**Backend Documentation**

**Overview**

This documentation describes the backend architecture of a user authentication and article management system. It includes information about the MongoDB schemas, the API routes, and their corresponding controllers.

**Table of Contents**

1. Schemas
   * Article Schema
   * Comment Schema
   * User Schema
2. Routes
   * User Routes
   * Article Routes
3. Controllers
   * User Controller
   * Article Controller
4. Usage
5. Dependencies

**Schemas**

**Article Schema**

**File: models/Article.js**

Defines the structure of an article document.

* title: The title of the article (String, required).
* author: The author of the article (String).
* content: The content of the article (String, required).
* genre: The genre of the article (String, required).
* comments: Array of comment references.
* likes: Number of likes (default: 0).
* dislikes: Number of dislikes (default: 0).
* liked\_by: Array of user IDs who liked the article.
* disliked\_by: Array of user IDs who disliked the article.
* Timestamps for creation and update.

**Comment Schema**

**File: models/Comment.js**

Defines the structure of a comment document.

* author: The author of the comment (String).
* comment: The content of the comment (String).
* articleId: Reference to the associated article (ObjectId).
* Timestamps for creation and update.

**User Schema**

**File: models/User.js**

Defines the structure of a user document.

* username: Unique username (String, required).
* email: Unique email address (String, required).
* password: Hashed password (String, required).
* friends: Array of user references for friends.
* pending\_friends: Array of user references for pending friend requests.
* Timestamps for creation and update.

**Routes**

**User Routes**

**File: routes/router.js**

Handles user-related operations.

* POST /createUser - Create a new user.
* POST /signIn - Sign in an existing user.
* POST /contact\_us - Contact us form submission.
* GET /getUser - Get user details.
* GET /getAllusers - Get all users.
* POST /:id/addFriend - Send a friend request.
* DELETE /:id/CanceladdFriend - Cancel a friend request.
* POST /:id/acceptFriend - Accept a friend request.
* DELETE /:id/rejectFriend - Reject a friend request.
* DELETE /:id/removeFriend - Remove a friend.
* GET /:id/getFriendsNumber - get number of friends.

**Article Routes**

**File: routes/router.js**

Handles article-related operations.

* GET /getArticlesSortedByTime - Get all articles sorted by time.
* GET /getMyArticlesSortedByTime - Get all user's articles sorted by time.
* GET /getMyFriendsArticles - Get all friends' articles.
* GET /getArticlesSortedByLikes - Get all articles sorted by likes.
* GET /getArticlesSortedByGenre - Get all articles sorted by genre.
* GET /getComments - Get all comments.
* POST /createArticle - Create a new article.
* POST /:id/createComment - Create a new comment on an article.
* POST /:id/likeArticle - Like an article.
* POST /:id/dislikeArticle - Dislike an article.
* GET /:id/getArticle - Get a single article by ID.
* DELETE /:id/deleteArticle - Delete an article.
* DELETE /:id/deleteComment - Delete a comment.
* PATCH /:id/updateArticle - Update an article.

**Controllers**

**User Controller**

**File: controllers/userController.js**

Handles the logic for user-related routes.

* createUser - Handles creating a new user.
* signIn - Handles user sign-in.
* contact\_us - Handles contact us form submissions.
* getUser - Retrieves user details.
* getAllusers - Retrieves all users.
* addFriend - Handles sending a friend request.
* CanceladdFriend - Handles canceling a friend request.
* acceptFriend - Handles accepting a friend request.
* rejectFriend - Handles rejecting a friend request.
* removeFriend - Handles removing a friend.
* getFriendsNumber - Handles getting number of friends.

**Article Controller**

**File: controllers/articleController.js**

Handles the logic for article-related routes.

* createArticle - Handles creating a new article.
* getArticle - Retrieves a single article by ID.
* getArticlesSortedByTime - Retrieves all articles sorted by time.
* getMyArticlesSortedByTime - Retrieves all user's articles sorted by time.
* getMyFriendsArticles - Retrieves all friends' articles.
* getArticlesSortedByLikes - Retrieves all articles sorted by likes.
* getArticlesSortedByGenre - Retrieves all articles sorted by genre.
* getComments - Retrieves all comments.
* createComment - Handles creating a new comment on an article.
* likeArticle - Handles liking an article.
* dislikeArticle - Handles disliking an article.
* deleteArticle - Handles deleting an article.
* deleteComment - Handles deleting a comment.
* updateArticle - Handles updating an article.

**Usage**

1. Clone the repository:

git clone <repository-url>

cd <repository-directory>

1. Install dependencies:

npm install

1. Set up environment variables: Create a .env file in the root directory and add the following variables:

MONGO\_URI=<your-mongodb-uri>

JWT\_SECRET=<your-jwt-secret>

1. Run the server:

npm start

**Dependencies**

* express - Web framework for Node.js.
* mongoose - MongoDB object modeling tool.
* bcrypt - Library for hashing passwords.
* jsonwebtoken - Library for creating and verifying JSON Web Tokens.
* nodemailer - Library for sending emails.