Project Description Web-Technologien WS 21

Currently, ordering food in a restaurant is typically realized as follows: A waiter registers the ordered food and beverages at the table, forwards the order to the kitchen, and later serves food and beverages. After the guests finished their meal, they pay for the order at their table via the waiter.

While this approach is typically appreciated due to the human interaction between guests and waiters, it imposes several problems:

- The process is labor-intensive, and waiters are rare nowadays.
- Guests often have to wait a long time to submit their order and finally pay.
- Often, the order is not correctly communicated to the kitchen.
- In the current pandemic, human contact should be reduced to a minimum.

We, therefore, aim to automate orders and payments in restaurants.

The guests scan a QR code at their table. This will open a web application of the restaurant, where guests can:

- Browse the Menu.
- Order items.
- Track orders.
- Pay for their orders.
- Call the waiter for consultation.

Waiters will carry the cooked meals and drinks to the guests and consult guests. The initial QR code should also contain information about the table where the guest is located.

You will develop the proposed web application in multiple teams:

Team 1: Management View (up to 3 Students)

In the management view, the restaurant manager can:

- Create, update and delete tables, print table-specific QR codes for guests.
  Tables have a seating capacity, a table number, and a location description.
- Create, update, and delete categories for food and beverages.
  Categories have a name and a type (food, beverages, specials).
- Create, update, and delete menu items.
  Menu items have a title, a description, a price, and information on allergens. Menu items are assigned to one or more categories.
- Create, update, and delete users for the management view, waiters, and kitchen view. A user has a role (management, waiter, kitchen) a user-id, a user-name, and a password.

Team 2: Guest View (Up to 2 students)

In the guest view, the guest can:

- Browse the menu and add available items to a shopping cart.

- Remove and modify (update number) items in a shopping cart.
- Submit an order. This step also includes payment. Only paid orders are forwarded. However, you do not need to include payments for the project. It's sufficient to call a mock payments service generating a JWT indicating the success or failure of the payment.
- Check the status of an order. An ordered item can have the status ordered, in production, ready-for pickup, out for delivery, delivered.
- Call the waiter for consultation.
- For groups with more than one member:
    - Guests can write reviews for the restaurant.
    - Guests can rate specific menu items.
    - Display a virtual product category "top seller" that always contains the 5 most frequently bought items of the last 30 days.

Team 3: Waiters and Kitchen View (Up to 2 Students)

For the waiter:

The waiter can:

- View the list of incomplete orders (where not all items are in status delivered).
- View details of incomplete orders.
- View the list of requests for consultations. Mark requests as processed.
- View the list of items that are ready for pickup.
- Mark items as "in-transit" and "delivered".
- Make sure that your views work well on mobile devices.
- Have Push Notifications when items are ready for pickup.

For the kitchen:

The kitchen personnel can:

- Browse all menu items and mark items as not available/not available. Update categories of items.
- View the list of orders to cook (status ordered), mark them as "in production" or "ready-for pickup".
- Change the order of items waiting to be processed according to the needs of the kitchen.
- Add a free-text comment to an item on an order. E.g., item is not available, ask you waiter for compensation.

Requirements for all Teams:

The application...

- … uses (proper) Angular for the client-side.
- … accesses data via (real) REST Service(s).
- … REST Services are implemented in Node.js. Data is stored in a database. The usage of Postgres is recommended.
- … has user authentication and users see only their own data.
- … should be responsive and must have a coherent UI.
- … is not easily hackable (use prepared statements and filter all inputs!). Details on lecture session on security.

- The given requirements are just starting points. If additional functionality is required for the successful execution of the entire process they must also be included. If you propose alternative functionality for the same application this can be negotiated with the client (the lecturer).
- Each sub-project must be integrated with at least two other groups covering the other sub-projects to demonstrate the overall use-case.
- The provided JSON files are meant as a starting point. You may need additional properties or even additional JSON representation for your backend services. Negotiate with the other groups!
- Be sure that the application for guests and waiters can be intuitively used on mobile phones.