Bilkent University

Department of Computer Engineering

Senior Design Project

Farket

High Level Design Report

Deniz Alkışlar | H. Buğra Aydın | M. Erim Erdal | M. Enes Keleş | Hakan Türkmenoğlu

Supervisor: Eray Tüzün

Jury Members: Mustafa Özdal and Özcan Öztürk

Innovation Expert: Barış Misman

Progress/Final Report

December 31, 2018

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.

Table of Contents

1. Introduction

Finding the best price is always a daunting task. Comparing prices between stores is both time and energy inefficient especially on essential items such as food, cleaning products, and cosmetics. This problem highly affects lives of people with low income including students, homemakers, working class families, especially in countries that have high inflation which results in market price fluctuations nearly on a daily basis. These people have to constantly track prices in order to cut off expenses. To this end they buy from gross markets and online stores, and collect discount coupons. Both of these solutions have downsides, collecting coupons is time and energy consuming and going to a gross market is not the best option in most times which we discuss in the following table.

We conducted a case study on grocery prices in order to see the price difference using a prepared list of most crucial items corresponding to basic needs such as provisions and cleaning products. When purchased from the overall cheapest grocery store, the total cost is 109 Turkish Liras. However when the cheapest option of every item is bought the total is 99 Turkish Liras which is about 10% cheaper than the former *(Table I)*

	Carrefour	**A-101**	**BIM**	**Akakçe**	**Cheapest option selected**
500g Spagetti	2.35 TL	1.15 TL	1.15 TL	3 TL	1.15
1 kg potatoes	2.79 TL	*	2.73 TL	*	2.73
1 kg tomatoes	6.95 TL	*	6.95 TL	*	6.95
1 kg onions	2.79 TL	*	2.98 TL	*	2.79
15 eggs	11.99 TL	8 TL	8 TL	10 TL	8
1 lt milk	2.75 TL	3.25 TL	3.25 TL	*	2.75

Price / diaper**	0.67 TL	0.36 TL	0.365 TL	0.342 TL	0.342
32x toilet papers**	31.90 TL	31 TL	31 TL	24 TL	24
Detergent / KG**	5.28 TL	4.18 TL	3.75 TL	*	3.75
Disher Capsule**	1.01 TL	0.79 TL	0.35 TL	0.83 TL	0.35
-TOTAL-	**169 TL**	**131 TL**	**107 TL**	**128 TL**	**99 TL**

Table 1. Price comparison between different markets [on 30th of October 2018]
*note that missing prices are plugged with the price of the cheapest option
**prices are normalized for quantities

1.1 Purpose of the system

Purpose of the system is to give Turkish citizens a way to efficiently and practically cope with the current extreme price fluctuations between markets born due to the unforeseen and high inflation rates. There is already a campaign called “Enflasyonla Topyekün Mücadele” which is run by government to ensure that prices are kept low and available for citizens. This application aims to help this campaign and help its users to be able to find and select the cheapest options among different markets without a tiring process.

1.2 Design goals

1.2.1 Reliability

Application has to be reliable on any boundary conditions. User will be informed by error messages and warnings as pop ups if a boundary condition occurs or if anything is not going right. On boundary conditions that cause immediate exit of the program, like running out of charge, program will be able to save the current state of the application, for example save the state of shopping list and when the application is reopened, user will be able to continue adding to that application.

1.2.2 Maintainability

Application should be able to be maintained by owners regularly. Program will be able to be quickly and easily recovered to operational status after a failure occurs. This will be achieved by closing application regularly and doing patching.

1.2.3 Scalability

Application will be scalable, being able to serve up to hundreds of thousand users at the same time. This will be achieved by using reliable and scalable backend databases such as MySQL.

1.2.4 User Friendliness

Application will have simple interface in order to make it easy to navigate in the application. By doing that, user makes interactions with the app easier, and total time required will decrease to understand the concept of the application.

1.2.5 Portability

Application should be portable and run on some different platforms. This will be achieved by the usage of React Native. Application will run on web, Android and iOS.

1.2.6 Supportability

The system should be adaptable to the future updates in data sources, APIs, and platforms. It is very important for the system to be flexible to the changes in these platform.

1.3 Definitions, acronyms, and abbreviations

API: Application Programming Interface
DB: Database
GUI: Graphical User Interface
UI: User Interface
SDK: Standard Development Kit, tools and libraries

1.4 Overview

Farkett is a mobile application that targets grocery shopping customers. It can be considered as a platform to help consumers buy the cheapest available products in the market. With the help of Farkett, consumers will be available to create a shopping list and find the cheapest form of that items in the list among many online stores, which will help them in their decision making process and potentially save money. This feature will not require registration of users.

Among the use cases of the program, user will be able to create a shopping list for both searching the cheapest of those items and for remembering purposes. After searching for the cheapest of these products, application will create a shopping route depending on the current location of the user and the locations of nearby stores, showing the user the cheapest way to buy these items with the minimal amount of travel between stores. User will be able to modify the total number of stores to visit thereby reducing the travel cost. Then the program will show the cheapest total of these products from a single store.

Users will be able to select if they want a specific brand if they prefer or not. If they do not prefer a specific brand for a product then they will be offered alternative, cheaper brands for the same product which the user can prefer to reduce the price of the same product. However, user will be able to lock in specific brands for product, Coca Cola would be an example of a brand lock for a cola purchase.

Users also will be able to scan their receipts to gain awards. This will require login in order to save their shopping preferences and awards gathered. By scanning their receipts, user will be able to both gain more personalized discount offers and awards which could be withdrawn after a certain amount of points are accumulated.

2. Current Software Architecture

2.1 cimri/akakce

Website and mobile app which compares prices among many online stores and lists the best options for the user. It compares for one product at a time. It is only available in Turkey [1].

2.2 Basket

Mobile app which shows consumer prices in grocery stores. The application gathers data by crowdsourcing, namely its users entering prices of products into the system and validating these prices by scanning receipt barcodes. Only available in the US [2].

2.3 Grocery Pal

Mobile app that points the user towards weekly sales at local supermarkets and discount stores, and matches shopping list items against products online with lower prices. Only available in the US [3].

2.4 GroceryIQ

A mobile app application that allows the user to build shopping lists with features like predictive search and barcode scanning. Only available in the US [4].

2.5 Bring!

Globally available grocery shopping list app allows users to create, sync, and share shopping lists [5].

2.6 Favado

Mobile app showing sales and coupons on nearby stores. The user can collect coupons by emailing to themselves and printing out. Only available in the US [6].

2.7 Google Shopping

Google Shopping is an online search engine for shopping. It displays prices on online stores and allows user to filter products depending on seller and price. It is globally available [7].

3. Proposed Software Architecture

3.1 Overview

Farkett is a cross platform mobile application which shows up-to-date grocery prices and recommends optimal routes for shopping lists. This task requires grocery price data, the majority of which will be collected by scraping online stores. However some grocery stores don’t have online stores. Therefore we have another data collection method that is receipt scanning (Table 2). In this method receipt information is recognized using OCR and consumer prices are extracted. To encourage users to scan receipts we consider giving rewards which will be determined later in the process. We aim to keep personalized receipt information in order to offer targeted advertisement.

	Carrefour	Migros	A-101	BIM	ŞOK
Has Online Store?	+	+	+	-	-
Has Web API?	-	-	-	-	-
Is web scraping possible?	+	+	+	-	-
Is receipt scanning possible?	+	+	+	+	+

Table 2. Available data collection methods for grocery stores.

The novel feature in our product is shopping route recommendation. Farkett scans prices in nearby grocery stores and offers optimized routes for shopping lists with criterias determined by user. Shopping recommendation is possible for online stores also. In this case, Farkett tells user what items to buy from which online store.

Finding optimized shopping routes is not an easy task because sometimes the cheapest route is not the most desirable. Most people don’t want to walk kilometers and visit many stores. Therefore Farkett takes distance, prices, and number of stores into account while offering routes. Furthermore, upon entering a store in the route Farkett will show the items to buy in that store, for convenience. Shopping route recommendation will be customizable to user’s will, namely range and product criterias can be selected. This makes Farkett a useful tool for many people.

Below is a table showing the comparison of Farkett with currently available applications in terms of features that are shopping list creation, shopping route recommendation, receipt scanning for data collection, barcode scanning to get price of a product, coupon giveaways, price and discount information, and having online stores included (Table 3).

	Basket	GroceryPal	Favado	Proposed System
Create a Shopping List for the customer	+	+	+	+
Finding the cheapest price for a given item	+	+	+	+
Create optimized shopping routes	-	-	-	+
Receipt Scanning	-	-	-	+
Barcode Scanning	+	-	-	-
Offering Coupons	-	+	+	-
Offering Discounts on selected items	+	+	+	+
Local Stores	+	+	+	+
Online Stores	+	-	-	+

Table 3. Feature comparison of proposed system with currently available alternatives

3.2 Subsystem decomposition

We decided to use a three-tier architectural design which allows subsystems to be upgraded independently which allows low amount of coupling. As a result, we separated Farkett implementation into subsystems that are named Data, Logic, and Presentation. Data and Logic subsystems are included in the server-side whereas Presentation subsystem is in the client-side. Below is a diagram showing the complete high level structure (Figure I).

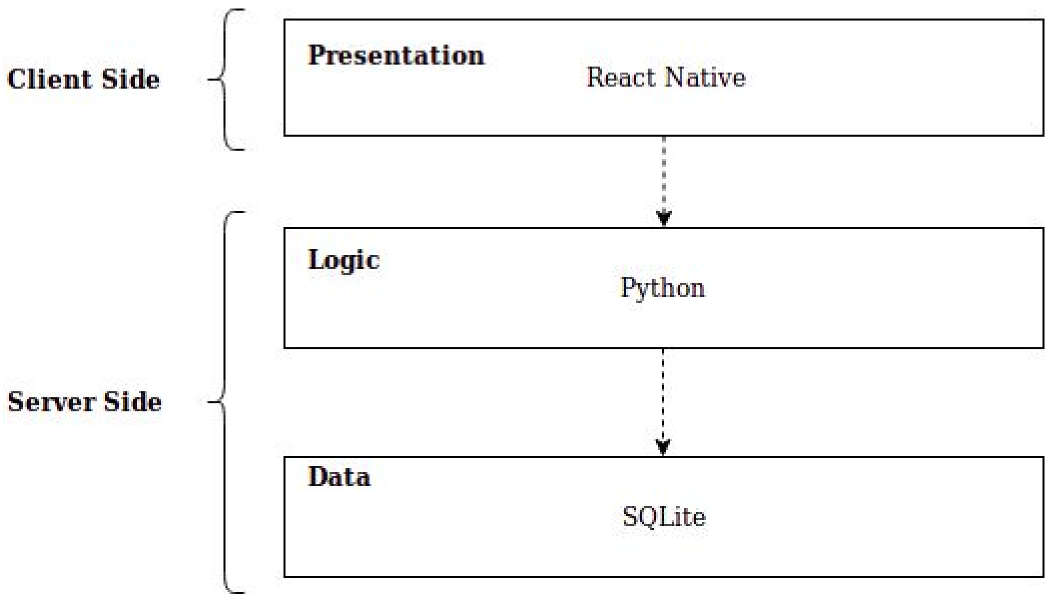

Figure 1. Three-tier architecture of the system

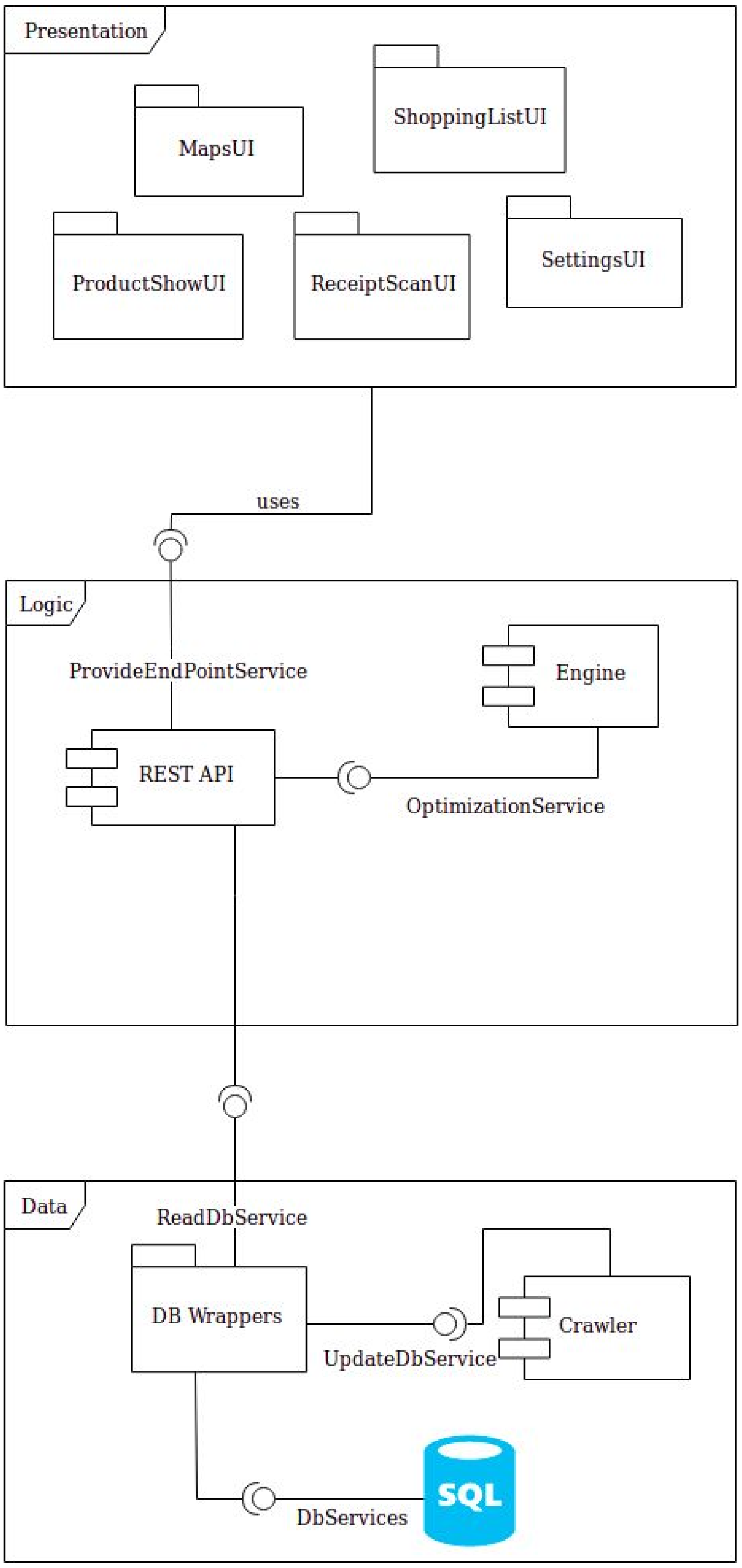

Figure 2. Subsystem decomposition of three-tier architecture

3.2.1 Data Subsystem

Data subsystem consists of the actual database, crawler, and wrappers package. This subsystem’s main purpose is to abstract out database operations by providing wrapper methods which cover every functionality that is needed by Logic subsystem and the crawler module. Crawler regularly updates the database by inserting newly fetched information from online market websites using wrapper methods. Note that wrappers package will be segmented into table-oriented modules i.e. there will be a module for every table in the database that contains functionality regarding that table.

3.2.2 Logic Subsystem

Logic subsystem is responsible of meaningfully manipulating and sending the data to client-side namely Presentation subsystem. It consists of REST API module and Engine module. Engine module is a module that will provide the functionality which is hard to achieve by using SQL for example optimization methods for shopping routes and search. REST API module contains HTTP endpoints which will receive client-side requests.

3.2.3 Presentation Subsystem

Presentation subsystem will render the application interface on mobile phone screen while interacting with the server via HTTP. It consists of many packages corresponding to screens including MapsUI for seeing optimized shopping routes on map, ProductShowUI for browsing products in physical stores and online stores, ReceiptScanUI for receipt scanning, ShoppingListUI for shopping list creation, and SettingsUI.

3.3 Hardware/software mapping

Our application will be developed for mobile so it will run on both the Android and iOS operating systems. In order to use the receipt scanning feature of the Farkett, users needs a mobile phone with camera. They also need GPS for the optimal shopping route construction. Internet access is crucial for communication with the server. A MySQL server is needed to serve our users. Our backend will be developed using Python. Frontend for both Android and iOS will mostly share the same code since React Native framework will be used for a cross platform app.

Below is the deployment diagram of our app:

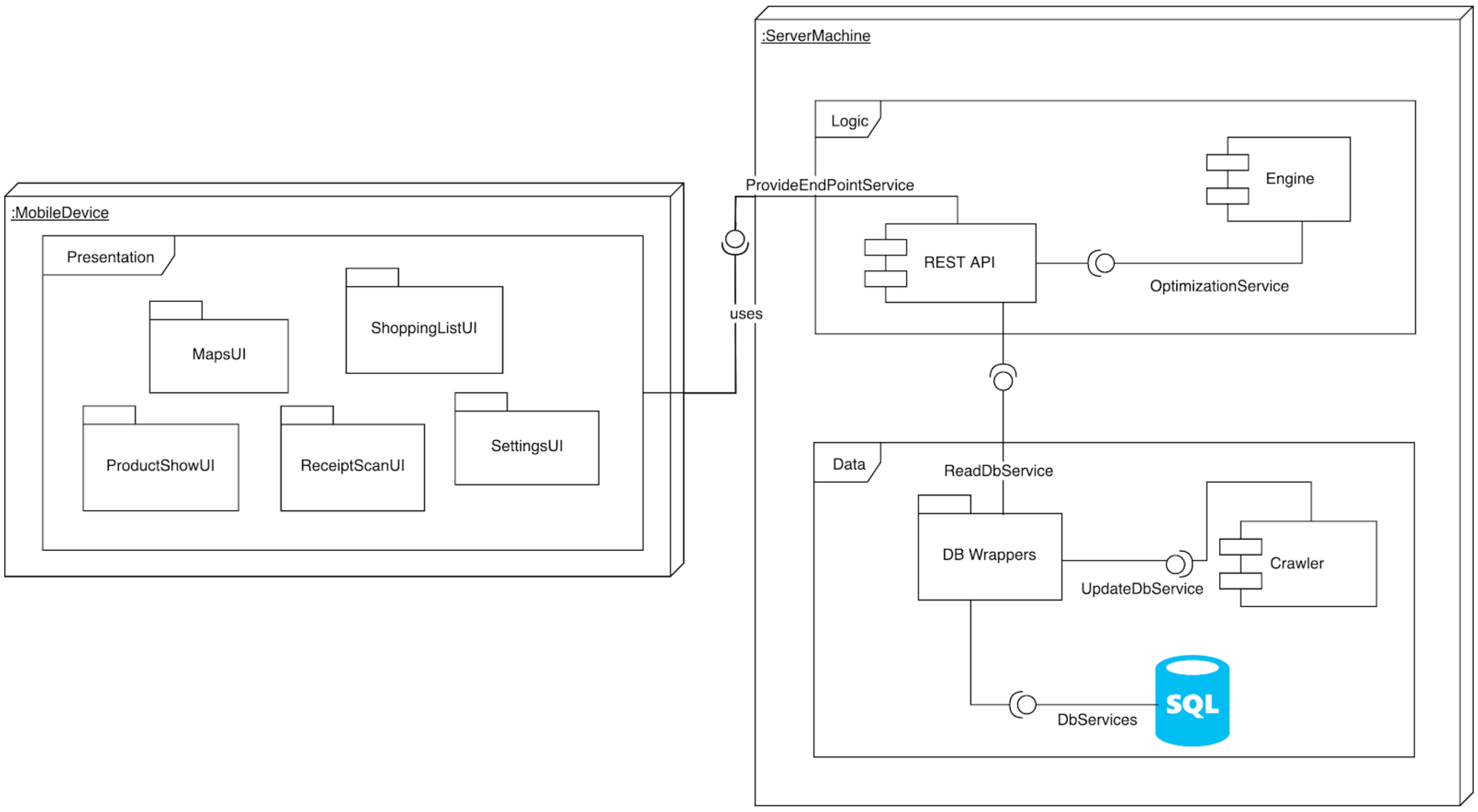

Figure 3. Deployment diagram of Farkett

3.4 Persistent data management

Farkett is heavily dependent on a persistent database, since the whole program needs legit and consistent data to work with concurrent readers and writers. Database will be automatically updated in backend, after the daily/hourly information is sent from the crawler. Keeping track of the updated prices, amounts etc. of the products will be ensured this way. The database will have a backup in each update, in case of any unwanted data loss or error. The database update times will be as fast of possible, since any change of information during a exchange may cause unintended consequences. We will try to implement fast queries and algorithms to obtain efficiency.

Consequently, database performance is crucial for the program and this important aspect will be obtained in our final product.

Access will be rate limited for users, expectedly 5000 read requests per day.

3.5 Access control and security

In order to use the program, user will need to authenticate with his/her Google account. This will prevent any users to perform any unwanted attacks like DDoSing the server with multiple accounts. It will also ensure that no automated programs will be ran on the server to obtain and steal the dataset.

Google's authentication system has advanced authentication system with different captchas to ensure that the user is an actual human. Also, we will be obtaining a token from the user in each request to the database. After a limited time, tokens expires. If a token is expired, the response from the database is not given. This will prevent any unwanted users company to obtain our data.

We will be checking the SQL queries sent in the client side, since there might be attempts of SQL injections. The addresses and other important information about the user will be encrypted with cryptographically secure AES256 algorithm to prevent any unwanted consequences. Additionally, passwords will be hashed with bcrypt algorithm.

The following access control matrix also describes the capabilities of admins and regular users in our system:

	Read Price Data	Update Price Data	Ban User	Add/Delete Store	Read Error Log
User	Rate limited	Partial control*	-	-	-
Admin	Unlimited	Full control	+	+	+

* With receipt scanning feature.

Table 4. Access control matrix of Farkett

3.6 Global software control

The communication between client and server shall be event-driven in order to avoid race conditions that can occur when multiple users request data simultaneously. That means price data will have centralized control by the server. Exception to this is the receipt scanning feature. On the client-side will verify that the photo user have taken indeed belongs to a receipt. When the receipt is uploaded to the server, the receipt will be further analyzed and double-checked. Since the first verification happens on the client-side, both decentralized control and centralized control are used for receipt objects.

3.7 Boundary conditions

3.7.1 Start-up

- The app will be downloaded from the Google Play/App Store
- The app will be automatically initialized by the operating system itself
- When user starts the app, the app will start running and will welcome the user
- The user will need to authenticate to his/her Google account
- The app will ask the user if he/she wants the app to remember the login information for future use

- The app will work on latest versions of the operating systems and older versions might cause errors

3.7.2 Shutdown

- After the authentication, user will be redirected to the homepage.
- The program can be shutdown with the default terminate operations in Android and iOS devices.
- User can also logout.

3.7.3 Exception Handling

- Even if we will try our bests to reduce the amounts of errors as much as possible, if an error occurs during the use of the application, the system will provide user an explanatory error message. The user will be able to send an error report to the admins and the error will be analyzed from the error log. After the analyzation, the admins will work on the prevention of any future errors.

4. Subsystem services

In this section we will explain services provided by the server and client side of the application in more detail. Descriptions of Data, Logic, and Presentation subsystems can be found at section 3.2. For reader to have a better understanding on these services, we include a template of database tables which will later be explained in more detail in low level design report (Figure 4).

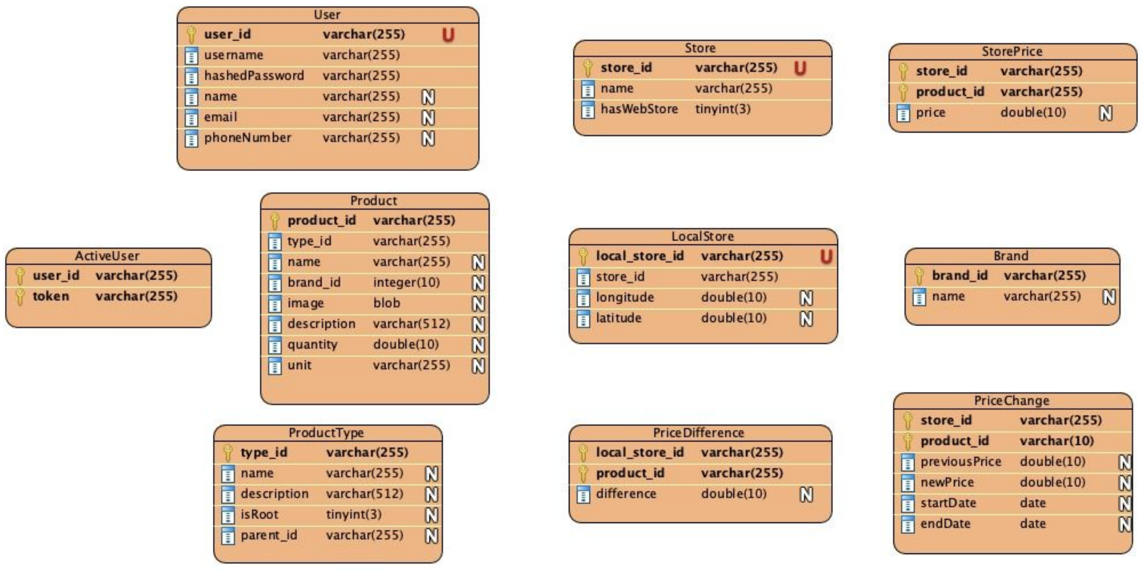

Figure 4. Tables in the Database

4.1 Server-side services

Our server provides an object based REST API meaning that every endpoint in the API corresponds to a table in the database (Figure 4). API endpoints will accept POST requests with JSON formatted payloads and will respond with JSON. Request payloads will include criterias that will be used to filter out objects by the server. For instance in order to receive Coca-Cola products in Migros, the client side will be have to send an HTTP request with brand name Coca-Cola and store name Migros so that the server response will contain only filtered products.

Authentication will be handled by OAuth2 protocol which enables Sign-in with Google feature.

4.2 Client-side services

Our client will have an event-driven asynchronous HTTP request service that will fetch the data from the server. It will be responsible from building the required request from the payloads and options given by the user. Response data will be processed and stored in a Redux store as a single source of truth, mirroring partial server data in the memory.

To display the data in a meaningful way to the users, we will use React components. These components are self-contained, meaning they don’t have coupling to the data. Because of this, it’s possible to test the user interface with mock data. This allows test-driven development practices and fast prototype implementations for new user interfaces.

Maps functionality will be handled by Google Places API and Google Maps SDK.

In order to provide a good user experience for the receipt scanning feature, we will serve an OpenCV based receipt verification system on the client side. To understand why we need such dependency, we must compare the alternative possible solutions.

The first option is handling the responsibility of taking a good shot to the user. We can ask for user to take a photo of the receipt. Problems arise if the photo capture is blurry or the user misses the receipt inside the frame. This faulty photo then will be sent to the server, which takes time. After that we can display an error and ask user to take another shot. Additional time here will be wasted because all mobile devices have an opening lag for the camera due to the hardware limitations. This can repeat over and over again which is a potential hazard for user frustration.

The second option is streaming the video input of the camera to the server. This way we can differentiate a good shot from the bad ones and select the best frame to analyze the receipt. There are a lot of things that can go wrong with this approach though. It is not suitable for users with mobile internet which have tight quotas. Even if the user has enough quota, handling network connectivity problems is a hard task. Additionally uploading this data over the network may take a lot of time. Finally, even if we solve all the network problems, processing all these data at scale is a difficult task.

The third and our preferred option is to have partial verification on the client side and the full verification on the server side, which is a hybrid of the first and the second option. Conceptually second option gives the preferred user experience at the expense of technical problems. First option has no technical difficulties but provide bad user experience. Instead of sending the video input to the server, we will process it on the client-side to give user important feedbacks like whether the camera is blurry or is it possible to read the texts inside the receipt. If the client detects a good candidate frame of image inside the video input, it will send this to the server to process it further and extract relevant task. If the server rejects this image for some reason, the client will continue recording and repeat the process until server accepts a receipt image.

We aim to provide similar user experience to a QR code app in the eyes of the user. This way, our app can be user friendly without facing much technical difficulties.

5. Glossary

React Native: A framework to build mobile apps using only JavaScript. It uses the same design as React, letting developers to compose a rich mobile UI from declarative components. [8]

OCR: API that converts PDF of Word or Image to text [9]

MySQL: MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more [10].

SQL Injection: SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input. [11]

HTTP: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen. HTTP was developed to facilitate hypertext and the World Wide Web. [12]

JSON: JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. [13]

REST API: Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating web services. Web services that conform to the REST architectural style, termed *RESTful* web services, provide interoperability between computer systems on the Internet. RESTful web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. Other kinds of web services, such as SOAP web services, expose their own arbitrary sets of operations. [14]

AES256: The Advanced Encryption Standard (AES), is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001 [15]. The number 256 denotes the key length of the AES algorithm.

Bcrypt: bcrypt is a password hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher, and presented at USENIX in 1999. Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power [16].

Redux: Redux is a predictable state container for JavaScript apps. It allows writing applications that behave consistently, run in different environments (client, server, and native), and are easy to test. Also it provides a great developer experience, such as live code editing combined with a time traveling debugger [17].

OpenCV: OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license [18].

6. References

[1] cimri.com.(2018) HomePage. [Online] Available at: https://www.cimri.com/ [Accessed 11 November. 2018].

[2] Basket. (2018) HomePage. [Online] Available at: https://basket.com/ [Accessed 11 November. 2018].

[3] Grocery Pal. (2018) on Play Store. [Online] Available at: https://play.google.com/store/apps/details?id=com.twicular.grocerypal.android [Accessed 11 November. 2018].

[4] Grocery IQ. (2018) HomePage. [Online] Available at: http://www.groceryiq.com/ [Accessed 11 November. 2018].

[5] Bring!. (2018) HomePage. [Online] Available at:https://getbring.com/#!/app [Accessed 11 November. 2018].

[6] Favado. (2018) on Play Store. [Online] Available at:https://play.google.com/store/apps/details?id=com.savings.android.grocery&hl=tr [Accessed 11 November. 2018].

[7] Google Shopping. (2018) HomePage. [Online] Available at: https://www.google.com/shopping?hl=tr [Accessed 15 December. 2018].

[8] React Native. (2018) HomePage. [Online] Available at : https://facebook.github.io/react-native/ [Accessed 15 December. 2018].

[9] OCR. (2018) Online Service. [Online] Available at : https://www.onlineocr.net/ [Accessed 15 December. 2018].

[10] About MySQL. (2018) HomePage. [Online] Available at : https://www.mysql.com/about/ [Accessed 15 December. 2018].

[11] SQL Injection. (2018) on W3Schools. [Online] Available at : https://www.w3schools.com/sql/sql_injection.asp [Accessed 15 December. 2018].

[12] HTTP. (2018) on Wikipedia. [Online] Available at : https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol [Accessed 15 December. 2018].

[13] JSON. (2018) HomePage. [Online] Available at: https://www.json.org/ [Accessed 15 December. 2018].

[14] REST. (2018) on Wikipedia. [Online] Available at : https://en.wikipedia.org/wiki/Representational_state_transfer [Accessed 15 December. 2018].

[15] Advanced Encryption Standard. (2018) on Wikipedia. [Online] Available at: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard [Accessed 15 December. 2018].

[16] bcrypt. (2018) on Wikipedia. [Online] Available at: https://en.wikipedia.org/wiki/Bcrypt [Accessed 15 December. 2018].

[17] reduxjs/Predictable state container for JavaScript apps. (2018) on Github. [Online] Available at: https://github.com/reduxjs/redux [Accessed 15 December. 2018].

[18] OpenCV. (2018) on Wikipedia. [Online] Available at: https://en.wikipedia.org/wiki/OpenCV [Accessed 15 December. 2018].