**FeedBuzz - Social Media Platform**

Requirements Document

1. Introduction

FeedBuzz is a web-based social media platform aimed at connecting people and facilitating content sharing and engagement. The platform allows users to create profiles, share posts, follow other users, and interact with content through likes, comments, and shares. FeedBuzz aims to provide a user-friendly and engaging experience for individuals to connect, discover, and share content with their network.

2. User Roles

FeedBuzz supports the following user roles:

Regular Users: Individuals who sign up for an account and use the platform to create and interact with content.

3. Functional Requirements

3.1 User Registration and Authentication

* Users can register for an account by providing basic information such as name, email, and password.
* Upon registration, users can authenticate themselves using their credentials to access the platform.

3.2 Content Creation and Sharing

* Users can create and share various types of content, including text posts, images, videos, and links.
* Users can have the privacy settings for their posts, visible to specific groups or followers.
* Users can delete their own posts.

3.3 Content Discovery and Interaction

* Users can view a personalized feed that displays posts from users they follow.
* Users can like, comment on, and share posts to engage with the content and interact with other users.
* Users can follow or unfollow other users to customize their feed and receive updates from their preferred connections.

3.4 User Connections and Messaging

* Users can send connection requests to other users to establish a mutual connection.
* Users can send direct messages to their connections for private conversations.

3.5 Real-Time Chat

* Users can engage in real-time chat conversations with their connections.
* Users can send text messages, emojis, and attachments in chat conversations.

4. Non-Functional Requirements

4.1 Performance

* The platform should be responsive and scalable to handle concurrent user interactions and content updates.
* Page load times should be optimized to provide a seamless user experience.

4.2 Security

* User data should be encrypted during transmission and storage to ensure privacy and security.
* Authentication mechanisms should be robust to prevent unauthorized access to user accounts.
* Content moderation tools should be in place to detect and prevent spam, fake accounts, and abusive behavior.

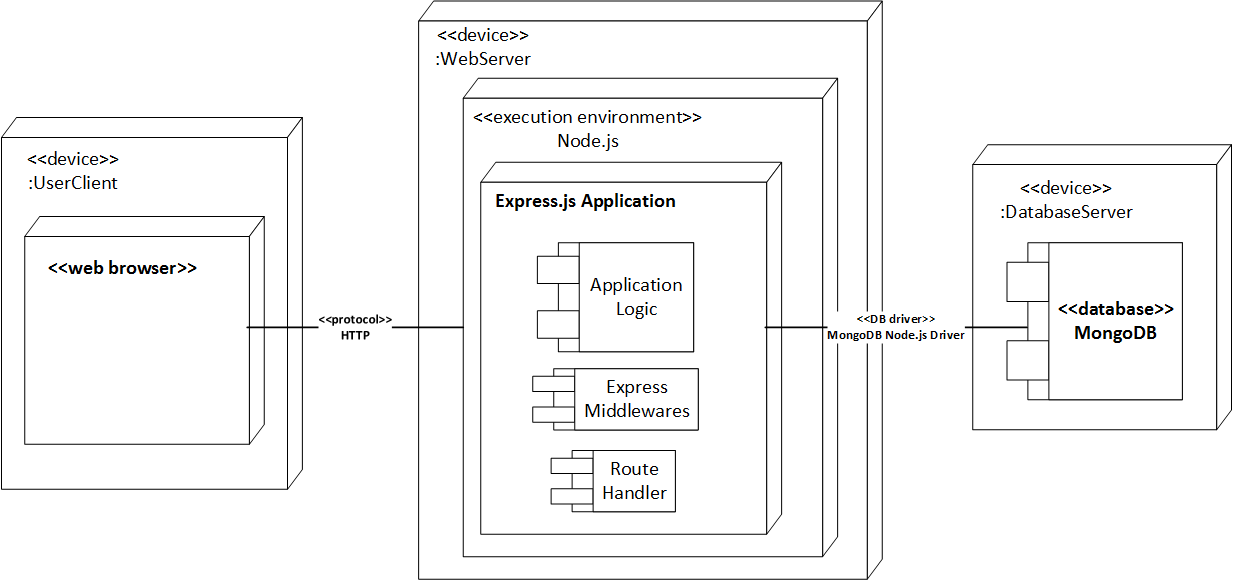
4.3 Usability

* The user interface should be intuitive and user-friendly, with clear navigation and informative feedback messages.
* Accessibility standards should be followed to ensure that the platform is usable by individuals with disabilities.

4.4 Compatibility

* The platform should be compatible with a wide range of devices and web browsers to accommodate diverse user preferences.

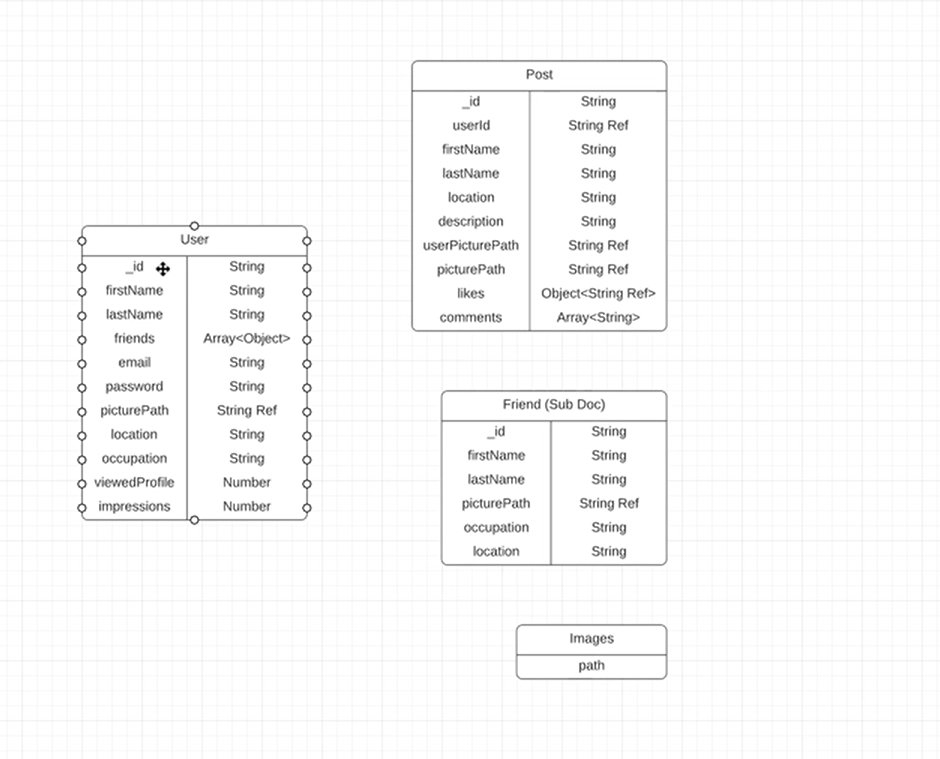
Design Document



1. Architecture Overview

FeedBuzz follows a modular architecture based on the MERN (MongoDB, Express.js, React.js, Node.js) stack:

1. Frontend: Developed using React.js, the frontend interacts with the backend APIs to fetch data and render the user interface.
2. Backend: Built using Express.js and Node.js, the backend provides RESTful APIs for user authentication, content management, user connections, notifications, and real-time chat functionality.
3. Database: MongoDB is used to store user data, posts, connections, chat messages, and other relevant information.



2. Database Design

The MongoDB database consists of the following collections:

* Users: Stores user information including name, email, password hash, profile picture, and bio.
* Posts: Contains post data including content, author, timestamps, privacy settings, and engagement metrics.
* Connections: Stores information about user connections, including follower and following relationships.
* Chat Messages: Stores chat messages between users, including sender, receiver, timestamps, and message content.

3. API Design

FeedBuzz backend provides the following RESTful APIs:

Authentication APIs:

/api/auth/register: Register a new user.

/api/auth/login: Authenticate user credentials and generate a JWT token.

/api/auth/user: Get current user details.

User Profile APIs:

* /api/profile/:userId: Get user profile information by user ID.
* /api/profile/:userId/edit: Update user profile information.

Post APIs:

* /api/posts: Get all posts or create a new post.
* /api/posts/:postId: Get, update, or delete a specific post by post ID.
* /api/posts/getallpost: Get the personalized feed for the current user.

Connection APIs:

* /api/removeoraddFriend/:userId/follow: Follow or unfollow a user by user ID.

Chat APIs:

* /api/chat/messages/:userId: Get chat messages between the current user and another user by user ID.
* /api/chat/messages/:userId/send: Send a chat message to another user by user ID.

4. Frontend Design

The frontend is designed using React.js and styled with CSS or CSS frameworks such as Bootstrap. It consists of the following main components:

* Navigation Bar: Provides navigation links to different sections of the platform, including the feed, profile, connections, and chat.
* User Profile: Displays user information, including profile picture and bio.
* Feed: Displays a personalized feed of posts from users the current user follows.
* Post Creation: Allows users to create and share new posts, including text, images, videos, and links.
* Post Interaction: Enables users to like, comment on, and share posts, as well as bookmark or save posts for later reference.
* Connections: Displays user connections, including followers and following, and allows users to manage their connections.
* Chat: Provides a real-time chat interface for users to engage in conversations with their connections.

5. Deployment

FeedBuzz can be deployed on a cloud platform such as AWS, Google Cloud Platform,github or Microsoft Azure. The frontend can be hosted using services like Amazon S3 or Netlify, while the backend can be deployed on platforms supporting Node.js applications like AWS Elastic Beanstalk or Heroku. MongoDB Atlas can be used for database hosting.

6. Security

FeedBuzz should implement security best practices, including encryption of sensitive data, input validation, protection against common web vulnerabilities such as cross-site scripting (XSS) and secure authentication mechanisms using JWT tokens.

7. Maintenance and Support

Regular maintenance and updates should be performed to keep the platform up-to-date with the latest technologies and security patches. A support system should be in place to address user inquiries, bug reports, and feature requests in a timely manner.

8. Conclusion

FeedBuzz is designed to provide a user-friendly and engaging social media platform for content sharing, interaction, and real-time chat. By following the outlined design principles and implementing the suggested features, FeedBuzz aims to create a vibrant and connected community where users can discover, share, and engage with content in a meaningful way.