DOCUMENTATION
PROJECT: ROYAL SANDWICH

Table of Contents

1. INTRODUCTION

Royal Sandwich is a web-based system that enables end-users to order sandwiches and track the state of their orders. This system comprises a backend and a frontend. The backend manages sandwich types, users, and sandwich orders' data, while the frontend also provides the user interface for placing orders and checking their status. In this document, we will describe the architecture and design of the system and provide instructions for using it.

Web name: Royal Sandwich
Development Team: Anh Duy Tran & Nghi Khanh Quyen

2. Design and Architecture

The figure above illustrates the architecture of the Royal Sandwich system, including a web frontend and a backend. The frontend provides a user interface, whereas the backend serves user requests asynchronously. The backend contains servers that communicate about tasks via message queues.

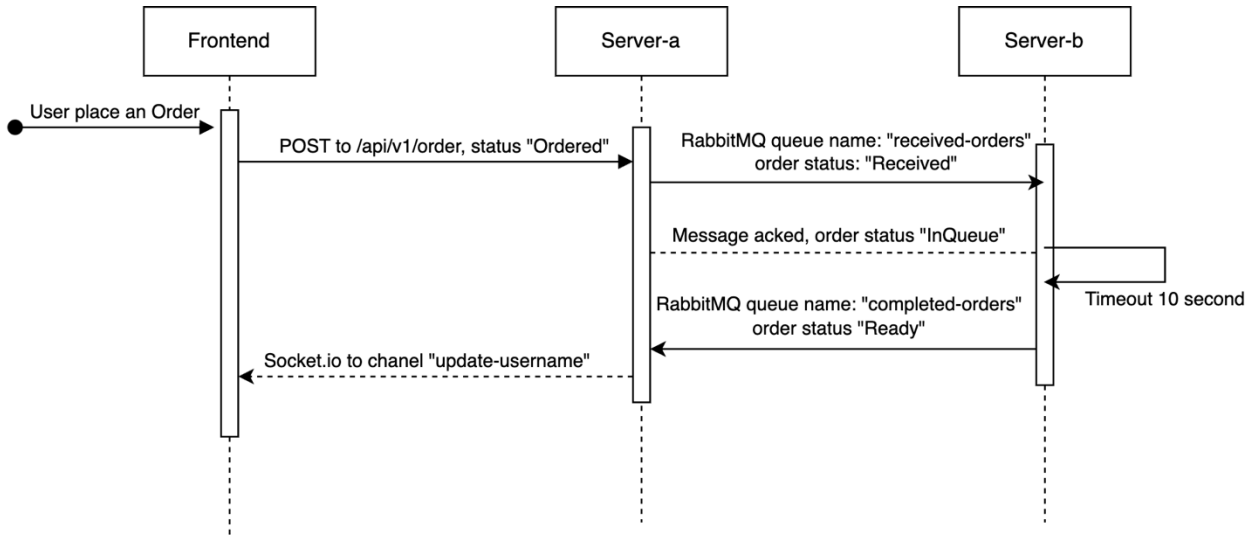


Figure A: A high-level view of the system architecture

2.1 Frontend

The frontend of the Royal Sandwich web-based system is built using React with TypeScript and Vite. The user interface is designed using MUI, a popular React component library that provides pre-built components for designing responsive and accessible UI. The frontend is responsible for providing an intuitive user interface for placing orders and checking their status.

2.1.1 Architecture and Design:

- The frontend of the Royal Sandwich web app follows a modular architecture to make it easier to manage and scale. The components are designed to be reusable, and the state is managed using React's context API. The app follows a declarative programming paradigm, where components are designed to render based on the state of the app.
- The app provides a login/signup feature for the users, which enables them to create an account to order sandwiches. Once logged in, the user can select the desired sandwich from the list of available sandwich types and add them to the cart. The app also provides the user with the option to remove sandwiches from the cart or update their quantities.
- Once the user has added the sandwiches to the cart, they can proceed to order by clicking on the "proceed to order" button. This action will take the user to the order confirmation page, where they can review their order and confirm it. The app provides the user with real-time updates on the status of their order, allowing them to track the progress of their order.

2.1.2 Instructions:

- To use the Royal Sandwich web app, the user needs to visit the login/signup page and create an account. Once logged in, the user can select the desired sandwich from the available sandwich types and add them to the cart. After adding the desired sandwiches, the user can click on the "proceed to order" button to review and confirm their order.

The app provides the user with real-time updates on the status of their order, which can be viewed on the order tracking page. The frontend establishes a socket connection with the backend to enable these updates.

2.1.3 Techonology

- The frontend of Royal Sandwich is built using React with TypeScript and Vite. React is a popular and widely used JavaScript library for building user interfaces, while TypeScript is a superset of JavaScript that adds static typing and other features to the language. Vite is a modern build tool that is designed to be fast and lightweight, making it a great choice for modern web applications.
- To help style the UI and provide pre-built components, the frontend uses the MUI (formerly known as Material-UI) library. MUI is a popular React UI library that provides a set of pre-built components and styles that can be easily customized to fit the needs of the application. This helps to save time and effort when building the user interface, and ensures that the application has a consistent and modern look and feel.
- For real-time communication between the server and the client, the frontend uses socket.io. Socket.io is a popular library for building real-time web applications that require bi-directional communication between the server and the client. This is especially useful for Royal Sandwich, as it allows users to receive updates on their orders in real-time without having to refresh the page.
- To handle authentication and authorization, the frontend uses JSON Web Tokens (JWT). JWTs are a popular and widely used standard for securely transmitting information between parties as a JSON object. In the case of Royal Sandwich, JWTs

are used to authenticate users and authorize them to perform certain actions, such as placing orders or viewing their order history.

2.2 Backend

The Royal Sandwich backend is responsible for managing sandwich types, users, and sandwich orders' data. It is designed to serve user requests asynchronously and utilizes a message queue to communicate about tasks.

2.2.1 Components

- The backend is comprised of several key components, including:
  - Server A: This component implements a pre-defined Swagger API to manage sandwich orders. It allows users to add new orders, show the state of earlier orders, and adds sandwich orders to a message queue.
  - Message Broker: The message broker delivers messages with two message queues: one for sandwich orders from Server A and another for sandwich status information from Server B.
  - Server B: This component receives sandwich orders from the message queue and publishes sandwich status information for each order when it finishes.

2.2.2 Architecture

- In terms of architecture, the backend is built using a Model-View-Controller (MVC) pattern, with the following key folders:
  - Controllers: This folder contains the backend logic for managing sandwich types, sandwich orders, users, and user authentication.
  - Routes: This folder contains the backend routing logic for managing HTTP requests and responses.
  - Models: This folder contains the data models for sandwich types, users, and sandwich orders.
  - Static: This folder contains static assets such as images, stylesheets, and JavaScript files.
  - Utils: This folder contains utility functions used throughout the backend.
  - Middleware: This folder contains the middleware functions used to intercept and process incoming requests.

- User credentials:
  - Using JSON Web Tokens (JWTs) for user authentication and authorization. When a user logs in, the backend generates a JWT that contains the user's name, email, and role. The role can either be "admin" or "customer", and it determines which parts of the system the user can access.

2.2.3 Swagger API

- The API of server A is defined in the Swagger.json file attachment. The definition was written with Swagger 2.0.

2.2.4 Technology

- The backend is built primarily using Node.js and Express framework for building the API. Additionally, we have implemented a message queue using RabbitMQ for communication between the different components of the system, with a MongoDB on

the cloud to store and manage data (MongoDB was selected for its scalability, flexibility, and ease of use)

- The backend utilizes Socket.io for real-time communication with the frontend, enabling users to check the status of their orders without the need for a full webpage refresh.
- Moreover, Docker containers are used to package and deploy the backend application. Docker allows creating of a self-contained environment for the application that includes all the dependencies and configurations it needs to run, regardless of the host system. Additionally, by using Docker Compose, we can define and manage the entire application stack (including the database and other services) as a single entity, simplifying deployment and configuration.

Overall, the backend of the Royal Sandwich system is designed to be highly scalable, efficient, and secure, providing users with a seamless experience when ordering sandwiches and tracking their progress.

3. Instructions

3.1 Fetching git code to local system

- Command to clone the repository from Gitlab:
  - clone with SSH: git clone git@course-gitlab.tuni.fi:compcs510-spring2023/quyen_and_tran.git
  - clone with HTTPS: https://course-gitlab.tuni.fi/compcs510-spring2023/quyen_and_tran.git

3.2 Installing project-specific tools

- The project requires RabbitMQ besides in the npm packages for running
  You can install it by referring to its official page: https://www.rabbitmq.com/

3.3 Commands/ Instructions required by the Project

1. Clone the repository: Start by cloning the Royal Sandwich web app repository to your local machine using Git. Open your terminal or command prompt and navigate to the directory where you want to clone the repository. Then, run the following command:

   git clone <repository-url>

2. Install Docker: Ensure that you have Docker installed on your local machine. If you don't have Docker installed, follow the official Docker installation guide for your operating system.
3. Run the app: Navigate to the cloned repository's root directory and run the following command:

   docker-compose up -d

   This command will start the containers required to run the app in the background.
4. Access the app: Once the app is up and running, you can access it by opening your web browser and navigating to **http://localhost:80** or **http://localhost**.
5. Authorization credentials:

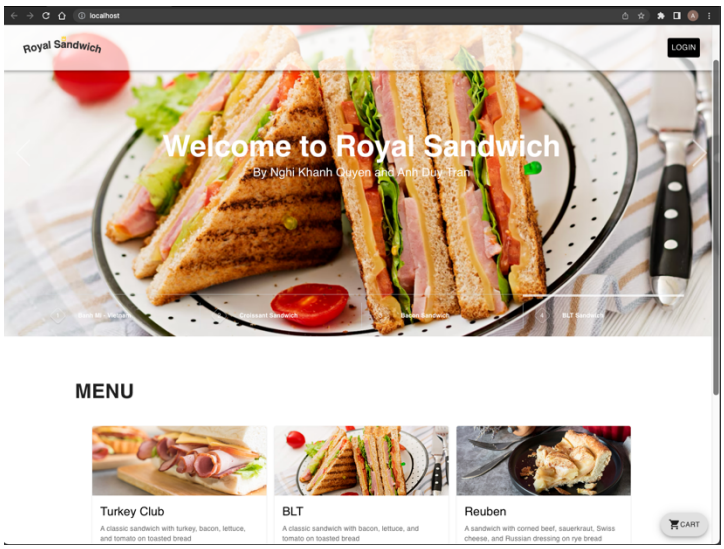
Admin credential: **username: admin, password: Admin123456**

3.4 Deployment steps

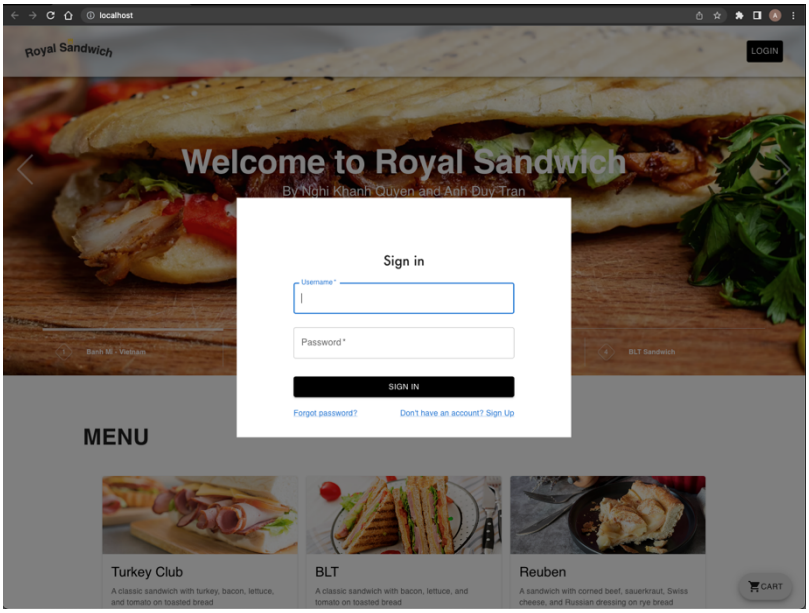
The deployment process is an ongoing effort that requires continuous attention and maintenance. It is recommended to regularly monitor and update the deployed components to ensure the system's reliability and security. Additionally, as the system evolves, it may require changes in the deployment process.

4. Screenshots:

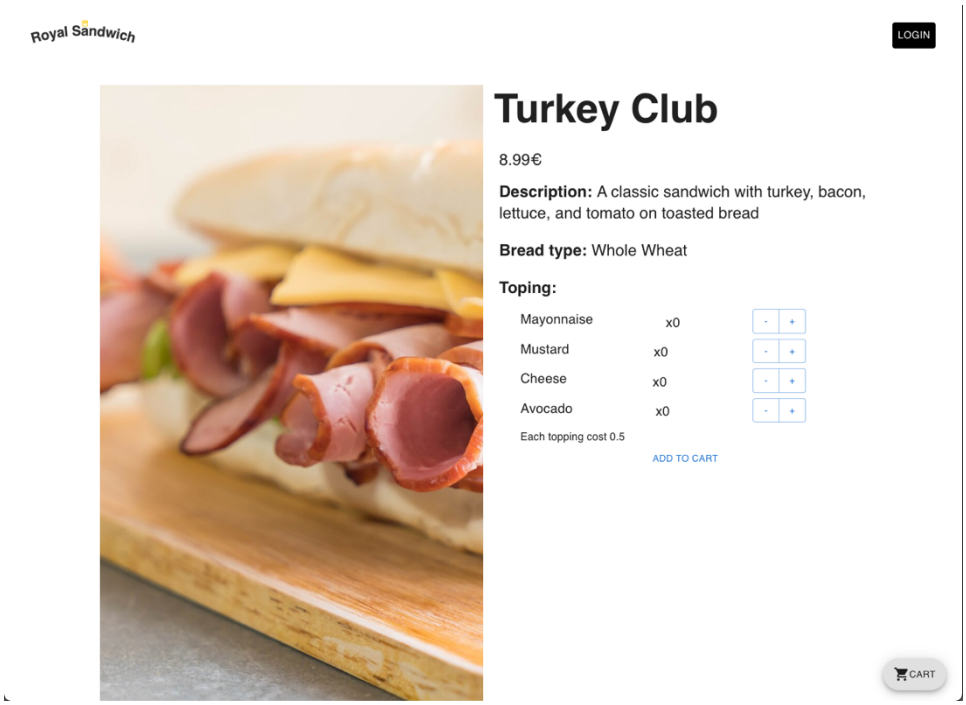
4.1.1 Store page:



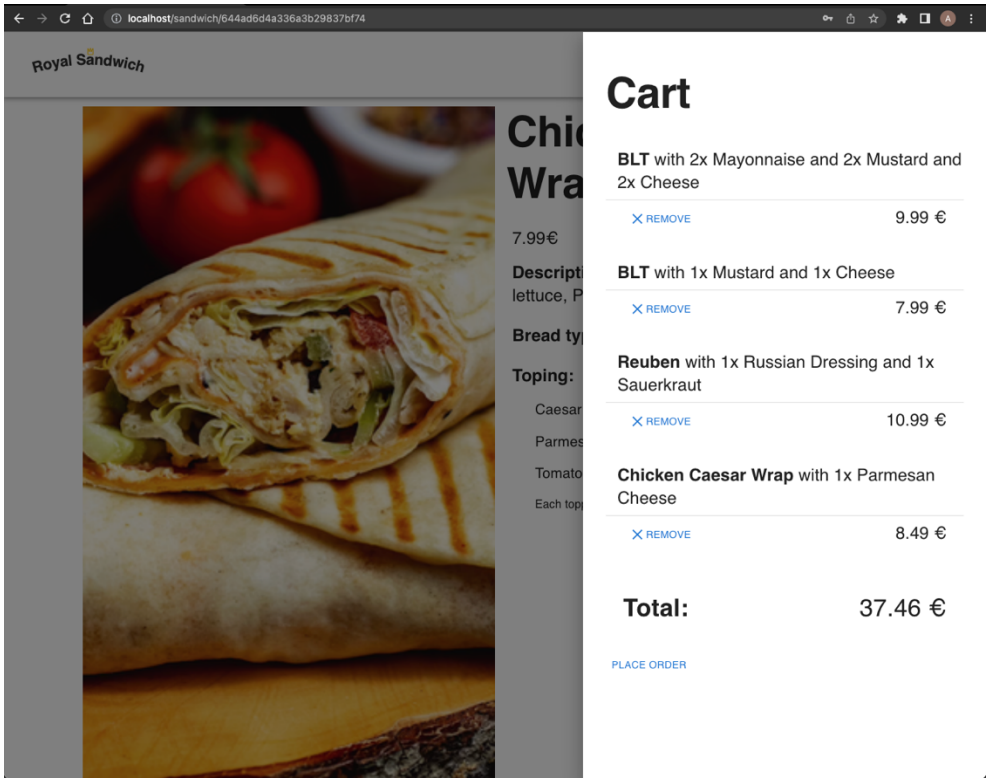
4.1.2 Login modal:



4.2.1 Product page:



4.2.2 Product page with open cart:



4.3.1 Order page