TED ÜNİVERSİTESİ

CMPE 491

Senior Design Project I

High-Level Design Report

Project Members:

Selin Akyurt - 17878041578

Melisa Uzulu - 3906408574

Recep Berkay Engin - 22271682594

Supervisor: Yücel ÇİMTAY

Jury Members: Aslı Gençtav, Venera Adanova

Table of Contents

1. Introduction

In our day-to-day basis, we all struggle with time management. We sometimes lose track of time, overwork because of the coming deadlines and always in a rush. Mobile apps help us to save time with making our lives easier with each click.

When it comes to going to a bakery and ordering a cake on your special day, it takes so much time. We even forget to celebrate special days due our busy schedules. In some cases, we need to order customized cakes for occasions such as birthdays, engagement/wedding parties etc. The process of going through each bakery, talk about your wants and needs, get prices and compare each bakery can take hours and days.

This is why we came up with an idea of solving this problem for you with an app called "Cakery". Cakery will list you bakeries that you can place orders. Customers can see each bakery items with descriptions or they can place a customized order. Bakeries will deliver the order to desired address on desired time and date.

However, when we think about this problem from a seller perspective it doesn't get any easier. Showing your hand made cakes and spreading your name to get recognition is not easy, especially for small businesses. We are also solving this problem with giving opportunities to bakeries to sell their products. We hope to give the recognition small businesses deserved for their hard work and make taking an order process easier for them.

1.1 Purpose of the system

Time is the most valuable thing in our world today and when we need a little help with saving time, web and mobile applications are here to serve us. Our application "Cakery" will help customers to order cakes for their special celebrations and will save time with allowing you to reach your local bakeries with just one app. Our application will not only serve ready to eat cakes but will give you an opportunity to create your customized cake from the bakery you choose. Your cake order will arrive on the date and time you want. All you need to do is just place your order using Cakery App. The whole purpose of the application is solving the problem of going to bakeries, choosing a cake or ordering a customized cake, which takes more time and effort to order, and bring it home.

1.2 Design goals

The main purpose of the Cakery App is to make the cake ordering process easier. In order to maintain a working system, we decided to include following properties in our app design:

1.2.1 Extensibility and Modularity

Cake app consists of two sub mobile application, one for buyer and one for seller, and a web page for the admin. Additional functionalities and changes can be added to the system to meet the needs of users in three different end systems. This is why the system will be an extensible product that has been designed from its earliest stages for enhancement in case of any need.

1.2.2 Maintainability

The three sub systems are highly modular and not highly coupled, this is why any problem within the system will most likely not affect the other systems. This is also important for the quality of the service.

1.2.3 Reusability

Due to system being designed as highly modular, any module of the project can be used in other software applications.

1.2.4 Usability

The whole system is not required to high learning curve. This is why Cakery app targets variable users. User friendly interface is needed for the targeted user base. Overall system will be easy to understand and use without a need for extensive guidance or instructions. However, due to its nature system needs users to successfully purchase products in order to run smoothly. Admin portal should be able to contact developers to make suggestions or report problems they are facing in order the improve the system usability.

1.2.5 Performance

Users should be able to purchase their desired wants and needs without waiting for a long time. This will keep usability and interactivity at acceptable levels as well. Human interactions, clicks/scrolls/selections, should not take long to provide good user experience.

1.2.6 Availability

System should be available for the users, sellers, and admin all the time. However, purchases can differ from the chosen seller's working hours and days. Cakery app will be free to use and can be downloaded from the mobile app store.

1.2.7 Reliability

Occurred errors and bugs should be fixed with quick and easy updates in order to make the system workflow smooth. Admin has rights to block the user/seller in case of any mis usage detected. Personal data of the users should be preserved well.

1.2.8 Scalability

System should be available for the users, sellers and admin all the time. However, purchases can differ from the chosen seller's working hours and days. Cakery app will be free to use and can be downloaded from the mobile app store.

1.3 Definitions, acronyms, and abbreviations

Cake: an item of soft sweet food made from a mixture of flour, fat, eggs, sugar, and other ingredients, baked and sometimes iced or decorated.

Bakery: a place where bread and cakes are made or sold.

Customer: a person who buys goods or services from a shop or business. a person who buys goods or services from a shop or business.

Order: request (something) to be made, supplied, or served.

Product: an article or substance that is manufactured or refined for sale.

Seller: a person who sells something.

Small Business: privately owned corporation, partnership, or sole proprietorship that has fewer employees and less annual revenue than a corporation or regular-sized business.

1.4 Overview

The Cakery apps serves the list of the local bakeries in your town and gives you an opportunity to look through their ready to serve cakes, choose one and order within a minute or if you have a special day coming up you can order a customized cake from the bakery you chose. Ordering customized cakes takes time but with the Cakery app, customers don't have to go through

each bakery, ask for prices, explain their detailed wants and needs. We solve this problem for you in this mobile app where you can order your dream cake with one button. Cakery app is also giving the opportunity to bakery owners present their desserts and cakes in broader perspective.

System works when you sign in the application, using your email, password, telephone number and address. After signing in, the application will show you the lists of all the bakeries in your town based on your location. All the member data will be stored in our database. In the database system we will store each bakery information as well as member information.

The application has two specific purposes. One, as explained previously, order ready to serve cakes from one of the bakeries you chose. Those ready to serve cakes are perfect for small celebrations. Bakeries in the application must give detailed information about the cake and they can place pictures of their cakes as well. After customers deciding what cake, they want to order, they can add it to their chart and finish their order choosing a payment method.

Second purpose is to give the freedom of ordering customized cakes without physically visiting each bakery in your town. Customers can select their bakery and if that bakery serves customized cake option, they can place an order. Customers have to fill out a detailed form about their customized cake and they can add notes or pictures to make sure everything is clear. After replacing their order, they can choose the delivery date and address.

3. Proposed Software Architecture

3.1 Overview

The proposed architecture gives a more detailed explanation about the structure of Cakery's system and brings together the structures and class implementations explained throughout the analysis report. In the subsections below, we intend to explain subsystem interactions, the technical decisions we have made so far and the class structures we are planning to deploy.

3.2 Subsystem Decomposition

Our application will be mainly following the client-server model. The client side is usable by a single person who is either a buyer or the seller. It allows the steps of a bakery and cake selection steps for the buyer. It also allows managing menu and displaying bakery data steps for the seller. The client side will be in the form of a mobile application. The server side will be handling the data storage and processing of orders for each buyer and seller.

The client side of the model can be divided into sections in order to be explained clearly. The first section is the presentation subsystem which is the graphical user interface that the seller or the buyer interacts with. The second subsystem is a controlling subsystem which manages interactions mainly with server and also with local storage. For example, when a seller wants to register to the application, presentation will communicate with controlling subsystem first. Then the controlling subsystem will take an action to send a request to the server and communicate the server response back to the presentation subsystem. Similarly, if the buyer creates an order, the controller communicates with the server in order to register the order and go through the steps of buying.

The server side of the model can also be divided into several subsystems. The server side can handle requests as well as managing and processing data. While handling the request, the server side will be communicating with each user of the system (either a buyer or a seller), and schedule the active requests made by the user, sends the request to the corresponding module, waits for the response and takes the response to the request back to user. For example, if a buyer sends a request of a cake order from a bakery, the request handler invokes the data management to save the order and finally sends the order to the seller. The subsystem we are achieving to create for storing and managing our data holds a storage space large enough to save the orders made from each user.

3.3 Hardware/ Software Mapping

Cakery is an application which is developed for mobile phones. There won't be any hardware component needed rather than a mobile device to use Cakery. We will be using Flutter for application creation and Google Firebase for data storing and management.

3.4 Persistent Data Management

Persistent data includes both products ordered by customers and information held on the server. When the customer adds their order to the cart and finally completes the purchase, the order information is saved both on the server and locally. Orders are saved as a template image. As long as the total size of the orders placed does not exceed the limit provided by the server, it remains on the server. In addition, this information can be used to obtain information about past orders and to view the order summary. The server keeps information about users and orders in the Google Cloud Firestore database. This information is used for services such as authentication and logging into the system. A database system is used to store orders, authentication data, and other data about customers and sellers.

3.5 Access control and security

Security issue is one of the important issues for our application. Our application contains the personal information and home addresses of our customers. Therefore, data should not be leaked in any way. That's why we care about security to protect our users from fake players like bots and other malicious users. To use the Cakery app, users must create an account. For authentication services, each user must have a unique password and email address. If the system accepts the user after entering their mail and password information, they can access the application. If a problematic user or vendor is active on the system, the admin can block the user or vendor from the admin portal. In this way, the malicious user or seller is kicked out of the system.

Each user can only see their account information and addresses. Other users do not have access to their information and home addresses. Passwords are stored as a hash in the database and authenticated with the client-side generated hash. The server never receives the password in plain text.

3.6 Global software control

Cakery application uses an event driven software control composed by mobile application and server. Every choice made by the user causes a request to be sent to the server. For example, the creation of the login request sent by the user with the username and password, and the verification of the entered information are performed between the server and the client. After authentication, the server is asked to display a list of sellers to the user.

- When the user wants to buy a product from the seller, the information is transmitted from the server to the seller after the purchase is made.

- After the seller displays the order information, the server sends the delivery information sent to the server back to the Cakery mobile application.

Finally, another example is;

- When the admin wants to delete the seller from the system, she/he sends his/her request to the server.

- The server processes the request and sends it to the Cakery application.

- As a result, the seller that the admin wants to delete is deleted from the Cakery application and the user cannot view this seller.

- These two services (Admin Web Portal and Cakery App) are running at the same time.

3.7 Boundary conditions

Boundary conditions are conditions that specify how the user and developer should react to the initial behaviour of the system. Users need access to the client to use the Cakery application. The client can be reached on mobile, thanks to smartphones.

- **Initialization**

If the user has an account, they are expected to verify their username and password before logging into the system. If the user does not have an account, a new account is created and it is expected to be registered in the system. These operations can be performed using the buttons on the login screen. If the information required by the user to log in does not match the expected information, the system will fail and return to the login screen. An internet connection is required for the Cakery application to work properly and to communicate with the server. After logging in to the Cakery application, the user is presented with the screen with the sellers.

- **Termination**

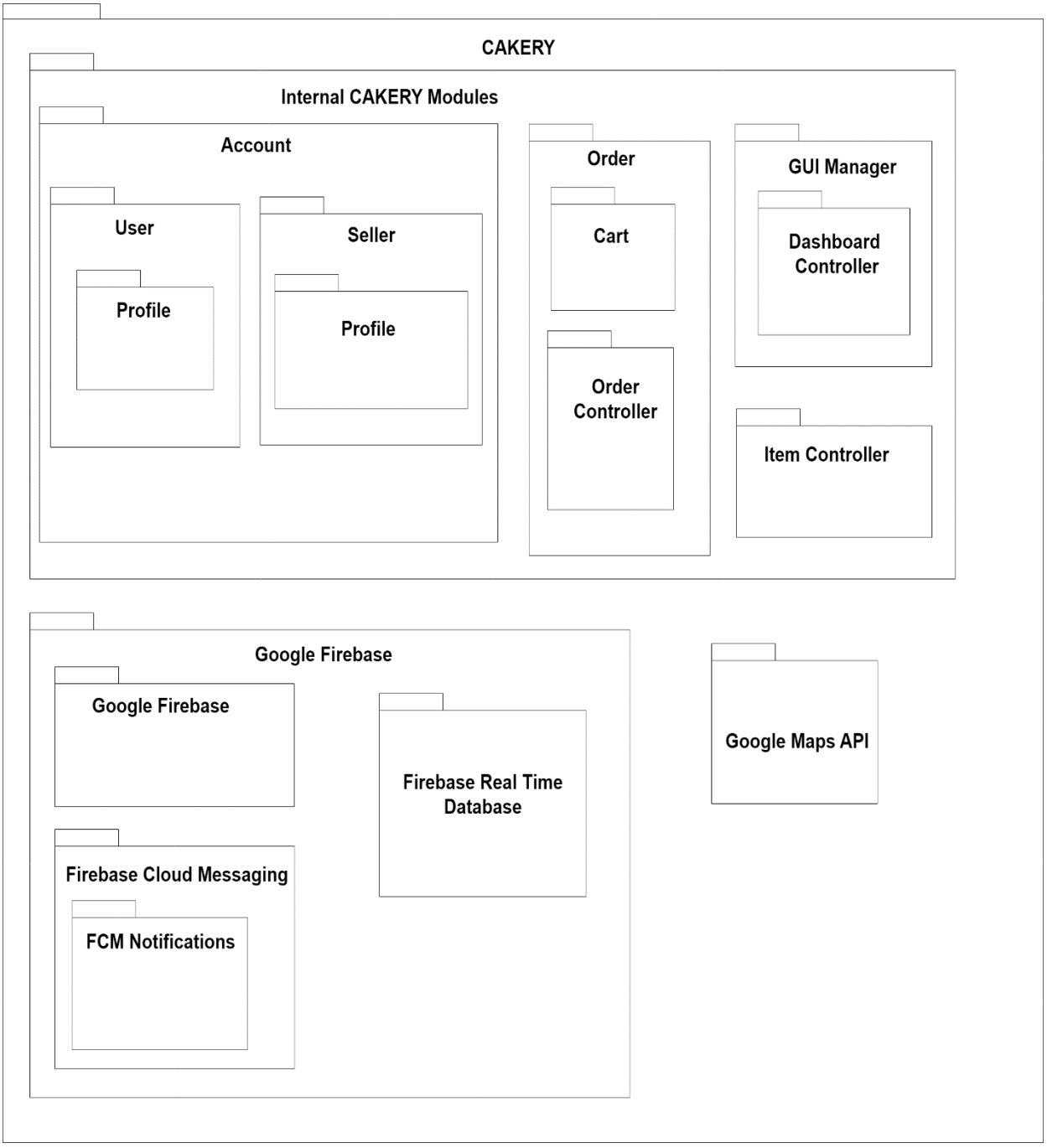
The user can logout of the Cakery application at any time, except that the client is waiting for the result of an operation from the server. For security reasons, the user is logged out when the application is terminated.

- **Failure**

If the client crashes, an attempt is made to save client data before the application is terminated. Lack of internet connection can cause errors in the application.

4. Subsystem Services

Subsystem services for Cakery application are detailed below. Here is the parse view of the Cakery app:



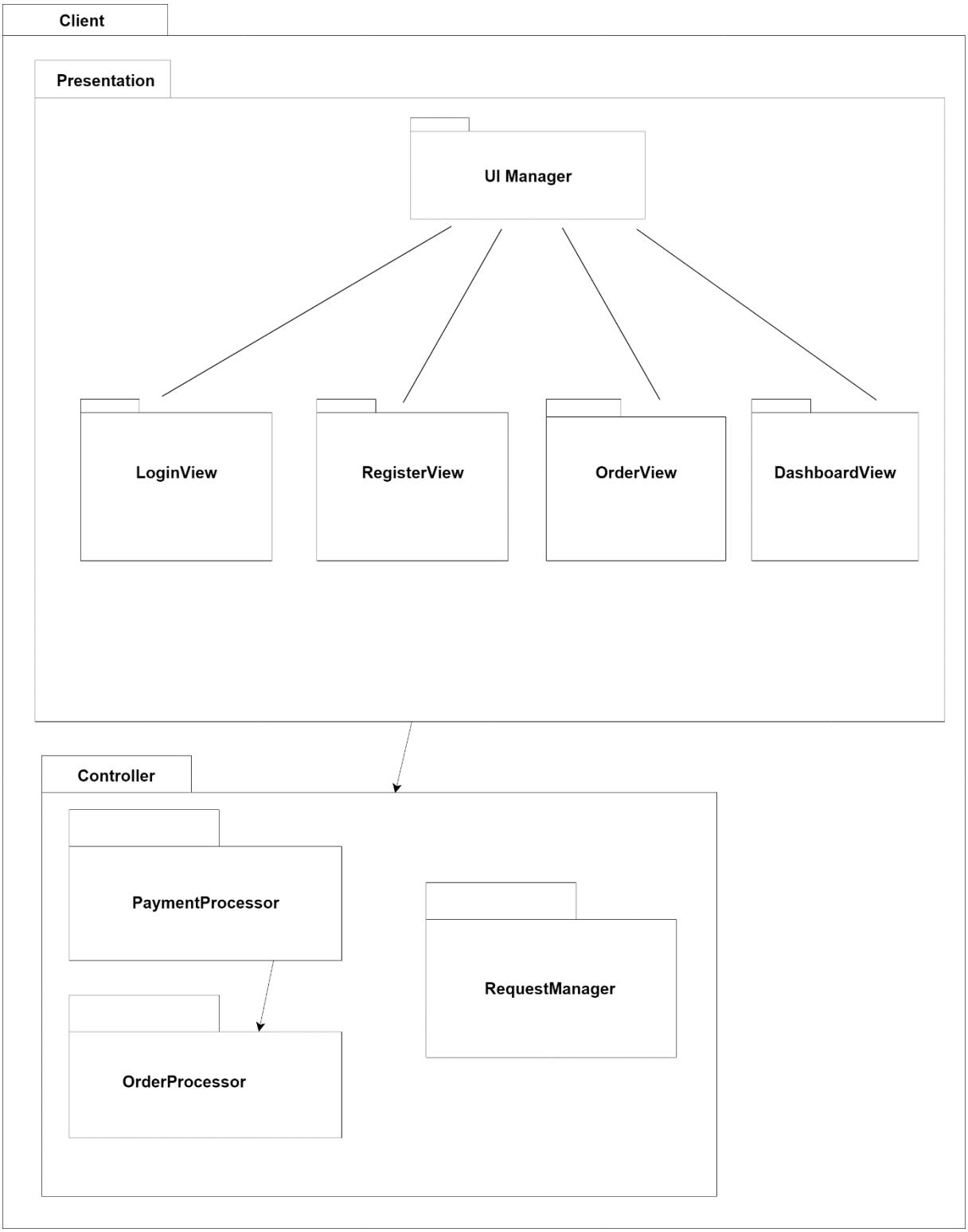
Subsystem services for Cakery App

Cakery emerges as a result of the interaction of two systems. These are client and server.

4.1 Client

The client can be used through smart devices with a mobile operating system. The client is the user interface layer where the user selects products and interacts with the system through which he/she orders and purchases products. It includes the client, presentation, and controller

subsystems. The presentation subsystem includes user interface elements for the user to interact with. The controller subsystem is responsible for generating the appropriate requests based on events initiated by the views and then communicating with the server to send the requests.



4.1.1 Presentation

The presentation layer contains views and their event triggering mechanisms.

- **UIManager**: A global administrator to mediate other view classes.
- **LoginView**: It is the view class for the login screen. It checks the credential formats before requesting the request creation. It uses RequestManager to make login requests.
- **RegisterView**: It is the view class for the registration screen. It checks the credential formats before requesting the request creation. It uses RequestManager to make registration requests.
- **OrderView**: It is the view class for the Orders screen. It lists the user's orders and allows them to view and access their details. It uses the RequestManager to list existing orders and the OrderProcessor to purchase products.
- **DashboardView** : It is the view class for the home screen. It is the part where the vendors are listed and shown. It uses RequestManager to get user information.

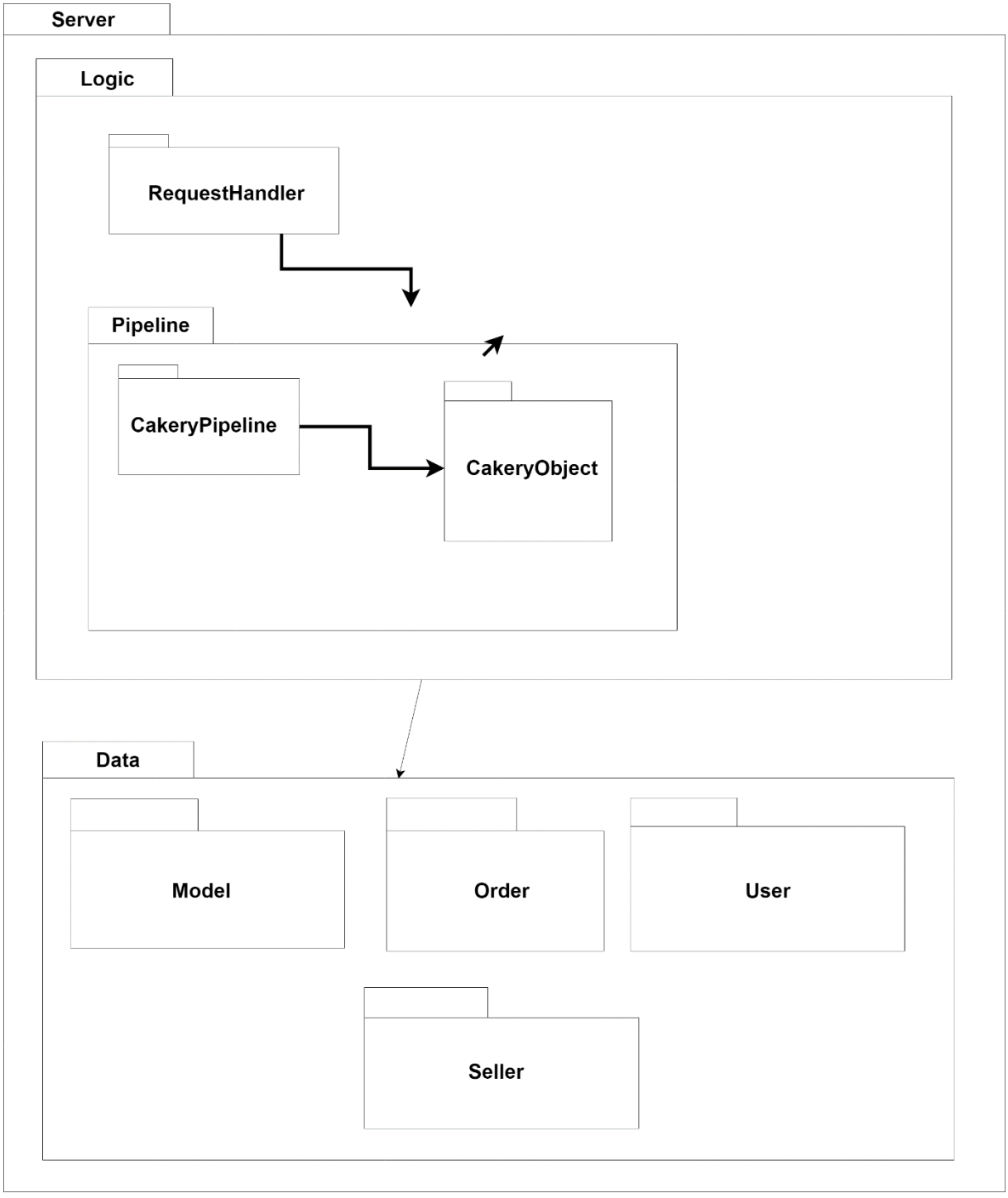
4.1.2 Controller

Client logic handles operations such as order processing and server communication based on events generated by views in the presentation layer.

- **PaymentProcessor**: Responsible for receiving payment in their mobile applications.
- OrderProcessor: It is responsible for creating the order, displaying the order in the order history and transmitting the order to the seller.
- **RequestManager**: Creates all requests based on events triggered by the presentation layer and serves to communicate with the server. It waits for the response from the server and reports the views of the result.

4.2 Server

The server is where all the transactions take place and then sent to the client. An order and the actions to be performed on that order are sent to the server as a request. The server then processes the ordered modules and generates an output. The output is then sent back to the user as a response. In addition, the server also manages authentication and updates to the user information database. Server is divided into Logic and Data layers. The logic layer accepts requests and updates/creates/removes data. The logic layer manages database operations and order processing. The data layer is where models and other persistent information are kept.



This defines the subsystem responsible for Cakery's logic and operation. Pipeline and Modules are the two packages that make up these processes.

- **RequestHandler:** Converts client requests into tasks and runs them in the pipeline. It generates responses and sends them back to the user.

4.2.2 Pipeline

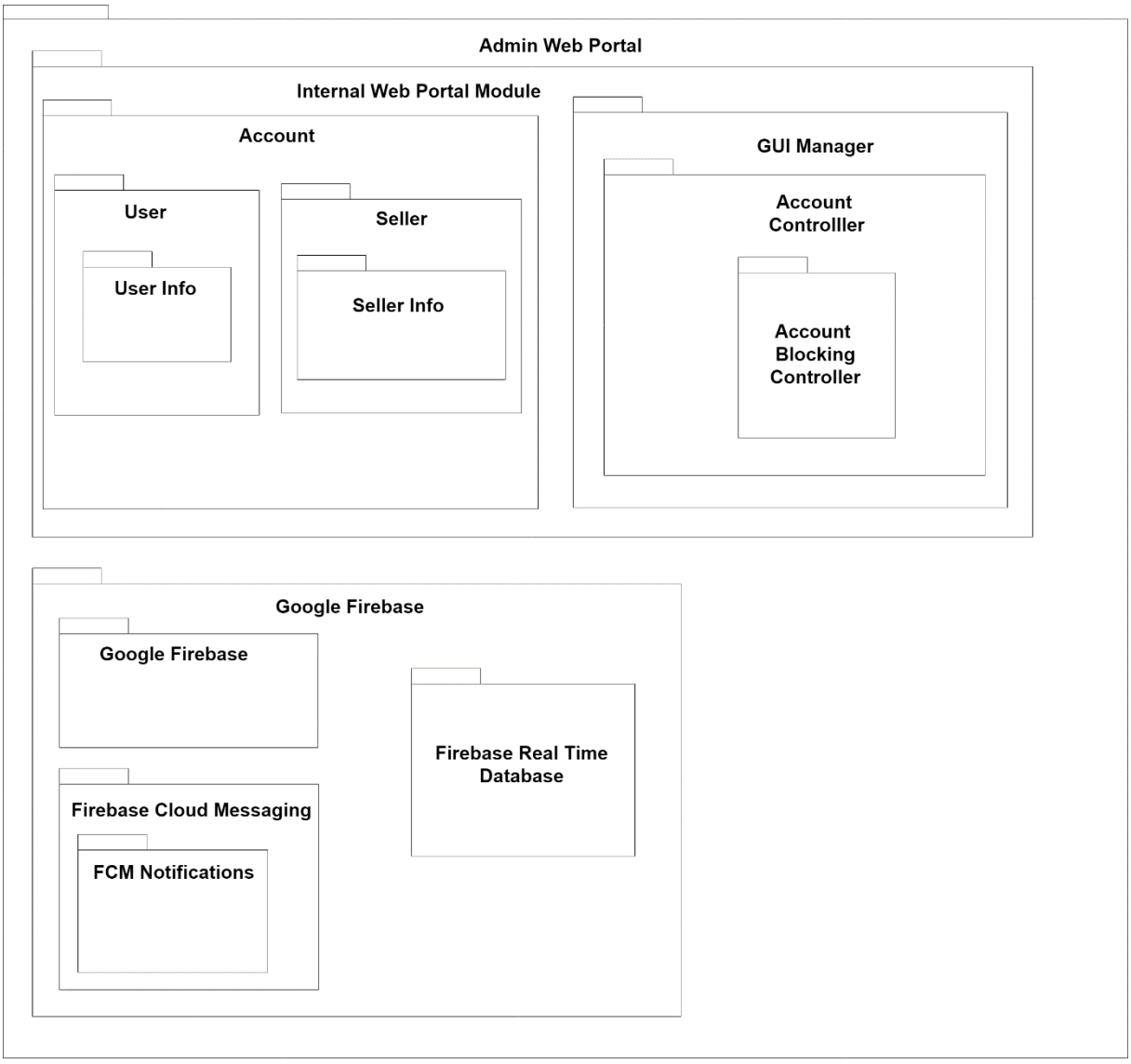
Cakery has a modular design structure.

- **CakeryPipeline:** After an incoming request, all necessary modules are taken from the Modules package and converted into sequential operations. The pipeline runs these modules one after another to produce an output.
- **CakeryObject:** This is the object created using a request. The pipeline then processes this object; does it to generate the final output note.

4.2.3 Data

This layer is where the data used by the application is kept.

- **Model**: Parameters for models used in ordering modules.
- **User**: Contains user information such as username, user passwords, phone numbers, address information kept in the NoSQL database.
- **Order**: Orders created by users.
- **Seller**: Contains information about the products and products (price information, product description, etc.) of the vendors kept in the NoSQL database.



Subsystem services for Admin Web Portal

Admin Web Portal emerges as a result of the interaction of two systems. These are client and server.

4.3 Client (For Admin Web Portal)

The client can be used through computers with Windows/MacOS operating system. The client is the admin interface layer where the admin sees information about the user and vendors and interacts with the system. It includes the client, presentation, and controller subsystems. The presentation subsystem includes admin interface elements for the admin to interact with. The controller subsystem is responsible for generating the appropriate requests based on events initiated by the views and then communicating with the server to send the requests.

4.3.1 Presentation

The presentation layer contains views and their event triggering mechanisms.

- **GUIManager**: It is a global manager to mediate other view classes.

4.3.2 Controllers

Client logic handles user and vendor activation/blocking based on events generated by views in the presentation layer.

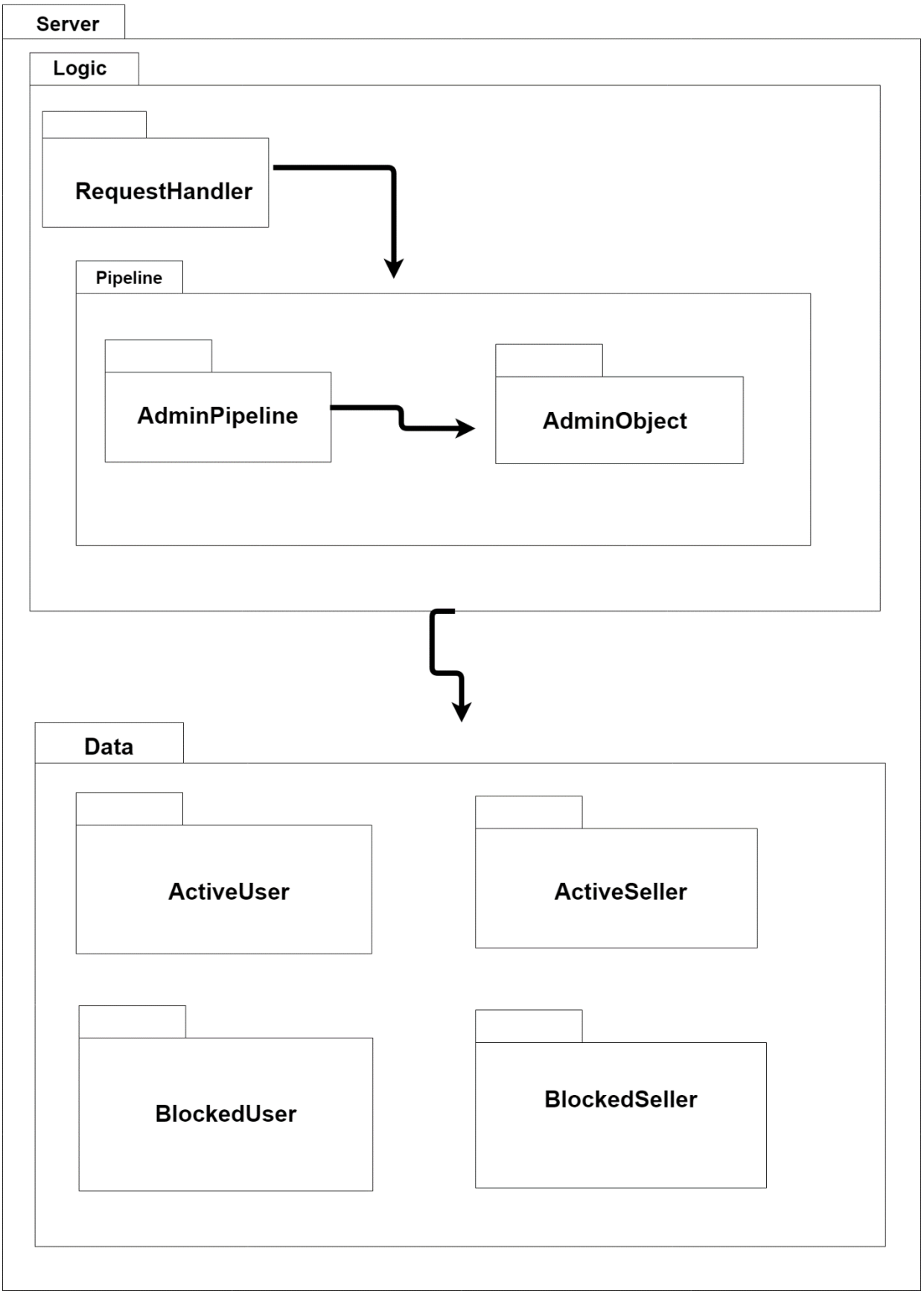
- **Account Controller:** It is responsible for displaying the information of the user and the seller.

- **Account Blocking Processor:** It is responsible for blocking the user or seller from the system and activating the user or seller in the system.

- **RequestManager:** Creates all requests based on events triggered by the presentation layer and serves to communicate with the server. It waits for a response from the server and reports the results.

4.4 Server (For Admin Web Portal)

The server is where all transactions take place and then sent to the application. An order and the actions to be performed on this order are sent to the server as a request. The server then processes the ordered modules and produces an output. The output is then sent back to the application in response. Server is divided into Logic and Data layers. The logic layer accepts requests and updates/creates/removes data. The logic layer handles database transactions and user/vendor processing. The data layer is where models and other persistent information are kept.


Server
Logic
RequestHandler
Pipeline
AdminPipeline
AdminObject
Data
ActiveUser
ActiveSeller
BlockedUser
BlockedSeller

This defines the subsystem responsible for the logic and operation of the Admin Web Portal. Pipeline and Modules are the two packages that make up these processes.

- **RequestHandler:** Converts client requests to tasks and runs them in the pipeline. Generates responses and sends them back to the administrator.

4.4.1 Pipeline

Admin Web Portal has a modular design structure.

- **AdminPipeline:** After an incoming request, all necessary modules are taken from the Modules package and converted into sequential processes. The pipeline runs these modules one after the other to produce an output.

- **AdminObject:** This is the object created using a request. The pipeline then processes this object; does it to generate the final output note.

4.4.2 Data

This layer is where the data used by the application is kept.

- **ActiveUser**: Contains active user information such as username, phone numbers, address information kept in the NoSQL database.

- **ActiveSeller**: Contains active vendor information such as vendor name, phone numbers, which are kept in the NoSQL database.

- **BlockedUser**: Consists of users blocked by admin.

- **BlockedSeller**: It consists of the information of the vendors blocked by the admin, kept in the NoSQL database.

5. Glossary

Admin: The person who is in charge of the Cakery App.

Cart: A basket where the items are added into.

Customized Cake: A cake which can be customized as buyer's wants and needs.

Dashboard: Home page of Cakery App.

Item: Cakes offered in bakeries.

User: Person who orders from the Cakery App.

My Earnings: Displays total earnings of bakery.

Product: Cakes offered in bakeries.

Profile: Detail of information about the user.

Seller: Bakeries that are enrolled in Cakery application.

6. References:

- 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.
- Resham Shinde, Priyanka Thakare, Neha Dhomne, Sushmita Sarkar, ”Design and Implementation of Digital dining in Restaurants using Android”, International Journal of Advance Research in Computer Science and Management Studies 2014.
- Khairunnisa K., “The application of wireless food ordering system”, MASAUM Journal of Computing, Vol.1 Issue 2, Sept. 2009.
- Patel Krishna, Patel Palak, Raj Nirali, Patel Lalit,” Automated Food Ordering System”, International Journal of Engineering Research and Development (IJERD) 2015
- Varsha Chavan, Priya Jadhav, Snehal Korade, Priyanka Teli, ”Implementing Customizable Online Food Ordering System Using Web Based Application”, International Journal of Innovative Science, Engineering Technology(IJISET) 2015.
- Patel, Mayurkumar, "Online Food Order System for Restaurants" (2015). Technical Library. Paper 219
- Ashutosh Bhargave, Niranjan Jadhav, Apurva Joshi, Prachi Oke, S. R Lahane,“Digital Ordering System for Restaurant Using Android”, International Journal of Scienti?c and Research Publications 2013.
- Asan, N. Badariah in the research paper, “Zigbee-based smart ordering system(S O S)”, International Journal of computer trends and technology (IJCTT) Volume 11 No-5, May 2014
- K.K. Reddy, B. Naresh, “Intelligent E-restaurant using Android OS”, Int. Jr. of Scientific Engg. and tech. research Vol. 3, Issue 22, September 2014, pp: 4383-4385
- Asan, N. Badariah in the research paper, “Zigbee-based smart ordering system(S O S)”, International Journal of computer trends and technology (IJCTT) Volume 11 No-5, May 2014.