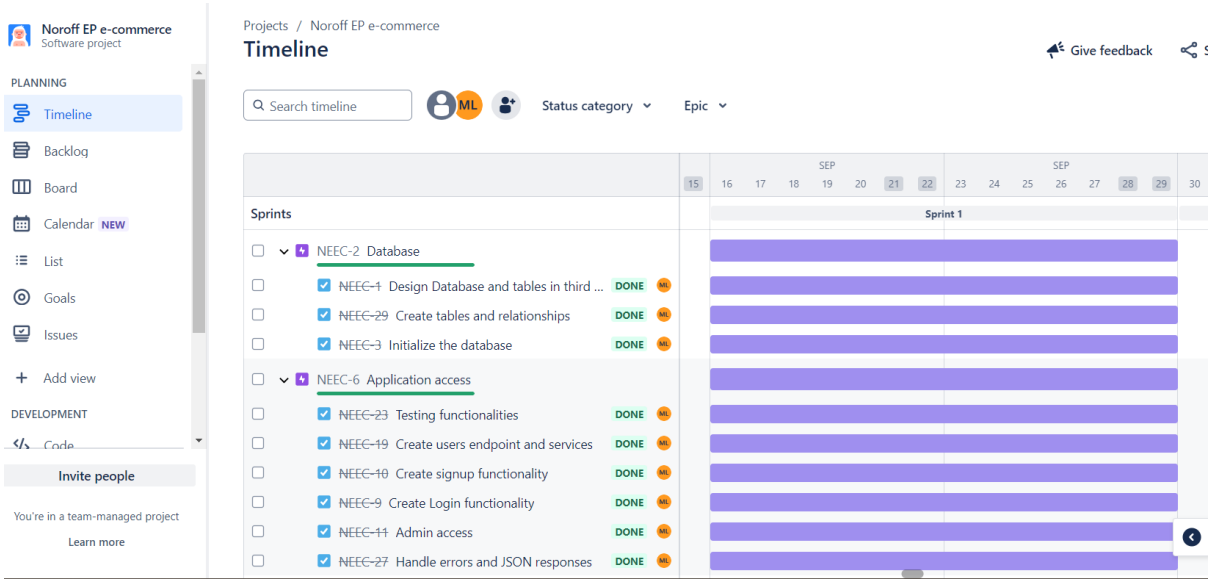
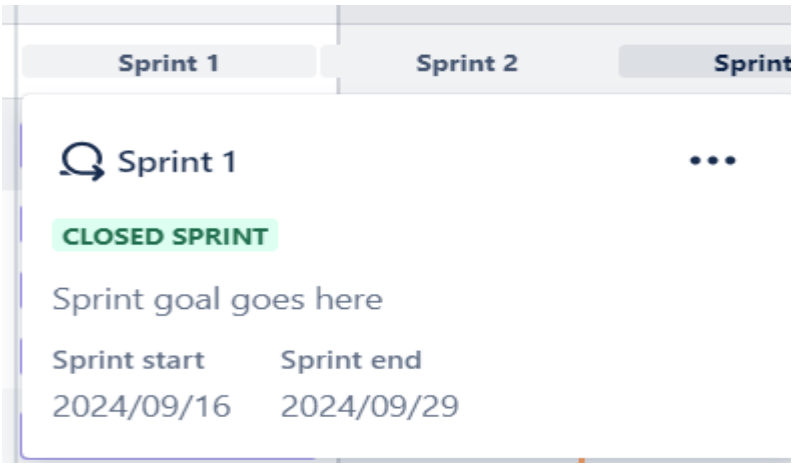
Reflection Report on the Exam Project

To initiate the project, I broke it down into into smaller, more manageable steps, prioritizing essential functionalities before focusing on features that enhance the user experience. The project was organized into distinct sprints, each lasting two weeks.

In the first sprint, I focused on two main ***epics:*** *Database and Application Access*, both of which were further divided into manageable tasks to ensure steady progress and smooth execution.

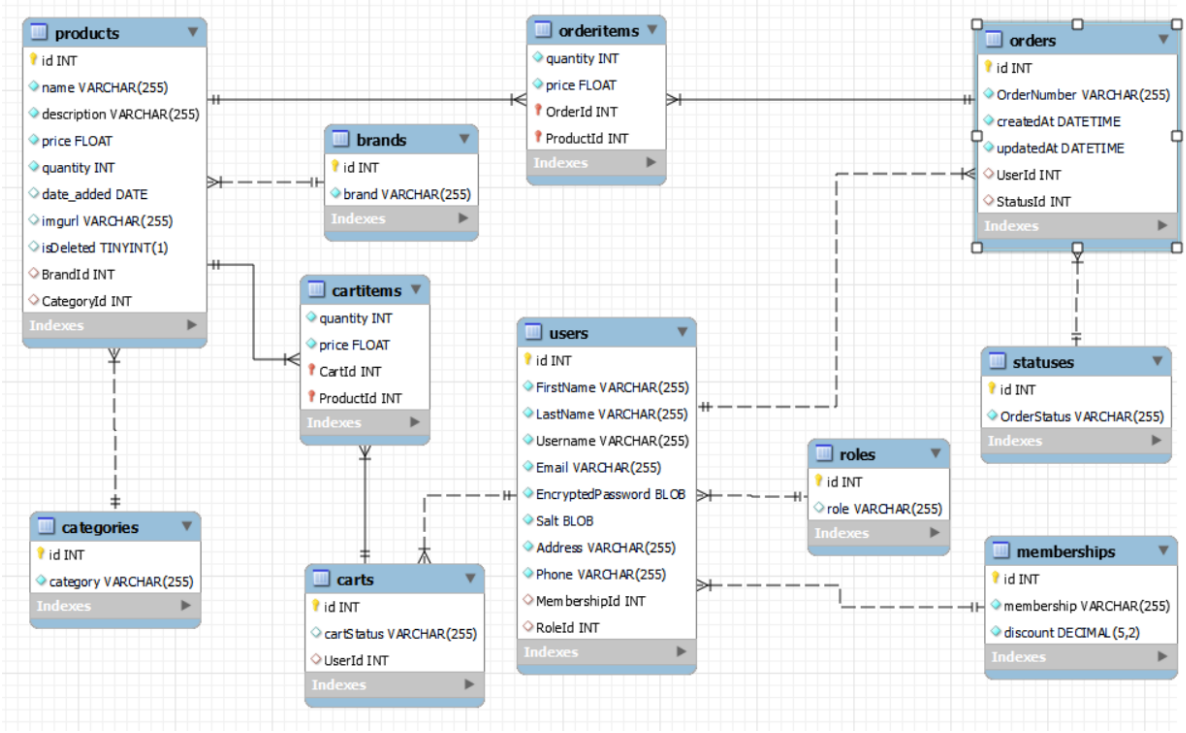


The first sprint started on September 16, 2024, and on September 29 was closed.



To commerce the project, I started by planning the database structure, ensuring that all tables adhered to the third normal form. Simultaneously, I created tables and defined their relationships to guarantee proper connectivity and consistent communication.

In the image below we can see a Database ERD where the tables' relationships are shown:



There are 11 tables in the ecommerce database:

- Products,
- Categories,
- Brands,
- Carts
- CartItems,
- Orders,
- Statuses,
- OrderItems,
- Users,
- Roles,
- Memberships

They all connected between each other through their own associations to communicate effectively.

Products table has:

One-To-Many relationships with Brands (a brand can have many products),

One-To-Many relationships with Categories (a category can have many products),

Many-To-Many relationships with Orders (through the OrderItems table as a junction table),

Many-To-Many relationships with Carts (through the CartItems table as a junction table).

Brands table has One-to-Many relationships with Products, where brands can have many products.

Carts table has:

One-to-One relationships with Users (a user has one cart),

Many-to-Many relationships with Products (through CartItems as the conjunction table),

One-to-Many relationships with CartItems (a cart can have multiple items).

CartItems table has:

Many-to-One relationships with Carts and Products, where many cart items can belong to the same cart and product.

Category table has:

One-To-Many relationships with Products table (a category can have many products).

Memberships table has:

One-to-Many relationships with Users (a membership can have many users).

Orders table has:

Many-to-One relationships with Users (each order belongs to one user),

One-to-Many relationships with OrderItems (an order can have many items),

One-to-One relationships with Status (an order has one status).

OrderItems table has:

Many-to-one relationships with Orders (an order can have many items),

Many-to-one relationships with Product (a product can be part of many orders).

Roles table has:

One-to-one relationships with Users (each user has one role).

Statuses table has:

One-to-Many relationships with Orders (a status can apply to multiple orders).

Users table has:

One-to-One relationships Roles (a user has one role),

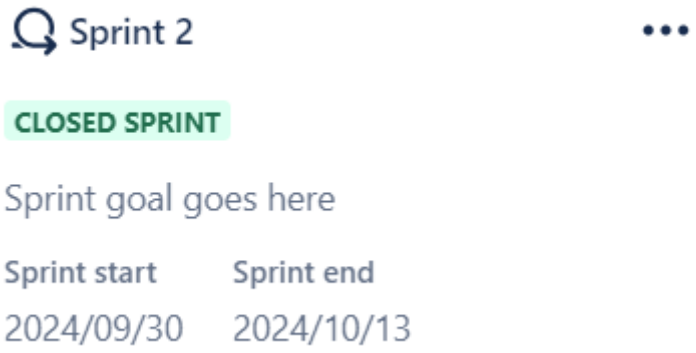
One-to-One relationships with Memberships (a user has one membership),

One-to-One relationships with Carts (a user has one cart),

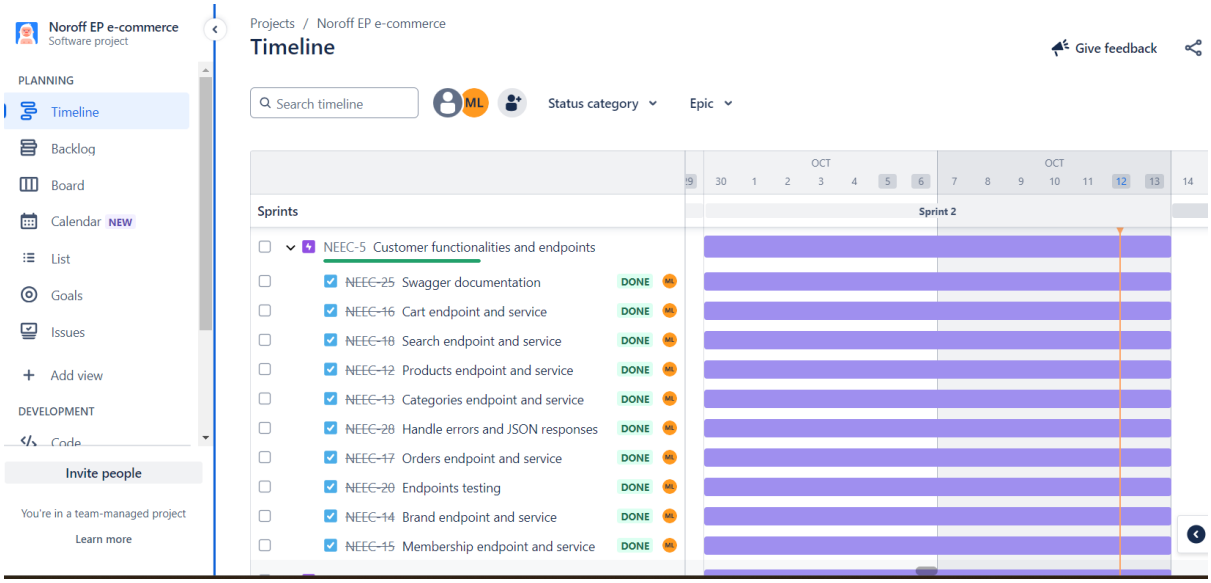
One-to-Many relationships with Orders (a user can have many orders).

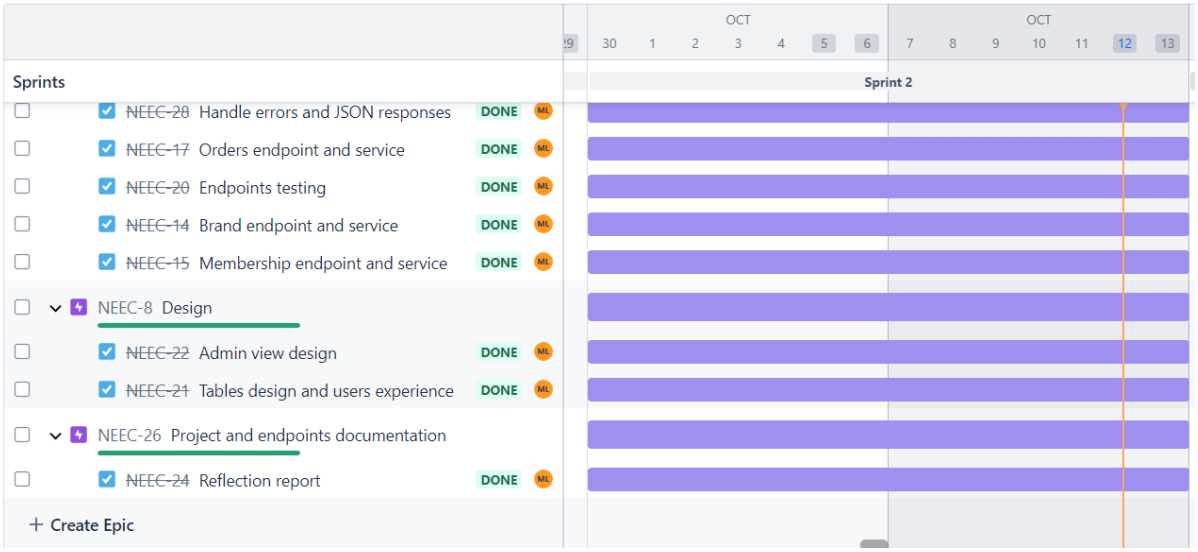
Next, I implemented the registration and login functionality. I developed a middleware function to restrict access to specific endpoints, so that only Admin user can access those endpoints. Additionally, I handled error responses and ensured consistent JSON responses across the whole application.

The second sprint started on September 30, 2024, and was closed on October 13, 2024.



There were 3 **epics**, and the number of **tasks** for each epic received different priority levels according to the end goal. More attention was given to user experience and testing functionality.





The Jira Timeline illustrates the successful completion of primary objectives. All tasks were executed efficiently, adhering to the predetermined deadlines.

One of the challenges I encountered during the exam project was search functionality. To get partial search working, I went to the forum Stack Overflow to gain insight into how to keep the query active since it must work even when only one of the search parameters is in use. In order to keep the search dynamic and return value when there is filtering, I found out that I can append the WHERE 1=1 condition and append queries for each parameter if it is “true” afterward. That gave me a lot of flexibility to use replacements and append them into the query result.

An example of .env file is presented below:

.env

```
ADMIN_USERNAME = "root"
ADMIN_PASSWORD = "P@ssword"
DATABASE_NAME = "ecommerce"
DIALECT = "mysql"
DIALECTMODEL = "mysql2"
PORT = "3000"
HOST = "localhost"
TOKEN_SECRET=
cea133f551cb77adca165827e5f52c51b1c6010addd392ef648442a470ea0e2d86a7c94fdd1656
89250a90c6c60d0013985ae972cc3dab074fadae7c42495381
```