Project document

Project name: Webapp Shop

Owner: Vo Khoi Nam Hai

Document Category:

1. Brief introduction and basic flow
2. Project components:
   1. Back end
   2. Data base
   3. Identity provider
   4. Front end
3. What I learnt
4. **Brief introduction and basic flow**

The Webapp shop project is a full-stack web application consists of several components:

* Backend API using NodeJS and Express.js framework
* MySQL database to store data
* IdentityServer 4 to authenticate user
* React/Redux in the frontend

All components are containerized by Docker.

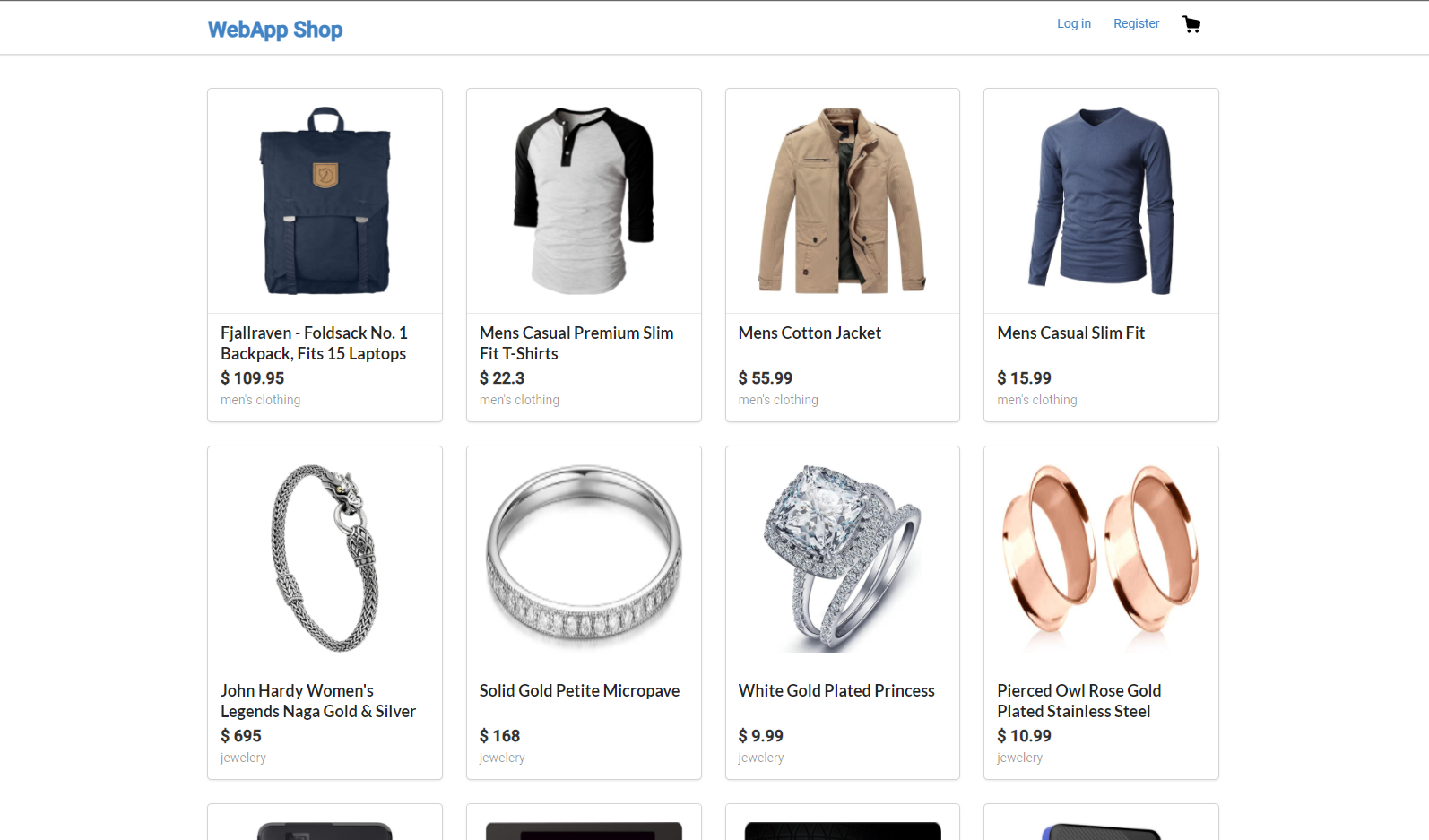
The project is the very basic implementation of the above technologies, which contains the functionalities to let user register a new account, log in/out their existing account, view product list, and view individual product.

Basic flow of the authentication process:

To run the demo on Docker, we need to follow these steps:

* Run docker-compose up.
* From the IdentityServer project folder run these scripts in command line to migrate necessary tables to the database:
  + update-database -context PersistedGrantDbContext
  + update-database -context ConfigurationDbContext
  + update-database -context ApplicationDbContext
* From the IdentityServer project folder run this script in command line to seed test data
  + dotnet run bin/Debug/netcoreapp3.1/IdentityService /seed
* From the Server folder run this script in command line to seed test data
  + npx sequelize-cli db:seed:all --url mysql://root:12345@localhost:3306/my-react-app.

After running the scripts successfully, we can access the project main page via <http://localhost:3000>.



To test the login/logout function we can use either test user/password:

- alice/Pass123$

- bob/Pass123$

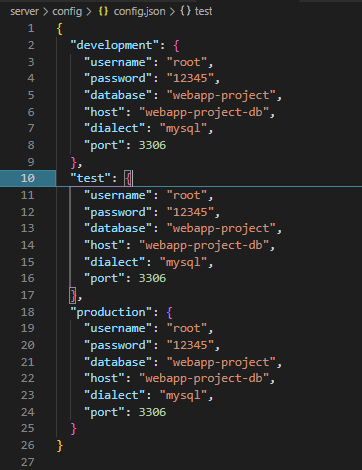
1. **Project components**
   1. Backend

The backend is written in NodeJS, utilizing ExpressJS framework to help with routing and exposing API to the frontend. I also use Sequelize ORM to simplify the interactions with database, like CRUD, model mapping, seeding data etc.

To protect secured APIs, I use express-jwt and jwks-rsa libraries to authenticate user access-token when they make request to protected APIs.

Disclaimer: the API authentication and authorization is not working as expected. When I tried accessing protected API it responded with the error “ERREFUSECONNECT”. I have yet to find the solution to this.

The configuration for sequelize to connect to the MySQL database can be found in the config.json file in the config folder.

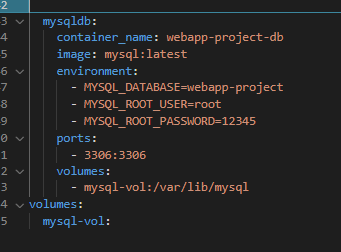


The backend is exposing port 3001 so for the frontend to connect and uses its APIs

* 1. MySQL database

For this project I use a MySQL image from docker hub and run it along with other components in the same docker-compose network so they can communicate with each other.

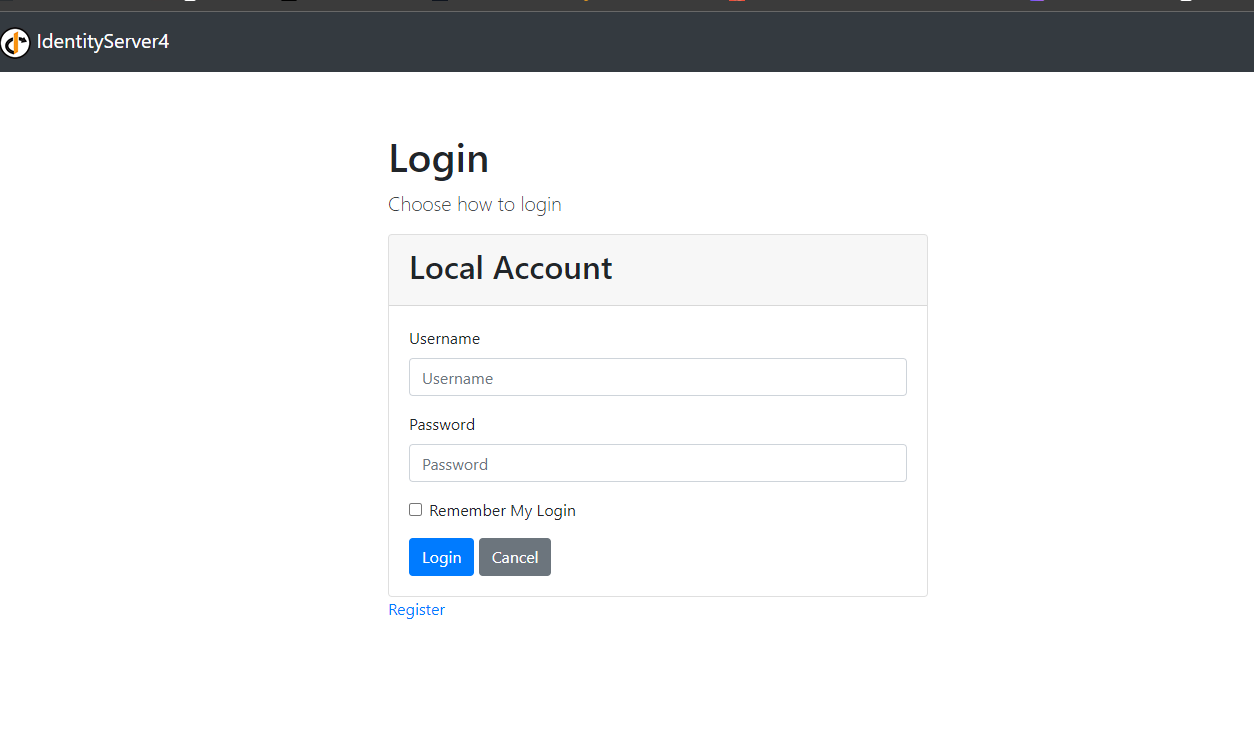
For the purpose of this project I am setting root user as ROOT, and other setting is configured in the docker-compose file.



* 1. Identity Server

The Identity Server Is implemented with IdentityServer4 and ASPNET Identity library to work as an identity provider for authenticating and providing identity token to users.

When user clicks on login link, they will be redirected to the identity server login page to authenticate user.

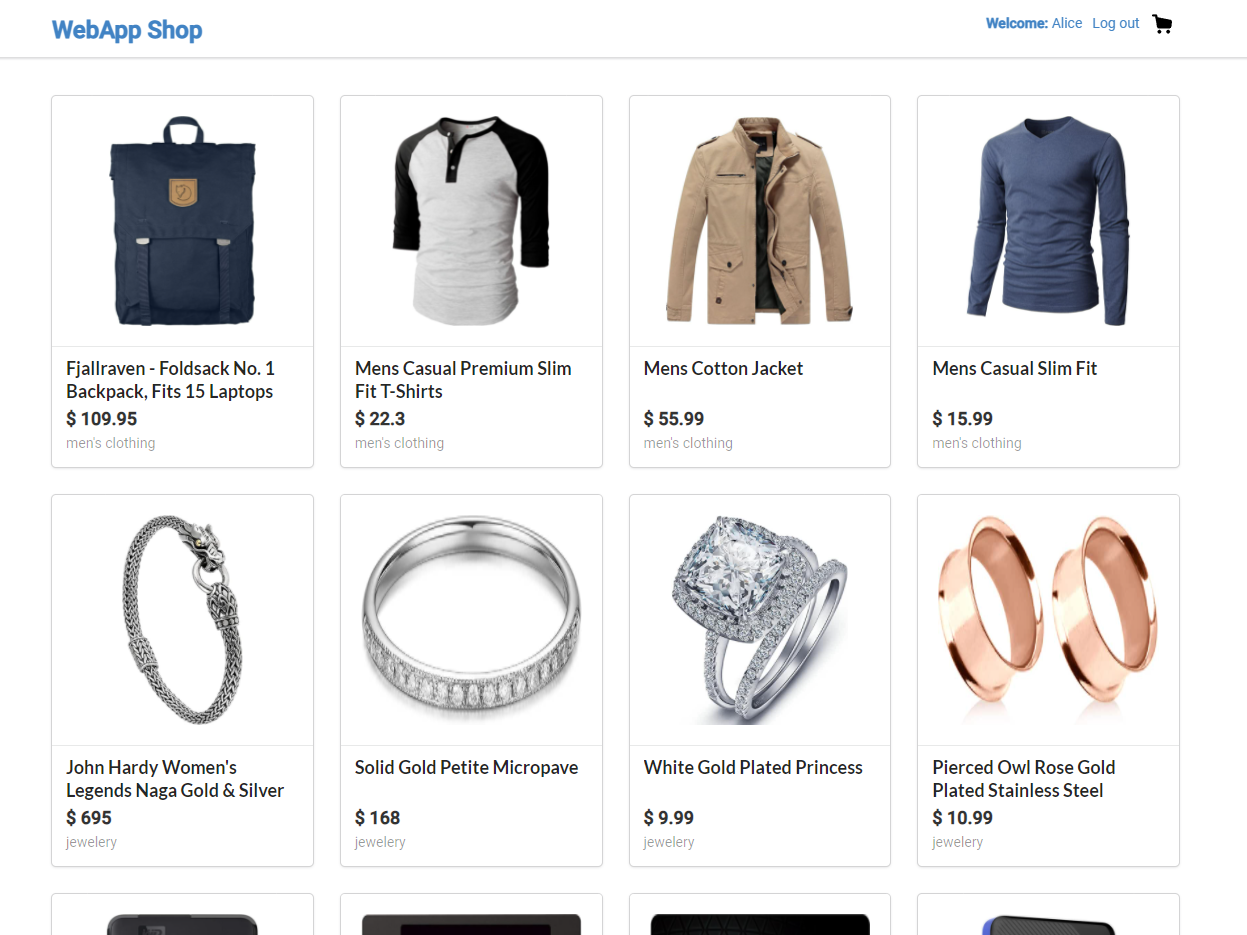


To test the login/logout function we can use either test user/password:

- alice/Pass123$

- bob/Pass123$

After successfully login in, user will be redirected back to the home page.

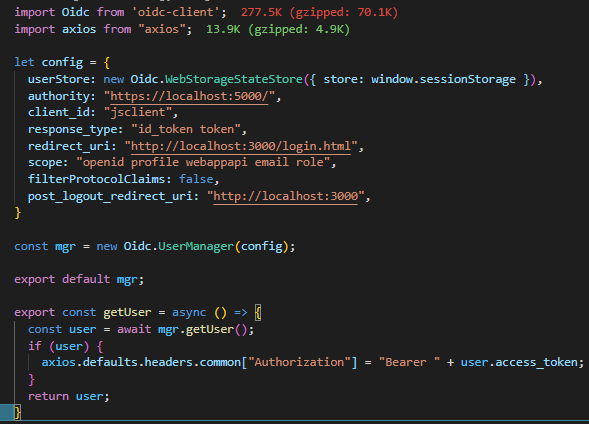


* 1. Frontend

The frontend is an SPA web application, powered by react and redux, and react-router to handle routing between components. I also use MaterialUI to help with styling and responsiveness.

To handle user login/out, and user registration and I use the oidc-client library to handle OpenId Connect protocol.

The oidc client config is show as below.



1. **What I learnt**