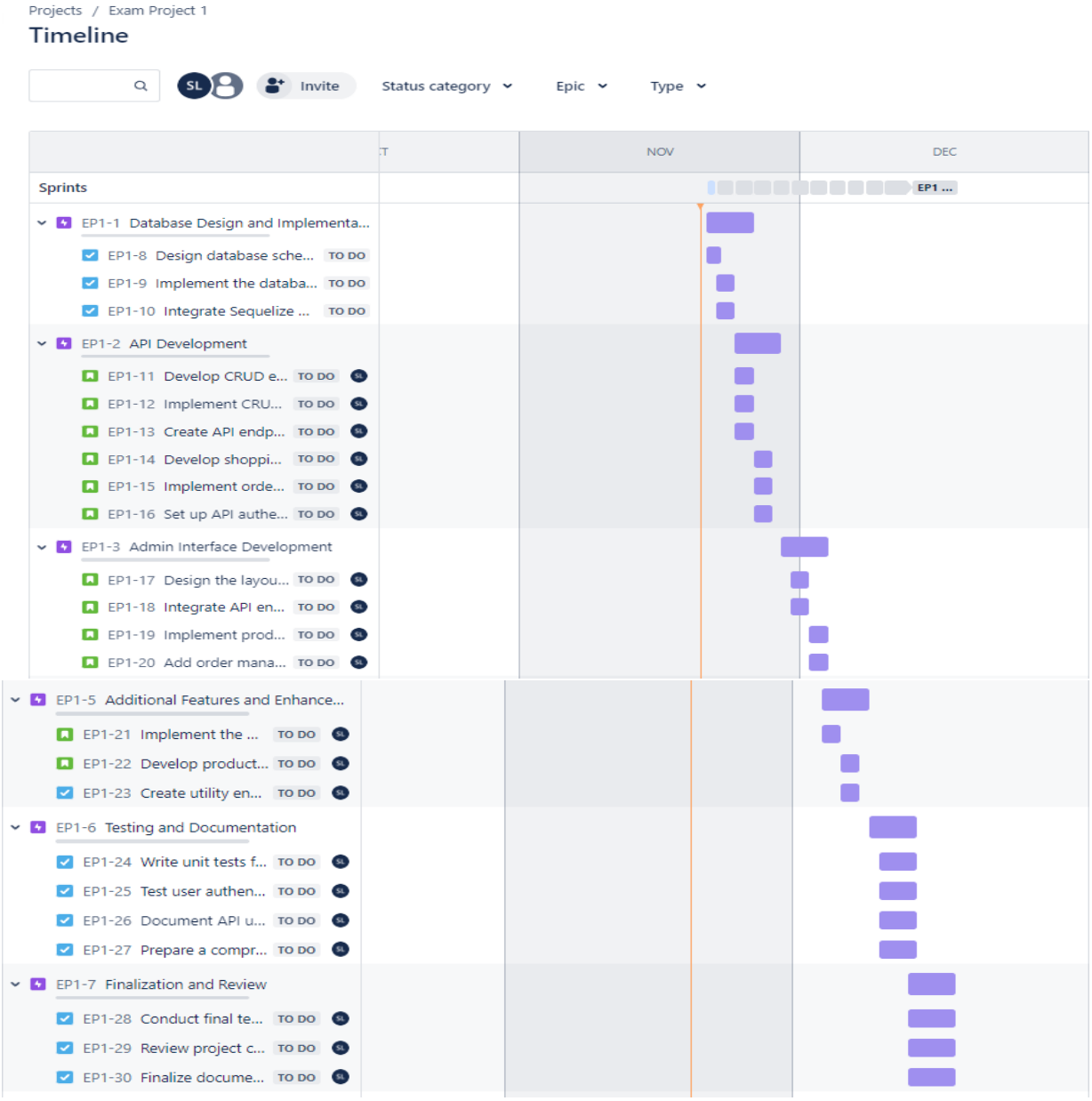
Jira Roadmap



EER Diagram

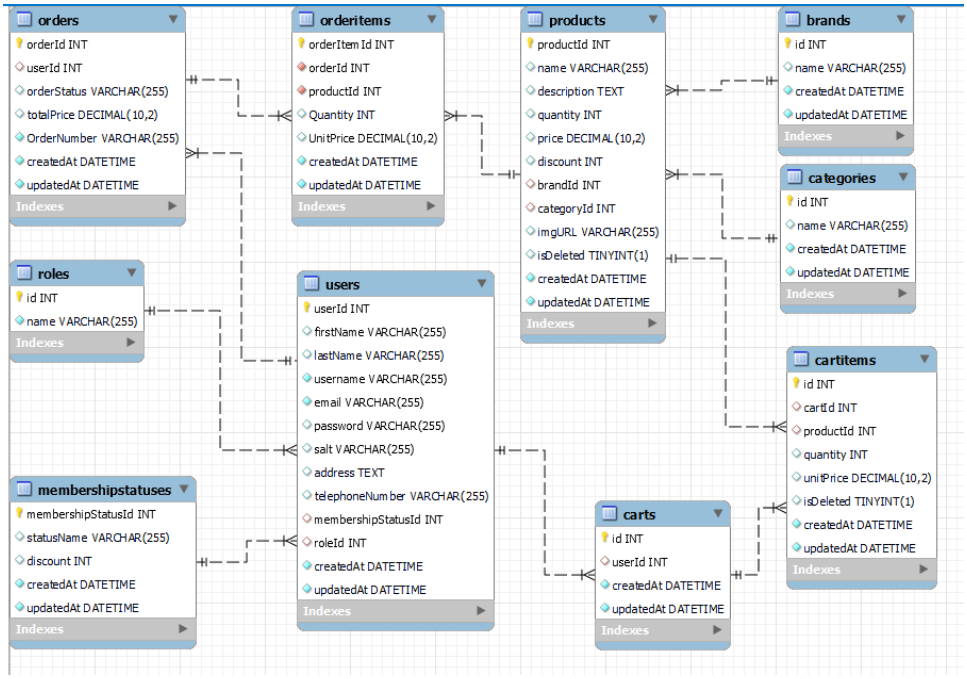


Table relationships

- **MembershipStatus to Users**: One-to-Many
- A membership status can apply to many users, but each user has one membership status

- **Users to Roles**: One-to-Many
- Each user can have one role, but roles can have many users

- **Users to Carts**: One-to-Many
- A user can have many carts, but each cart is associated with one user

- **Carts to CartItems**: One-to-Many
- A cart can have many items, but each cart item is a part of one cart

- **Users to Orders**: One-to-Many
- A user can have multiple orders, but each order is associated with one user

- **Orders to OrderItems**: One-to-Many
- An order can have many items, but each order item belongs to one order

- **Products to OrderItems**: One-to-Many
- A product can be in many order items, but each order item refers to one product

- **Products to CartItems**: One-to-Many
- A product can be in multiple cart items (across different carts).

- **Products to Categories**: Many-to-One
- Many products can belong to one category.

- **Products to Brands**: Many-to-One
- Many products can belong to one brand.

Reflection report

I decided to make a cookie out of the JWT token as we had learned about cookies, and I wanted to make one that served a functional purpose for my application. I also wanted to make sure that if a person did not want to use cookies or maybe if their browser did not support them, they could use the normal way of adding the token in the authorization header. I also had a “isMember” middleware but since there were no “must be member, no admin” routes, it was enough to use the authMiddleware as a global “member or admin” check.

The hardest part certainly was the cart, and especially the checkout where so many moving parts had to come together. And because of so many moving parts it was necessary with the use of transactions, as we update membership status, reduce stock, soft-delete cart items and creating order and we can't really have any of those actions take place if one of them fail. I decided to mark the cart items as soft-deleted after the checkout, so the user could continue shopping for other items afterwards. Also added a discount field where a product could have a flat percentage discount that were calculated in the checkout. If I were to design the front-end part of the application, I would reflect on that on the products page to calculate it from the total price. Added the most comments to this part as it helped me navigate through the task.

I chose to use ‘crypto’ over ‘bcrypt’ as it is a core module of node.js removing the need for additional packages and potential compatibility issues. And introducing external

module from npm I read had the possibility of malicious code injection which would defeat the original security objective. I also went with not using passport as the hashing was done with crypto and the session via JWT.

Not sure what the best practice for the product testing part is, as creating a test product with the test category makes the test category not able to be deleted due to the product assigned to it, and the product only able to be soft deleted. Easy enough to clean up later with `db.sequelize.sync({ force: true })` and initialize the database again but would be interested to know if I were supposed to do it another way. I also had to add db.sequelize.close() after the tests or I would receive the error that jest did not exit after the tests. Alternative was to use –forceExit or –detectOpenHandles but felt they were more loopholes then fixes.

I did not want to focus too much on the front-end part, so instead of adding modals to the category and brand “add” buttons I went with a simple prompt to fill in the names and an alert if something went wrong. Also added an isAdmin flag to the auth login for a simplified way of determining if the user is admin for the front-end login, which I think should not compromise the security as all the routes are protected with server-side middleware.

Seemed wrong to add the admin routes to swagger as they were simply used to render the front-end part, but I didn't find any way to hide them eighter, so I'm curious to know if this was the correct approach.

Fun and challenging task. I can’t say I followed the Jira roadmap to a dot as sometimes things took more time and other things took less time. I also wanted to be done quicker than the planned epics and sprints to have more time looking over the project and fixing things where it was needed. This was possible because I was alone on this project but as a team, I can really see the benefits of using Jira. I learned from my roadmap though and would in the future add a greater deal of time for the reviewing and error fixing sprints.