Low-Level Design Report

CS 492: Senior Design Project

Toproffer

Doğacan Kaynak

Burak Kırımlı

Mert Çerçiler

Servan Tanaman

Yiğit Kutay Gülben

Supervisor: Halil Altay Güvenir

Jury Members: Hamdi Dibeklioğlu, Varol Akman

Innovation Expert: Burcu Coşkun Şengül

1. Introduction

The state of the Turkish economy has deteriorated for some years. Between these years, inflation has increased and Turkish Lira has been losing its value against foreign currencies, especially the US dollar and Euro.

Restaurants have innovative tactics to inspire potential customers to visit their restaurants. For a specific time period, they bring their low-priced or under-sold products into the promotions.

There is no way to announce their ads to a social platform except for potential customers on social media. So Toproffer came as an idea to help people save their time and save their economy with less effort compared to looking for restaurants on the streets blindly.

Moreover, Related to the low-level design report, the detailed and improved version of the high-level design report is going to be the low-level design study. The scope and validity of the design principles will be explained in detail in this article. The solutions to the design problems will be considered as well as describing the engineering standards. The study also includes trade-offs, descriptions of implementations, bundles and configurations of classes.

1.1 Object design trade-offs

In software development, the people who are the team members have to determine how the functionality in the app will be implemented. This can change everything in the product because some of the features or functions of the product or the portion of the other features need to be sacrificed. The team members rely on tradeoffs so as not to lose the application's spirit. Trade-offs that we considered will be discussed in the following sections.

1.1.1 Performance vs Portability

Toproffer aims at getting the attention of the millions. Portability therefore in the application is important. The chosen team members are Flutter that is cross-platform for Android and IOS. Flutter will create an application for both android and ios at once. Nevertheless, although we try to minimize the performance deficiency, if we compare that Toproffer was only for one platform, there might be a bit of a performance deficiency.

1.1.2 Complexity vs Usability

Toproffer is an application for socialization for people who want to know what the latest campaigns are about multiple places around him or a particular area. Therefore, as students, kids, parents, couples, friends, etc. There may be lots of different users. These different user groups will make easy use of the program. While team members concentrate on usability, in the meantime, complexity is getting higher. Machine learning algorithms and the use of data collected from users make the context a bit complicated and complex while the user interface is simple and easy to use.

1.2 Interface documentation guidelines

In this report all the class interfaces are described as ‘ClassName’ format. The variable and method names of the classes are specified. Some important pages of our system are shown as classes with their methods which are mostly consist of connections between the front-end and back-end part of our system since we produce a mobile application. Also, some important database tables and its attributes are shown as classes in order to show the models of our system. Template of the description of interfaces is shown below.

Class Name: Name of the class
Class Description: Brief description of class
Attributes: Attributes of the class. The format of the attributes are <attribute name>: <data type>
Methods: Methods of the class. The format of the methods are <Return Type> <method name>(<Parameters (if any)>) async (if the method is asynchronous function)

1.3 Engineering standards (e.g., UML and IEEE)

Engineering standards are a type of documents that specifies characteristics and technical details that must be satisfied by the end product, systems, including processes that the standards cover. The application must be stable and that is a very important objective for the sake of this project. We are using different software development environments such as Flutter[3], Firebase[4], and so on.
Our project Toproffer designed to serve an excessive amount of users at the same time with real-time feedback from the system. Potentially many users should be able to use the system with real-time feedback without and glitch. Thus we chose the Cloud Firestore | Firebase Database and since we used Flutter, we followed the Dart Language Specification standards[5]. Besides the ethical concerns that we followed, for the technical part of the application, we drew Toproffer subsystem decomposition structure according to UML[6] standardizations. By this means, we ensure that classes and objects of Toproffer are created based on the Object-Oriented Programming[7] specifications.

1.4 Definitions, acronyms, and abbreviations

- **US:** United States
- **UML:** Unified Modeling Language

- **IEEE:** Institute of Electrical and Electronics Engineers
- **Flutter:** Flutter is Google’s UI toolkit to build mobile, web and desktop applications from a single codebase.[3]
- **Firebase:** Firebase is a mobile and web application development platform developed by Firebase, Inc. [8]
- **Dart:** Dart is a client-optimized language for fast apps on any platform. [9]
- **Object-Oriented Programming:** Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as *methods*). [7]
- **Cloud Firestore:** Cloud Firestore is a NoSQL document database that lets store, sync, and query data for your mobile and web apps. [10]
- **ads:** advertisements

2. Packages

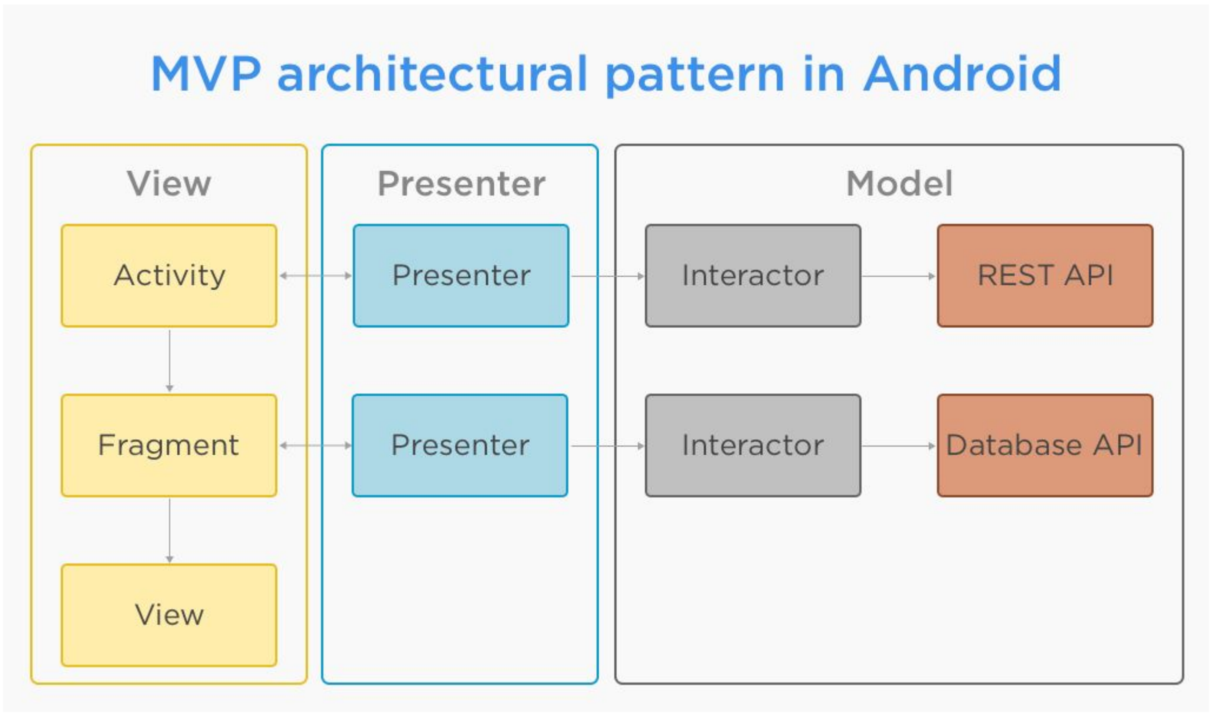
2.1 Dart Packages

Flutter framework uses MVP(Model View Presenter) design pattern. Dart packages contains necessary model, view and presenter files for the Android and IOS application. Due to Flutter Framework, all widgets are set into a default stereotype. This means in both devices it will be compatible.

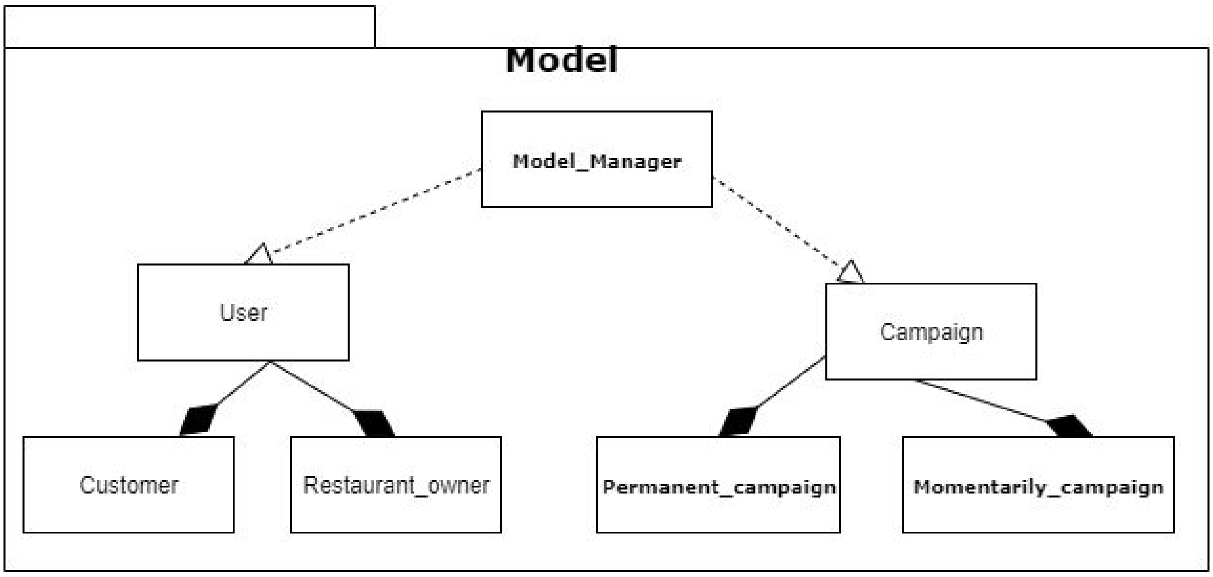
Presenter classes are either Fragment or Activity classes and they extend the corresponding base class from Dart SDK. Each presenter class is interacting with a view.

Model classes represent the data structures that we have in the server.

The packages don’t include detailed methods. The methods of them explained on the 3rd subsection. In Flutter framework classes of view, presenter and modelAlso there is a mapping of design pattern on the below for better understanding.



2.1.1 Model



Model_manager: It is the parent of user and campaign objects.

User: It is parent of customer and Restaurant_owner. It is responsible for modelling customers and restaurant owners in application.

Campaign: It is parent of Permanent_campaign and Momentarily_campaign. It is responsible for modelling campaigns in application.

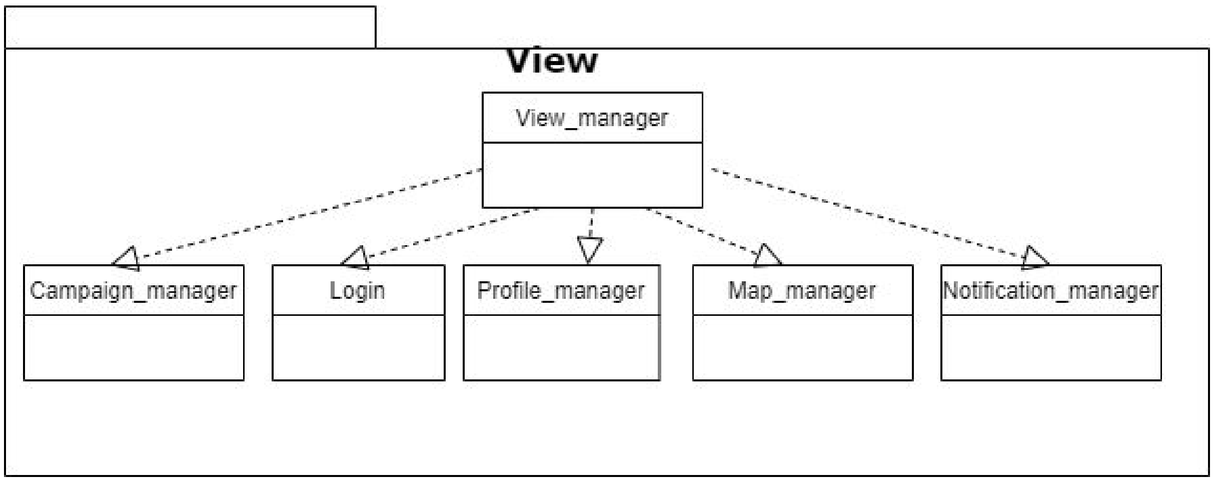
Momentarily_campaign: It is responsible for modelling short-term campaigns.

Permanent_campaign: It is responsible for modelling long-term campaigns.

Restaurant_owner: It is responsible for modelling restaurant owners.

Customer: It is responsible for modelling customers.

2.1.2 View



View_manager: It is the parent of Login, Campaign_manager, Profile_manager, Notification_manager and Map_manager.

Login: It is responsible for setting up the before accessing the home page of application.

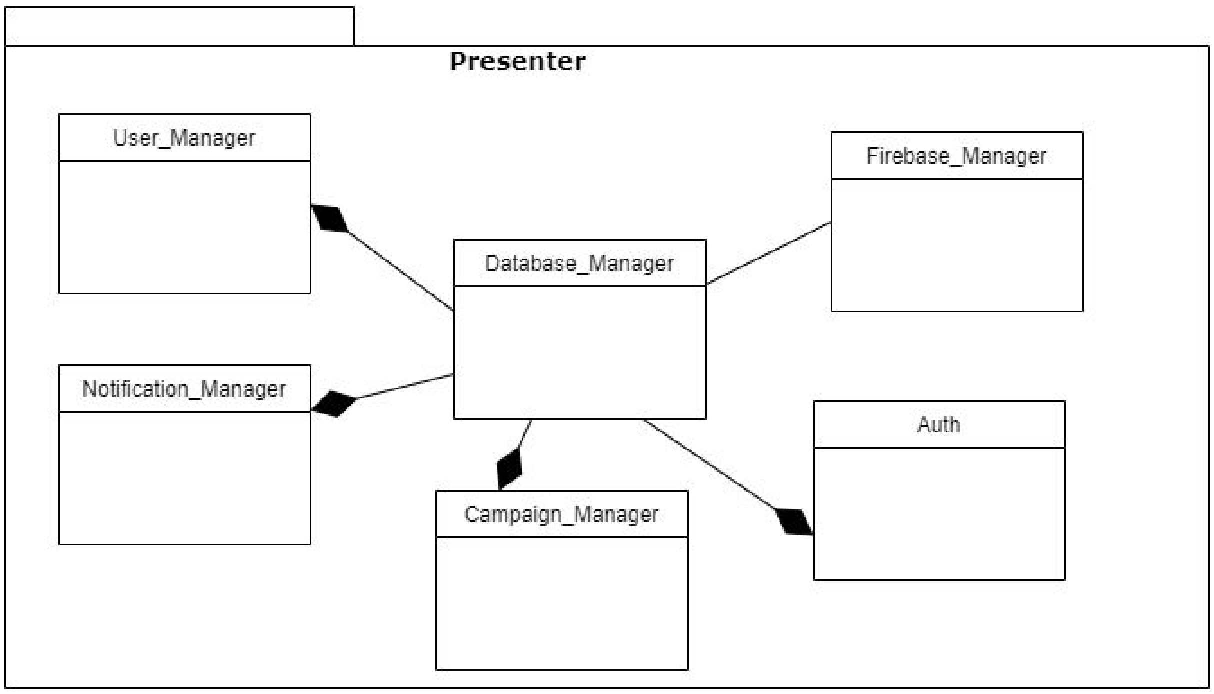
Campaign_manager: It is responsible for the view of momentarily and permanent campaigns.

Profile_manager: It is responsible for the view of users' and restaurants' profile page.

Notification_manager: It is responsible for the view of ordering notifications.

Map_manager: It is responsible for the view of map which you can see campaigns.

2.1.3 Presenter



Database_manager: It is responsible for database access and querying.

Auth: It is responsible for user related tasks in database. Login/Signup and profile updates are done from this class.

Notification_manager: It is responsible for delivering notifications according to user.

Campaign_manager: It is responsible from delivering campaigns to users and recomending campaigns to restaurant owners.

Firebase_manager: It is responsible from building bridge between server and user

User_manager: It is responsible from storing the user’s information.

2.2. Server Packages

Server packages maintain the communication between Android-Client and IOS-Client, Database and Machine Learning packages, they are not allowed to interact with each other without using the server.

pubspec.yaml

2.2.1 Machine Learning Package

This package is stored in the server-side and executed upon a request from restaurant owner. It performs recommendations on campaigns via machine learning algorithm to restaurant owner.

campaign_generator.py

3. Class Interfaces

Class Name: LoginPage
Class Name: This is the class of our login page that either customers or restaurant owners can login to our system via their emails and password combinations, facebook accounts or gmail accounts.
Attributes: instance: Auth username: String password: String auth: FirebaseAuth isLogin: boolean gso: GoogleSignInOptions credential: FacebookAuthProvider
Methods: Future<void> signInWithEmailAndPassword(email, password) async Future<void>firebaseAuthWithGoogle(GoogleSignInAccount acct) async Future<void> sendEmailVerification() async Future<String> signInWithFacebok(String accessToken) async void submitData() Widget build(BuildContext context)

Class Name: CampaignCreator
Class Description: This is the class where restaurant owners create the campaigns for their restaurants.
Attributes: title: String content1: Menu content2: Menu or null content3: Menu or null newPrice: double duration: var

isPermanent: boolean
Methods: void submitData() boolean checkPrice(Menu content1, Menu content2, Menu content3, double newPrice) Widget build(BuildContext context)

Class Name: CampaignConfirmation
Class Description: This is the class where customers verify the code of the campaigns in order to apply the campaigns.
Attributes: code: String isTrue: boolean campaignCounter: int
Methods: void submitData() boolean checkCode() String getCode() Widget build(BuildContext context)

Class Name: RestaurantSettings
Class Description: This is the class where restaurant owners can edit their detail informations about the restaurants.
Attributes: restaurantName: String description: String profilePhoto: Image restaurantPhotos: Image email :String password: String address: String
Methods: void submitData() Widget build(BuildContext context)

Below tables are represents the database tables to indicate models of our system.

Class Name: Customer
Class Description: Customers are users that are able to benefit from the campaigns.
Attributes: user_id: int username: String password: String email: String age: int following_restaurants: Restaurant used_campaigns: Campaign interests: String profile_photo: Image

Class Name: Restaurant
Class Description: Restaurants are users that are able to create campaigns to attract customers.
Attributes: user_id: int username: String password: String email: String followers: Customer address: String active_campaigns: Campaign old_campaigns: Campaign tax_number: int profile_photo: Image restaurant_photos: Image

Class Name: Campaign
Class Description: Campaigns are the main theme of our system that are created by the restaurants to discount desired products and announce to customers.
Attributes: campaign_id: int user_id: int campaign_title: String campaign_content: String campaign_type: String

new_price: int address: String campaign_application_counter: int campaign_codes: Code campaign_duration: int

Class Name: Code
Class Description: This class generates codes to apply campaigns.
Attributes: campaign_id: int user_id: int code_id: String time: Time

4. Glossary

Inflation: A decline in the value of money

Algorithm: A precise step-by-step plan for a computational procedure that begins with an input value and yields an output value in a finite number of steps

US: United States

customer: One who purchases or receives a product or service

ads: Advertisements

portability: *Portability* is a characteristic attributed to a computer program if it can be used in an operating systems other than the one in which it was created without requiring major rework.

usability: Usability is the ease of use and learnability of a human-made object such as a tool or device

complexity: Complexity is a term that includes many properties of a piece of software, all of which affect internal interactions.

campaign: Promotional strategies are those aimed at bringing in more revenue by running deals and discounts, and using ads in restaurants to reach out to customers.

feedback: to convey by means of specialized communications channel

UML: Unified Modeling Language

IEEE: Institute of Electrical and Electronics Engineers

Flutter: Flutter is Google's UI toolkit to build mobile, web and desktop applications from a single codebase.[3]

Firebase: Firebase is a mobile and web application development platform developed by Firebase, Inc. [8]

Dart: Dart is a client-optimized language for fast apps on any platform. [9]

Object-Oriented Programming: Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as *methods*). [7]

Cloud Firestore: Cloud Firestore is a NoSQL document database that lets store, sync, and query data for your mobile and web apps. [10]

5. References

[3] "Flutter - Home Page". https://flutter.dev/ . [Accessed: February 13, 2020]

[4] "Firebase - Home Page". https://firebase.google.com/ . [Accessed: February 14, 2020]

[5] "Dart Language Specification". https://dart.dev/guides/language/spec . [Accessed: February 14, 2020]

[6] "UML – Wikipedia". https://en.wikipedia.org/wiki/Unified_Modeling_Language . [Accessed: February 14, 2020]

[7] "OOP – Wikipedia". https://en.wikipedia.org/wiki/Object-oriented_programming . [Accessed: February 15, 2020]

[8] "Firebase – Wikipedia". https://en.wikipedia.org/wiki/Firebase . [Accessed: February 15, 2020]

[9] "Dart - Home Page". https://dart.dev/ . [Accessed: February 13, 2020]

[10] "Cloud Firestore". https://firebase.google.com/products/firestore . [Accessed: February 14, 2020]