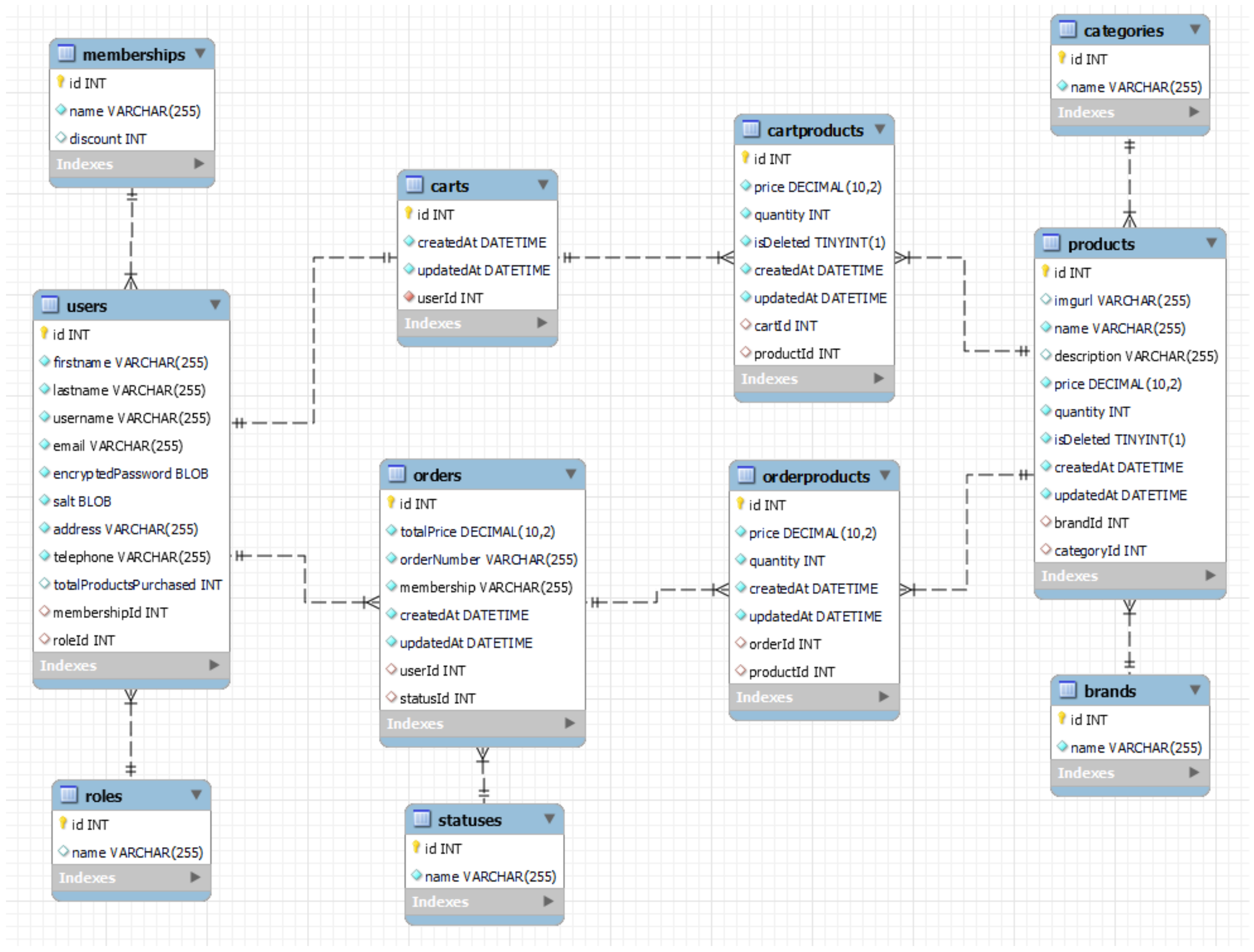
# REFLECTION REPORT

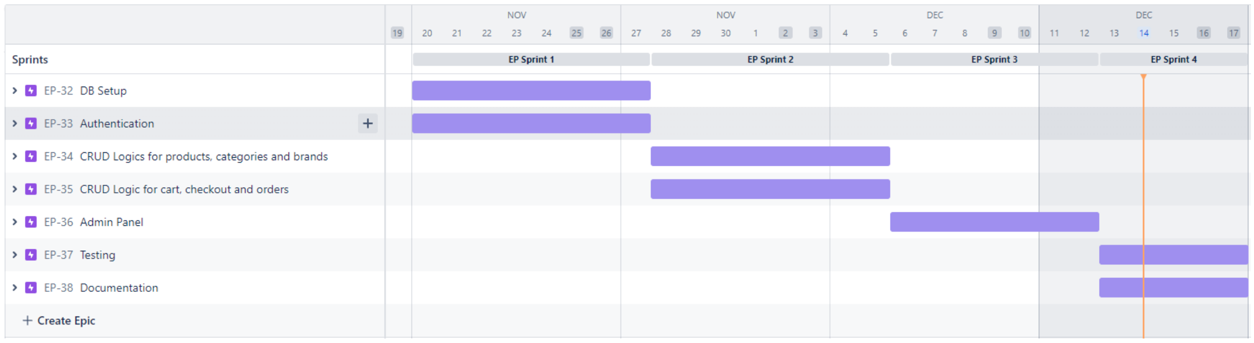
## Database ERD:



- **Users to Carts:** A user can only have one cart, and a cart can only have one user therefore this is a one-to-one relationship.
- **Users to Orders:** A user can have many orders, but an order can only have one user therefore this is a one-to-many relationship.
- **Memberships to Users:** A user can only have one membership, but a membership can have many users therefore this is a one-to-many relationship.
- **Roles to Users:** A user can only have one role, but a role can have many users therefore this is a one-to-many relationship.
- **Statuses to Orders:** A status can have many orders, but an order can only have one status therefore this a one-to.many relationship.
- **Orders to Orderproducts:** An order can have many orderproducts, but an orderproduct can only have one order therefore this is a one-to-many relationship.
- **Products to Orderproducts:** An orderproduct can only have one product, but a product can have many orderproducts therefore this is a one-to-many relationship.
- **Products to Orders:** Orders and products use orderproducts as a link to simulate a many-to-many relationship.

- **Brands to Products:** A product can only have one brand, but a brand can have many products therefore this is a one-to-many relationship.
- **Categories to Products:** A product can only have one category, but a category can have many products therefore this is a one-to-many relationship.
- **Products to Cartproducts:** A product can have many cartproducts, but a cartproducts can only have one product therefore this is a one-to-many relationship.
- **Carts to Cartproducts:** A cart can have many cartproducts, but a cartproduct can only have one cart therefore this a one-to-many relationship.
- **Carts to Products:** Carts and products use cartproducts as a link to simulate a many-to-many relationship.

## Roadmap:



## Progression:

I'm very happy with the progression planning. The timelines I set for the different sprints were very accurate to how it played out.I estimated approximately two weeks for the back-end development, followed by less than a week each for the front-end development and for testing and documentation.

After planning the database relationships and creating the Model files, I focused on developing the authentication middleware. This was crucial for enabling continuous testing of all CRUD operations throughout the project.

The second part was the most time consuming. Here I started with the CRUDs in all the router files and service files. Starting with the handlers like product, category and brand cause I knew I would need some of these for the Cart and order logic. It took me a lot of time with all the validation because I wanted this to be thorough and flexible. If the request had multiple invalid properties or values I wanted this to show in one error so that the user will see all the problems right away and not just the first problem the first validation catched.

After I was done with all the CRUDs I started with the front-end part. I met some challenges with authorization and this will be explained in the challenges section.

The last part of the project was testing, documentation and some debugging and small improvements of the code.

## Challenges:

Authorization:
Initially, I used the 'Authorization' header for token-based authorization. However, this approach required fetching the API and manually setting the header in the admin.js. This method proved to be difficult with the  '/admin/…' endpoints since I couldn't fetch them the same way I did with the other endpoints. To streamline the process, I switched to using cookies for storing tokens, as they are automatically included in the header and was a lot simpler. To ensure security, I researched and implemented 'httpOnly: true' and 'sameSite: strict' attributes. If this app was to be released I would have changed it to https instead of http to make it more secure.

TotalPurchasedProducts:
One consideration was whether to create a new table for tracking the total number of products each user purchased. Ultimately, I decided to add this as a property in the users table. This approach, while simpler and less code-intensive, effectively met the project's needs without the overhead of managing an additional table.