User Authentication Application Documentation

Backend Routes

The backend application uses Express.js to handle user authentication with a MongoDB database.

1. User Authentication Routes

Route: POST /auth/signup

Registers a new user.

Request:

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securepassword"
}
```

Response:

If email already exists:

```
{
  "message": "Email is already registered"
}
```

If successful insertion:

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

If an exception is raised:

```
{
  "message": "Error registering user",
  "error": "<error-details>"
}
```

Route: POST /auth/login

Logs in an existing user.

Request:

```
{
  "email": "john.doe@example.com",
```

```
  "password": "securepassword"
}
```

Response:

If the user does not exist:

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

If the password is incorrect:

```
{
  "message": "Invalid password"
}
```

If the user logs in successfully:

```
{
  "message": "Login successful",
  "token": "<jwt-token>",
  "user": {
    "name": "John Doe",
    "email": "john.doe@example.com",
    "role": "user"
  }
}
```

If an exception is raised:

```
{
  "message": "Error logging in",
  "error": "<error-details>"
}
```

Route: GET /auth/protected

Protected route to verify authentication.

Headers:

```
{
  "Authorization": "Bearer <JWT_TOKEN>"
}
```

Response:

If the token is valid:

```
{
  "message": "Hello John Doe, you accessed a protected route!"
}
```

If the token is invalid or expired:

```
{
  "message": "Forbidden"
}
```

Frontend Application Structure

The frontend application uses React.js and is organized as follows:

Folder Structure

```
src/
 |── components/
 │   |── Signup.js
 │   |── Login.js
 │   |── Dashboard.js
 |── App.js
 |── index.js
```

Frontend Components

1. Signup Component

Allows users to register with fields for "name", "email", and "password". Displays success messages or error messages as appropriate.

2. Login Component

Allows users to log in with fields for "email" and "password". On successful login, stores the JWT token and Displays success messages. Displays error messages for invalid credentials.

3. Logout Behavior

Clears the JWT token from sessionStorage and redirects the user to the Signup page.

4. Dashboard Component

Displays a protected dashboard for authenticated users. Includes a personalized welcome message and a success message.

App.js

Manages routing and navigation. Includes routes for Signup, Login, and Dashboard, with conditional navigation links based on authentication status.