Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: QRDER

Analysis Report

Mustafa Oğuz Güngör

Taylan Bartu Yoran

Mehmet Akif Kılıç

Supervisor: Varol Akman

Jury Members: Uğur Güdükbay, Fazlı Can

Analysis Report
March 23, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

Analysis Report

Project Short-Name: Project Title

1 Introduction

Comparing last decades, more people prefer eating outside rather than at home. Since there may be many reasons [1] , it can be said making restaurants serve meals fast and good has become harder. Since number of people who go to well known restaurants increases, restaurant owners try to find a new ways to maintain quality of dishes and restaurant. Since there are some knows standards [2] , it may be impossible to meet those when restaurant is too crowded. Also, these standards are dependent to waiters / waitresses.

Finding a good waiter or making menu updated and well designed may be costly. So , any owner that wants good service quality and make this sustainable must pay big amounts of money. However, “Qrder” aims to make ordering and payment online so that restaurants may focus on food quality rather than getting order and checking the bill.

The key idea behind the “Qrder” is make ordering, choosing the meal , checking the bill and displaying the menu digital. So that, waiters will have specific missions like delivering the meal instead of being responsible for any mission. To sum up . Qrder automizes the order and payment process so that it will reduce the workload of restaurants.

For example, imagine yourself going a crowded restaurant with your friends. You would waste a lot of time waiting for a water to make an order or making payment. When that is the case, Qrder helps you in that case. If you would make an order and payment by using

Qrder, you would not had to waste time for ordering and payment. Furthermore, if restaurants let Qrder to be used, their responsibility will be decreased.

This application can be used for different purpose. For example, a restaurant wants to change their menu design anytime they want. For example, when they want to update the price of the meal , they can change it instantly. Another example is changing order. When clients want to change/ cancel the order, they can do it via Qrder. So, to sum up , Qrder let clients and restaurants to changes their decisions instantly.

In this report, analysis of the system will be demonstrated. We will start by giving information about existing systems and their scopes. Then, Qrder will be explained detailed and it's highlighting features will be demonstrated. After that, functional , nonfunctional and pseudo requirements will be showed. So that, we will have an idea about scope of the application and technologies that we might need. Then, system models and diagrams will be generated. Then, possible scenarios will be produced to guess what may happen. Later that, screen mock-ups and navigational paths will be created. Finally, social dimensions will be considered.

2 Current System

Since there are some systems that notifies clients when the meals are ready. There are also some systems that provides digital menu for clients. However, there is no system that make ordering, menu and payment. Furthermore , systems that are explained above are hardware dependent and expensive. So, our application provides efficient and cheap solution.

3 Proposed System

3.1 Overview

Qrder is a web application that makes ordering and payment online. Qrder is innovative since there is no application that provides online ordering and payment. Online ordering can be challenging because while client orders a meal through the menu or pays the check, restaurant must be notified and these processes must be handled flawlessly. On the other hand, when the restaurant makes the meal ready, client must be notified. In addition, these orders must be kept in order with respect to ordering time. In other words, this application works as first come first serve.

Since this application is a web based application, tools that we will use are Ruby, MySQL and a web server. Furthermore, since network is crucial for this application, sub topic “Computer Network” and terms that related to that such as sockets etc. will be used. We hope that this application shows that good network , database and backend design is the key for the good web based application.

We think that there may be large amounts of user that uses our application. To make our program accessible for everyone, it will have to versions: desktop and mobile. In addition, it should run on both IOS and Android. Desktop version will be used by the restaurant. On the other hand, clients will prefer mobile version.

3.2 Functional Requirements

In this part, functional requirements will be discussed. If this requirements are understood, system will be easier to understand. In addition to that, pseudo non-functional requirements will be discussed too.

3.2.1 System Functionality

The system should:

- make register available for clients/restaurant.
- make a login system for clients/restaurants.
- gather info such as address , phone number and menu for restaurants.
- allow restaurants to create and alter their menu.
- show restaurants that is close to user.
- allow user to look for any restaurant that is close to s/he.
- show restaurants that meets the needs of the search.
- ask clients to scan the qr code.
- find corresponding restaurant that matches the qr code.
- display the corresponding menu.
- receive the order.
- inform restaurant about the order.
- calculates the check.
- inform user about the check.
- provide online payment.
- work interactively with the diet application.
- provide any interface about advertisements.
- display any special offer (if any)

3.2.2 User Functionality

The user should:

- register the system.
- provide required information while registering,
- login the system.
- scan the qr code.
- choose the meal that he/she wants among the menu.
- look for any restaurant that uses Qrder app by searching.
- make any change about the order.
- look and use for special offers.
- use diet program interactively.
- learn the nutrients of the meal.
- see the information about whole food that are displayed on the menu,
- change it's profile.
- cancel the order

3.3 Non-functional Requirements

In this section nonfunctional requirements will be discussed. "**NON-FUNCTIONAL REQUIREMENT** (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system."[3] . That's why, we spent a lot of time while thinking about nonfunctional requirements. Our subtopics are extensibility , reliability , usability, accessibility, portability and efficiency,

3.3.1 Extensibility

The system should:

- be easy to maintain.
- be available on multiple platforms.

3.3.2 Reliability

The system should:

- do not store any credit card information unless the user state otherwise.
- be prepared for any possible attacks. For example, whole information on the database will be hashed.
- ensure that user's data is safe.
- ensure that there will be no fake orders , restaurants or clients.

3.3.3 Usability

The system should:

- be user friendly.
- create options in terms of language and theme.
- provide exact pictures of food and accurate info about meals.
- work well with the diet program.

3.3.4 Accessibility

The system should:

- be downloadable for free.
- be downloadable from the official website for the desktop version.
- be downloadable from the App Store or Google Play Store for the mobile version.

3.3.5 Portability

The system should:

- run in any OS.
- be able to work cross-platform.

3.3.6 Efficiency

The system should:

- not lag when communicating with the server especially when in the restaurant has many clients.
- be light.

3.4 Pseudo Requirements

3.4.1 Version Control System & Project Management

- GitHub will be used for version control for tracking changes.

- Code coverage will be enforced in the push action to Github by automatically running a code formatter, a prettifier and tests before pushing. If any of them fails, push will not be done.
- GitHub will also be used to create a Kanban style board for project management. Sprints will be always up-to-date and jobs will be self-contained to be able to have more detailed information about the process of the implementation.
- Google Sheets will be used for tracking sprint points to be able to analyze performance of the team.

3.4.2 Issue Tracking

- Firebase Crashlytics library in the Android app will be used for real-time detailed crash reports.
- Loggly, Loglevel or more efficient library will be used for a logging system to generate non-critical error logs. End of every session, these error logs will be sent to a service and deleted from local storage of the user.

3.4.3 Target Platform & User Experience (UX)

- Business side must run on Windows and Linux.
- Client side must run on Android.
- Accessibility, flexibility and responsible User Interface (UI) must be followed.

3.4.4 Testing

- Jest will be used as a JavaScript testing framework.

- All members must write their own tests for the assigned part to themselves.
- Alpha & Beta testing will be done by a chosen group of people from different disciplines.

3.4.5 External Tools and Technologies

- Slack will be used for main communication as the chatting platform.
- Google Hangouts will be used for the online meetings.
- Google Docs will be used for future reports of the project.
- Zeplin will be used for creating a design flow and representing it.
- Adobe XD will be used to design User Interface (UI).
- Eslint & prettifier will be used for the code formatting and have a single-type code.

3.5 System Models

In this part, we will show how our app makes things work during different actions. This part is crucial for us because the way that how Qrder works is explained in summary by giving highlighting operations.

3.5.1 Scenarios

Scenario 1	
Use Case Name	Register
Participating Actors	Nathan & Restaurant A
Entry Conditions	● Application should be open and user should not be logged in
Exit Conditions	● User either completes registration as clicking register button or cancels it as clicking cancel button
Main Flow Events	User: 1. Should open the app 2. Clicks on Register button 3. Completes required areas 4. either click on register button to complete registration or cancel button to cancel it.

Scenario 2	
Use Case Name	Login
Participating Actors	Nathan & Restaurant A
Entry Conditions	● Application should be open and user should not be logged in
Exit Conditions	● User either fills required areas and clicks login button
Main Flow Events	User: 1. Opens the app 2. Completes required areas 3. clicks login button

Scenario 3	
Use Case Name	List Meals
Participating Actors	Nathan
Entry Conditions	● QR Code of the table should be scanned
Exit Conditions	● User presses the “order” button to order a meal or ”back” button to get back to the main page.
Main Flow Events	User: 1. login 2. scans the QR Code at the table of the restaurant 3. presses “order” button

Scenario 4	
Use Case Name	Make order
Participating Actors	Nathan
Entry Conditions	● Application should be open ● User should be logged in ● User should have scanned the QR code on the table
Exit Conditions	● User either completes order as clicking “done” button or cancels it as clicking cancel button
Main Flow Events	User: 1. Opens the app 2. login 3. scans the QR code 4. decides what to order from listed meals 5. either clicks on the “done” button to complete the order or cancel button to cancel it.

Scenario 5	
Use Case Name	Search Nearby Restaurants
Participating Actors	Nathan
Entry Conditions	● Application should be open ● User should be logged in ● User should have allowed access to location information before.
Exit Conditions	● User clicks “back” button
Main Flow Events	User: 1. Opens the app 2. login 3. Clicks “Nearby Restaurants” button 4. allows access for location info 5. Click the “search” button. 6. Clicks “back” button when he/she is done.

Scenario 6	
Use Case Name	Call Waiter
Participating Actors	Nathan
Entry Conditions	● QR Code of the table should be scanned ● user presses “call waiter” button
Exit Conditions	● User presses the “done” button to cancel the call or to confirm the waiter has come.
Main Flow Events	User: 1. Scans the QR Code 2. presses call waiter button 3. says to waiter what he/she planned to say 4. presses “done” button

Scenario 7	
Use Case Name	Payment
Participating Actors	Nathan
Entry Conditions	● User should complete an order ● user presses “go to payment” button
Exit Conditions	● Restaurant confirms payment.
Main Flow Events	User: 1. User makes his/her order 2. User get his/her order 3. User press “go to payment” button to pay his/her meal 4. User completes payment 5. Restaurant confirms payment.

Scenario 8	
Use Case Name	Rate
Participating Actors	Nathan
Entry Conditions	● User should complete the payment ● user presses “rate meal” button
Exit Conditions	● User presses the “done” button to finish rating.
Main Flow Events	1. User completes payment 2. Restaurant confirms payment. 3. User presses the “rate” button. 4. User completes rating the meal and the service. 5. User presses the “done” button.

Scenario 9	
Use Case Name	List Past Orders
Participating Actors	Nathan
Entry Conditions	● Application should be open ● User should be logged in
Exit Conditions	● User presses the "back" button to go back to the main page or presses the "show" details" button for any specified past orders in the list.
Main Flow Events	User: 1. Opens the app 2. login 3. Clicks "List Past Orders" button 4. Clicks "back" button when he/she is done or "show details" button for specified order.

Scenario 10	
Use Case Name	Modify Order
Participating Actors	Nathan
Entry Conditions	● user must have an existing order.
Exit Conditions	● user "back" button. ● user modifies the order
Main Flow Events	1. user chooses order to modify 2. user clicks "modify order" 3. user modifies the order 4. clicks " modify order" button.

Scenario 11	
Use Case Name	Cancel Order
Participating Actors	Nathan
Entry Conditions	● user already has an order.
Exit Conditions	● user clicks “back” button. ● user cancels the order.
Main Flow Events	1. clicks desired order to cancel. 2. clicks “cancel order” button.

Scenario 12	
Use Case Name	Check Order Status
Participating Actors	Nathan
Entry Conditions	● user already has an order.
Exit Conditions	● clicks “back” button
Main Flow Events	1. chooses an order from order list 2. clicks the desired order. 3. clicks “check order status” button.

Scenario 13	
Use Case Name	Check Orders
Participating Actors	Restaurant A
Entry Conditions	● application must be open. ● restaurant must be logged in
Exit Conditions	● clicks “back” button
Main Flow Events	1. clicks “Check Orders” button.

Scenario 14	
Use Case Name	Modify Order
Participating Actors	Restaurant A
Entry Conditions	● application must be open. ● restaurant must be logged in ● client must have an order
Exit Conditions	● clicks “back” button. ● clicks “done” button.
Main Flow Events	1. clicks “show orders” button. 2. Finds desired order. 3. Clicks “edit order” button.

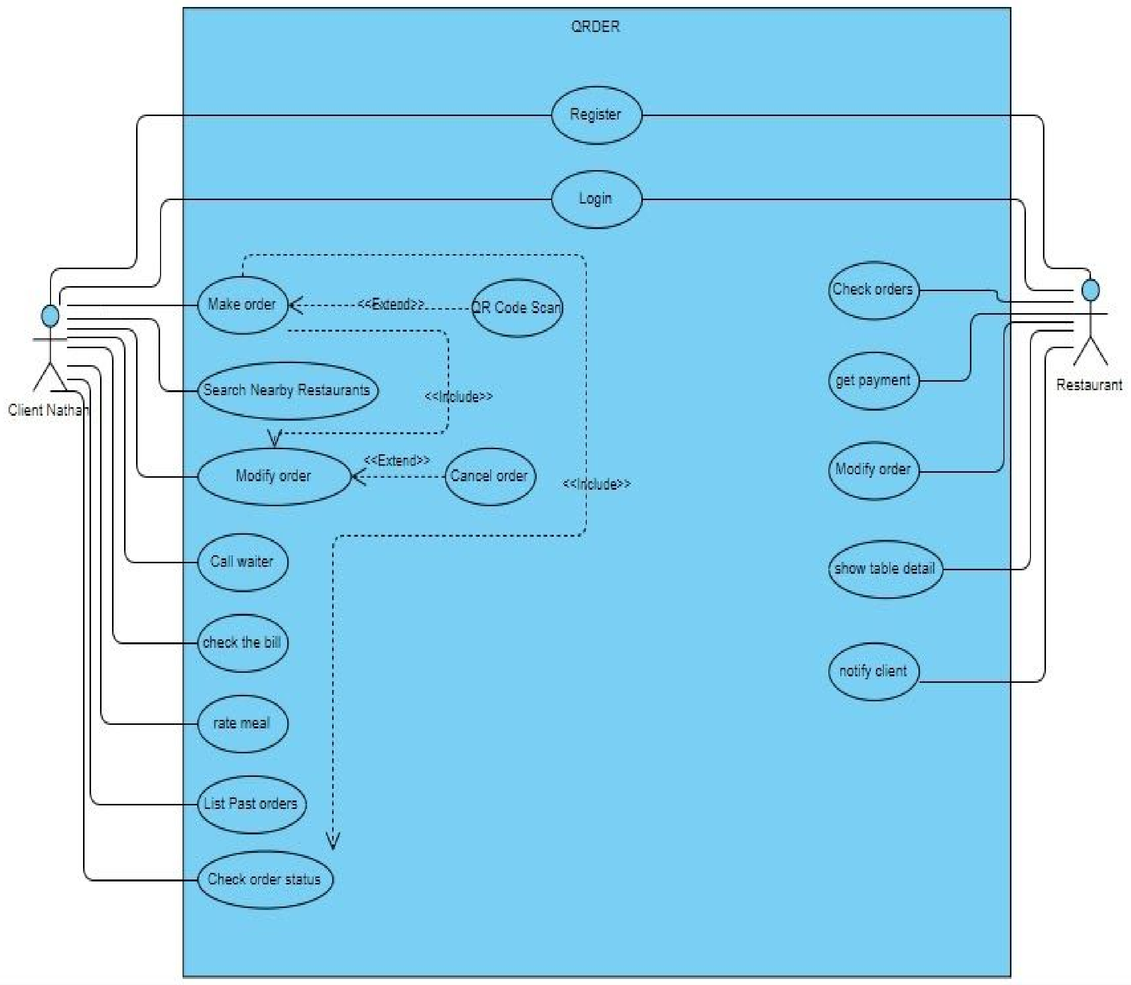
Scenario 15	
Use Case Name	Show Table Detail
Participating Actors	Restaurant A
Entry Conditions	● application must be open. ● restaurant must be logged in
Exit Conditions	● clicks “back” button.
Main Flow Events	1. Clicks desired table on the main page.

Scenario 16	
Use Case Name	Notify Client
Participating Actors	Restaurant A
Entry Conditions	● application must be open. ● restaurant must be logged in ● order must be ready
Exit Conditions	● clicks “back” button.
Main Flow Events	1. clicks desired order. 2. clicks “ notify client” button.

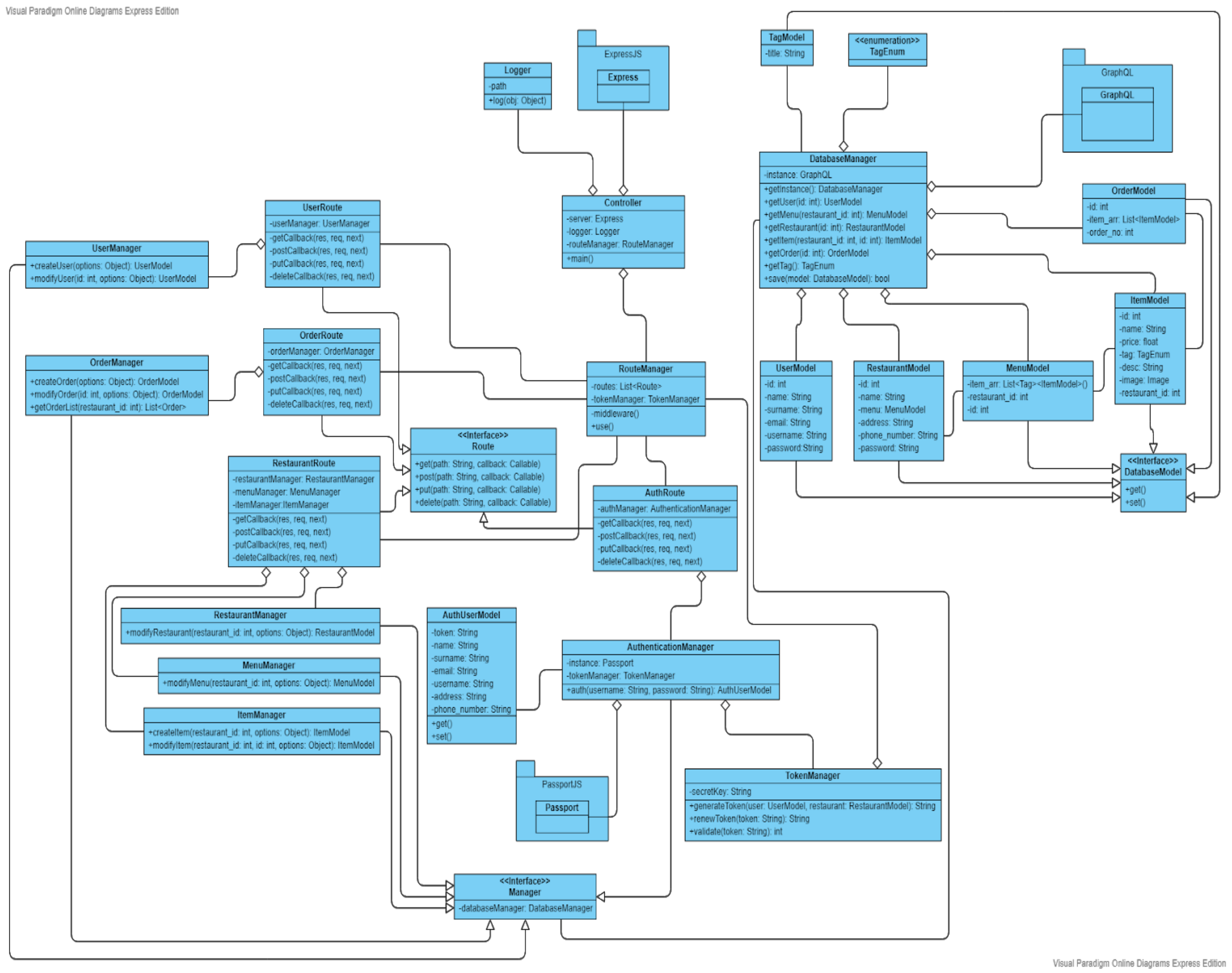
Scenario 17	
Use Case Name	get payment
Participating Actors	Restaurant A
Entry Conditions	● application must be open. ● restaurant must be logged in ● client must ready to pay
Exit Conditions	● clicks “tick” button.
Main Flow Events	1. clicks “ticks” button to get payment. 2. assign someone to get money.

3.5.2 Use-Case Model

In this part, use case will be introduced. This methodology is used to identify, visualize and clarify system requirements. By looking that diagram, you may understand who does what and interactions between actors like server , client and restaurant.



3.5.3 Object and Class Model


Visual Paradigm Online Diagrams Express Edition
Logger
-path
+log(obj: Object)
ExpressJS
Express
TagModel
-title: String
<<enumeration>>
TagEnum
GraphQL
GraphQL
DatabaseManager
-instance: GraphQL
+getInstance(): DatabaseManager
+getUser(id: int): UserModel
+getMenu(restaurant_id: int): MenuModel
+getRestaurant(id: int): RestaurantModel
+getItem(restaurant_id: int, id: int): ItemModel
+getOrder(id: int): OrderModel
+getTag(): TagEnum
+save(model: DatabaseModel): bool
OrderModel
-id: int
-item_arr: List<ItemModel>
-order_no: int
ItemModel
-id: int
-name: String
-price: float
-tag: TagEnum
-desc: String
-image: Image
-restaurant_id: int
UserModel
-id: int
-name: String
-surname: String
-email: String
-username: String
-password:String
RestaurantModel
-id: int
-name: String
-menu: MenuModel
-address: String
-phone_number: String
-password: String
MenuModel
-item_arr: List<Tag><ItemModel>()
-restaurant_id: int
-id: int
<<Interface>>
DatabaseModel
+get()
+set()
Controller
-server: Express
-logger: Logger
-routeManager: RouteManager
+main()
UserRoute
-userManager: UserManager
-getCallback(res, req, next)
-postCallback(res, req, next)
-putCallback(res, req, next)
-deleteCallback(res, req, next)
UserManager
+createUser(options: Object): UserModel
+modifyUser(id: int, options: Object): UserModel
OrderRoute
-orderManager: OrderManager
-getCallback(res, req, next)
-postCallback(res, req, next)
-putCallback(res, req, next)
-deleteCallback(res, req, next)
OrderManager
+createOrder(options: Object): OrderModel
+modifyOrder(id: int, options: Object): OrderModel
+getOrderList(restaurant_id: int): List<Order>
RouteManager
-routes: List<Route>
-tokenManager: TokenManager
-middleware()
+use()
<<Interface>>
Route
+get(path: String, callback: Callable)
+post(path: String, callback: Callable)
+put(path: String, callback: Callable)
+delete(path: String, callback: Callable)
RestaurantRoute
-restaurantManager: RestaurantManager
-menuManager: MenuManager
-itemManager:ItemManager
-getCallback(res, req, next)
-postCallback(res, req, next)
-putCallback(res, req, next)
-deleteCallback(res, req, next)
AuthRoute
-authManager: AuthenticationManager
-getCallback(res, req, next)
-postCallback(res, req, next)
-putCallback(res, req, next)
-deleteCallback(res, req, next)
RestaurantManager
+modifyRestaurant(restaurant_id: int, options: Object): RestaurantModel
MenuManager
+modifyMenu(restaurant_id: int, options: Object): MenuModel
ItemManager
+createItem(restaurant_id: int, options: Object): ItemModel
+modifyItem(restaurant_id: int, id: int, options: Object): ItemModel
AuthUserModel
-token: String
-name: String
-surname: String
-email: String
-username: String
-address: String
-phone_number: String
+get()
+set()
AuthenticationManager
-instance: Passport
-tokenManager: TokenManager
+auth(username: String, password: String): AuthUserModel
PassportJS
Passport
TokenManager
-secretKey: String
+generateToken(user: UserModel, restaurant: RestaurantModel): String
+renewToken(token: String): String
+validate(token: String): int
<<Interface>>
Manager
-databaseManager: DatabaseManager
Visual Paradigm Online Diagrams Express Edition

Class Diagram for Server Side

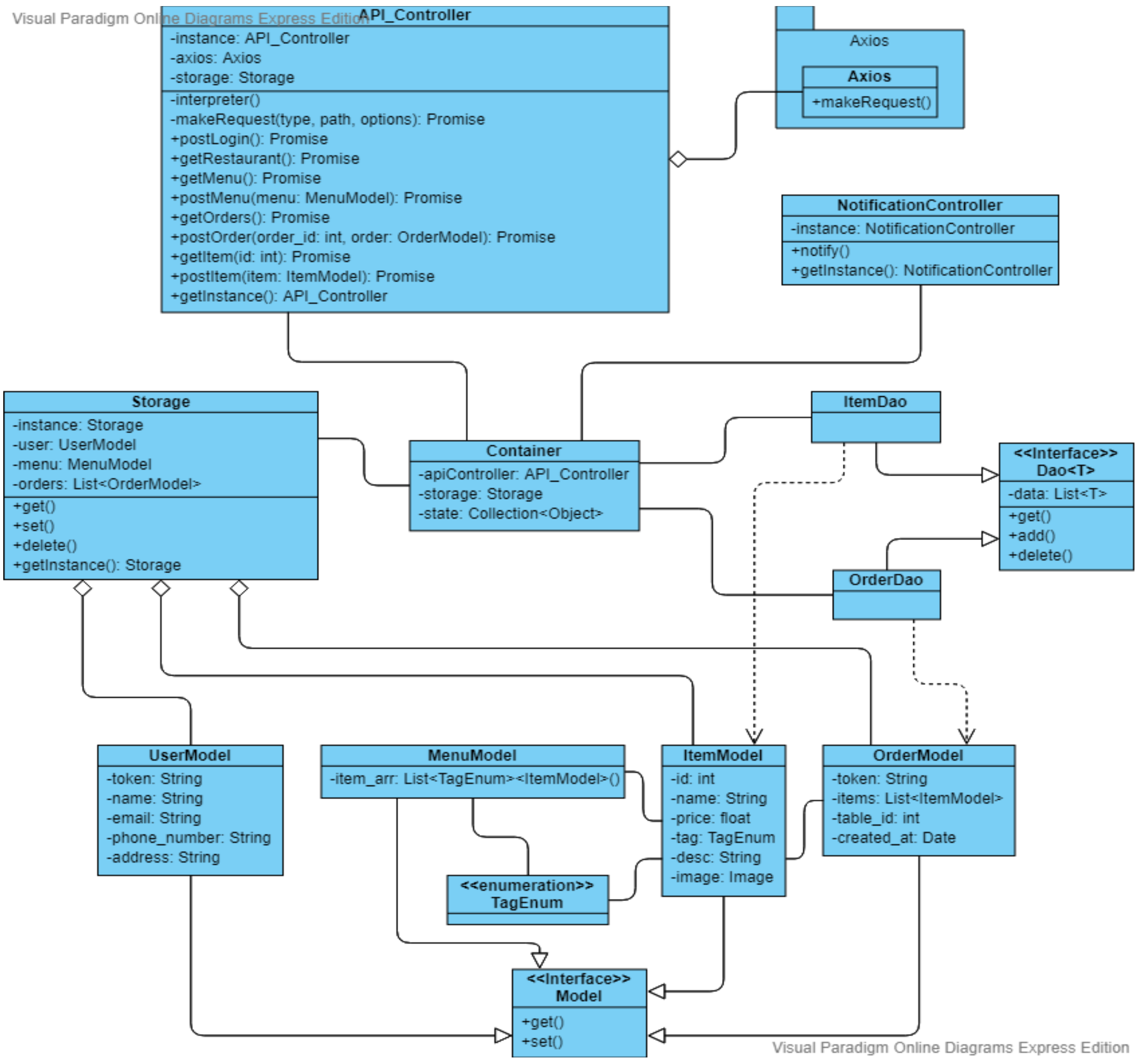
Class	**Description**
Controller	This class handles server status and requests via REST API concept. It uses middleware for security, flexibility and efficiency. It distributes requests to the instances of Route class, according to the request path.
Express	This class is provided by third-party package ExpressJS and creates a web framework or web service in the NodeJS environment [5].
Logger	This class manages logging of server-side.
TokenManager	This class generates, renews and validates JWT tokens via constant secret key.
Route <<Interface>>	This class manages http requests that are get, post, put and delete.
UserRoute ~ *Route*	This class manages http requests to the path is “/user/*”
OrderRoute ~ *Route*	This class manages http requests to the path is “/order/*”
RestaurantRoute ~ *Route*	This class manages http requests to the path is “/restaurant/*”
AuthRoute ~ *Route*	This class manages http requests to the path is “/auth/*”
Manager <<Interface>>	This interface creates a bridge with the DatabaseManager class.
UserManager ~ *Manager*	This class provides the business logic for user operations.
OrderManager ~ *Manager*	This class provides the business logic for order operations.
RestaurantManager ~ *Manager*	This class provides the business logic for restaurant operations.
MenuManager ~ *Manager*	This class provides the business logic for menu operations.
ItemManager ~ *Manager*	This class provides the business logic for item operations.
AuthenticationManager ~ *Manager*	This class provides the business logic for authentication operations.

AuthUserModel	This class is a model for authenticated users from client side and business side.
Passport	This class is provided by third-party package PassportJS and generates middlewares and authentication protocols for the web service by Express [3].
DatabaseManager	This class provides control of SQL database via GraphQL class.
GraphQL	This class is provided by third-party package GraphQL and controller for a database [4].
DatabaseModel <<Interface>>	This interface provides get and set methods.
UserModel ~ *DatabaseModel*	This class provides a model of the User entity in the database.
RestaurantModel ~ *DatabaseModel*	This class provides a model of the Restaurant entity in the database.
MenuModel ~ *DatabaseModel*	This class provides a model of the Menu entity in the database.
ItemModel ~ *DatabaseModel*	This class provides a model of the Item entity in the database.
OrderModel ~ *DatabaseModel*	This class provides a model of the Order entity in the database.
TagModel ~ *DatabaseModel*	This class provides a model of the Tag entity in the database.
TagEnum <<enumeration>>	This enum provides tags of items and is initialized dynamically.

Class Diagram for Client Side (Mobile App)

Class	**Description**
Container	This class provides UI classes access to local storage, connection with server, private state and listening notification service.
Storage	This class provides access to the local storage of the device. Any type of data can be saved as an encrypted form.
NotificationController	This is a singleton class that handles notifications of the app.
API_Controller	This class provides sending http requests -get, post, put, delete- to the server via

	instance of Axios class.
Axios	This class belongs to third-party package Axios and is able to send http requests to a specified url and a port [6].
Dao <<Interface>>	Dao stands for Data Access Object and isolates the layer of business from relational databases by providing abstract API [2].
RestaurantDao ~ *Dao*	This class provides a Dao for RestaurantModel class.
OrderDao ~ *Dao*	This class provides a Dao for OrderModel class.
Model <<Interface>>	This interface provides get and set methods.
UserModel ~ *Model*	This class provides a model of a User object.
RestaurantModel ~ *Model*	This class provides a model of a Restaurant object.
MenuModel ~ *Model*	This class provides a model of a Menu object.
ItemModel ~ *Model*	This class provides a model of an Item object.
OrderModel ~ *Model*	This class provides a model of an Order object.
TagEnum <<enumeration>>	This enum provides tags of items and is initialized dynamically.



Class Diagram for Business Side (Desktop App)

Class	**Description**
Container	This class provides UI classes access to local storage, connection with server, private state and listening notification service.
Storage	This class provides access to the local storage of the device. Any type of data can be saved as an encrypted form.
NotificationController	This is a singleton class that handles notifications of the app.
API_Controller	This class provides sending http requests -get, post, put, delete- to the server via

	instance of Axios class.
Axios	This class belongs to third-party package Axios and is able to send http requests to a specified url and a port [6].
Dao <<Interface>>	Dao stands for Data Access Object and isolates the layer of business from relational databases by providing abstract API [2].
ItemDao ~ *Dao*	This class provides a Dao for ItemModel class.
OrderDao ~ *Dao*	This class provides a Dao for OrderModel class.
Model <<Interface>>	This interface provides get and set methods.
UserModel ~ *Model*	This class provides a model of a User object.
MenuModel ~ *Model*	This class provides a model of a Menu object.
ItemModel ~ *Model*	This class provides a model of an Item object.
OrderModel ~ *Model*	This class provides a model of an Order object.
TagEnum <<enumeration>>	This enum provides tags of items and is initialized dynamically.

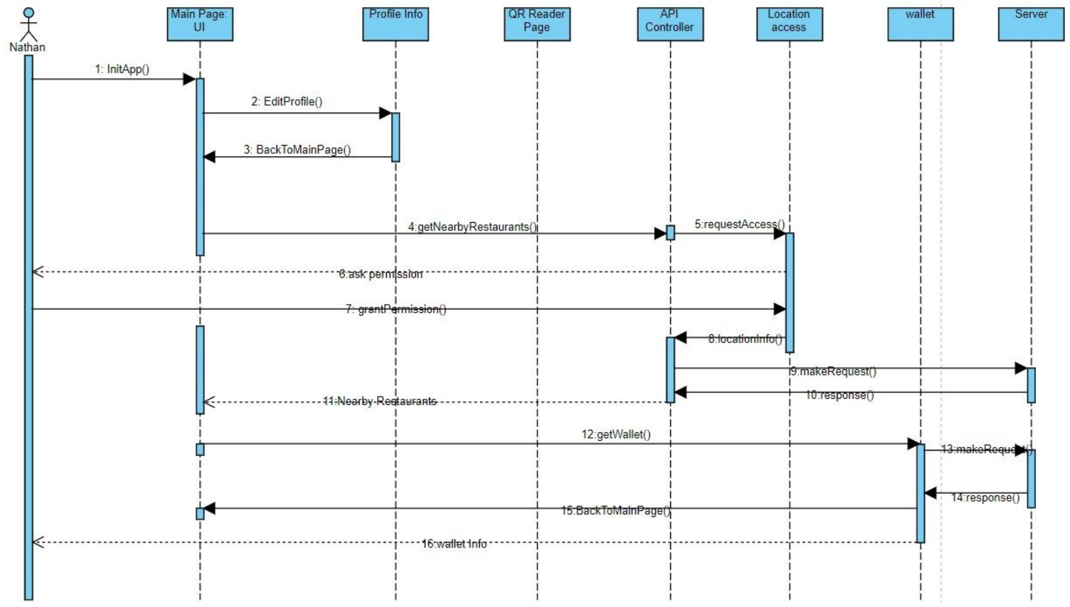
3.5.4 Dynamic Models

Sequence Diagrams

Sequence diagrams show how things work in an application. Unlike use case diagrams and scenarios it shows interactions between objects and functions.

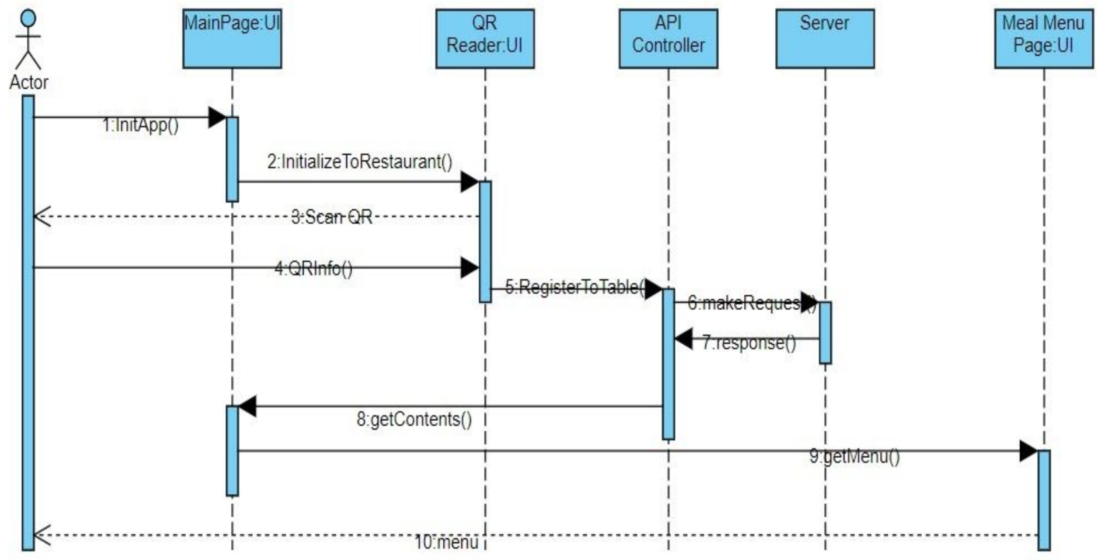
Main Menu Options for Client Side

This sequence illustrates that how Nathan loops inside main page by choosing several options such as show nearby restaurants , wallet and edit profile.



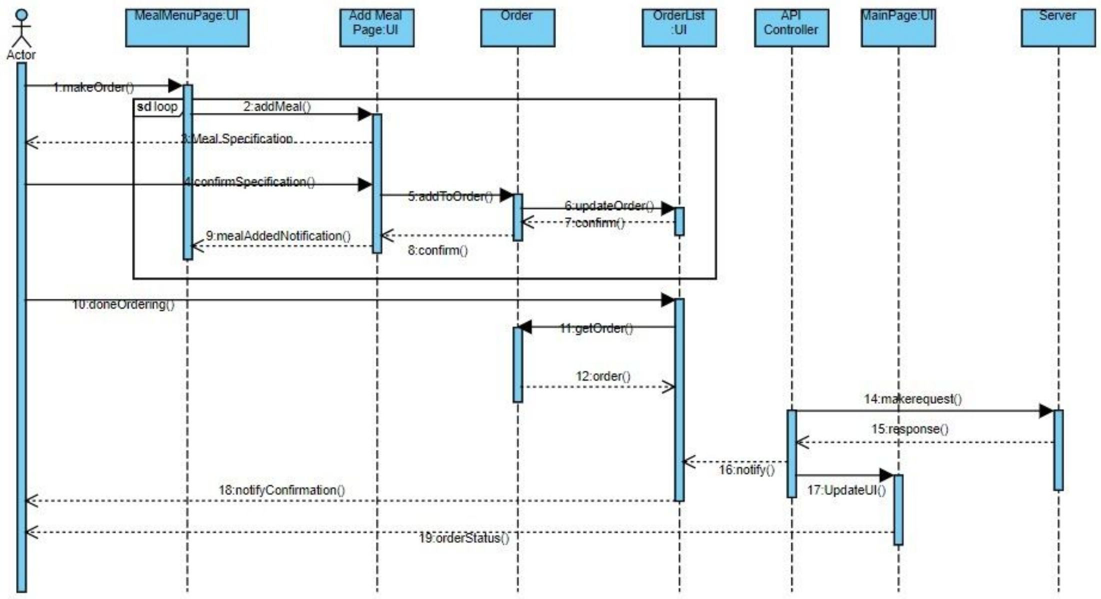
Scanning QR, Registering a Table and Getting a Menu for Client Side

This sequence shows that what is happening when Nathan comes to the restaurant and getting a menu by scanning the QR code.



Creating, Updating and Showing Order for Client Side

This diagram illustrates interactions between objects when Nathan wants to create or update an order or when he needs to see the status of the order.



Order Operations for Business Side

In this diagram, operations on the order that is applied by business side is shown.

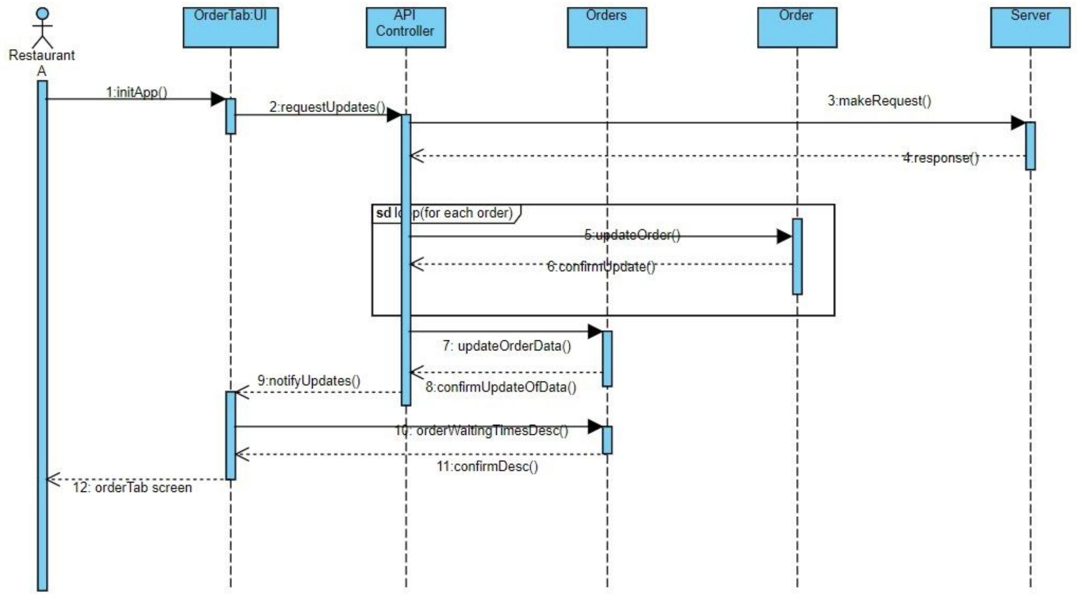
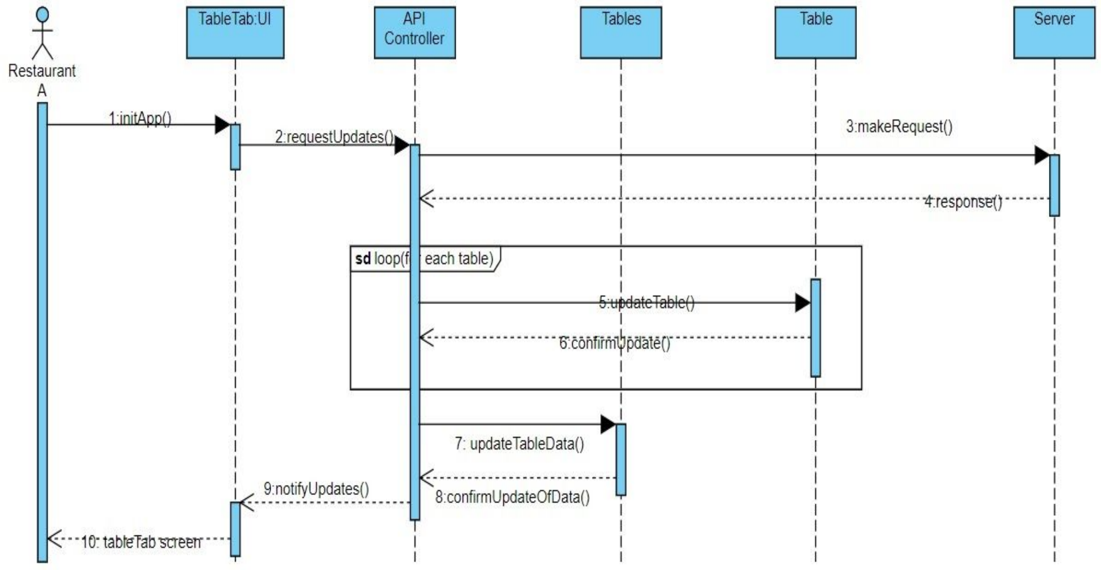


Table Operations for Business Side

This diagram shows which objects interact with each other in which way to make an operation on a table that is assigned to a specific restaurant.

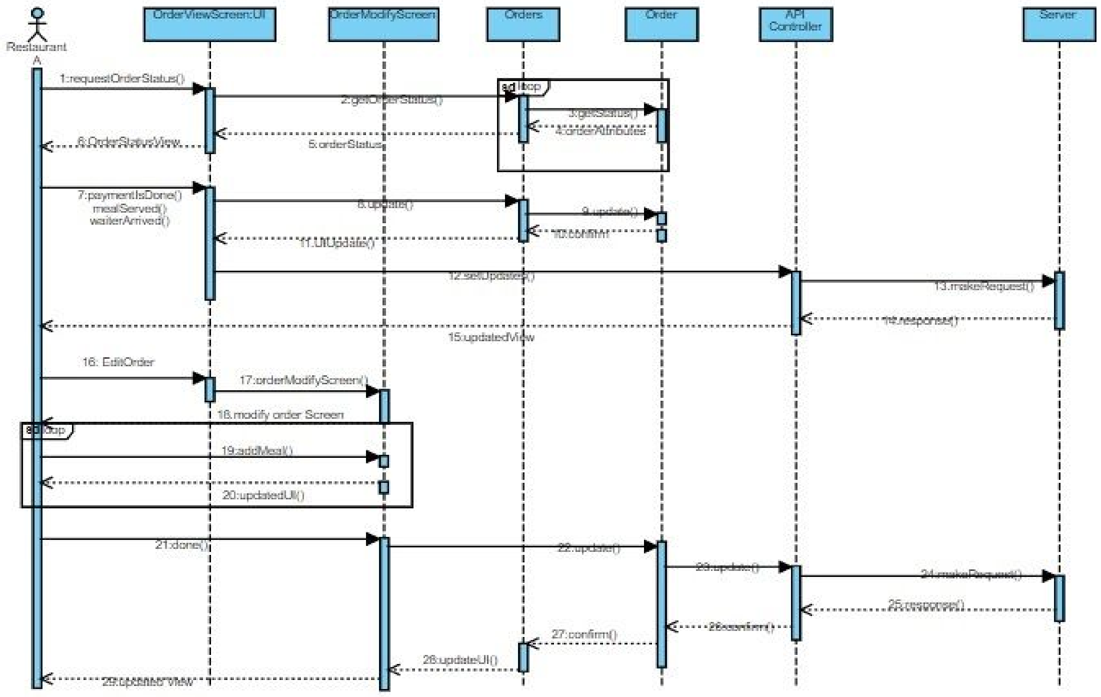


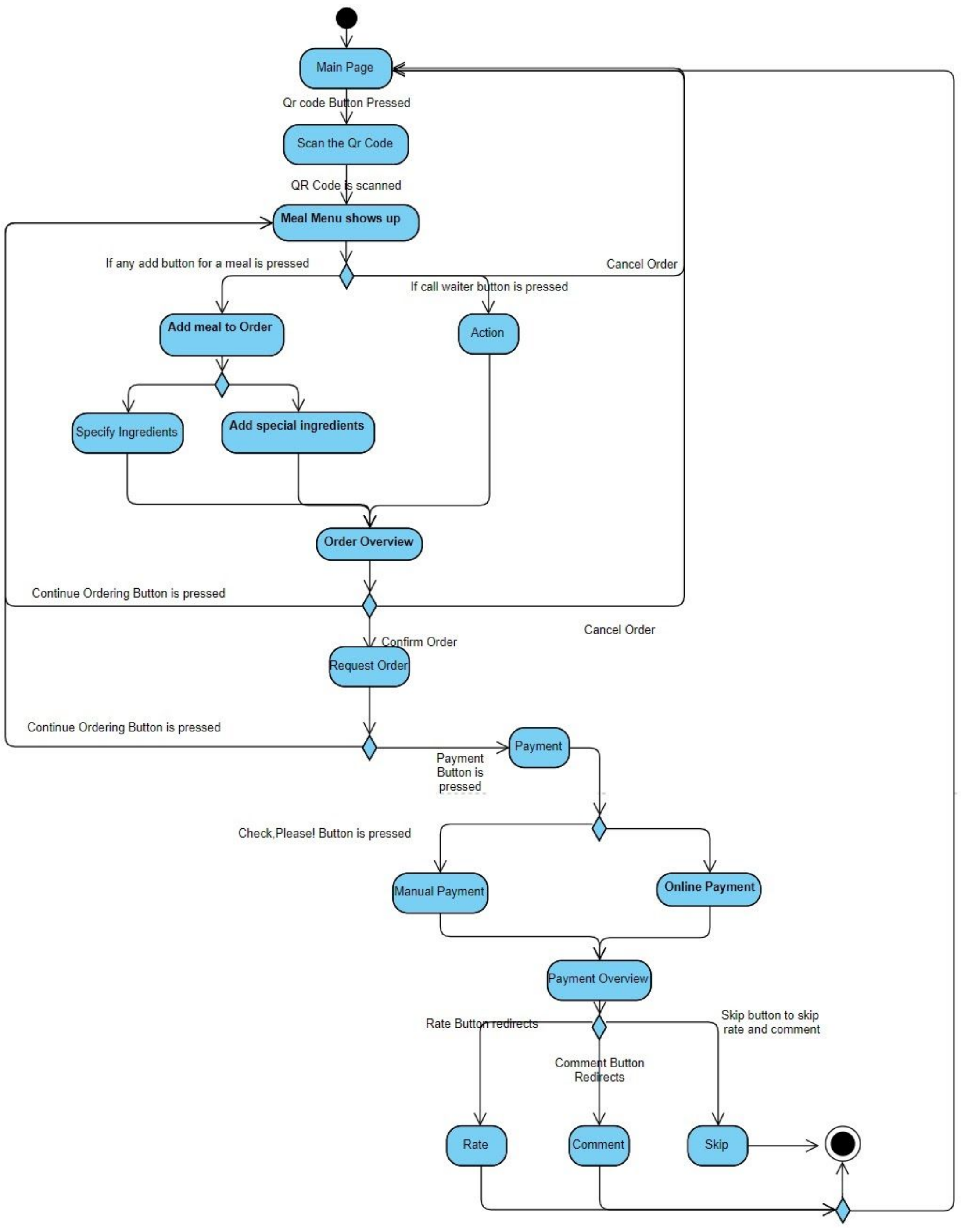
Order Control for Business Side

This diagram shows how order is controlled.

Client Side Activity Diagram

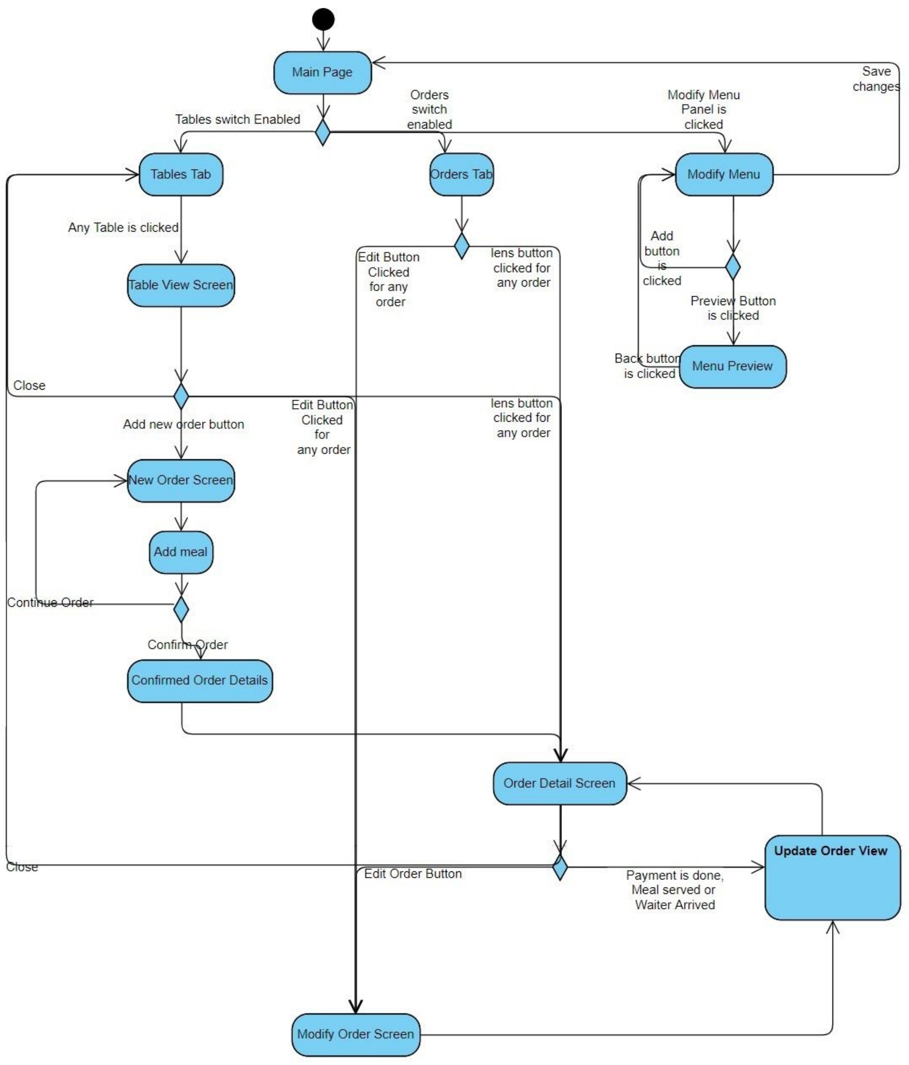
In addition to the sequence diagrams, main flow of the client side application is shown in the Activity diagram below.This diagram shows the main flow of events in clients side, instead of specific flow of events.Entry point is the login of the client to the application, and exit point is the payment in diagram.




Main Page
Qr code Button Pressed
Scan the Qr Code
QR Code is scanned
Meal Menu shows up
If any add button for a meal is pressed
Cancel Order
If call waiter button is pressed
Add meal to Order
Action
Specify Ingredients
Add special ingredients
Order Overview
Continue Ordering Button is pressed
Cancel Order
Confirm Order
Request Order
Continue Ordering Button is pressed
Payment
Payment Button is pressed
Check,Please! Button is pressed
Manual Payment
Online Payment
Payment Overview
Rate Button redirects
Skip button to skip rate and comment
Comment Button Redirects
Rate
Comment
Skip

Business Side Activity Diagram

Main flow for the business side of the the program is shown below. There is main flow of the program after initialization. Entry point is the point that user login to the program. There is no exit point because business side is kept open . Logout may be exit conditions but it is not resides in the main flow.



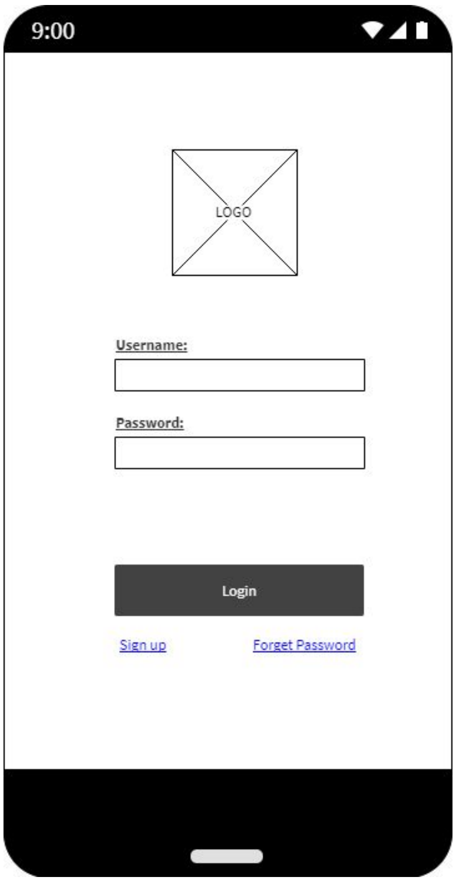
3.5.5 User Interface

User Interface mockups are shown in this topic. There are user client and administrative clients subtopics for both sides of the program. Administrative side is a desktop application so it it shown in a desktop monitor, where client side is a mobile application so it is shown in mobile phone screen.

3.5.5.1 User Client

3.5.5.1.1 Login Page

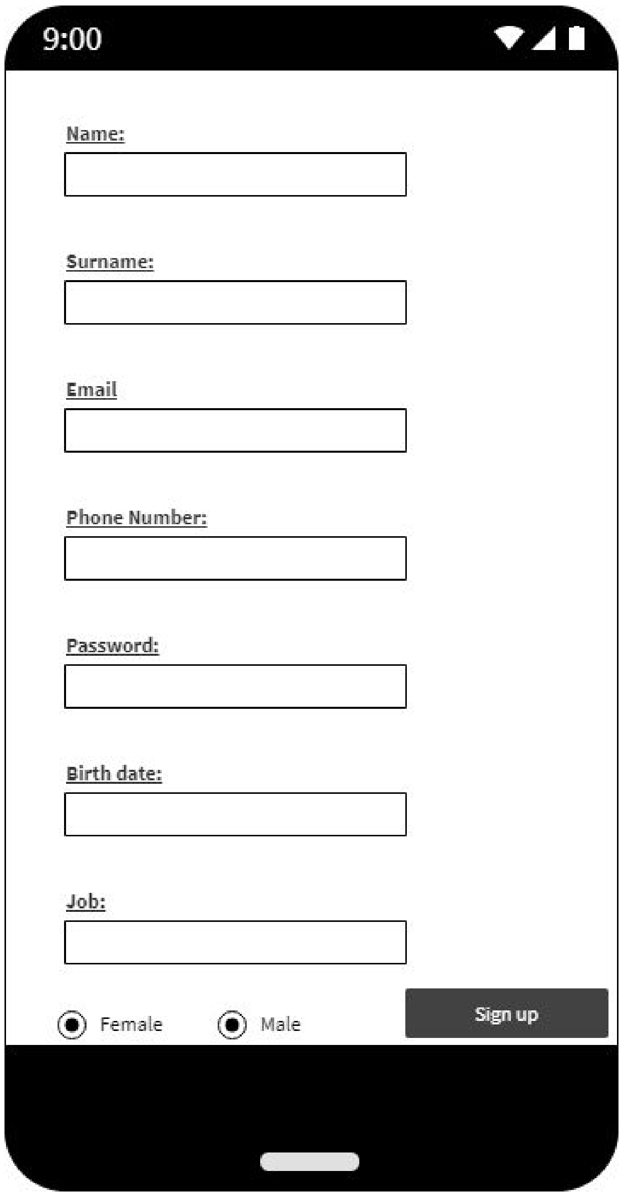
First screen that is seen when the application is opened by a user. There are 2 input areas that are expected to be filled by the user with correct identity information to use the program with specified authorization. If the user did not register yet, he/she should register by clicking the sign up button and be directed to the registration page. If a user forgets his/her password, he/she should click the forgot password button to get his/her new password to his/her specified email address. If a user has already registered, he/she can login to use the program after filling required input areas.



Login page (1.1)

3.5.5.1.2 Sign up Page

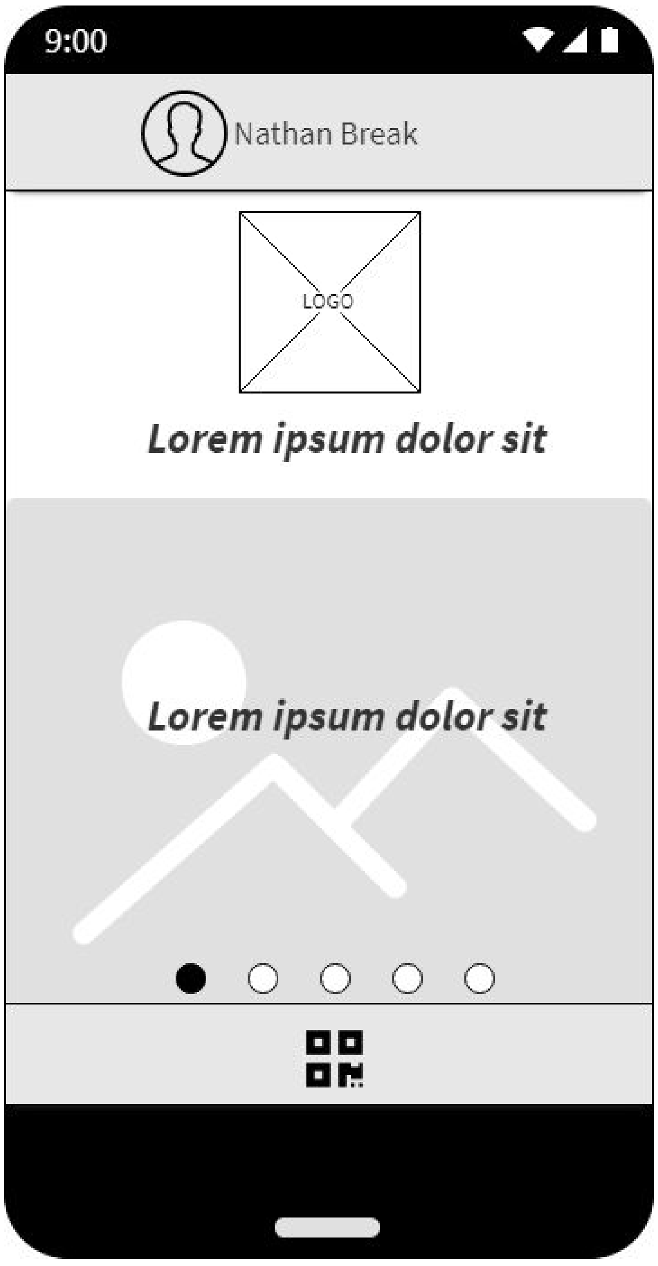
Sign up Page can be directed from the login page. Users should use Sign up Page to register the system. Name, surname, email and password are the required areas to register; where phone number,birth date, job and gender are the areas that are requested just for creating statistics and providing recommendations for users. Therefore, these areas do not have to be filled. Once a user completes to required areas, he/she can login with his/her email and password that he/she used in registration.



Sign Up page (1.2)

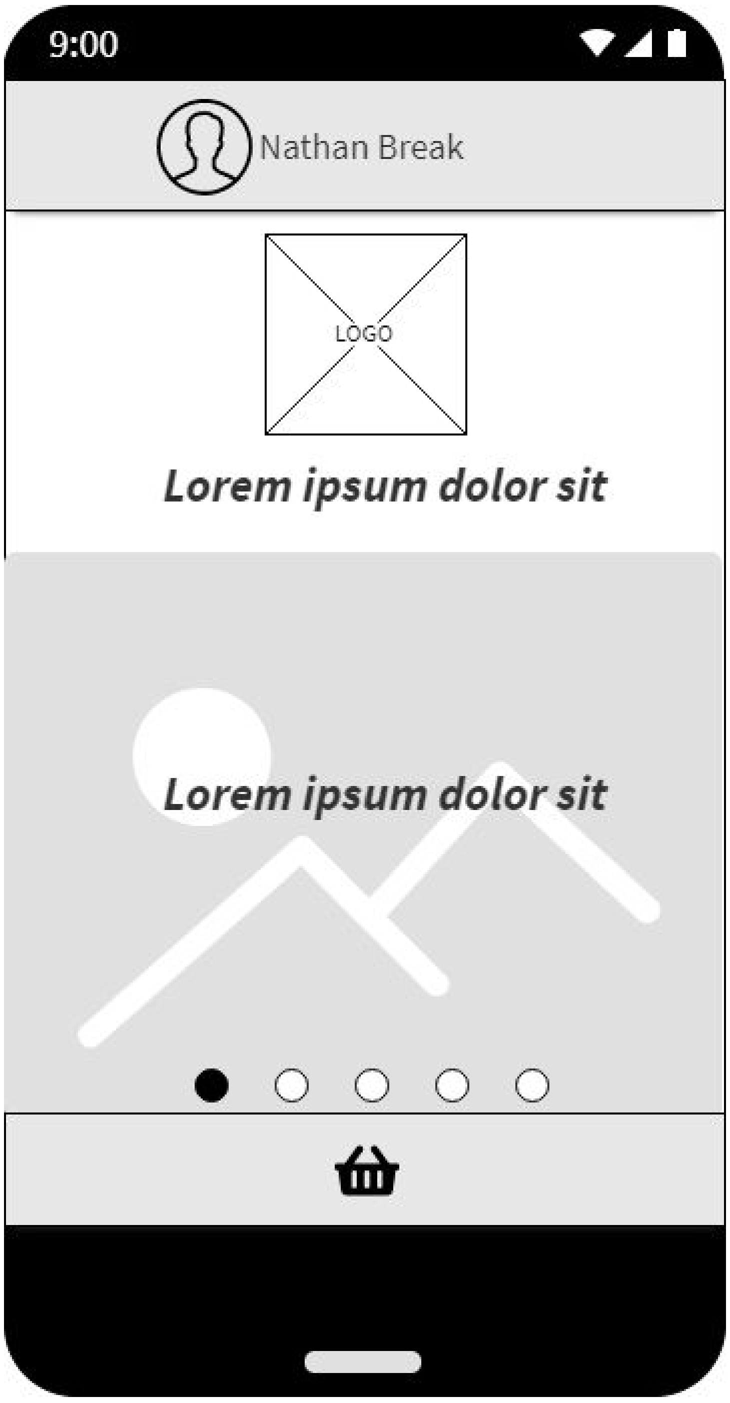
3.5.5.1.3 Main Page

The Main Page is the first page that is seen after the login. At the top, the logo and the catchy motto of the application is seen. Under it, there is a slide menu which shows the special offers of restaurants that the user went to before. In the footer panel, there is a button in the shape of a QR Code. Users should press it to be directed to the QR reader page. There is also a navigation menu for preferences.



Main page - 1 (1.3)

When the user scans the QR Code in the table, the QR button is replaced by a basket button. Users can be directed to the Order Page by clicking it whenever they want. This button appears and is enabled only after the user scans the QR code and changes back to QR button after payment is done.



Main page - 2 (1.4)

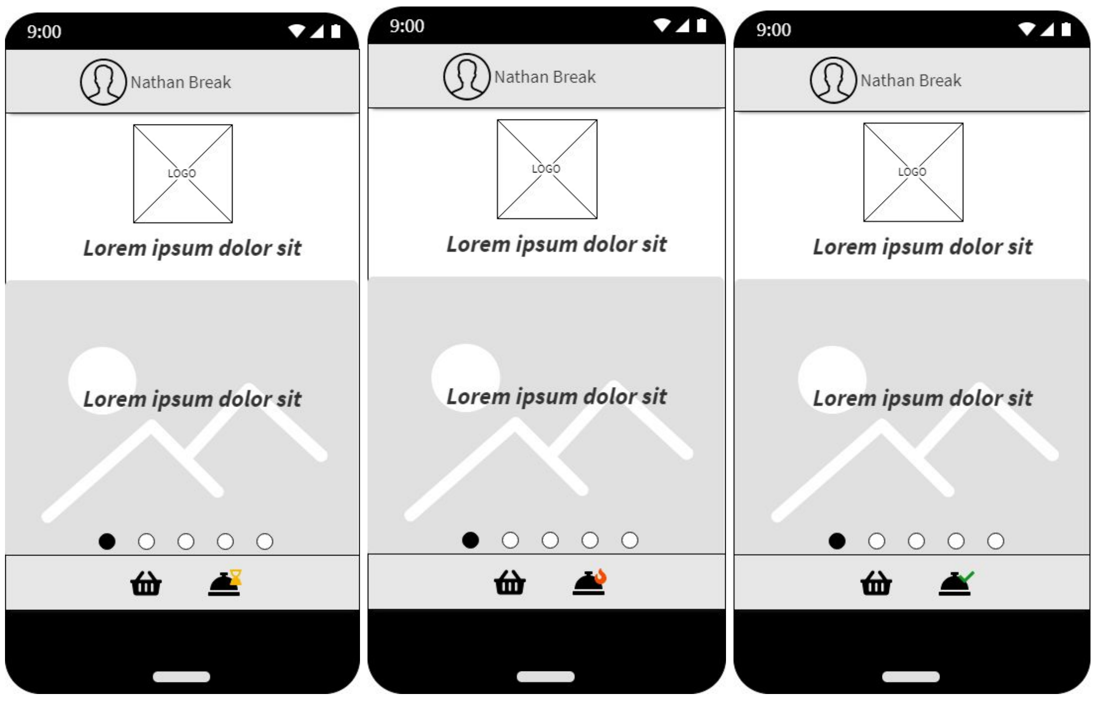
When complete ordering, A plate-cover icon appears next to the basket button. This icon shows the status of order.

Yellow Hourglass: It means order has been sent to the database and directed to the restaurant, waiting for confirmation of the restaurant account.

Red Fire: It means order has been confirmed by restaurant account and is being prepared to be served.

Yellow Check: It means Order is stated as ready by restaurant account and expected to be served soon.

When the serving of the meal is done, the waiter reminds you to press the plate-cover icon to confirm the meal is served. Once the plate-cover icon is pressed, it disappears until a new order is given.



Main page - Order status: Waiting (1.5.1)

Users can use the nav menu to use preferences. Preferences are:

Wallet: Redirects user to Wallet Page to edit his card info or to do payment.

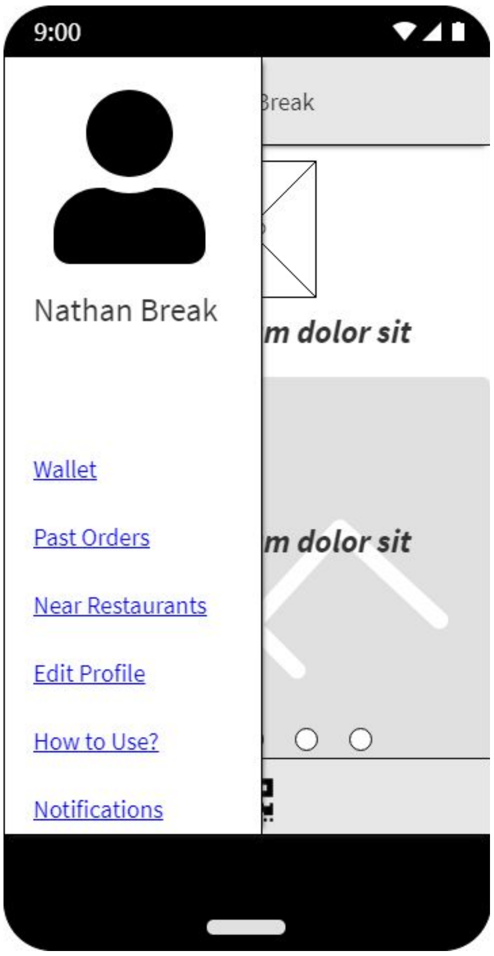
Nearby Restaurants: Asks permission for location information and redirects user to the Nearby Restaurants Page.

Past Orders: Redirects user to Past Orders Page where user can see his/her order history.

Edit Profile: Redirects user to edit profile page, which is the same with register page.

How to Use: Redirects user to internet page of application to inform him/her about use.

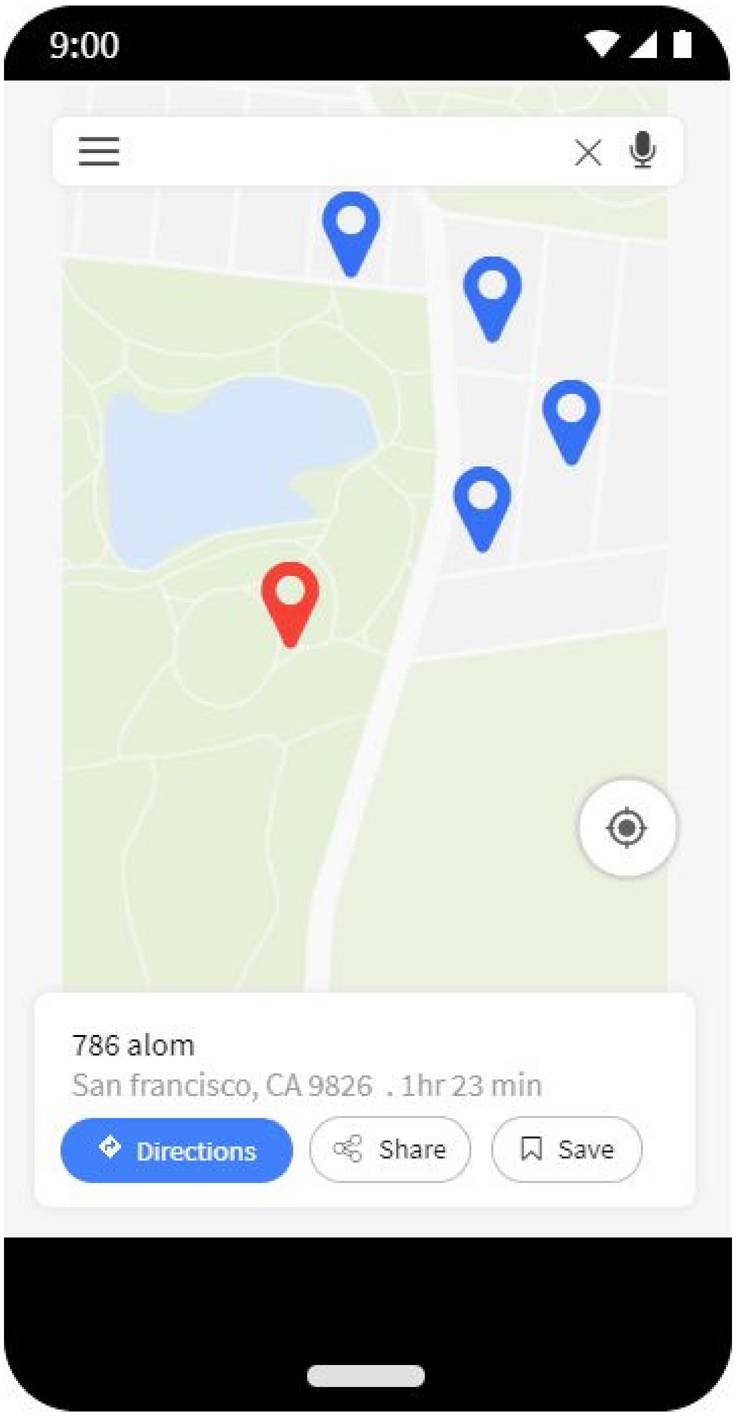
Notifications: Shows restaurant announcements as pop-ups.



Navbar (1.6)

3.5.5.1.4 Nearby Restaurants Page

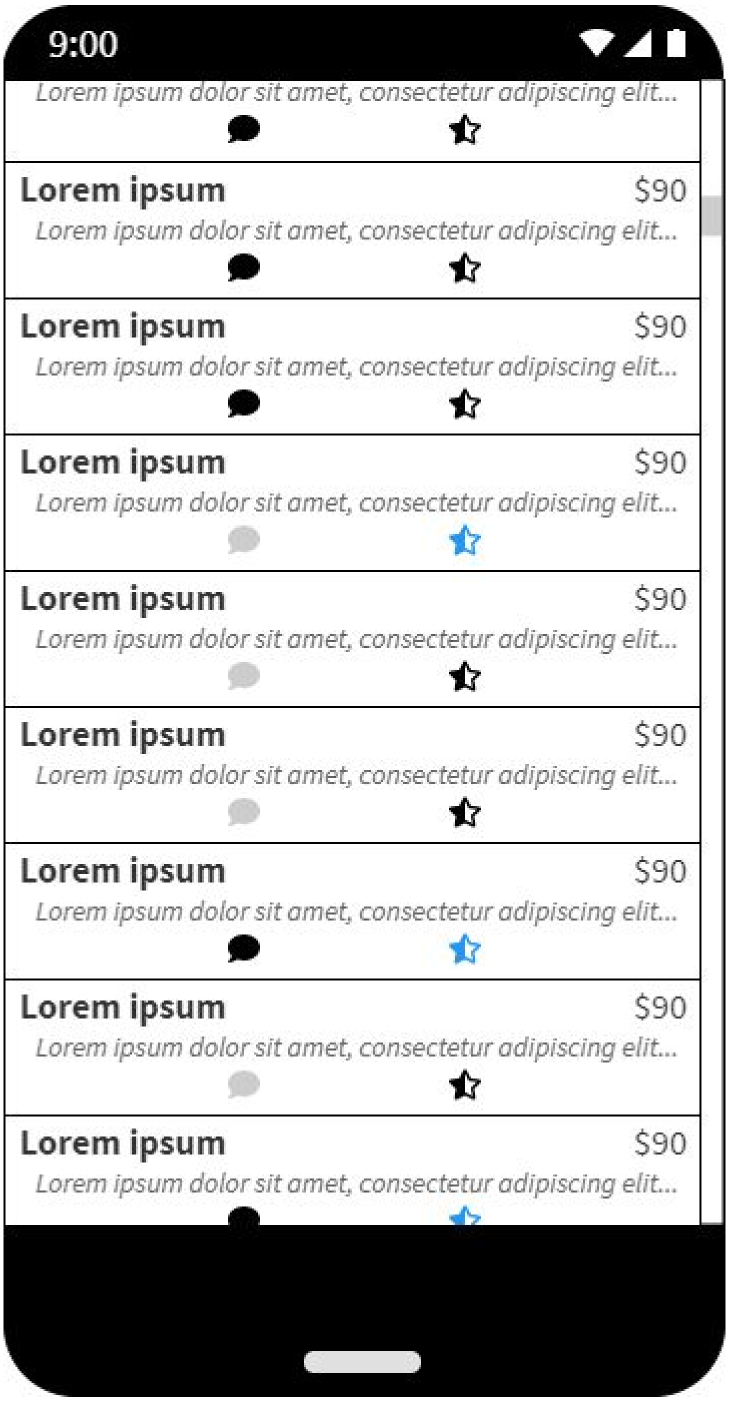
Once the “Nearby Restaurants” button is pressed, the application asks for permission to get location information. User can see nearby restaurants that use QRDER system and their intensity(empty tables, number of clients/orders etc.).



Nearby Restaurants page (1.7)

3.5.5.1.5 Past Orders Page

Users can see their past orders on this page. Also comments and rating can be attached to past orders and meals that were ordered before. User can get price,date and description information of past orders.



9:00

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

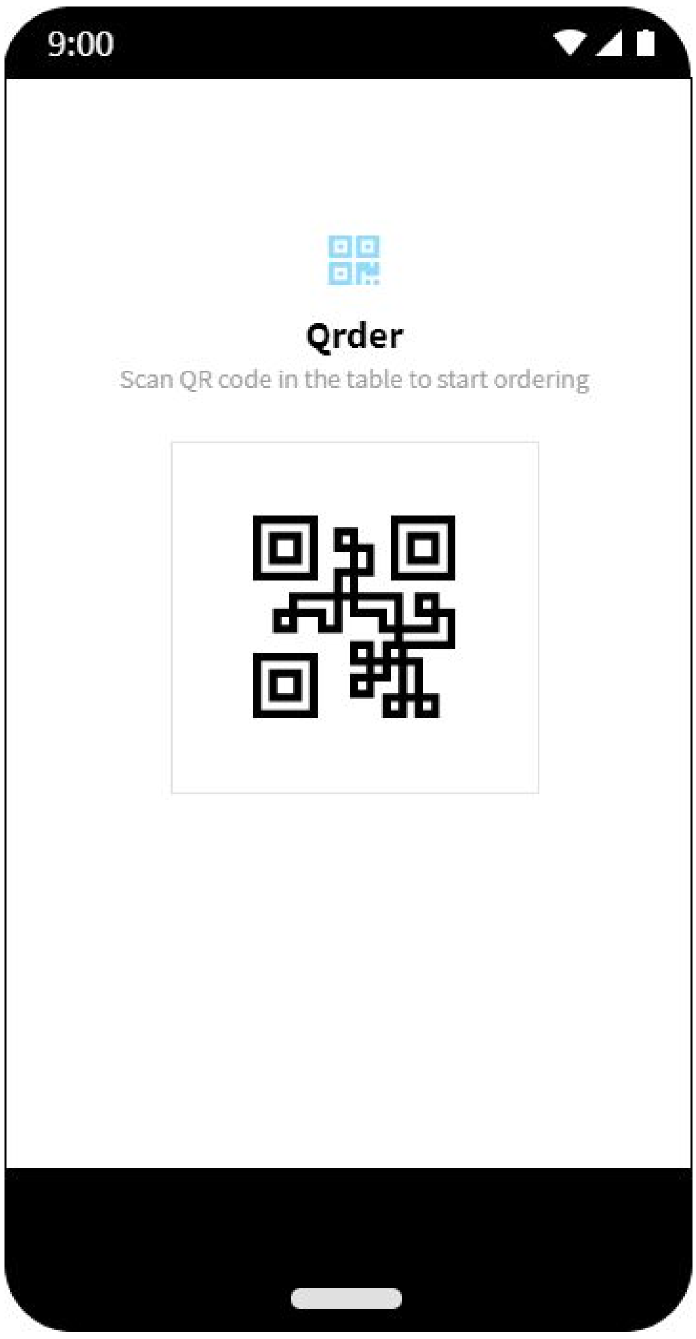
Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Lorem ipsum $90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Past Orders page (1.8)

3.5.5.1.6 QR Reader Page

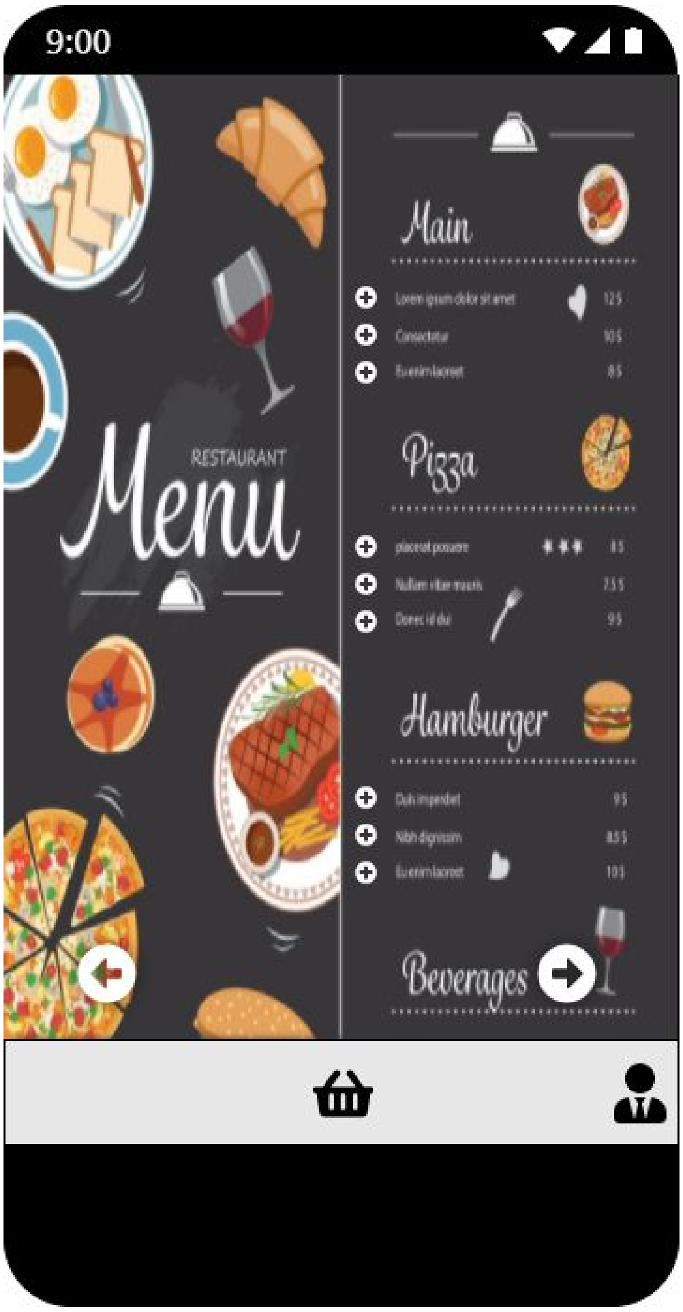
User scans the QR Code to see the meal list and orders some meals. Once a user clicks the QR button in the main Page, QR Reader Screen below appears and expects a QR Code to be scanned by the user. When QR Code on the restaurant table is scanned, the user sees the meal menu and can order something.



QR Reader page (1.9)

3.5.5.1.7 Meal Menu Page

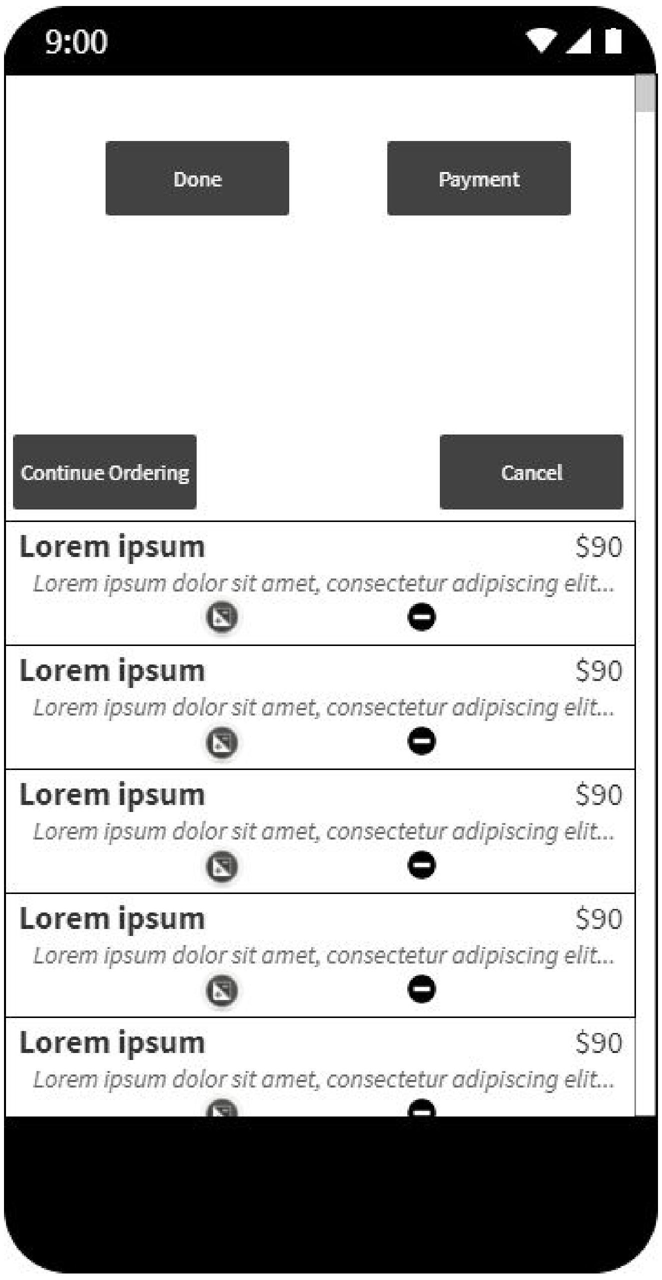
When the QR Code on the table of the restaurant is scanned by the user on QR Reader Page, the menu of the restaurant appears on the phone screen like the one is shown below. Users can specify their order by adding his offers from the menu on this page. Users also can call a waiter by using the button on the right-down corner of the page for any other request.



Menu page (1.10)

3.5.5.1.8 Order Page

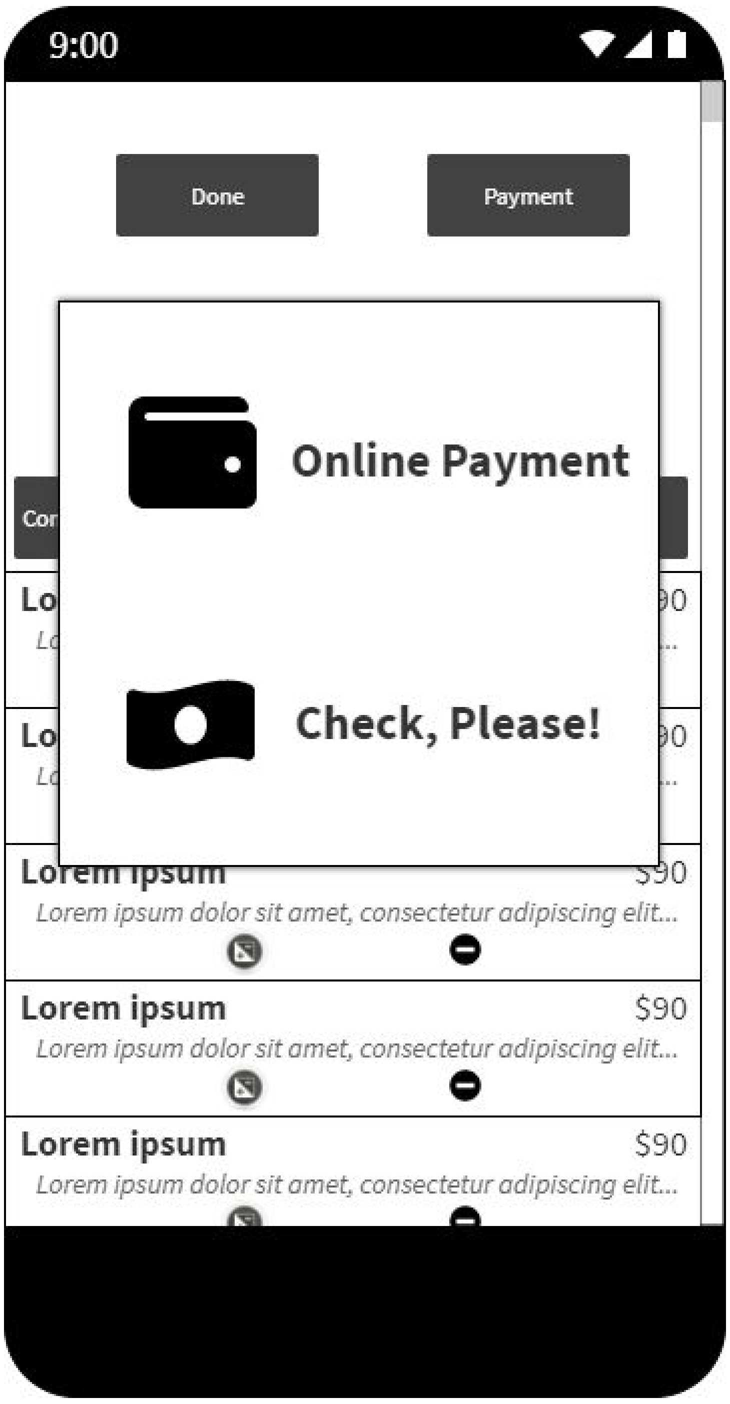
Order page shows the orders of the users. When users click the "done" button, order is sent to the database and the restaurant will be aware of the order. Users can order something new by clicking the "Continue Ordering" button. Payment button should be used when the user finishes his meal and decides to pay and leave. All orders and Order page can be accessed until the payment is done.


9:00
Done
Payment
Continue Ordering
Cancel
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Order page (1.11.1)

Payment Options

When the payment button is clicked, payment options are shown in a pop-up screen which is shown in the figure below. Users can pay normally with selecting "Check, Please!" option or can pay online with selecting "online payment" option and specifying their wallet.

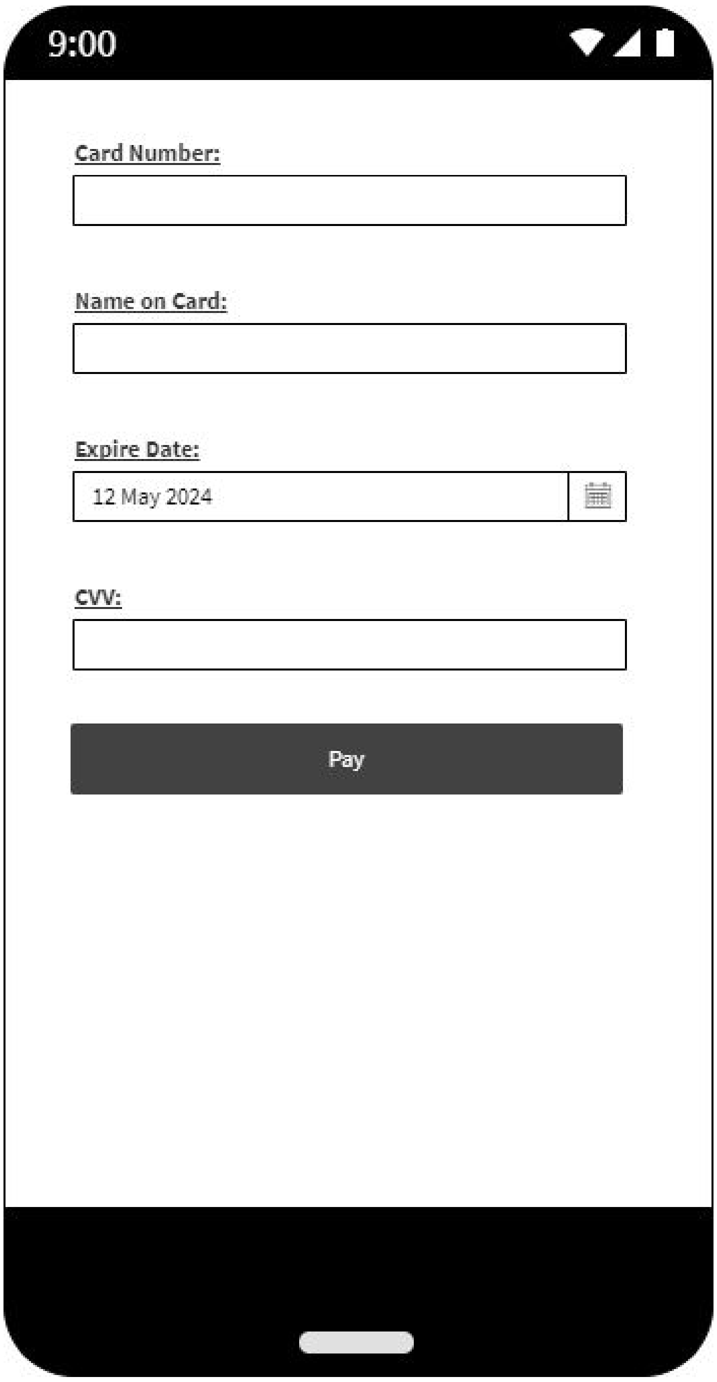

9:00
Done
Payment
Online Payment
Check, Please!
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...
Lorem ipsum
$90
Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Order page - Payment options (1.11.2)

3.5.5.1.9 Online Payment Page

To use:

- Online payment by specifying a credit card,
- Transfering money to the account of the application to do quick payment(like sodexo).

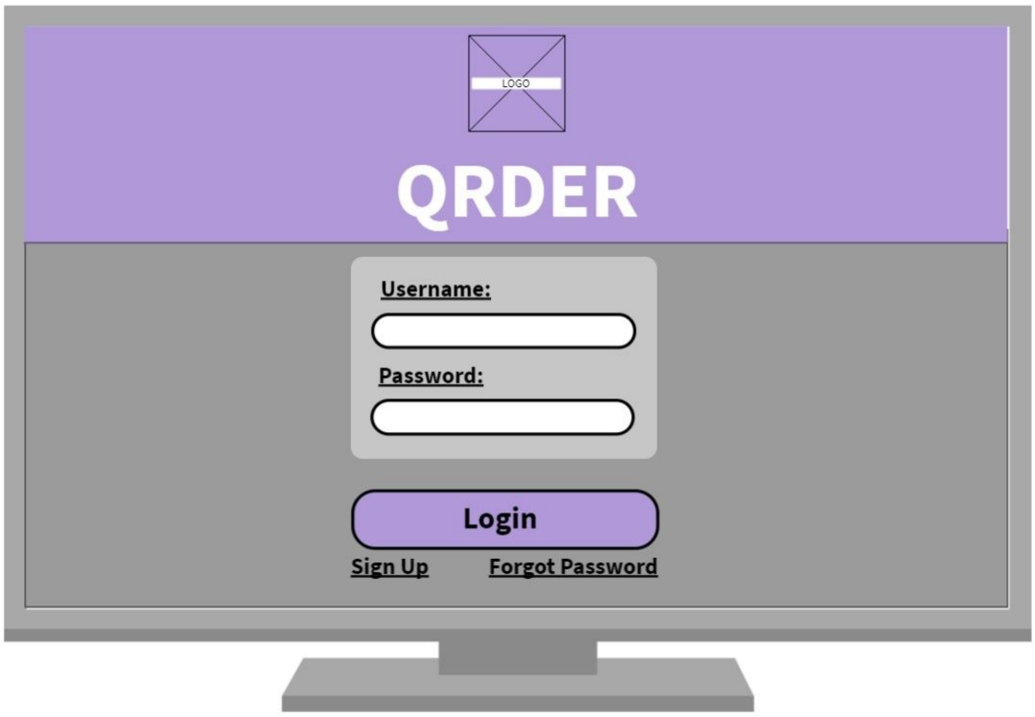


Online Payment page (1.12)

3.5.5.2 Administrative(Restaurant) Client

3.5.5.2.1 Login Screen

First screen that is seen when the application is opened by a user. There are 2 input areas that are expected to be filled by the user with correct identity information to use the program with specified authorization. If the user did not register yet, he/she should register by clicking the sign up button and be directed to the Web page of the system. If a restaurant wants to use this system, owner of the restaurant should contact with QRDER HR and demands the set up for the restaurant. This setup includes authorization, synchronisation of tables with QR Stickers and Registration for program.If a user forgets his/her password, he/she should click the forgot password button to get his/her new password to his/her specified email address. If a user has already registered, he/she can login to use the program after filling required input areas.



Login Page(2.1)

3.5.5.2.2 Main Screen

Main Screen of the administrative client composed of 2 main screen tabs, tables and the orders.On the left-upper corner, there are profile edit settings and log out buttons. Right of the User panel, There is the “Current Menu” panel, which can be pressed to go to the “Modify Menu” screen. On the right-upper corner of the screen,there is a statistics panel which shows the number of the corresponding information as it is told below.

☑ : Number of tables with delivered order

👛 : Number of tables that waits payment

👤 : Number of tables that waits waiter

⏳ : Number of tables that waits their order to be served

👤 : Button with specified profile picture to show profile information

⚙ : Settings button to arrange scaling and color settings of the UI

✎ : Edit button to edit profile informations

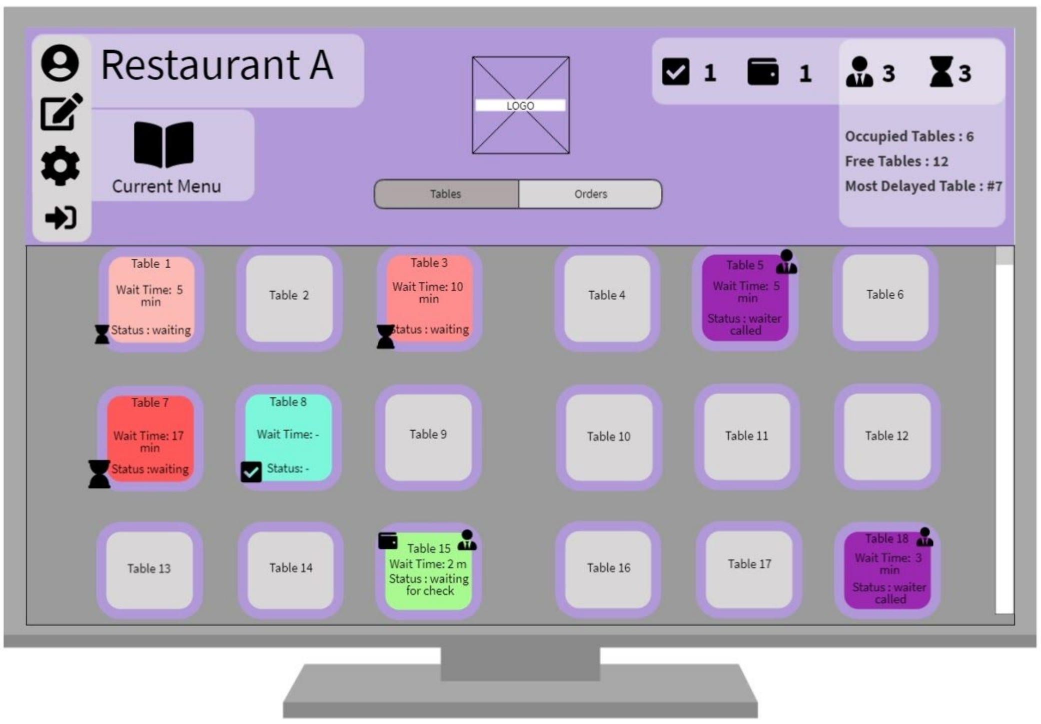
➜ : Logout Button

Tables Tab

Tables tab shows the status of the tables and the orders belong to them in the restaurant. Colors of the tables are getting warmer to notify restaurant employers as the wait time increases. Icons attached to the table are also getting bigger as wait time increases.

- Red scaled colors are specified for tables that wait orders.
- Purple is specified for tables that wait waiter
- Blue is specified for tables that are delivered with their orders.
- Green is specified for tables that wait for payment

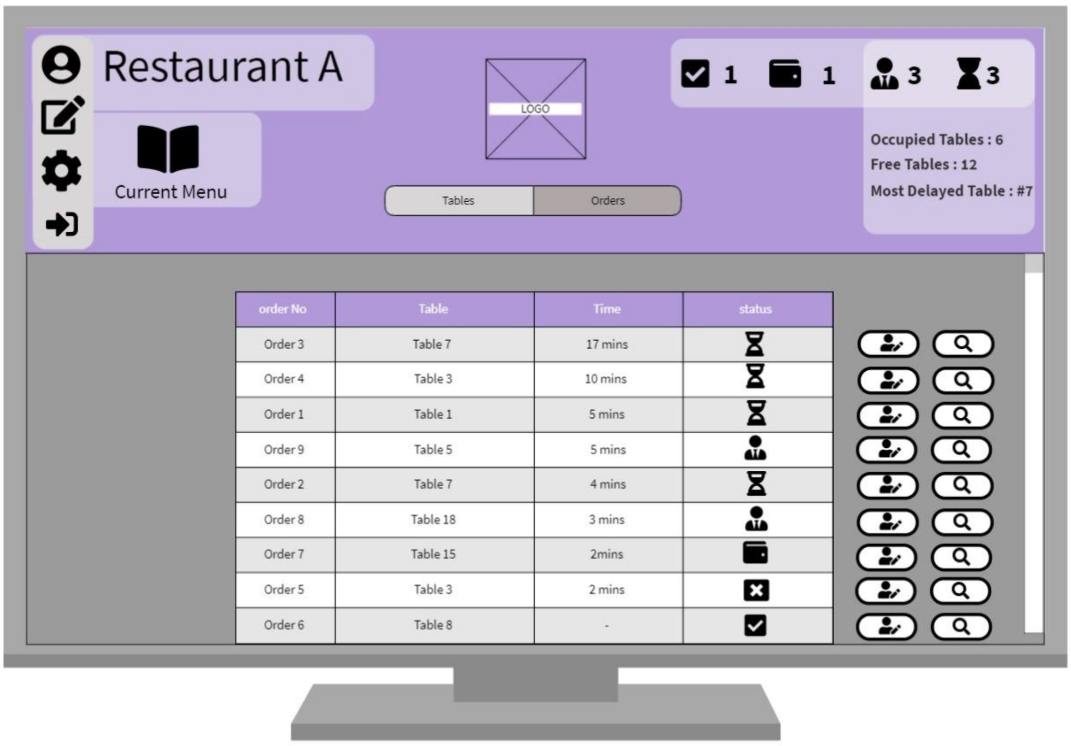
Table Detail/Modification Screen can be redirected by clicking on any table to modify orders of the specified table.



Main Page-Tables Screen(2.2.1)

Orders Tab

Order tabs show all orders that exist for currently occupied tables in the restaurant. One table can have more than one order. Orders shown in descending order respect to wait time. There are edit and show details buttons for each order in the list. Show details button is being used to be redirected to Order Detail/Status Screen for order that corresponds to clicked button. Edit button redirects to Modify Order Screen. Icons correspond to situations that are mentioned in previous parts.

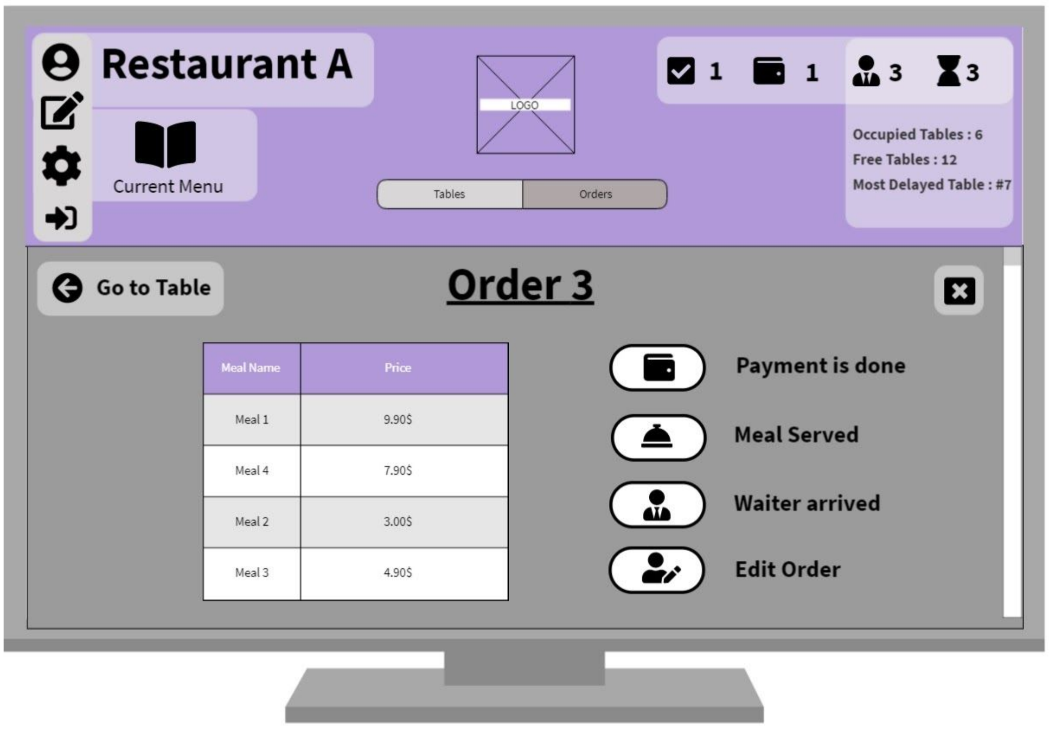


order No	Table	Time	status
Order 3	Table 7	17 mins	⏳
Order 4	Table 3	10 mins	⏳
Order 1	Table 1	5 mins	⏳
Order 9	Table 5	5 mins	👤
Order 2	Table 7	4 mins	⏳
Order 8	Table 18	3 mins	👤
Order 7	Table 15	2mins	👛
Order 5	Table 3	2 mins	☒
Order 6	Table 8	-	☑

Main Page-Orders Screen(2.2.2)

3.5.5.2.3 Order Detail/Status Screen

Status and contents of the order can be modified on this screen. If payment for a corresponding order is done normally, then the employer of the restaurant clicks the “payment is done” button. If the order is served to the customer, then the “meal served” button should be clicked. After the waiter visited the calling table, the ”waiter arrived” button should be clicked. Edit order button redirects the screen to the Modify Order Screen where new orders can be added or current orders can be removed.

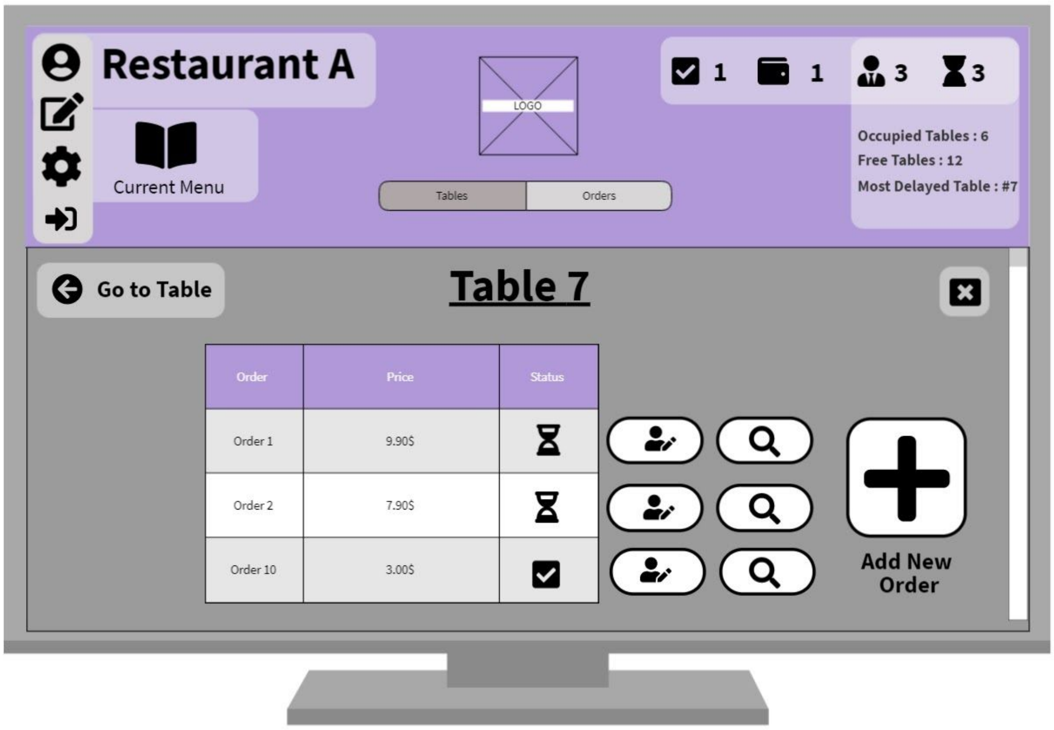


Meal Name	Price
Meal 1	9.90$
Meal 4	7.90$
Meal 2	3.00$
Meal 3	4.90$

Order Detail/Status Screen(2.3)

3.5.5.2.4 Table Detail/Modification Screen

Edit and Details buttons make the same redirections with the orders tab. This screen only lists the orders of specified tables.There is an additional “Add New Order” button to add orders manually to the table. This service provides currently used order tracking system for restaurants, so that the customers that do not use the QRDER can also be served by the restaurant. “Add New Order” button redirects to “Create Order Screen".



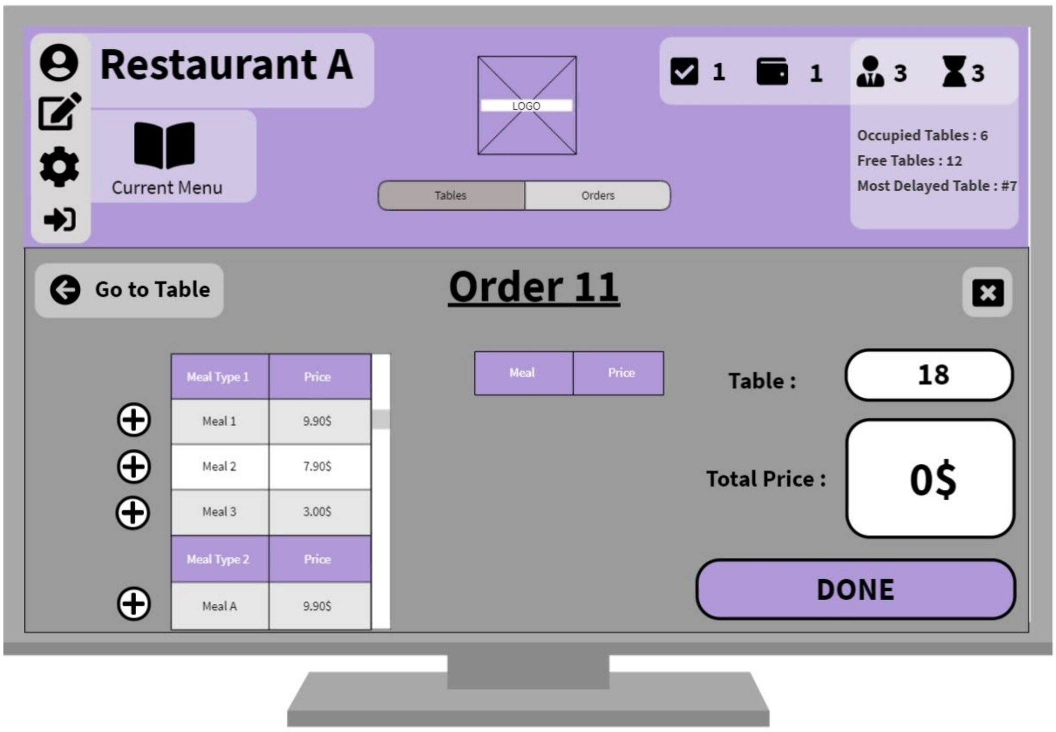
Occupied Tables : 6
Free Tables : 12
Most Delayed Table : #7

Order	Price	Status
Order 1	9.90$	⧗
Order 2	7.90$	⧗
Order 10	3.00$	✓

Table Detail/Modification Screen(2.4)

3.5.5.2.5 Create Order Screen

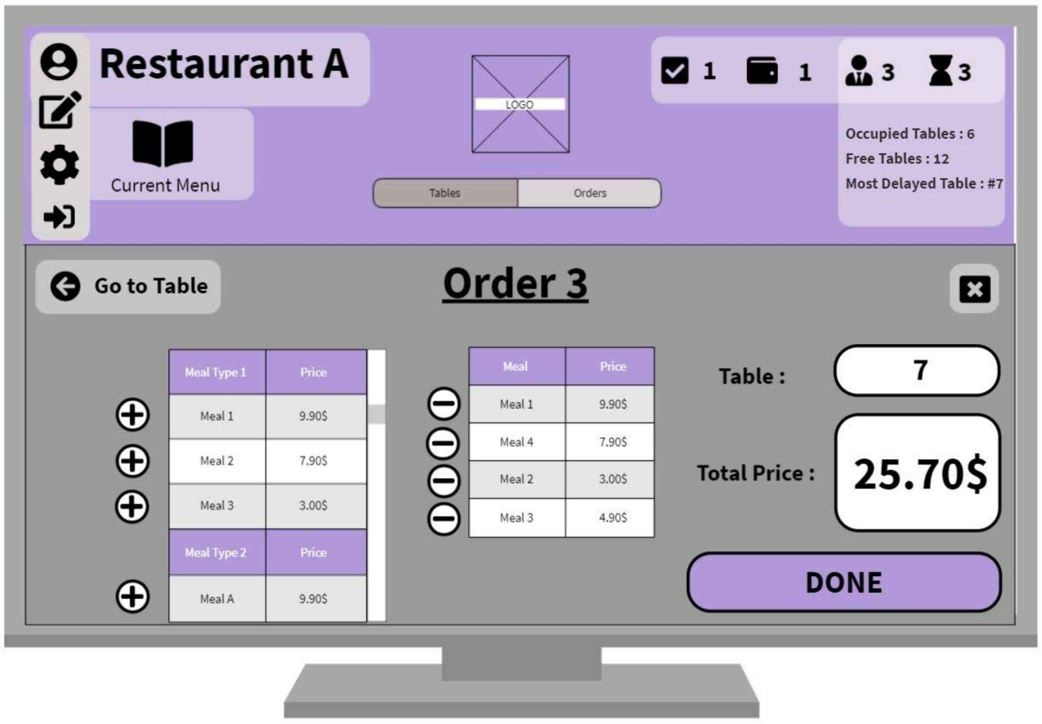
Create order screen is used to create orders normally. Waiter gets the order from the customer and saves it via this interface. When the “add new order” button is clicked, the system creates a new order with a new order number and order id in the database and shows its details to employers of the restaurant to specify orders manually.



Create Order Screen(2.5)

3.5.5.2.6 Modify Order Screen

Same functionality with “Create Order Screen”, this screen is used to modify existing orders. New orders can not be created via this interface, only existing ones can be modified.



Meal Type 1	Price
Meal 1	9.90$
Meal 2	7.90$
Meal 3	3.00$
Meal Type 2	**Price**
Meal A	9.90$

Meal	Price
Meal 1	9.90$
Meal 4	7.90$
Meal 2	3.00$
Meal 3	4.90$

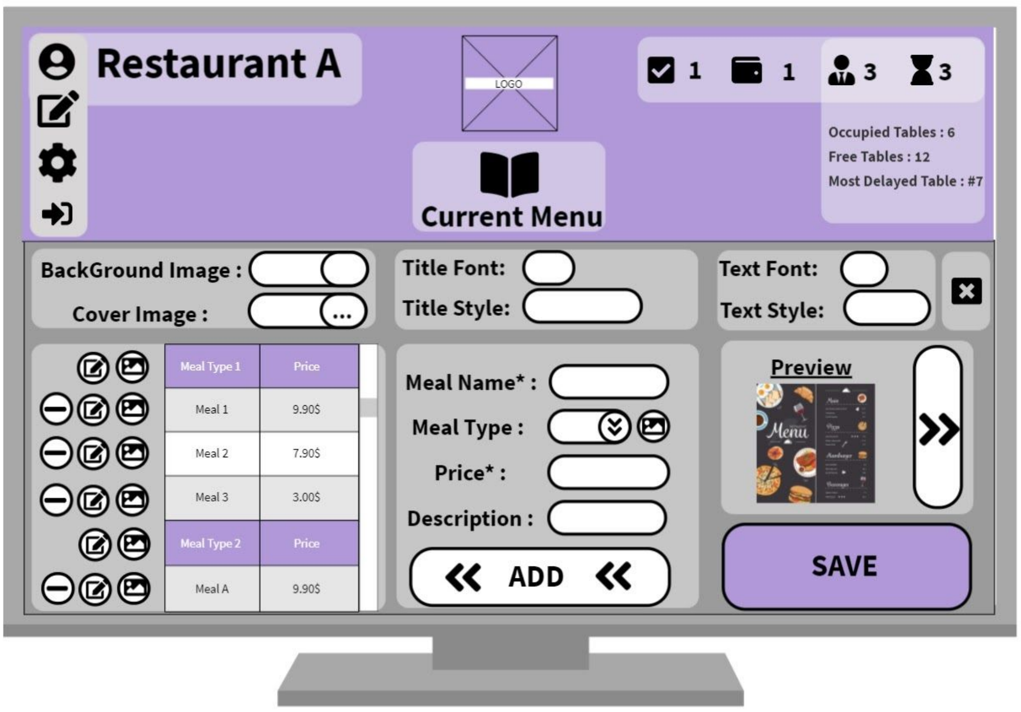
Table : 7

Total Price : 25.70$

Modify Order Screen(2.6)

3.5.5.2.7 Create/Modify Menu Screen

Current meal menu can be modified through this interface. New meals can be added to the meal list or existing ones can be removed from the list. Restaurants can also arrange Fonts and styles from existing ones or upload new assets to create more interesting designs. To strengthen the menu design, restaurant can specify the cover and background images of the menu. “>>” button inside the preview panel is used to show the preview version of the menu before saving it. After all modifications are done, the restaurant can update the menu by pressing the “save” button to save the arrangements in the database.



Meal Type 1	Price
Meal 1	9.90$
Meal 2	7.90$
Meal 3	3.00$
Meal Type 2	**Price**
Meal A	9.90$

Create/Modify Menu Screen(2.7)

4 Other Analysis Elements

In this part of the report, dimension and possible risk that may be encountered will be discussed.

4.1 Consideration of Various Factors

There are some factors that needs to be considered may affect our project. Below, brief discussion about these factors can be seen.

Public Health

We think that Qrder has no positive or negative effect on public health. For clients, these app will be used only when they are hungry and when they want to eat out instead of their home. So, this app will not require high amount of time to be used accurately. Therefore, this app will not affect any client. For restaurant, nearly whole of them use a system that tracks orders, therefore our app will not create any confusion for workers. Furthermore, since it is simple it will make the work easier for them. So, it can be said that people who use Qrder will not face any positive or negative effect since it is an application that can be used only when needed. In other words, unlike social media, it will not have an effect on daily life. People will not be obsessed with it. So, in addition tho physical health psychological health will not be affected.

Public Safety

When in comes to public safety, the most important thing that needs to be considered is online payment. Qrder provides online payment system, but it will not store the credit card information of the users on the database. Furthermore, online payment service will be done by 3rd party companies since there are regulations and it is safer for the clients. Another thing is users and restaurants will have profiles, therefore Qrder will keep them safe. To keep them safe, whole information will be hashed and will not be published to any company or person.

Public Welfare

Qrder will be free to download and free to use. In addition, there will not be paid services. So any person that has a mobile phone that is connected to internet or any restaurant that has a desktop can use Qrder freely. Then, it can be said that Qrder will not discriminate any group or company with respect to their budget. In other words, anyone wants to use Qrder can use it. Therefore, Qrder has nothing to do with the context of public welfare.

Global Factors

There are world wide regulations that needs to be followed by any web based application. Qrder will strictly follow these regulations. For example, it will not do any operation on user's personal information as GDPR stated. [9]. In addition to that, Qrder will also strictly follow the security protocols for any network operation especially for online payment. To sum up, Qrder will not cross any boundary that is drawn by regulations that every country has an agreement on it.

Cultural Factors

Since each culture has distinct dishes, we know that we may encounter different menu, interface and restaurant design . That's why, to make Qrder be applicable for whole cultures, instead of using standard menu and restaurant layout design, we will let restaurants to create and change their designs. By doing that, we purpose that any restaurant owner that wants to use Qrder can use it without doubting about losing their local features. Then, a restaurant owners who work in touristic places can introduce their food culture to tourists that use Qrder in a digital way. Furthermore, menu will support different language models, so

Qrder will minimize the risk of communication problems . On other hand it will support to show local features of the restaurants.

Social Factors

We cannot find any social factor that is related to Qrder. Since any person can use Qrder, Qrder has nothing to do with terms like age , gender , race etc. That’s why, we can say that social factors are not applicable for Qrder.

4.2 Risks and Alternatives

Before implementing Qrder, we considered some possible risks. After that, we thought about effects of risks on Qrder then we created B plans with respect to that. These B plans will be executed in case of risk that will be explained below occurs.

1. Unsatisfactory Server Performance

Since servers will be busy when the restaurants are crowded, the servers must remain functional. Therefore , if we realize that currents systems are not good enough to provide a full support for clients, we will just find a new ones that is better equipped.

2. Infeasible Server Cost

Better quipped server is a good choice obviously, but it must not exceed the economic threshold that will be determined. Then, if price of the server is too high, then making some operations on local computer’s instead of server can be considered.

3. One of Team Members Leaving

Since this project is handled by 3 people, there is a high risk that if one of them leaves. If it happens, to make loss tolerable, we purpose that each member has a knowledge about any part of project. We have an equal workload among participant and in case of leaving, others can share the leaving one’s workload equally. Since they will know what he has been working for , they can easily adapt to new and extra responsibility. So , the loss will be tolerable.

4.3 Project Plan

WP #	Work package title	Leader	Members involved
WP 1	Project Specification	Akif	Oğuz, Taylan
WP 2	Analysis Report	Oğuz	Akif, Taylan
WP 3	High-Level Design Report	Taylan	Akif, Oğuz
WP 4	Low-Level Design Report	Akif	Oğuz, Taylan
WP 5	Final Report	Oğuz	Akif, Taylan
WP 6	Presentations & Demonstrations	Oğuz	Akif, Taylan
WP 7	Web Site	Taylan	Akif, Oğuz
WP 8	Client Side	Taylan	Akif, Oğuz
WP 9	Business Side	Oğuz	Akif, Taylan
WP 10	Server Side	Akif	Oğuz, Taylan
WP 11	Database	Akif	Oğuz, Taylan
WP 12	Deployment	Oğuz	Akif, Taylan
WP 13	Web Service Management	Taylan	Akif, Oğuz
WP 14	Design	Akif	Oğuz, Taylan
WP 15	Project Management	Taylan	Akif, Oğuz

WP 1: *Project Specifications*			
Start date: *17 February 2020* **End date:** *24 February 2020*			
Leader:	*Akif*	**Members involved:**	Oğuz, Taylan
Objectives: *The initial requirements are defined for the project.*			
Tasks: ***Task 1.1 Introduction*** ***Task 1.2 Description*** ***Task 1.3 Constraints*** ***Task 1.4 Professional and Ethical Issues*** ***Task 1.5 Requirements***			
Deliverables ***D1.1:*** *Project Specifications*			
WP 2: Analysis Report			

Start date: *16 March 2020* **End date:** *23 March 2020*			
Leader:	*Oğuz*	**Members involved:**	Akif, Taylan
Objectives: *The analysis report contains a detailed analysis of the problem. It should address all relevant issues.*			
Tasks:			
Task 2.1. Introduction			
Task 2.2. Current System (if any)			
Task 2.3. Proposed System			
Task 2.3.1 Overview			
Task 2.3.2 Functional Requirements			
Task 2.3.3 Nonfunctional Requirements			
Task 2.3.4 Pseudo Requirements			
Task 2.3.5 System Models			
Task 2.3.5.1 Scenarios			
Task 2.3.5.2 Use Case Model			
Task 2.3.5.3 Object and Class Model			
Task 2.3.5.4 Dynamic Models			
Task 2.3.5.5 User Interface - Navigational Paths and Screen Mock-ups			
Task 2.4. Other Analysis Elements			
Task 2.4.1. Consideration of Various Factors			
Task 2.4.2. Risks and Alternatives			
Task 2.4.3. Project Plan			
Task 2.4.4. Ensuring Proper Team-work			
Task 2.4.5. Ethics and Professional Responsibilities			
Task 2.4.6. New Knowledge and Learning Strategies			
Task 2.5. Glossary			
Deliverables			
D2.1: *Analysis Report*			
WP 3: *High-Level Design Report*			
Start date: *1 May 2020* **End date:** *15 May 2020*			
Leader:	*Taylan*	**Members involved:**	*Akif, Oğuz*
Objectives: *High-level or system design is the transportation of the analysis model into a system design model.*			
Tasks:			
Task 3.1. Introduction			
Task 3.1.1 Purpose of the system			
Task 3.1.2 Design goals			
Task 3.1.3 Definitions, acronyms, and abbreviations			
Task 3.1.4 Overview			
Task 3.2. Current software architecture (if any)			
Task 3.3. Proposed software architecture			
Task 3.3.1 Overview			
Task 3.3.2 Subsystem decomposition			
Task 3.3.3 Hardware/software mapping			
Task 3.3.4 Persistent data management			
Task 3.3.5 Access control and security			
Task 3.3.6 Global software control			
Task 3.3.7 Boundary conditions			
Task 3.4. Subsystem services			
Task 3.5. New Knowledge Acquired and Learning Strategies Used			
Task 3.6. Glossary			

Deliverables ***D3.1:*** *High-Level Design Report*			
WP 4: *Low-Level Design Report*			
Start date: *25 September 2020* **End date:** *5 October 2020*			
Leader:	*Akif*	**Members involved:**	*Oğuz, Taylan*
Objectives: *Extent and validity of the design principles that were used to carry out this phase of the project must be explained in detail. Also, creativity, that is the extent to which the team developed a novel solution to the design problem while still achieving a functional design, at the low-level design phase, must be clear.*			
Tasks: ***Task 4.1. Introduction*** ***Task 4.1.1 Object design trade-offs*** ***Task 4.1.2 Interface documentation guidelines*** ***Task 4.1.3 Engineering standards (e.g., UML and IEEE)*** ***Task 4.1.4 Definitions, acronyms, and abbreviations*** ***Task 4.2. Packages*** ***Task 4.3. Class Interfaces*** ***Task 4.4. Glossary***			
Deliverables ***D4.1:*** *Low-Level Design Report*			
WP 5: *Final Report*			
Start date: *5 December* **End date:** *17 December 2020*			
Leader:	*Oğuz*	**Members involved:**	*Akif, Taylan*
Objectives: *The final report is the culmination of the project. The final architecture and design of the system as well as the final status of the project is presented in this report.*			
Tasks: ***Task 5.1. Introduction*** ***Task 5.2. Requirements Details*** ***Task 5.4. Final Architecture and Design Details*** ***Task 5.5. Development/Implementation Details*** ***Task 5.6. Testing Details*** ***Task 5.7. Maintenance Plan and Details*** ***Task 5.8. Other Project Elements*** ***Task 5.8.1.Consideration of Various Factors*** ***Task 5.8.2.Ethics and Professional Responsibilities*** ***Task 5.8.3.Judgements and Impacts to Various Contexts*** ***Task 5.8.4 Teamwork and Peer Contribution*** ***Task 5.8.5 Project Plan Observed and Objectives Met*** ***Task 5.8.6 New Knowledge Acquired and Learning Strategies Used*** ***Task 5.9. Conclusion and Future Work*** ***Task 5.10. Glossary***			
Deliverables ***D5.1:*** *Final Report*			
WP 6: *Presentations & Demonstrations*			
Start date: *5 December 2020* **End date:** *21 December 2020*			
Leader:	*Oğuz*	**Members involved:**	*Akif, Taylan*
Objectives: *Presentation of the project.*			
Tasks: ***Task 6.1 Presentation*** ***Task 6.2 Demo***			

Deliverables ***D6.1:*** *Presentations & Demonstrations*			
WP 7: *Website*			
Start date: *20 February 2020* **End date:** *1 December 2020*			
Leader:	*Taylan*	**Members involved:**	*Akif, Oğuz*
Objectives: *The website of the project.*			
Tasks: ***Task 7.1 Design*** ***Task 7.2 Implementation***			
Deliverables ***D7.1:*** *Website*			
WP 8: *Client Side*			
Start date: *15 May 2020* **End date:** *1 December 2020*			
Leader:	*Taylan*	**Members involved:**	*Akif, Oğuz*
Objectives: *A mobile app that provides functionality and accessibility to customers.*			
Tasks: ***Task 8.1 Storage*** **:** *Connection between local storage of the device and the app. Files will be encrypted while saving and will be decrypted while loading for data security.* ***Task 8.2 API Connection*****:** *Connection between the app and server via REST API.* ***Task 8.3 Model*****:** *Data structures for the objects received from the server.* ***Task 8.4 View*** **:** *User Interface part of the app.* ***Task 8.5 Controller*****:** *Controller part of the app. This part is contained inside of the View part.* ***Task 8.6 Test*** **:** *Code coverage and testing.*			
Deliverables ***D8.1:*** *Qrder App (Android)*			
WP 9: *Business Side*			
Start date: *20 February 2020* **End date:** *1 December 2020*			
Leader:	*Oğuz*	**Members involved:**	*Akif, Taylan*
Objectives: *<briefly explain the objectives of this work package (3-5 sentences) >*			
Tasks: ***Task 9.1 Storage*** **:** *Connection between local storage of the device and the app. Files will be encrypted while saving and will be decrypted while loading for data security.* ***Task 9.2 API Connection*****:** *Connection between the app and server via REST API.* ***Task 9.3 Model*****:** *Data structures for the objects received from the server.* ***Task 9.4 View*** **:** *User Interface part of the app.* ***Task 9.5 Controller*****:** *Controller part of the app. This part is contained inside of the View part.* ***Task 9.6 Test*** **:** *Code coverage and testing.*			
Deliverables ***D9.1:*** *Qrder Business App (Desktop)*			
WP 10: *Server Side*			
Start date: *20 February 2020* **End date:** *1 December 2020*			
Leader:	*Akif*	**Members involved:**	*Oğuz, Taylan*
Objectives: *Server side of the project and provides database connection, authentication and service controls.*			
Tasks: ***Task 10.1 REST API Handler*** **:** *Handles requests and responses via REST API. Interpreters and middlewares provide request security and response structure.* ***Task 10.2 Services*** **:** *Subrequest handler and controller for database.*			

***Task 10.3 Database Controller* :** *Managing database/s connection and queries.* ***Task 10.4 Model* :** *Data structuires for the entities in the database.* ***Task 10.5 Authentication* :** *Request authentication and session handler via JWT* ***Task 10.6 Test* :** *Code coverage and testing.*			
Deliverables ***D10.1:*** *Web service (NodeJS)*			
WP 11: *Database*			
Start date: *20 February 2020* **End date:** *1 November 2020*			
Leader:	*Akif*	**Members involved:**	*Oğuz, Taylan*
Objectives: *Database/s of the project.*			
Tasks: ***Task 11.1 Decision* :** *Choosing type of the SQL database and decision about whether NO-SQL database would be used.* ***Task 11.2 Diagram* :** *Creating structures and schemas for the database.* ***Task 11.3 Setup & Build* :** *Setup and building database. Creating scripts for server side database connection via dummy data.*			
Deliverables ***D11.1:*** *Database* ***D11.2:*** *Scripts for Server side*			
WP 12: *Deployment*			
Start date: *1 May 2020* **End date:** *1 December 2020*			
Leader:	*Oğuz*	**Members involved:**	*Akif, Taylan*
Objectives: *Deploy services to the web.*			
Tasks: ***Task 12.1 Decision* :** *Choosing deployment service such as AWS or Heroku.* ***Task 12.2 Setup* :** *Setup deployment settings, CLI, pipeline tools, etc. Automating deployment, if it is possible and free.* ***Task 12.2 Maintenance***			
Deliverables ***D12.1:*** *Deployment* ***D12.2:*** *Automated Deployment (Optional)*			
WP 13: *Web Service Management*			
Start date: *1 May 2020* **End date:** *20 December 2020*			
Leader:	*Taylan*	**Members involved:**	*Akif, Oğuz*
Objectives: *Managing deployed web service.*			
Tasks: ***Task 13.1 Maintenance***			
Deliverables ***D13.1:*** *Web Service Management*			
WP 14: *Design*			
Start date: *24 February 2020* **End date:** *1 November 2020*			
Leader:	*Akif*	**Members involved:**	*Oğuz, Taylan*
Objectives: *Design of the website, mobile app and desktop app. Color palette of the projects.*			
Tasks: ***Task 14.1 Decision* :** *Choosing a color palette and coming up with logo ideas* ***Task 14.2 Design* :** *Designing logo, website, mobile and desktop app, finding icons, assets, etc.*			
Deliverables ***D14.1:*** *Color Palette*			

D14.2: *Logo* ***D14.3:*** *Website Design* ***D14.4:*** *Mobile App Design* ***D14.5:*** *Desktop App Design*			
WP 15: *Project Management*			
Start date: *17 February 2020* **End date:** *20 December 2020*			
Leader:	*Taylan*	**Members involved:**	*Akif, Oğuz*
Objectives: *Management of tools, meetings, etc.*			
Tasks: ***Task 15.1 Sprint*** ***Task 15.2 Tools Management*** ***Task 15.3 Meetings*** ***Task 15.4 Teamwork*** ***Task 15.5 Learning*** ***Task 15.6 Enjoy***			
Deliverables ***D15.1:*** Qrder			

4.4 Ensuring Proper Team-Work

In order to have successful and reliable systems, teamwork is one of the significant parts of the job. In our project, we have chosen different leaders for the separated parts that are Client side, Business side and Server side, according to knowledge of members. However, that does not mean, members will work in seperate parts. Every member must collaborate in all 3 parts and one must be the leader. The reason for applying shared leadership was increasing efficiency of team-work and to be able to simulate a good leadership experience. A leader must follow projects related news and technologies, analyze members, control teamwork and make decisions in favor of the team and the project. Therefore, having separated leadership will provide all of us to understand each other's skills and performance. The analyzing members and teamwork will done as in the following:

- Git commits of all members will be followed by the leader of the part. Therefore the leader would be able to analyze the skill and the performance of collaborators and be able to support them properly, if needed.

- Google Sheets will be used to represent sprint points of members and the process of the parts separately. Leaders will also follow their parts from there too. Therefore, performance would be graphed.
- Kanban style boards will be used in parts to create sprints and link them with the code source.
- GitHub issues will be tracked and solved by team members to have more involved teamwork.

We do not expect to have the same number of commits from every member because quality is more important than quantity. However, the average of commit numbers of each member per week must be consistent with the whole sprint. Slack will be used as the main communication tool for the project. After every sprint, performance analyzes will be done by members to increase efficiency in Slack or Google Hangouts. Therefore, we expect good teamwork in the project.

4.5 Ethics and Professional Responsibilities

First of all, global impact of Qrder will be discussed. Qrder will be free to download and free to use. It's purpose is making ordering and payment digital. Then, anyone can download and use it. This application will not discriminate against a certain group of people, country or any social class. Then , we expect the global impact of the application to make things faster and easier for restaurants and clients anywhere , anytime.

About economic impact, there is no such dominating system that nearly gets the whole market share in the food industry. Also we could not find any value that represents total market value in the digital food industry. But, since our project is small for now, we will not claim any market share. Instead, if it will be used , we may put ads to earn some money for our servers. Besides from server cost , Google Play Store and App Store , we do not think that any constraint that costs us money. However, three constraints above can be affordable for us.

Another impact is Environmental. Qrder will have servers and these servers will consume energy. It will have a negative impact on environment. But on the other hand, restaurants may try to introduce their menu digitally. By doing that they will use less paper.T t has a positive effect on environment. To sum up, Qrder will both positive and negative effect on environment and we will try to minimize negative effects while on the other hand we will try to maximize positive events.

Last issue that we thought of is societal impact and data privacy. Qrder will not publish any private information with 3rd parties as GDPR stated.

4.6 New Knowledge and Learning Strategies

In order to come up with a successful system and the project, we need to know technologies, languages and concepts in the following:

- JavaScript
- React Native
- NodeJS
- Electron / a desktop framework
- Computer Network
- Android Development
- Microservices
- Software Project Management
- Customer Relationship Management (CRM)

All members of the team have some knowledge about skills in the above and different interestings. Therefore, in order to have good communication and efficiency, members must learn all skills at some level. Computer Network concept could be learnt by Bilkent’s course (CS421) and Software Project Management could also be learnt by Bilkent’s course (CS413). Rest of the technologies, languages and concepts will be learned by applying the following methods:

- Online Learning (Udemy, Coursera, edX, etc)
- Literature Review (Recommended books in Bilkent’s courses related with the topic, official documents of technologies)
- Trial and Error (Well written test script could help to catch errors)

“Online Learning” will be the main method of learning these technologies because web services like Udemy have lots of good and concise courses about technology. “Literature Review” and “Trial and Error” will always be followed because both are quick and encountered everyday solutions.

5 Glossary

UI - User Interface

DAO - Direct Access Object

UX - User Experience

GDPR - General Data Protection Regulation

JS- JavaScript

SQL - Structured Query Language

JWT - JSON Web Token

REST API - Representational state transfer API

AWS - Amazon Web Services

CRM - Customer Relationship Management

6 References

[1]Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2]Baeldung. “The DAO Pattern in Java.” *Baeldung*, 21 Mar. 2020, www.baeldung.com/java-dao-pattern.

[3]“What Is Non-Functional Requirement? Types and Examples.” *Guru99*, www.guru99.com/non-functional-requirement-type-example.html.

[4]“Passport.js.” *Passport.js*, www.passportjs.org/.

[5]“GraphQL: A Query Language for APIs.” *A Query Language for Your API*, graphql.org/.

[6]“Node.js Web Application Framework.” *Express*, expressjs.com/.

[7]Axios. “Axios/Axios.” *GitHub*, 7 Mar. 2020, github.com/axios/axios.

[8]Kapetanakis, Matos. “GDPR - Changing the Rules of Identity and Access Management.” *ITProPortal*, ITProPortal, 2 Feb. 2018, www.itproportal.com/features/gdpr-changing-the-rules-of-identity-and-access-managem ent/.