Engineering Design Doc

Authored by: IAmInevitable

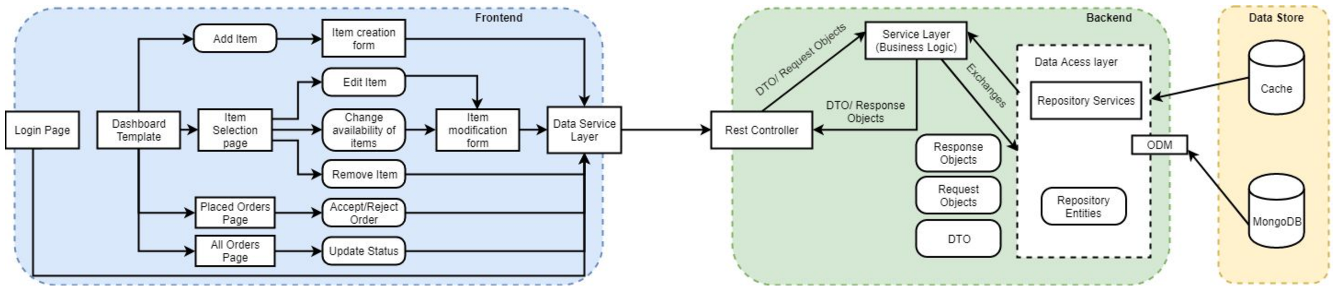
Intro & Goal:

Our goal is to build a Restaurant App for QEats to help restaurant owners and managers in managing their menu (items, availability, prices), order preparation time, order status, order history, and order notifications (like the arrival of order and/or cancellation of orders). This application will provide restaurant owners (or managers) a seamless and easy-to-use interface for managing their restaurant over QEats platform. By providing intuitive and easy access for restaurant owners (or managers) will result in an increase in the number of restaurants over QEats.

Why build it?

1. With the increase in adaptation and penetration of technology, it is necessary for restaurant businesses to adapt and incorporate such technology in managing orders easily.
2. With the high reachability of restaurants via online food ordering app, it is tremendously difficult for restaurants to manage orders without any technical solution.
3. Easy management of restaurant menu and updating the menu items
4. Quick update of availability of menu items based on the physical constraints
5. Updation of time of preparation on an item, to provide ETA of orders for the customers
6. Quick update of the status of orders and intimidation to users
7. To improve delivery time and quality of food by taking constant feedbacks
8. For providing order history for later reference of restaurant owners
9. For providing relevant statistics of the restaurant, so that they the owner can make the required decisions for increasing the business.

Design of Interactions (Block Diagram) -



Tech Stack for each Component -

1. Front-End
   a. Templates (Views) -
      - HTML5
      - CSS
      - Angular-Material

   b. Service Layer
      - Angular
      - TypeScript
2. Back-End
   a. Rest Controllers, Exchanges, Services, Repository Services, Entities -
      - Java
      - Spring boot
3. Data Store
   a. Database -
      - MongoDB
   b. Cache -
      - Redis

API Endpoints -

1. Adding an item to the menu
   `POST /qeats/v1/menu/item`

2. Removing an item from the menu
   `DELETE /qeats/v1/menu/item`

3. Edit the item in the menu
   `PUT /qeats/v1/menu/item`

4. Edit the quantity of items available
   `PUT /qeats/v1/item/available`

5. Get the quantity of items available
   `GET /qeats/v1/item/available?itemId=&restaurantId=`

6. Get all the placed orders
   `GET /qeats/v1/orders/placed?restaurantId=`

7. Accept/Reject the placed orders
   `PUT /qeats/v1/orders/placed`

8. Get all the orders (which are not yet finished and are accepted)
   `GET /qeats/v1/orders?restaurantId=`

9. Update status of order
   `PUT /qeats/v1/orders/placed`

10. User Login
    `POST /qeats/v1/user/login`

11. Get the menu of a restaurant
    `GET /qeats/v1/menu/item?restaurantId=`

API Contracts -

1. **Adding item to menu**
   (DTO => item;
   new response type with field: List <Menu> and menuResponseType)

```
API URI: /qeats/v1/menu/item
Method: POST
Request Body format
{
   "item": {
        "attributes": [
            "South Indian"
        ],
        "id": "1",
        "imageUrl": "www.google.com",
        "itemId": "10",
        "name": "Idly",
        "price": 45,
   },
   "restaurantId": "12"
}
//
Success Output:
1). If restaurantId is not valid return with BadHttpRequest
2). If item already exists in the menu then set menuResponseType
as
//
HTTP Code: 200
Response body contains the menu and menuResponseType
 {
   "menu": {
        "items": [
            {
                "attributes": [
                    "South Indian"
                ],
                "id": "1",
                "imageUrl": "www.google.com",
                "itemId": "10",
                "name": "Idly",
                "price": 45
            }
        ],
        "restaurantId": "11"
    },
    "menuResponseType": 0
}
Error Response:
HTTP Code: 4xx, if client side error.
         : 5xx, if server side error.
```

2. **Removing item from the menu**

```
API URI: /qeats/v1/menu/item
 Method: DELETE
 Request Body format:
  {
     "itemId": "10",
     "restaurantId": "12"
  }
 //
 Success Output:
 1). If restaurantId is not valid return with BadHttpRequest
 2). If item does not exists in the menu then set
menuResponseType as
QEatsException.ITEM_NOT_FOUND_IN_RESTAURANT_MENU
 //
 HTTP Code: 200
 Response body contains the menu and menuResponseType
  {
    "menu": {
      "items": [       ],
      "restaurantId": "11"
     },
     "menuResponseType": 0
  }
 Error Response:
 HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

3. **Editing item in menu**

```
API URI: /qeats/v1/menu/item
Method: PUT
Request Body format:
{
   "item": {
       "attributes": [
           "South Indian"
       ],
       "id": "1",
       "imageUrl": "www.google.com",
       "itemId": "10",
       "name": "Idly",
       "price": 50,
   },
   "restaurantId": "12"
}
//
Success Output:
1). If restaurantId is not valid return with BadHttpRequest
2). If item does not exists in the menu then set menuResponseType
as QEatsException.ITEM_NOT_FOUND_IN_RESTAURANT_MENU
//
HTTP Code: 200
Response body contains the menu and menuResponseType
 {
   "menu": {
       "id": "1",
       "items": [
           {
               "attributes": [
                   "South Indian"
               ],
               "id": "1",
               "imageUrl": "www.google.com",
               "itemId": "10",
               "name": "Idly",
               "price": 45
```

```
            }
        ],
        "restaurantId": "11"
    },
    "menuResponseType": 0
}
Error Response:
HTTP Code: 4xx, if client side error.
         : 5xx, if server side error.
```

4. **Edit the quantity of items available**
   (requires new DTO called Quantity or ItemAvailable or anything on the lines of this)

```
API URI: /qeats/v1/item/available
 Method: PUT
 Request Body format:
  {
     "itemId": "10",
     "quantity": 10,
     "restaurantId": "12"
  }
 //
 Success Output:
 1). If restaurantId or itemId is not valid return with
BadHttpRequest
 2). If item does not exists in the menu then set
menuResponseType as
QEatsException.ITEM_NOT_FOUND_IN_RESTAURANT_MENU
 //
 HTTP Code: 200
 Response body contains the updated quantity of item
  {
     "itemId": "10",
     "quantity": 10,
     "restaurantId": "12"
  }
 Error Response:
 HTTP Code: 4xx, if client side error.
```

```
          : 5xx, if server side error.
```

5. **Get the quantity of items available**

```
API URI: /qeats/v1/item/available?itemId=10&restaurantId=12
 Method: GET
  //
 Success Output:
 1). If restaurantId or itemId is not valid return with
BadHttpRequest
 //
 HTTP Code: 200
 Response body contains the quantity of items available in that
restaurantId
  {
     "itemId": "10",
     "quantity": 10,
     "restaurantId": “12”
  }
 Error Response:
 HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

6. **Get the placed orders**

```
API URI: /qeats/v1/orders/placed?restaurantId=12
 Method: GET
  //
 Success Output:
 1). If restaurantId is not valid return with BadHttpRequest
 //
 HTTP Code: 200
 Response body contains the placed orders (whose status is equal
to placed) list with restaurantId
    {
  "orders": [
      {
          "id": "1",
          "items": [
              {
                  "attributes": [
                      "South Indian"
                  ],
                  "id": "1",
                  "imageUrl": "www.google.com",
                  "itemId": "10",
                  "name": "Idly",
                  "price": 45
              }
          ],
          "restaurantId": "11",
          "total": 45,
          "userId": "arun",
          "status": "placed"
      }
  ],
  "restaurantId": "12"
}
```

7. **Accept/Reject the placed orders**

```
API URI: /qeats/v1/orders/placed
Method: PUT
Request Body format:
  {
     "orderId": "10",
     "status": “accepted”,
     "restaurantId": “12”
  }
 //
 Success Output:
 1). If restaurantId is not valid return with BadHttpRequest
 2). If orderId is not valid (is not a pending order) then do
nothing

 //
 HTTP Code: 200
 Response body contains the orders list (whose status is equal to
accepted, preparing, packed) with restaurantId
    {
```

```
  "orders": [
      {
          "id": "1",
          "items": [
              {
                  "attributes": [
                      "South Indian"
                  ],
                  "id": "1",
                  "imageUrl": "www.google.com",
                  "itemId": "10",
                  "name": "Idly",
                  "price": 45
              }
          ],
          "restaurantId": "11",
          "total": 45,
          "userId": "arun",
          "status": "accepted"
      }
  ],
  "restaurantId": “12”
}
```

```
 Error Response:
 HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

8. **Get all orders**

```
API URI: /qeats/v1/orders?restaurantId=12
 Method: GET
  //
 Success Output:
 1). If restaurantId is not valid return with BadHttpRequest
 //
 HTTP Code: 200
 Response body contains the list of orders (whose status is not
equal to placed) and restaurantId
```

```
  {
"orders": [
    {
        "id": "1",
        "items": [
            {
                "attributes": [
                    "South Indian"
                ],
                "id": "1",
                "imageUrl": "www.google.com",
                "itemId": "10",
                "name": "Idly",
                "price": 45
            }
        ],
        "restaurantId": "11",
        "total": 45,
        "userId": "arun",
        "status": "accepted"
    }
],
"restaurantId": "12"
}
```

```
Error Response:
HTTP Code: 4xx, if client side error.
         : 5xx, if server side error.
```

9. Update status of orders

```
API URI: /qeats/v1/orders/
Method: PUT
Request Body format:
  {
     "orderId": "10",
     "status": "packed",
     "restaurantId": "12"
  }
```

```
 //
 Success Output:
 1). If restaurantId is not valid return with BadHttpRequest

 //
 HTTP Code: 200
 Response body contains the updated list of orders (whose status
is not equal to placed or rejected or out for delivery) and
restaurantId
    {
  "orders": [
      {
          "id": "1",
          "items": [
              {
                  "attributes": [
                      "South Indian"
                  ],
                  "id": "1",
                  "imageUrl": "www.google.com",
                  "itemId": "10",
                  "name": "Idly",
                  "price": 45
              }
          ],
          "restaurantId": "11",
          "total": 45,
          "userId": "arun",
          "status": "packed"
      }
  ],
  "restaurantId": "12"
}


 Error Response:
 HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

10. **User login**
    (NEW DTO => user (id, username, password, restaurantId);
    new response type with field: restaurantId and userResponseType)

```
API URI: /qeats/v1/user/login
 Method: POST
 Request Body format:
  {
     "username": "tester",
     "password": "tester123"
  }
 //
 Success Output:
 1). If username and password does not matches the one stored in
DB then set the userResponseType as QEatsException.USER_NOT_FOUND
 //
 HTTP Code: 200
 Response body contains the restaurantId and userResponseType
  {
     "restaurantId": 10,
     "userResponseType": 0
  }
 Error Response:
 HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

11. **User registration**
    (NEW DTO => user (id, username, password, restaurantId);
    new response type with field: restaurantId and userResponseType)

```
API URI: /qeats/v1/user/register
Method: POST
Request Body format:
{
   "username": "tester",
   "password": "tester123",
   "name": "Tester",
   "restaurantName": "My testing restaurant"
}
```

```
Success Output:
1). If username and password already exists in DB then set the
userResponseType as QEatsException.USER_ALREADY_EXISTS
 HTTP Code: 200
Response body contains the restaurantId and userResponseType
{
   "restaurantId": 15,
   "userResponseType": 0
}
 Error Response:
 HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

12. Get the Menu for the given restaurantId

```
Success Output:
1). If restaurantId is present return Menu
2). Otherwise respond with BadHttpRequest.
HTTP Code: 200

{
   "menu": {
       "items": [
           {
               "attributes": [
                   "South Indian"
               ],
               "id": "1",
               "imageUrl": "www.google.com",
               "itemId": "10",
               "name": "Idly",
               "price": 45
           }
       ],
       "restaurantId": "11"
   }
}
  Error Response:
  HTTP Code: 4xx, if client side error.
          : 5xx, if server side error.
```

Eg:

```
curl -X GET
"http://localhost:8081/qeats/v1/menu?restaurantId=11"
```