Bilkent University

Department of Computer Engineering

Project: Appeatite

High Level Design Report

Group Members
Alemdar Salmoor
Faaiz Ul Haque
Unas Sikandar Butt

Supervisor: H. Altay Guvenir

Jury Members
Ozcan Ozturk
Shervin R. Arashloo

Innovation Expert: *Emin Okutan*

Table of Contents

1. Introduction

1.1 Purpose of the system

The purpose of Appeatite is to create an interactive experience for people to be able to eat at fine dining restaurants with ease. This will benefit both customers and eatery management teams. Customers will be able to select, and order their desired meal via a mobile phone application with the help of QR code scanning. They will have the ability to learn detailed information about the menu items, such as prices, pictures, calories, ingredients and proportions. Restaurant staff will be required to monitor the incoming orders of customers, and can learn details such as the time of order, and from which table the order is being sent. The restaurant's will not require the need for additional staff to wait tables, as all order processes will be done via Appeatite's server. Furthermore, users are able to review different restaurants and can rate different dishes they have eaten to provide an insight to future potential customers. The main purpose of our system is to provide benefits to both the customer and the restaurant owners.

Customer Benefits:

- Fast service
- Access to an up to date menu
- Ability to learn more about what they are ordering
- Ability to remove specified ingredients from a dish
- Call for assistance at the touch of a button
- Learn about which dishes are preferred by other customers through review system

Restaurant Benefits:

- Minimal hardware required
- Have a up to date online menu at all times
- Learn information of where order is being sent from
- No need to hire extra employees; save costs
- Attract new customers with this new system

1.2 Definitions, acronyms, and abbreviations

QR Code: Quick Response code
In a process known as mobile tagging, the smartphone's owner points the phone at a QR code and opens a barcode reader app which works in conjunction with the phone's camera [3].

MVC: Model View Controller
MVC is an architectural pattern commonly used for developing user interfaces that divides an
application into three interconnected parts [4].

ROP: Restoran Otomasyon Programi
Explained in detail in section 2 of this report.

1.3 Overview

This report will state and explain the revised design goals, based off the feedback given in our analysis report regarding our application for both the user's side and restaurant service's side. Current relevant products that exist in today's market will be explained and its features will be compared with that of Appeatite's. The proposed software architecture will be illustrated using relevant diagrams and explanations. In particular, the subsystem decomposition of our application from both the restaurant client's side and customer client's side will be shown. Additionally, hardware/software mapping of our system will be analyzed in detail and relevant sections of our architecture will be mentioned such as data management, control and security and boundary conditions. Finally our primary subsystem services will be stated and briefly explained from both the server and client perspectives.

1.4 Design goals

Response Time:

Average Response time of the server to the client applications in the order of 1 second. This is the critical because the the system is a consumer application and needs to be interactive.

Throughput:

The system should be able to respond to many users simultaneously. The client applications will interact with one central database and should be able to retrieve information with very low latency. This requires that server functions are written intelligently and with correct usage of asynchronous functions. The final system should be able to accommodate at least 10 000 users using the system simultaneously. Firebase Firestore with its design to suit large amounts of data should be helpful.

Reliability:

The system should be reliable in the way that client application in general behaves in the same way for any restaurant, that is only the information provided by the restaurant changes and the rest stays the same. This way, user can observe specified behaviour regardless of the restaurant they are in. This means that user experience should be the same for any one person for different restaurants and include as minimum deviations as possible.

Availability:

The application should be online all the time, 24 hours a day. This is crucial because the user may want to visit a restaurant or view his history at any time.

Usability:

The application should be user friendly and be intuitive for different age groups. For example, reading the QR code and using digital menu is not a commonplace experience therefore the user experience design should deploy intuitive user interface components. We want every age group be able to use the application.

2. Current Software (Competitors)

Doing a market research and survey we have come to the conclusion that the service that we are proposing is not available in the market currently. However different variants could be seen throughout. After conducting interviews with some restaurant managers we realized that the current market is dominated by a brand named Restoran Otomasyon Programi (ROP) [5]. This program offers order management and store accounting management to the restaurant. This product boasts simple and tabular user interface and a fast overturn of tasks. According to the restaurant managers this program costs them $15000.

How is Appeatite different:

- Appeatite involves the restaurant customers in the order management process
- Provides easy to connect capabilities using QR-codes
- Eliminates the need of waiters
- Provides fully updated and visual digital menu
- Owners can add unlimited amount of information to the menu regarding meals like calories, ingredients, specialities etc.
- Customers will be able to rate their food

3. Proposed software architecture

3.1 Overview

The system is going to be based on the client/server architecture. The server will provide services to restaurant customer client application and restaurant client application. The request for a service will be done via HTTP requests.
This architectural style is chosen because it will provide centralization of control. A dedicated server will be able to control all data passing through the system which will ensure that the system is not damaged by any third party program or unauthorized access. Client-server architecture will also provide scalability. Any new restaurants can be added to the server without other services being affected and any number of customer clients can be created without any changes to the system. This architecture will also provide ease in maintenance as the server will be able to be updated with minimal effect on the clients.

There are going to be two client components:
1- The restaurant web application
2- The customer mobile application

3.2 Subsystem decomposition

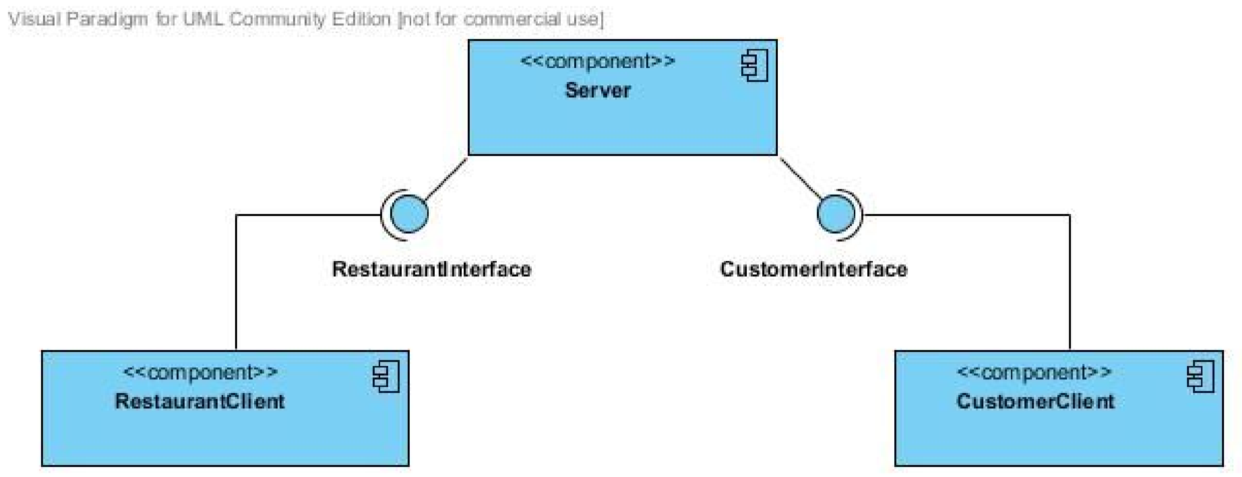

Figure: Subsystem Decomposition

3.2.1 Server

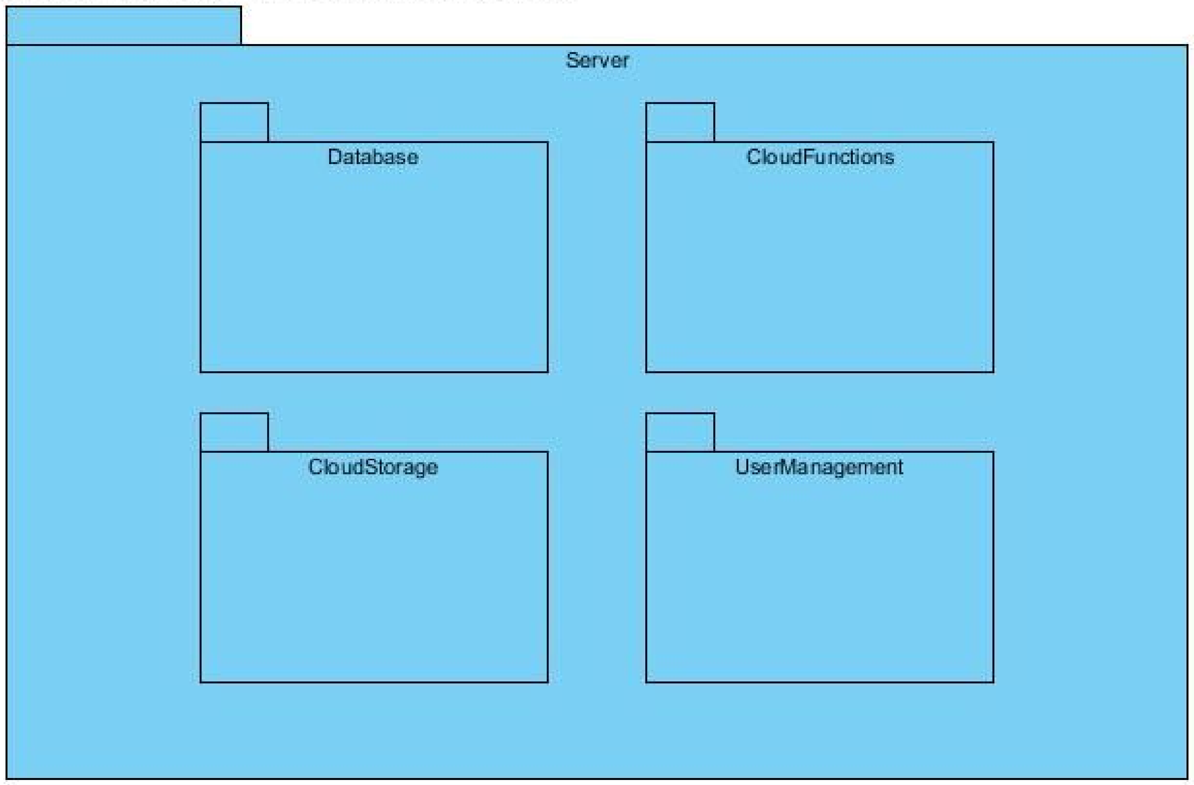

Figure: High Level Server Component Design

Server subsystem services are explained in section 4.1

3.2.2 Clients

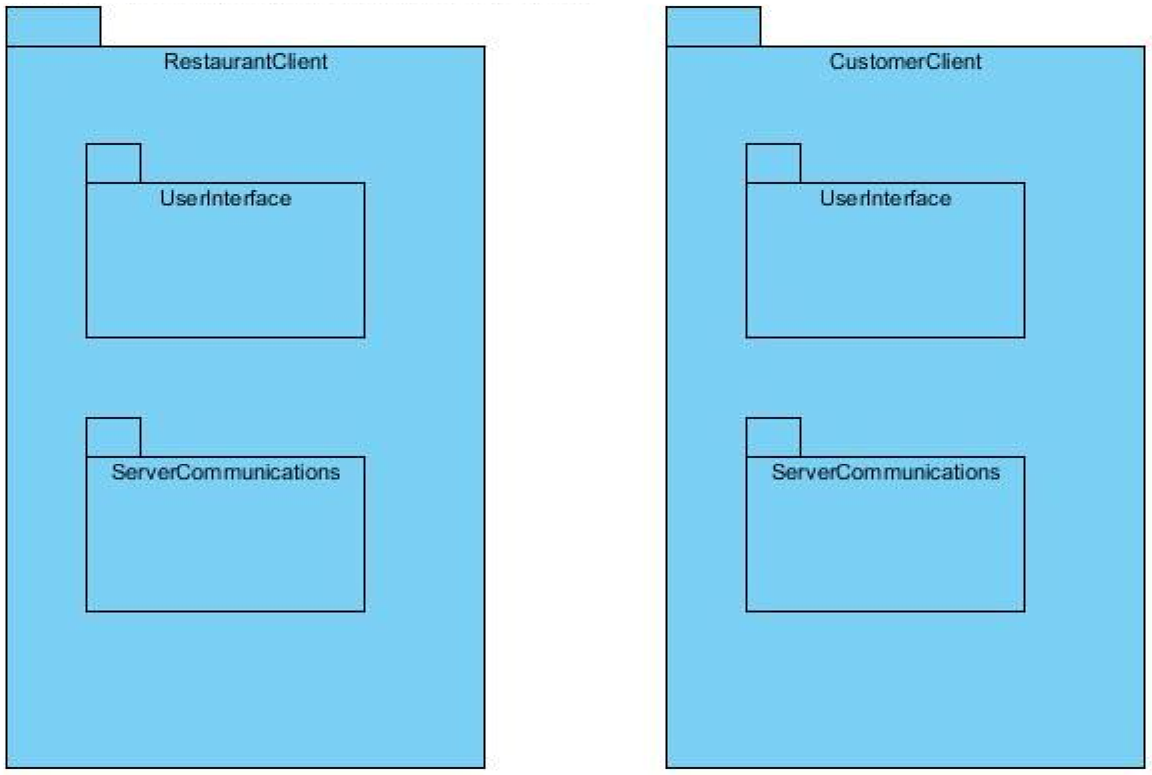

Figure: High Level Restaurant and Customer Clients Design

Client subsystem services are explained in section 4.2

3.3 Hardware/software mapping

Our system will require the restaurants to have a web browser supported device. The application will be able to run on any computer with an HTML5 supported web browser. The computer must be connected to the internet to connect to the server. To generate/print the QR code the restaurant must also have some kind of printing device. A general mobile phone or a tablet will be required from the customers to use the application. The cellphone/tablet must have a camera to scan the QR codes. The mobile application will also require an internet connection.

3.4 Persistent data management

Persistent data will be stored in Cloud Database, Cloud Storage, and Google Firebase Authentication.
Below is the data that will be stored in the Cloud Database, in the format of objects and its data members contained within it:

- Restaurant
  - id,
  - name,
  - rating

- Table
  - id,
  - restaurantId,
  - qrCode

- Review
  - id,
  - userId,
  - restaurantId,
  - text,

- MenuItem
  - id,
  - type,
  - restaurantId,

  - calories,
  - rating,
  - expectedWaitTime,
  - price,
  - imageLink,
  - discountPercent,
  - ingredients

- User
  - id,
  - name,
  - username,

- Favorite
  - userId,
  - menuItemId,

- Platter
  - id,
  - totalPrice,
  - menuItems
- Order
  - id,
  - date,
  - userId,
  - platterId

❖ The large data such as images will be stored in Cloud Storage.

❖ Information related to user authentication will be stored in Google Firebase Authentication that provides a secure and trusted platform.

3.5 Access control and security

For restaurant customer clients, access to the mobile application will require a one-time register where users enter personal details such as their usernames, email addresses and passwords. Once registration is complete, users will always be automatically logged in when using their original mobile device they used to register. In the case where they delete their application or attempt to log-in with a new device they will be prompted to the sign-up menu where they will be requested to enter their existing credentials that will now be stored in Appeatite’s database. The security of user authentication details is delegated to Google Authentication. Google does not disclose this information to anyone even the creators of the service that this authentication information is used for.
For restaurant staff client application, web application will require login on the machines they use the restaurant client application on and they will be logged in the system until the explicitly log off or shut down the machine. Restaurant client application will also require one time registration to identify them in the system.

No restaurant will be able to see information related to users such as Order History or favorite restaurants. No user will be able to see information that restaurants do not share explicitly like orders from all of the customers.

All of this will be accomplished by complying to correct design practices regarding the security like principle of least privilege. All of the data member variables will be private in client applications. Proper database rules will be deployed in the server that will disable anyone from viewing database entries for the customer users or restaurants.

3.6 Global software control

Customer order synchronization: Customer orders have to be synchronized in the way that the customer and the restaurant must have the same instances of the order at all times. Updating an order might be an issue with this regard. However the customer must have this feature of updating an order, this will be implemented by considering each update by the customer as a separate order instance and adding it to the list of orders.

Restaurant registration synchronization: The system must be implemented such that before the completion of restaurant registration no customer client can access that restaurant information.

Google Firebase sync, async and promises: Sync, async and promises are functions supported by Google Firebase which help manage functions' lifecycle and ensure that they resolve properly. This is done by handling the termination of the functions that run for too long or loop infinitely. These services also ensure that the Cloud Functions instance that is running client functions does not terminate before they fulfill their terminating condition.

3.7 Boundary conditions

Server Initialization: The server will be initialized once using Google Cloud. Once online the server must always be running except during server maintenance. During the initialization the database will be setup and cloud storage will be allocated to the system.

Restaurant Client Initialization: Restaurant client can be initialized by going to the system webpage using any browser and registering the restaurant in the system. Registering will involve going through multiple forms and inputting restaurant information.

Customer Client Initialization: Customer client initialization will require the customer to download and install the mobile application and then going through the one-time sign-in using the Google or Facebook login services or email/password registration.

Restaurant Client Termination: When the restaurant client terminates, the server must ensure that no new customers can check into that restaurants. Additionally all of the data related to this restaurant should be deleted in the database, and the ones kept for statistical purposes should be managed properly. If some customers are checked into that restaurants, they must be prompted that the restaurant services have shut down.

Customer Client Termination: The customer client can be terminated at anytime according to the users will.

Server Termination: Termination of the server will be a rare occurrence. Server termination can be caused due to server maintenance or certain unforeseeable circumstances. Server termination will cause all access to the server to be terminated and users will not be able to access system services. This however is once again to occur in natural circumstances because most of the server components used in the system will be physically located in Google company's facilities.

4. Subsystem services

4.1 Server:

The server will act as the central component of the Appeatite system. Restaurant and customer clients will be bridged via the server. The server will include many functionalities that are crucial for the system. These functionalities can be divided into four main subpackages.

1. **Cloud Functions:** Cloud Functions will enable us to automatically run backend code which can be triggered by client requests such as retrieving menu information and authenticating user access.

2. **Database:** The database will store data relevant to both customer and restaurant clients. For customers this includes user ids, restaurant history, favorites, and reviews. For restaurants this includes, generic menu details, current orders placed, and a list of QR codes corresponding to their relevant tables. The interfaces of both the mobile application and web application will be updated by interacting with information from the database via cloud functions.

3. **Cloud Storage:** Some persistent data, like the food item images require large amounts of storage. This kind of data will be stored on the cloud storage and the links to these items will be stored in the corresponding database entries.

4. **User Management:** User management package will handle all the user authentications for both the customer clients and the restaurant clients. This package will store the user credentials and passwords.

4.1 Clients:

The client components comprise of minimal design and are very similar to each other. As the server component will be handling all of the complex operations, the clients are left with the user interface and the server communications packages.

1. **User Interface:** This package can be considered as the primary package of the clients' component. The user interface will provide a meaningful interface for the restaurant and the customer clients which will be responsible for all the user interaction with the system. The user interface will receive user input and relay it to the server communications package which in turn will respond to the client requests.
   The restaurant and customer clients will differ in the user interface with different functionalities in each case.

2. **Server Communication:** This package will handle all the clients' communications with the server. Every request from the clients will be sent to the server using this package and every response from the server to the client will be received through this package. This package in restaurant and customer clients will differ based on the use cases for each client.

5. Glossary

While the terms below may be very familiar to the Members of Bilkent CS Department, they were included if anyone from general public would like to view the document.

Response Time: the length of time taken for a system to react to a given event. In our case, event triggered from client application managed in the server and sent back to the client application

Throughput: the number of responded requests at the server at any given time.

Web application: Software application that runs on a remote server. Web browsers are used to access Web applications, over the Internet

HTML5: Latest evolution of a markup language used for structuring and presenting content on the World Wide Web [6]

Cloud Database: Database that typically runs on a cloud computing platform, access to it is provided as a service

Cloud Storage: Cloud Storage allows world-wide storage and retrieval of any amount of data at any time [7]

Firebase Authentication: Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to the application. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter. [8]

Cloud Firestore: Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform [9]

Asynchronous function is a function which operates asynchronously via the event loop, using an implicit Promise to return its result [10]

Promise: A promise is an object that may produce a single value some time in the future: either a resolved value, or a reason that it’s not resolved (e.g., a network error occurred). A promise may be in one of 3 possible states: fulfilled, rejected, or pending. Should be handled by attaching a callback [11]

6. References

[1] Bruegge, Bernd, and Allen H. Dutoit.
Object-oriented Software Engineering: Using Uml, Patterns and Java. Upper Saddle River,
NJ: Prentice Hall, 2003. Print.

[2] *Visual Paradigm*, Visual Paradigm, 2018

[3] “QR code” [Online]
https://whatis.techtarget.com/definition/QR-code-quick-response-code [Accessed Dec 29]

[4] “MVC architecture“ [Online]
https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture [Accessed Dec 29]

[5] “ROP - Restoran Otomasyon Programı” [Online]
https://www.rop.com.tr/ [Accessed Dec 29]

[6]”HTML5” [Online]
https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5 [Accessed Dec 29]

[7] “Cloud Storage” [Online]
https://cloud.google.com/storage/docs/ [Accessed Dec 29]

[8] “Firebase Authentication” [Online]
https://firebase.google.com/docs/auth/ [Accessed Dec 29]

[9] “Firebase Firestore” [Online]
https://firebase.google.com/docs/firestore/ [Accessed Dec 29]

[10] “Async function” [Online]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function [Accessed Dec 29]

[11] "What is a Promise" [Online] https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-promise-27fc71e77261 [Accessed Dec 29]