TED UNIVERSITY

Department of Computer Engineering

2023 Spring

CMPE 492 Senior Project II

Low-Level Design Report

Name of the Project: Receipt Analyzer

The URL of webpage: https://bzelihagunay.wixsite.com/receipt-analyzer

Team Members: Ahmet Berat Akdoğan, Başar Aslan, Burçak Zeliha Günay, Hüseyin Hikmet Fındık

Supervisor: Aslı Gençtav

Jury Members: Gökçe Nur Yılmaz, Venera Adanova

Date of Submission: 27/03/2023

Table of Contents

1. Introduction

In today’s world, increasing population, raising production, demand, and variety of products causes product range to increase rapidly. While the number of different products raises, the ability to follow these microeconomic symbols gets more confusing every day. While these symbols are affecting all of our daily lives, almost all parts of the world suffer from not knowing or analyzing enough of these numbers on products. Also, considering the changing inflation in some parts of the world including our country, the prices of various products change frequently. Considering these facts, it becomes more difficult to observe these changes for a person that constantly goes to markets to meet their needs in daily life. While small parts of consumers try to check their receipts regularly, most of the consumers just throw them into the trash without even looking at their receipts. Those consumers who keep their receipts can also barely observe and analyze how much the price of a specific product has changed in the short-long term.

Nowadays, although some projects on easing the tracking of product prices are popular, there is a need for a project that focuses on personal receipt tracking and analysis. A mobile application created based on the market products from the scanned data is not a quite common idea. In addition, our project plans to analyze the information about products by using visual graphics and charts after the market product receipts are scanned. The implementation of this idea distinguishes our project from other consumer-friendly software ideas in the industry.

To summarize, our project will be a mobile application that will help consumers to observe the price changes of products by using OCR technology. OCR technology, which stands for “Optical Character Recognition”, converts images into text and gets the necessary data from the text. With the help of our mobile application, customers will be able to scan the receipt they purchased from the market by OCR technology and observe the price change of products they desire.

This report provides an overview of the design and development of our Receipt Analyzer application. The report provides detailed information on the tools, technologies, and methods used in the design and development of the application. The first section begins with an explanation of the decisions made in the design of the application and how those decisions were effective. This section compares design goals such as usability, portability, security, readability, and efficiency.

Other design factors such as interface documentation guidelines and engineering standards are then addressed. The second section describes the package structure of the application. The third section describes the modeling tools used to create the models of the application, such as class diagrams, UML diagrams. The fourth section presents a list of technical terms, abbreviations, and definitions used in the design of the application. This report provides a detailed look at the design and development process of the application. It can be a useful resource for anyone who wants to learn about the technologies and methods used in developing the application.

1.1 Object Design Trade-offs

In the current project, many object design trade-offs were made. In this section, we will discuss the different design options evaluated under these decisions and provide information on the advantages and disadvantages of each. We will detail these comparisons under the following subsections.

1.1.1 Usability vs Portability

In the project design, one of the trade-off decisions was made between usability and portability. To ensure that the application runs on as many devices as possible, many of the features used or planned to use are embedded in the Android platform. However, using customized features for Android can reduce the portability of the application. While considering the advantages of using other operating systems, for the earlier stages of our development, it is important for the application to reach as wide an audience as possible and be usable. Therefore, in the project design, Android-specific features were used to increase usability.

1.1.2 Flexibility vs Performance

In the design phase of the project, a trade-off decision was made between flexibility and performance. To increase the flexibility of our application, a modular approach was adopted, and code reusability was increased. However, in terms of performance, each module had to have a modular structure to work independently. Thus, the application performance will be optimized to provide consistency among users.

1.1.3 Ease of Use vs Comprehensive Features

In the project design phase, a trade-off decision was made between ease of use and comprehensive features. To make it easier for users, the application interface was planned to design in a simple manner and be as understandable as possible. However, the application was also equipped with comprehensive features to provide users with a wide range of options including different analyses of different data. These features will allow users to have a more comprehensive experience.

1.1.4 Readability vs Efficiency

This trade-off expresses the balance between the readability of the code and the performance of the code. Readable code reduces the likelihood of errors during the development, maintenance, and testing phases. However, in some cases, the readability of the code can affect performance. Thus, our project design aims to balance the readability of the code with its efficiency.

1.1.5 Reusability vs. Performance

This trade-off refers to the balance between the reusability of the code and its performance. Reusable code reduces the need for writing more code and speeds up the development process. However, in some cases, reusable code may negatively affect performance. In this project, performance is a priority factor. Optimizing the code and making it run fast will provide users with a better experience.

1.1.6 Reliability vs. Flexibility

This trade-off refers to the balance between the reliability of the code and its flexibility. Reliable code ensures that the application runs smoothly and without errors. However, in some cases, reliability may come at the cost of flexibility. In this project, reliability is a priority factor. Ensuring that the application runs smoothly and safely will allow users to use the application safely.

1.2 Interface Documentation Guidelines

Class Name (File Type Extension)
Class descriptions
Attributes
Attribute lists
Methods
Method list and descriptions

Table 1: Interface Documentation Guideline

For this project, the interface documentation must meet specific design objectives. These design objectives are:

1.2.1 Usability

Interface documents should help users interact with the application. All necessary functions required for users to perform tasks and communicate with the application should be clearly defined. Additionally, the interface documentation should include sample screenshots to assist users in navigating the application.

1.2.2 Consistency

Interface documents should provide a consistent appearance across all components of the application. This means that components with the same functionality should be described in the same way. This allows users to easily navigate the application and transition between components with different functionalities.

1.2.3 Accuracy

Interface documentation should accurately reflect the real functionality of the application. To prevent users from having incorrect expectations about the application, all information in the interface documents must be accurate and up to date.

1.2.4 Innovation

Interface documents should reflect the innovative functionality of the application. Describing the unique features of the application shows users how it differs from others and why it should be preferred.

1.2.5 Easy Access

Interface documentation should provide users with easy access to the necessary information. Descriptions of components and sample screenshots help users quickly learn about the application.

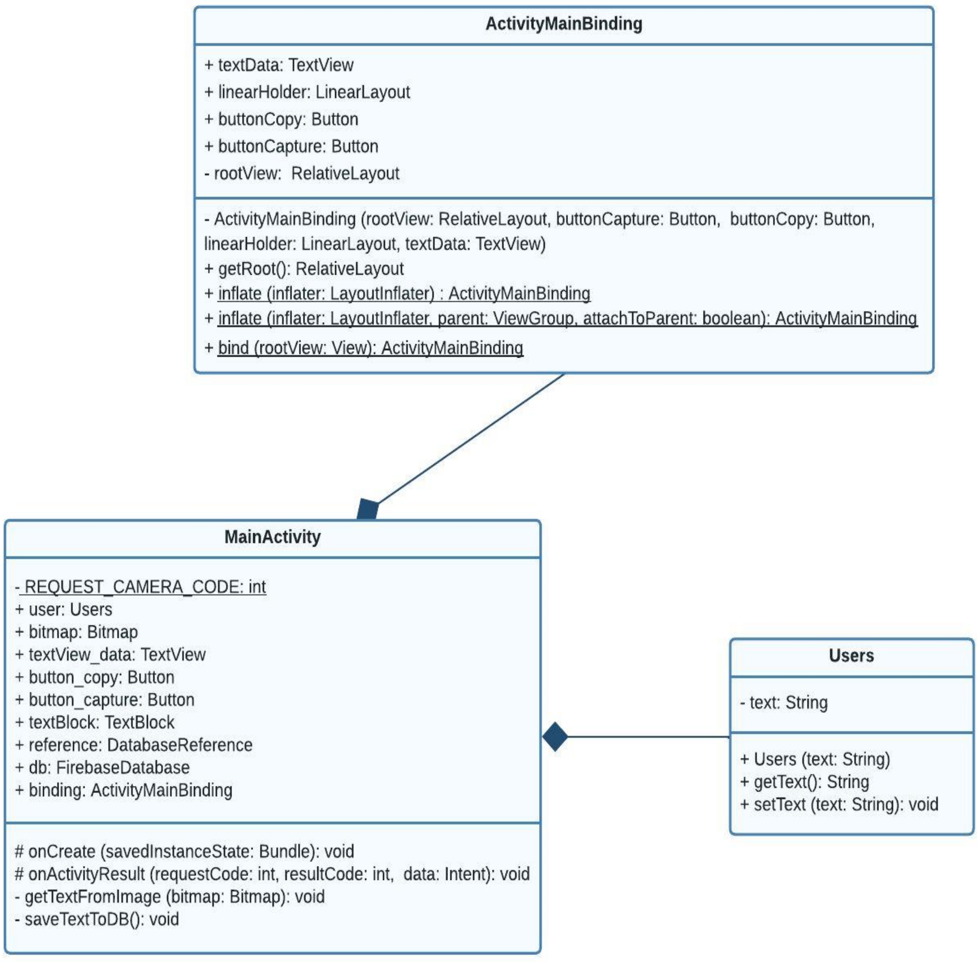
1.2.6 Flexibility

Interface documents should enable the application to work on different devices and in different sizes. This requires the interface documentation to be organized according to different screen sizes and resolutions. Additionally, factors such as different operating systems should also be considered.

Taking into account the design trade-offs we mentioned in 1.1, we can also focus on the same goals in interface documentation guidelines. For example, for usability and portability goals, it should not be forgotten that the interface documentation should be easy to use, understandable, and well-structured. In addition, it is important for the documentation to be organized in a way that can work on multiple platforms. For readability and efficiency goals, it should not be forgotten that the interface documentation should be both understandable and detailed. Good interface documentation should clearly explain to users what they are doing, why they are doing it, and how they will do it. However, overly detailed documentation can prevent users from quickly finding the information they need. In addition, for the reliability goal, it is important that the interface documentation is accurate and up to date. Documentation should always be consistent with the current version of the software and provide users with accurate information. Finally, for the customizability goal, the interface documentation should provide users with the information they need to customize different features of the software. The ability for users to configure the software to their needs demonstrates that the software is user-friendly and improves the user experience. Taking all of these goals into account, interface documentation can help improve the overall usability, performance, reliability, and customizability of the software.

1.3 Engineering Standards

To design and develop the data models of the Receipt Analyzer application, several modeling tools were used, including class diagrams, UML diagrams. These diagrams provide a visual representation of the structure, behavior, and interactions of the different components of the application. The class diagrams define the data structures of the application, including the relationships between different classes and their attributes and methods.


ActivityMainBinding
+ textData: TextView
+ linearHolder: LinearLayout
+ buttonCopy: Button
+ buttonCapture: Button
- rootView: RelativeLayout
- ActivityMainBinding (rootView: RelativeLayout, buttonCapture: Button, buttonCopy: Button, linearHolder: LinearLayout, textData: TextView)
+ getRoot(): RelativeLayout
+ inflate (inflater: LayoutInflater) : ActivityMainBinding
+ inflate (inflater: LayoutInflater, parent: ViewGroup, attachToParent: boolean): ActivityMainBinding
+ bind (rootView: View): ActivityMainBinding
MainActivity
- REQUEST_CAMERA_CODE: int
+ user: Users
+ bitmap: Bitmap
+ textView_data: TextView
+ button_copy: Button
+ button_capture: Button
+ textBlock: TextBlock
+ reference: DatabaseReference
+ db: FirebaseDatabase
+ binding: ActivityMainBinding
# onCreate (savedInstanceState: Bundle): void
# onActivityResult (requestCode: int, resultCode: int, data: Intent): void
- getTextFromImage (bitmap: Bitmap): void
- saveTextToDB(): void
Users
- text: String
+ Users (text: String)
+ getText(): String
+ setText (text: String): void

UML Diagram of Java classes for Android based mobile application project.

1.4 Definitions, Acronyms, and Abbreviations

- OCR: Optical Character Recognition
- API: Application Programming Interface
- UI: User Interface
- SDK: Software Development Kit
- IDE: Integrated Development Environment
- XML: Extensible Markup Language
- UML: Unified Modeling Language

1.5 Testing Approach

The testing approach that will be used for this project is based on defining test scenarios in the early stages of the development process and conducting tests as early as possible. This approach allows for errors to be detected and corrected earlier, as well as providing feedback as soon as possible. Test scenarios are initially written and updated at each stage of the development process. These scenarios are and will be prepared to verify the functionality that users expect. The testing approach includes both manual and automated tests. Manual tests are used for scenarios related to user interface and user interaction, while automated tests are used for remaining functionality and data verification tests.

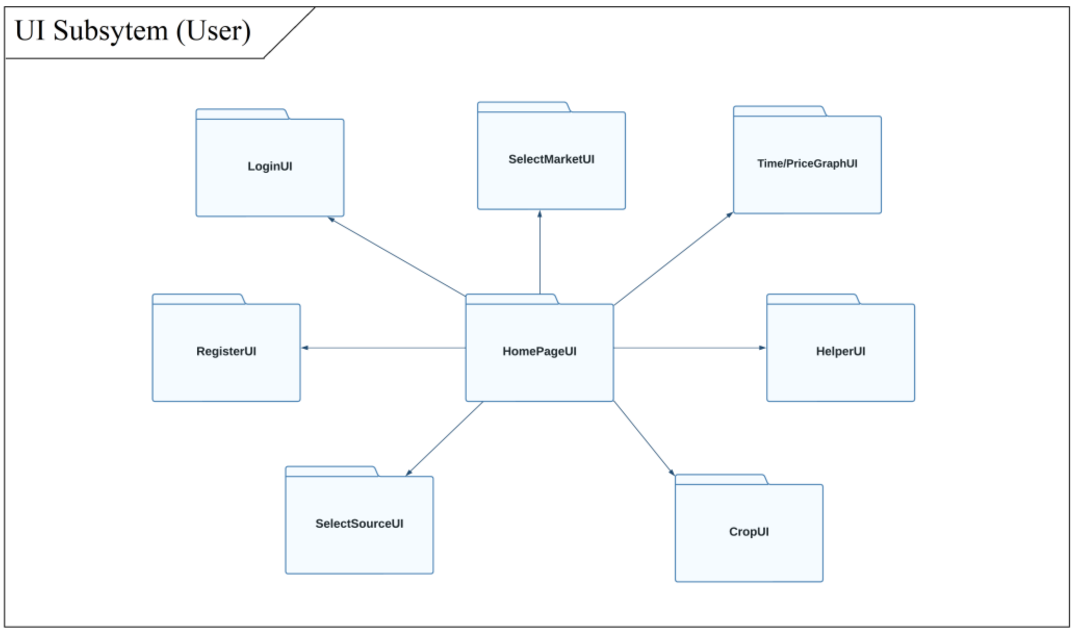
Test results will be documented in a test report. The test report includes the results of the test scenarios, details of any errors found, the status of error reports, and progress status. Errors are monitored through project management software and corrected when necessary. This approach provides a timely and effective method for completing quality control and testing processes, helping to meet project goals, and ensuring that the product is valuable and functional for customers.

2. Packages

This project has a modular design using packages. Packages allow the project to be divided into sections, each with different functionality, making the code easier to read, understand, and manage. Receipt Analyzer is a mobile application that provides options with various screens and buttons to be used on the front end (client) and updates itself with the detected text on the backend with the help of a database (server). By using these 2 structures or architectures, the application provides service and response with the help of a database that updates itself in the server part in line with the requests from the client part.

2.1 Client

2.1.1 User View



- **HomePageUI:** This is the first home page screen that welcomes the user after the mobile application is opened. Other features of the application can be accessed via this screen. Examples of this are logging in, registering, taking a photo, or selecting a picture from the gallery to start the detection process.

- **LoginUI:** With the login button on the home page screen, the login feature is accessed, and the user logs in with the username and password to start a receipt scanning process in the Receipt Analyzer application.

- **RegisterUI:** Users who use the Receipt Analyzer application for the first time must register with their username, email, and password on this screen before logging in. This functionality of the application is displayed on home page screen as a button.

- **SelectMarketUI:** Although the application focuses on only one market grocery, user also encounters a market selection button before starting scanning process. In this way, it is observed on which market brand the application is supported and working.

- **SelectSourceUI**: In addition to camera feature that is going to be used from user to scan market receipt photos, it is also provided a gallery selection option which allows to use receipts photos taken in the past.

- **CropUI:** After the scanning the receipt, since the application is based on comparing prices of products taken from receipts, a crop image screen is shown to the user to extract unnecessary parts from receipts like market location, QR code, etc. By this way, the necessary parts of the detected texts are written on the database.

- **HelperUI:** Currently the Helper UI contains three important features. Firstly, it shows all detected text on the screen to the user which allows to see if the OCR operation is successful or not in the implementation. Secondly, by pressing the "DETECTED TEXT" button, the content of the data in the scanned receipt is written to the Realtime Database. Lastly, this interface also allows users to start the scanning process again with the "RETAKE" button which redirects them to the SelectSourceUI.

- **TimePriceGraphUI:** This is the last interface that is shown to the user in the application. The main purpose of this UI is to illustrate the changes of prices of products that are sold in grocery stores or supermarkets, which is the main goal of the project. The user experiences the application with a kind of graph where the x-axis is time, and the y-axis is price of the product in TL.

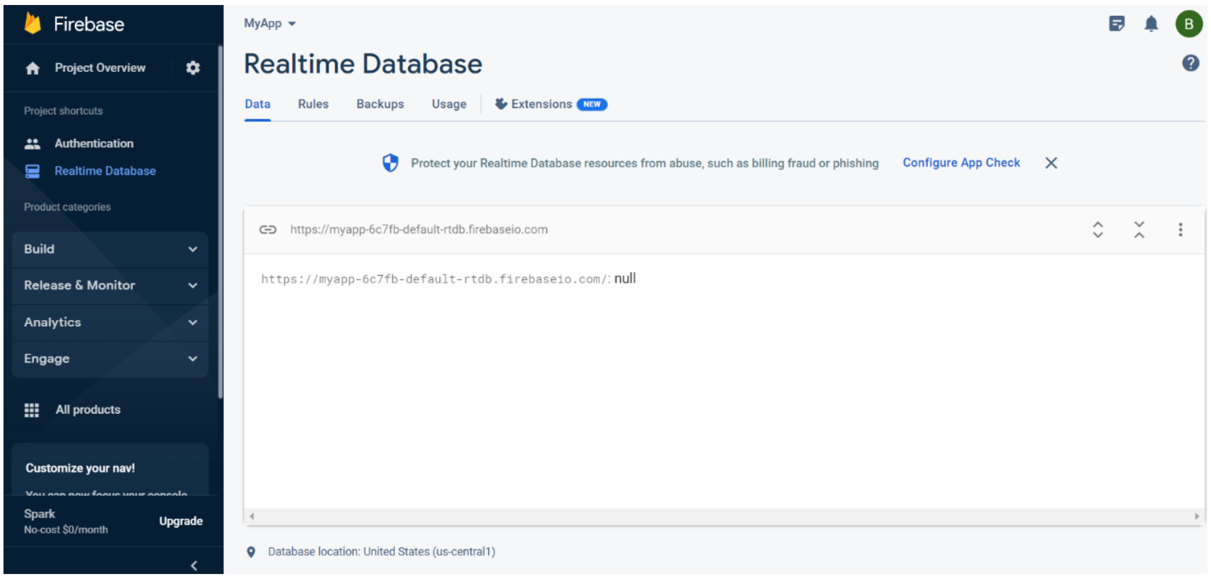
2.2 Server

2.2.1 Storage

2.2.1.1 Data Storage

As a storage mechanism of the Android Project, the Realtime Database, which is the solution of Firebase is used to hold the data of detected text from the market receipt. Firebase Realtime Database allows to keep data on a cloud-hosted database. This sync data is stored with NoSQL cloud database. By the help of JSON files, data is synchronized, when client connected to the mobile application. In the project there is no REST API usage. Since the data is synchronized, in Realtime Database there is no HTTP requests along the implementation of mobile application.

The project overview of Firebase has a console screen for previously created Android projects. In this console, information written to the database can be viewed. In this project, the information is the detected text from the market receipt. For instance, this figure illustrates a simple database screen with null, which does not contain any data yet.



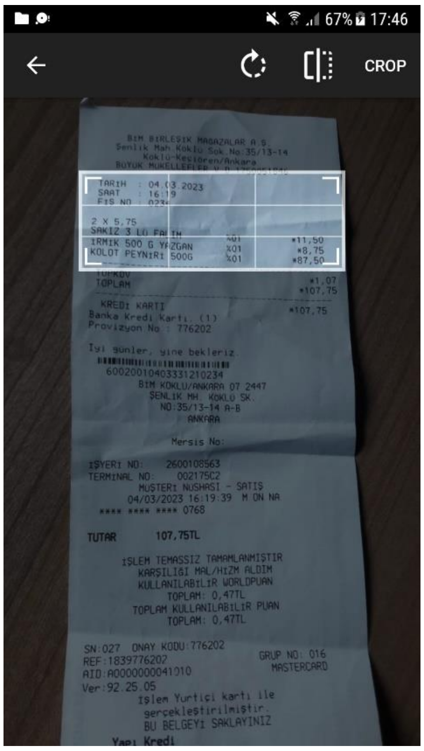
Protect your Realtime Database resources from abuse, such as billing fraud or phishing

https://myapp-6c7fb-default-rtdb.firebaseio.com

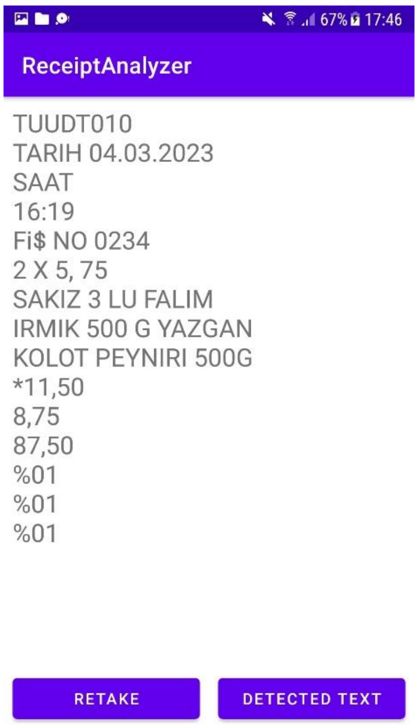
```
https://myapp-6c7fb-default-rtdb.firebaseio.com/: null
```

Database location: United States (us-central1)

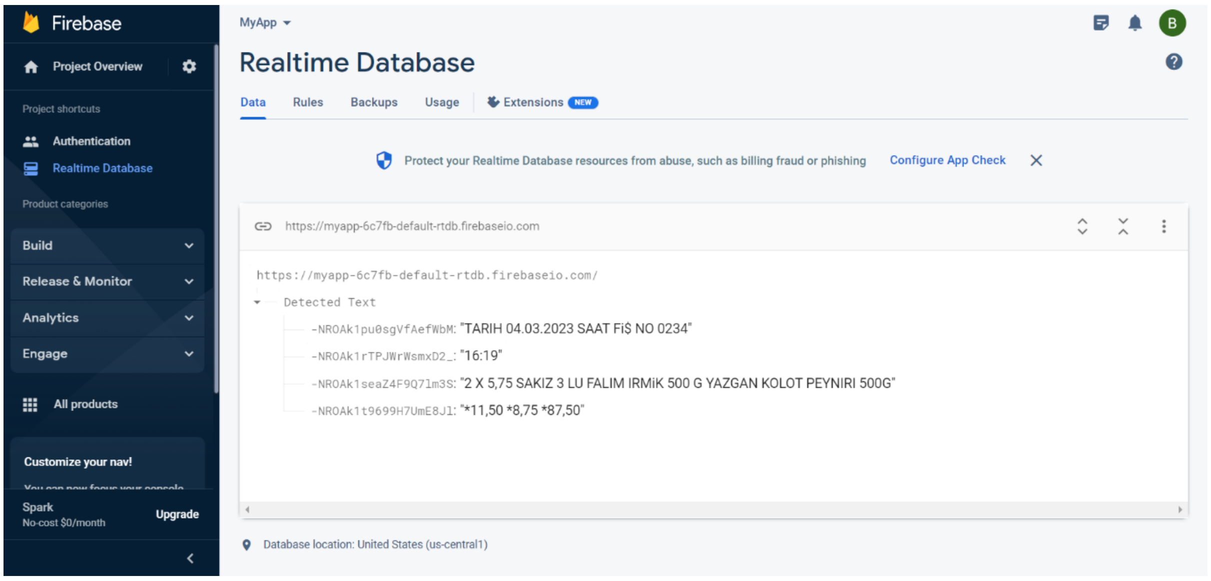
After the mobile application run, text detection operation completes successfully. When the user clicks on the DETECTED TEXT button, the current state of the database can be observed as below.



```
BIM BIRLESIK MAGAZALAR A.S.
Senlik Mah Koklu Sok.No:35/13-14
Koklü-Keçiören/Ankara
BUYUK MUKELLEFLER V.D.
TARIH  : 04.03.2023
SAAT   : 16:19
FIS NO : 0234
2 X 5,75
SAKIZ 3 LU FALIM
IRMIK 500 G YAZGAN       %01     *11,50
KOLOT PEYNIRI 500G       %01      *8,75
                         %01     *87,50
TOPKDV                            *1,07
TOPLAM                          *107,75
KREDI KARTI                     *107,75
Banka Kredi Kartı. (1)
Provizyon No : 776202
Iyi günler, yine bekleriz.
6002001040333121O234
BIM KOKLU/ANKARA 07 2447
ŞENLIK MH. KOKLU SK.
NO:35/13-14 A-B
ANKARA
Mersis No:
IŞYERI NO:    2600108563
TERMINAL NO:    002175C2
MUŞTERI NUSHASI - SATIŞ
04/03/2023 16:19:39  M ON NA
**** **** **** 0768
TUTAR     107,75TL
IŞLEM TEMASSIZ TAMAMLANMIŞTIR
KARŞILIĞI MAL/HIZM ALDIM
KULLANILABILIR WORLDPUAN
TOPLAM: 0,47TL
TOPLAM KULLANILABILIR PUAN
TOPLAM: 0,47TL
SN:027  ONAY KODU:776202
REF:1839776207               GRUP NO: 016
AID:A0000000041010             MASTERCARD
Ver:92.25.05
Işlem Yurtiçi kartı ile
gerçekleştirilmiştir.
BU BELGEYI SAKLAYINIZ
Yapı Kredi
```



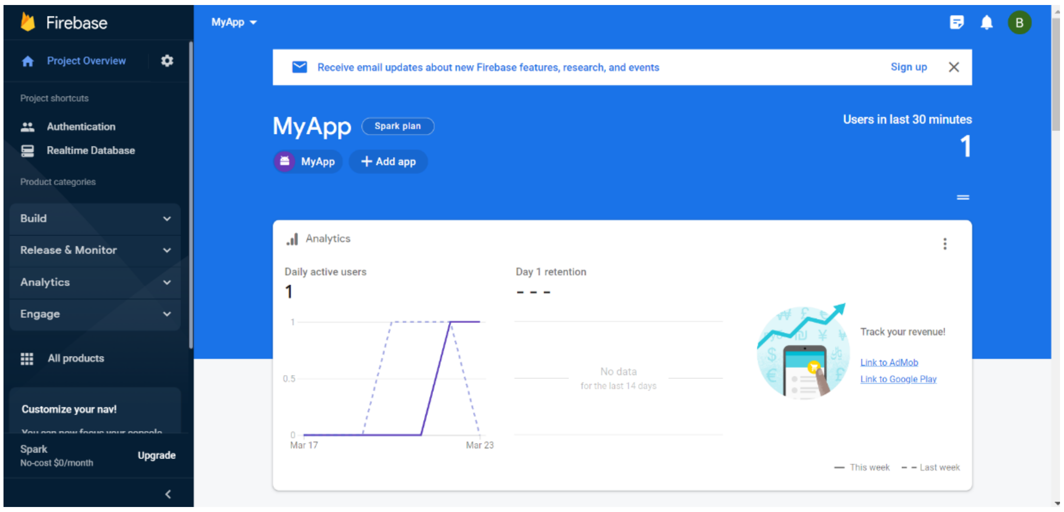
```
TUUDT010
TARIH 04.03.2023
SAAT
16:19
Fi$ NO 0234
2 X 5, 75
SAKIZ 3 LU FALIM
IRMIK 500 G YAZGAN
KOLOT PEYNIRI 500G
*11,50
8,75
87,50
%01
%01
%01
```



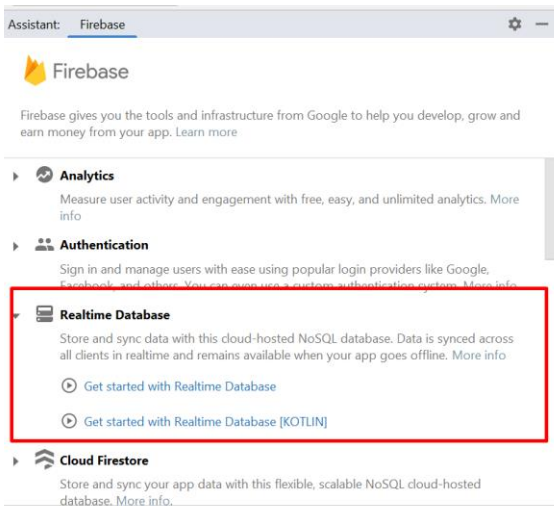
```
https://myapp-6c7fb-default-rtdb.firebaseio.com/
▾ Detected Text
    -NROAk1pu0sgVfAefWbM: "TARIH 04.03.2023 SAAT FiŞ NO 0234"
    -NROAk1rTPJWrWsmxD2_: "16:19"
    -NROAk1seaZ4F9Q7lm3S: "2 X 5,75 SAKIZ 3 LU FALIM IRMiK 500 G YAZGAN KOLOT PEYNIRI 500G"
    -NROAk1t9699H7UmE8J1: "*11,50 *8,75 *87,50"
```

2.2.1.2 Database Connection

When entering the Firebase console, a project is created from scratch. When creating the project, the package name of the project must be the same as the package in the code, and the necessary Firebase SDK and dependencies must be added into code part.



Android Studio, which is the IDE used in the development of the Android-based mobile project, defines the ability to work with and connect with Firebase to the user. Thanks to the buttons on the following screens, the necessary dependencies are added, and the Android-based mobile application is successfully connected to Firebase and Firebase Realtime Database.



Assistant: Firebase

Firebase

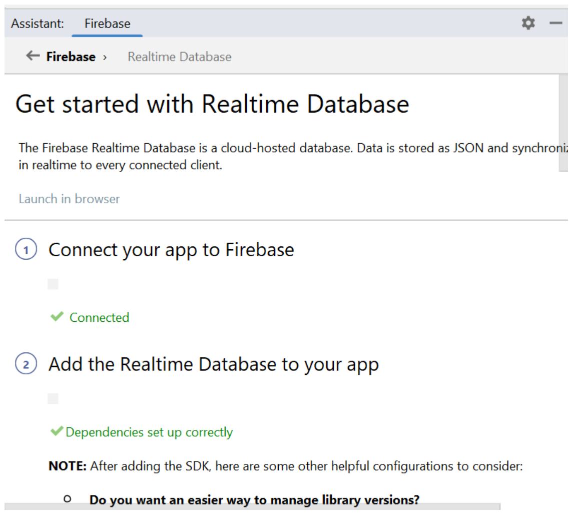
Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more

Analytics
Measure user activity and engagement with free, easy, and unlimited analytics. More info

Authentication
Sign in and manage users with ease using popular login providers like Google, Facebook, and others. You can even use a custom authentication system. More info

Realtime Database
Store and sync data with this cloud-hosted NoSQL database. Data is synced across all clients in realtime and remains available when your app goes offline. More info
Get started with Realtime Database
Get started with Realtime Database [KOTLIN]

Cloud Firestore
Store and sync your app data with this flexible, scalable NoSQL cloud-hosted database. More info.



Assistant: Firebase

Firebase › Realtime Database

Get started with Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchroni in realtime to every connected client.

Launch in browser

1 Connect your app to Firebase

✓ Connected

2 Add the Realtime Database to your app

✓ Dependencies set up correctly

NOTE: After adding the SDK, here are some other helpful configurations to consider:

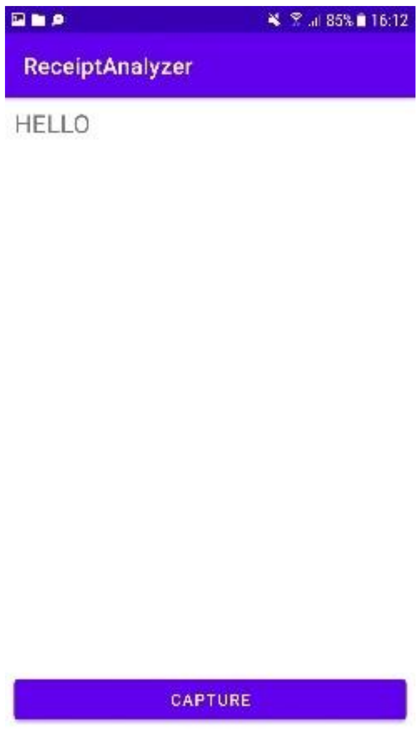
- **Do you want an easier way to manage library versions?**

3. Class Interfaces

3.1 Client and Frontend Interfaces

3.1.1 Home Page UI

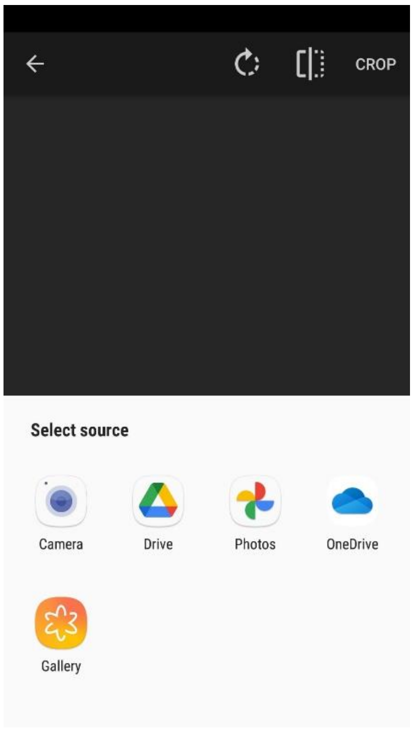
activity_main.xml
This is the class that establishes the front-end site of the mobile application project, which consist of capture button and simple welcoming message. The user first encounters this screen after the application is started on the mobile phone.
Attributes
+ textData: TextView + linearHolder: LinearLayout + buttonCapture: Button + rootView: RelativeLayout
Methods
The xml file does not contain any methods. These methods work in the backend of the MainActivity.java class.

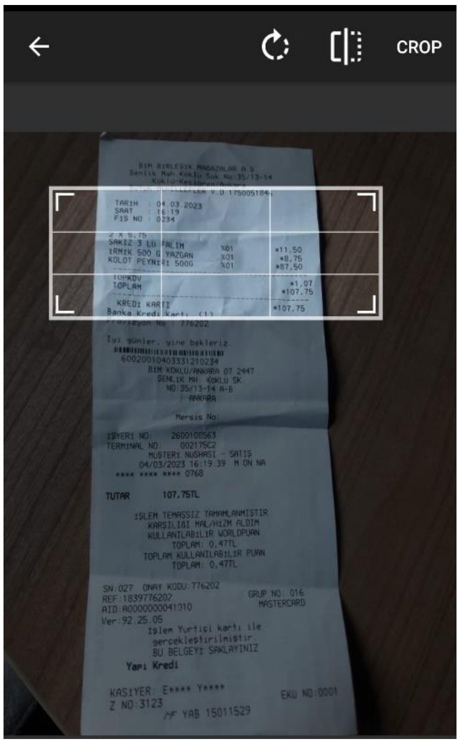


Screenshot taken from the mobile app.

3.1.2 Crop Page UI

activity_main.xml
After user presses on capture button, they encounter a crop page which consists of photo selection and camera screen. By pressing one of them, they can click crop button to extract the necessary part of the receipt.
Attributes
+ buttonCapture: Button + linearHolder: LinearLayout + rootView: RelativeLayout
Methods
The xml file does not contain any methods. These methods work in the backend of the MainActivity.java class.

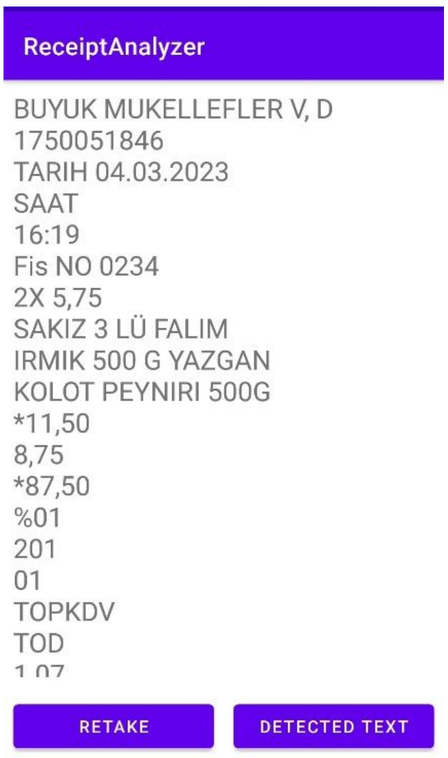




Screenshots taken from the mobile app.

3.1.3 Helper UI

activity_main.xml
After the image is cropped and scanned successfully, this Helper UI displays detected text on the screen, and provides two helper buttons. One of them stands for repeating the capture process, other button allows to transmit and write the data to Firebase Realtime Database.
Attributes
+ buttonCopy: Button + linearHolder: LinearLayout + buttonCapture: Button + rootView: RelativeLayout
Methods
The xml file does not contain any methods. These methods work in the backend of the MainActivity.java class.



```
BUYUK MUKELLEFLER V, D
1750051846
TARIH 04.03.2023
SAAT
16:19
Fis NO 0234
2X 5,75
SAKIZ 3 LÜ FALIM
IRMIK 500 G YAZGAN
KOLOT PEYNIRI 500G
*11,50
8,75
*87,50
%01
201
01
TOPKDV
TOD
1 07
```

Screenshot taken from the mobile app.

3.2 Server

3.2.1 Backend Structure

MainActivity.java
This Java class empowers the main working of the application. In this class, by adding commands to the buttons, new functionalities are developed. The development of photo taking, cropping and getting desired text features is done in this class. Realtime Database connection and saving data also takes place in this class. The permission to take a camera from the phone is also done within the framework of this class.
Attributes
- REQUEST_CAMERA_CODE: int + user: Users + bitmap: Bitmap + textView_data: TextView + button_copy: Button + button_capture: Button + textBlock: TextBlock + reference: DatabaseReference + db: FirebaseDatabase + binding: ActivityMainBinding
Methods
onCreate (savedInstanceState: Bundle): void In this method, the button variables created in the xml file are initialized with their ids and the buttons are prepared for the tasks. In addition, the camera permission is developed in this method, which allows the application to execute.
onActivityResult (requestCode: int, resultCode: int, data: Intent): void In this method, with the help of the dependency and libraries, the helper CropImage class is used to crop functionalities of the pictures taken or selected from the gallery. At the same time, with the helper MediaStore class, the photos in the mobile phone's gallery can be accessed to obtain the texts on the receipt picture.
- getTextFromImage (bitmap: Bitmap): void Thanks to this method, the text in the receipt photograph is divided into text blocks with line by line thanks to the Array List data structure, and the texts on the receipt are obtained in the form of lines. This method also allows to perform the implementation again thanks to the button_capture.
- saveTextToDB(): void In this part, the obtained data in the previous method, which is getTextFromImage, is written and saved with the helper push() keyword method into Firebase Realtime Database line by line. At the same time, thanks to the helper class Users, the data obtained line by line is saved with the help of Realtime Database child methodology.

Users.java
This is the helper class of the project which allows program to construct line by line scanned texts with child nodes in Realtime Database.
Attributes
- text: String
Methods
+ Users (text: String) This is the constructor of the User class. Since the detected text is stored as String, the constructor takes a single String parameter called ‘text’ to get the desired detected text on Realtime Database.
+ getText(): String Getter method that gets the text as a String.
+ setText (text: String): void Setter method that sets the text as a String.

3.2.2 Storage

Firebase Realtime Database (MainActivity.java)
This structure allows to store the receipt as a sync data with NoSQL cloud database provided from Google services.
Attributes
+ user: Users + reference: DatabaseReference + db: FirebaseDatabase
Methods
- saveTextToDB(): void The User object is created in this method and the information scanned in the receipt is put into the User object as a parameter. Then the database is created using the FirebaseDatabase class and the words scanned with the push() method are written to the database through unique ids, line by line.

4. Glossary

Mobile Application	A mobile application or app is a computer program or software application designed to run on a mobile device such as phone, tablet, or watch.
Receipt Analyzer	The name of our project which is a mobile application to scan market receipts.
OCR	Optical Character Recognition is a process that converts an image of text into a machine-readable text format.
Firebase	Firebase is a platform developed by Google for building mobile and web applications.
Database	A database is an organized collection of structured information, or data, typically stored electronically in a computer system. Considering the wide variety of products in a market, for this project, it is important to store detailed information about the products in the database after the receipts are scanned.
Realtime Database	Realtime Database, which is the solution of Firebase is used to keep data on a cloud-hosted database. This sync data is stored with NoSQL cloud database.
Frontend/Backend	Front-end is a visual representation of an application which the user can see like colors, buttons, menus, etc. Backend is the part of the application which is not visible to the user. It contains structure, logic, data, and programming of the application.
Client/Server	Client-Server involves one program (the client) asking another program (the server) for a service or resource.
SDK	A software development kit (SDK) is a group of installable software development tools.
XML	The markup language XML (Extensible Markup Language) is used to create documents that are simple to comprehend by both people and computer systems.
JSON	JavaScript Object Notation is referred to as JSON. It is used for storing and transporting data.
HTTP	The Hyper-Text Transfer Protocol is a source-distributed, open-source application-level communication protocol for hypermedia information networks.
UI	The point of human-computer interaction and communication in a device. Since the idea of scanning receipts provides its service on the mobile application, we should be considering the user interface carefully as developers.
Sketch/Plot	Sketch/Plot is a graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables. In our project, we use this technique to graphically display the long-term price and name information of the scanned products to the users.
Android (Operating System)	A Linux-based mobile operating system that primarily runs on smartphones and tablets.

5. References

1. *Andrews, J. G. (2001). Requirements Trade-offs During UML Design. IEEE Communications Magazine, 39(6), 162-169. Retrieved from https://fileadmin.cs.lth.se/serg/old-serg-dok/docs-serg/262_andrewsa_tradeoffs.pdf*

2. *Association for Computing Machinery. (2018). ACM code of ethics and professional conduct. Retrieved from https://www.acm.org/code-of-ethics*

3. *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.*

4. *Firebase. (n.d.). Firebase Realtime Database. Retrieved from https://firebase.google.com/docs/database*

5. *Mobile app. (n.d.). In Wikipedia. Retrieved January 3, 2023, from https://en.wikipedia.org/wiki/Mobile_app*

6. *What is a database? (n.d.). In Oracle. Retrieved January 3, 2023, from https://www.oracle.com/database/what-is-database/*

7. *Frontend GmbH. (2022). What is frontend and what is backend? Retrieved from https://www.frontend-gmbh.de/en/blog/what-is-frontend-what-is-backend/*

8. *ComputerScience.org. (2022). Frontend vs. Backend: What's the Difference? Retrieved from https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/#:~:text=What's%20the%20Difference%20Between%20Front,system%2C%20data%2C%20and%20logic.*

9. *Wikimedia Foundation. (2022). Firebase. Retrieved from https://tr.wikipedia.org/wiki/Firebase*

10. *TechTarget. (2019). Client-server definition. Retrieved from https://www.techtarget.com/searchnetworking/definition/client-server#:~:text=Client%2Dserver%20is%20a%20relationship,another%20program%20(the%20server)*

11. *Wikipedia. (2022). Software development kit. Retrieved from https://en.wikipedia.org/wiki/Software_development_kit*

12. *Wikipedia. (2022). XML. Retrieved from https://tr.wikipedia.org/wiki/XML*

13. *W3Schools. (2021). JSON Introduction. Retrieved from https://www.w3schools.com/whatis/whatis_json.asp*

14. *Wikipedia. (2022). Hypertext Transfer Protocol. Retrieved from https://tr.wikipedia.org/wiki/HTTP*