AnswerAi Backend Service API Documentation

Overview

This document provides details about the RESTful API endpoints for the backend service. The service is built using Node.js and Express.js, is designed to handle user authentication, question submissions, and AI-generated answers. The API is secured with JWT tokens and is intended to be used with a front-end application.

Base URL

All endpoints are prefixed with `/api`.

Authentication

The API uses JSON Web Tokens (JWT) for authentication. Users must include a valid JWT token in the **`Authorization`** header for endpoints that require authentication.

Example:

```
Authorization: Bearer <your-jwt-token>
```

Endpoints

1. User Authentication

1.1 User Login

Endpoint: **`POST /api/auth/login`**

Description: Authenticate a user and generate a JWT token.

Request:

```
{
  "email": "string",
  "password": "string"
}
```

Response:

- 200 OK

```
{
  "token": "Generated-jwt-token"
}
```

- 400 Bad Request

```
{
  "error": "Error Message"
}
```

- 404 Not Found

```
{
  "error": "User not found"
}
```

1.2 User Logout

Endpoint: `POST /api/auth/logout`

Description: Invalidate the user's session (usually handled client-side).

Response:

- 200 OK

```
{
    "message": "User logged out successfully and token
                blacklisted"
}
```

1.3 User Refresh session

Endpoint: `POST /api/auth/refresh`

Description: Invalidate the user's session (usually handled client-side).

Body :

```
{
     "refreshToken": “current token number”
}
```

Response:

- **200 OK**

```
{
  "newToken": "regeneratedToken"
}
```

- **400 BadRequest**

```
{
    "error": "Invalid refresh token"
}
```

2. User Management

2.1 Create a New User

Endpoint: `POST /api/users/newUsers`

Description: Create a new user account. (Typically handled by the register endpoint)

Request:

```
{
  "username": "string",
  "email": "string",
  "password": "string"
}
```

Response:

- **201 Created**

```
{
  "message": "User created successfully"
}
```

- **400 Bad Request**

```
{
  "error": "Error message"
}
```

2.2 Retrieve User Profile

Endpoint: `GET /api/users/:userId`

Description: Retrieve a user profile by user ID.

Response:

- **200 OK**

```
{
    "questions": “Array”,
    "username": "String",
    "email": "String",
    "password": "String",
    "createdAt": "String"",
    "updatedAt": "String",
    "user_id": "String"
}
```

- **404 Not Found**

```
{
  "error": "User not found"
}
```

2.3 Retrieve User's Questions

Endpoint: `GET /api/users/:userId/questions`

Description: Retrieve all questions asked by a user.

Response:

- **200 OK**

```
[
  {
    "id": "string",
```

```
    "userId": "string",
    "question": "string",
    "answer": "string",
    "createdAt": "timestamp",
    "updatedAt": "timestamp"
  },
  ...
]
```

3. Questions Management

3.1 Submit a Question

Endpoint: `POST /api/questions`

Description: Accept a user question and return an AI-generated answer.

Request:

```
{
  "userId": "string",
  "question": "string"
}
```

Response:

- **201 Created**

```
{
  "id": "string",
  "userId": "string",
  "question": "string",
  "answer": "string",
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}
```

- **400 Bad Request**

```
{
  "error": "Error message"
}
```

3.2 Retrieve a Specific Question

Endpoint: `GET /api/questions/:questionId`

Description: Retrieve a specific question and its answer by question ID.

Response:

- **200 OK**

```
{
  "id": "string",
  "userId": "string",
  "question": "string",
  "answer": "string",
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}
```

- **404 Not Found**

```
{
  "error": "Question not found"
}
```

Error Handling

The API uses standard HTTP status codes to indicate the success or failure of an API request:

- **200 OK**: The request was successful.
- **201 Created**: A new resource has been successfully created.
- **400 Bad Request**: The request was invalid or cannot be otherwise served.
- **401 Unauthorized**: Authentication is required and has failed or has yet to be provided.
- **404 Not Found**: The requested resource could not be found.

In addition, the response body will contain a JSON object with an `error` field that provides more details about the error.

Example Requests

Example: Register a New User

Request:

```
Postman: POST  https://localhost:5003/api/users

Header: "Content-Type: application/json" \
Boby : Raw[json]
      {
          "username": "Ruchi Gupta",
          "email": "gupta.ruchi@gmail.com",
          "password": "ddkcsadee3"
      }
```

Response:

```
      {
          "message": "Ruchi Gupta registered successfully"
      }
```

Example: Login a User

Request:

```
Postman: POST  https://localhost:5003/api/auth/login

Header: "Content-Type: application/json"
Body: raw[Json]
      {
          "email": "gupta.ruchi@gmail.com",
          "password": "ddkcsadee3"
      }
```

Response:

```
      {
        "token": "your-jwt-token"
      }
```

- If the mail ID is incorrect:

```
      {
          "error": "User not found"
      }
```

- If the password is incorrect:

```
      {
          "error": "Invalid credentials"
      }
```

Example: Submit a Question

Request:

```
Postman: POST https://localhost:5003/api/questions

Headers: "Content-Type: application/json”
          "Authorization: Bearer your-jwt-token"
Body: raw[Json}
      {
          "userId": "66712597eeee662056c0fa93",
          "question": "What is the capital of India?"
      }
```

Response:


```
{
    "userId": "66712597eeee662056c0fa93",
    "question": "What is the capital of India?",
    "answer": "The capital of India is New Delhi. It is located in the
          northern part of the country and serves as the seat of the
          Government of India. New Delhi is part of the larger
          metropolis of Delhi, which is the second most populous city
          in India after Mumbai. The city is known for its rich
          history, diverse culture, and iconic landmarks such as the
          Red Fort, India Gate, and the Lotus Temple.",
    "createdAt": "2024-06-18T06:22:12.660Z",
    "updatedAt": "2024-06-18T06:22:12.660Z",
    "que_id": "66712794eeee662056c0fa9f"
}
```


Conclusion

This API documentation provides a detailed overview of the available endpoints, their usage, and expected responses. It serves as a guide for developers to integrate with the backend service efficiently and securely. If you have any questions or need further assistance, please contact the support team.