Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: QRDER

High Level Design Report

Mustafa Oğuz Güngör

Taylan Bartu Yoran

Mehmet Akif Kılıç

Supervisor: Varol Akman

Jury Members: Uğur Güdükbay, Fazlı Can

High Level Design Report
May 22, 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

Introduction 4

Purpose of the System 5

Design Goals 6

Extensibility 6

Reliability 6

Usability 6

Accessibility 6

Portability 7

Efficiency 7

Definitions,Acronyms and Abbreviations 8

Overview 8

Current System Architecture 9

Proposed Software Architecture 10

Overview 10

Subsystem Decomposition 10

Hardware/Software Mapping 11

Persistent Data Management 12

Access Control and Security 13

Global Software Control 14

Boundary Conditions 16

Subsystem Services 16

Client 16

View Tier 17

Controller Tier 17

Model Tier 18

Administrative User 19

View Tier 20

Controller Tier 20

Model Tier 21

Server 22

Route Tier 23

Logic Tier 23

Data Tier 24

New Knowledge Acquired and Learning Strategies Used 24

References 24

1 Introduction

Comparing last decades, more people prefer eating outside rather than at home. Since there may be many reasons [2] , 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 3 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.

1.1 Purpose of the System

Our system is not only for restaurant owners or customers. Since we want to make it can be used for different aims and can be used by different types of users , our system has several purposes.

First of all, our system tries to make everything faster for customers. Since ordering and payment can be a very time consuming events in a crowded restaurant. Our app aims to make things faster so that customers will do not have to waste their times.

Secondly, our system tries to provide a safe environment for restaurant owners. Since order will be delivered through app, problems that are related to human factors will be minimized. Furthermore, since customers can pay the check through app, our app aims to make payment totally secure.

To sum up, our system aims to be powerful , fast , safe and interactive system.

1.2 Design Goals

In this section, Qrder's design goals will be presented.

1.2.1 Extensibility

The system should:

- be easy to maintain
- be available on multiple platforms

1.2.2 Reliability

The system should:

- do not store any credit card information unless user states 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, restaurant or clients

1.2.3 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

1.2.4 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.

1.2.5 Portability

The system should:

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

1.2.6 Efficiency

The system should:

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

1.3 Definitions,Acronyms and Abbreviations

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

1.4 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.

2 Current System Architecture

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.

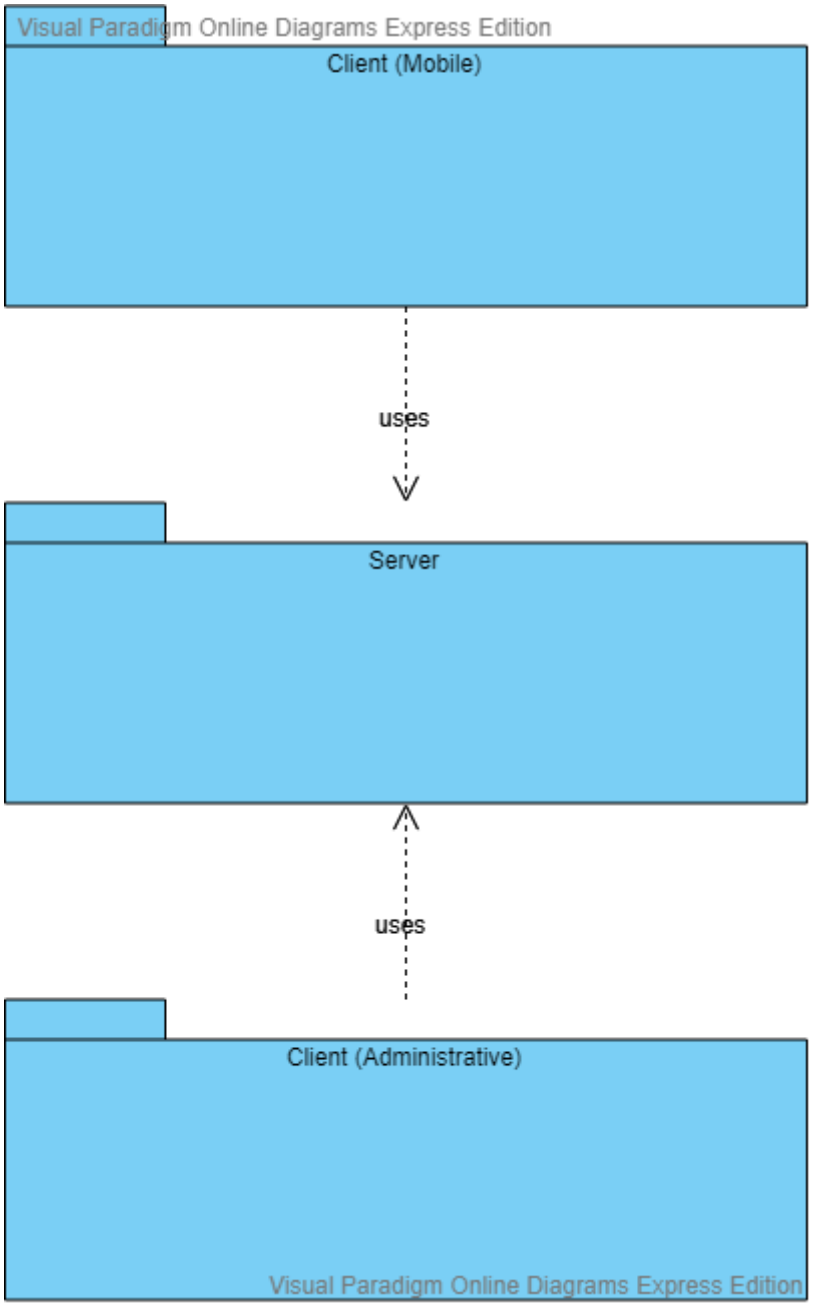
3 Proposed Software Architecture

3.1 Overview

Interactions and authorizations of endpoints in the QRDER system need to be defined for more efficient/secure system architecture. Therefore, decomposition of each subsystem should be analysed,their access control and architecture should be designed in this manner. With this approach: decompositions, protocols will be used in hardware/software levels, Data control and dependencies, Authorization and access

control and global software control subtopic will be analysed and designs for these sub topics will be done for QRDER system.

3.2 Subsystem Decomposition



QRDER has a client-server subsystem decomposition as in the figure below. Both mobile and administrative client subsystems are independent from each other, and only dependent on the server subsystem.

Client subsystems are similar to each other, due to the similarity between the aim of them. In both client sides, MVC (Model, View, Controller) design pattern is followed. View of the client tier represents the UI part of the client apps that are mobile and desktop. Controller of the client tier represents the management and business logic of the apps. Model of the client tier represents the data models that are custom and DAO. The communication from clients to the server tier is provided via a JavaScript library called Axios that handles http responses and requests.

Server subsystem is the core of the project QRDER because every business logic of the project is provided by it. Server is an instance of stateless server in order to increase efficiency, reliability and stability because stateless servers do not handle session or state specific data. Therefore, the outcome of these servers are only dependent on business logic, not any state. Server tier provides apps management of the user, restaurant, menu, item and order though by routes. Database is also handled by server tier via an ORM (Object-relational mapping) tool. Server subsystem has 3 main tiers that are Route, Logic and Data. Route tier handles routes and communication between server and the apps. Logic tier handless all business logic of the project QRDER. Authentication, token and other provided magements of the data is the part of the Logic tier. Data tier is the bridge between Database and the apps via models that are provided by an ORM tool.

3.3 Hardware/Software Mapping

There are two types of users in the proposed system of QRDER, which are administrative users and customers. Administrative users will be using the system via desktop application, where customers will be using the system via smartphones. Two of these desired systems need small sizes of memory for caching techniques of the libraries and APIs for connection and depiction properties. Internet connection and its link are also other requirements to use the system. Smartphone applications for customers will also require a camera to scan QR Stickers. If a recommendation system is wanted to be used, then smartphones with GPS receivers will also be needed to get location information of the user.AWS Cloud is planning to be used to design and maintain the server side of the system[3].

All of the Components in the system will be implemented with JavaScript language, where different frameworks will be included for each component. Reasons to choose javaScript are:

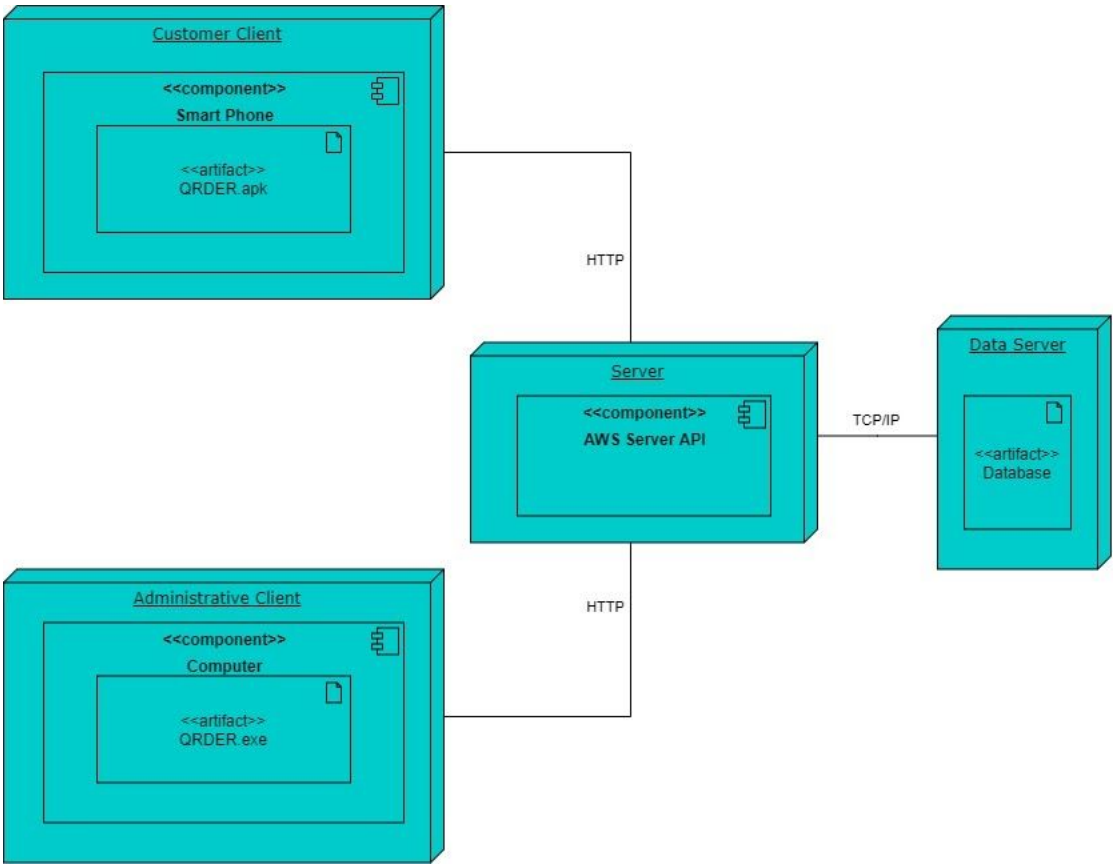
- It is used in React framework for mobile application development,
- It is used in the Electron framework for desktop application development.
- Easy to use with relational databases.
- qrcode.react package for mobile application with QR Code Scanner.
- Can be used to implement HTTP connection and REST API.

Node.js runtime will be used to implement backend processes ,database integration and usage for the server-side of the system[4]. As mentioned before, AWS Cloud will be used to implement a backend server and will run on a linux operating system,due to usability and efficiency in terms of speed of computations.

React framework will be used to implement customer application of client-side, connection with REST API and QR Code decryption[5]. Easy-to-use render libraries, qrcode library and integrability are the main reasons for the react framework to be chosen. Electron framework will be used in administrative application of client-side due to the same reasons, but it will be used to implement desktop application of the client-side[6].

Relational databases will be used to achieve more reliable persistent data management and to ease the maintenance of the system. REST API,Triggers and queries will be used to provide information for different components and to set up communication[7].

It is required to transfer data for notifications,verifications and information towards different components of the system. Priority for this transfer is reliability, efficiency and maintainability. Therefore, HTTP and REST API can provide such aspects for the network between components, therefore they will be used to set up communications other than database related data.



3.4 Persistent Data Management

Database interactions and queries are the main underlying structure of the system. There is no specific data that resides in memory of endpoints, only cache memory exists in systems to acquire frequent requests faster. Therefore, persistent data consist of relational database properties and is reached with queries. Triggers and updates will be used in specific cases, but generally authorization of the client side will be read-only. In addition to these, there will be REST API caching in endpoints.

3.5 Access Control and Security

QRDER system consists of different types of users with different authentications. There are 3 types of users in the system which are :

- Administrative Client: Restaurant's order management application.
- Customer Client: Customer's smartphone app.
- Server Administrator/Developer: Server-side developers.

Clients will use HTTP to invoke any action in the system, and then verified action will update the database. Clients will only access the database to get updated data/information and monitor it. Server will use invocations and messages of clients to update the database, so the server will be able to read and change the database.Therefore, database access of the clients will be read-only access and server access will be read and write.

Access of the customer client should be approved/verified by the administrative client. Without approval/verification of the administrative client, customer client cant establish the connection between the server and itself. Such access control is to prevent abusive and unintended QR logins.

Access to update the database will be restricted for clients to ensure access control and provide security for updating queries. However, communication between server and clients will be done via HTTP and REST API which should be adjusted to a secure form to make the system less vulnerable. Therefore, key encryption will be used for two sides to encrypt information and to prevent unexpected messages to the server. Symmetric encryption will be used for this mechanism.

Data privacy is another concern for the security aspect of the system. User information and order history should be kept in the database to maintain such a system,

which may cause violation or doubt of violation of data privacy for clients. Therefore, permission to contain such data will be asked from users in terms of use agreement. This agreement will be prepared by considering GDPR[8].

3.6 Global Software Control

As it is mentioned before, abstraction of a system consists of 3 main parts which are customer (Customer Client), restaurant(Administrative Client) and server. Control of interactions between these 3 systems consists of 2 main phases which are monitoring phase and connection phase. These interactions include communication between 3 parts and queries from the database.

1. **Monitoring Phase**

Clients(customer & administrative) are in a steady phase when there is no request that is sent and no response that is expected in endpoints. This steady phase is the monitoring phase in which clients can monitor information according to the last update that is done in the database. There is no interaction between endpoints untilla request invocation is done. Basic read-only queries are done between the database and clients in this phase.

2. **Communication Phase**

When a customer invokes a request, the communication phase starts. In this phase, a new request for adding/updating the database is sent from the client and the server verifies this request before updating the database. Once verification/approval is done, server updates database with its write-access to database and sends response to related clients with success code.

   a. **Request Phase**

Clients can send requests to the server via user interface to give or modify orders. This phase can be done by both 2 clients where their header and data information is different in the request message. Reason of these difference is that request of administrative client does not need an approval and skips approval phase in server where request of customer client need an approval before processing/updating phase. Therefore,the request message includes the type of the user, encrypted request text and encryption key.

b. Approval/Verification Phase

If a customer client sends a request in the request phase, the server sends a request message to verify order or client to relative administrative client. This process is done to avoid fake order and abusive use of the system. If administrative client verify order in its response, then server can maintain for processing/updating phase; otherwise, administrative client sends a message with unverified code to server and order and request is removed from server cache, then customer client is notified as unverified order.

c. Processing/updating Phase

Once the server gets approval from the administrative client, start the processing phase for decrypting messages that came from the customer client. After ending decrypting and parsing the message, the server manages demanded updates in the database.

d. Response Phase

If an error occurs in the processing/updating phase, the server notifies clients with a response that has corresponding error code and aborts the process;

otherwise, it notifies clients with a response that has success code and the demanded information in the message body.

3.7 Boundary Conditions

There are 4 main boundary conditions for the given system, which are initialization, termination, restriction and failure. These conditions explain the system work space and requirements, which are described in detail below.

Initialization

Clients should initialize the programs in their machines. For the initialization process, programs should be downloaded to the machine. Set up after opening the program requires connection which provides communication between server to maintain services of the program. Local memory is only being used for caches, which also requires a negligible amount of disk space in the machine.

Termination

After connection set up is done, programs should stand on to maintain connection between end points. If programs are terminated, then communication closed and initialization of connection should be done again to use the program. Disconnection does not directly terminates communication, gives timeout instead. However, long disconnection may occur in termination of the program as well.

Restriction

If use of program for a customer is restricted by server due to unauthorized login, abusive use or verification problems, then program will not manage ordering meal and prohibit user to use its services.

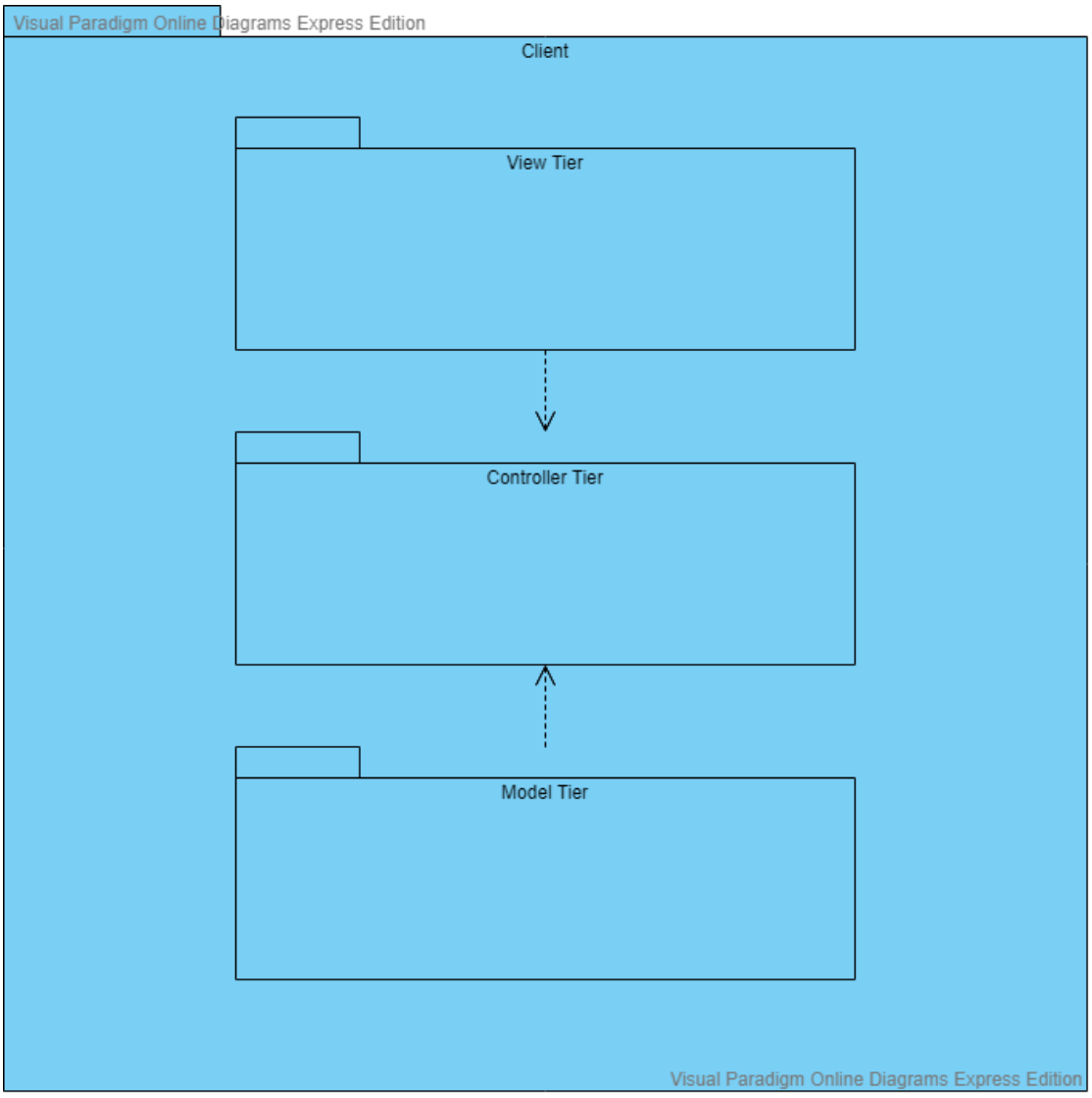
Failure

Unexpected errors in server side or communication network may cause failures ofr the system. These failures will disable services of the system and block the use of the system. There are 3 probable failures for the system:

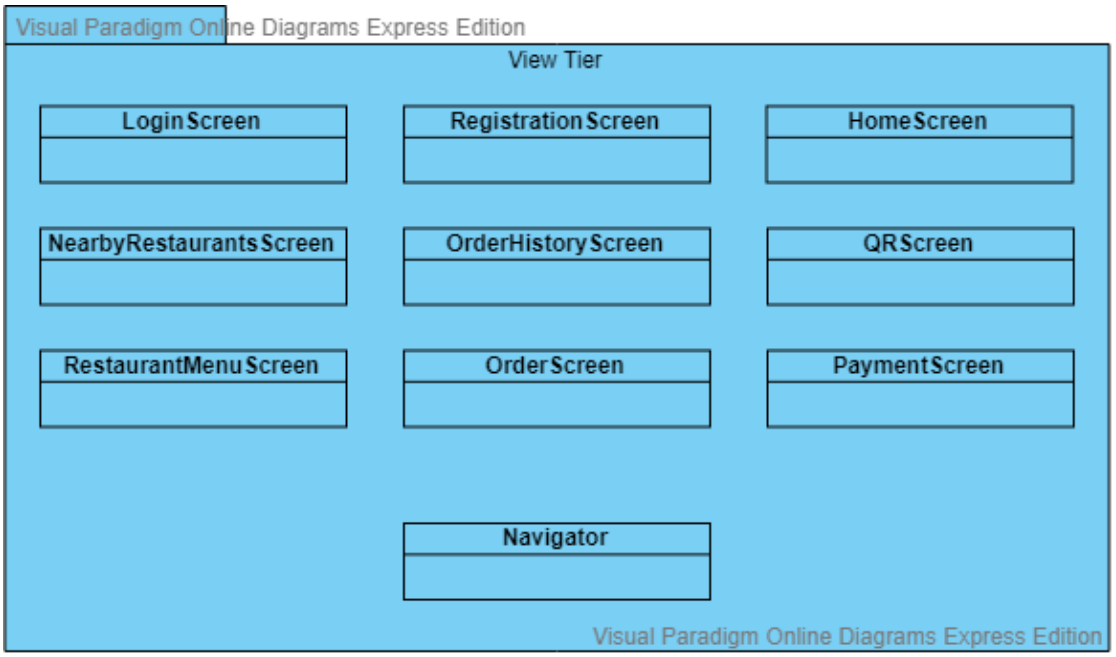
- **Disconnection:** If users remains disconnected from server for a long time, then connection setup,cookies and cache variables from both sides will be removed and re-initialization will be needed.
- **Server overload:** If server encounters an intense connection traffic, then scheduling or server processing may become bottleneck for the connection which means unavailable services for a time period for users.
- **Probable Bug occurrences:** Mistakes in development phase that are not determined in testing period may cause bugs in the run-time of the program and can block the use of the services of the system.

4 Subsystem Services

4.1 Client



4.1.1 View Tier

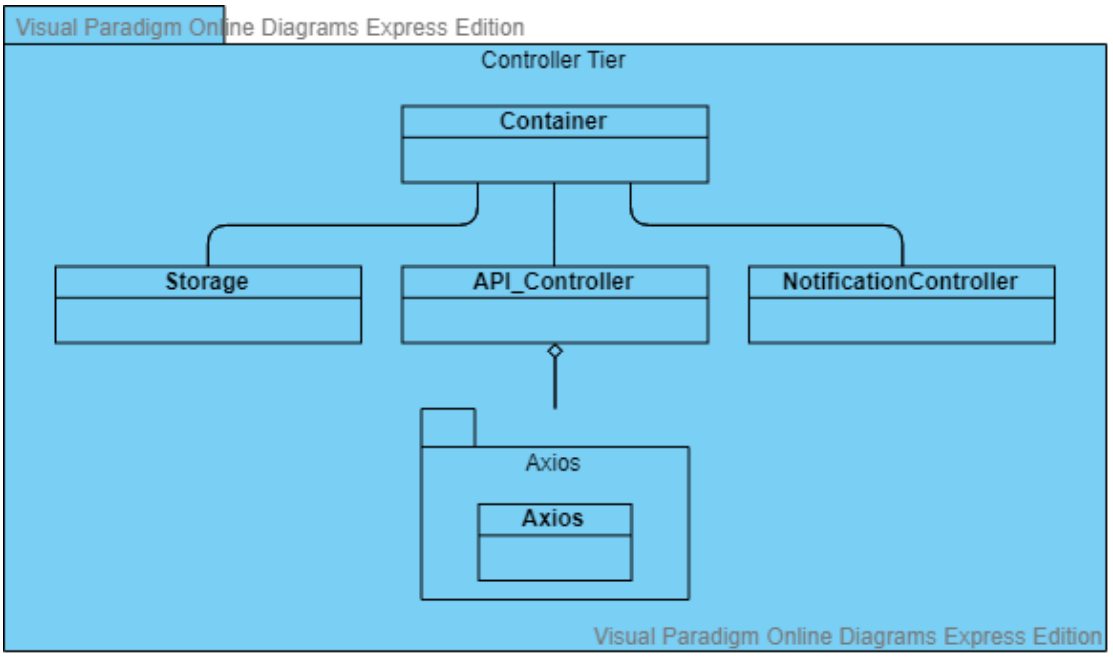


View tier is the subsystem for UI components.

Class	**Description**
Navigator	This class provides a navigation system to the app. It handles switches between screens.
LoginScreen	This class provides the login components to the user.
RegistrationScreen	This class provides the registration components to the user.
HomeScreen	This class provides special offers to the user and also links to the other pages.

NearbyRestaurantsScreen	This class provides users the restaurants that are in the near, according to the location of users.
OrderHistoryScreen	This class provides users their own order history.
QRScreen	This class provides users to scan QR codes that are provided by restaurants.
RestaurantMenuScreen	This class provides users the menu of the selected restaurant via scanner QR code.
OrderScreen	This class provides users management of their orders.
PaymentScreen	This class provides users to pay the order.

4.1.2 Controller Tier

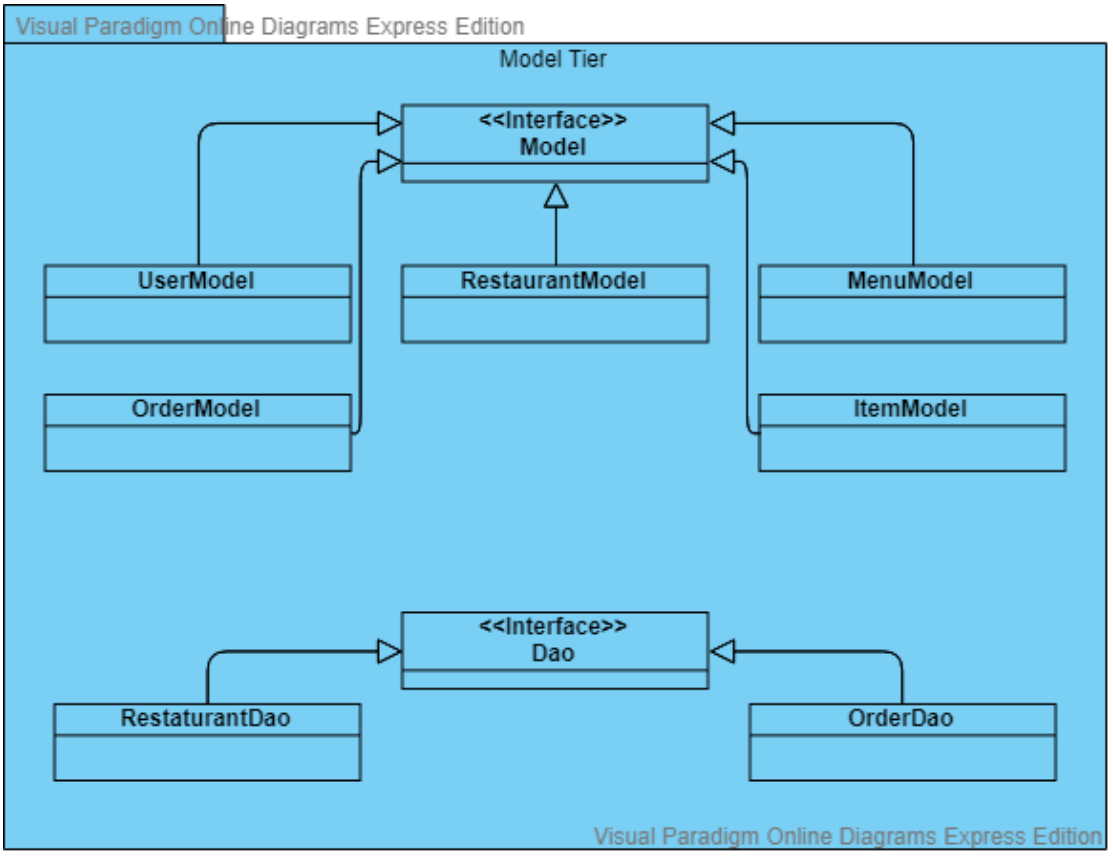


Controller tier is the subsystem in order to handle events, requests and responses.

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.
API_Controller	This class provides sending http requests -get, post, put, delete- to the server via 25 instance of Axios class.

NotificationController	This is a singleton class that handles notifications of the app
Axios	This class belongs to third-party package Axios and is able to send http requests to a specified url and a port [9].

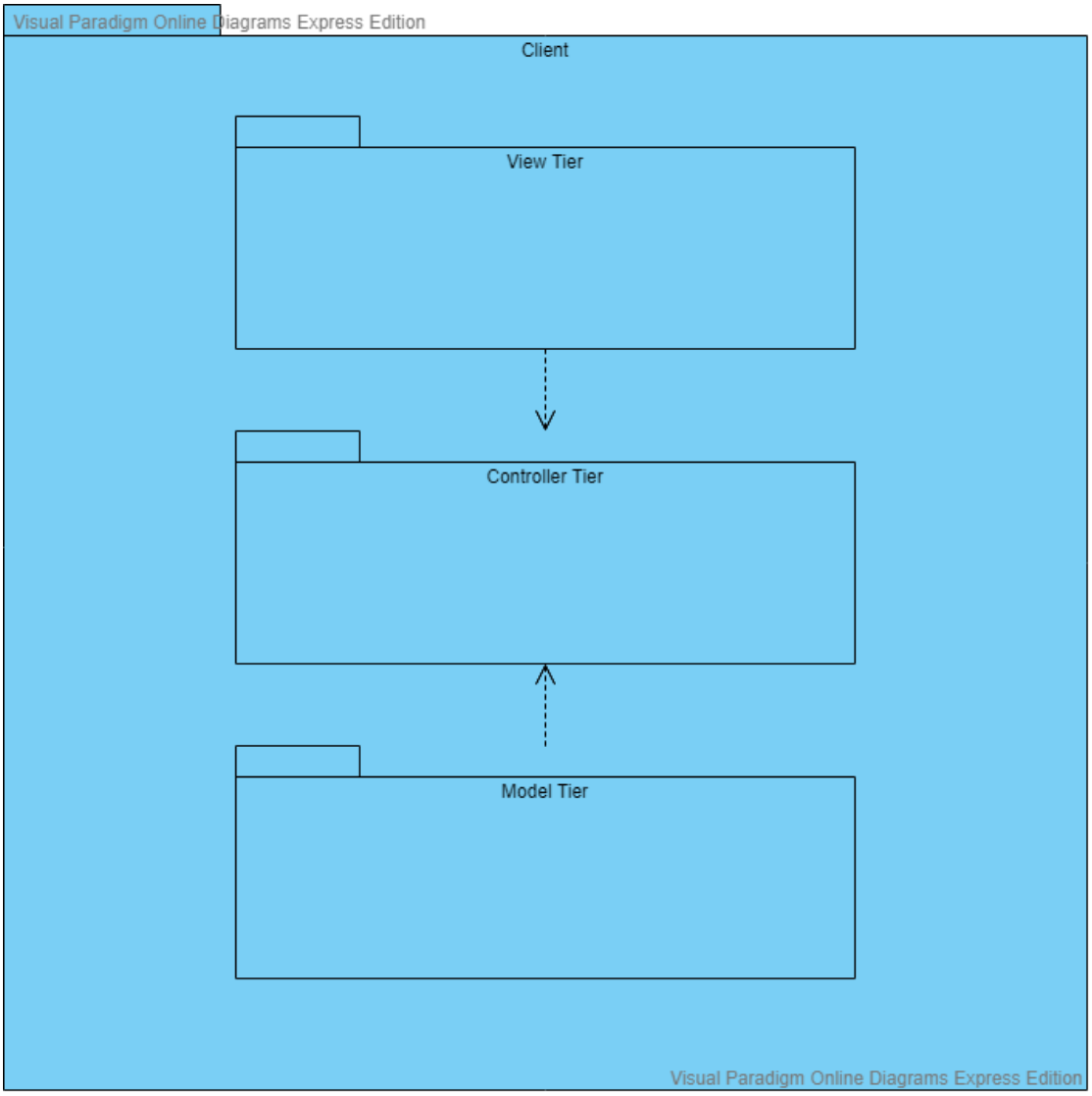
4.1.3 Model Tier



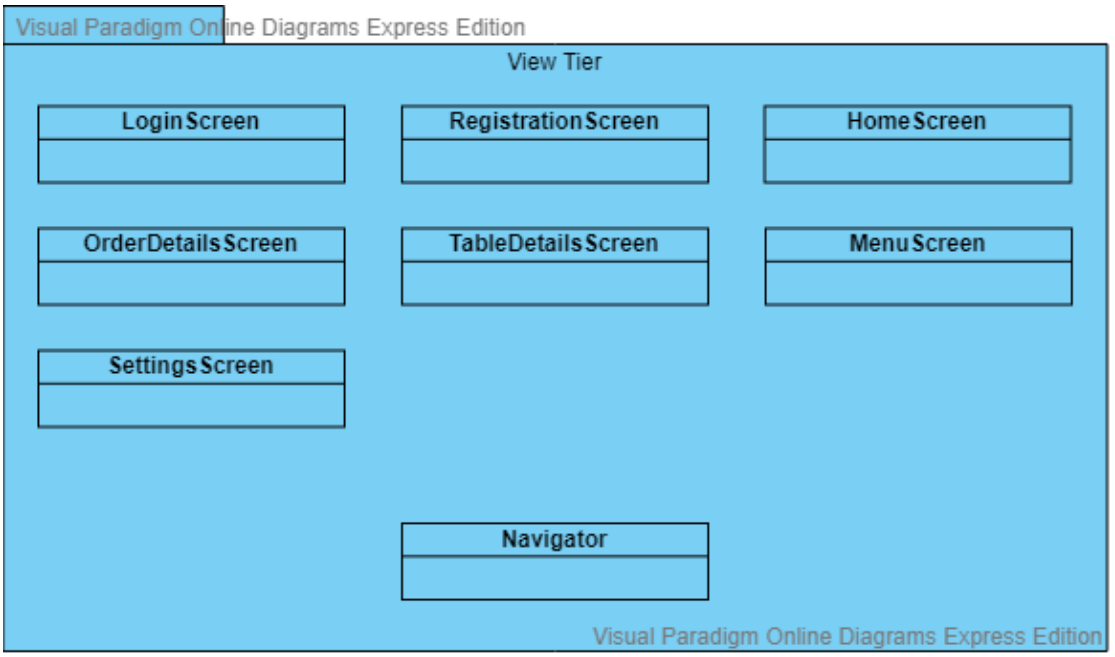
Model tier is the subsystem for Data elements in order to communicate easily and reliably.

Class	**Description**
Dao <<Interface>>	Dao stands for Data Access Object and isolates the layer of business from relational databases by providing abstract API [10].
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.

4.2 Administrative User



4.2.1 View Tier

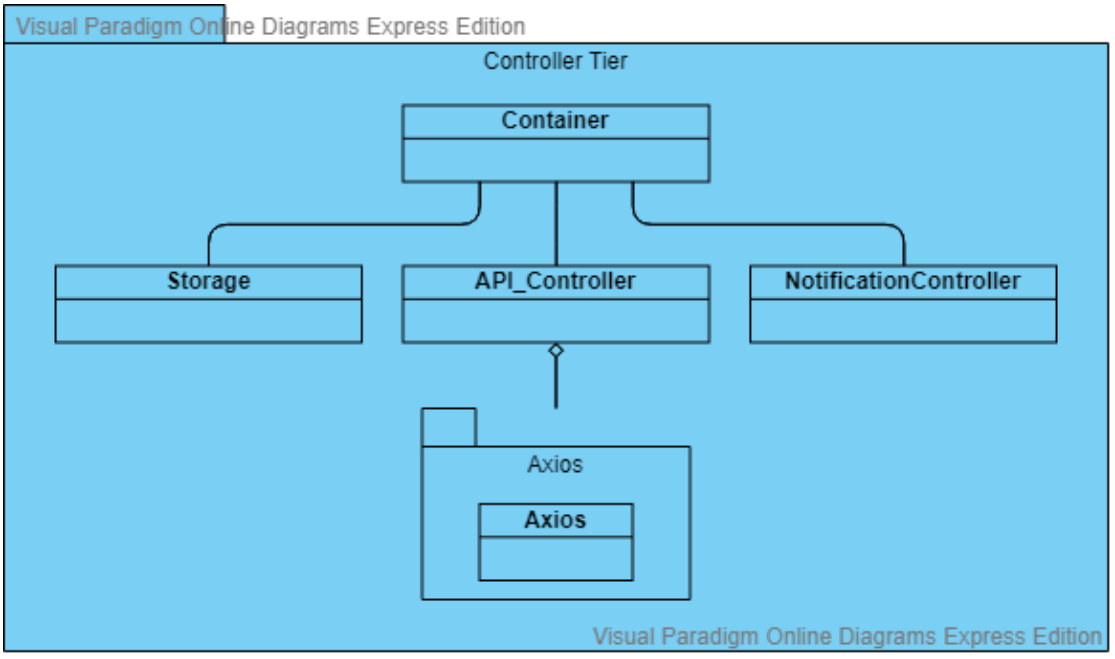


View tier is the subsystem for UI components.

Class	**Description**
Navigator	This class provides a navigation system to the app. It handles switches between screens.
LoginScreen	This class provides the login components to the user.
RegistrationScreen	This class provides the registration components to the user.
HomeScreen	This class provides tabs for the management of the orders and tables.

OrdersDetailsScreen	This class provides management of the orders and details of them.
TableDetailsScreen	This class provides management of the tables and details of them.
MenuScreen	This class provides restaurant owners management of the restaurant menu.
SettingsScreen	This class provides restaurant owners specific settings.

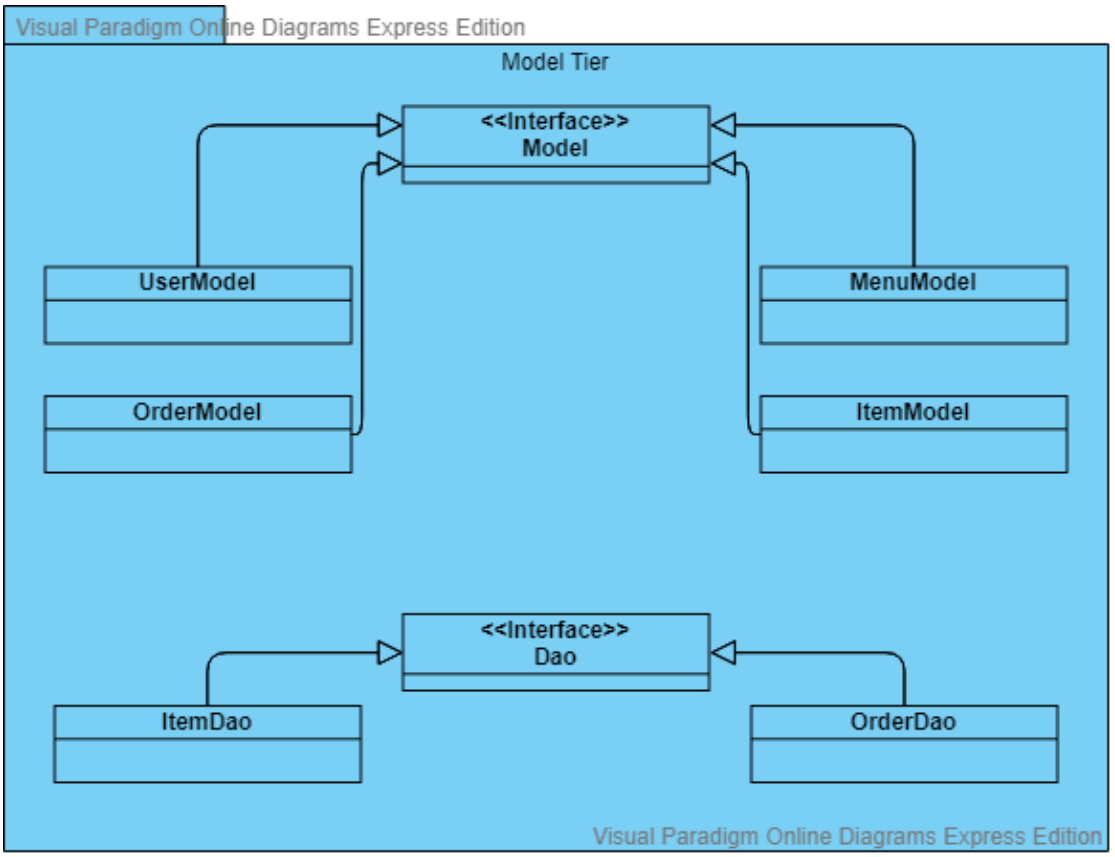
4.2.2 Controller Tier



Controller tier is the subsystem in order to handle events, requests and responses.

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.
API_Controller	This class provides sending http requests -get, post, put, delete- to the server via 25 instance of Axios class.
NotificationController	This is a singleton class that handles notifications of the app
Axios	This class belongs to third-party package Axios and is able to send http requests to a specified url and a port [9].

4.2.3 Model Tier

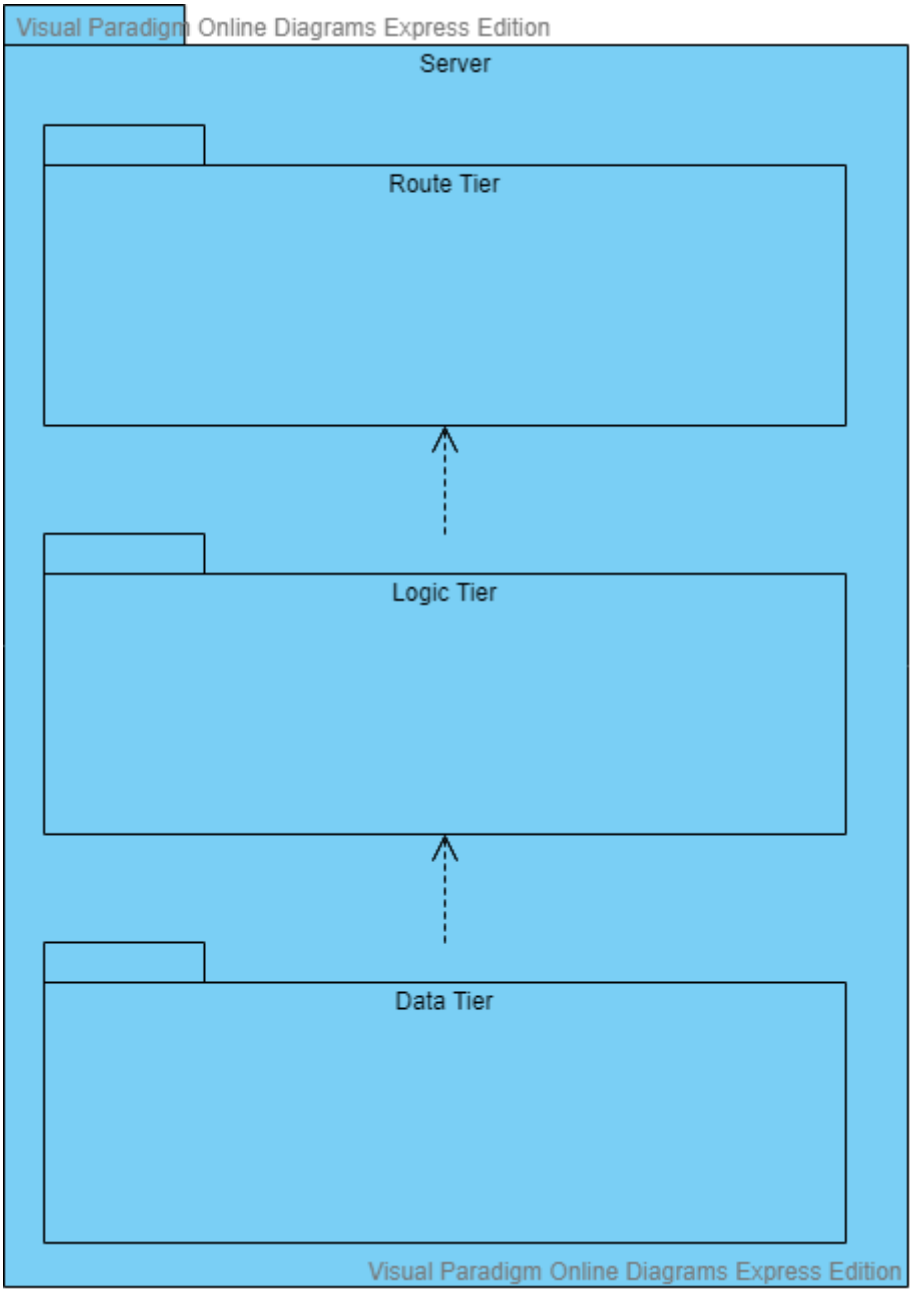


Model tier is the subsystem for Data elements in order to communicate easily and reliably.

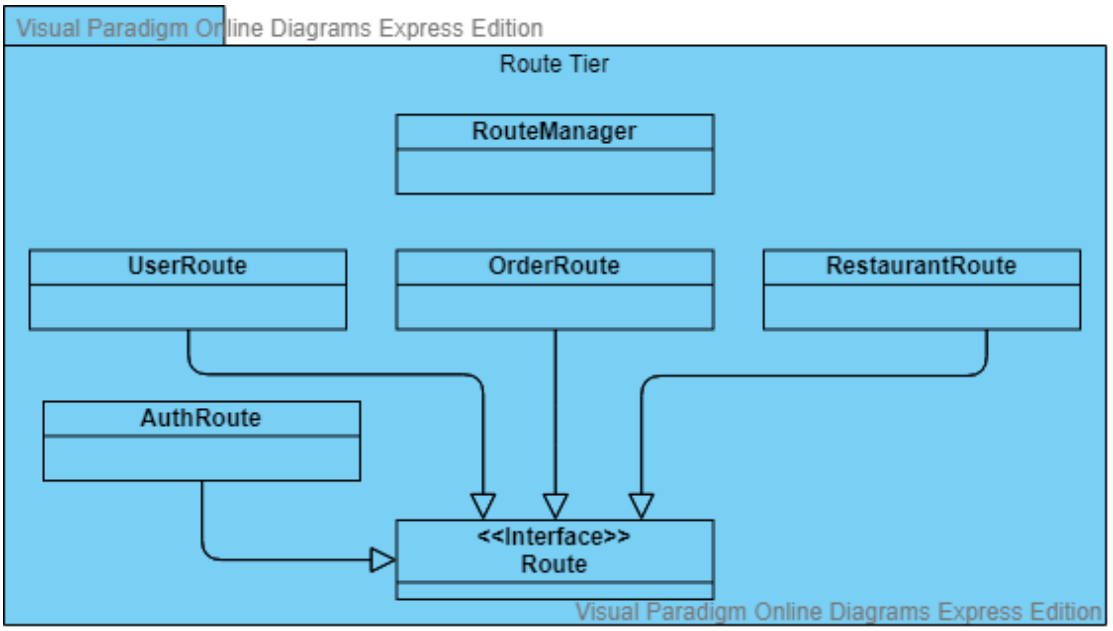
Class	**Description**
Dao <<Interface>>	Dao stands for Data Access Object and isolates the layer of business from relational databases by providing abstract API [10].
ItemDao ~ *Dao*	This class provides a Dao for

	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.

4.3 Server


Visual Paradigm Online Diagrams Express Edition
Server
Route Tier
Logic Tier
Data Tier
Visual Paradigm Online Diagrams Express Edition

4.3.1 Route Tier

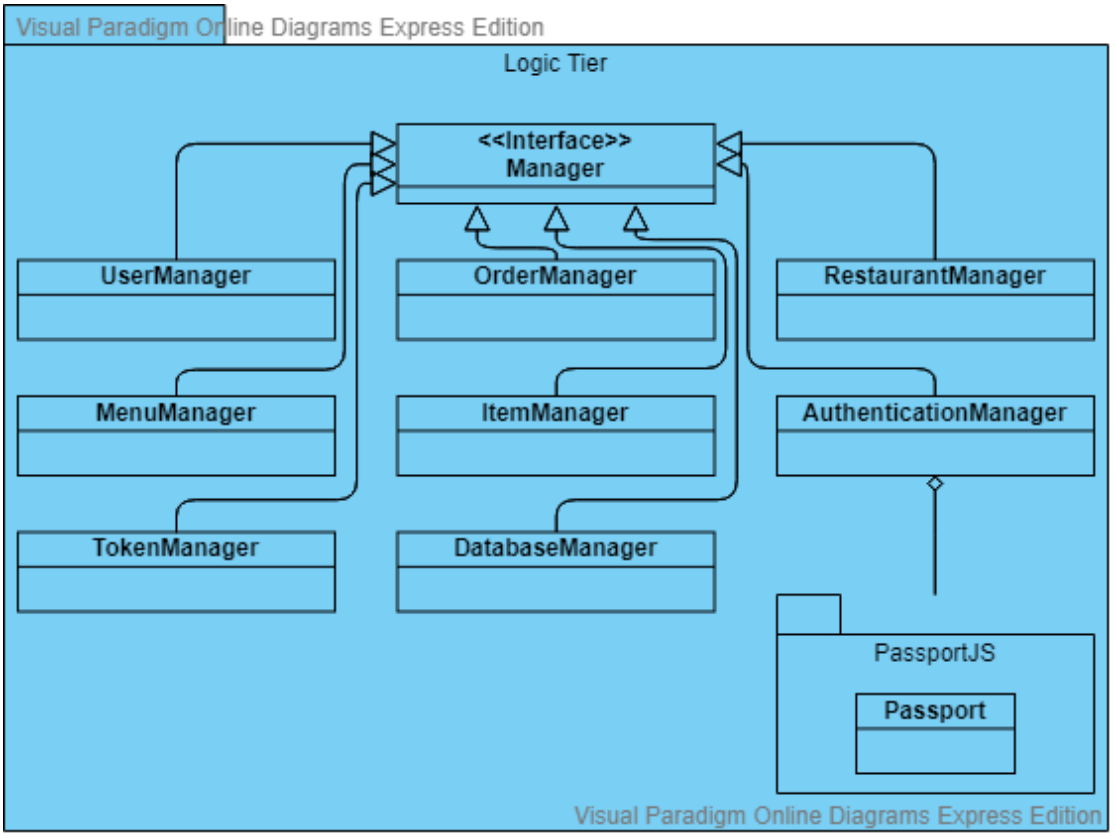


Route tier is the subsystem for Route elements in order to provide the concept of rest api.

Class	**Description**
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/*"
RouteManager	This class provides server management of the routes.

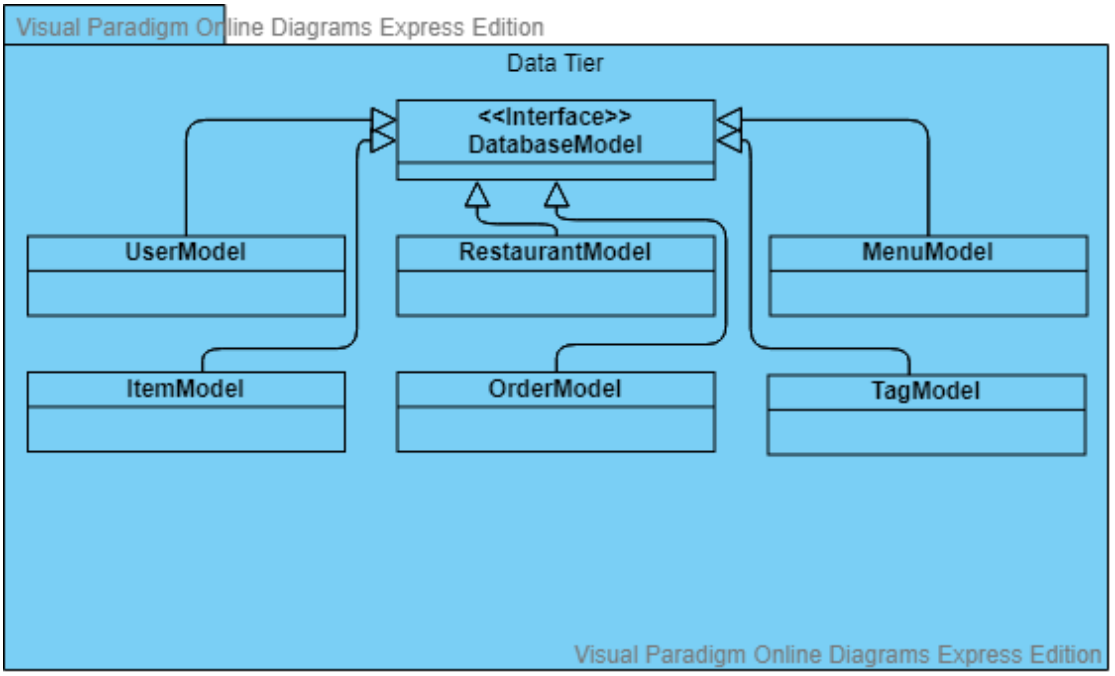
4.3.2 Logic Tier



Logic tier is the subsystem for management elements in order to provide the business logic.

Class	**Description**
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.
DatabaseManager	This class provides control of SQL database via GraphQL class.
Passport	This class is provided by third-party package PassportJS and generates middlewares and authentication protocols for the web service by Express [11].

4.3.3 Data Tier



Model tier is the subsystem for Data elements in order to communicate easily and reliably.

Class	**Description**
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.

5 New Knowledge Acquired and Learning Strategies Used

Knowledge Acquired	**Learning Strategy Used**
Relational Database Model	Literature Review : Review from Database System Concepts, A. Silberschatz; H. Korth; S. Sudarshan, 2011/6th, McGraw-Hill
ORM (Object-relational mapping)	Online Learning
Client-Server Architectures	Literature Review: Review from application layer topics of J.F. Kurose and K.W. Ross, Computer Networking, 5th ed., Addison Wesley, 2010.

Scan/Create QRCode with encryption/decryption in JS	Hands-on Experience
Electron, React frameworks	Online Learning: Online researches are done and official online courses are covered.
JavaScript basics	Online Learning: Online courses are covered

Table: Knowledge Acquired and Learning Strategies Used

6 Glossary

AWS - Amazon Web Services

REST - Representational State Transfer

HTTP - Hyper-Text Transfer Protocol

JS - JavaScript

QR Code - Quick response code

GDPR - General Data Privacy Regulation

ORM - Object-relational mapping

7 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] "Service Quality In Restaurants" *Service Quality In Restaurants*, https://www.ukessays.com/essays/marketing/service-quality-in-restaurants-marketing-essay.php/. [Accessed: 21- May-2020].

[3] "Amazon Web Services (AWS) - Cloud Computing Services." *Amazon Web Services, Inc.*, www.amazonaws.cn/en/. [Accessed : 20 -May -2020]

[4] Node.js. *Node.js*, nodejs.org/en/. [Accessed : 18 -May -2020]

[5] "React – A JavaScript Library for Building User Interfaces." – *A JavaScript Library for Building User Interfaces*, reactjs.org/. [Accessed : 18 -May -2020]

[6] "Build Cross-Platform Desktop Apps with JavaScript, HTML, and CSS." *Electron*, www.electronjs.org/. [Accessed : 19 -May -2020]

[7] "What Is REST?" *Codecademy*, www.codecademy.com/articles/what-is-rest. [Accessed : 20 -May -2020]

[8] "General data privacy regulation." https://eugdpr.org/. [Accessed: 21- May-2020].

[9] Axios. "Axios/Axios." GitHub, 7 Mar. 2020, github.com/axios/axios

[10] Baeldung. "The DAO Pattern in Java." Baeldung, 21 Mar. 2020, www.baeldung.com/java-dao-pattern.

[11] "Passport.js." Passport.js, www.passportjs.org/.