Fuel Price Tracker - Phase 3: User Accounts & Preferences

1. Introduction

In Phase 3, you'll add user accounts to your Fuel Price Tracker app. This means users can register, log in, and save preferences like favorite stations and alert thresholds.

2. What You?ll Build

- A user registration + login system
- Secure password handling
- JWT-based authentication
- MongoDB storage for user data
- API endpoints to manage favorites and preferences

3. Tools You'll Use

- Express.js (backend)
- bcryptjs (for password hashing)
- jsonwebtoken (for auth)
- MongoDB (for user storage)
- React (frontend forms)
- Axios (API calls)

4. Step-by-Step: Backend Setup

1. Install:
`npm install express bcryptjs jsonwebtoken mongoose cors`
2. Create `User.js` model:
```js
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: String,
  favorites: Array,
```

Fuel Price Tracker - Phase 3: User Accounts & Preferences

```
  alertThreshold: Number
});
module.exports = mongoose.model('User', userSchema);
```

5. Auth Routes (Register/Login)

Set up `/register` and `/login` routes in Express:

- Hash passwords using bcrypt
- Use JWTs to return tokens
- Store tokens in localStorage on frontend

6. Frontend Auth Forms

Create login/register components in React:

- Use controlled inputs for email/password
- On submit, call backend via Axios
- On success, store token in localStorage

7. Protected Routes

- In backend, create `authMiddleware.js` to verify JWTs
- Protect API routes that require login (favorites, alerts)
- On frontend, hide/show features based on login status

8. Saving Preferences

- Use authenticated routes to GET/POST:
  - Favorite stations (by ID)
  - Alert settings (like max fuel price)
- Store and retrieve these from the user model in MongoDB

9. Final Checklist

Fuel Price Tracker - Phase 3: User Accounts & Preferences

- Passwords hashed ?
- JWT tokens issued and verified ?
- API secured ?
- Frontend integrated ?