process.

State Transition Testing:

Identify the different states that the app can transition between.

Define the valid and invalid transitions between states.

Test different sequences of state transitions to verify the app's behaviour.

Error Guessing:

Use experience and intuition to guess potential errors or issues in the app.

Create test cases based on those guesses to verify if the app handles the errors correctly.

Compatibility Testing:

Test the app on different platforms, browsers, or devices to ensure compatibility.

Verify that the app functions correctly and displays appropriately across different environments.

Usability Testing:

Evaluate the app's user interface and interactions from the perspective of an end-user.

Test common user scenarios and assess the app's ease of use, intuitiveness, and overall user experience.

Security Testing:

Test the app for potential security vulnerabilities or weaknesses.

Verify if the app handles user authentication, data encryption, and access control appropriately.

Performance Testing:

Test the app's performance under different load conditions, such as a high number of concurrent users or large data sets.

Verify if the app responds within acceptable time limits and performs efficiently.

During black box testing, test cases are designed based on the app's specifications, requirements, and user expectations. The focus is on validating the functionality, user interactions, and expected outputs without considering the internal implementation details of the app.

White-Box Testing

White box testing, also known as structural testing or glass box testing, is a software testing technique that examines the internal structure and implementation details of the application. It aims to ensure that the code functions as intended and covers all possible execution paths. Here are some examples of white box testing techniques that can be applied to the KIET Event Management App:

Unit Testing:

Test individual units or components of the app, such as functions or methods, to verify their correctness.

Use techniques like code coverage analysis (e.g., statement coverage, branch coverage) to ensure that all code paths are exercised.

Integration Testing:

Test the interaction between different components or modules of the app to ensure they work together seamlessly.

Verify the flow of data and control between the modules and check for any integration issues or errors.

Path Testing:

Identify and test different paths or execution flows through the app, including both positive and negative scenarios.

Execute test cases that cover all possible paths within the code to ensure complete coverage.

Decision Coverage:

Ensure that every decision point in the code (e.g., if statements, switch cases) is tested for both true and false conditions.

Validate that the app makes the correct decisions based on the specified conditions.

Code Review:

Analyse the code and its structure to identify any potential issues or vulnerabilities.
Review the adherence to coding standards, best practices, and potential optimizations.
Performance Testing:
Assess the app's performance from a code perspective, such as identifying any bottlenecks or inefficient algorithms.
Measure the execution time of critical code sections and evaluate resource usage.

Security Testing:
Review the code for potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), or authentication weaknesses.
Verify the implementation of secure coding practices, data encryption, and access control mechanisms.

Error Handling Testing:
Test how the app handles and recovers from unexpected errors or exceptions.
Validate that error messages are clear, meaningful, and do not expose sensitive information.

Code Coverage Analysis:
Use tools to measure the code coverage achieved by the tests, such as statement coverage, branch coverage, or path coverage.
Aim for high code coverage to ensure that all parts of the code are exercised.
During white box testing, the tester has access to the application's internal code, allowing for amore detailed examination of its behavior.

System Testing :
System testing is a level of software testing that evaluates the complete system as a whole, rather than focusing on individual components or modules. It ensures that all components of the KIET Event Management App work together seamlessly and meet the specified requirements. Here are some examples of system testing techniques that can be applied to the app:

Functional Testing:

Verify that all functional requirements of the app are met.

Test various functionalities such as event creation, registration, club directory search, user log in and registration, event notifications, etc.

Validate that the app behaves as expected and produces the correct outputs based on different inputs.

User Interface Testing:

Test the graphical user interface (GUI) of the app for usability, consistency, and responsiveness.

Check the layout, navigation, buttons, forms, and other UI elements to ensure they are visually appealing and intuitive.

Validate that the app adheres to the design guidelines and provides a seamless user experience.

Performance Testing:

Evaluate the performance of the app under different load conditions.

Measure response times, throughput, and resource utilization to ensure the app can handle the expected user load without significant degradation.

Identify and address any performance bottlenecks or scalability issues.

Compatibility Testing:

Test the app on different devices, platforms, and browsers to ensure compatibility.

Verify that the app works correctly on various operating systems (e.g., iOS, Android) and different screen sizes.

Validate that the app functions properly on different web browsers (if applicable).

Security Testing:

Assess the app's security measures to protect user data and prevent unauthorised access.

Perform vulnerability scanning, penetration testing, and authentication testing to identify and address any security vulnerabilities.

Test the app's resilience against common security threats, such as cross-site scripting (XSS) and SQL injection.

Integration Testing:

Test the integration of the app with external systems, such as databases, mapping services, or notification services.

Validate that data is exchanged correctly between the app and external systems.

Verify that the app's functionality remains intact when integrated with other systems.

Recovery Testing:

Simulate system failures or interruptions and evaluate the app's ability to recover and resume normal operation.

Test scenarios such as unexpected shutdowns, network failures, or interrupted database connections.

Ensure that the app can gracefully handle such situations and recover without data loss or integrity issues.

Regression Testing:

Re-test previously tested features and functionalities to ensure that recent changes or additions did not introduce new bugs or regressions.

Execute a set of comprehensive test cases to cover critical areas of the app and ensure that no existing functionality is compromised.

CHAPTER 7

RESULTS

7.1Landing Page

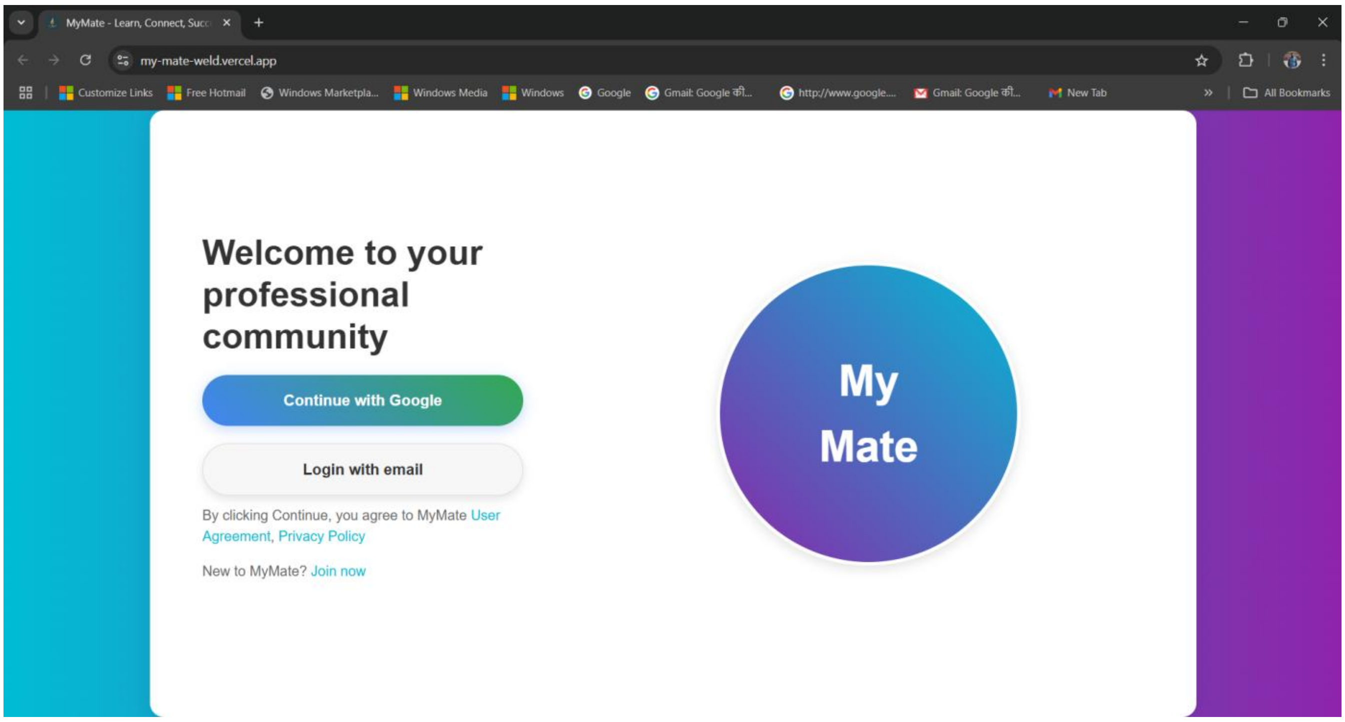

MyMate - Learn, Connect, Succ
my-mate-weld.vercel.app
Welcome to your professional community
Continue with Google
Login with email
By clicking Continue, you agree to MyMate User Agreement, Privacy Policy
New to MyMate? Join now
My Mate

Fig 7.1 Home page

The image shows a webpage from a blog site named "Mymate," which has the tagline "To come and collaborate" This page only bring the options to either login or sign up.

7.2Registration Page

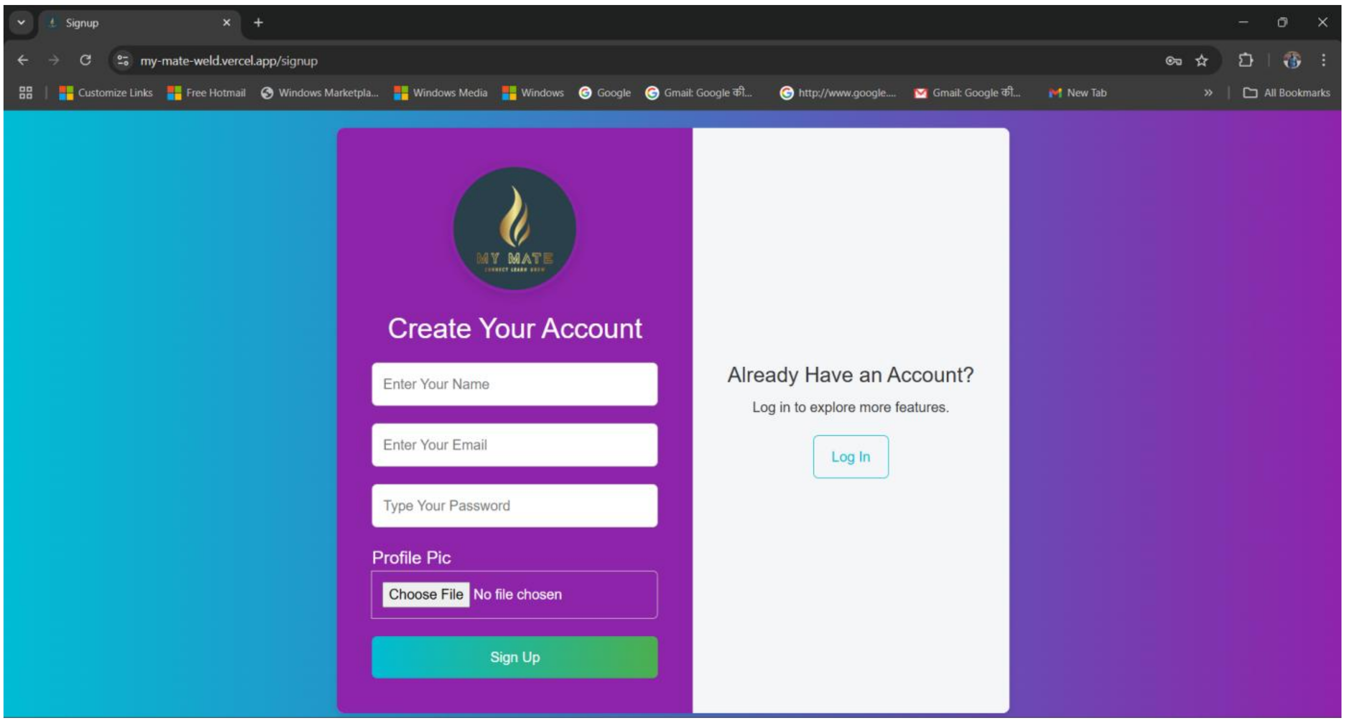


Fig 7.2 Registration page

7.3 Login Page

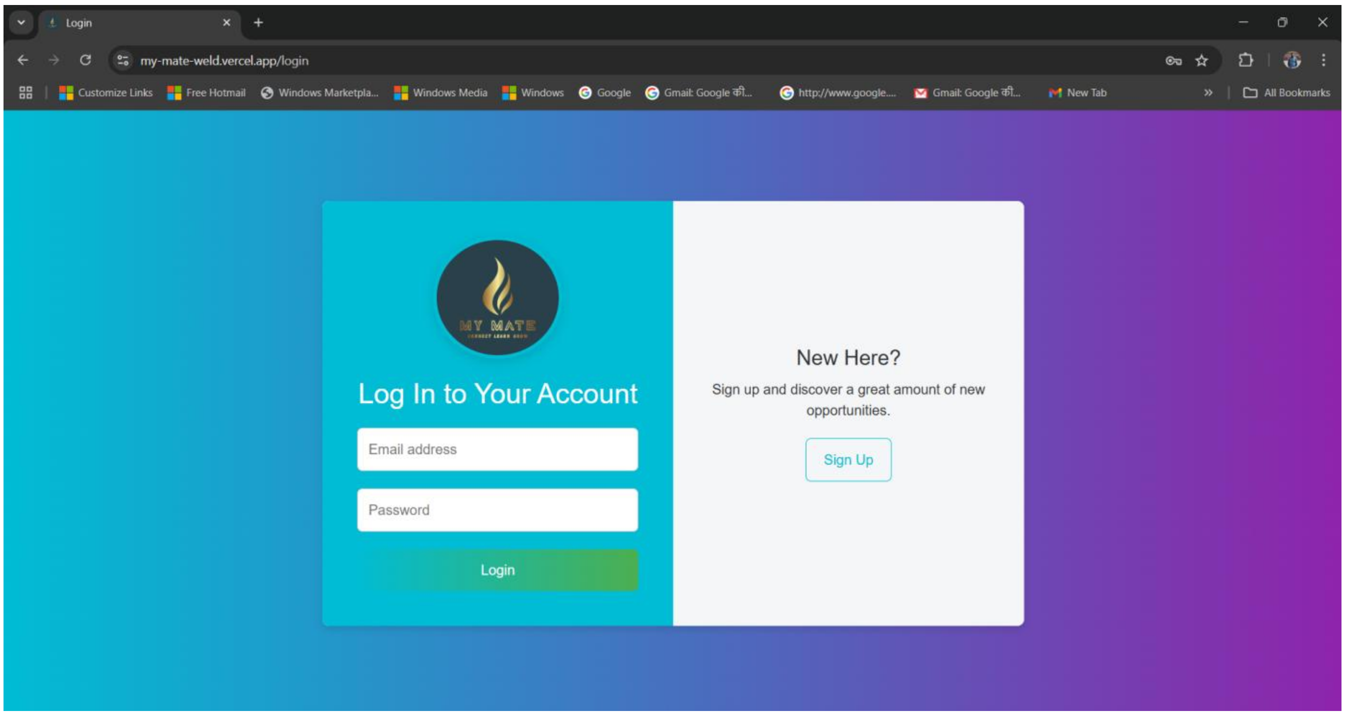

Login
my-mate-weld.vercel.app/login
MY MATE
Log In to Your Account
Email address
Password
Login
New Here?
Sign up and discover a great amount of new opportunities.
Sign Up

Fig 7.3 Login Page

7.4 Home Page

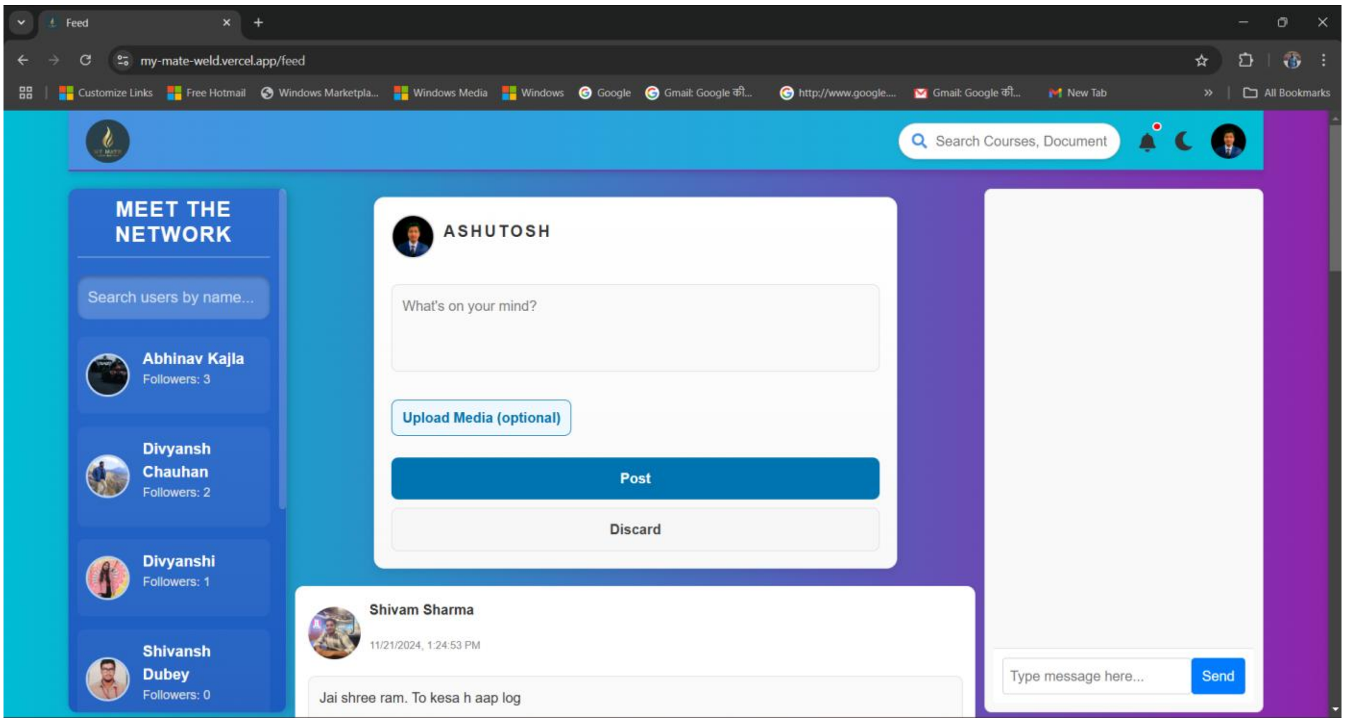


Fig 7.4 Create a post page

The homepage of the MyMate platform serves as a dynamic hub for user interactions and content sharing, designed to foster community engagement and streamline navigation. On the left, the "Meet the Network" sidebar provides a searchable list of users, displaying their follower count to encourage connections and interaction. At the center of the page is the Post Creation Area, allowing users to share their thoughts or updates effortlessly. This section features a text input field with a placeholder prompting users to express themselves, alongside options to upload media and buttons for posting or discarding drafts. Below this, the Feed Section displays user-generated posts in chronological order, complete with profile details, timestamps, and content, creating a cohesive and engaging space for updates.

7.5 Create Post

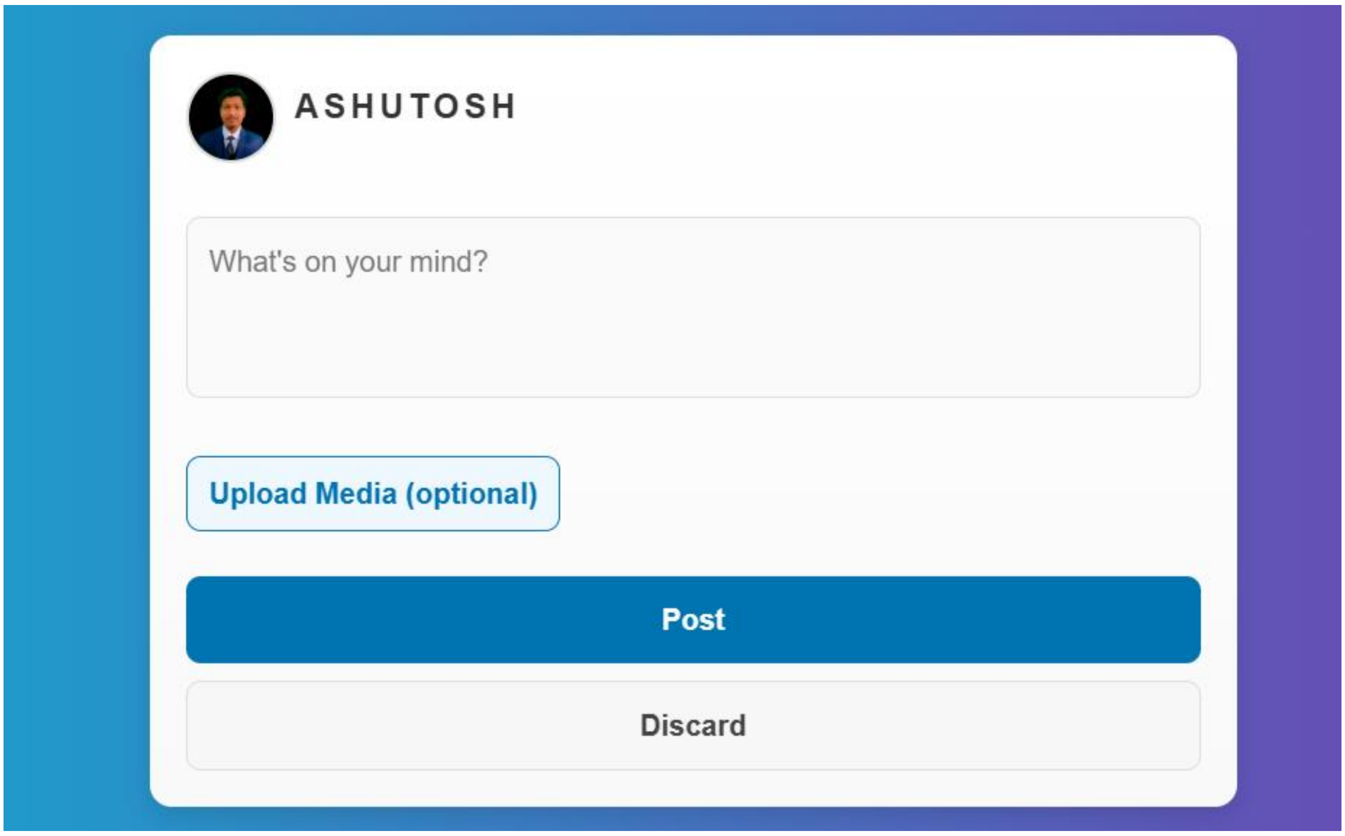
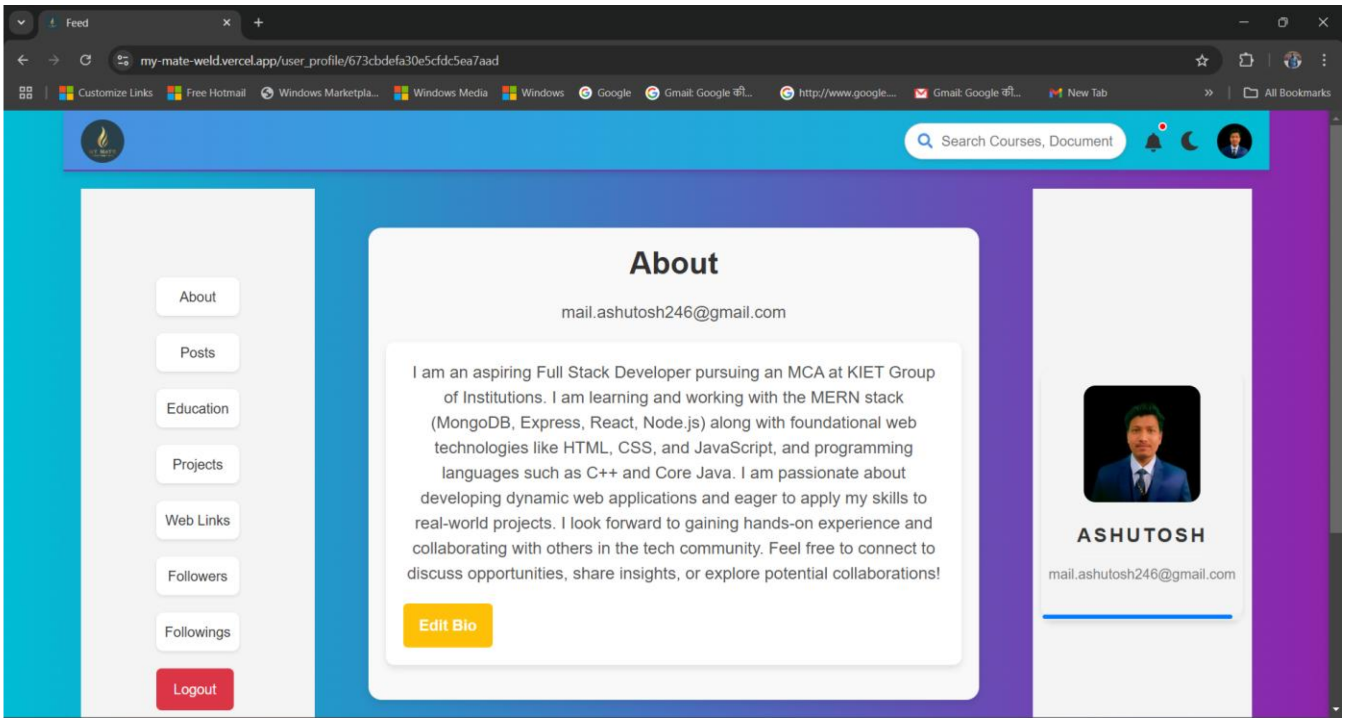


Fig 7.5 Create post

The Post Creation Component of the MyMate platform is a user-friendly interface designed to enable seamless sharing of thoughts and updates. At the top, it displays the user's profile picture and name, giving a personal touch to the posting process. Below this, a text input field with the placeholder "What's on your mind?" invites users to compose their posts, ensuring a straightforward and intuitive experience. Additionally, users have the option to upload media files, such as images or videos, enhancing the expressiveness of their posts. The component is equipped with two prominent buttons: a "Post" button to publish the content and a "Discard" button to cancel the post if needed. This simple yet functional design ensures users can share their content effortlessly while maintaining control over their inputs.

7.6 Profile Page



About

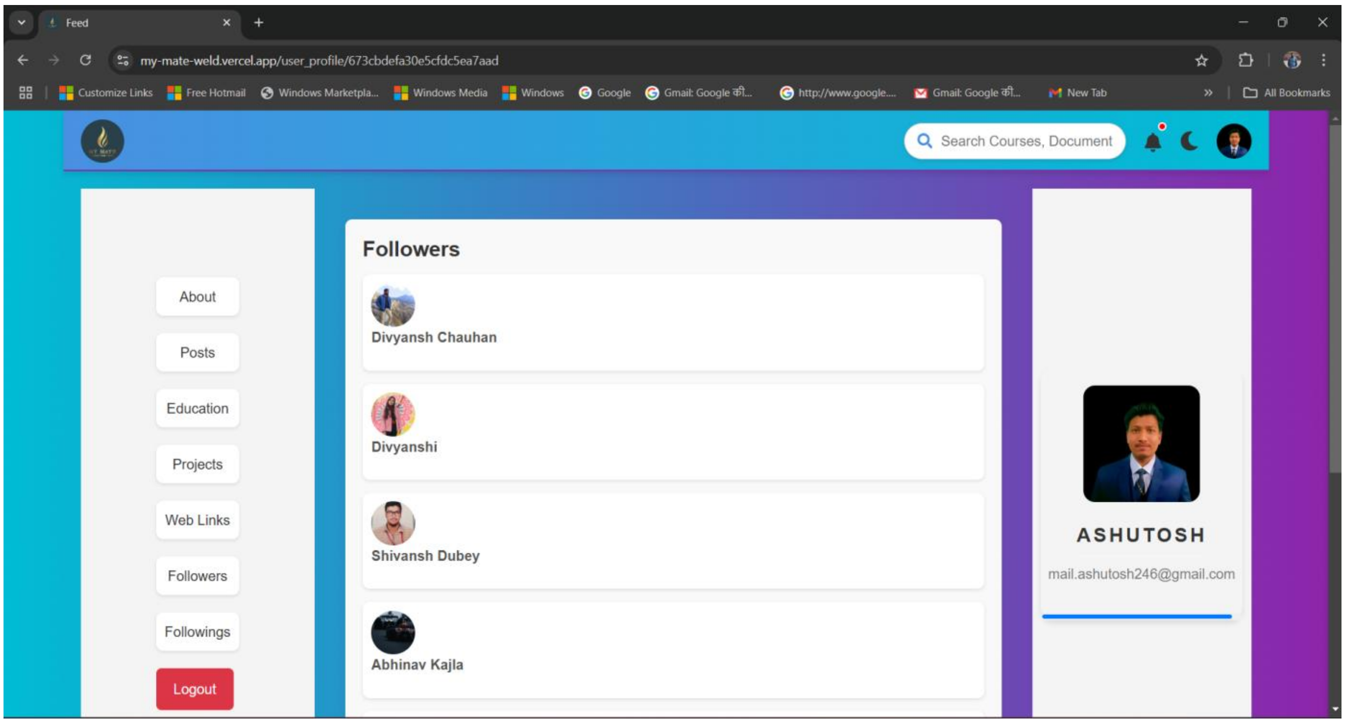
mail.ashutosh246@gmail.com

I am an aspiring Full Stack Developer pursuing an MCA at KIET Group of Institutions. I am learning and working with the MERN stack (MongoDB, Express, React, Node.js) along with foundational web technologies like HTML, CSS, and JavaScript, and programming languages such as C++ and Core Java. I am passionate about developing dynamic web applications and eager to apply my skills to real-world projects. I look forward to gaining hands-on experience and collaborating with others in the tech community. Feel free to connect to discuss opportunities, share insights, or explore potential collaborations!

Fig 7.6 Profile Page

The User Profile View Component provides an organized and comprehensive interface for showcasing a user's details and activities. At the center, the "About" section highlights the user's email and a detailed bio, offering insights into their professional background, skills, and aspirations. This section also includes an "Edit Bio" button, allowing users to update their personal information easily. To the left, a sidebar menu grants quick access to various profile sections, such as "Posts," "Education," "Projects," "Web Links," "Followers," and "Followings," ensuring seamless navigation. On the right, a compact card displays the user's profile picture and email, serving as a consistent identifier across the platform. The layout is clean and user-friendly, designed to help users manage and present their profiles effectively while encouraging connection and collaboration.

7.7 Followers Component



Followers

Divyansh Chauhan

Divyanshi

Shivansh Dubey

Abhinav Kajla

Fig 7.7 Follower component

In Update post page user can simply update the post by filling given details.

CHAPTER 8
CONCLUSION

8.1 Future Scope

In conclusion, developing a social media platform has been an enriching project that blends creativity, technical expertise, and strategic thinking. By incorporating a user-friendly interface, responsive design, and a dynamic content management system, the platform enhances user engagement and provides a seamless experience across devices. The integration of real-time features like posting, commenting, and notifications, coupled with robust backend functionalities, fosters interaction and builds a vibrant online community. Additionally, implementing security measures and scalable architecture ensures data integrity and supports the platform's growth, creating a secure and reliable space for users to connect and share ideas.

The project has emphasized the importance of meticulous planning, teamwork, and adaptability to emerging trends in the digital landscape. Addressing challenges such as managing complex relationships between entities, optimizing database queries, and ensuring cross-platform compatibility has not only honed technical skills but also highlighted the importance of problem-solving and resilience in web development.

Furthermore, significant attention was given to creating a personalized and engaging user experience, incorporating features like customizable profiles, multimedia sharing, and a visually appealing design. This project has laid a strong foundation for future iterations and enhancements, setting the stage for a dynamic and scalable social networking solution that meets user expectations and adapts to the evolving needs of the digital community.

8.2 Future Scope

The future scope of the social media platform is expansive and brimming with opportunities to innovate, enhance user experience, and foster community growth. Below are some potential areas for future development:

Personalized Feed and Recommendations:

Integrating machine learning algorithms to analyze user interactions and preferences can enable personalized feed recommendations, making content discovery more relevant and engaging for each user.

Advanced Multimedia Features:

Expanding the platform to support diverse multimedia formats such as live streaming, augmented reality (AR) filters, and interactive polls can make user interactions more dynamic and appealing, catering to a wider audience.

Community-Driven Enhancements:

Introducing features like group discussions, events, and topic-specific communities can foster deeper connections among users. User-generated content moderation and reward systems can further incentivize active participation and loyalty.

Comprehensive Analytics for Users:

Providing advanced analytics tools for users, such as insights into post engagement, audience demographics, and reach, can empower users to optimize their content strategies and build stronger personal or business profiles.

Monetization Opportunities:

Exploring revenue-generating options like targeted advertisements, subscription-based premium features, influencer marketing tools, and marketplace integrations can help sustain and grow the platform while providing value to users.

Mobile App Development:

Creating a dedicated mobile app with features like offline mode, push notifications, and quick-sharing capabilities can enhance accessibility and keep users connected on the go.

Internationalization and Localization:

Expanding the platform to support multiple languages and localizing content for different cultural contexts can attract a global audience, allowing users from diverse backgrounds to connect seamlessly.

Integration with Emerging Technologies:

Incorporating emerging technologies like voice commands, blockchain for enhanced security, and virtual reality (VR) for immersive interactions can elevate the platform's appeal and user experience.

Advanced Security Features:

Strengthening security measures by integrating AI-powered content moderation, multi-factor authentication, and robust data protection protocols can ensure user safety and privacy, fostering trust and reliability.

By focusing on these future enhancements, the social media platform can evolve into a more dynamic, innovative, and engaging ecosystem, catering to the ever-changing needs of its users while staying ahead in the competitive digital landscape.

BIBLIOGRAPHY

- "Full-Stack React Projects" by Shama Hoque
- "Learning React: Modern Patterns for Developing React Apps" by Alex Banks and Eve Porcello
- "MongoDB: The Definitive Guide" by Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow
- "Express in Action: Writing, building, and testing Node.js applications" by Evan Hahn
- "Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node" by Vasan Subramanian
- **Kumar, S., & Sharma, A. (2021).** *Building Scalable Social Media Platforms: A Comprehensive Guide.* TechPress Publishing
- **Smith, J. (2020).** *The Art of UI/UX Design for Web Applications.* Design Insights

Documentation

- MongoDB Documentation
- Express.js Documentation
- React Documentation
- Node.js Documentation