Twibbles Technical Design Document *Providing a Dynamic and Engaging Platform for Microblogging and Social Interactions*

Author: Harshit Behal
Date: 24 - December - 2024

Preface

This Technical Document outlines the architecture, features, and implementation strategies for **Twibbles**, a MERN Stack application designed as a social media platform. It serves as a blueprint for the development process, ensuring alignment among team members and stakeholders.

Scope

Twibbles is a microblogging platform that allows users to share thoughts, interact with others, and explore trending topics. The system will include features such as user registration, post creation, following and follower management, likes and retweets, notifications, and an admin panel for platform moderation.

Document Structure

This document is organized into the following sections:

1. **System Overview**: High-level description of the system architecture and deployment strategy.
2. **Requirements**: Detailed breakdown of functional and non-functional requirements.
3. **Technical Design**: Database schemas, API endpoints, and UI/UX components.
4. **Technology Stack**: Overview of the technologies used in Twibbles.
5. **Deployment**: Strategies for deploying and maintaining the application.
6. **Future Enhancements**: Planned features and scalability considerations.

7. **Appendix**: References, additional notes, and technical diagrams.

TABLE OF CONTENTS

Introduction

Project Overview

Twibbles is a MERN stack-based microblogging platform where users can post short messages ("twibbles"), follow others, and engage with content via likes, comments. The platform emphasizes real-time interactions, a user-friendly interface, and scalable architecture to handle high traffic efficiently.

Objectives

- Provide a seamless platform for real-time microblogging.
- Encourage user engagement through interactive features like likes and comments.
- Deliver a user-friendly and accessible experience on both web and mobile platforms.
- Ensure secure and scalable operations for growing user bases.

Target Audience

The primary audience for **Twibbles** includes:

- **Casual Users**: Individuals looking to share and explore content in a social setting.
- **Content Creators**: Users aiming to grow their audience and share engaging content.
- **Businesses and Brands**: Companies promoting their services and connecting with audiences.

System Overview

Architecture

Twibbles utilizes a MERN stack architecture:

- **MongoDB**: Stores user data, posts, and activity logs.
- **Express.js**: Backend framework handling API requests and business logic.
- **React.js**: Frontend framework delivering a dynamic and responsive user interface.
- **Node.js**: Runtime environment for server-side operations.

This architecture supports full-stack JavaScript development, ensuring efficiency and a cohesive development process.

Core Components

Frontend (React.js)

- **Dynamic and Responsive Interface**: Built using React's component-based architecture for a seamless experience.
- **Routing**: React Router enables Single Page Application (SPA) behavior for smooth navigation.
- **Styling**: Tailwind CSS provides a clean and modern UI.

Backend (Node.js and Express.js)

- **RESTful APIs**: Endpoints for user authentication, post management, and notifications.
- **Middleware**: Implements request validation, rate limiting, and error handling.
- **Authentication and Authorization**: Uses JSON Web Tokens (JWT) for secure user sessions and role-based access control.

Database (MongoDB)

- **Flexible Schema**: Ideal for storing user data, posts, comments, and follower relationships.
- **Indexes**: Optimized for fast search and retrieval operations.
- **Relationships**: Handles user-following relationships and embedded comments.

Features

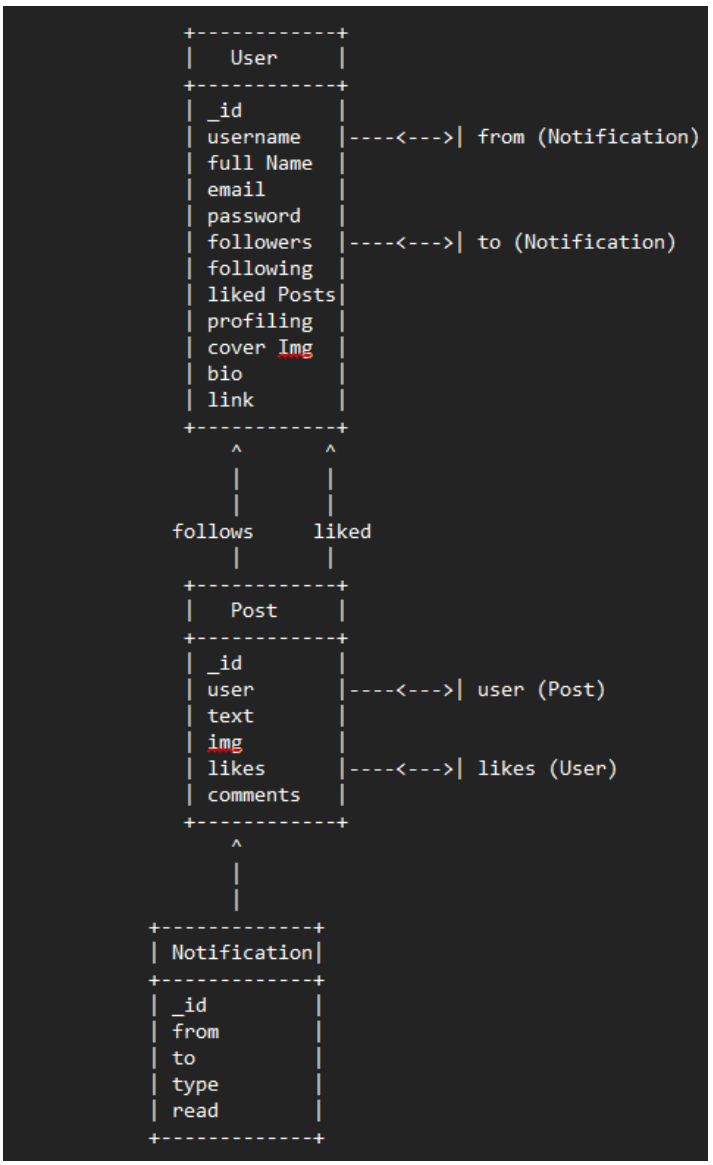
- **User Authentication**: Sign-up, login, logout.
- **Post Creation**: Users can create and delete posts up to 280 characters and upload one image up to 5MB.
- **Likes and Retweets**: Interactive features to engage with posts.
- **Follow/Unfollow**: Manage connections between users.
- **Notifications**: Notify users of new followers, likes.
- **Scalability**: Scale seamlessly to accommodate user growth.
- **Security**:
    - **JWT Tokens for Authentication**: Securely generated and signed with a secret key to authenticate users. Tokens are stored in cookies with `httpOnly`, `secure`, and `sameSite: 'strict'` flags to prevent XSS and CSRF attacks. Tokens are set to expire after a specified duration to mitigate misuse.
    - **Password Security**: User passwords are securely hashed using bcrypt before being stored in the database. Plain-text passwords are never stored.

    - **Secure Cookie Storage**: JWT tokens are stored in cookies with secure flags, ensuring they cannot be accessed by client-side JavaScript or intercepted during transmission over HTTPS.
    - **Token Validation**: Backend API routes are accessible only with a valid JWT token. Invalid or expired tokens are rejected with a `401 Unauthorized` response.
    - **Error Handling**: Errors are handled gracefully, and sensitive data is not exposed in error messages. Unauthorized users are given appropriate `401 Unauthorized` or `404 Not Found` responses.
- **Accessibility**: Ensure the platform is usable on various devices.

Technical Design

Database Schema

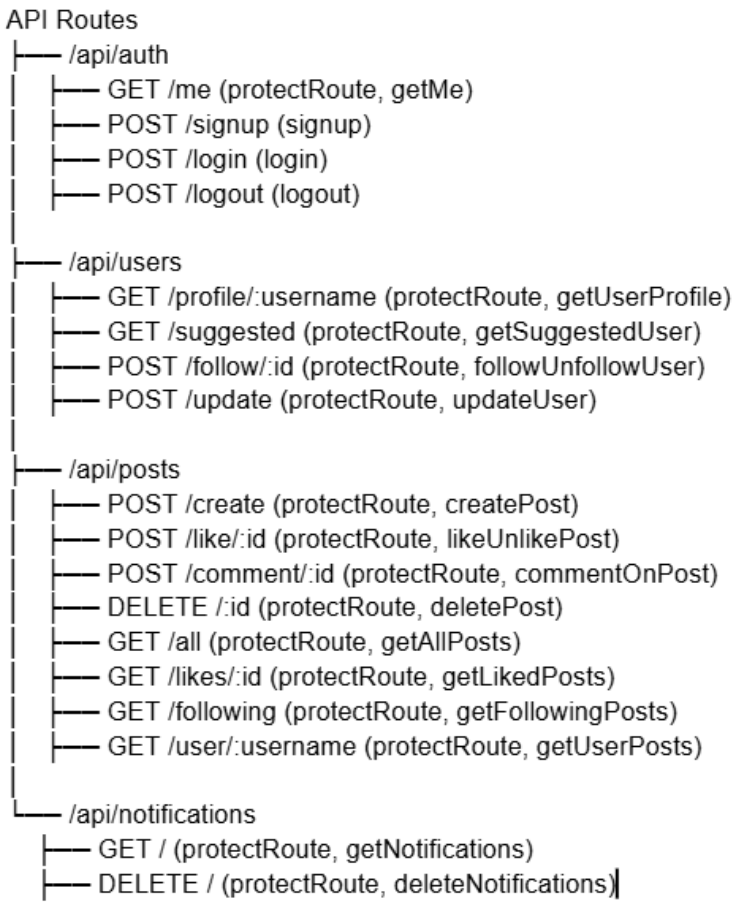
- **User Collection**: Stores user details (username, email, password, bio, avatar).
- **Post Collection**: Stores posts, likes and comments.
- **Follower Collection**: Manages user-follow relationships.
- **Notification Collection**: Tracks user notifications.



API Endpoints

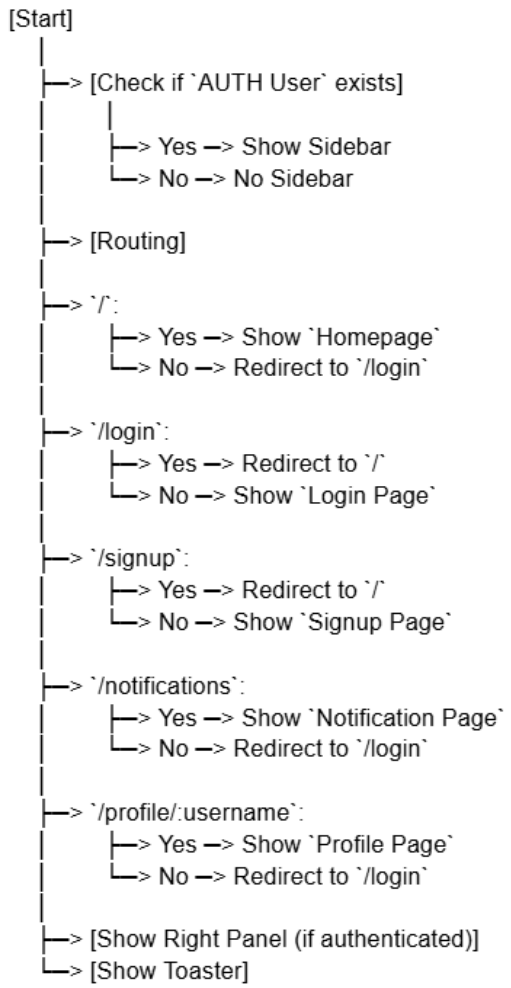
- `GET /profile/:username`: Fetches the profile details of a user by their username.
- `GET /suggested`: Retrieves suggested users to follow.
- `POST /follow/:id`: Follows or unfollows a user based on the provided ID.
- `POST /update`: Updates the logged-in user's profile information.
- `POST /create`: Creates a new post.
- `POST /like/:id`: Likes or unlikes a post based on the provided post ID.
- `GET /`: Retrieves notifications for the logged-in user.
- `DELETE /`: Deletes all notifications for the logged-in user.

- `GET /me`: Retrieves the profile of the logged-in user.
- `POST /signup`: Registers a new user.
- `POST /login`: Authenticates and logs in a user.
- `POST /logout`: Logs out the currently authenticated user.
- `POST /comment/:id`: Adds a comment to a post by its ID.
- `DELETE /:id`: Deletes a post by its ID.
- `GET /all`: Retrieves all posts.
- `GET /likes/:id`: Retrieves posts liked by a user based on post ID.
- `GET /following`: Retrieves posts from users the logged-in user is following.
- `GET /user/:username`: Retrieves posts by a specific user based on their username.



UI/UX Components

- **Homepage**: Displays a feed of posts from all users, including those followed by the logged-in user, and an interface to create a new post.
- **User Profile**: Showcases a user's profile details along with their posts, followers, following count, and liked posts.
- **Notification**: Displays notifications for the logged-in user (e.g., new follows, likes etc.).
- **Login Page**: Allows users to authenticate and log in using their credentials.
- **Signup Page**: Provides an interface for users to register a new account.
- **SideBar Panel**: Contains navigational links to Home page, Notification page and Profile page.
- **RightSide Panel**: Displays Recommended users to follow



Technology Stack

- **Frontend**: React.js, Tailwind CSS, daisyUi.
- **Backend**: Node.js, Express.js.
- **Database**: MongoDB.
- **Authentication**: JSON Web Tokens (JWT).
- **Hosting**: Render.

Deployment

1. **Frontend Deployment**: Hosted on Render
2. **Backend Deployment**: Deployed on Render.
3. **Database Hosting**: MongoDB Atlas for cloud-based storage.
4. **Version Control**: GitHub for collaboration and code management.

Future Enhancements

1. **Direct Messaging**:
    - Private, real-time conversations between users.
    - Supports text, images, and emojis with encryption for security.
    - Powered by WebSockets for instant messaging.
2. **Advanced Analytics**:
    - Insights for users and brands, tracking post performance, engagement, and demographics.
    - Visualized through interactive dashboards using React.js.
3. **Trending Hashtags**:
    - Dynamically displays popular hashtags based on real-time activity.
    - Allows users to explore posts related to trending topics.
4. **Multimedia Support**:
    - Supports video and GIF uploads alongside images.
    - Optimized for smooth playback and easy sharing of multimedia content.
5. **Mobile App**:
    - Native apps for Android and iOS with full platform functionalities.
    - Push notifications and native media upload support via React Native.

Appendix

1. **References**:
    - **MongoDB, Node.js, React.js, Tailwind CSS**: Official documentation will guide the database, backend, frontend, and design implementations.
2. **Diagrams**:
    - **System Architecture & API Flowcharts**: Illustrate platform components and API interactions for clear development planning.
3. **Additional Notes**:
    - **Scalability**: Design ensures the platform can handle growth, with horizontal scaling and caching.
    - **Real-Time Updates**: WebSockets for live interactions and notifications.