Online MarketPlace Documentation

Summary :

* This is the Application used to do the CRUD operations of the products. You can add, get all product, get the product by Id, update the product and delete the product.
* Need to create two database to go with the online marketplace application.
  + online\_marketplce\_productdetails
  + online\_marketplce\_productdetails\_test

# Authentication & Authorization Endpoints:

* /signup - For create a user
* /login - To create a JWT token when the user login.

# Application Endpoints:

* GET /products/ - Retrieve a list of all products.
  + To get all the products from the database table as a slice of struct values.
* POST /products/ - Create a new product.
  + Create a new product on the online marketplace application, the create endpoint also takes the inputs to create a single product.
* GET /products/:id/ - Retrieve a detailed view of a single product.
  + Get the product by the specific ID from the product table.
* PUT /products/:id/ - Update a single product.
  + Used to update the product details using the unique ID of the product.
* DELETE /products/:id/ - Delete a single product.
  + Delete the particular product from the product table using the ID of the product
  + As we are using a gorm, delete endpoint does the soft delete and it update the timestamp in deletedAt column. But the record didn’t retrieved using get and getall endpoint.

# JWT Authentication:

* Created a JWT authentication token using username and password. And stored the token once the user login.
* Once you logged in means need to give the JWT token on the bearer token for each Application API endpoint.
* Here, I’ve also created an encrypted password and stored it in the database.

# Input Validation:

* For an API request I’ve validated the input field that you mentioned in the requirement document.
* Also I’ve checked the length of the input greater than 200 character.

# Unit Testcase:

* I’ve created a unit test case for createProduct only, Because of less timing I’ve only created createProduct test case with multiple scenarios. That includes fail scenarios as well.
* Also I've created a test DB for testing API functionalities with diff input data.

# Containerization:

* Due to the less time I can’t able to create a docker container for the application now. But I can create a docker container for the application.

# Postman Collection:

* I’ve also created a postman collection for all the endpoints with the sample input .