**Full Stack Development with MERN**

**API Development and Integration Report**

|  |  |
| --- | --- |
| Date | 13 April, 2025 |
| Team ID | SWTID1743172790 |
| Project Name | Social Media Web App |
| Maximum Marks |  |

**Project Title:** Social Media Web Application  
**Date:** 10/04/25  
**Prepared by:** Vanshika Jain, Komal Baid, Shreya Sania, Shreeya Joshi

**Objective**  
The objective of this report is to document the API development progress and core backend service implementation for the **Social Media Web App Project**, focusing on features like user interaction, post management, real-time messaging, and profile functionalities.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** JSON Web Token (JWT)
* **Real-Time Communication:** Socket.IO

**Key Directories and Files**

1. **/controllers**
   * Contains logic for handling request and response lifecycles.
   * Each controller corresponds to a specific resource (e.g., userController, postController, messageController).
   * Integrated in server.js
2. **/models**
   * Contains Mongoose schemas representing MongoDB collections.
   * Key schemas:
     + UserSchema: User data and credentials.
     + PostSchema: Captions, images, likes, etc.
     + MessageSchema: Chat messages and sender/receiver references.
     + CommentSchema: Post comments with timestamps.
     + StorySchema: Temporary 24-hour stories.
3. **/routes**
   * Defines all RESTful API endpoints.
   * Connects the frontend to backend logic through controller functions.
   * Includes:
     + userRoutes.js
     + postRoutes.js
     + messageRoutes.js
     + commentRoutes.js
4. **/middlewares**
   * Houses custom middleware for handling requests and responses.
   * Includes:
     + **Auth Middleware**: Verifies JWT tokens.
     + **CORS Handling**: Enables cross-origin access.
5. **/config**
   * Database configuration and environment variables.
   * Contains MongoDB URI and secret keys for JWT and other services.

**API Endpoints**  
A summary of the main API endpoints and their purposes:

**User Authentication**

* **POST /api/register –** Register a new user (checks duplicates, hashes password).
* **POST /api/login –** Logs in the user and returns a secure JWT.

**User Management**

* **GET /api/user/:id –** Fetch user profile details.
* **PUT /api/user/:id –** Update profile details (username, bio, profile picture).
* **GET /api/user/search/:query –** Search users by username**.**

**Posts**

* **POST /api/posts –** Create a new post with caption and image.
* **GET /api/posts/:id –** Get a specific post by ID.
* **GET /api/posts/user/:userId –** Get all posts by a user.
* **PUT /api/posts/:id –** Edit a post.
* **DELETE /api/posts/:id –** Delete a post.
* **PUT /api/posts/:id/like –** Like/unlike a post.

**Comments**

* **POST /api/comments –** Add a comment to a post.
* **GET /api/comments/:postId –** View all comments on a post.

**Real-time Messaging**

* **POST /api/messages –** Send a message.
* **GET /api/messages/:conversationId –** Get all messages in a conversation.

**Stories**

* **POST /api/stories –** Upload a new story (auto-deletes after 24 hours).
* **GET /api/stories/:userId –** Get active stories of a user.

**Integration with Frontend**

The backend integrates with the React frontend using RESTful APIs and WebSocket (Socket.IO) for real-time messaging.  
Key integrations:

* **JWT Authentication**: Frontend stores and sends tokens with protected requests.
* **Feed & Explore**: API calls populate feeds and explore suggestions.
* **Chat UI**: Socket.IO handles live messages and typing indicators.

**Error Handling and Validation**

* **Centralized Error Handling**: Middleware captures and responds with appropriate error messages.
* **Validation**: All user inputs are validated (e.g., empty fields, image format, email structure).
* **Try-Catch Blocks**: Each controller function is wrapped with try-catch for graceful error logging.

**Security Considerations**

* **JWT Authentication**: All protected routes are token-secured.
* **Password Hashing**: User passwords are hashed using **bcrypt**.