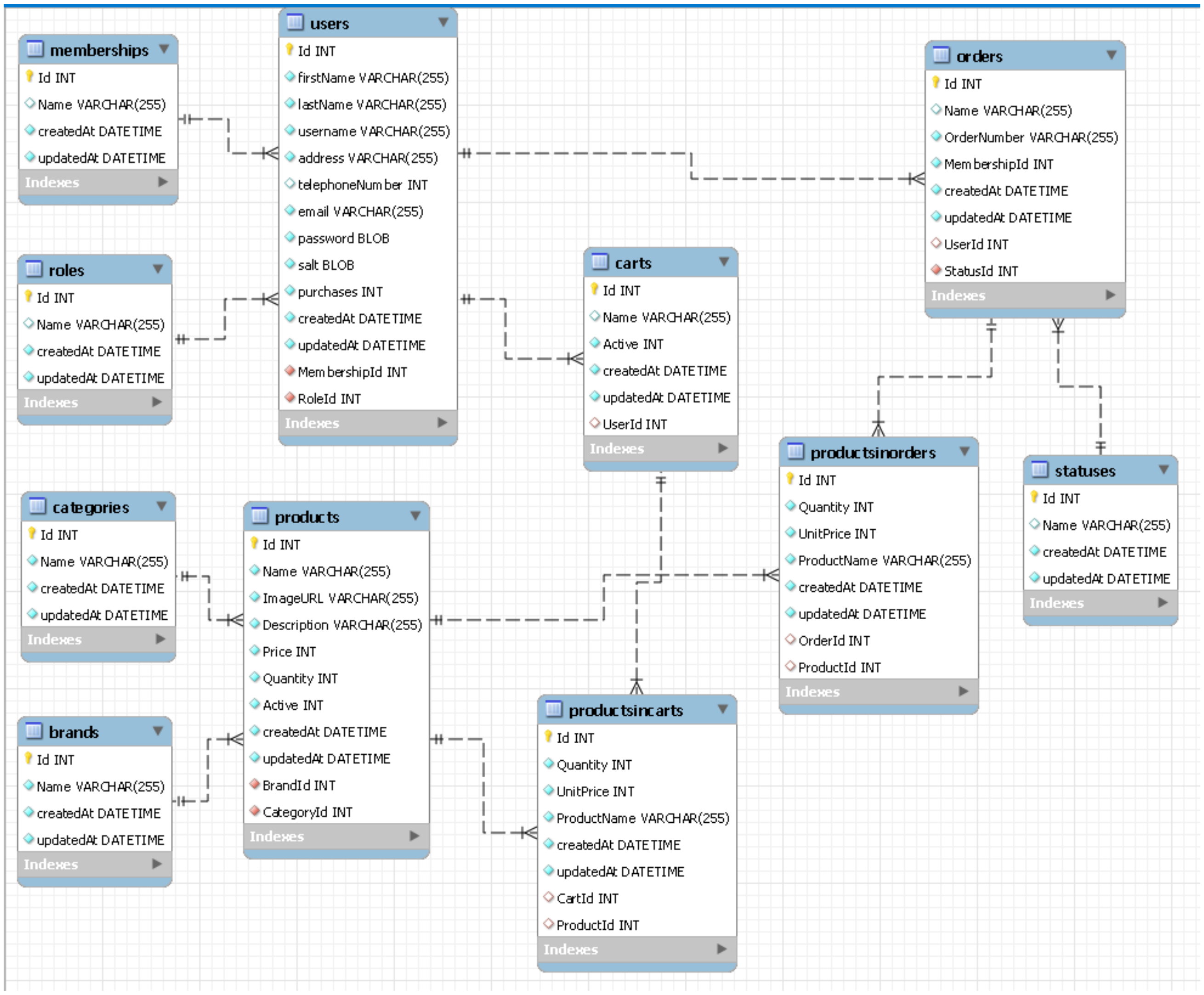
Reflection Report

Exam project BED1 David Hansen

Database:



Relationships:

A *user* can have only one *membership*, but a *membership* can have many *users* –therefore this is a one-to-many relationship.

A *user* can have only one *role*, but a *role* can have many *users* – therefore this is a one-to-many relationship.

A *user* can have many *orders*, but an *order* can have only one *user* – therefore this is a one-to-many relationship.

A *user* can have many *carts*, but a *cart c*an have only one *user* – therefore this is a one-to-many relationship.

An *order* can have only one *status*, but a *status* can have many *orders* - therefore this is a one-to-many relationship.

An *order* can have many *productsInOrder*, but *productsInOrder* can have only one *order* - therefore this is a one-to-many relationship.

A *cart* can have many *productsInCart*, but *productsInCart* can only have one *cart* - therefore this is a one-to-many relationship.

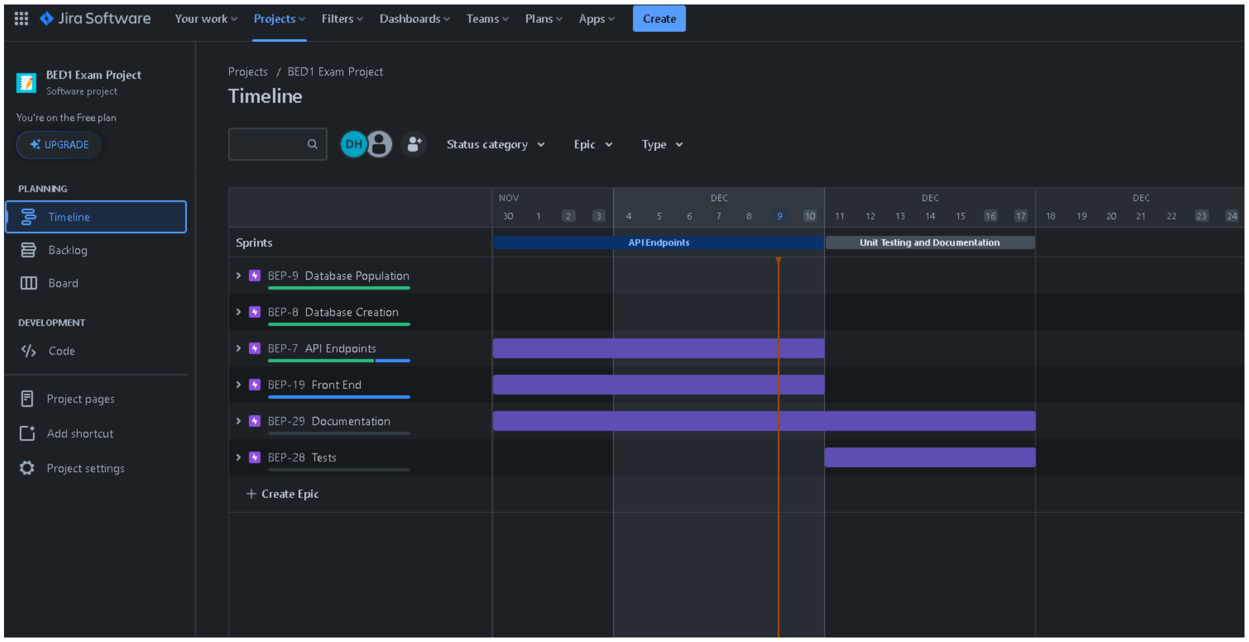
ProductsInOrder can have only one *product*, but a *product* can have many *productsInOrder* - therefore this is a one-to-many relationship

ProductsInCart can have only one *product*, but a *product* can have many *productsInCart* - therefore this is a one-to-many relationship

A *product* can only have one *category*, but a *category* can have many *products* - therefore this is a one-to-many relationship

A *product* can only have one *brand*, but a *brand* can have many *products* - therefore this is a one-to-many relationship

Jira:



Sadly, I took the screenshot after I completed the first sprint :/

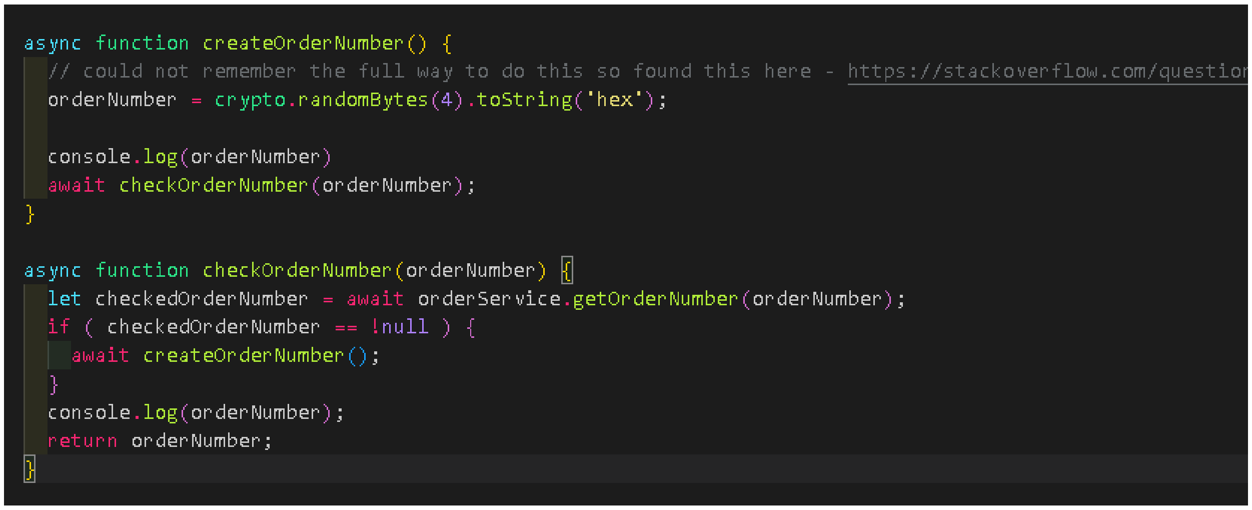
Reflection:

Discussion: I was working on this project alone, so I did not discuss it with anyone.

Challenges:

1. The first challenge came when I was supposed to populate the database. It did not populate the database in the order I was providing them. And since the product page is dependent on the category table and the brand table, this was an issue. Solved it by making async functions and putting them in a try {.

2. I had a problem with this function. And could not figure it out.

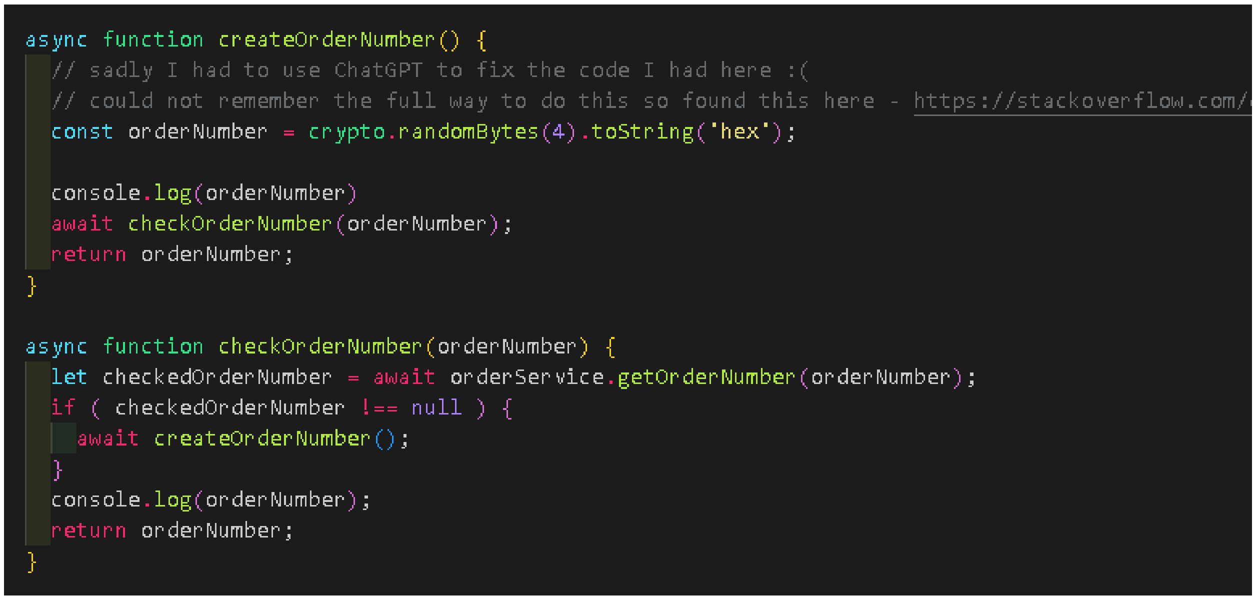


```
async function createOrderNumber() {
  // could not remember the full way to do this so found this here - https://stackoverflow.com/questior
  orderNumber = crypto.randomBytes(4).toString('hex');

  console.log(orderNumber)
  await checkOrderNumber(orderNumber);
}

async function checkOrderNumber(orderNumber) {
  let checkedOrderNumber = await orderService.getOrderNumber(orderNumber);
  if ( checkedOrderNumber == !null ) {
    await createOrderNumber();
  }
  console.log(orderNumber);
  return orderNumber;
}
```

Used ChatGPT that resulted in this small adjustment



```
async function createOrderNumber() {
  // sadly I had to use ChatGPT to fix the code I had here :(
  // could not remember the full way to do this so found this here - https://stackoverflow.com/
  const orderNumber = crypto.randomBytes(4).toString('hex');

  console.log(orderNumber)
  await checkOrderNumber(orderNumber);
  return orderNumber;
}

async function checkOrderNumber(orderNumber) {
  let checkedOrderNumber = await orderService.getOrderNumber(orderNumber);
  if ( checkedOrderNumber !== null ) {
    await createOrderNumber();
  }
  console.log(orderNumber);
  return orderNumber;
}
```

3. I had a problem with JWT and passing it when I was using the web page. Was working perfectly with postman. But not on the web page as it was not being passed. Spent a little time here looking through the studies and the old files I have from the studies. Until I used ChatGPT and got the idea to use cookies. I had to modify the middleware function and it worked perfectly.

4. The admin front end proved challenging. But once I checked through my files and I figured out I could use axios to get all I needed from my other API endpoints, it was easier. A lot of trial and error but it worked.

5. The Search API proved to be the most challenging since nothing worked like I wanted. I had to use ChatGPT to help me here to pass along the search results to the page.

What I learned:

I always liked to use console.log() to figure out things and why things went wrong. During this exam I used it all the time. One of the most important tool I have 😛

Try { } catch – really learned the importance of this to make the code work and so the server is not going to crash constantly.

Async functions and await. This is really important to make shure a code is finished before the nest code runs.

This exam really helped me see the whole picture of the programming and the way it connects. And all the errors and problems I will get. And the importance of consistent code and error handling. Now at the end I am afraid to change the inconsistencies whit that little time left in case I mess it up :/

References:

I used ChatGPT when I got really stuck and could not figure it out. It was not always that helpful but might have given me ideas as to how to move forward. Or editing the code it gave me.

The checkbox and making it unclickable. - Samiya Khan -

https://stackoverflow.com/questions/6411662/how-do-i-make-a-checkbox-unclickable-without-it-being-greyed-out-in-the-browse

Email validator - Sandeep Singh -
https://singh-sandeep.medium.com/email-validation-in-node-js-using-email-validator-module-20b045b0c107

Cookie usage – expressjs - https://expressjs.com/en/api.html#res.cookie