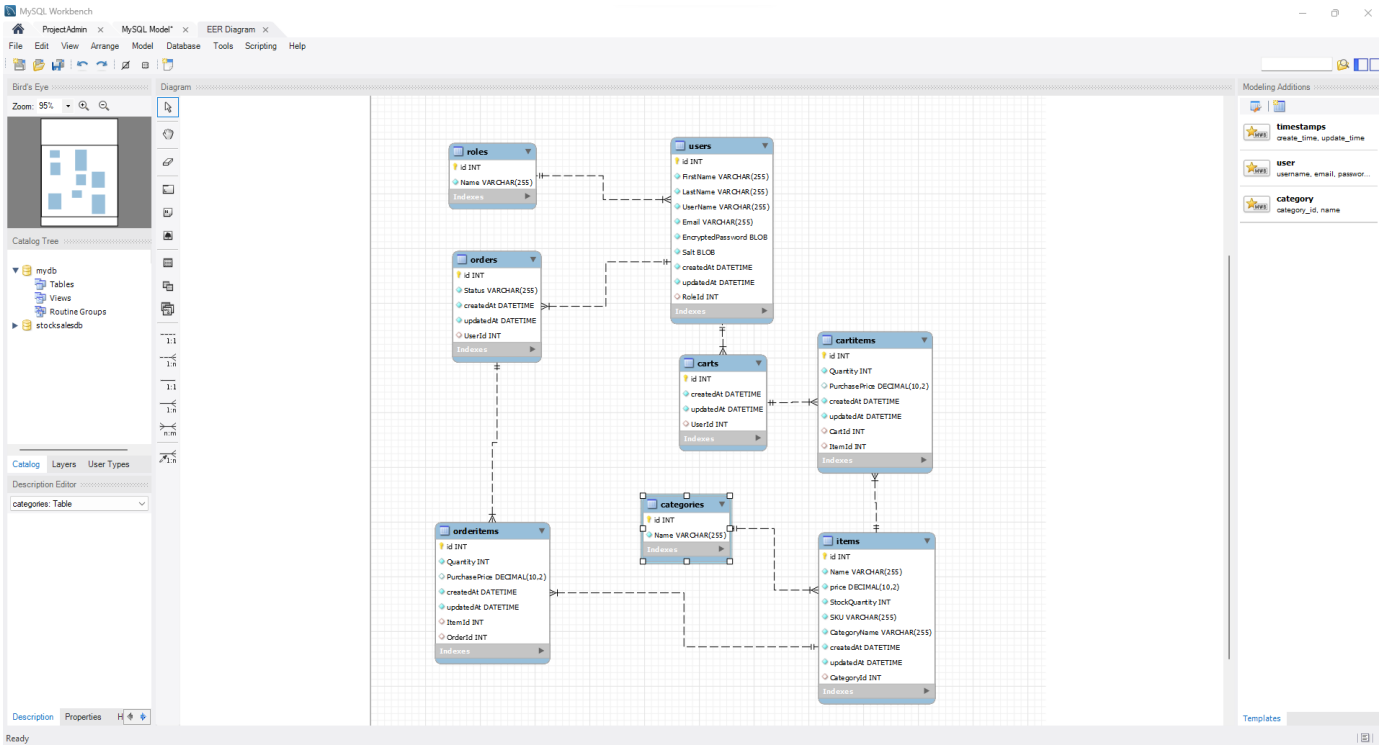
Exam documentation

Github directory: https://github.com/fadezilla/Exam-project

Database setup:



Explanation of the relationship between database tables:

User and role

- User belongs to Role, this a one-to-one relationship where the user belongs to the role. This relationship is chosen to associate a user with a specific role, such as "Admin" or "user"(customer).

Role and user

- Role has many users. This defines a one-to-many relationship where a role can have many users. So many users can have the same role.

User and Cart

- User has one to Cart. This defines a one-to-one relationship where a user has a cart. This is so that a user can only have one cart associated with them.

Cart and User

- Cart belongs to user. this defines a one-to-one relationship where a cart belongs to a specific user. This is chosen so that we can associate a cart with a specific user.

User and Order

- User has many orders. This defines a one-to-many relationship where a user can have multiple orders. This is to represent the concept of a user(customer) making multiple purchases by placing multiple orders.

Order and User

- Order belongs to user. This defines a one-to-one relationship where one order belongs to one user. This is chosen so we can associate an order to one specific user.

Item and Category

- Item belongs to category. This defines a one-to-one relationship where an item belongs to a category. This relationship is chosen to associate an item to a specific category. one item can only belong to one category.

Category and Item

- Category has many items. This defines a one-to-many relationship where a category can have multiple items. This is chosen so we can group multiple items under a specific category.

Item and CartItems

- Item has many cart Items. This defines a one-to-many relationship where an item can have multiple cart items. This means that each item can be added to multiple shopping carts. so, one item can be added to many different user carts.

CartItems and item

- Cart items belong to item. This defines a one-to-one relationship where a cart item belongs to an item. this is chosen so that we can associate a cartItem with a specific item.

Item and OrderItems

- Item has many order items. This defines a one-to-many relationship where an item can have multiple orderItems. This means that each item can be included in multiple orders. so, one item can be added to many different orders made by the user.

OrderItems and item

- OrderItems belongs to item. This defines a one-to-one relationship where an orderItem belongs to an item. This is chosen so we can associate an orderItem with a specific item.

Cart and CartItems

- Cart has many cartItems. This defines a one-to-many relationship where a cart can have multiple cart items. This purely means that each cart can contain multiple items. This was chosen to represent the concept of having a cart with multiple items.

CartItems and cart

- Cart items belongs to Cart. This defines a one-to-one relationship where a cart Item belongs to a cart. this is chosen to associate a cartItem with a specific cart.

Order and orderItems

- Order has many order items. This defines a one-to-many relationship where an order can have multiple orderItems. This is so we can represent the concept of an order containing multiple items.

OrderItems and order

- Order Items belong to Order. This defines a one-to-one relationship where an order Item belongs to an order. This is so we can associate an order Item with a specific order.

These relationships were chosen based on the business logic and requirements for the application.

Project retrospective write-up

1. This project went through many different key stages and phases. It started with planning and requirement gathering. Planned for what was needed to accomplish the client's needs and how I should implement all the desired features and functionalities.

Once the requirement was understood, I focused on designing the project's architecture and the database schema. How the relationships should work to have less implications later in the project as well as on thinking about the future scale-ability for the project moving forward.

After this I started with development and implementation of different functionalities as well as adding the required endpoints and functionalities within those endpoints.

2. Challenges

the first challenge I approached was choosing the correct relationships between the different tables as doing these wrong would make everything in the project much more problematic, but after some failure and testing, I reached an effective way to describe the relationships and after this handling the endpoints become much easier. I first created all the endpoints without any authentication, so I could purely focus on the functions and how to implement the unique features required to achieve the goal of each endpoint.

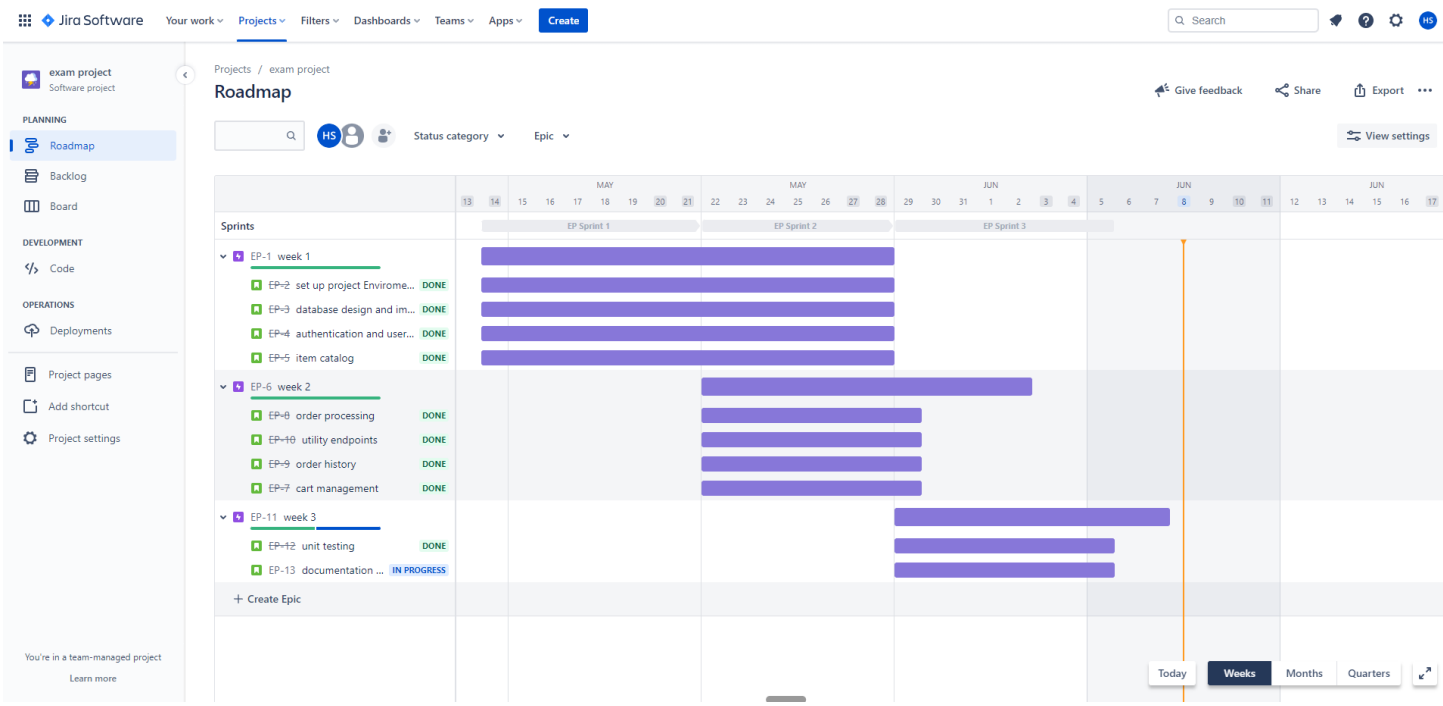
What was most challenging here must have been how the orders and cart were interacting with each other as well as orderItems and cartItems and how we should use those to represent an actual item. But after some problem solving, I believe we made the right approach to it, as all the endpoints are working as intended. As for testing everything, I had to create some more endpoints that was purely used by an admin user, this was so I can run a full test, testing each endpoint and after the test, I can simply reset all the orders and make everything as it was after we ran the setup of filling the database. I did this as its a nice functionality to have as well as thinking of the future, it's not a good approach to have to delete the database and remake the database each time we want to do testing. we might have valuable data there that we don't want to delete after all. I also added one more endpoint for the orders. as well as having the endpoint where we can order one item from the cart we have, I also added a /checkout endpoint which takes the logged in userId and uses that to find the belonging cart, and then it places an order on all the items inside that cart, this is so that we can check out all the items at the same time if we want to, instead of having to check out one item at a time.

Before testing, however, we added authentication, this wasn't as hard as I thought it would be. for this I used a JWT token as well as hash encrypted passwords for the database and saving all the sensitive information inside the .env file. for endpoints where an admin user is needed to use, we use two functions, first the authentication, which checks for the jwt token and then isAdmin function which checks the user role for admin.

for endpoints where a logged in user is needed, we just use the authentication function.

for endpoints where guest users can use, we don't have any checks.

Jira screenshot:



Postman documentation:
https://documenter.getpostman.com/view/25180838/2s93saaZ4D