Dharmsinh Desai University, Nadiad

Faculty of Technology,
Department of Computer Engineering

B. Tech. CE Semester – V
Subject: (CE – 520) Advanced Technologies

Project Title: DevChatter (Social networking application for Developers)

Nai Rajat Nareshbhai (CE084)
Parekh Mandar (CE090)

Guided by: Ankit Vaishnav Sir (APV sir)

Content

Abstract

DevChatter is an application built with the sole purpose of bringing developers together on a single platform. It aims to create an inclusive environment that encourages learning, collaboration, and personal growth. The application provides essential functionalities such as user registration and login, post creation and interaction, following/unfollowing users, updating user profiles, participating in chat rooms, and taking up learning challenges. With these features, developers can seamlessly connect with like minded individuals, learn from each other's experiences, and gain valuable insights into the world of software development.

Introduction

In the dynamic landscape of advanced technologies, our project, titled DevCatter, emerges as a versatile and innovative solution to address contemporary challenges. This endeavor was undertaken by Rajat Nai & Mandar Parekh, students of B. Tech. Computer Engineering in the fifth semester at Dharmsinh Desai University, Nadiad.

Brief Overview:

Welcome to DevChatter, a pioneering Social Networking Application tailored specifically for developers. This dynamic platform is meticulously designed to cater to the unique needs of the developer community, fostering collaboration, learning, and meaningful interactions. Here's a concise overview of the key features encapsulated within the DevChatter ecosystem:

Technology/Platform/Tools Used:

The project leverages cutting-edge technologies and platforms to deliver robust and efficient functionality. Key components of our technological stack include:

Programming Languages: JavaScript,
Framework: React, NodeJs, Express
Database Management System: MongoDB
Version Control: Git & GitHub
Development Environment: Visual Studio & GitHub workspace
Other Tools: Selenium(Testing Frontend) & Postman(Testing Backend API)

Software Requirement Specifications (SRS)

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to specify the software requirements for the "Social Networking Application, DevChatter" version 1.0. This document outlines the scope of the product and describes the features and functionalities to be implemented in the application. It serves as a reference for all stakeholders, including developers, testers, and project managers, to ensure a common understanding of the system's requirements.

Project Description:

The "Social Networking Application, DevChatter" is a web based platform that allows users to connect with friends, share posts, participate in coding challenges, and engage in realtime chat rooms. The application aims to provide a user friendly and interactive social networking experience to its users.

1.2 Document Conventions

This SRS follows standard typographical conventions, where requirements are presented in numbered lists or bullet points. Priority levels are assigned to higher level requirements, and these priorities are assumed to be inherited by detailed requirements unless explicitly stated otherwise. Each requirement statement in this document is assigned a priority based on its significance to the overall functionality of the application.

1.3 Intended Audience and Reading Suggestions

The intended audience for this SRS includes but is not limited to:

- Developers: To understand the functional requirements and design the application accordingly.
- Testers: To create test cases and verify that the application meets the specified requirements.
- Project Managers: To plan and track the development process based on the defined scope.
- Marketing Staff: To understand the features of the application for promotional purposes.
- Users: To gain insights into the application's capabilities and functionalities.

Reading Suggestions:

For a comprehensive understanding of the document, readers are suggested to follow the sequence below:

1. Begin with the "Introduction" and “Overall Description” sections to grasp the purpose, scope, and intended audience of the SRS and get an overview of the software's description and objectives..
2. Proceed to the "External Interfaces" section to know how an application is connected to other interfaces.
3. Review the "System Features" section and further to understand functionalities of the application.

1.4 Product Scope

This SRS document covers the overall functionality and behavior of the "Social Networking Application." It includes the core features, such as user registration and login, posting functionality, liking and commenting on posts, following and unfollowing other users, updating user profiles, participating in coding challenges, and joining chat rooms. The document does not address specific implementation details or technical specifications, focusing solely on the functional requirements of the application.

1.5 Reference

IEEE Standard for Software Requirements Specifications
Software Requirements Specification by Karl E. Wiegers.

2. Overall Description

2.1 Product Perspective

The "Social Networking Application, DevChatter" is a new, selfcontained product designed to provide users with a platform for social interactions, content sharing, and skill improvement through coding challenges. It is not a followon member of an existing product family or a replacement for any existing system. The application will be developed from scratch as a standalone system.

Product Context:

The "Social Networking Application, DevChatter" will be a webbased platform accessible via standard web browsers on desktop and mobile devices. Users will be able to register, log in, create profiles, share posts, like and comment on posts, follow other users, participate in coding challenges, and engage in realtime chat rooms.

Interfaces:

- User Interface: The application will have an intuitive and userfriendly interface that allows users to interact with the various features easily.
- Database Interface: The application will interact with a database to store user information, posts, challenge data, and chat room messages.
- GitHub Integration: To enable users to post their coding challenge progress on GitHub, the application will integrate with the GitHub API.
- Realtime Communication: The chat room functionality will involve realtime communication between users using WebSocket or similar technology.

2.2 Product Functions

The major functions of the "Social Networking Application" are as follows:

User Registration: Users can create new accounts by providing their name, email, and password.
User Login: Registered users can log in to the application using their credentials.
User Profile: Users can update their profile information, including their name, profile image, and email.
Post Sharing: Users can share posts in text or image format with their followers and the community.
Like and Comment: Users can like and comment on their own and others' posts to interact with each other.
Follow and Unfollow: Users can follow or unfollow other users to stay updated with their activities.
Coding Challenges: Users can participate in coding challenges like "100 Days of Code" or "30 Days of DSA."

GitHub Integration: Users can post their coding challenge progress which they have done on GitHub using the application for which GitHub API is helpful.
Realtime Chat Room: Users can join chat rooms to communicate with other participants in realtime.

2.3 User Classes and Characteristics

The "Social Networking Application, DevChatter" is designed to cater to various user classes with different characteristics and usage patterns. The following user classes are anticipated to use this product:

1. Regular Users:

- Frequency of Use: Regular users are expected to use the application frequently, engaging with posts, likes, comments, and chat rooms regularly.
- A subset of Functions: Regular users will use the majority of the product's functions, including post sharing, liking, commenting, and participating in coding challenges.
- Technical Expertise: Regular users may have varying levels of technical expertise, ranging from novice to experienced users.
- Security/Privilege Levels: Regular users will have standard privileges, allowing them to interact with the platform's basic features.
- Educational Level/Experience: Educational backgrounds and experiences may vary among regular users.

2. Developers:

- Frequency of Use: Developers are expected to use the application frequently to participate in coding challenges, share their progress, and engage in discussions.
- A subset of Functions: Developers will be more focused on coding challenges, GitHub integration, and technical discussions in chat rooms.
- Technical Expertise: Developers are likely to have higher technical expertise and coding skills.
- Security/Privilege Levels: Developers may have elevated privileges, allowing them to access additional features related to coding challenges and GitHub integration.
- Educational Level/Experience: Developers will have a higher level of technical education and experience in software development.

3. Project Managers and Administrators:

- Frequency of Use: Project managers and administrators will use the application for monitoring user activity and managing the overall platform.
- A subset of Functions: Their primary focus will be on user management, content moderation, and system administration.
- Technical Expertise: Project managers and administrators may have moderate to high technical knowledge, allowing them to handle administrative tasks effectively.
- Security/Privilege Levels: They will have elevated privileges to manage user accounts, posts, and other platform settings.

- Educational Level/Experience: Project managers and administrators may have backgrounds in project management or system administration.

2.4 Operating Environment

The "Social Networking Application, DevChatter" will operate in the following environment:

- **Hardware Platform**: The application will run on standard web browsers across desktop computers, laptops, tablets, and mobile devices.
- **Operating System**: The application will be compatible with various operating systems, including Windows, macOS, Linux, iOS, and Android.
- **Web Server**: The application will be hosted on a web server with adequate processing power and memory to handle user requests and data storage.
- **Database**: The application will interact with a backend database system(e.g., MongoDB) to store user information, posts, challenge data, and chat room messages.
- **Software Components**:The application will utilize programming languages (e.g., JavaScript, CSS, HTML) and frameworks (e.g., Node.js, React.js, tailwindCss, Redux) to build its functionalities.
- **External Applications**: The application will integrate with the GitHub API to facilitate the user's post their GitHub progress on the application.

2.5 Design and Implementation Constraints

The design and implementation of the "Social Networking Application, DevChatter" are subject to the following constraints:

1. **<u>Technology Stack:</u>** The development team must use specific technologies and frameworks, such as Node.js for backend development and React.js for frontend development, as specified by the project's technical requirements.

2.**<u>GitHub Integration</u>**: The application must integrate with the GitHub API to enable users to the user's post their GitHub progress on the application. The integration must adhere to the security guidelines and authentication mechanisms set by GitHub.

3. **<u>Security Considerations</u>**: The application must implement secure authentication and authorization mechanisms to protect user data and prevent unauthorized access. It must also include measures to prevent common security threats, such as SQL injection, crosssite scripting (XSS), and crosssite request forgery (CSRF).
4. **<u>Responsive Design</u>**: The application must be designed to provide a seamless user experience across various devices, including desktops, laptops, tablets, and mobile phones.

5. **Performance Requirements**: The application should be designed with performance in mind to ensure smooth and responsive user interactions, even with a large number of active users and extensive content.

6. **Scalability**: The architecture of the application should allow for future scalability, enabling it to handle a growing number of users and content without compromising performance.

7. **Database Management**: The database system chosen for the application must be capable of handling concurrent read and write operations efficiently to ensure data consistency and integrity.

8. **Compliance and Regulations**: The application must comply with relevant corporate and regulatory policies regarding user data privacy, content moderation, and intellectual property rights.

2.6 User Documentation

The "Social Networking Application, DevChatter" will be accompanied by comprehensive user documentation to assist users in understanding and utilizing its features effectively. The user documentation will include the following components:

1. **User Manual:** A detailed user manual will provide stepbystep instructions on how to use the application's various features, including user registration, postsharing, coding challenge participation, and chat room engagement.

2. **Online Help**: The application will feature online help accessible from within the application interface. This help documentation will offer contextspecific guidance on using specific functions and resolving common issues.

3. **Tutorials**: Tutorials and guides will be provided to support users, especially developers, in leveraging GitHub integration, participating in coding challenges, and making the most of the application's codingrelated functionalities.

4. **Support and Contact Information**: Contact information for customer support and technical assistance will be provided to users, enabling them to seek help or report issues related to the application.

2.7 Assumptions and Dependencies

The development of the "Social Networking Application, DevChatter" is based on certain assumptions and dependencies. These factors could affect the requirements stated in the SRS, and it is crucial to address them to ensure the project's success:

Assumptions:

1. **Availability of ThirdParty Components**: It is assumed that the necessary third party or commercial components, such as Node.js, React.js, and GitHub API, will be available and compatible with the project requirements.

2. **User Internet Connectivity**: The application assumes that users will have stable internet connectivity to access and interact with the platform effectively.

3. **Secure GitHub Integration**: It is assumed that the integration with the GitHub API will be implemented securely, adhering to GitHub's authentication and security guidelines.

4. **Responsive Web Design**: The application assumes that users will primarily access the platform through various devices with different screen sizes, necessitating responsive web design for optimal user experience.

5. **Data Backup and Recovery**: It is assumed that appropriate data backup and recovery mechanisms will be in place to ensure data integrity and availability in the event of data loss or system failure.

Dependencies:

1. **GitHub API Integration**: The successful implementation of GitHub integration is critical to enable users to post their GitHub progress on the application.

2. **Database Management System**: The application depends on the chosen database management system to efficiently store and manage user information, posts, challenge data, and chat room messages.

3. **Web Hosting Infrastructure**: The availability and reliability of the web hosting infrastructure are essential for the application to be accessible to users at all times.

4. **Availability of Development Team**: The timely completion of the project depends on the availability and dedication of the development team.

5. **Compliance with Corporate Policies**: The project's success relies on adhering to corporate policies and regulatory requirements, especially concerning data privacy and security.

6. **External Services and APIs**: The application may rely on external services or APIs (e.g., for image hosting, realtime chat) that should be available and functional for the application to operate as intended.

7. **User Adoption and Engagement**: The success of the "Social Networking Application, DevChatter" depends on user adoption and engagement, as the platform's value is directly related to the active participation of its user base.

3. External Interface Requirements

3.1 User Interfaces

The "Social Networking Application, DevChatter" will have a userfriendly and intuitive user interface to facilitate seamless interactions between users and the platform. The user interface will be accessible through standard web browsers on desktop computers, laptops, tablets, and mobile devices. Below are the logical characteristics of the user interface

1. **User Registration and Login Interface:**
    - This interface will provide fields for users to enter their name, email, and password during registration.
    - Users will have the option to log in using their registered email and password.
    - Error message display standards will be followed in case of invalid inputs or login credentials.

2. **User Profile Interface:**
    - The user profile page will allow users to view and edit their profile information, including name, profile image, and email.
    - Users can update their profile details and save the changes.
    - The user interface will follow layout constraints to ensure consistency and ease of navigation.

3. **PostSharing Interface:**
    - Users will have a text box to write and share posts in text format.
    - An option to upload and share images as posts will be provided.
    - The user interface will include buttons for post submission and options to add tags or categories to the post.

4. **Like and Comment Interface:**
    - The interface will display the "Like" button and the number of likes underneath each post.
    - Users can click on the "Like" button to add or remove their likes from a post.
    - A comment button will be available to add comments to posts, and users can also delete their own comments.

5. **Follow and Unfollow Interface:**
    - Users can follow or unfollow other users by clicking on the respective buttons on the user profiles or postcards.
    - The interface will provide visual cues to indicate follow status.

6. **Coding Challenge Interface:**

- The interface will display a list of available coding challenges, each with a brief description.
- Users can select a challenge to view its details and rules before participating.
- Upon completion, users can post their GitHub challenges which are conducted by application progress on the application.

7. Realtime Chat Room Interface:
- The interface will display an input box for users to type and send realtime messages in chat rooms.
- Users will see messages from other participants as they are sent, providing a live chat experience.
- The user interface will include icons or indicators to show the online status of chat room participants.

8. Error Handling Interface:
- In case of errors, the interface will display clear and informative error messages to guide users on how to rectify the issue.
- Consistent error message display standards will be followed throughout the application.

3.2 Hardware Interfaces

The "Social Networking Application" will interact with hardware components in the following ways:

1. Supported Device Types:
- The application will be accessible through standard web browsers on desktop computers, laptops, tablets, and mobile devices running various operating systems (Windows, macOS, Linux, iOS, and Android).

2. Data and Control Interactions:
- The application will handle user input through hardware peripherals such as keyboards, mice, and touchscreens for interaction with the user interface.
- The hardware components will process user commands and transmit data to the software for processing.

3. Communication Protocols:
- The application will utilize standard communication protocols such as HTTP and WebSocket for interactions with web servers and realtime chat functionality.

3.3 Software Interfaces

The "Social Networking Application" will have connections with the following software components:

1. **Databases:**
    - The application will interact with a backend database system (e.g., MongoDB) to store and retrieve user information, posts, challenge data, and chat room messages.

2. **Operating Systems:**
    - The application will be compatible with various operating systems, including Windows, macOS, Linux, iOS, and Android.

3. **Programming Languages and Frameworks:**
    - The application will be developed using programming languages like JavaScript (Node.js) for the backend and React.js for the front end.

4. **GitHub API:**
    - The application will integrate with the GitHub API to enable users to post their coding challenge progress on GitHub.

5. **RealTime Communication:**
    - To enable realtime chat functionality, the application may utilize technologies like WebSockets for bidirectional communication between the client and the server.

Data Sharing and Communication:

- Data items such as user profile information, posts, likes, comments, and chat room messages will be shared between different software components through APIs and database queries.
- The application will utilize RESTful APIs to communicate with the backend server for various operations, such as posting updates to the database or retrieving user information.

3.4 Communications Interfaces

The "Social Networking Application" will have the following communication requirements:

1. Web Browser:
    - The application will be accessible through standard web browsers, such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.

2. HTTP and HTTPS:
    - Communication between the client and server will occur over HTTP/HTTPS protocols for secure data transfer.

3. RealTime Chat:
    - The realtime chat functionality will require WebSocket communication to enable live messaging between users in chat rooms.

4. Data Transfer Rates and Synchronization:
   - The data transfer rates will depend on the users' internet connectivity and server response times.
   - The application may implement synchronization mechanisms to ensure realtime updates of user activities, such as post likes or new chat messages.

5. Communication Security and Encryption:
   - Communication will be secured using HTTPS to protect sensitive user data during transmission.
   - WebSocket connections may also use encryption for secure realtime chat communication.

4. System Features

Below we can see all the system features of the “Social Networking Application, DevChatter”.

4.1 User registration and login:

Description: Users should be able to create an account and log in to the application to access features for sharing posts on the other apps.

4.1.1 For signup

Input: name, email, and password.

Process: The system will check for the username in the database and give appropriate feedback regarding its availability.

Output: If the email is available then a New account will be created for the user after validating the email address, and the user will be redirected to the homepage. If an email is not available then prompt the email is not available and go to **R1.1.**

4.1.2. For login of users

Input: Username and password created by the user.

Process: The system will validate the entered information and give appropriate feedback regarding the user's identity.

Output: If Correct credentials are given prompt with a message of successful login and redirected to the homepage else go to R1.2 again.

4.2 Post on the application:

4.2.1 Share Post

Description: The user can share a post in text or image format on the application.

Input: The user writes the post he wants or adds the image he wants to upload.

Process: If the post contains something that violates the community guideline then he can not post it, or if doesn’t violate then he can share the post on the application.

Output: The post will be uploaded to the application or It gives an error about the violating guideline.

4.2.2 Delete Post

Description: The user can delete his post from the application

Input: Select the delete option on the post

Output: The post will be removed from the application

4.3 Like and Comment on others posts:

Description: User can like and comments on their own and others' posts.

4.3.1 Like Functionality

Description: Users can Like and Unlike the post

- **4.3.1.1 Like the post**
  Input: Click on the Like button which is located underneath the post
  Output: User like added to the post
- **4.3.1.2 Unlike the post**
  Input: Click on the Like button which is located underneath the post
  Processing: If a post is already liked then remove it
  Output: User like will be removed from the post

4.3.2 Comment Functionality

Description: Use can add and remove the comment from the post

- **4.3.2.1 Add Comment on the post**
  Input: Select the comment option in the post
  Processing: New prompt will be open for the input of the comment
  Output: After clicking on the post button comment will be added in the post comment section.

- **4.3.2.2 Remove Comment from the post**
  Input: Click on the comment button which is located underneath the post, and side the like button
  Processing: New prompt will be open for the input of the
  Output: After clicking on the post button comment will be added in the post comment section.

4.4 Follow and Unfollow other users:

Description: User can follow and unfollow any other user he want.

Input: User click on the follow button of the other user whom he want to follow Or if already is following than click on that for unfollow user

Output: Based on the user selection user will followed or unfollowed.

4.5 Update user profile:

Description: The user can update his profile in the profile section. Where he can update his name, password, profile image, email, etc.

Input: The user will click on the update which opens the form for an update that.

Processing: If updated details, follow all the regulations then it will update the user profile or it will give an error message.

Output: The user profile will be updated.

4.6 Chat Room:

Description: Users can join a chat room and communicate with everyone in the room.

Input: The user clicks on the "Join Chat Room" option, and enters sends messages in the chat input box.

Processing: When the user clicks on "Join Chat Room," the system adds the user to the chat room. The chat room displays all the messages sent by users in real time. When a user sends a message, the system broadcasts the message to all the participants in the chat room.

Output: Users can see all the messages sent in the chat room. Participants can see the display names or usernames of other users who have joined the chat. If a user joins or leaves the chat room, a notification may be displayed to other participants.

4.7 Learning Challenges

Description: Where user can take a coding challenge like “100 Days of Code" or "30 Days of DSA.” When user participate in the challenge and push their learning on GitHub from that application make a post about what they did today and will post it, and after completing the challenge they get a badge

4.7.1 Participate in Challenges

Input: The user can browse the list of available coding challenges and the user selects a challenge they want to participate in.

Processing: Upon selecting a challenge, the user is provided with the challenge details, rules, and any specific requirements.

Output: The user can see the challenge details and start working on it and after completing the challenge, the user proceeds to the next steps (R7.2 and R7.3)

4.7.2 Post Challenge Progress

Input: The user completes the coding challenge and the user integrates their GitHub account with the application.

Processing: After completing the challenge, the user has to push their code solution on the GitHub repository.

Output: The user's challenge progress is posted on the application.

4.7.3 Earn Badges

Input: The user participates in the coding challenge.

Processing: The application evaluates the user's performance based on the challenge completion time or other defined criteria. If the user meets the requirements, they are awarded a badge for that particular challenge.

Output: The user receives a badge for completing the challenge successfully.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The "Social Networking Application, DevChatter" must meet specific performance requirements to ensure a smooth and responsive user experience under various circumstances. The performance requirements are as follows:

1. **Response Time:**
    - The application should respond to user interactions (e.g., clicks, submissions) within 1 to 2 seconds to provide a seamless and interactive user experience.

2. **Loading Time:**
    - The application's initial loading time should be optimized to ensure that users can access the platform quickly without significant delays.

3. **Scalability:**
    - The application should be scalable to accommodate a growing number of users and increased data traffic without significant degradation in performance.

4. **Concurrency Handling:**
    - The application should handle concurrent user interactions (e.g., likes, comments) efficiently to avoid delays or conflicts in data updates.

5. **Chat Room RealTime Communication:**
    - The realtime chat feature should have minimal latency to ensure smooth and instantaneous communication between chat room participants.

6. **Coding Challenge Response Time:**
    - The application should promptly respond to users' coding challenge submissions and GitHub integration, especially during peak times when multiple users may be participating simultaneously.

7. **Image Upload Performance:**
    - The image upload feature should efficiently handle the upload and display of images to avoid user frustration due to slow image loading.

5.2 Safety Requirements

The "Social Networking Application, DevChatter" is not primarily safetycritical, but certain safety requirements should be considered to ensure the wellbeing and privacy of users. These safety requirements include:

1. **User Privacy Protection:**
    - The application must implement robust data privacy measures to safeguard user information and prevent unauthorized access to sensitive data.

2. Content Moderation:
    - The platform should employ content moderation techniques to identify and remove inappropriate or harmful content to maintain a safe and positive user experience.

3. Data Backup and Recovery:
    - Regular data backups should be performed to ensure data integrity and facilitate recovery in the event of data loss or system failure.

4. Error Handling:
    - The application should handle errors gracefully, providing clear and informative error messages to users to prevent unintended actions that could lead to data loss or other safety concerns.

5. User Reporting and Blocking:
    - The application should allow users to report inappropriate or harmful content and provide the option to block or unfollow other users to protect their safety and wellbeing.

6. Prevention of Malicious Activities:
    - The application should implement security measures to prevent activities such as hacking, spamming, or phishing that may compromise user safety.

External Policies and Regulations:

The application should comply with relevant data protection laws and regulations, ensuring that user data is handled with the utmost care and adherence to privacy standards.

Safety Certifications:

The "Social Networking Application, DevChatter" is not subject to specific safety certifications. However, compliance with data protection regulations and industry best practices will contribute to user safety and data privacy.

5.3 Security Requirements

The "Social Networking Application, DevChatter" must prioritize security and privacy to protect user data and ensure a secure user experience. Security requirements include:

1. User Identity Authentication:

- The application should employ secure user authentication mechanisms, such as password hashing and encryption, to verify the identity of users.

2. Secure Communication:
    - The application must use secure communication protocols (e.g., HTTPS) to encrypt data transmission between the client and the server, preventing data interception or tampering.

3. Access Control:
    - Rolebased access control should be implemented to restrict access to sensitive features and data based on user roles and privileges.

4. Data Encryption:
    - Sensitive user data, such as passwords and personal information, should be stored in an encrypted format to prevent unauthorized access to the data store.

5. Protection against CrossSite Scripting (XSS) and SQL Injection:
    - The application should implement measures to prevent XSS and SQL injection attacks, ensuring that user input is sanitized and validated before processing.
6. User Session Management:
    - User sessions should be managed securely to prevent session hijacking and unauthorized access to user accounts.

7. Security Auditing and Monitoring:
    - Regular security audits and monitoring should be conducted to identify and address potential security vulnerabilities and threats.

External Policies and Regulations:

The application should comply with relevant data protection and privacy regulations, as well as security standards and guidelines.

Security Certifications:

Depending on the application's scale and intended usage, security certifications such as ISO/IEC 27001 or SOC 2 Type II may be sought to demonstrate the application's commitment to security and privacy standards.

5.4 Software Quality Attributes

The "Social Networking Application, DevChatter" should possess the following software quality attributes to meet customer expectations and facilitate efficient development:

1. Usability:
    - The application should have an intuitive and userfriendly interface, ensuring ease of use for both novice and experienced users.

- Usability can be measured through user satisfaction surveys and usability testing.

2. Reliability:
   - The application should operate consistently without unexpected crashes or errors.
   - Reliability can be measured through Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR).

3. Performance Efficiency:
   - The application should respond promptly to user interactions and provide quick loading times for posts and content.
   - Performance efficiency can be measured through response time and loading time metrics.

4. Security:
   - The application must ensure the confidentiality, integrity, and availability of user data, employing strong encryption and secure authentication mechanisms.
   - Security can be measured through vulnerability assessments and penetration testing.

5. Maintainability:
   - The application's codebase should be wellstructured and modular to facilitate easy maintenance and updates.
   - Maintainability can be measured through code complexity metrics and change request response time.

6. Scalability:
   - The application should be capable of handling an increasing number of users and data without compromising performance.
   - Scalability can be measured through load testing and stress testing.

7. Portability:
   - The application should be designed to be easily deployable on different platforms and operating systems.
   - Portability can be measured through successful deployments on multiple environments.

8. Testability:
   - The application's components should be designed for easy testing, with comprehensive unit tests and integration tests in place.
   - Testability can be measured through code coverage and test success rates.

5.5 Business Rules

These are the business rules:

1. User Roles and Permissions:

- Only registered users with valid credentials can access the application's features, such as posting, liking, commenting, and following/unfollowing.
- Certain administrative functions (e.g., content moderation) may be restricted to authorized personnel or administrators.

2. Content Moderation:
   - The application should enforce content moderation rules to ensure that users adhere to community guidelines and avoid posting inappropriate or harmful content.
   - Violation of content guidelines may result in user suspension or content removal.

3. Challenge Participation:
   - Users can participate in coding challenges and push their progress on GitHub to earn badges.
   - The application should track challenge completion and provide users with badges upon successful completion.

4. Chat Room Participation:
   - Users can join chat rooms and participate in realtime conversations with other users.
   - The application may have chat room moderators to enforce chat room rules and maintain a positive environment.

6. Other Requirements

Database Requirements:

- The application should utilize a robust and scalable database system (e.g., MOngoDBL) to store user information, posts, chat messages, and coding challenge data.
- The database design should adhere to normalization principles to ensure data integrity and efficiency.
- The database should be regularly backed up to prevent data loss.

Internationalization Requirements:

- The application should be designed to support multiple languages to cater to a diverse user base.
- User interface elements, such as labels, buttons, and messages, should be translatable to different languages.
- Internationalization should adhere to industry standards and best practices.

Legal Requirements:

- The application should comply with all relevant laws and regulations governing data privacy, intellectual property rights, and online content.

- Users should be required to agree to the application's terms of service and privacy policy during registration.

Reuse Objectives:

- The development team should strive to identify and leverage reusable components and libraries to expedite development and maintain consistency across projects.

Appendix A: Glossary

- TBD - To Be Determined
- API - Application Programming Interface
- XSS - Cross-Site Scripting
- HTTP - Hypertext Transfer Protocol
- HTTPS - Hypertext Transfer Protocol Secure
- MTBF - Mean Time Between Failures
- MTTR - Mean Time to Repair

Appendix B: To Be Determined List

1. TBD - Performance test results and thresholds for response times.
2. TBD - Specific internationalization standards and languages supported.
3. TBD - Database schema and normalization details.
4. TBD - Chat room moderation guidelines and rules.
5. TBD - Security certifications, if applicable

Database Design

1. Introduction to Database Design:

The application uses MongoDB as the database management system (DBMS). Mongoose is the ODM (Object Data Modeling) library for MongoDB and is used to define the structure of the data.

2. Entity-Relationship Diagram (ERD):

User Entity:

Attributes:

- _id (Primary Key)
- Name
- avtar (nested object with public_id and url)
- Email
- password (hashed)
- posts (references the Post model)
- followers (references the User model)
- following (references the User model)
- keywords (array of strings)
- createdAt
- resetPasswordToken
- resetPasswordExpire

Post Entity:

Attributes:

- _id (Primary Key)
- Caption
- image (nested object with public_url and url)
- owner (references the User model)
- createdAt
- likes (array of objects, each with a user field referencing the User model)
- comments (array of objects, each with user and comment fields referencing the User model)

3. Database Schema:

User Schema:

- name: String
- avtar: Object
- public_id: String

- url: String
- email: String (Unique)
- password: String (Hashed, not selected in queries)
- posts: Array of ObjectIds (References Post model)
- followers: Array of ObjectIds (References User model)
- following: Array of ObjectIds (References User model)
- keywords: Array of Strings
- createdAt: Date
- resetPasswordToken: String
- resetPasswordExpire: Date

Post Schema:
- caption: String
- image: Object
- public_url: String
- url: String
- owner: ObjectId (References User model)
- createdAt: Date
- likes: Array of Objects
- user: ObjectId (References User model)
- comments: Array of Objects
- user: ObjectId (References User model)
- comment: String

4. Normalization:

The schemas appear to be adequately normalized with separate entities for users and posts.

5. Tables and Relationships:

User Table:
- Primary Key: _id
- Foreign Keys: posts, followers, following (Referencing Post model, User model)

Post Table:
- Primary Key: _id
- Foreign Key: owner (Referencing User model)

6. Constraints:

- Unique constraint on the email field in the User schema.
- Passwords are hashed before being stored in the database.

7. Indexes:

- None explicitly defined in the provided schemas.

8. Security Measures:

- Passwords are hashed using bcrypt before being stored in the database.

This database design provides a foundation for storing and retrieving user and post-related data efficiently. It establishes clear relationships between users and posts, allowing for a scalable and well-structured data model.

Implementation Detail

i) Modules Created and Brief Description:

Post Management Module:

createPost:

Description: Creates a new post with an optional image.
Dependencies: Post model, User model, Cloudinary for image storage.
Major Steps:
Uploads image to Cloudinary.
Creates a new post using Post.create.
Associates the post with the owner (req.user) and updates user's posts.
Responds with the created post.

deletePost:

Description: Deletes a post and removes its reference from the user's posts.
Dependencies: Post model, User model.
Major Steps:
Verifies ownership of the post.
Deletes the post and updates user's posts.
Responds with a success message.

likeAndUnlikePost:

Description: Handles post likes and unlikes.
Dependencies: Post model, User model.
Major Steps:
Checks if the user has already liked the post.
Adds or removes the like accordingly.
Responds with the updated like count.

getPostOfFollowing:

Description: Retrieves posts of users being followed by the authenticated user.
Dependencies: User model, Post model.
Major Steps:
Retrieves users being followed by the authenticated user.
Fetches posts of those users.
Responds with the posts.

updateCaption:

Description: Updates the caption of a post.
Dependencies: Post model.

Major Steps:
Verifies ownership of the post.
Updates the caption.
Responds with a success message.

commentOnPost:

Description: Adds or updates a comment on a post.
Dependencies: Post model.
Major Steps:
Finds the post.
Checks if the user has already commented.
Adds or updates the comment.
Responds with a success message.

deleteComment:

Description: Deletes a comment from a post.
Dependencies: Post model.
Major Steps:
Verifies ownership of the post or comment.
Deletes the specified comment.
Responds with a success message.

getMyPosts:

Description: Retrieves posts created by the authenticated user.
Dependencies: Post model.
Major Steps:
Fetches posts associated with the authenticated user.
Responds with the posts.

getLikedPosts:

Description: Retrieves posts liked by the authenticated user.
Dependencies: Post model.
Major Steps:
Fetches posts where the user's ID is in the likes array.
Responds with the posts.

getRecommendation:

Description: Retrieves posts for recommendation.
Dependencies: Post model.
Major Steps:
Fetches all posts.

Responds with the posts.

getPostById:

Description: Retrieves a specific post by ID.
Dependencies: Post model.
Major Steps:
Fetches the post by ID.
Responds with the post.

User Authentication Module:

register:

Description: Registers a new user with optional avatar.
Dependencies: User model, Cloudinary for avatar storage.
Major Steps:
Uploads avatar to Cloudinary.
Creates a new user using User.create.
Generates a token for authentication.
Sets a cookie with the token.
Responds with the user and token.

login:

Description: Logs in a user and sets an authentication token.
Dependencies: User model.
Major Steps:
Verifies user credentials.
Generates a token for authentication.
Sets a cookie with the token.
Responds with the user and token.

logout:

Description: Logs out a user by clearing the authentication cookie.
Major Steps:
Clears the authentication cookie.
Responds with a success message.

followUser:

Description: Follows or unfollows a user.
Dependencies: User model.
Major Steps:
Checks if the user is already following the target user.

Adds or removes the relationship accordingly.
Responds with a success message.

updatePassword:

Description: Updates the password of the authenticated user.
Dependencies: User model.
Major Steps:
Verifies the old password.
Updates the password with the new one.
Responds with a success message.

updateProfile:

Description: Updates the profile details and avatar of the authenticated user.
Dependencies: User model, Cloudinary for avatar storage.
Major Steps:
Uploads a new avatar to Cloudinary.
Updates the user's profile details.
Responds with a success message.

deleteMyProfile:

Description: Deletes the profile of the authenticated user, including posts and relationships.
Dependencies: User model, Post model.
Major Steps:
Logs the user out by clearing the authentication cookie.
Deletes user's posts.
Removes user from followers' following list.
Deletes the user.
Responds with a success message.

myProfile:

Description: Retrieves the profile of the authenticated user.
Dependencies: User model.
Major Steps:
Fetches the user details.
Responds with the user.

forgotPassword:

Description: Initiates the password reset process by sending an email with a reset link.
Dependencies: User model, email sending functionality.

Major Steps:
Generates a reset token and sets an expiration time.
Sends a password reset email.
Responds with a success message.

resetPassword:

Description: Resets the password using a valid reset token.
Dependencies: User model.
Major Steps:
Verifies the reset token.
Updates the password with the new one.
Responds with a success message.

getAllUsers:

Description: Retrieves all users in the system.
Dependencies: User model.
Major Steps:
Fetches all users.
Responds with the users.

getUser:

Description: Retrieves a specific user by ID.
Dependencies: User model.
Major Steps:
Fetches the user by ID.
Responds with the user.

searchUser:

Description: Searches for users by name using a case-insensitive regex.
Dependencies: User model.
Major Steps:
Fetches users matching the provided name.
Responds with the matching users.

getUserByEmail:

Description: Retrieves a user by email.
Dependencies: User model.
Major Steps:
Fetches the user by email.
Responds with the user.

Testing

Backend Testing:

Testing Method:

For the backend testing, manual testing using Postman is employed. Postman allows the creation and execution of HTTP requests to test the functionality of each API endpoint. This method is suitable for verifying the correctness of API responses, authentication processes, and database interactions.

Test Cases:

User Registration:

Test Case: Verify that a new user can successfully register.
Steps:
Provide valid registration details.
Submit the registration form.
Expected Result: User is successfully registered, and data is stored in the database.

User Login:

Test Case: Ensure that a registered user can log in.
Steps:
Enter valid login credentials.
Submit the login form.
Expected Result: User is authenticated and receives a valid access token.

Create Post:

Test Case: Check if a user can create a new post.
Steps:
Authenticate a user.
Submit a new post with valid data.
Expected Result: Post is created in the database, and the user's posts list is updated.

Like/Unlike Post:

Test Case: Validate the functionality of liking and unliking a post.
Steps:
Authenticate a user.
Like a post and then unlike it.

Expected Result: Post likes are updated accordingly.

Follow/Unfollow User:

Test Case: Test the ability to follow and unfollow another user.
Steps:
Authenticate a user.
Follow a user and then unfollow the same user.
Expected Result: Follower and following lists are updated correctly.

Update User Profile:

Test Case: Ensure that a user can update their profile details.
Steps:
Authenticate a user.
Update profile details and save.
Expected Result: User profile details are updated in the database.

Reset Password:

Test Case: Confirm that the password reset functionality works.
Steps:
Request a password reset.
Use the reset link to set a new password.
Expected Result: Password is successfully updated.

Delete Post:

Test Case: Check if a user can delete their own post.
Steps:
Authenticate a user.
Delete one of their posts.
Expected Result: Post is deleted, and user's posts list is updated.

Frontend Testing:

Testing Method:

We are using the Selenium for the filling up all the forms so we don’t have to fill up the form repetitively.

Test Cases:

User Interface Rendering:

Test Case: Verify that key UI components render as expected.
Steps:
Navigate through different pages and components.
Expected Result: UI components render without errors.

User Authentication:

Test Case: Ensure that the login and registration forms function correctly.
Steps:
Test login with valid and invalid credentials.
Test registration with valid and invalid input.
Expected Result: Forms validate input and handle authentication flows correctly.

Post Creation:

Test Case: Validate the post creation process.
Steps:
Create a new post.
Check if the post appears in the user's feed.
Expected Result: New post is created and displayed.

User Interaction:

Test Case: Test user interactions such as liking a post or following another user.
Steps:
Like and unlike a post.
Follow and unfollow another user.
Expected Result: User interactions update the UI and trigger backend actions.

Profile Management:

Test Case: Check if users can update their profile details.
Steps:
Navigate to the profile page.
Update profile details.
Expected Result: User profile is updated and reflects the changes.

Responsive Design:

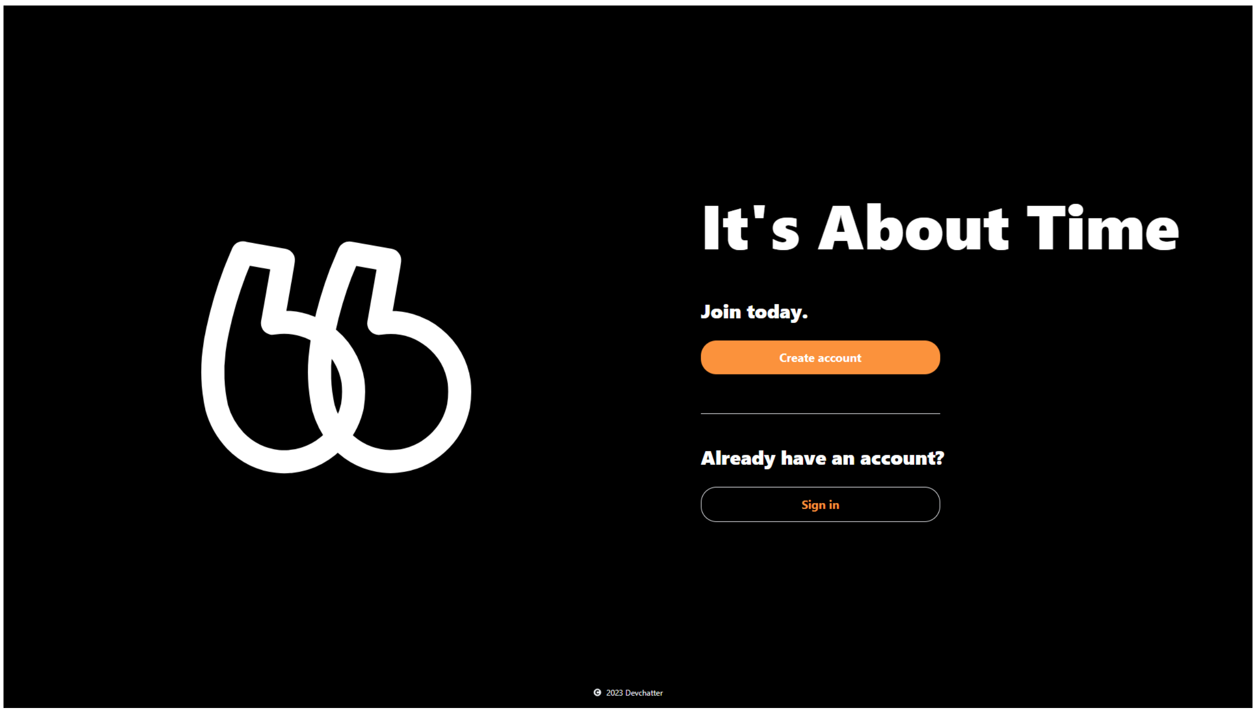
Test Case: Ensure that the application is responsive across different devices.
Steps:
Access the application from various devices and screen sizes.

Expected Result: UI elements adapt to different screen sizes appropriately.

Error Handling:

Test Case: Validate error handling mechanisms.
Steps:
Intentionally trigger errors, such as submitting invalid forms.
Expected Result: User-friendly error messages are displayed.

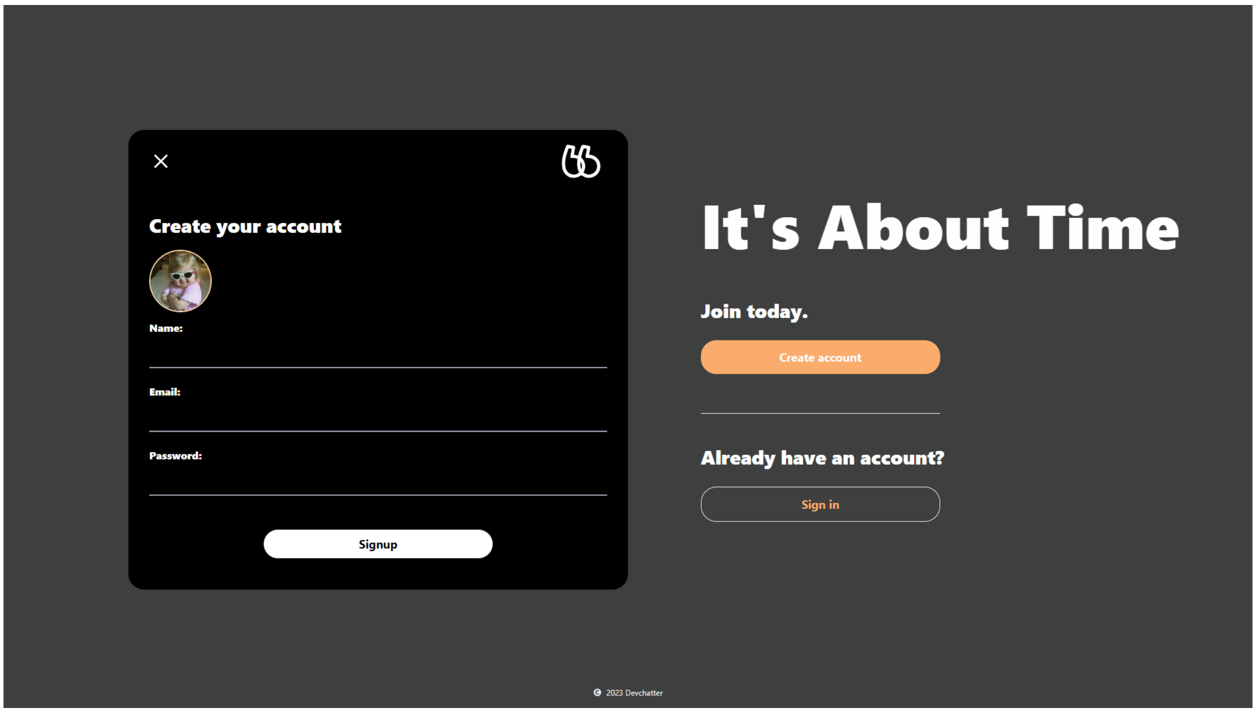
Screen-shots


It's About Time

Join today.

Already have an account?

© 2023 Devchatter


Create your account

Name:

Email:

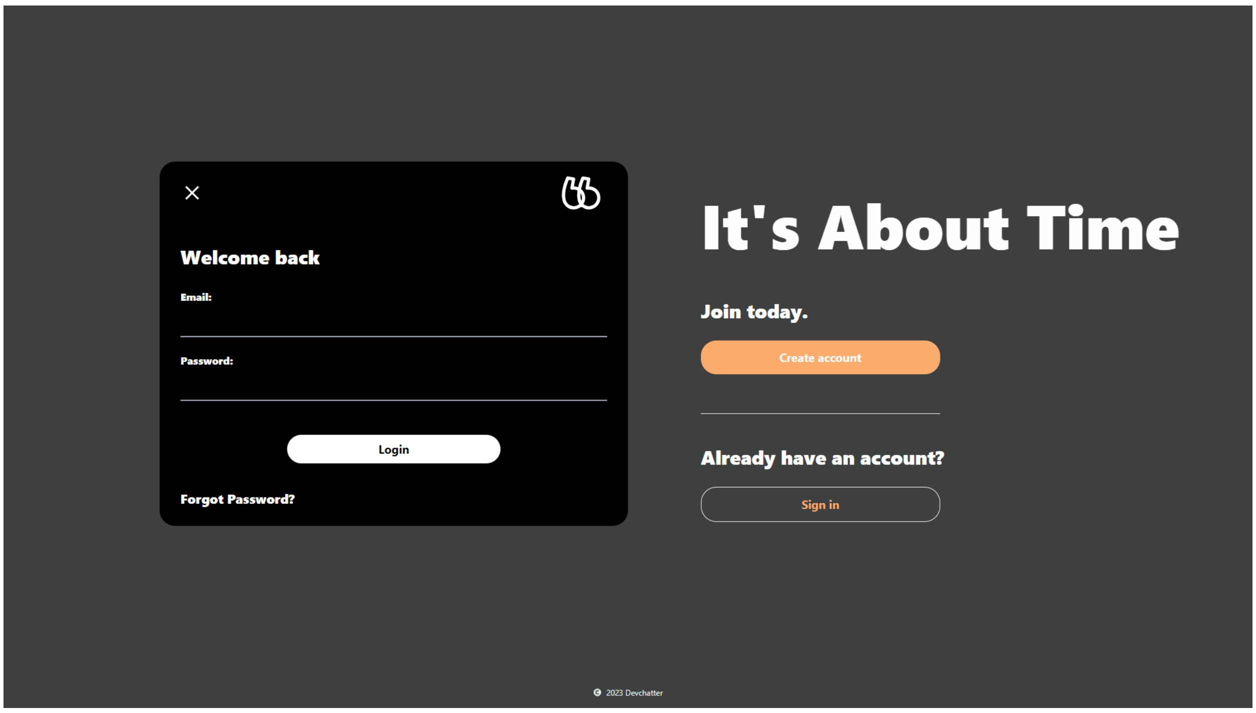
Password:

It's About Time

Join today.

Already have an account?

© 2023 Devchatter



Welcome back

Email:

Password:

Login

Forgot Password?

It's About Time

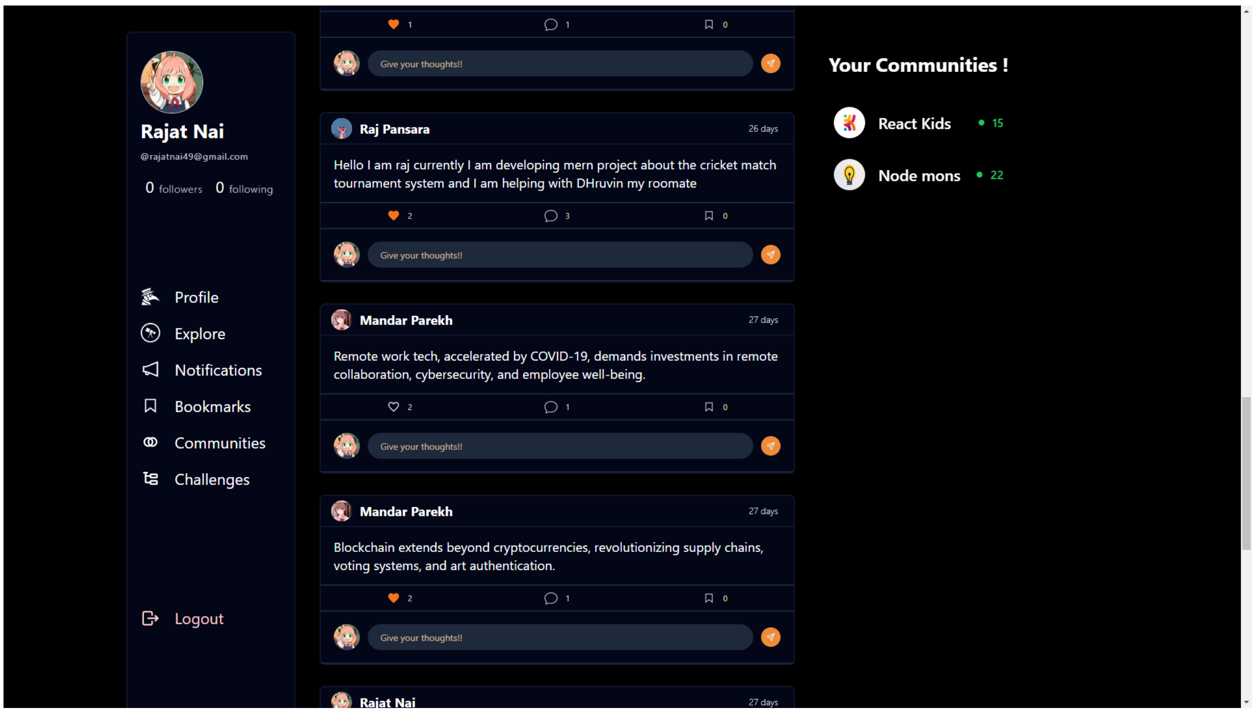
Join today.

Create account

Already have an account?

Sign in

© 2023 Devchatter



Rajat Nai

@rajatnai49@gmail.com

0 followers 0 following

Profile

Explore

Notifications

Bookmarks

Communities

Challenges

Logout

1 1 0

Give your thoughts!!

Raj Pansara 26 days

Hello I am raj currently I am developing mern project about the cricket match tournament system and I am helping with DHruvin my roomate

2 3 0

Give your thoughts!!

Mandar Parekh 27 days

Remote work tech, accelerated by COVID-19, demands investments in remote collaboration, cybersecurity, and employee well-being.

2 1 0

Give your thoughts!!

Mandar Parekh 27 days

Blockchain extends beyond cryptocurrencies, revolutionizing supply chains, voting systems, and art authentication.

2 1 0

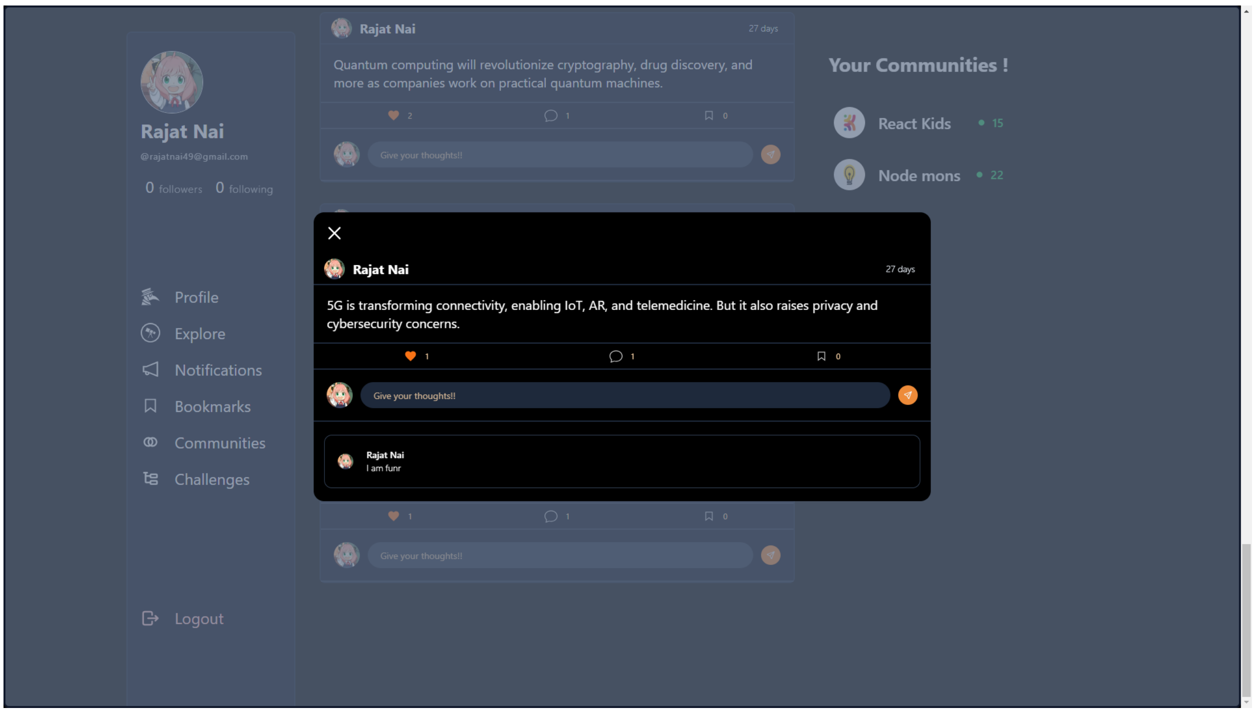
Give your thoughts!!

Rajat Nai 27 days

Your Communities !

React Kids ● 15

Node mons ● 22



Rajat Nai

@rajatnai49@gmail.com

0 followers 0 following

Profile
Explore
Notifications
Bookmarks
Communities
Challenges

Logout

Rajat Nai 27 days

Quantum computing will revolutionize cryptography, drug discovery, and more as companies work on practical quantum machines.

2 1 0

Give your thoughts!!

Rajat Nai 27 days

5G is transforming connectivity, enabling IoT, AR, and telemedicine. But it also raises privacy and cybersecurity concerns.

1 1 0

Give your thoughts!!

Rajat Nai
I am funr

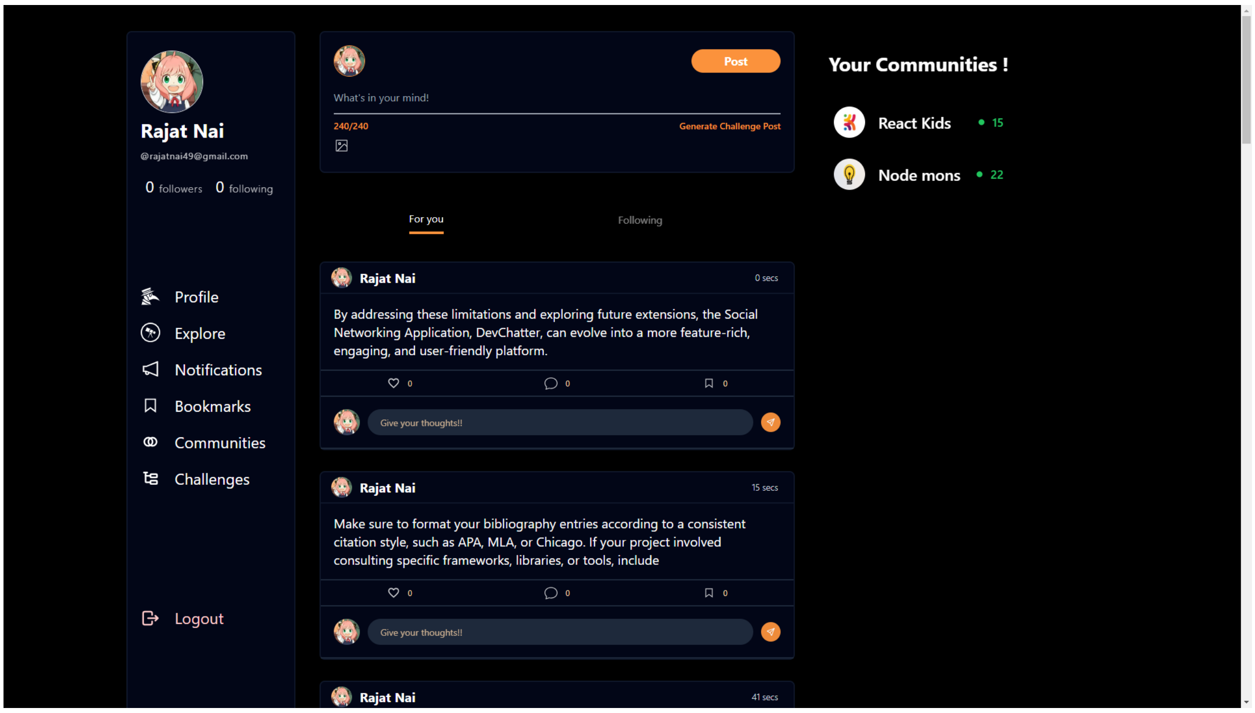
1 1 0

Give your thoughts!!

Your Communities !

React Kids ● 15

Node mons ● 22



Rajat Nai

@rajatnai49@gmail.com

0 followers 0 following

Profile
Explore
Notifications
Bookmarks
Communities
Challenges

Logout

Post

What's in your mind!

240/240 Generate Challenge Post

For you Following

Rajat Nai 0 secs

By addressing these limitations and exploring future extensions, the Social Networking Application, DevChatter, can evolve into a more feature-rich, engaging, and user-friendly platform.

0 0 0

Give your thoughts!!

Rajat Nai 15 secs

Make sure to format your bibliography entries according to a consistent citation style, such as APA, MLA, or Chicago. If your project involved consulting specific frameworks, libraries, or tools, include

0 0 0

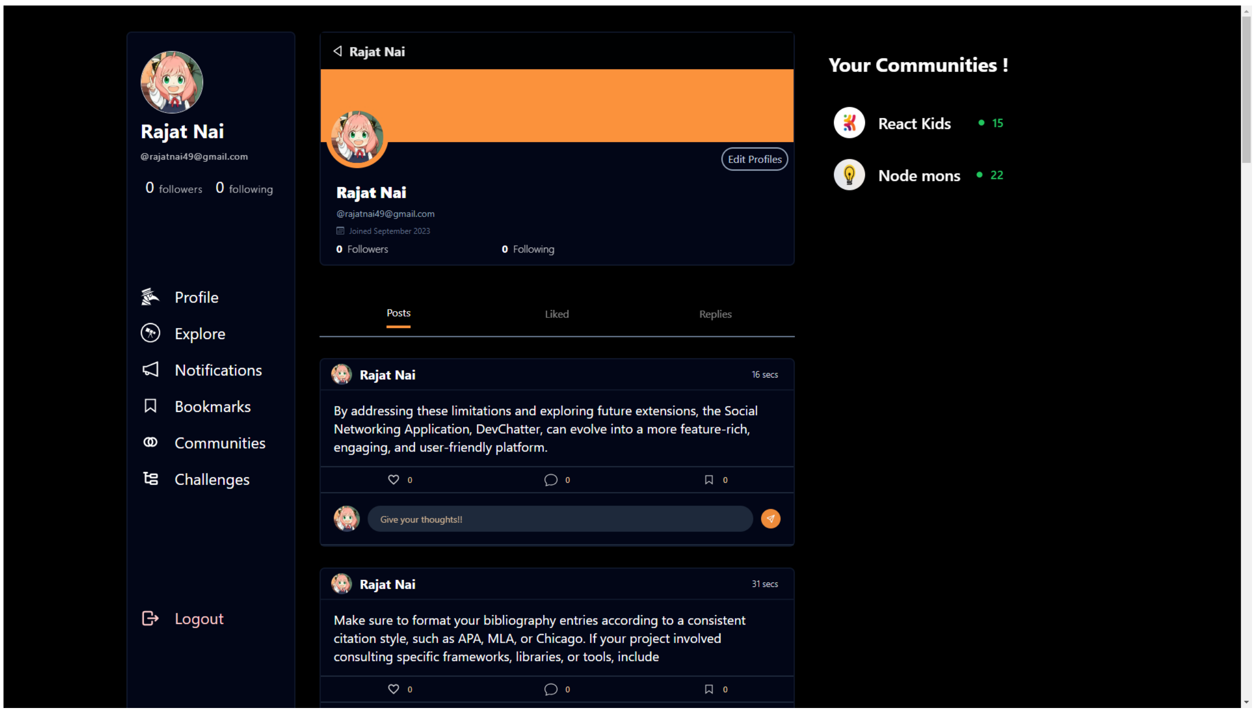
Give your thoughts!!

Rajat Nai 41 secs

Your Communities !

React Kids ● 15

Node mons ● 22



Rajat Nai
@rajatnai49@gmail.com
0 followers 0 following

Profile
Explore
Notifications
Bookmarks
Communities
Challenges
Logout

Rajat Nai

Edit Profiles

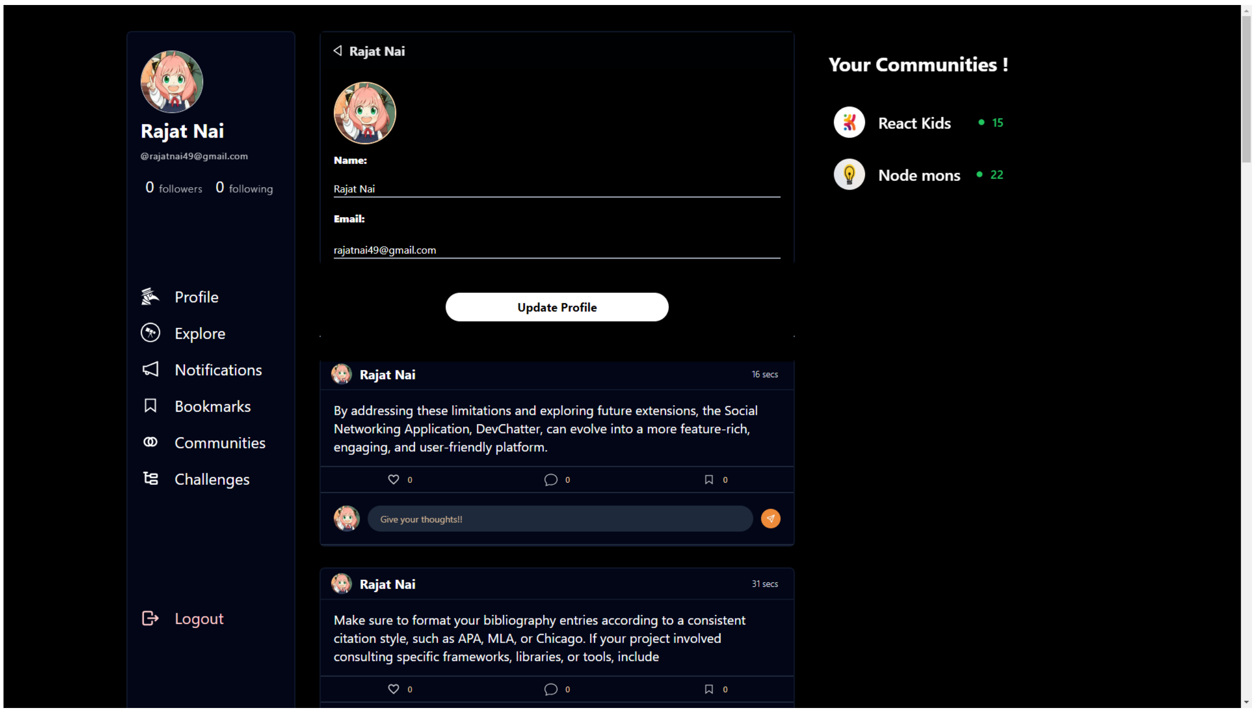
Rajat Nai
@rajatnai49@gmail.com
Joined September 2023
0 Followers 0 Following

Posts Liked Replies

Rajat Nai 16 secs
By addressing these limitations and exploring future extensions, the Social Networking Application, DevChatter, can evolve into a more feature-rich, engaging, and user-friendly platform.
0 0 0
Give your thoughts!!

Rajat Nai 31 secs
Make sure to format your bibliography entries according to a consistent citation style, such as APA, MLA, or Chicago. If your project involved consulting specific frameworks, libraries, or tools, include
0 0 0

Your Communities !
React Kids ● 15
Node mons ● 22



Rajat Nai
@rajatnai49@gmail.com
0 followers 0 following

Profile
Explore
Notifications
Bookmarks
Communities
Challenges
Logout

Rajat Nai

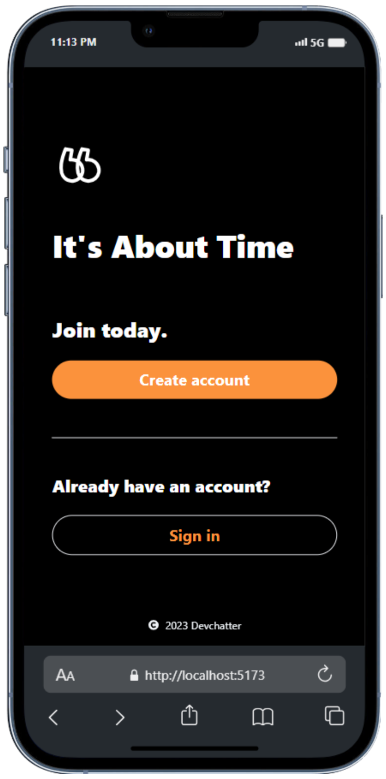
Name:
Rajat Nai
Email:
rajatnai49@gmail.com

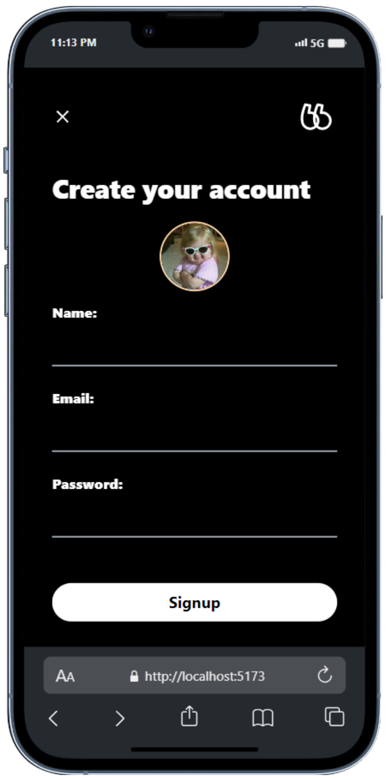
Update Profile

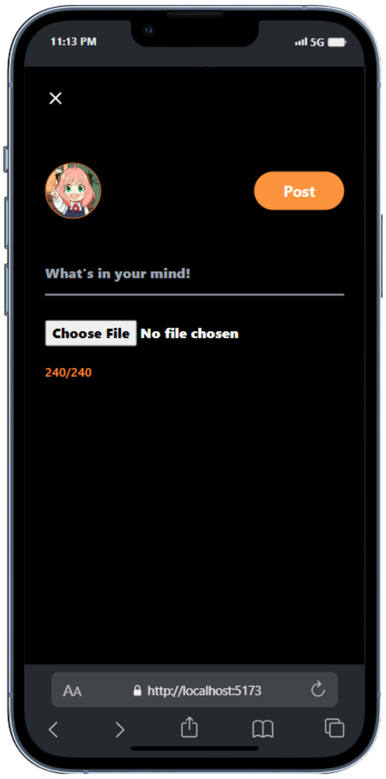
Rajat Nai 16 secs
By addressing these limitations and exploring future extensions, the Social Networking Application, DevChatter, can evolve into a more feature-rich, engaging, and user-friendly platform.
0 0 0
Give your thoughts!!

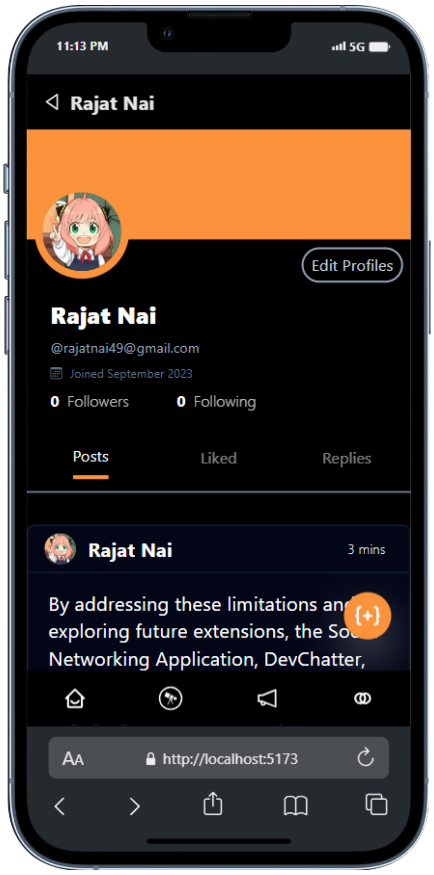
Rajat Nai 31 secs
Make sure to format your bibliography entries according to a consistent citation style, such as APA, MLA, or Chicago. If your project involved consulting specific frameworks, libraries, or tools, include
0 0 0

Your Communities !
React Kids ● 15
Node mons ● 22

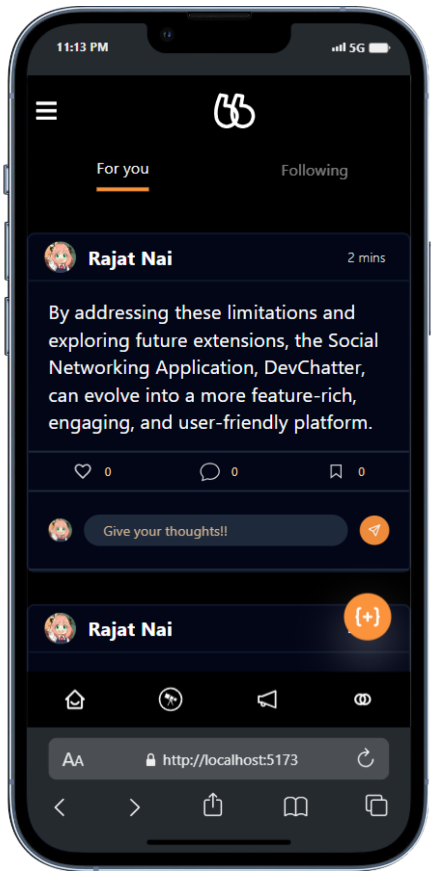








By addressing these limitations and exploring future extensions, the Social Networking Application, DevChatter,



By addressing these limitations and exploring future extensions, the Social Networking Application, DevChatter, can evolve into a more feature-rich, engaging, and user-friendly platform.

Conclusion

In the **backend** development, we have successfully implemented the following key functionalities:

User Authentication:
Users can register, log in, and log out securely.
Passwords are encrypted for security, and tokens are generated for authenticated sessions.

Password Reset:

Users can request a password reset, and a secure token-based mechanism is in place for updating passwords.

Post Management:

CRUD operations for posts are fully functional, allowing users to create, read, update, and delete their posts.
Users can like and unlike posts, and the count is accurately reflected.

User Interaction:

Follow and unfollow functionalities are implemented, updating the follower and following lists accordingly.
Users can view posts of the users they follow, creating a dynamic and interactive social media experience.

User Profile:

Users can update their profile details, including the profile picture.
Followers, following, and post counts are accurately maintained.

Comments:

Users can comment on posts, and comments are stored and displayed appropriately.
Reset Password:

Secure and user-friendly reset password functionality has been implemented.

On the **frontend**, we are in the process of completing the following functionalities:

Reset Password:

The frontend is being enhanced to provide a seamless and user-friendly experience for resetting passwords.

Image Display:

The frontend will be updated to appropriately display post images, enhancing the visual experience for users.

Limitation and Future Extension

Limitations:

Dynamic Feed Generation:

Currently, the system features do not explicitly mention the implementation of dynamic feed generation based on user preferences or algorithms. This could limit the diversity and personalization of the user's feed.
Learning Challenges and Chat Room:

As mentioned, features such as learning challenges (Section 4.7) and a chat room (Section 4.6) have not been implemented. These were part of the initial design but were not prioritized in the current development phase.
Testing Coverage:

The testing section hasn't been detailed in the provided information. A robust testing strategy is crucial for the reliability of the application.
Functionality Not Implemented:

Learning Challenges:

The functionality to participate in coding challenges, post challenge progress, and earn badges (Section 4.7) has not been implemented. This would have added a gamification element to the platform.
Chat Room:

The chat room functionality is also missing. Integration of real-time communication features could enhance user engagement.
Dynamic Feed Generation:

The system lacks a mechanism for dynamically generating user feeds based on preferences, interests, or algorithms.

Possible Future Extensions:

Enhanced Feed Algorithm:

Implement a more sophisticated algorithm for generating user feeds. This could consider user behavior, preferences, and trending topics to create a personalized and engaging feed.

Learning Challenges Integration:

Integrate the learning challenges feature, allowing users to participate in coding challenges, share their progress, and earn badges. This would encourage a sense of community and skill development.

Real-Time Chat:

Expand the chat room feature to include real-time messaging capabilities. Users could create public or private chat groups, fostering more immediate and interactive communication.

Notifications:

Implement a notification system to keep users informed about new followers, likes, comments, and other relevant activities on their posts.

User Analytics:

Introduce analytics tools to gather insights into user behavior. This data could be used to further refine algorithms, improve user experience, and provide valuable feedback for future updates.

Mobile Application:

Develop a mobile application version of the platform to cater to users who prefer mobile devices for social media interaction.

Accessibility Features:

Enhance accessibility features to ensure that the application is usable by individuals with diverse abilities. This could involve optimizing the user interface for screen readers and ensuring compatibility with accessibility standards.

Security Measures:

Strengthen security measures, including continuous monitoring for potential vulnerabilities, to safeguard user data and privacy.

Bibliography

1. Mozilla Developer Network. (2023). JavaScript Documentation. Link↗
2. React.js. (2023). React Documentation. Link↗
3. TaiwindCSS (2023) Link↗
4. Framer Motion (2023) Link↗
5. MongoDB (2023) Link↗
6. NodeJs v18.7.0 (2023) Link↗