Order Management System

About:

Small application written in NodeJS+Angular to demonstrate order management system.

Working Flow:

Front End-

As, a user I should be able to create a new order -

1. I have to log in to the application using email and password. In the backend, the user will be authenticated and return the response as per the authentication. A dummy token will be sent in response when the authentication is successful. All the requests further made in the process will send the jwt token and will be checked in backend if its valid or not.

2. Once the user has logged in successfully, he will be landed on home page where he can see the Products, Orders, Health menu items in navigation bar.

3. When user clicks on Products, he is able to see the products cards. When he clicks on the card, he will be taken to product details page, where he can see the available quantity of product, price of product and a button to buy.

4. When user clicks on the buy button, the user will be prompted to enter the authentication pin. Once user enters the authentication pin, the pin verification will happen in backend and once successful verification the amount of the product will be deducted from the user's credits and order is created in database with 'CREATED' state.

5. Once the order is created, user will be notified with the notification on the screen 'Order Created Successfully'.

As, a user I should be able to view all the order -

1. As logged in user, to see all the orders which I have placed. I have to click on the Orders menu on the navigation bar and it will show the orders page.

2. This page will show all the orders placed by me. It will show the list of orders based on the latest updated one's on the top. It also shows the status, order id, product, amount, timestamp of the order.

3. Also, user has given access to view the order details and cancel the order.

As a user, when I click on logout -

1. As a logged in user, when I click on the Log Out menu, I should be able to logged out from the application and able to see the login screen.

2. Once logged out, I should not be able to perform any operations on UI and also should not be able to call any API's in backend.

Backend-

Following are the important modules of the application

1. server/db/models.js - This file has all the initialization of the database connections and database models. It uses the Sequelize ORM node module for the database connection, authentication and creation of tables and relationships in database.
2. server/db/init-db.js - This file will create the mock data in the database. It creates the mock users and mock products.

3. server/app.js - It creates the express server. It is responsible for authentication, authorization and providing API services for the front end. It has following API's

   a. /api/health - It returns the health of the server and the uptime of the server

   b. /api/order/create - This POST api takes the data from the front end and creates the new order in the database for the requested user.

   c. /api/order/cancel/:id - This GET api will check the passed order is present in database or not and will update the status of the order to 'CANCELLED'.

d. /api/order/status/:id - This GET api will return the current status of the passed order.
e. /api/products - This API will returns the list of all the products from the database
f. /api/product/:id - This API will return the product details of the passed product id from the database
g. /api/orders/:userId - This API will return all the products created by passed user from the database.
h. /api/payments/:userId - This API will return all the payments done by user from the database.
i. /api/order/:id - This API will return the order details of the passed order id
j. /api/user/authenticate - This POST API takes the user authentication details -email and password. It checks the validates the user with the passed authentication details and returns the user details will dummy jwt token in response when authentication is successful, else it throws Authentication Failed.

4. server/services/order-service.js - This service listens for the following events:
   a. create_order - This service will create order for the user. It takes userId, productId, authPin. It checks if the passed user is present in database or not. If not present, it will throw error `User not present`. If user is present, it will check if the product is present in the database or not. If product is not present it will throw error, 'Product Not Present'. If product is present, it will check the quantity of the product. If product quantity is 0 then it will return 'Product Out ofStock' error. When all the above conditions satisfy, it will create new Order in the database with the status 'CREATED' and will call 'process_payment' event of the payment-service. When payment is processed it will return the created order id to the calling function. After 30 mins, this will call the event'check_and_update_order_status'.
   b. 'check_and_update_order_status': This event will get the order based on the passed orderId. It will check if the order status is 'CONFIRMED', if it is then it will update the order status to 'DELIVERED' in the database. Otherwise, it will print that cannot update the order status in database.

c. ‘cancel_order’: This event will cancel the order based on the passed orderId. This will check if the order status is ‘CONFIRMED’ or ‘CREATED’ and then it will update the order status to ‘CANCELLED’ in the database.

d. ‘check_order_status’: This event will return the order status of the passed orderId. It checks for the passed order id in the database and return the Order status.

5. server/services/payment-service.js: This service will listen for the below events:

   a. process_payment: This event will first check the authentication pin passed by user matches with the authentication pin stored in database of the same user. If it does not match it will create a payment record entry in database with the payment details like - orderId, userId and status as ‘CANCELLED’ and return will error ‘AUTHENTICATION_FAILED’. If the authentication pin matches, it will check user has sufficient credits available for the payment. If not available it will update the record in database with ‘CANCELLED’ and return error ‘INSUFFICIENT CREDITS’ to user. If all the above criteria satisfies, then it will deduct the credits from the user account and will create order entry with ‘CONFIRMED’ status.