Readme

```
# REST API - Course Assignment

# Application Installation and Usage Instructions
```

```
change the env variables if its needed
```

```
A MySQL Database called “StockSalesDB” is to be created for this web application.
Use the following SQL script to create an “admin” Database User with all database privileges:
CREATE USER 'admin'@'localhost' IDENTIFIED WITH mysql_native_password BY 'P@ssw0rd';
ALTER USER 'admin'@'localhost' IDENTIFIED WITH mysql_native_password BY 'P@ssw0rd';
GRANT ALL PRIVILEGES ON database_name.\* TO 'admin'@'localhost'
```

```
npm install
npm start
npm run test - to run test cases (**tests**/Todos.test.js)
```

```
# Environment Variables
```

```
HOST = "localhost"
ADMIN_USERNAME = "admin"
ADMIN_PASSWORD = "P@ssw0rd"
DATABASE_NAME = "StockSalesDB"
DIALECT = "mysql"
PORT = "3000"
TOKEN_SECRET='V8Z4H4pOj3qB61Xb1K94F3rNpP3oqPf6GZU6ml1L6b2fzR1M9X'
```

```
# Additional Libraries/Packages
```

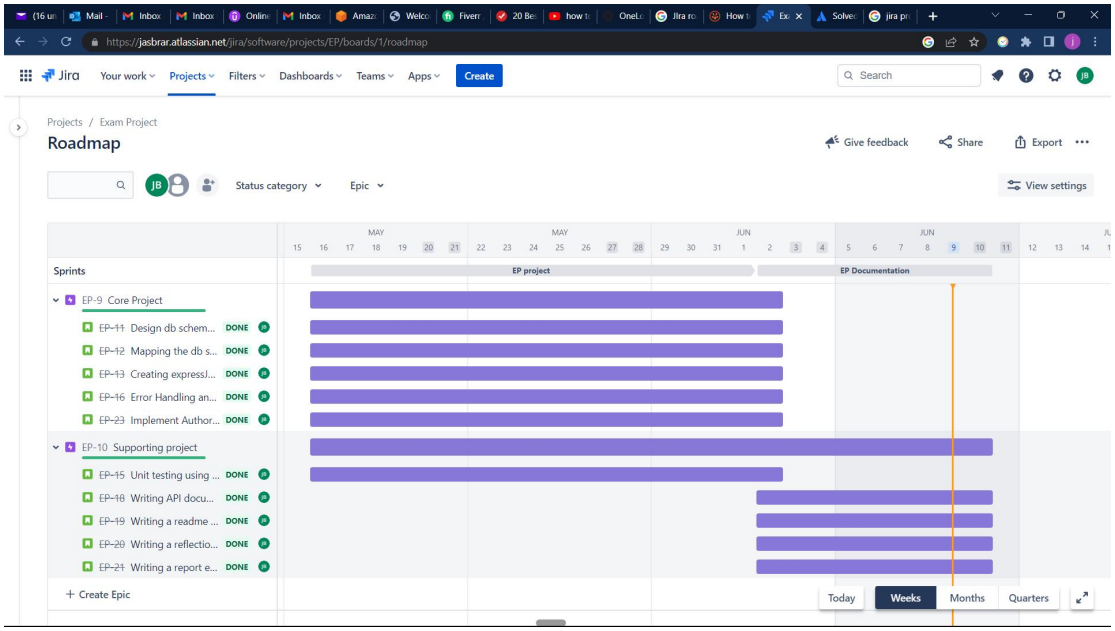
```
    "supertest": "^6.3.3"
    "jest": "^29.5.0",
    "bcrypt": "^5.1.0",
```

```
# NodeJS Version Used
```

```
v16.16.0
```

```
# POSTMAN Documentation link
```

```
https://documenter.getpostman.com/view/25399698/2s93sc4CKp
```


https://jasbrar.atlassian.net/jira/software/projects/EP/boards/1/roadmap
Jira
Your work
Projects
Filters
Dashboards
Teams
Apps
Create
Search
Projects / Exam Project
Roadmap
Give feedback
Share
Export
Status category
Epic
View settings
MAY
JUN
Sprints
EP project
EP Documentation
EP-9 Core Project
EP-11 Design db schem... DONE
EP-12 Mapping the db s... DONE
EP-13 Creating expressJ... DONE
EP-16 Error Handling an... DONE
EP-23 Implement Author... DONE
EP-10 Supporting project
EP-15 Unit testing using ... DONE
EP-18 Writing API docu... DONE
EP-19 Writing a readme ... DONE
EP-20 Writing a reflectio... DONE
EP-21 Writing a report e... DONE
Create Epic
Today
Weeks
Months
Quarters

Explanation of the relationships

One-to-Many Relationship:

A category can have multiple items. An item can belong to only one category.

One user can have multiple orders. An order belongs to one user.

A user can have only one role. A role can be associated with multiple users.

One-to-One Relationship:

A user can have only one cart. A cart belongs to one user.

Many-to-Many Relationship:

An order can contain multiple items. An item can be associated with multiple orders.

A cart can contain multiple items. An item can be associated with multiple carts.

Project Retrospective

During the development of the project, the progression followed a structured approach. Initially, the team started by designing the database schema using MySQL and mapping it to Sequelize ORM, which facilitated the management of database interactions. The next step involved creating the ExpressJS backend APIs using the Sequelize models to handle data operations. The APIs were designed to cater to the project requirements, allowing CRUD operations on various entities, such as users, orders, and items. Additionally, authentication and authorization mechanisms were implemented to secure the APIs.

1. Integration with Sequelize and MySQL: One of the main challenges encountered was integrating Sequelize ORM with MySQL. Configuring the Sequelize models, defining associations between entities, and handling complex queries required careful attention to ensure data consistency and performance. Debugging and resolving issues related to database connections, query optimizations, and ORM-specific errors added complexity to the development process.

2. Authentication and Authorization: Implementing a secure authentication and authorization system posed challenges. Designing and integrating authentication mechanisms like JWT (JSON Web Tokens) or OAuth, handling user roles and permissions, and ensuring secure access to API endpoints required careful consideration of security best practices.

3. Error Handling and Logging: Managing and handling errors effectively throughout the application was a challenge. Implementing robust error handling mechanisms, logging errors and exceptions, and providing meaningful error messages to clients were essential for debugging, monitoring, and maintaining the application.

StockSalesDB

Items

POST Insert Item

http://localhost:3000/item

Accessible to admin users only. Adds a new item to the database with the specified category.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
    "name": "toy",
    "price": "20",
    "stockQuantity": "4",
    "sku": "TK122",
    "categoryId": 3
}
```

PUT Edit Item

http://localhost:3000/item/3

Accessible to admin users only. Updates an existing item in the database with the provided ID. Returns an error for invalid IDs.

AUTHORIZATION Bearer Token

Token <token>

Body raw (json)

```json
{
  "name" : "tv",
  "price" : 1000,
  "stockQuantity" : 2,
  "categoryId" : 3
}
```

GET View Items

http://localhost:3000/items

Returns all items in the database, including their categories. Accessible to guest users without authentication.

DELETE Delete Item

http://localhost:3000/item/3

Accessible to admin users only. Deletes an item from the database with the provided ID. Returns an error for invalid IDs.

AUTHORIZATION Bearer Token

Token <token>

Category

POST Insert Category

http://localhost:3000/category

Accessible to admin users only. Adds a new category to the database. Returns an error message for any errors.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```
json

{
  "name" : "prop"
}
```

GET View Category

http://localhost:3000/category

Returns all categories in the database. Accessible to guest users without authentication.

PUT Edit Category

http://localhost:3000/category/3

Accessible to admin users only. Updates the name of a category with the provided ID. Returns an error for any errors.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```
json

{
  "name" : "vehicle"
}
```

DELETE Delete Category

http://localhost:3000/category/9

Accessible to admin users only. Deletes a category from the database with the provided ID. Returns an error for invalid IDs and if the category has associated items.

AUTHORIZATION Bearer Token

Token	<token>

Cart

POST Insert Cart

http://localhost:3000/cart_item

Accessible to logged-in registered users. Adds items to the user's cart. The item references the items table, and the purchase price is stored in the cart item table.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "item_id" : 4,
  "quantity": 2
}
```

GET View Cart

http://localhost:3000/cart

Accessible to logged-in registered users. Returns the cart for the logged-in user, extracted from the JWT.

AUTHORIZATION Bearer Token

Token	<token>

GET View All Cart

http://localhost:3000/allcarts

Accessible to the admin user. Returns all carts, including items and user information.

AUTHORIZATION Bearer Token

Token	<token>

PUT Edit Cart

http://localhost:3000/cart_item/4

Accessible to logged-in registered users. Changes the desired quantity of a specific item in the user's cart. Updates the item's stock quantity once the order is placed.

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

```json
{
  "quantity": 2
}
```

DELETE Delete Cart Item

http://localhost:3000/cart_item/4

Accessible to logged-in registered users. Deletes an item from the user's cart using the item ID.

AUTHORIZATION Bearer Token

Token	<token>

DELETE Delete Cart

http://localhost:3000/cart/1

Accessible to logged-in registered users. Deletes all items from the user's cart using the cart ID.

AUTHORIZATION Bearer Token

Token	<token>

Oder

POST Insert Order

http://localhost:3000/order/6

Accessible to logged-in registered users. Creates a new order in the order table, with the item referencing the items table. Adjusts the item's stock level in the items table.

AUTHORIZATION Bearer Token

Token	<token>

GET View Order

http://localhost:3000/orders

Accessible to logged-in registered users. Returns orders for the logged-in user. User information is extracted from the JWT.

AUTHORIZATION Bearer Token

Token	<token>

GET View All Orders

http://localhost:3000/allorders

Accessible to the admin user. Returns all orders, including items and user information, regardless of order status.

AUTHORIZATION Bearer Token

Token	<token>

PUT Edit Cart Copy

http://localhost:3000/order/2

Accessible only to the admin user. The admin user can update the order status for any user. Valid status values are "In Process," "Complete," and "Cancelled."

AUTHORIZATION Bearer Token

Token	<token>

Body raw (json)

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

POST Setup

http://localhost:3000/setup

StartFragment

This endpoint is used for the initial database population. This endpoint should only populate the database if no records exist in the items table.

EndFragment

Body raw (json)

```json
{
    "username": "oshan",
    "password": "1234"
}
```

POST SignUp

http://localhost:3000/signup

Used to register new users. Returns a specific error message for invalid information.

Body raw (json)

```json
{
    "username": "test",
    "email": "test@gmail.com",
    "password": "1234"
}
```

POST Login

http://localhost:3000/login

Allows registered users to log in and receive a JWT token. Returns a token for valid logins and a specific error for invalid logins.

Body raw (json)

```json
{
```

```
    "username": "test",


    "password": "1234"
}
```

POST Search

http://localhost:3000/search

StartFragment

This endpoint is used to search for items in the database. Depending on the search criteria, items or categories should be searched from the database and returned as a JSON object

EndFragment

Body raw (json)

json

```
{
    "type": "item",
    "keyword": "Laptop"
}
```

stocksalesdb

carts

Column	Type	Null	Default	Links to	Comments	Media type
cart_id *(Primary)*	int(11)	No				
user_id	int(11)	No		users -> id		
total_price	decimal(10,2)	No				
created_at	datetime	No				
updated_at	datetime	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	cart_id	1	A	No	
user_id	BTREE	No	No	user_id	1	A	No	

cart_items

Column	Type	Null	Default	Links to	Comments	Media type
item_id *(Primary)*	int(11)	No		items -> id		
cart_id *(Primary)*	int(11)	No		carts -> cart_id		
quantity	int(11)	No				
price	decimal(10,2)	No				
created_at	datetime	No				
updated_at	datetime	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	item_id	1	A	No	
				cart_id	1	A	No	
cart_id	BTREE	No	No	cart_id	1	A	No	

categories

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
name	varchar(255)	No				
created_at	datetime	No				
updated_at	datetime	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	8	A	No	
name	BTREE	Yes	No	name	8	A	No	

items

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
name	varchar(255)	No				
price	float	No				
stock_quantity	int(11)	No				
sku	varchar(255)	No				
created_at	datetime	No				
updated_at	datetime	No				
category_id	int(11)	No		categories -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	32	A	No	
category_id	BTREE	No	No	category_id	16	A	No	

orders

Column	Type	Null	Default	Links to	Comments	Media type
order_id *(Primary)*	int(11)	No				
user_id	int(11)	No		users -> id		
created_at	datetime	No				
updated_at	datetime	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	order_id	2	A	No	
user_id	BTREE	No	No	user_id	2	A	No	

order_items

Column	Type	Null	Default	Links to	Comments	Media type
item_id *(Primary)*	int(11)	No		items -> id		
order_id	int(11)	No		orders -> order_id		
quantity	int(11)	No				
price	decimal(10,2)	No				
status	varchar(255)	No				
created_at	datetime	No				
updated_at	datetime	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	item_id	1	A	No	
order_id	BTREE	No	No	order_id	1	A	No	

roles

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
name	varchar(255)	No				
created_at	datetime	No				
updated_at	datetime	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	3	A	No	
name	BTREE	Yes	No	name	3	A	No	

users

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	int(11)	No				
username	varchar(255)	No				
email	varchar(255)	No				
encrypted_password	blob	No				
salt	blob	No				
created_at	datetime	No				
updated_at	datetime	No				
role_id	int(11)	Yes	*NULL*	roles -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	3	A	No	
username	BTREE	Yes	No	username	3	A	No	
role_id	BTREE	No	No	role_id	3	A	Yes	