# Restaurant Management System

Technical Documentation

Royal Holloway University of London
24-25 CS2810/CS2815: Team Project
Team 08

# Technical Documentation

# Technological Choices

This section describes in detail the full technology stack used in our restaurant management system. Each technology was carefully chosen to create an professional system which is complete with front-end user interfaces, back-end capabilities, database management, and payment processing

## Front-End Technology Stack

Our project is built using 'Remix.run' with 'TypeScript'. 'TypeScript' was chosen for its strict type system, which improves code readability. Remix.run was selected because it simplifies webpages by inferring their page address, supports easy implementation of nested and parameterized routes, and is compatible with 'React.js'. 'Tailwind CSS' was used as opposed to traditional 'CSS' as it allows for faster styling using utility classes. Furthermore, additional libraries were used such as 'Sonner' for displaying notifications, 'Lucide Icons' for modern-esque user interface icons, and 'Recharts' for beautiful and dynamic graphs/charts (to name a few).

- **React.js -** Core UI library for building component-based interfaces
- **Remix.run -** React framework for client-side rendering and routing
- **TypeScript -** For type-safe JavaScript development
- **Tailwind CSS -** Convenient CSS framework for styling
- **ShadCn -** Ui Component library
- **Other Stylistic Component Libraries -** Used mainly for User interface styling purposes

## Back-End Technology Stack

The backend of our system is powered by 'Node.js' alongside 'Remix.run' for handling server-side logic and API routes. 'Node.js' was used in order to run the website locally on our devices. For database access/manipulation, we used 'Prisma ORM', which provides a way to interact and manage our database. 'Remix.run' also allows us to integrate user authentication which enables us to conduct role-based access control. Lastly, 'Stripe' was chosen for payment processing as recommended by the product owner.

- **Node.js -** JavaScript local environment
- **Remix.run -** Server-side rendering and API routes
- **Prisma ORM-** Database access and manipulation
- **Stripe -** Payment processing integration

## Database Technology Stack

'Prisma ORM' was the ORM (Object-Relational Mapper) for our schema as it works in tandem with 'Remix.run' which supports 'PostgreSQL'. 'Prisma ORM' allows us to define entity attributes and relationships. With the help of 'Remix.run', this enables us to easily issue requests to POST, and GET data to-and-from the database. 'Xata.io' was also used in order to change the database externally and write queries for testing. It is essentially a GUI-based data management system.

- **PostgreSQL -** Database management system
- **Prisma ORM-** Database schema modeling
- **Xata.io -** For external database manipulation and hosting

# User Stories

This section lists and describes the user stories that had been competently implemented into the web application throughout its development life cycle. The progress of these user stories were meticulously tracked in a shared 'GitLab' environment and external documents. Our team strived to complete as many initial user stories as possible. The **reasonably completed** user stories will be listed down alongside their importance, aptly quantified by the number of stars. For the sake of brevity, the phrase "As a _, I want to" had been removed within each user story.

## User Stories For Customer

This section displays all of the customer user stories that had been developed,

- ***(⭐⭐⭐)*** *View the menu, so that I can browse the available items before deciding what to order.*
- ***(⭐⭐⭐)*** *Place an order, so that I can request the food and drinks I want.*
- ***(⭐⭐)*** *See allergies and calories information, so that I can make informed choices based on my dietary needs.*
- ***(⭐⭐)*** *Filter the menu, so that I can easily find items that match my preferences or dietary restrictions (e.g., vegetarian, gluten-free).*
- ***(⭐)*** *Intuitive ordering system, so that I can easily navigate and complete my order without confusion.*
- ***(⭐)*** *Be able to call the waiter, so that I can ask for assistance or service when needed.*
- ***(⭐⭐⭐)*** *Be able to pay from the electronic interface I have at the table, so that I can leave at my own convenience.*
- ***(⭐⭐⭐)*** *Be able to track the progress of my order, so that I know how much longer I have to wait.*
- ***(⭐⭐)*** *See pictures of the food so that I can see what the food looks like.*

## User Stories For Waiter

This section displays all of the waiter user stories that had been developed,

- ***(⭐⭐⭐)*** *Be able to change the menu, so that I can update item availability or make adjustments as needed.*
- ***(⭐⭐⭐)*** *Confirm orders, so that I can ensure the correct items have been requested before sending them to the kitchen.*
- ***(⭐⭐⭐)*** *Track order times, so that I can monitor how long orders have been in progress and ensure timely service.*
- ***(⭐⭐⭐)*** *Mark an order as delivered, so that I can notify the system when a customer's order has been served.*

- ***(⭐⭐⭐)*** *Cancel an order, so that I can remove items or orders that were mistakenly placed or need to be changed.*
- ***(⭐⭐⭐)*** *Receive notifications when an order is ready for delivery, so that I can promptly deliver it to the customer.*
- ***(⭐) C****hange the status of an order, so that I can keep the system updated with the current state (e.g., in progress, ready for delivery, completed).*
- ***(⭐⭐)*** *Place an alert in the system when a client needs help, so that I can notify other team members to assist the customer as needed.*
- ***(⭐⭐⭐)*** *Notified when the client needs help so that I can assist them.*
- ***(⭐⭐)*** *Assigned to a certain table, so that I won't interfere with other waiters.*
- ***(⭐⭐)*** *Be able to see which tables have received their orders but have yet to pay, so that I can prevent people from leaving without paying.*
- ***(⭐⭐)*** *Be able to add items to a customer's selections, so that I can make extra sales by making suggestions to the client.*

## User Stories For Kitchen Staff

This section displays all of kitchen staff user stories that had been developed,

- ***(⭐⭐⭐)*** *I want to confirm customer orders, so that the team can be assured that the food and drinks ordered by the customer are available.*
- ***(⭐⭐⭐)*** *I want to notify waiters when an order is ready, so that they can deliver the food to the customer.*
- ***(⭐)*** *I want to track order times, so that I can ensure food is prepared and served within an appropriate time frame.*

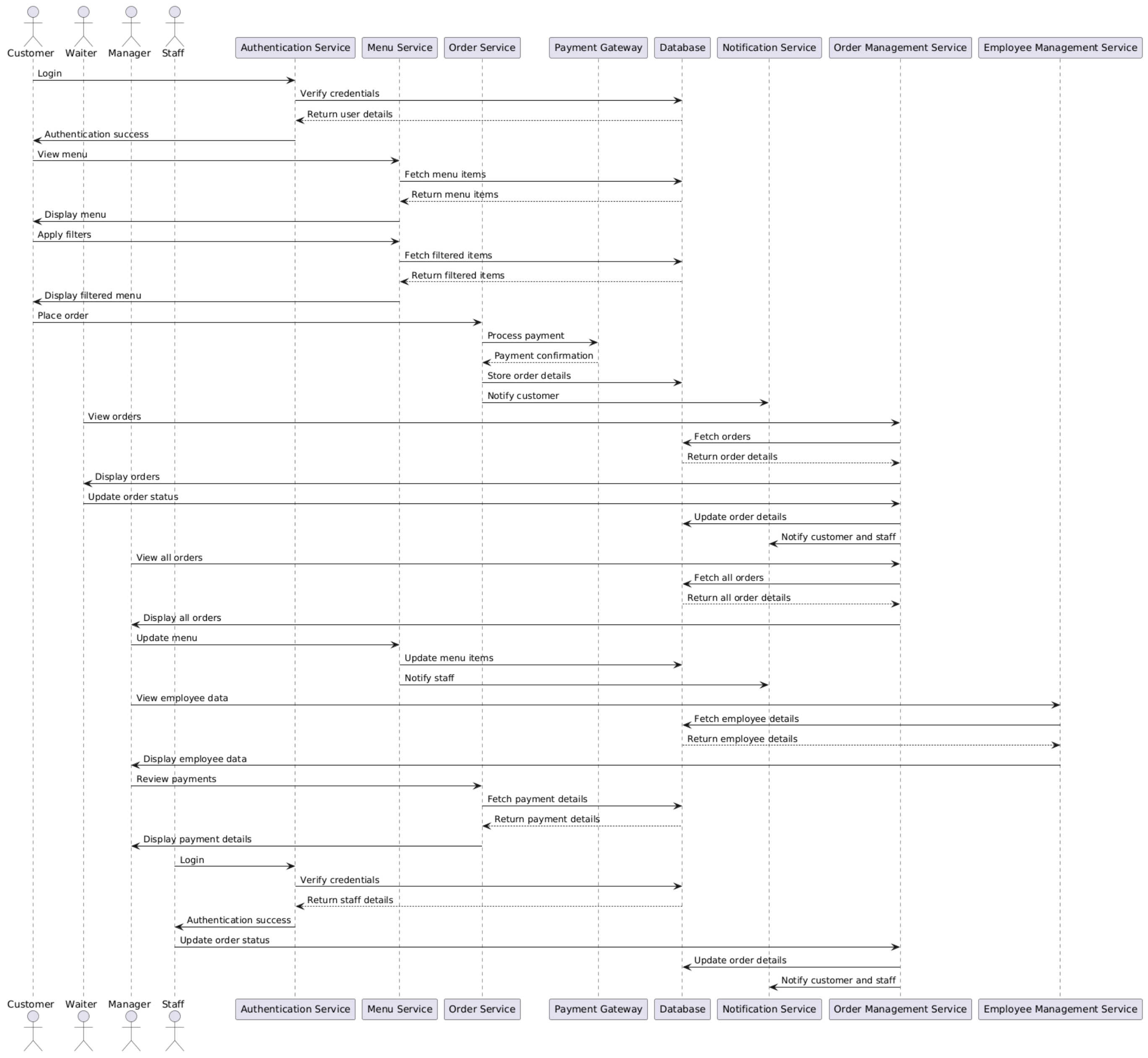
## User Stories For Manager

This section displays all of the manager user stories that had been developed,

- ***(⭐⭐⭐)*** *View data for each table, orders status, outstanding service requests and the stock in real time, so that I can ensure the overall operations are smooth and plan improvements*
- ***(⭐⭐⭐)*** *Adjust the menu, so that I can keep it updated.*
- ***(⭐⭐⭐)*** *Decide on prices with the help of the system, so that the price does not fall below the 60% profit margin.*
- ***(⭐⭐⭐)*** *Be able to see employee data, including sales, so that I can keep track of human resources.*

# Architecture

This section will show an architecture diagram of all the core modules used within the front-end, back-end, and database of the application. This includes all the files, packages, and classes for each corresponding layer. The corresponding description for this diagram will be given below.



All-encompassing UML sequence diagram displaying interactions between all functions/features

## Actors and Participants

- **Customer:** A user who interacts with the system as an external user.
- **Waiter:** A staff member responsible for serving the customer able to view and update order statuses, and receive notifications about order changes.
- **Manager:** A higher-level staff member who can view all orders, update the menu, review payments, and access employee data.
- **Staff:** General staff members who can log in and update order statuses.

## Services

- **Authentication Service:** Handles user authentication by verifying credentials against the database.
- **Menu Service:** Manages menu-related functionalities, including viewing and filtering menu items.
- **Order Service:** Manages order placement, payment processing, and order confirmation.
- **Payment Gateway:** External service for processing payments.
- **Database:** Central storage for user credentials, menu items, orders, and employee data.
- **Notification Service:** Sends notifications to customers and staff about order updates and other relevant information.
- **Order Management Service:** Handles order viewing and status updates.
- **Employee Management Service:** Provides access to employee data for managers.

## Key Interactions

**Authentication:** Customers, waiters, managers, and staff log in through the Authentication Service, which verifies their credentials against the database.

**Menu Viewing and Filtering:** Customers view the menu and apply filters to see specific items. The Menu Service fetches and displays the relevant items from the database.

**Order Placement:** Customers place orders through the Order Service, which processes payments via the Payment Gateway. Upon successful payment, order details are stored in the database, and notifications are sent to the customer.

**Order Management:** Waiters and staff view and update order statuses using the Order Management Service. Changes are reflected in the database, and notifications are sent to relevant parties.

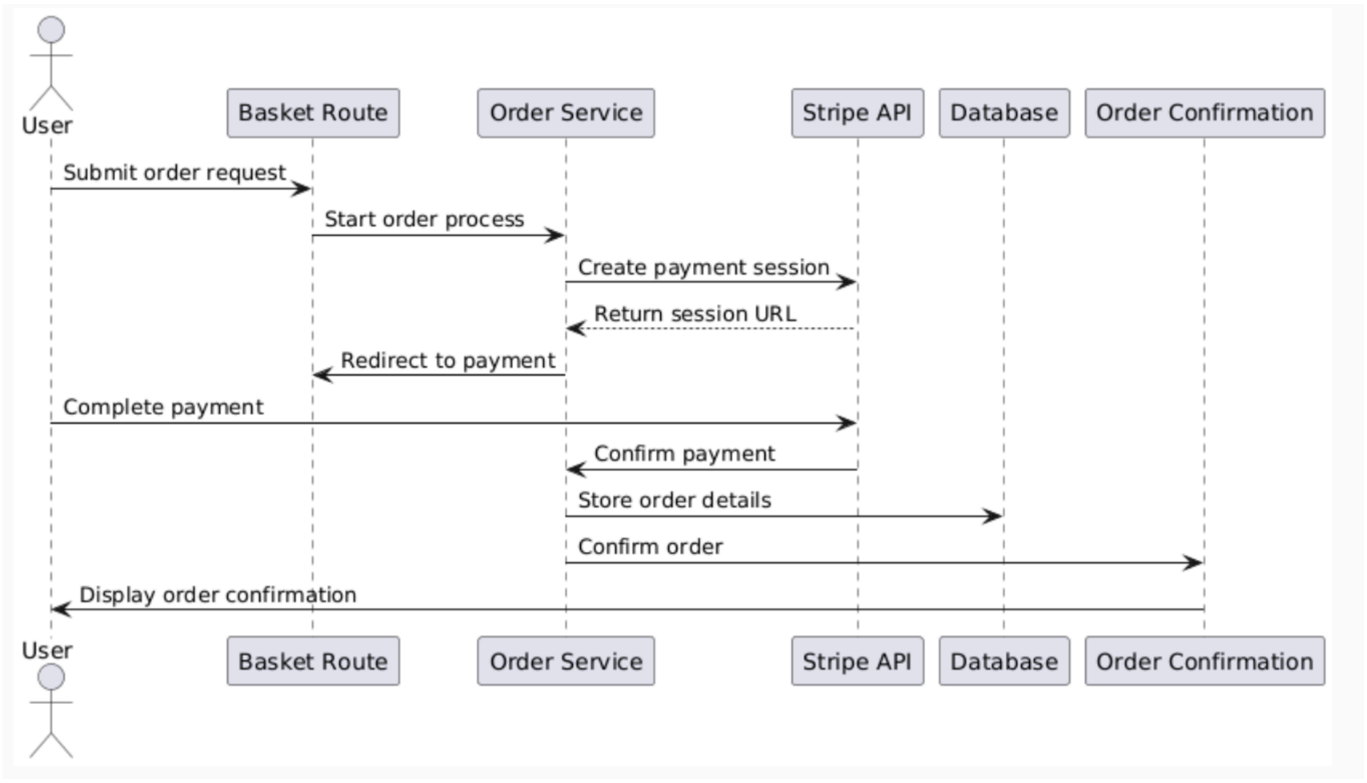
**Menu Updates:** Managers update the menu through the Menu Service, which updates the database and notifies staff of changes.

**Employee Data Access:** Managers access employee data through the Employee Management Service, which retrieves and displays
information from the database.

**Payment Review:** Managers review payment details through the Order Service, which fetches and displays payment information from the database.

## User Story Documentation Example

This section will display in detail a UML sequence diagram for the user story; *"As a customer, I want to place an order, so that I can request the food and drinks I want."* This diagram shows how each component and feature interact with one another during the process of a user ordering food. Notable inclusions within the successful implementation of this user story is the ability to POST and GET data directly from the database, an interactive front-end user interface, and a fully functional payment system, leveraging Stripe's API.

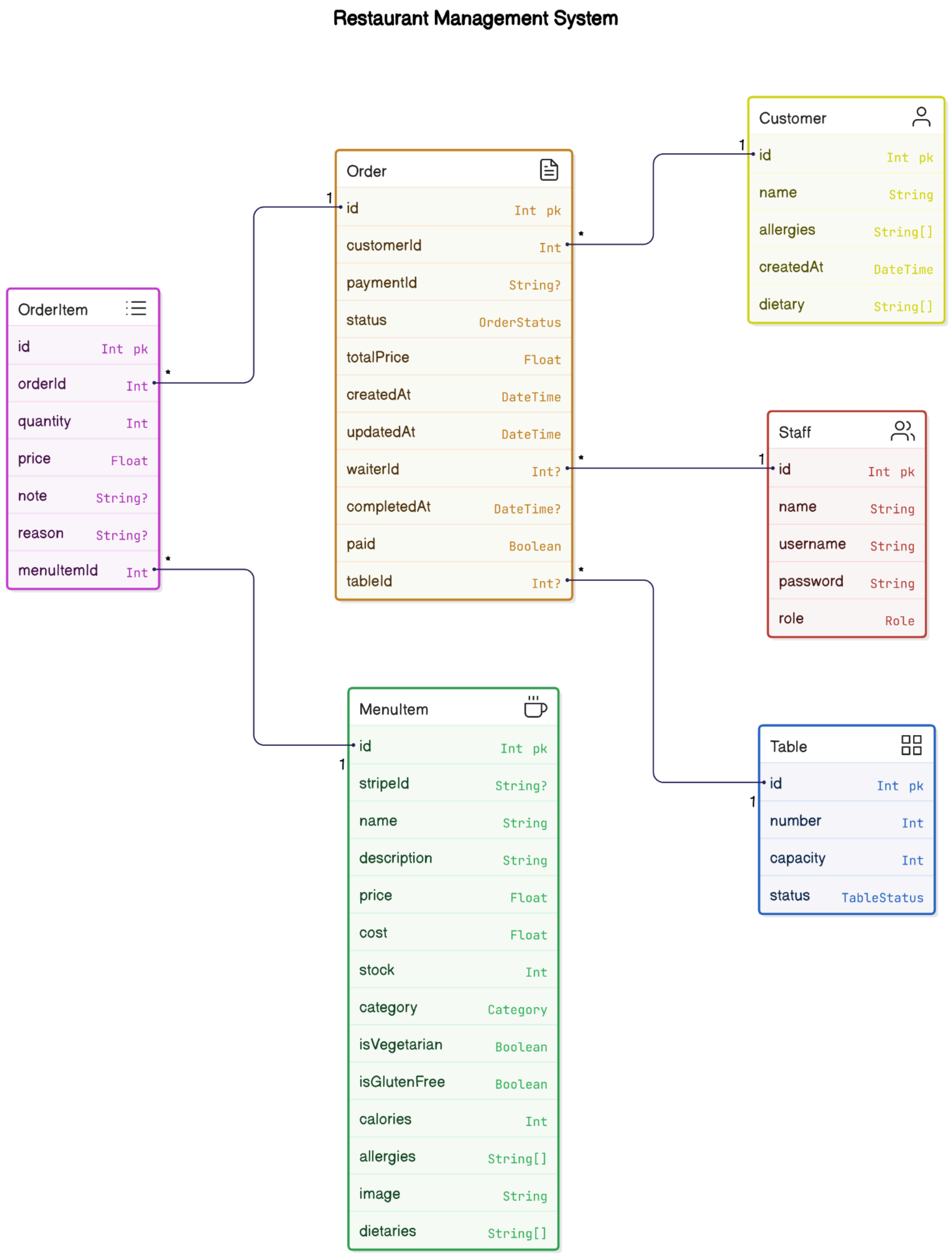


UML Sequence diagram of entire customer payment system

- **User:** Initiates the request to place an order.
- **Basket Route:** Handles the request and uses an action function to process the order.
- **Order Service:** Starts the order process by interacting with the Stripe API for payment.
- **Database:** Updates the order details and stores them in the database.
- **Order Confirmation:** Confirms the order and updates the order status.

## Database Design

This section displays the final proposed ER Diagram showing the database structure with PostgreSQL as the chosen SQL version that was used for this project.



## Video Demonstration

This section provides a 10 minute recorded demo of the final version of the Restaurant application; demonstrating all the implemented interactions and features

CS2810 Restaurant Management System Team 8 Video Demonstration Link: https://youtu.be/ksM1KBHBK0A
**(Set the Quality to 1080p)**