Project Overview

This project is a web application that allows users to register, log in, and manage their to-do items. The backend is built using Node.js with Express and SQLite, while the frontend is developed with React. The application features user authentication, CRUD operations for to-do items, and JWT-based session management.

Setup Instructions

Prerequisites

- Node.js and npm (Node Package Manager) installed on your machine.
- SQLite installed for the database.

Backend Setup

1. **Clone the Repository**

   ```
   git clone https://github.com/your-repo/your-project.git
   cd your-project/backend
   ```

2. **Install Dependencies**

   ```
   npm install
   ```

3. **Create the `.env` File**

   Create a file named `.env` in the `backend` directory with the following content:

   ```
   JWT_SECRET=your_jwt_secret_key
   ```

   Replace `your_jwt_secret_key` with a secure random key.

4. **Run Database Migrations**

   The database schema will be set up automatically when you run the server for the first time.

5. **Start the Server**

   ```
   node server.js
   ```

   The backend server will start on `http://localhost:4000`.

Frontend Setup

1. **Navigate to the Frontend Directory**

   ```
   cd frontend
   ```

2. **Install Dependencies**

   ```
   npm install
   ```

3. **Run the Development Server**

   ```
   npm start
   ```

   The React application will start on `http://localhost:4000`.

Technologies Used

- **Node.js**: JavaScript runtime used for the backend.
- **Express**: Web framework for building the REST API.
- **SQLite**: Lightweight database for storing user and to-do data.
- **bcryptjs**: Library for hashing passwords securely.
- **jsonwebtoken**: Library for creating and verifying JWTs.
- **cors**: Middleware to enable cross-origin requests.
- **dotenv**: Library to manage environment variables.
- **React**: JavaScript library for building the user interface.
- **axios**: HTTP client for making API requests from the frontend.

API Endpoints

User Endpoints

- **POST /register**
  - **Description**: Register a new user.
  - **Request**:

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

  - **Response**:

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

- **POST /login**
  - **Description**: Log in a user and receive a JWT.
  - **Request**:

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

  - **Response**:

```
{
  "token": "jwt_token"
}
```

To-Do Endpoints

- **POST /todos**
  - **Description**: Add a new to-do item.
  - **Request**:

```
{
  "description": "string",
  "status": "string"
}
```

  - **Response**:

```
{
  "id": "integer",
  "description": "string",
  "status": "string"
}
```

- **GET /todos**
  - **Description**: Retrieve all to-do items for the logged-in user.
  - **Response**:

```
[
  {
    "id": "integer",
    "description": "string",
    "status": "string"
  }
]
```

- **PUT /todos/**
  - **Description**: Update a specific to-do item.
  - **Request**:

    ```
    {
      "description": "string",
      "status": "string"
    }
    ```

  - **Response**:

    ```
    {
      "id": "integer",
      "description": "string",
      "status": "string"
    }
    ```

- **DELETE /todos/**
  - **Description**: Delete a specific to-do item.
  - **Response**:

    ```
    {
      "message": "Todo item deleted successfully."
    }
    ```

Database Schema

Users Table

- **id**: INTEGER, Primary Key, Auto Increment
- **username**: TEXT, Unique
- **password**: TEXT

Todos Table

- **id**: INTEGER, Primary Key, Auto Increment
- **user_id**: INTEGER, Foreign Key (references Users.id)
- **description**: TEXT
- **status**: TEXT

Deployment Steps

Deploy Backend on Render

1. **Create a Render Account:**
   - Go to Render and sign up or log in.
2. **Create a New Web Service:**
   - Click on the "New" button and select "Web Service".
3. **Connect GitHub Repository:**
   - Connect your GitHub account to Render and select the repository containing your backend code.
4. **Configure Service:**
   - Fill in the details for your service:
     - **Name:** Choose a name for your service.
     - **Branch:** Select the branch that contains your backend code (e.g., `main`).
     - **Build Command:** `npm install`
     - **Start Command:** `npm start`
     - **Environment:** Add your environment variables from your `.env` file:
       - `JWT_SECRET` (the value from your local `.env` file)
   - Click on "Create Web Service".
5. **Deploy:**
   - Render will now build and deploy your backend. Once the deployment is complete, you will have a URL for your backend service (e.g., `https://your-backend.onrender.com`).

Deploy Frontend on Netlify

1. **Create a Netlify Account:**
   - Go to Netlify and sign up or log in.
2. **New Site from Git:**
   - Click on "New site from Git" and connect your GitHub account.
3. **Select Repository:**
   - Select the repository containing your frontend code.
4. **Configure Build Settings:**
   - **Branch to deploy:** `main` (or the branch with your frontend code)
   - **Build Command:** `npm run build`
   - **Publish Directory:** `build`
   - **Environment Variables:** Add any necessary environment variables if needed.
   - Click on "Deploy Site".
5. **Deploy:**

- Netlify will build and deploy your frontend. Once the deployment is complete, you will have a URL for your frontend (e.g., `https://your-frontend.netlify.app`).

Connect Frontend to Backend

1. **Update Frontend API URL:**
   - In your frontend code, update the `API_URL` in `src/services/api.js` to point to your Render backend URL.

```
const API_URL = 'https://your-backend.onrender.com';
```

2. **Rebuild and Redeploy Frontend:**
   - After updating the API URL, rebuild your frontend and redeploy it on Netlify.

```
npm run build
```

Commit and push the changes to your GitHub repository to trigger a new build on Netlify.

Verify Integration

1. **Test Registration and Login:**
   - Open your deployed frontend URL and try registering a new user. Verify that the backend is correctly handling the registration and login requests.
2. **Test To-Do Management:**
   - After logging in, add, update, and delete to-do items to ensure that the frontend and backend are communicating correctly.

Challenges and Solutions

- **JWT Secret Management**: Initially faced issues with managing JWT secrets. Resolved by using the `dotenv` package to securely store secrets in the `.env` file.
- **Frontend State Updates**: Faced challenges with state updates not reflecting immediately. Resolved by ensuring correct state management and re-fetching data as needed.

Future Improvements

- **Enhanced Authentication**: Implement more advanced authentication features such as two-factor authentication (2FA).
- **Error Handling**: Improve error handling on both frontend and backend to provide more informative user feedback.
- **User Profiles**: Add user profile management features.
- **Search and Filter**: Implement search and filter functionalities for to-do items.
- **Mobile Optimization**: Ensure the application is fully responsive and optimized for mobile devices.