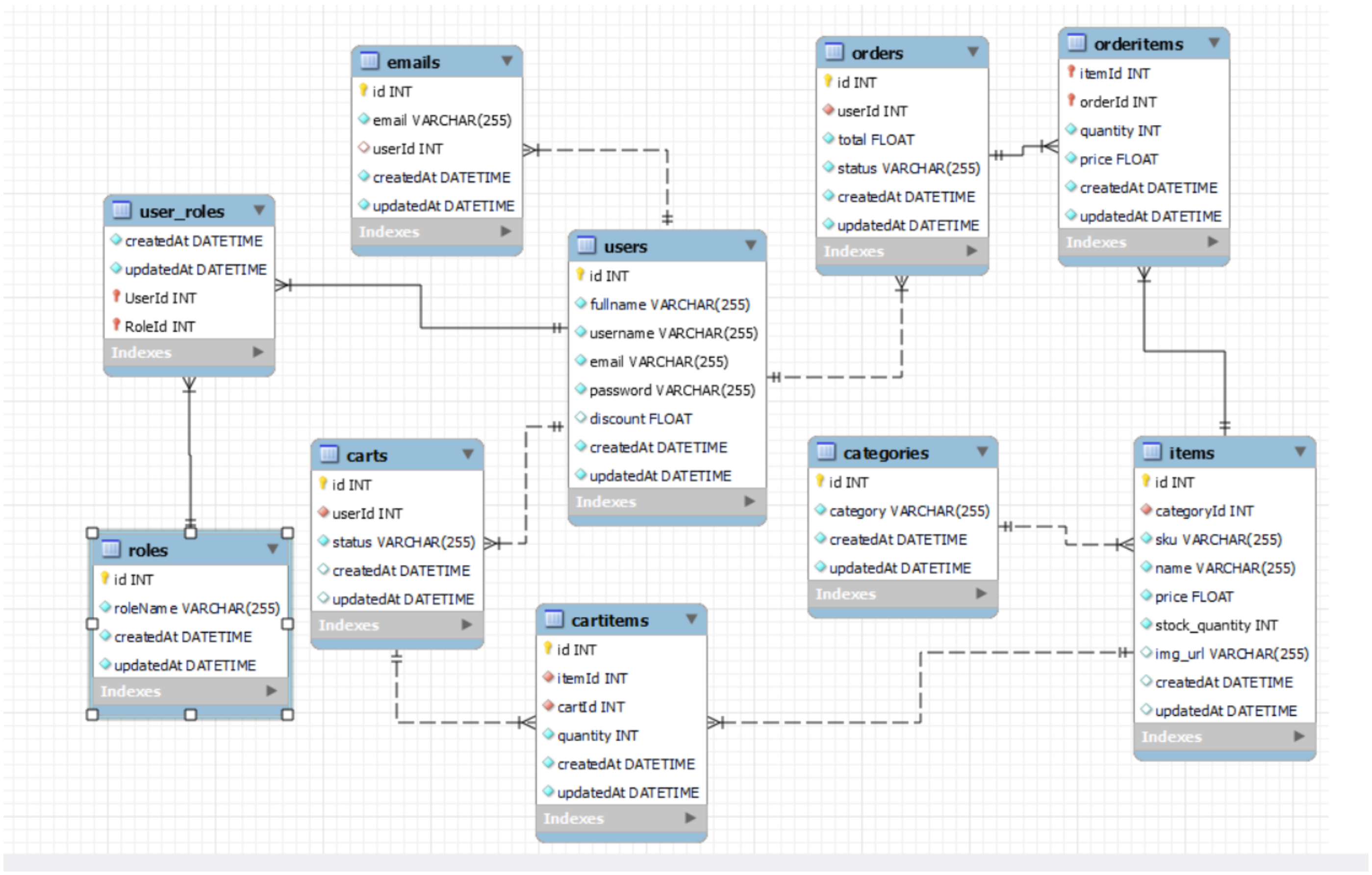
Retrospective Report

Fatin Izzati Hj.Mahari | BED1 Exam Project FT AUG22 | May – June 2023

Database EER Diagram



Explaination of database tables relationship

- **Users**
  User table has one-to-many relationship with Cart table where a User has one Cart, but a Cart belongs to one User. The User table also have many-to-many relationship with Role table through "user_roles" where a user can have many Roles, and a Role can be associated with many Users. The User table has a one-to-many relationship with Email table where a User can have many Emails, but each Email is associated with one User. Lastly, User table have one-to-many relationship with order table where one User can have many orders.

- **Roles**
  Role table have many-to-many relationship with User through "user_roles" table where a Role can be associated with many Users, and a User can have many Roles.

- **Orders**
  Order table have one-to-many with User table where an Order belongs to one User, but a User can have many Orders. The order table also have one-to-many relationship with OrderItem table where an Order can have many OrderItems, but each OrderItem is associated with one Order.

- **OrderItems**
  OrderItem table has many-to-one relationship with Item table where many OrderItem can be associated with one Item, but there can only be one item per OrderItem. The OrderItem table also has many-to-one with Order table where many OrderItem can be associated with one Order.

- **Items**
  Item table have one-to-many relationship with Category table where an Item belongs to one Category, but a Category can have many Items. The Item table also have many-to-many relationship with Cart table through CartItem table where an Item can be in many Carts, and a Cart can have many Items. Lastly, the Item table also have many-to-many relationship with Order table through OrderItem table where an Item can be in many Orders, and an Order can have many Items.

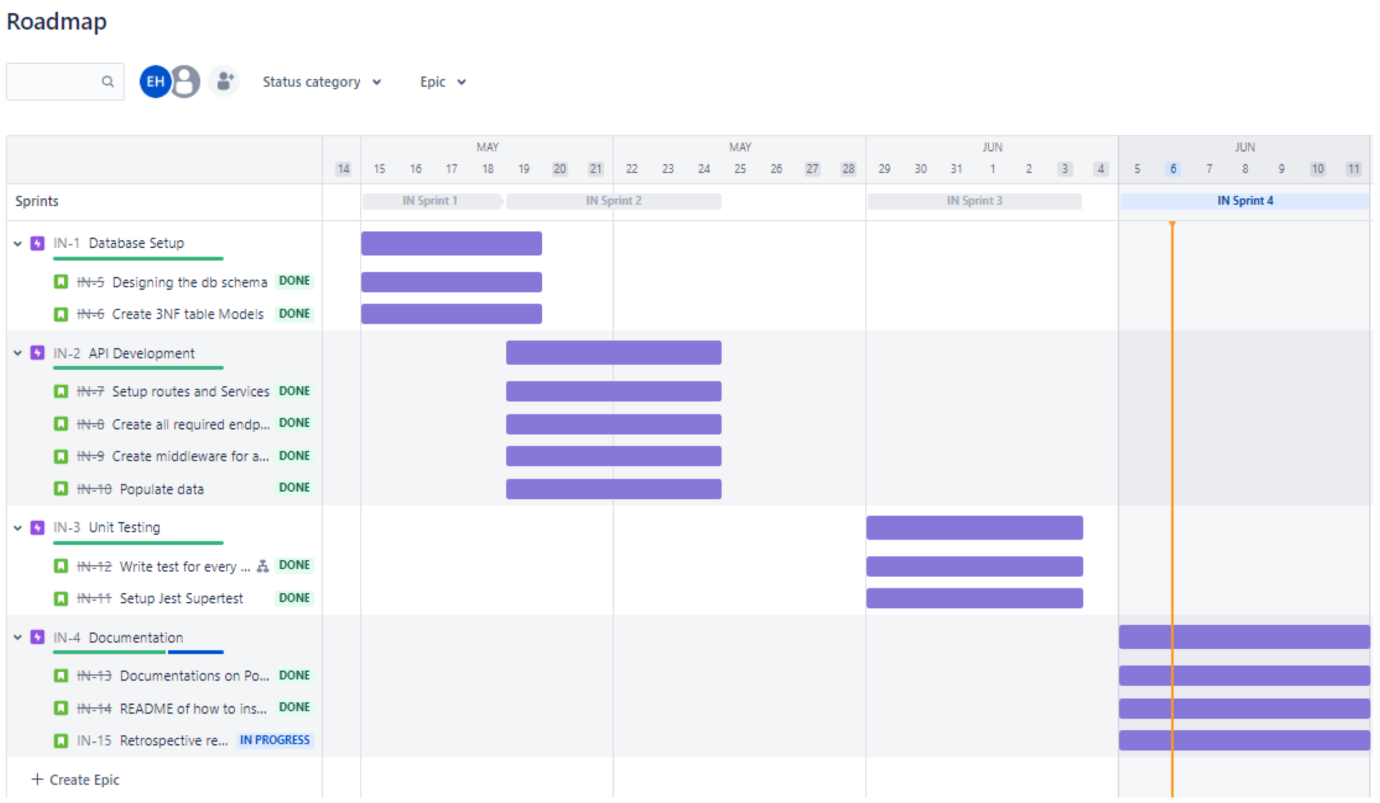
- **Emails**
  Email has many-to-one relationship with User table where many Email can be associated with one User.

- **Categories**
  Category table has one-to-many relationship with Item table where a Category can have many Items, but many Item belongs to one Category.

- **CartItems**
  CartItem table has many-to-one relationship with Cart table where many CartItem can be associated with one Cart. CartItem table also has many-to-one relationship with Item table where many CartItem can be associated with one Item.

- **Carts**
  Cart table has many-to-one relationship with User table where many Cart can be associated with one User. The Cart table also have one-to-many relationship with CartItem table where a Cart can have many CartItems, but each CartItem is associated with one Cart. Lastly, the Cart table also have many-to-many relationship with Item through CartItem where a Cart can have many Items, and an Item can be in many Carts.

Screenshot of the Jira Roadmap (Showing Epics and Sprints)



Sprints

IN-1 Database Setup
~~IN-5~~ Designing the db schema DONE
~~IN-6~~ Create 3NF table Models DONE

IN-2 API Development
~~IN-7~~ Setup routes and Services DONE
~~IN-8~~ Create all required endp... DONE
~~IN-9~~ Create middleware for a... DONE
~~IN-10~~ Populate data DONE

IN-3 Unit Testing
~~IN-12~~ Write test for every ... DONE
~~IN-11~~ Setup Jest Supertest DONE

IN-4 Documentation
~~IN-13~~ Documentations on Po... DONE
~~IN-14~~ README of how to ins... DONE
IN-15 Retrospective re... IN PROGRESS

Discussion of the progression of the project

The objective of this project was to design and implement a back-end stock-control and sales system for a warehouse that sells products directly to consumers from an item catalog. The system needed to align with the provided API design specifications, ensuring seamless integration with an external UX design company to host the interface of this system.

The initial phase of the project involved a comprehensive understanding and visualizing of project requirements. I utilized Jira Software to manage and track project tasks as per requirements, segregating the development process into four sprints. This approach not only ease organization but also enabled my tracking of progress in the process of developing.

In the first development stage, the focus was on designing and setting up a database, trying to ensure it will be in the Third Normal Form (3NF).

Upon completion of the database structure, the project transitioned into the API endpoint development phase. This required me to create endpoints that are aligned with the project requirements, ensuring each endpoint served its intended function.

Following the API endpoint development, the project progressed to the testing phase. I used Postman to perform thorough testing for each endpoint, confirming their necessary functions. I also created Jest tests and Supertest tests during this phase.

The final phase of the project to date has been the documentation process. I documented all endpoint scenarios using Postman, created instructions in the README file, and wrote this progress report.

With this I believe the back-end system for this project is fully developed and tested as the requirements suggested.

Challenges faced during the development of this project.

Throughout the development of this project, I faced several challenges. The first major obstacle was the vagueness in the requirements. Some requirements were clear and straightforward, while others were abstract and somewhat disorganized. This resulted in occasional misunderstandings due to the wording layout. Additionally, changes were made to the requirements after the project had begun, which added to the complexity and aggravation.

The second challenge was designing the relationships between database tables. The description and alteration of the API requirements resulted in modifications to the database structure throughout the development of the endpoints.

My approach to the API endpoint and database table creation also offered a challenge. Initially, I opted to create all the endpoints and tables without testing each one individually using Postman. Looking back, this approach proved to be a learning experience as it led to extensive debugging and a deeper understanding of Sequelize from frequently referring to its documentation.

Writing unit tests using Jest and Supertest was another area of uncertainty. I struggled with deciding whether to mock the entire test or not, given that it was named as "unit testing" , this baffled me.

Reflecting on the project, it has been a learning experience despite the challenges. If I had to do this assignment again, I would consider testing each API endpoint individually during the development process to minimize debugging and reconstructing codes later on. I also realized the value of clarity in project requirements and having the necessity of adapting to changes. Given the scope and difficulties of the project, collaborating with a team could have eased some of the challenges, a plan I am making for projects in the future.