Authentication API Documentation

Introduction

This documentation provides an overview of the authentication API implemented using Node.js. The API allows users to register, log in, log out, view and edit their profiles, set profile privacy, and upload profile pictures. It includes authentication and authorization mechanisms to ensure secure access to user data.

Base URL

The base URL for the API is `http://localhost:3000` (or the URL where your server is hosted).

Authentication Endpoints

Signup User Profile

- **Endpoint**: `POST /signup`
- **Description**: Registers a new user with the provided username, email, and password.
- **Request Body**:

```
{
  "username": "example_user",
  "email": "user@example.com",
  "password": "password",
  "profile": {
    "name": "",
    "bio": "",
```

```
    "phone": "",
    "photo": "",
    "privacy": "public"
  }
}
```

- **Response**: Returns the newly registered user object.

Sign Up with Google OAuth

- **Endpoint**: `GET /auth/google`
- **Description**: Initiates the Google OAuth sign-up flow for registering a new user.
- **Response**: Redirects the user to the Google OAuth consent screen to authorize the application.

Log In User

- **Endpoint**: `POST /login`
- **Description**: Logs in an existing user with the provided username and password.
- **Request Body**:
- ```
  {
  "username":"example_user",
  "password":"password"
  }
  ```
- **Response**: Returns an access token in a cookie.

Log Out User

- **Endpoint**: `POST /logout`
- **Description**: Logs out the currently authenticated user by clearing the authentication token stored in the client-side cookie.
- **Response**: Returns a success message indicating that the user has been signed out successfully.
  ```

Profile Endpoints

View User Profile

- **Endpoint**: `GET /profile`
- **Description**: Retrieves the profile details of the currently authenticated user.
- **Authentication**: Required
- **Response**: Returns the user's profile object.

Update User Profile

- **Endpoint**: `PUT /profile`
- **Description**: Updates the profile details of the currently authenticated user.
- **Authentication**: Required
- **Request Body**: Includes the fields to be updated (e.g., name, bio, phone, email, password).
- **Response**: Returns the updated user profile object.

Set Profile Privacy

- **Endpoint**: `PUT /profile/privacy`
- **Description**: Sets the profile privacy setting (public or private) for the currently authenticated user.
- **Authentication**: Required
- **Request Body**:
- ```
  {
  "privacy":"public"
  }
  ```
- **Response**: Returns the updated user profile object.

Upload Profile Picture

- **Endpoint**: `POST /profile/upload`
- **Description**: Uploads a new profile picture for the currently authenticated user.
- **Authentication**: Required
- **Request Body**: Includes the image file to be uploaded.
  ```

- **Response**: Returns the updated user profile object with the path to the uploaded picture.

Error Handling

- The API handles errors gracefully and returns appropriate error messages and status codes for invalid requests or server errors.

ESLint Configuration

To maintain code quality and consistency, ESLint is used with the following basic configuration:

```
module.exports = {
  root: true,
  env: {
    node: true,
    es6: true,
  },
  extends: ['eslint:recommended'],
  parserOptions: {
    ecmaVersion: 2021,
  },
  rules: {
    'indent': ['error', 2], // Enforce 2-space indentation
    'semi': ['error', 'always'], // Require semicolons at the end of
statements
    'quotes': ['error', 'single'], // Use single quotes for strings
    'no-unused-vars': 'warn', // Warn about unused variables
    'no-console': 'warn', // Warn about console statements
  },
};
```

Husky Integration

Husky is integrated into the project to enforce ESLint rules before commits. Here's how it's configured in the **`package.json`**:

```
"husky": {
  "hooks": {
    "pre-commit": "eslint ."
  }
}
```

Docker Configuration

Build the Docker image

```
docker build -t authentication-api .
```

Run the Docker container

```
docker run -p 3000:3000 authentication-api
```