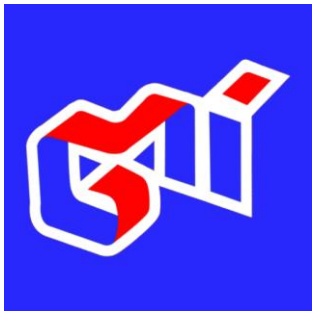


Feature-Rich Social Network Application Project Plan

Project Name	Version	Prepared By	Date
Developing a Real-Time Chat website	V 1.0	GMIndia	21-May-2024

1. **Objectives**

The objective of the project is to develop a comprehensive social networking platform by leveraging modern web technologies and best practices in software development. By focusing on user experience, scalability, security, and rich feature sets, the project seeks to deliver a robust and engaging social network that meets the needs of its users.

2. Scope

The scope of this project is to develop a feature-rich social network from scratch. The social network will use Django for the backend, Django REST Framework (DRF) for creating RESTful APIs, JWT authentication for secure user authentication, ReactJS for the frontend, and Tailwind CSS for responsive and modern styling. The platform aims to provide a robust, scalable, and interactive user experience, incorporating core social networking features such as user authentication, profile management, content creation, social interactions, and more.

3. Features

3.1 Core Features of the applications are to Develop core social network features and implement the user profiles, post creation and search functionality.

Core Features Development

Objectives:

- Develop the essential features of the social network.
- Implement functionalities for user profiles, post creation and search.

Actual Tasks:

User Profile Management:

- Develop views for user profiles.
- Enable editing of user profiles.

Post Creation and Display:

- Implement functionality for creating text posts.
- Display posts in the user's feed.

Search Functionality:

- Implement basic search functionality to find users and posts.

Enhanced Feed:

- Enhance the feed to include posts from friends and followed users.

Deliverables:

Fully functional core social network features, including:

- Post creation and display.
- User profile views and editing.

- Basic search functionality.
- Enhanced feed with posts from friends.

3.2 Additional Features

Friendships and Messaging Development

Objectives:

- Implement features for fostering friendships and facilitating messaging.
- Enhance user interactions and engagement within the social network.

Actual Tasks:

Friend Request Functionality:

- Implement functionalities for sending, accepting, and declining friend requests.

Messaging System:

- Create database models, serializers, and views for messaging.
- Develop frontend components for messaging, including list and detail views.

Real-Time Messaging (Optional for Future Sprints):

- Integrate WebSockets for real-time messaging capabilities.

Deliverables:

- Fully functional features for friendships and messaging, including:
- Friend request functionality.
- Messaging system with frontend components.
- Optional integration of real-time messaging using WebSockets.

3.2 Advanced Features and Enhancements

Objectives:

- Integrate advanced features such as image attachments, comments, and likes to enrich user interactions.
- Enhance the overall user experience within the social network platform.

Actual Tasks:

Image Upload and Display:

- Implement functionality for users to upload and display images in posts.

Commenting and Liking:

- Develop functionalities for users to comment on and like posts.

Personalized Feed:

- Enhance the feed algorithm to display personalised content based on user preferences and interactions.

"Who is Typing" Indicator:

- Add indicators to show when a user is typing in the messaging interface.

Deliverables:

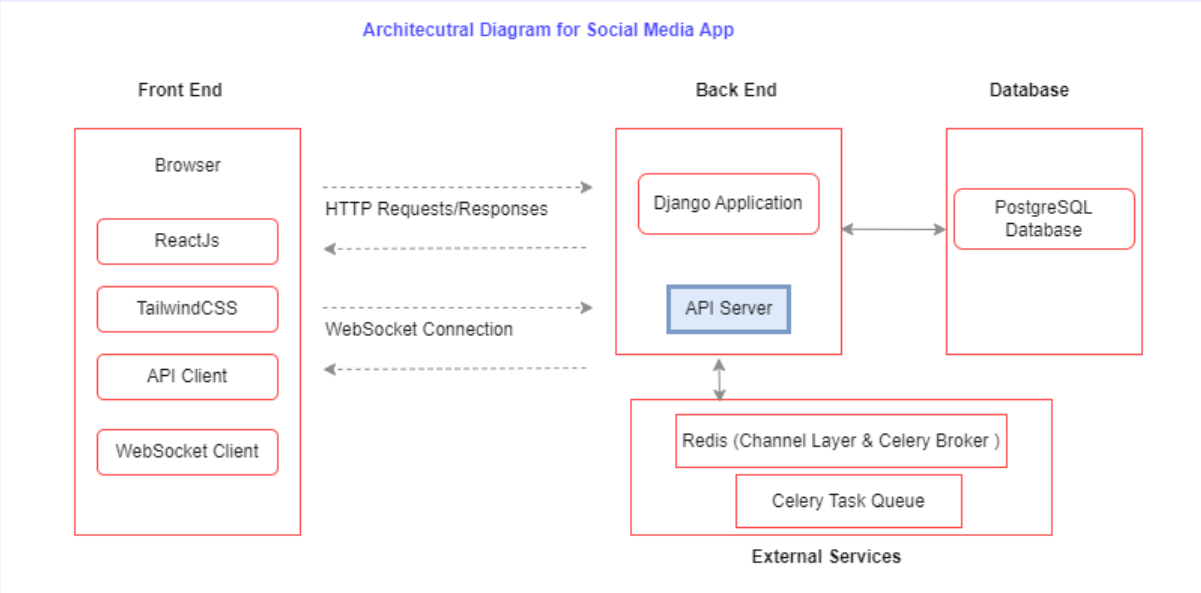
Advanced social network features, including:

- Image attachments for posts.
- Commenting and liking functionalities.
- Enhanced feed with personalised content.
- "Who is typing" indicator for messaging.

4. Architecture

4.1 System Architecture

The proposed design pattern is the Model-View-Template (MVT) software design pattern.



Components Overview

1. Frontend

- User Interface: Built with ReactJS and styled with Tailwind CSS.
- API Client: Axios or Fetch for making API requests.
- WebSocket Client: JavaScript code to handle real-time communication with the server.

2. Backend

- Django Application: Core logic of the application, handling HTTP requests and responses.
- Django REST Framework: For building and consuming RESTful APIs.
- Database: PostgreSQL for data storage.
- Redis: As a message broker for real-time notifications and WebSocket.

- Celery: For handling asynchronous tasks such as email notifications and background processing.

3. Authentication

- JWT (JSON Web Tokens): For user authentication and authorization.

4. WebSocket Protocol

- Enables real-time, two-way interaction between the client and the server.

Components:

- **Django Framework (V 3.10/3.15):**
  Django will handle routing, views, and user authentication.

- **Django Channels:**
  Django Channels will be used for handling WebSocket connections, enabling real-time bidirectional communication between clients and the server.

- **Channels Consumers:**
  Consumers will be implemented to handle WebSocket events such as connecting, disconnecting, sending, and receiving messages.

- **Database Models:**
  Django ORM will define database models for storing user profiles, messages, chat rooms, and other relevant data.

- **Authentication:**
  Django's built-in authentication system or third-party libraries (e.g., Django Rest Framework JWT) will be used for user authentication and authorization.

- **Business Logic:**
  Business logic will handle message formatting, notifications, file uploads, and other application-specific functionalities.

- **Asynchronous Tasks:**
  Asynchronous tasks may be implemented using Django Channels or Celery for handling background tasks such as message delivery and notifications.

- **WebSocket Server:**
  Django Channels runs an ASGI (Asynchronous Server Gateway Interface) server for handling WebSocket connections. This server will manage WebSocket connections, routing messages between clients, and handling real-time events.

Environment Setup:

- **Development Environment:** Set up a Dev environment and ensure developers have access to a development environment with the necessary tools, libraries, and dependencies installed.
- **Staging Environment:** Set up a staging environment to mirror the production environment for testing and validation purposes.
- **Production Environment:** Prepare the production environment with the required infrastructure, including servers, databases, and networking configurations.

5. Development Process

5.1 Methodology

The Agile development methodology will be followed for the entire development.

The project will have typical stakeholders like Product Owner, Scrum Master, Development and Testing Team.

5.2 Milestones

- **Initiation:** Project environment setup, requirements gathering, and initial planning.
- **Development:** Actual coding, designing, and building of the software.
- **Testing:** Quality assurance, bug fixing, and user acceptance testing.
- **Deployment:** Release preparation, deployment planning, and final rollout.

6. Resource Allocation

6.1 Team

This project involves Backend, Front End, UI/UX and QA Teams.

- Back End - 3 FTE
- Front End - 2 FTE
- UI/. UX - 1 FTE
- QA - 2 FTE

7. Risks and Mitigation

7.1 Risk Assessment

Here are some potential risks to consider:

- **WebSocket Stability:** WebSocket connections may face stability issues due to network fluctuations, firewalls, or proxy restrictions, leading to intermittent connectivity problems for users.

- **Performance Bottlenecks:** Inefficient code, database queries, or WebSocket handling may lead to performance bottlenecks, causing delays in message delivery and degraded user experience, particularly under high load conditions.

- **Compatibility Issues:** Compatibility issues may arise across different browsers, devices, and operating systems, affecting the usability and functionality of the application for certain users.

- **User Experience Challenges:** Design and usability issues, such as complex navigation, inconsistent UI elements, or unclear feedback mechanisms, may lead to poor user experience and reduced user engagement.

- **Change Management:** Changes in project requirements, technology stack, or stakeholder expectations may disrupt the development process and require adjustments in planning, resources, and timelines.

7.2 Mitigation Strategies

- It's essential to conduct thorough risk assessments,
- Implement robust security measures, adhere to best practices in coding and testing, continuously monitor and optimise performance
- Maintain clear communication and collaboration within the project team and stakeholders.
- Additionally, regular reviews, testing, and contingency planning can help identify and address potential risks throughout the project lifecycle.

8. Quality Assurance

8.1 Testing Plan

Planned to perform the below mentioned testing approaches;

1. Unit testing - Developers have to perform unit testing.
2. Functional Testing
3. Integration Testing
4. Usability Testing
5. Performance Testing
6. Compatibility Testing
7. Regression Testing
8. Test Case Documentation
9. Bug Tracking and Reporting
10. User Acceptance Testing (UAT)

9. Deployment

9.1 Deployment Plan

We maintain the various development environments and deployment plans like

- Version Control
- Continuous Integration/Continuous Deployment (CI/CD) Pipeline
- Deployment Strategy
- Database Migration
- Monitoring and Alerts
- Rollback Plan
- Post-Deployment Validation:

10. Conclusion

This project will utilize Django, Django REST Framework, JWT authentication, ReactJS, and Tailwind CSS to develop a comprehensive social networking platform. The goal is to create a responsive, user-friendly, and secure application that supports rich social interactions and content sharing. The application will be designed to scale and provide a seamless user experience across different devices.