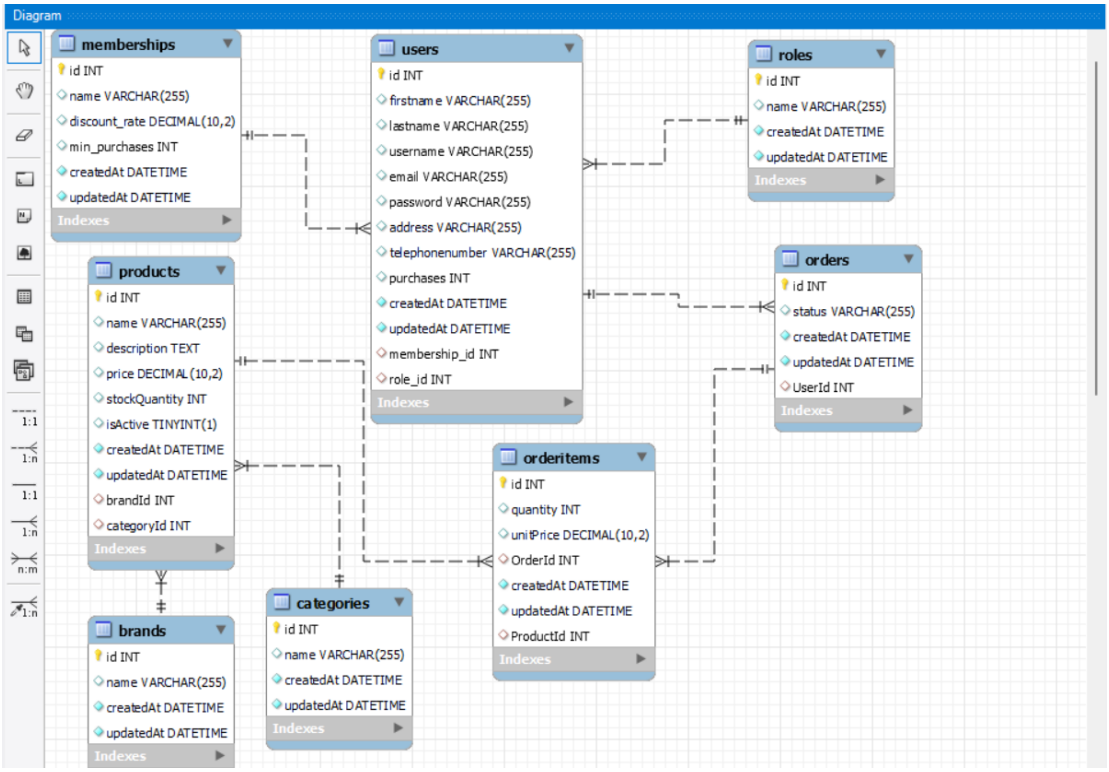
Final Exam Backend Web Dev

Charles Warrener mar23 fulltime.

Reflection report



Relationships:

users to roles: Users are assigned roles. This is a many-to-one relationship, meaning many users can have the same role.

users to memberships: Users have memberships. It's a many-to-one relationship as well since many users can have the same membership.

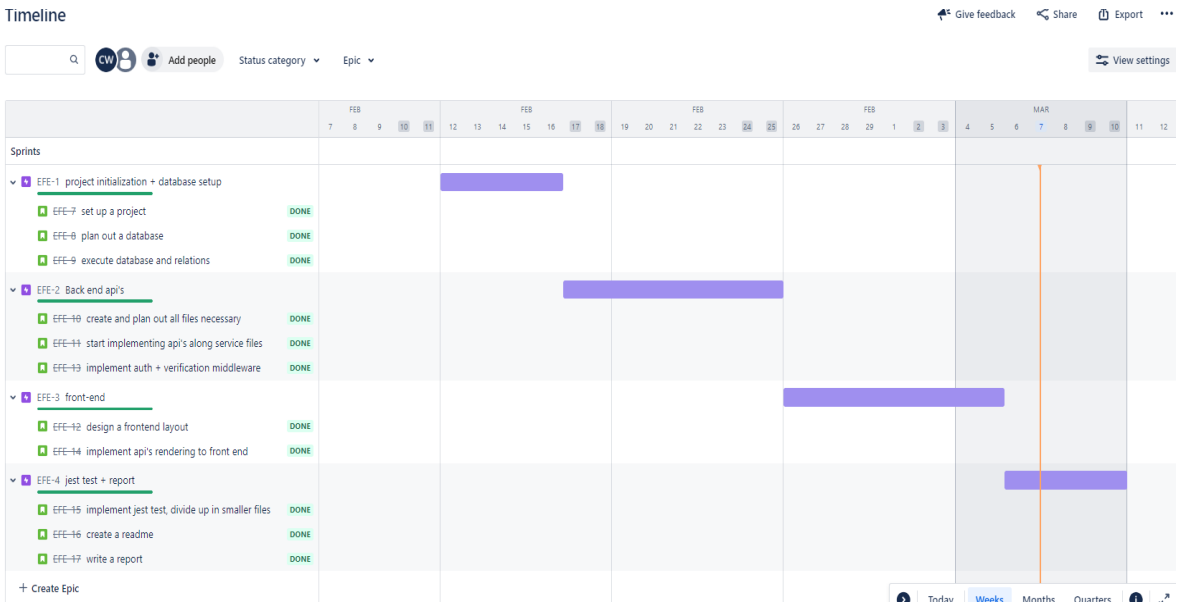
users to orders: Users place orders. There is a one-to-many relationship from users to orders, so one user can place many orders.

products to categories: Products are placed into categories. It's a many-to-one relationship, where many products can belong to one category.

products to brands: Products are associated with brands. This is also a many-to-one relationship, as many products can be associated with a single brand.

orders to orderitems: Each order can consist of several order items, which is a one-to-many relationship.

products to orderitems: The relationship between products and orderitems is a one-to-many relationship. Each product can be associated with multiple orderitems indicating that it can be part of many orders, each orderitem relates to one product.



Project write-up:

Discussion of the Progression:

The project began with the setup of a Node.js environment utilizing Express for server management and Sequelize ORM for database interactions. The initial phase involved setting up the structure of the project. Planning out what folders and files would go where

to keep the project maintainable. After that I set up the connection to the database, creating the necessary models and relations.

I moved along to start building the backend of the application, service layers were introduced. These services provide clean separation and allow for easier maintenance and scalability. The RESTful API endpoints were then defined in router files. I kept any authorization at bay in the beginning because I knew I'd be testing these endpoints frequently. Before beginning the front-end of the project, I added the middleware for authorization and completed the back end.

Now I was ready to take on the front-end. Most of the endpoints were very similar so I knew just a basic rewrite to render to frontend instead of return JSON was necessary. This part therefore moved along at a good pace. The front-end was fairly similar to the MEMES project we had earlier this year, therefore this was fairly painless too.

What I'm really happy about with this project is taking my time planning it out in the beginning. This project was much larger than any previous ones we've built. Therefore, scalability and structure were crucial. Without careful consideration of how this project would be built it could easily have become a mess.

Challenges Faced:

Tackling authentication was one of my first major hurdles, especially integrating Passport for handling user sessions. The aim was to create a secure and effective login system, but it definitely tested my patience and skills even though i did have some previous projects with similar set-ups.

Then came the challenge of unit testing, particularly for endpoints guarded by authentication. Simulating login states to test these secured endpoints required a significant amount of work. Crafting tests for categories tests and project tests meant I had to dive deep into understanding how to obtain and use tokens correctly, ensuring the tests could work effectively.

The API endpoint management also threw some curveballs. This is where a couple relation issues in the database came forth. The detail needed in the JSON responses and the specific structure they required swallowed quite a bit of my time.

Integrating session management through Passport brought a set of challenges, even though I've worked and set Passport up before I recall having issues last time too. This time around

I wanted to divide the project properly and keep the app.js file clean. Demanding more attention and time than I had initially estimated.