**Summary of Generative AI & Chatbot Project**

**Khanh Nguyen Tran**

Contents

[I. Back-end 3](#_Toc178636744)

[1. User authentication 3](#_Toc178636745)

[2. Blog Management 3](#_Toc178636746)

[3. Character Management 4](#_Toc178636747)

[4. Chatbot Interaction 4](#_Toc178636748)

[5. Middleware and Authentication 4](#_Toc178636749)

[6. Error Handling 5](#_Toc178636750)

[7. Database connection 5](#_Toc178636751)

[II. Front-end 5](#_Toc178636752)

[1. App structure and routing: 5](#_Toc178636753)

[2. Styling and UI 6](#_Toc178636754)

[3. Admin Dashboard and Character Management 6](#_Toc178636755)

[4. Blog Forum Functionality 6](#_Toc178636756)

[5. Character Exploration 7](#_Toc178636757)

[6. Chatbot 7](#_Toc178636758)

[7. User Authentication 7](#_Toc178636759)

[8. Utility 7](#_Toc178636760)

# Back-end

## User authentication

Registration and Login: Handled user sign-up and login, including the predefined admin user. Passwords are hashed for security, and a JWT token is generated upon successful authentication.

* Registration:
* Registers users and checks if the user is the predefined admin
* Path: controllers/authController.js
* Route: POST /api/auth/register
* Login:
* Authenticates users and generates a token.
* Path: controllers/authController.js
* Route: POST /api/auth/login
* JWT Handling:
* JWT tokens are generated and verified to protect routes\
* Path: utils/jwtUtils.js

## Blog Management

Create a CRUD for user to post, view, like and comment on post.

* Create blog post:
* Allows authenticated users to create blog posts
* Path: controllers/blogController.js
* Route: POST /api/blogs
* Retrieve Blog Posts:
* Fetches all blog posts with associated comments and likes:
* Location: controllers/blogController.js
* GET /api/blogs
* Like and Comment:
* Enables users to like and comment on posts
* Location: controllers/blogController.js
* Route: PUT /api/blogs/:id/like and POST /api/blogs/:id/comment

## Character Management

Create a CRUD to manage historical character. Admin can view, update, delete and create characters while user can only view characters.

* Path: controllers/characterController.js
* Route: POST /api/characters

GET /api/characters

GET /api/characters/:id

PUT /api/characters/:id

DELETE /api/characters/:id

Middleware:

* Protected with admin authorization middleware
* Path: middlewares/adminMiddleware.js

Model: Character details such as name, era, notable works, and image URL are stored in MongoDB.

* Path: models/Character.js

## Chatbot Interaction

Chatbot Interaction: Integrates OpenAI's GPT model to simulate conversations with historical characters. The chatbot will responds as the selected character.

* Location: controllers/chatbotController.js
* Routes: POST /api/characters/:id/chat

Service:

* Communicates with the OpenAI API to generate responses.
* Location: chatbot/openaiService.js

API Call:

* Uses axios to interact with the OpenAI API.

## Middleware and Authentication

Protect Middleware:

* Protects routes by verifying JWT tokens.
* Location: middlewares/authMiddleware.js

Admin Middleware:

* Ensures only admins can access certain routes.
* Location: middlewares/adminMiddleware.js

## Error Handling

Global Error Handling:

* Custom error handling middleware is in place to catch and format errors across the application.
* Location: middlewares/errorMiddleware.js

## Database connection

Database Configuration:

* Utilizes Mongoose to connect to MongoDB.
* Location: config/dbConfig.js

Environment Variables:

* MongoDB URI and other configuration details are loaded from .env.
* Location: config/env.js

# Front-end

## App structure and routing:

React Router:

* Utilizes react-router-dom for client-side routing, with protected routes for authenticated users and admin access.

Pages:

* IndexPage: Landing page.
* LoginPage, SignupPage: User authentication pages.
* LoginAdmin, SignupAdmin: Admin-specific authentication pages.
* AdminDashboard, CharacterManagement: Admin dashboard and character management.
* HomePage, CharacterPage, CharacterDetailPage, BlogPage: Main user pages to explore characters and interact with the chatbot and interact with other user through a forum.

Protected Routes:

* Access to certain pages (admin dashboard, character management, etc.) is restricted based on user roles (admin or regular user).
* Location: components/ProtectedRoute.js

## Styling and UI

Tailwind CSS:

* Used for styling throughout the application, providing a responsive design.
* Customization: Utility classes were used to adjust layouts, button styles, modal overlays, and form elements.
* Location: Global styles and imports are in index.css.

## Admin Dashboard and Character Management

Admin Dashboard:

* Admin users can manage characters and navigate to the character management page.
* Add Character Modal: Admins can add new characters via a modal form.
* Location: pages/AdminDashboard.js

Character Management:

* CRUD operations for characters: Admins can view, edit, delete, and add characters.
* The character list is displayed in a responsive table with scrollable overflow.
* Location: pages/CharacterManagement.js

## Blog Forum Functionality

Blog Page:

* Users can view blog posts, like posts, and add comments.
* Blog Creation: Users can create new blog posts.
* Location: pages/BlogPage.js

Components:

* BlogForm: Form for creating new blog posts.
* BlogPostList: Lists blog posts and displays them on the page.
* BlogPost: Handles liking and commenting functionality for individual blog posts.
* Location: components/BlogForm.js, components/BlogPostList.js, components/BlogPost.js

## Character Exploration

Character Page:

* Users can explore a list of historical characters.
* Carousel: A character carousel allows users to scroll through character cards.
* SearchBar: Allows users to search for a character by name.
* Location: pages/CharacterPage.js

Components:

* CharacterCarousel: Carousel to display characters.
* CharacterCard: Displays a character's image and name.
* SearchBar: Provides a search interface for characters.
* Location: components/CharacterCarousel.js, components/CharacterCard.js, components/SearchBar.js

## Chatbot

Character Detail Page:

* Allows users to view detailed information about a character.
* Users can ask questions to a chatbot, and the chatbot responds as the historical character.
* Location: pages/CharacterDetailPage.js

ChatBox Component:

* Users type questions, and the chatbot responds with answers generated by the backend using OpenAI's GPT model.
* Location: components/ChatBox.js

## User Authentication

Login and Signup:

* User login and signup pages for both regular users and admin users.
* After successful login, tokens are stored in local storage, and the user is redirected to their respective dashboards.
* Location: pages/LoginPage.js, pages/SignupPage.js, pages/LoginAdmin.js, pages/SignupAdmin.js

## Utility

Time Ago Utility:

* Formats the timestamp for blog post comments and likes.
* Location: utils/timeAgo.js

Modal Component:

* Used for showing forms and interaction modals across the application.
* Location: components/Modal.js