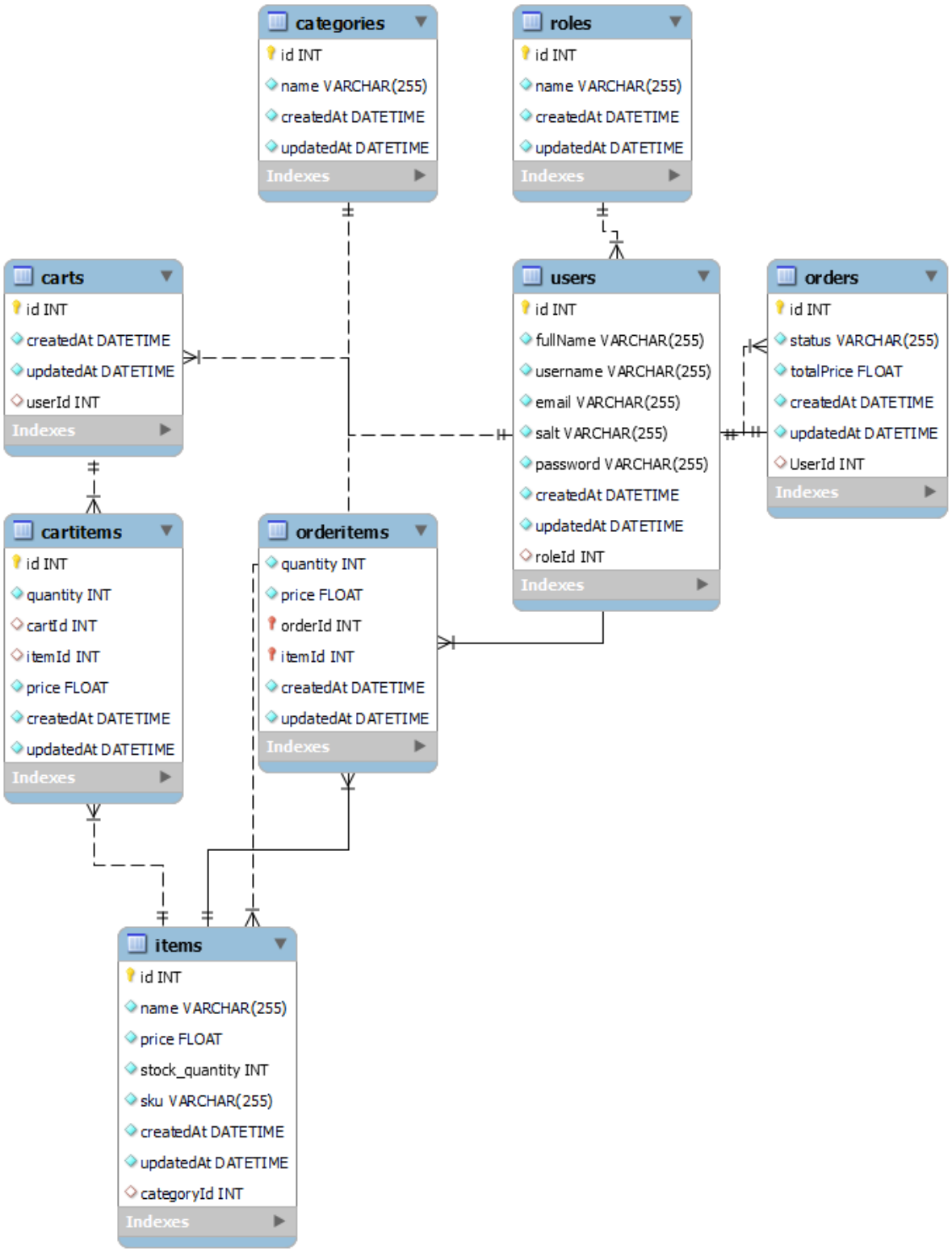
Retrospective report



Relationship explanation

User Table:

- Belongs to Role (One-to-One)
- Has One Cart (One-to-One)
- Has Many Orders (One-to-Many)

Role Table:

- Has Many Users (One-to-Many)

Cart Table:

- Belongs to User (One-to-One)
- Has Many CartItems (One-to-Many)
- Belongs to Many Items through CartItem (Many-to-Many)

CartItem Table:

- Belongs to Cart (One-to-One)
- Belongs to Item (One-to-One)

Item Table:

- Belongs to Category (One-to-One)
- Belongs to Many Carts through CartItem (Many-to-Many)
- Belongs to Many Orders through OrderItem (Many-to-Many)
- Has Many OrderItems (One-to-Many)

Category Table:

- Has Many Items (One-to-Many)

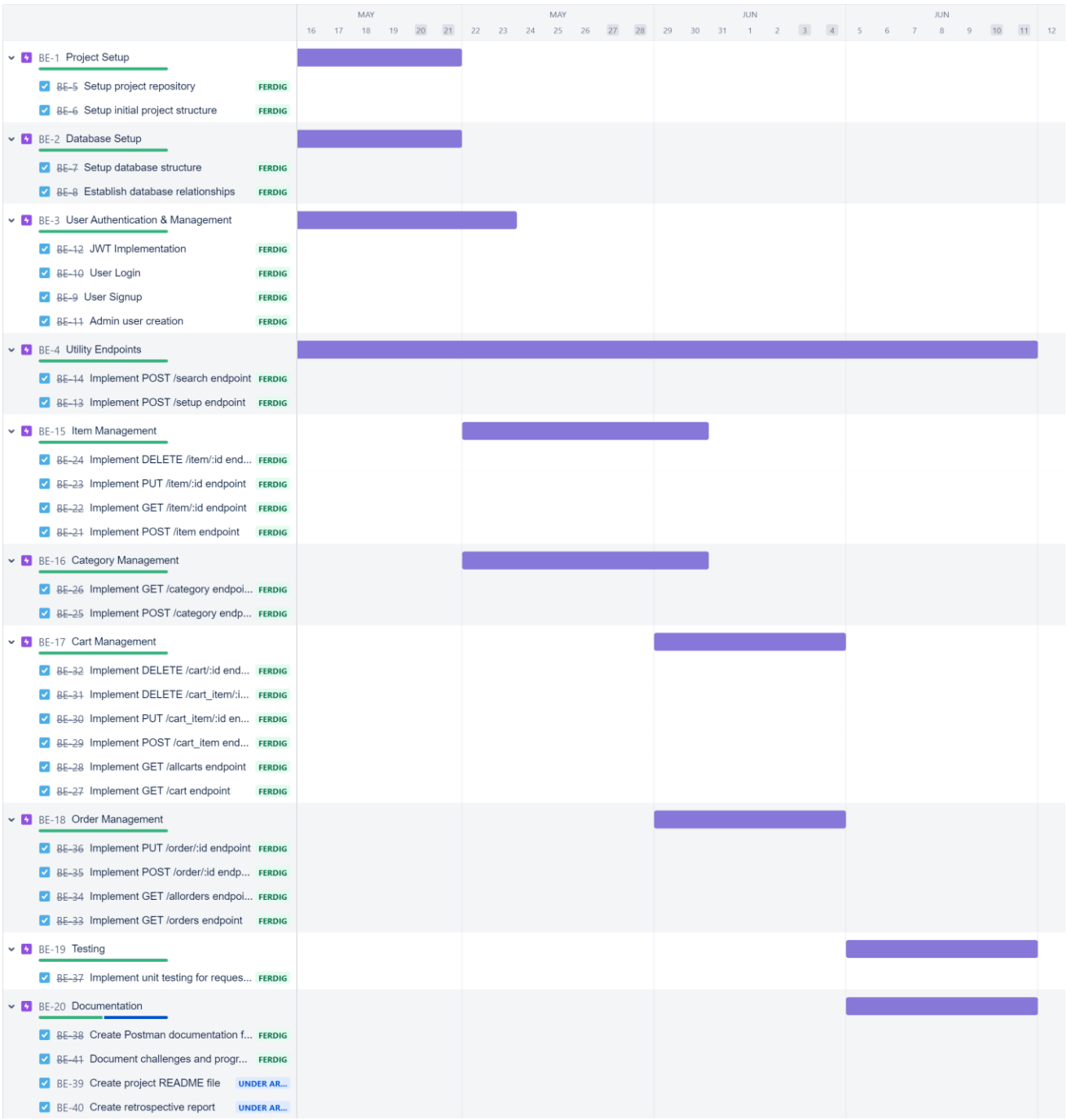
Order Table:

- Belongs to User (One-to-One)
- Belongs to Many Items through OrderItem (Many-to-Many)
- Has Many OrderItems (One-to-Many)

OrderItem Table:

- Belongs to Item (One-to-One)
- Belongs to Order (One-to-One)

JIRA Roadmap



Progression of the Project

The project began with an initial setup phase where the project repository and structure were established. The primary focus of the first week was setting up the database, figuring out where to start and how to go about it. I started laying out all the models to populate the database with correct tables and associations. After finishing the models and database I focused on the /setup endpoint to correctly populate the database with items, categories, and the admin user. I Then added the user authentication. A major key factor this week was the implementation of JWT token and assigning it to the user in the login.

Following the added JWT token I started the implementation of items and category-related endpoints. Most of the focus where on creating all the endpoints and comprehending them with the authentication process from the server. This was quite time consuming as they were the first endpoints I made, and the authentication process gave me some headaches.

Following the items and category I started developing the cart and order endpoints. Both the cart and order were complex tasks due to the relationships between different entities. The backend logic required careful consideration, especially for operations like increasing the desired purchase quantity or placing an order. All the tables needed to work accordingly with and/or without each other. Like items being decreased when orders were placed.

My last focus of coding was to make the unit test and search function work properly. I used the unit tests to validate the functionality of my API. These two areas went okay; the unit test was time consuming but eventually passed all the criteria and the /search endpoint meets the task's criteria and searches what needs to be found.

Lastly I made the documentation, Postman API documentation, retrospective report, and a detailed README file for the installation and functionality.

Challenges During the Development of this Project

One of my first challenges was designing and implementing the database schema. Given the various relationships between entities such as Users, Carts, Orders, Items, and Categories, establishing and managing these relationships required some try and error. Overall, I got a better understanding of the Sequelize ORM and MySQL.
Together with the database schema the utility endpoint /setup gave me an interesting challenge, specifically the database populating from the Noroff API call. This was a great implementation as it gives great experience in handling API calls and managing database entries.

Implementing user authentication and JWT tokens was another challenging task. Understanding and correctly using encryption, salting, and token generation techniques required some brainpower, not something we have been focusing a lot on in the modules, but a great learning curve here and feel like I have good knowledge about it now.

For me, the most complex part of the project was the cart and order operations. Implementing the backend logic to manage the cart items and updating the item stock when orders are placed was tricky and required a lot of errors before getting it right. I also had some difficulties with the POST /order/:id. At first, I did not really understand the purpose of the endpoint, so I made a /checkout endpoint for checking out the entire cart. After some questions and answering in the Teams chat, I figured out that this was the checkout endpoint only for a single item. I chose to keep both endpoints as they have a different purpose.

Lastly, testing the all the API and making sure all endpoints are working as expected was a time-consuming process. Creating the tests using Jest and Supertest was challenging and not something I have used too much time on during this course. Rewarding as it helped ensure the reliability of the API.

I also want to add the retrospective report as a challenging part for me. I am not the best writer and often has problems articulating correctly when writing, especially in English.

Overall, despite these challenges, the project was great for my learning, and I liked the task.

Exam Project

Utility

POST /setup

http://localhost:3000/setup

Populates the database with initial data and creates an admin user.

POST /search

http://localhost:3000/search

Searches for items in the database based on specified criteria.

Auth

POST /signup

http://localhost:3000/signup

Registers new users by providing their information.

Body raw (json)

json

```
{
    "fullName": "Test 1",
    "username": "Test1",
    "email": "test@mail.com",
    "password": "Test123"
}
```

POST /login

http://localhost:3000/login

Allows users to log in with their credentials and obtain a JWT token.

AUTHORIZATION

Secret	<secret>
Algorithm	<algorithm>
Is Secret Base64Encoded	<is-secret-base64encoded>
Payload	<payload>
Add Token To	<add-token-to>
Header Prefix	<header-prefix>
Query Param Key	<query-param-key>
Header	<header>

Body raw (json)

json

```
{
  "username": "Test1",
  "password": "Test123"
}
```

Cart

GET /cart

http://localhost:3000/cart

Retrieves cart for the logged-in user.

HEADERS

Authorization	Bearer {token}

GET /allcarts

http://localhost:3000/allcarts

Retrieves all existing carts, including their items and user information. (ADMIN)

HEADERS

Authorization	Bearer {token}

POST /cart_item

http://localhost:3000/cart_item

Adds an item to the user's cart.

HEADERS

Authorization Bearer {token}

Body raw (json)

```
json

{
    "itemId": 140,
    "quantity": 1
}
```

PUT /cart_item/:id

http://localhost:3000/cart_item/:id

Updates the quantity of a specific item in the user's cart.

HEADERS

Authorization Bearer {token}

PATH VARIABLES

id

Body raw (json)

```
json

{
    "quantity": 2
}
```

DELETE /cart_item/:id

http://localhost:3000/cart_item/:id

Removes an item from the user's cart.

HEADERS

Authorization Bearer {token}

PATH VARIABLES

id

DELETE /cart/:id

http://localhost:3000/cart/:id

Deletes all items from the user's cart.

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

Item

GET /items

http://localhost:3000/items

Retrieves all items from the database.

POST /item

http://localhost:3000/item

Adds a new item to the database. (ADMIN)

HEADERS

Authorization	Bearer {token}

Body raw (json)

json

```
{
  "name": "Hockey Skates",
  "price": 100,
  "stock_quantity": 10,
  "sku": "SG143",
  "categoryId": 7
}
```

PUT /item/:id

http://localhost:3000/item/:id

Updates an existing item based on its ID. (ADMIN)

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

Body raw (json)

json

```
{
  "name": "Updated Item",
  "price": 150,
  "stock_quantity": 5,
  "sku": "SKU123",
  "categoryId": 7
}
```

DELETE /item/:id

http://localhost:3000/item/:id

Deletes an item from the database. (ADMIN)

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

Category

GET /categories

http://localhost:3000/categories

Retrieves all categories from the database.

POST /category

http://localhost:3000/category

Adds a new category to the database. (ADMIN)

HEADERS

Authorization	Bearer {token}

Body raw (json)

```json
{
    "name": "Gaming"
}
```

PUT /category/:id

http://localhost:3000/category/:id

Updates the name of a category based on its ID. (ADMIN)

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

Body raw (json)

```json
{
    "name": "Updated Category Name"
}
```

DELETE /category/:id

http://localhost:3000/category/:id

Deletes a category from the database. (ADMIN)

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

Order

GET /orders

http://localhost:3000/orders

Retrieves orders for the logged-in user.

HEADERS

Authorization	Bearer {token}

GET /allorders

http://localhost:3000/allorders

Retrieves all existing orders, including items and user information. (ADMIN)

HEADERS

Authorization	Bearer {token}

POST /checkout

http://localhost:3000/checkout

Place an order for the entire user's cart

HEADERS

Authorization	Bearer {token}

POST /order/:id

http://localhost:3000/order/:id

Places an order for an item in the user's cart.

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

PUT /order/:id

http://localhost:3000/order/:id

Updates the status of an order.

HEADERS

Authorization	Bearer {token}

PATH VARIABLES

id

Body raw (json)

json

```
{
    "status": "Complete"
}
```