Bilkent University

Department of Computer Engineering

Senior Design Project

InPackt

Final Report

May 6, 2022

Vural Doğan Akoğlu - 21602479

Kutsal Bekçi - 21602163

Şükrü Can Erçoban - 21601437

Fadime Selcen Kaya - 21801731

Kadir Mert Laleci - 21601960

Supervisor: Asst. Prof. Dr. Hamdi Dibeklioğlu
Jury Members: Asst. Prof. Dr. Shervin Arashloo, Asst. Prof. Dr. Hamdi Dibeklioğlu

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

After the industrial revolution, the number of packaged products has increased drastically. Nowadays markets are full of packaged products and comparing the products is a time consuming activity in our lives. Reading the table of contents for each product we bought, understanding chemicals in the product and deciding which one to put into the shopping cart became a growing problem as the chemicals and number of packaged products increased.

Health is without question the single-most important value in life. Without sound health, nothing else can be important. Therefore, great importance must be given to how we fuel our bodies. Many food items, especially packaged ones, contain additives that the average person is not even able to pronounce, let alone know the effects. People need to be educated about these ingredients in order to make informed choices about their most basic need.

Based on these issues, InPackt will offer a quick and healthy shopping experience and help to compare between products. Thanks to algorithms and developing technologies, we can compare and decide the best products for us without looking at their table of contents directly. The user only needs to point their phone camera at the product and InPackt will take care of everything from there. It will display to the user the ingredients, packaging information, dangerous content warnings as well as similar product recommendations based on user preferences.

In this report, requirements details will be explained. Final architecture and design details of the application will be shown using different diagrams and visuals. Development tools and implementation details of the application will be explained. The test cases and test environments will be demonstrated. In addition to testing, maintenance process of the application after the release will be mentioned. Then other project elements such as engineering design factors, ethics and teamwork will be explained in detail. Finally, any further applications will be considered in the conclusion.

2. Requirements Details

2.1. Functional Requirements

2.1.1. System

The system should:

- Provide an option to create an account with user liked/disliked lists..
- Provide an option to continue without creating an account.
- Display the focused object apart from surrounding objects on the screen.
- Provide an option to search the product manually.
- Send the recognized object picture to the backend server for processing.
- Find the product from the database and receive its contents.
- Compare the contents of the product with the user's disallowed list.
- Display warnings about products which contain ingredients from the user's unwanted ingredient list.
- Suggest similar products without the ingredients from the unwanted list according to the user's personal choices.
- Provide options that the users can personalize their preferences by updating favorite or unwanted ingredients list.
- Provide an option to see the general ingredients list to choose and add to the unwanted list.

- Provide an option to remove ingredients from the unwanted list.

2.1.2. User

The user can:

- Create an account to save their preferences and personalized choices to the cloud.
- Continue without creating an account.
- Select a product to focus on the camera view.
- Search the product manually.
- View product ingredients.
- View warnings about the products that contain disliked ingredients.
- View similar products that do not contain the unwanted ingredients.
- View general ingredient list.
- Personalize their preferences by updating the favorite or unwanted ingredients list.
- Choose ingredients to add to the disallowed list.
- Remove ingredients from the disallowed list.

2.2. Nonfunctional Requirements

The non-functional requirements are explained below in separate headers.

2.2.1. Availability

- InPackt application should be available 24 hours a day, 7 days a week for all users who have devices Android operating systems.

- InPackt application should be available mainly for users who live in Turkey, and ideally worldwide.

2.2.2. Accuracy

- InPackt application should be able to display the ingredients of the product scanned by user with at least %90 accuracy.
- InPackt application should be %90 accurate when recommending products according to the personal characteristic specified by the user while creating a profile for the product scanned by the user.

2.2.3. Response Time

- InPackt application should display the ingredient of the product in less than 8 seconds after the user scans.
- InPackt application should display the alternative products in less than 8 seconds.

2.2.4. Extendibility

- The design of the InPackt application should be extensible for future updates covering additional features according to the needs of the users.
- The user data of the InPackt application should be stored to be extensible for a Web application or other platforms.

2.2.5. Scalability

- InPackt application database should be scalable according to the increased number of users. Initially, at least 100 different devices can access simultaneously.

2.2.6. Privacy

- User information should not be stored and used outside the user’s device.
- User information should be encrypted to store their own devices.
- The InPackt application should ask the user’s permission for the alternative product recommendation system according to the private user information that has been created by the user.

2.2.7. License

- All used databases, libraries and the like must be licensed.

2.2.8. Maintainability

- The InPackt application should have a design that will allow the maintainability of the system without any problems with future updates and ensuring that some changed parts are not affected to others functionality.

2.2.9. Usability

- The application aims to present to the user what is in the content of the product that the user wants to buy, whether it is suitable for the profile created by the user, and if it is not suitable, what other alternative products are available to the user in a fast, easy and reliable manner.
- Tutorial and guideline for how to use the InPackt app is available within the app.

2.2.10.Reliability

- The information displayed on the application must be accurate.
- There must be a brief information page about most used and unknown harmful ingredients to be displayed when there is no internet connection for informing users when they need to be informed.

2.3. Pseudo Requirements

2.3.1. Language Constraints

- Our application will support both Turkish and English languages.

2.3.2. Economic Constraints

- Github is used to maintain the website without any cost.
- For version control and code sharing Github will be used which is free to use.
- Free APIs will be used for recognition.
- Open Food Facts is an open-source and collaborative project which allows the use of its database for free.
- Open-source libraries will be used.

2.3.3. Implementation Constraints

- Jira and GitHub will be used for project management, source control and code review.
- The frontend will be designed by Flutter because the application will be on android.
- Object Recognition will be implemented based on YOLOv5 [1].
- The backend will be implemented by Flutter, Dart, Python [2] because we will use libraries for Python and Google APIs.
- The product data and object images which are used for object training are stored in Firebase and JSON [3].

- The product data will be obtained from internet marketplaces by parsing.

2.3.4. Social Constraints

- Understandable language must be used to describe the harmful ingredients in order to not misguide anyone.

2.3.5. Ethical Constraints

- The application will have a minimized, optimally no, error rate for allergens in order to protect users from harmful, potentially deadly, products. The information and database used must be accurate, flawless and up to date.
- In order to not affect the market competition, a new product should be added to the database as soon as possible.

2.3.6. Health and Safety Constraints

- Because of the ongoing pandemic, ideally the users should be able to identify a product in front of them without touching the package.
- The application will be customizable according to the user, since understanding of a healthy product may differ for each user. So, the recommendation system must be unbiased.

2.3.7. Sustainability Constraints

- The products must be updated and added regularly for long term use.
- Users should be able to give feedback or add non-existent products in the database.

- Any errors or bugs must be fixed as soon as possible.

3. Final Architecture and Design Details

Final architecture and design are shown in detail below.

3.1. Diagrams

3.1.1. Use-case Diagram

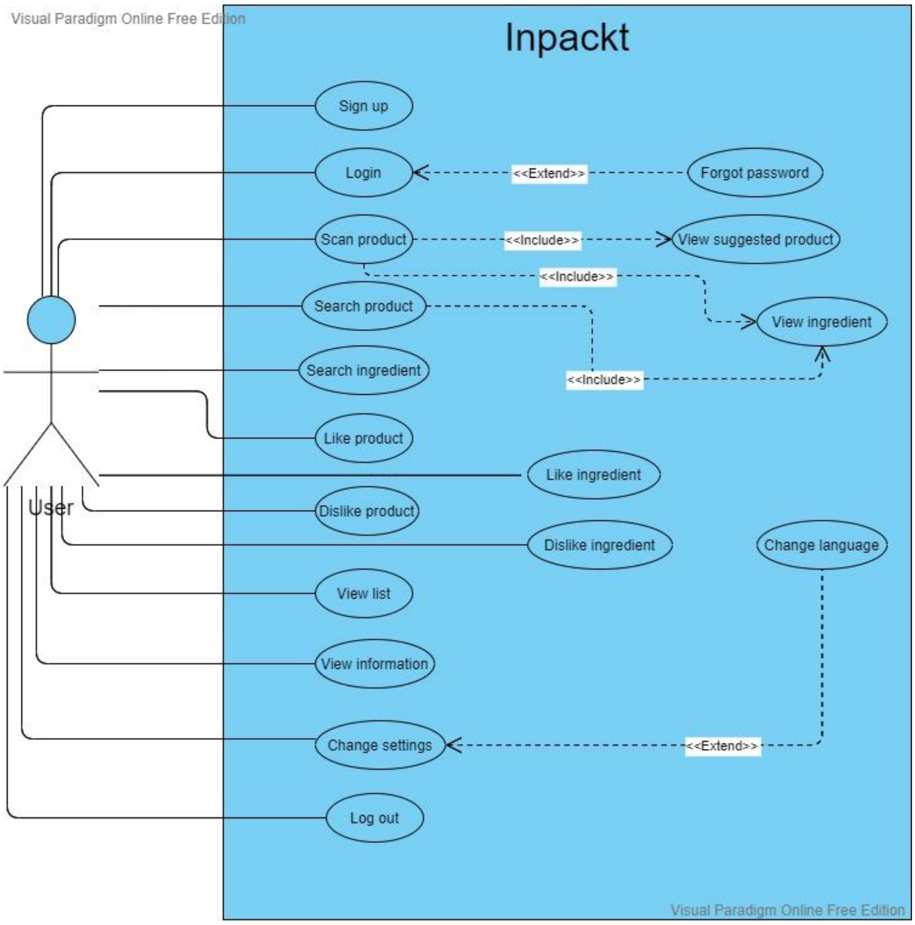


Figure 1: Use Case Diagram

3.1.1.1 Scenarios

Scenario 1	Sign up
Actor	User
Entry condition	User clicks the sign up button
Exit condition	1. Sign up process is successful. 2. User clicks the back button.
Flow of event	1. User enters the required information for login. 2. User completes the sign up process. 2.1 If the process is successful, the user is directed to the home page. 2.2 If the process fails, the user enters again.

Scenario 2	Login
Actor	User
Entry condition	User opens the application and clicks the login button.
Exit condition	1. Login process is successful. 2. User clicks the back button.
Flow of event	1. User enters the required information which is email and password for login. 2. User completes the login process. 2.1 If the process is successful, the user is directed to the home page. 2.2 If the process fails, the user enters again.

Scenario 3	Forgot password
Actor	User
Entry condition	User clicks forgot password button in login page
Exit condition	1. User’s password is reset 2. User clicks the back button.
Flow of event	1. User enters his/her email 2. User choose new password

Scenario 4	Scan a product
Actor	User
Entry condition	User enters scanning page from home page
Exit condition	1. The app redirects to the ingredient detail page 2. User clicks the back button.
Flow of event	1. User enables phone's camera 2. Scan the product 3. User views scanned product ingredients

Scenario 5	View product ingredient
Actor	User
Entry condition	1. User scans a product from scanning page with camera 2. User searches a product from searching page
Exit condition	User clicks the back button.
Flow of event	1. User scans or searches a product 2. The product emerges with its image and ingredient information

Scenario 7	Search a product
Actor	User
Entry condition	1. User enters searching page from home page 2. User write the searched word in searching bar
Exit condition	1. The app redirects to the product detail page 2. User clicks the back button.
Flow of event	1. User encounters with listed products according to search 2. User selects a product to see details

Scenario 8	Search an ingredient
Actor	User

Entry condition	1. User enters searching page from home page 2. User write the searched word in searching bar
Exit condition	1. The app redirects to the ingredient detail page 2. User clicks the back button.
Flow of event	1. User encounters with listed ingredient according to search 2. User selects a ingredient to see details

Scenario 9	Like product
Actor	User
Entry condition	User clicks like button in product detail
Exit condition	User selects the product as favorite
Flow of event	1. User access the product detail by scanning or searching 2. User add the product to the liked list

Scenario 10	Dislike product
Actor	User
Entry condition	User clicks dislike button in product detail
Exit condition	User selects the product as unwanted
Flow of event	1. User access the product detail by scanning or searching 2. User add the product to the disliked list

Scenario 11	Like ingredient
Actor	User
Entry condition	User clicks like button in ingredient detail
Exit condition	User selects the ingredient as favorite
Flow of event	1. User access the ingredient detail by scanning or

	searching 2. User add the ingredient to the liked list

Scenario 12	Dislike ingredient
Actor	User
Entry condition	User clicks dislike button in ingredient detail
Exit condition	User selects the ingredient as unwanted
Flow of event	1. User access the ingredient detail by scanning or searching 2. User add the ingredient to the disliked list

Scenario 13	View list
Actor	User
Entry condition	User clicks preferences in the side menu.
Exit condition	User clicks the back button.
Flow of event	1. User enters his/her profile and clicks preferences 2. User sees his/her liked and disliked products and ingredients list.

Scenario 14	View suggested product
Actor	User
Entry condition	User scans a product from scanning page with camera and views product ingredient
Exit condition	User clicks the back button.
Flow of event	1. User scans or searches a product and its ingredient information come up 2. Suggested product emerges as pop up according to preference list properties

Scenario 15	View information
Actor	User
Entry condition	User click information button in side menu
Exit condition	User clicks the back button.
Flow of event	1. User clicks information button in side menu 2. User sees a text about how to use InPackt app 3. User sees developers

Scenario 16	Change settings
Actor	User
Entry condition	User click change account settings button in setting page
Exit condition	User clicks the back button.
Flow of event	1. User click change account settings button in setting page 2. User can change profile name 3. User can change language 4. User can clear everything as reset all choices 5. User can delete his/her account

Scenario 17	Log out
Actor	User
Entry condition	User clicks the logout button
Exit condition	1. User logged out
Flow of event	1. User log out from the application

3.1.2. Object and Class Model

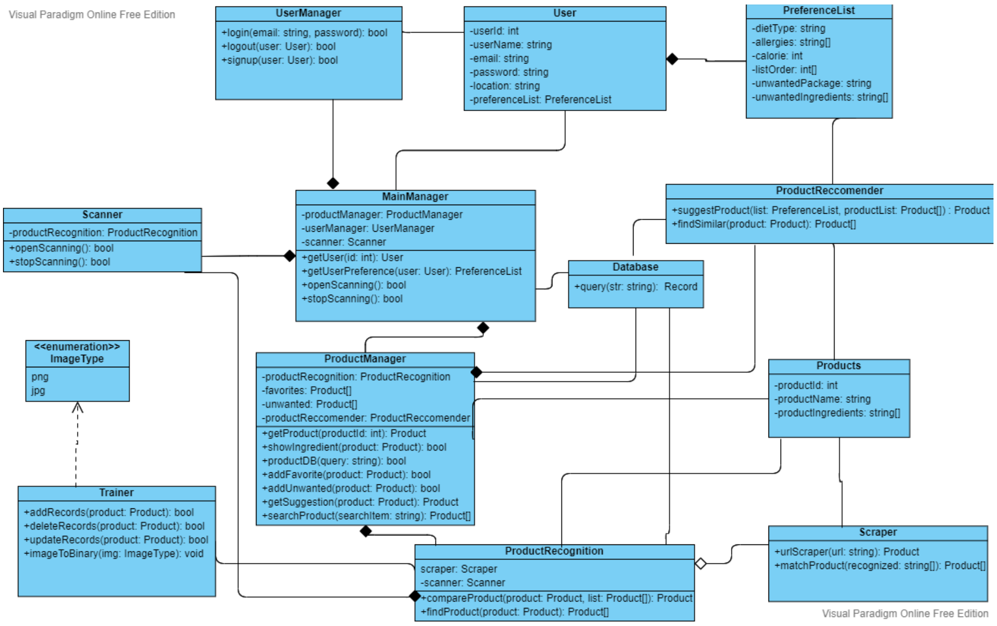

Visual Paradigm Online Free Edition
UserManager
+login(email: string, password): bool
+logout(user: User): bool
+signup(user: User): bool
User
-userId: int
-userName: string
-email: string
-password: string
-location: string
-preferenceList: PreferenceList
PreferenceList
-dietType: string
-allergies: string[]
-calorie: int
-listOrder: int[]
-unwantedPackage: string
-unwantedIngredients: string[]
Scanner
-productRecognition: ProductRecognition
+openScanning(): bool
+stopScanning(): bool
MainManager
-productManager: ProductManager
-userManager: UserManager
-scanner: Scanner
+getUser(id: int): User
+getUserPreference(user: User): PreferenceList
+openScanning(): bool
+stopScanning(): bool
ProductReccomender
+suggestProduct(list: PreferenceList, productList: Product[]) : Product
+findSimilar(product: Product): Product[]
Database
+query(str: string): Record
<<enumeration>>
ImageType
png
jpg
ProductManager
-productRecognition: ProductRecognition
-favorites: Product[]
-unwanted: Product[]
-productReccomender: ProductReccomender
+getProduct(productId: int): Product
+showIngredient(product: Product): bool
+productDB(query: string): bool
+addFavorite(product: Product): bool
+addUnwanted(product: Product): bool
+getSuggestion(product: Product): Product
+searchProduct(searchItem: string): Product[]
Products
-productId: int
-productName: string
-productIngredients: string[]
Trainer
+addRecords(product: Product): bool
+deleteRecords(product: Product): bool
+updateRecords(product: Product): bool
+imageToBinary(img: ImageType): void
ProductRecognition
scraper: Scraper
-scanner: Scanner
+compareProduct(product: Product, list: Product[]): Product
+findProduct(product: Product): Product[]
Scraper
+urlScraper(url: string): Product
+matchProduct(recognized: string[]): Product[]
Visual Paradigm Online Free Edition

Figure 2: Class Diagram

3.1.3. Activity Diagram

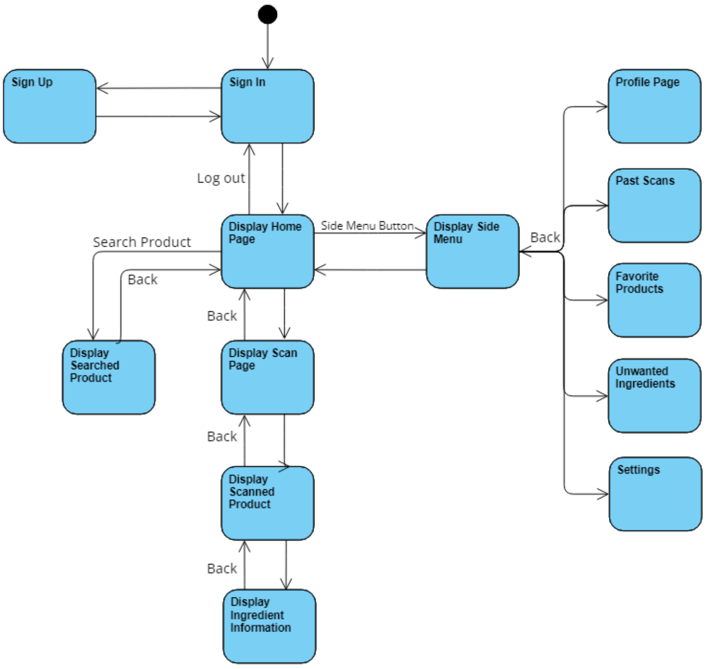


Figure 3: Overall Activity Diagram

3.1.4. State Diagram

First diagram shows the scanning state diagram which starts when the scanning request is received from the user. If the product can be recognisable by the application, the app moves to the product found state that shows the recognised product to the user. Otherwise, the app moves to product not found state and comes back to scanning for another try. If the identified product is confirmed by the user, the app moves to the display state that shows the ingredients of the identified product. In this state, the app offers two options to the user, one for end scanning and other for showing the package recycling information of the product. If the user requests for the package information, the app moves to the show package information state where the user can end the scanning process or request to scan another product.

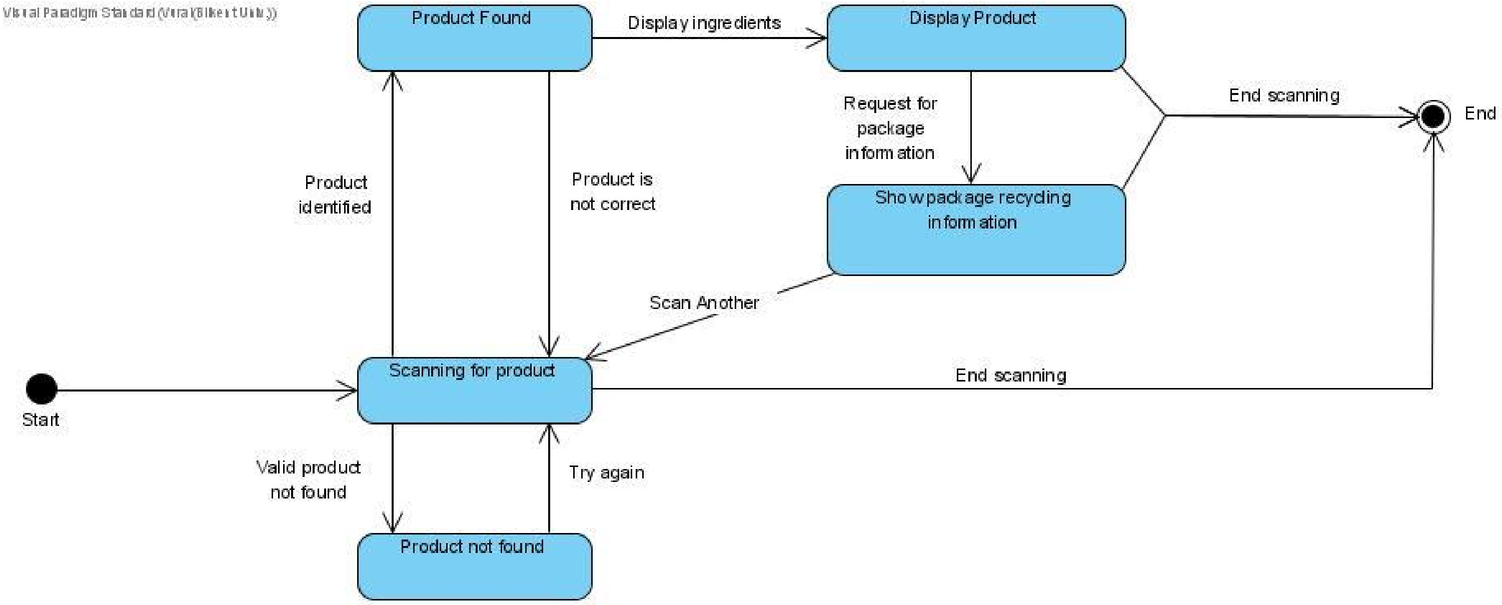


Figure 4: State Diagram

Second diagram shows the communication between the database and the app. When a product has been identified, the application requests the ingredient information from the database. The database returns the ingredient information and the app comes back to idle state for new requests until the application is closed by the user.

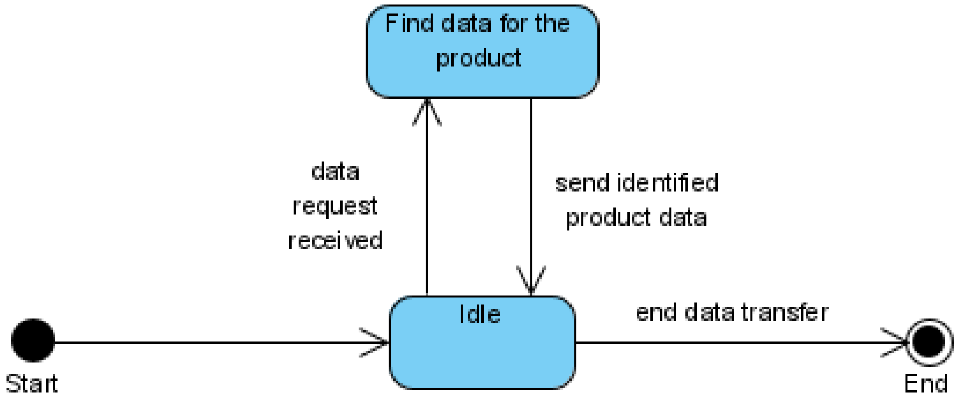


Figure 5: State Diagram of application request

Third diagram shows the product identification process in the backend application. The backend application has two states. In idle state the application waits for a product identification request and when received the request, the application goes to identify image state. After the identification of the image the identified product will be sent to the application and goes back to the idle state until the app is closed.

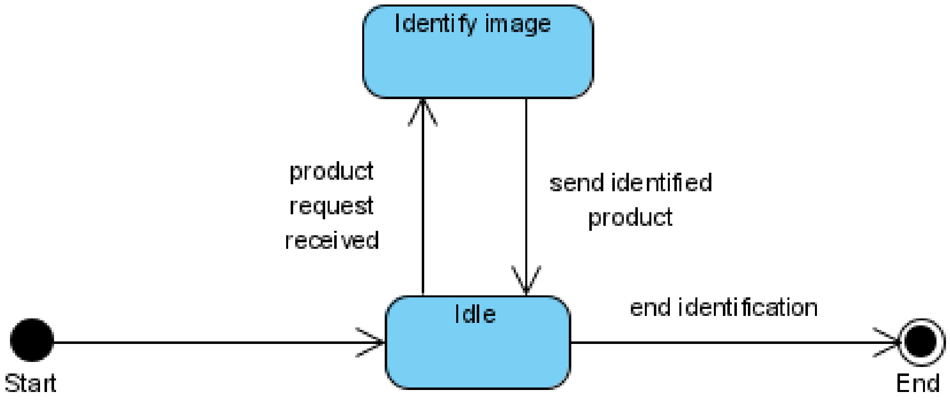


Figure 6: State Diagram of Identify Image

3.1.5. Sequence Diagram

3.1.5.1 Scan Product

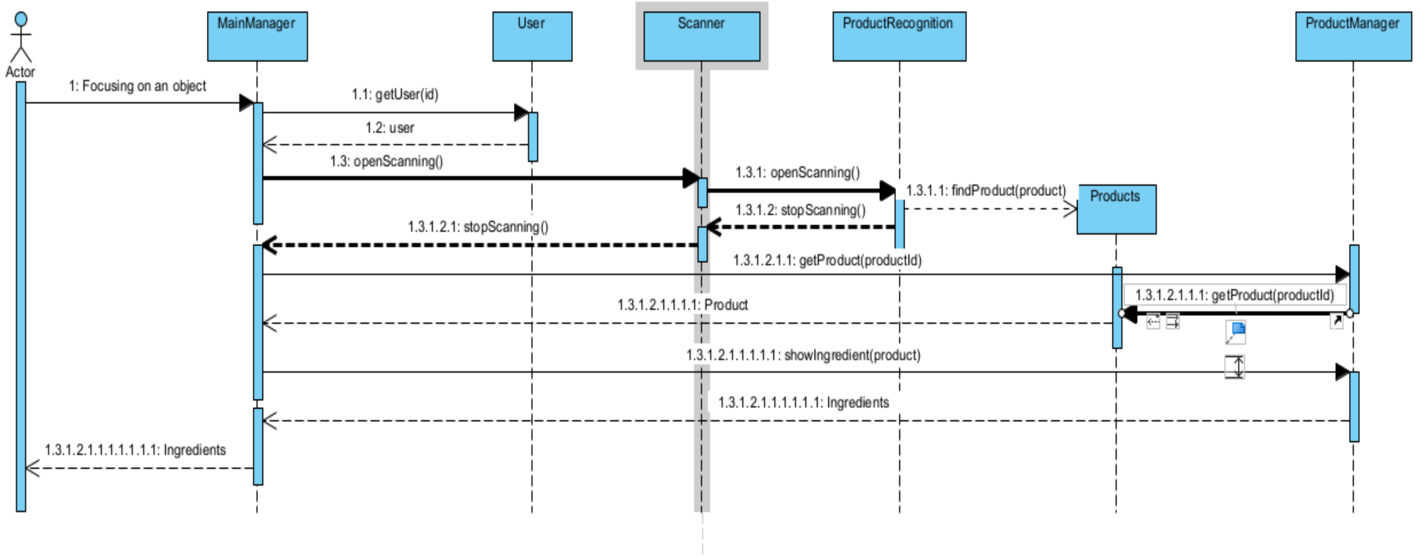

Actor
MainManager
User
Scanner
ProductRecognition
ProductManager
Products
1: Focusing on an object
1.1: getUser(id)
1.2: user
1.3: openScanning()
1.3.1: openScanning()
1.3.1.1: findProduct(product)
1.3.1.2: stopScanning()
1.3.1.2.1: stopScanning()
1.3.1.2.1.1: getProduct(productId)
1.3.1.2.1.1.1: getProduct(productId)
1.3.1.2.1.1.1.1: Product
1.3.1.2.1.1.1.1.1: showIngredient(product)
1.3.1.2.1.1.1.1.1.1: Ingredients
1.3.1.2.1.1.1.1.1.1.1: Ingredients

Figure 7: Sequence Diagram of Scanning a product

When the user focuses an object with the camera of the application, InPackt catches the image and gets the user id at the same time. While focusing, image and user information will be passed MainManager and triggers openScanning(). openScanner() triggers Scanner and ProductRecognition to find the product from the database. After finding the product MainManager asks product to ProductManager and gets the product with getProduct(product). After getting the product, MainManager asks for ingredients to ProductManager with showIngredients(product). It returns Ingredients information to the user.

3.1.5.2 View Suggestions

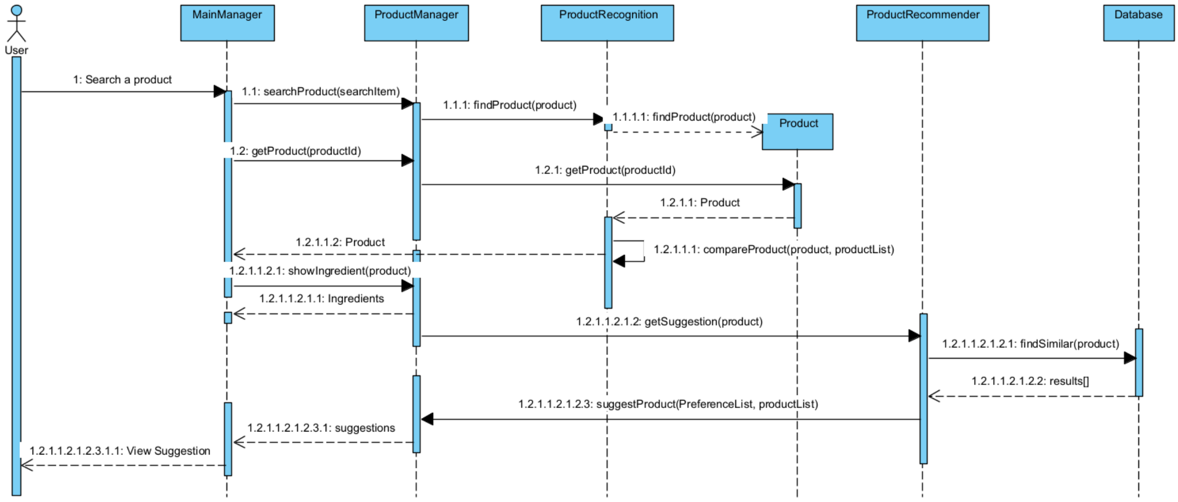

User
MainManager
ProductManager
ProductRecognition
ProductRecommender
Database
Product
1: Search a product
1.1: searchProduct(searchItem)
1.1.1: findProduct(product)
1.1.1.1: findProduct(product)
1.2: getProduct(productId)
1.2.1: getProduct(productId)
1.2.1.1: Product
1.2.1.1.2: Product
1.2.1.1.1: compareProduct(product, productList)
1.2.1.1.2.1: showIngredient(product)
1.2.1.1.2.1.1: Ingredients
1.2.1.1.2.1.2: getSuggestion(product)
1.2.1.1.2.1.2.1: findSimilar(product)
1.2.1.1.2.1.2.2: results[]
1.2.1.1.2.1.2.3: suggestProduct(PreferenceList, productList)
1.2.1.1.2.1.2.3.1: suggestions
1.2.1.1.2.1.2.3.1.1: View Suggestion

Figure 8: Sequence Diagram of Viewing suggestions

When the user searches a product, MainManager triggers searchProduct(searchItem) to ProductManager. ProductManager triggers findProduct(product) to ProductManager for finding the product from the database. After finding the product, It compares the other products with ProductManager and sends it to MainManager. MainManager shows ingredients and triggers getSuggestion(product) to ProductRecommender. ProductRecommender triggers findSimilar(product) to find similar products from the database using kNN algorithm. Database sends results and ProductRecommender lists suggested products with suggestProduct(PreferenceList, productList). After that, it sends suggestions to the Main Manager and user.

3.2. Packages

In this section, packages of the application are introduced. The application is divided into 3 layers which are presentation, logic and data for enhancing the independence and they communicate with each other.

3.2.1. Presentation Packages

The presentation package is responsible for receiving and displaying the inputs which come from the logic layer and sending the actions and requests to the logic layer.

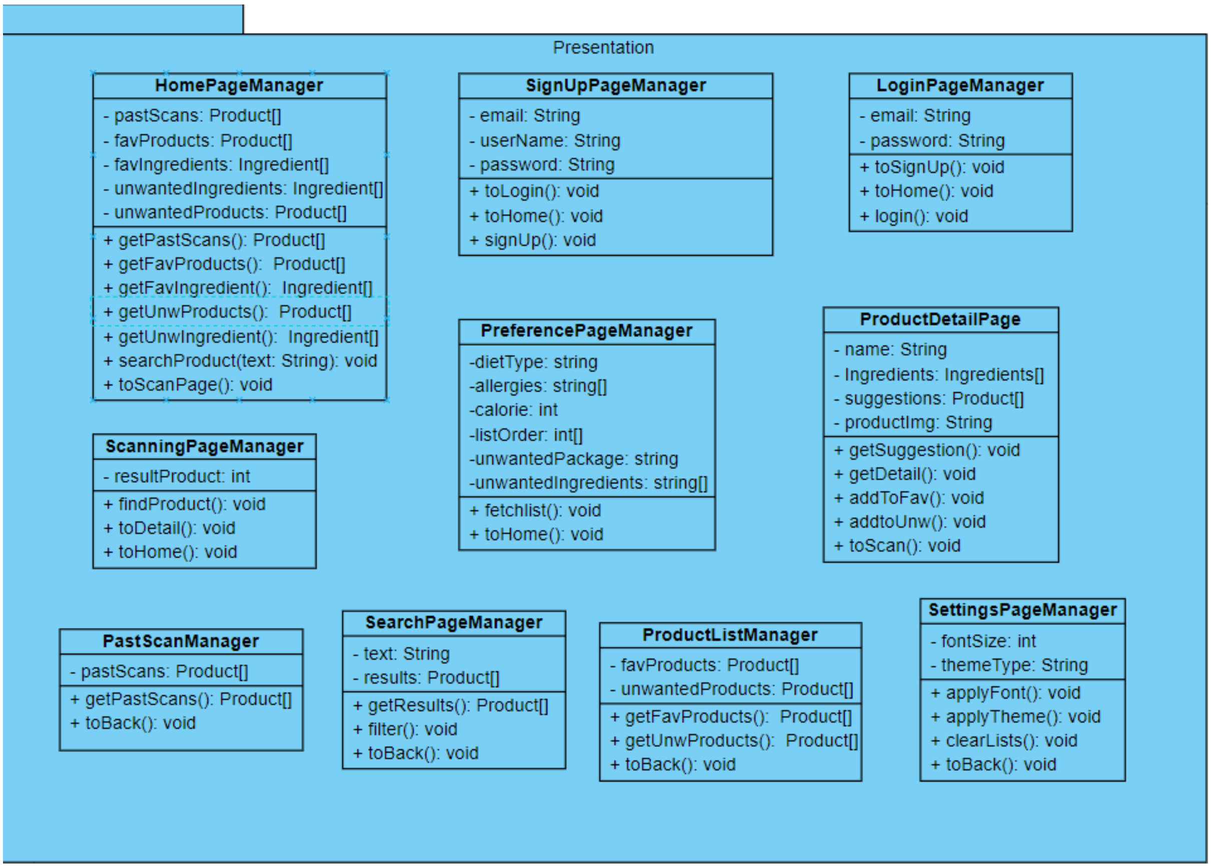


Figure 9: Presentation Package

3.2.2. Logic Package

3.2.2.1 Controller Package

The controller package is responsible for handling the events that are received from the UI components that are mentioned in the presentation package. This package contains the whole controllers in the application. There are 4 controllers and 3 of them are AuthManager, UserManager, ProductManager and there is one main controller which manages these 3 controllers.

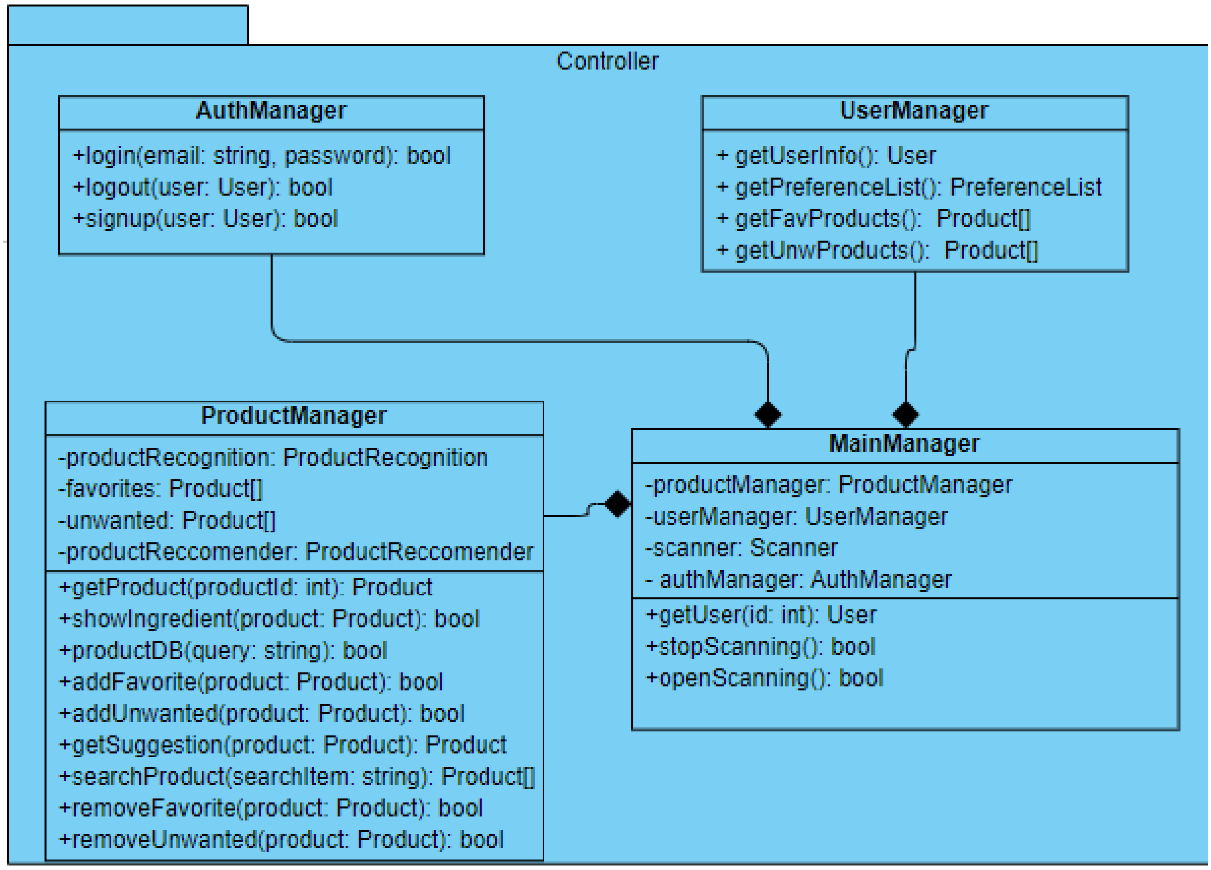

Controller
AuthManager
+login(email: string, password): bool
+logout(user: User): bool
+signup(user: User): bool
UserManager
+ getUserInfo(): User
+ getPreferenceList(): PreferenceList
+ getFavProducts(): Product[]
+ getUnwProducts(): Product[]
ProductManager
-productRecognition: ProductRecognition
-favorites: Product[]
-unwanted: Product[]
-productReccomender: ProductReccomender
+getProduct(productId: int): Product
+showIngredient(product: Product): bool
+productDB(query: string): bool
+addFavorite(product: Product): bool
+addUnwanted(product: Product): bool
+getSuggestion(product: Product): Product
+searchProduct(searchItem: string): Product[]
+removeFavorite(product: Product): bool
+removeUnwanted(product: Product): bool
MainManager
-productManager: ProductManager
-userManager: UserManager
-scanner: Scanner
- authManager: AuthManager
+getUser(id: int): User
+stopScanning(): bool
+openScanning(): bool

Figure 10: Controller Package

3.2.2.2 Engine Package

This package contains the main mechanisms of the application which are product detection and product suggestion. ProductRecognition is used for detecting the product and finding its category. According to detection action, Product Recommender gives a suggestion product list which are in the same category with detected product. These both are handled with ProductEngine.

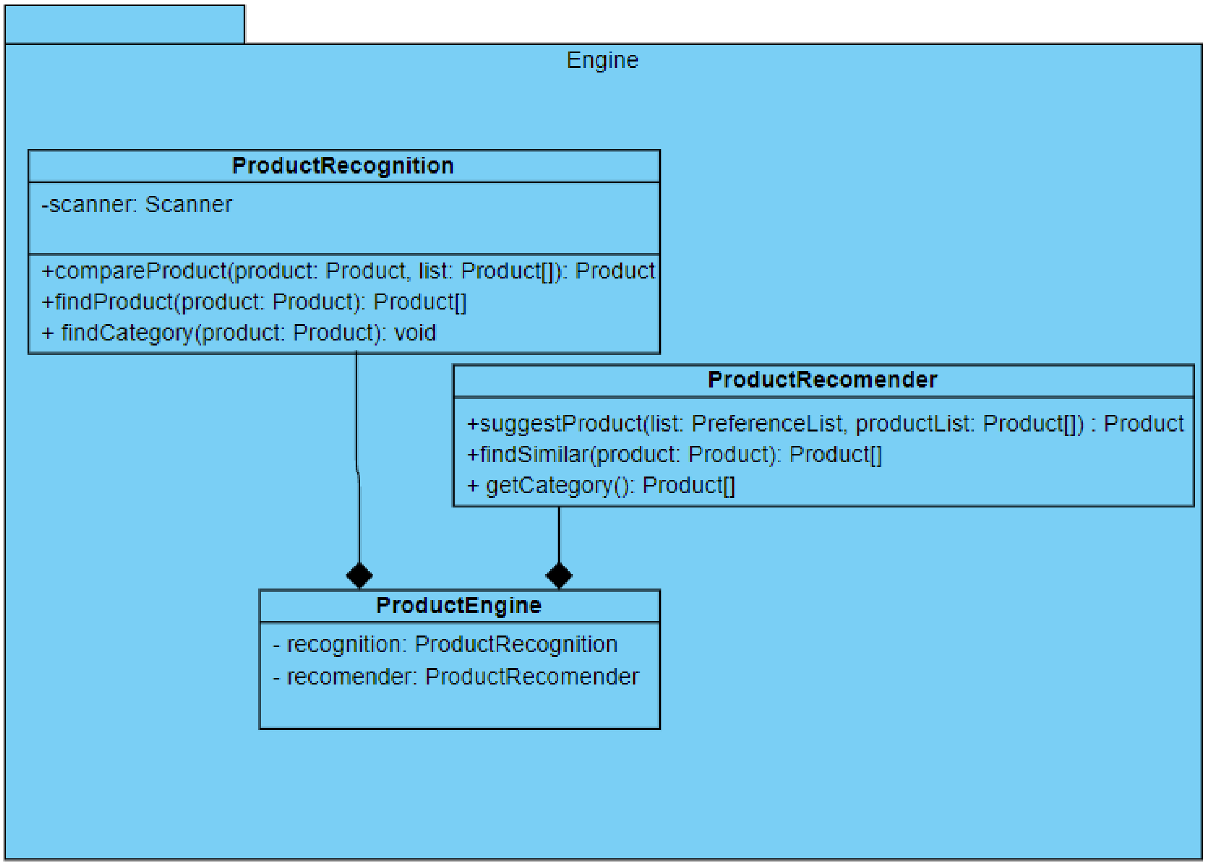

Engine
ProductRecognition
-scanner: Scanner
+compareProduct(product: Product, list: Product[]): Product
+findProduct(product: Product): Product[]
+ findCategory(product: Product): void
ProductRecomender
+suggestProduct(list: PreferenceList, productList: Product[]) : Product
+findSimilar(product: Product): Product[]
+ getCategory(): Product[]
ProductEngine
- recognition: ProductRecognition
- recomender: ProductRecomender

Figure 11: Engine Package

3.2.3. Data Package

Data package holds the necessary information for products and users. These data are accessible for logic packages to use.

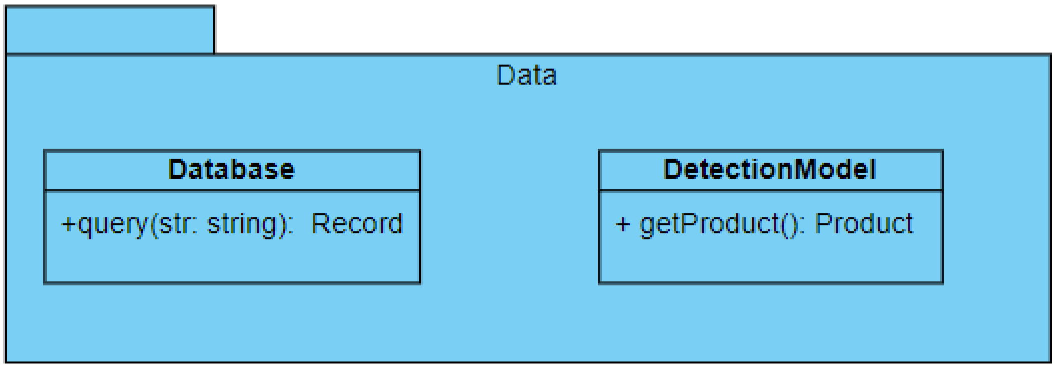


Figure 12: Data Package

4. Development/Implementation Details

In InPackt, the application is suitable for android devices, so Android Studio and emulators are used for the IDE platform. For the frontend, flutter is used and for the server side python is used.

InPackt is a mobile application and uses object detection model from server side. Yolov5 is used as a pre-trained model and it is trained by the Pytorch framework. Detecting API is developed by fastapi and it is deployed on Heroku server.

The application uses the data from the JSON file to serve offline and for the product images, they are obtained from Firebase since images allocate data more. Moreover, the user data is stored in Firebase. Product data is obtained by Web scraping and the Migros's website is used.

5. Testing Details

5.1. Object Recognition Model Testing

We have tested our object recognition model in pytorch. This helps us to improve our model before integrating the model into the front end of our project. Figure 13 and 14 are representing two of our test results. Figure 13 is an example of a successful test and Figure 14 is an example of a failure.

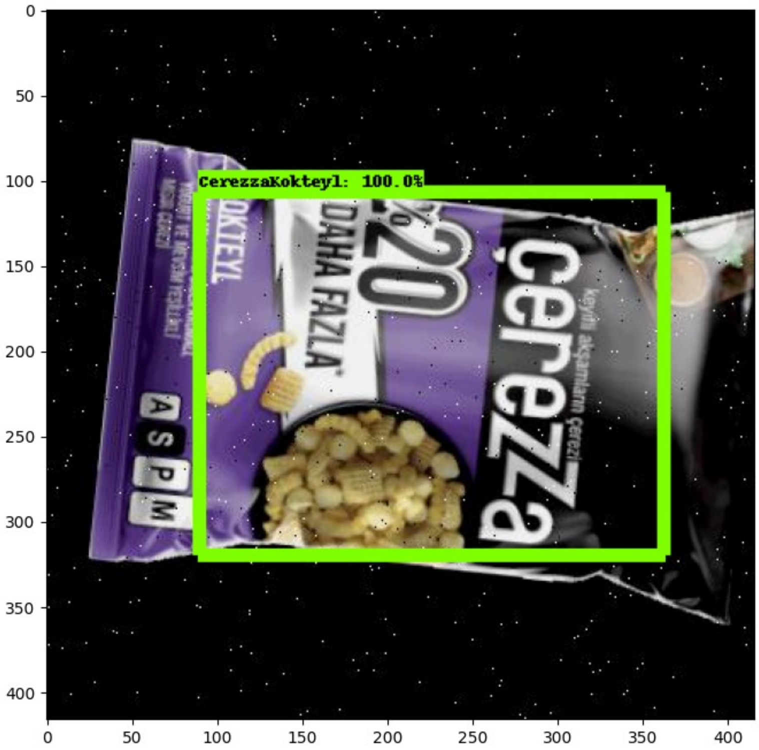


Figure 13: Object Detection Model Test Result

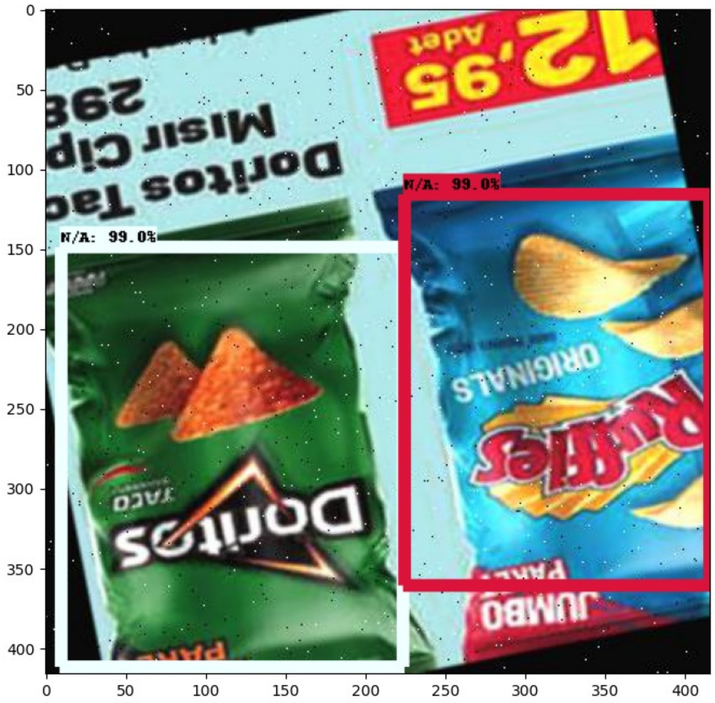


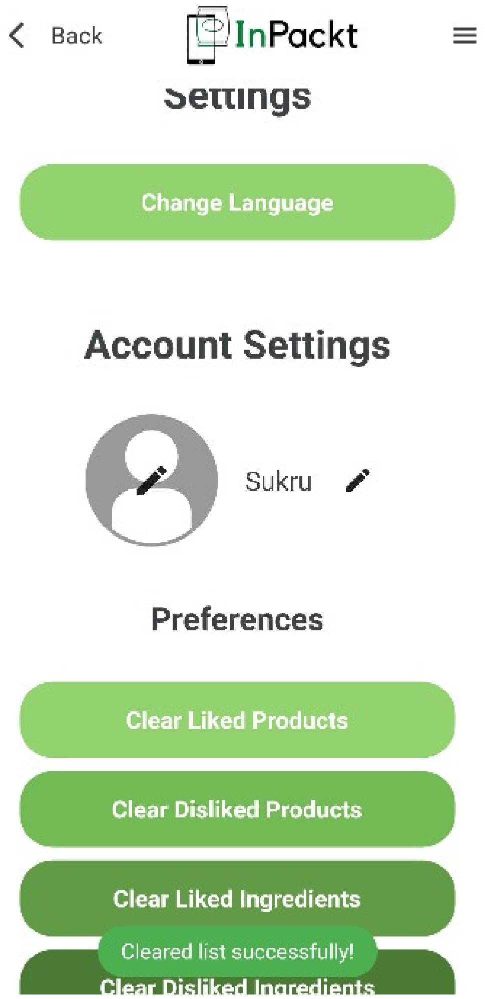
Figure 14: Object Recognition Model Failure

5.2. Code Reviews

Since every member of the team implements a unit and combines them together with the meeting, our working discipline is like Extreme Programming. Therefore, while combining the units and discussing the code in the meetings, every unit is reviewed by at least one member.

5.3. Testing with Different Emulators

We have tested InPackt in different emulators to test if the application is applicable to a variety of devices. Since nowadays the device pool is large and applications should fit to different sizes of the devices.



Cleared list successfully!

Figure 15: Redmi Note 8 - 6.3 inches

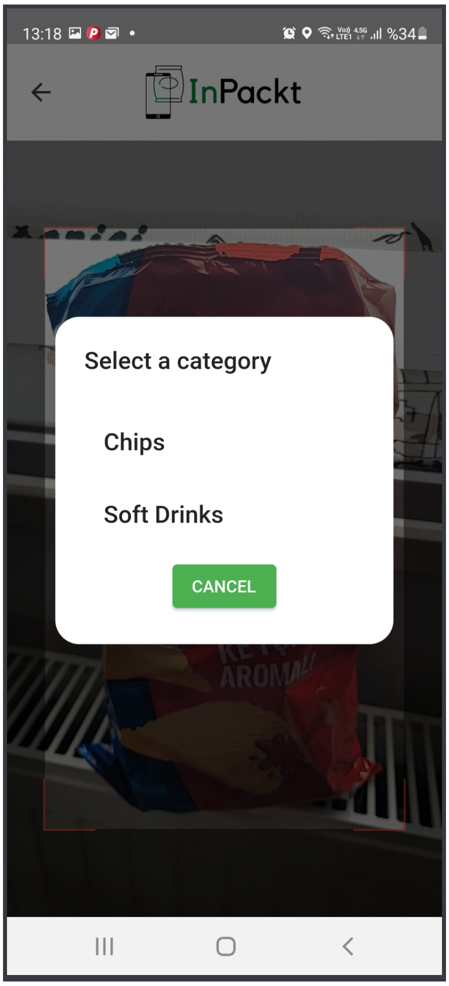
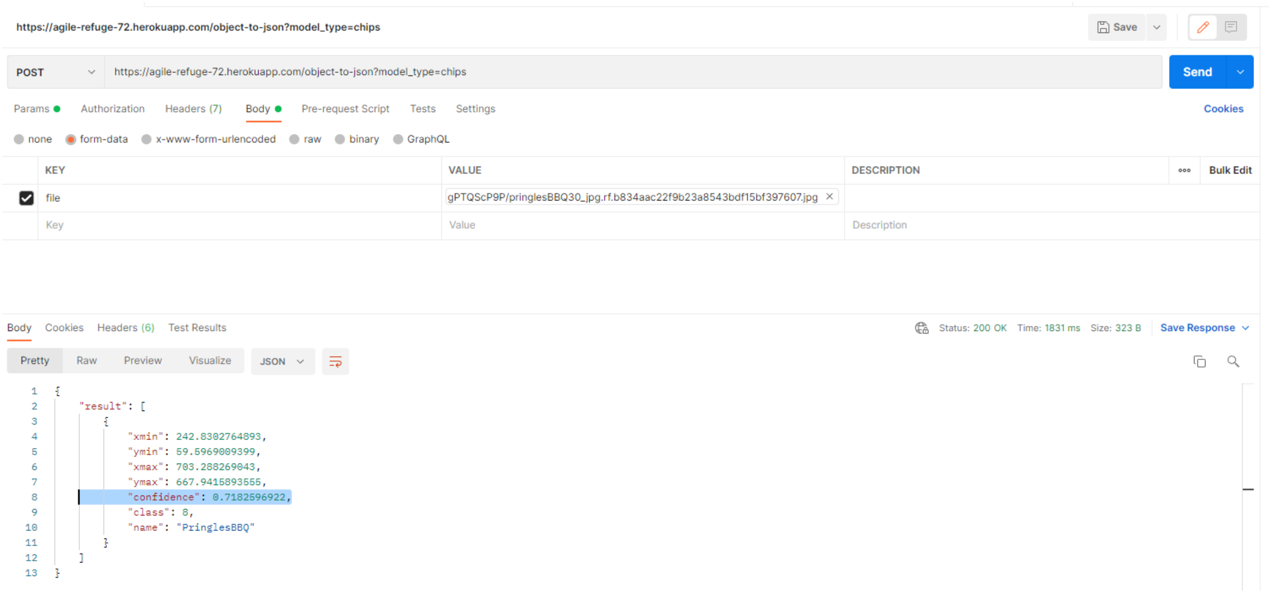


Figure 16: Samsung Galaxy S20 FE - 6.5 inches

5.4. Postman API Testing



https://agile-refuge-72.herokuapp.com/object-to-json?model_type=chips

POST https://agile-refuge-72.herokuapp.com/object-to-json?model_type=chips

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	file	gPTQScP9P/pringlesBBQ30_jpg.rf.b834aac22f9b23a8543bdf15bf397607.jpg	
	Key	Value	Description

Body Status: 200 OK Time: 1831 ms Size: 323 B

JSON

```
{
    "result": [
        {
            "xmin": 242.8302764893,
            "ymin": 59.5969009399,
            "xmax": 703.288269043,
            "ymax": 667.9415893555,
            "confidence": 0.7182596922,
            "class": 8,
            "name": "PringlesBBQ"
        }
    ]
}
```

Figure 17: Postman API Testing Screen Shot

We deployed our object recognition in Heroku and the models are tested on postman with heroku server.

6. Maintenance Plan and Details

Maintenance of the application depends on the database updates and bug resolutions. The application is server dependent due to the product information changes and product updates. Therefore, server updates of the application can be crucial for the maintenance and reliability of the application.

6.1. Server Maintenance

Free version of the Firebase is used to store user related information such as authentication data and preferences. Therefore, regular database maintenance is needed. For future applications, the current server can be replaced with a faster server depending on the number of active users. Since the product ingredient information will be stored locally no server needed for products.

6.2. Google Play Store

After the feedback that we got from the other instructors, companies and every potential user, the team planned to release the application on the Google Play Store. The application can also support itself via Google ads.

6.3. New Versions

New versions will be released depending on user feedback and new feature updates. One of the new versions will be about AR addition to the live camera. This feature will increase usability of the app since users do not have to take pictures of the products. The versions that can increase the success rate both for the product recommendation and product recognition can be released. In addition to feature version releases, depending on the bugs other versions can be released.

6.4. Issue Tracking

InPackt's version control will be via Github. Therefore, Github issues can be used to track and report the issues. Since InPackt will be publicly available as open source after the first release, anyone can report an issue to the Github repository. User feedback will be taken via a report system inside the app.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

As the name of the application implies, our project has an impact on many keystones of human life such as health, safety, welfare, global, cultural, social, environmental and economical. Effects of the algorithm in various categories are discussed below.

7.1.1. Public Health Factors

One of the main uses of the InPackt is to show the users unhealthy ingredients and help users to avoid harmful ingredients. Therefore, public health will be enhanced by InPackt as the user scanning products.

7.1.2. Public Safety Factors

The required data which are username, email password will not be shared anywhere and the safety of the information will be considered. The required data will be used for storing the preference list of the user.

7.1.3. Public Welfare Factors

InPackt doesn't have negative impact on public welfare. On the other hand, it can impact positively because InPackt will guide the users to do better shopping for themselves. Moreover, it will be a free application. Therefore, the application might increase public welfare with its guidance and affordability.

7.1.4. Global Factors

InPackt will focus on as many markets as possible in the world, so products will not be bought only from Turkey. Moreover, the application language will be English and this helps to be adapted easily to other countries. Therefore, by focusing on global market and main language of the application, InPackt has a global perspective as well.

7.1.5. Cultural Factors

InPackt will not be affected by cultural factors because it does not have cultural features to affect.

7.1.6. Social Factors

InPackt interacts directly with users instead of other users like social networks. Therefore, InPackt will not be affected by social factors.

7.1.7. Environmental Factors

The use of InPackt ease the shopping for the picky eaters such as vegetarians. Since eating less meat decreases CO2 and methane emissions, our application can have a positive effect on the environment in the future [x1].

7.2. Ethics and Professional Responsibilities

There are many ethical issues of InPackt and there are solutions to prevent ethical issues that may arise. First of all, the very first goal of the InPackt is to access ingredients of the product in an easy way and if there is any nutrition prohibited by the user in the product, suggest a similar product without that nutrition. Food companies might consider advertising on the app and want to feature their own product. To prevent this situation, advertised products will be displayed under the advertisement category and if they comply with certain healthy product principles. Also, user specification which is determined by user’s choices is added to the application. Therefore, when a product is suggested, it is a proper product in terms of the user's choices.

In terms of health InPackt have an impact on being more healthy since the application allows that the choice of banning harmful ingredients. If the use of the application becomes widespread, it is possible to affect food companies to produce more healthy food.

7.3. Judgements and Impacts to Various Contexts

Judgments and impacts of the InPackt application are divided into four different subheadings respectively global, economic, social and environmental.

7.3.1. Impact in Global Context

InPackt is a mobile-oriented multi-platform application that offers people easier and more personalized grocery shopping. This application aims to make a process easier that every individual has to do in order to continue his/her life in the global environment, whether the economic situation is good or bad. The language of the application is English and Turkish, making it suitable for use worldwide. Also, even though the products in the database are currently common in Turkey, if the database is expanded with future updates, the effect on global context will be high.

7.3.2. Impact in Economic Context

Even though the InPackt application has information such as the content, categories, names, calorie amounts of many products, it does not have any effect on the economic environment since it does not show price information and there is no advertising option in the current state of the application.

7.3.3. Impact in Social Context

It is thought that if the InPackt application becomes widespread among people due to the increasing consciousness about products and their ingredients and the spread of healthy life and balanced diets in social media, people will affect each other as a snowball effect and effectiveness of the social context at a high level.

7.3.4. Impact in Environmental Context

InPackt application can lead people to healthy eating and organic diets by removing artificial and harmful ingredients. Expanding nutrition in this way can lead to supporting an environmentally friendly production by changing the way products are produced

	Impact Level	**Impact Effect**
Impact in Global Context	9	It makes grocery shopping easier.
Impact in Economic Context	0	Since no information is given about product prices and there is no advertising option at the moment, it has no economic impact.
Impact in Social Context	7	-Helps to gain a healthy eating habit. -Helps to facilitate the selection of products which don't contain alcohol or pork meat.
Impact in Environmental Context	3	Helps people with organic diets that can result in a positive environmental effect.

Table 1: Impacts of Judgements

7.4. Teamwork Details

7.4.1. Contributing and Functioning Effectively on the Team

Member Name	Contribution in Work Package
Fadime Selcen Kaya	Specification Report Analysis Report High Level Design Report Database Development Data Processing
Vural Doğan Akoğlu	Specification Report Analysis Report High Level Design Report User Interface Frontend Development
Şükrü Can Erçoban	Specification Report Analysis Report High Level Design Report Object Recognition Development
Kadir Mert Laleci	Specification Report Analysis Report High Level Design Report User Interface Frontend Development
Kutsal Bekçi	Specification Report Analysis Report High Level Design Report Object Recognition Development

Table 2: Contribution in Work Package

7.4.2. Helping Creating a Collaborative and Inclusive Environment

We have used the Jira environment which is a planning and tracking program for software applications. We have assigned tasks with deadlines to keep track of the process. On the other hand, we used github to check and pull the current process of the project. In our project, we have a github for each project workforce that are frontend, backend, data processing. In the end we are planning to gather all of our sources in one github. Before each task, we met to make work sharing among us and save all of the meetings in a spreadsheet.

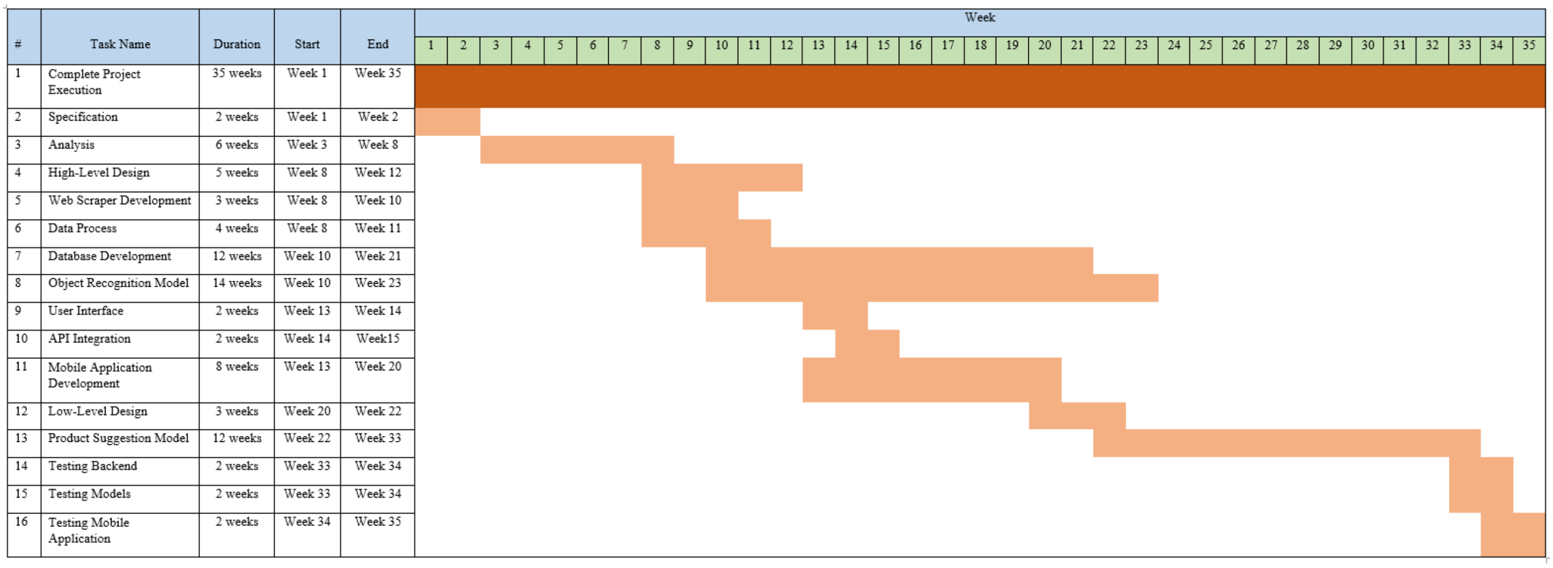
7.4.3. Taking Lead Role and Sharing Leadership on the Team

WP#	Work package title	Leader	Members involved
WP1	Analysis and High-Level Design Decisions UI API connection	Kadir Mert Laleci	Everybody
WP2	Database Development and Data Processing	Fadime Selcen Kaya	Everybody
WP3	Object Recognition Development	Şükrü Can Erçoban	Everybody
WP4	Machine Learning For Product Suggestion	Kutsal Bekçi	Everybody
WP5	Low-Level Design	Kadir Mert Laleci	Everybody
WP6	User Interface - Demo	Vural Doğan Akoğlu	Everybody

Table 3: Leader Role of Work packages

We have decided to assign each member a leadership role to be fair. The roles assigned to each person are presented in the table above. Other than the leader, each member contributed to each of the work packages, with the guidance and assistance of the leader.

7.4.4. Meeting objectives



#	Task Name	Duration	Start	End
1	Complete Project Execution	35 weeks	Week 1	Week 35
2	Specification	2 weeks	Week 1	Week 2
3	Analysis	6 weeks	Week 3	Week 8
4	High-Level Design	5 weeks	Week 8	Week 12
5	Web Scraper Development	3 weeks	Week 8	Week 10
6	Data Process	4 weeks	Week 8	Week 11
7	Database Development	12 weeks	Week 10	Week 21
8	Object Recognition Model	14 weeks	Week 10	Week 23
9	User Interface	2 weeks	Week 13	Week 14
10	API Integration	2 weeks	Week 14	Week15
11	Mobile Application Development	8 weeks	Week 13	Week 20
12	Low-Level Design	3 weeks	Week 20	Week 22
13	Product Suggestion Model	12 weeks	Week 22	Week 33
14	Testing Backend	2 weeks	Week 33	Week 34
15	Testing Models	2 weeks	Week 33	Week 34
16	Testing Mobile Application	2 weeks	Week 34	Week 35

Figure 18: Gantt Chart

The gantt chart given above shows the objectives determined in the analysis report. Although we have completed these determined objectives in general terms, we have not been able to complete them between the expected time periods, and the parts that cannot be completed are as follows, respectively. While it has been determined that all products will be recognized with a single model in object recognition, there are different models under different categories in our currently working project. Searching the product audibly, searching the product by scanning its barcode, selection of categories determined in the user preference list (vegan, glucose and lactose free) were not met. Also, we stated that we could use AR in the project if we had time, but we could not afford it. Apart from these, we have successfully completed the objectives we set about our project in the analysis report.

7.5. New Knowledge Acquired and Applied

At the beginning of the project, the team decided to implement InPackt on Flutter because it is used for cross platform applications from a single codebase. Since no one in the team has experience with Flutter, we have followed the Flutter documentation provided by

Google. In addition to Flutter, we have searched for object recognition tools like YOLOv5 and tensorflow lite. However, integration of these tools to Flutter was challenging since Flutter can be considered new according to its competitors. There were limited tutorials and unanswered StackOverFlow questions. For the first demo, we used tensorflow lite. It has slower model training and poor resources of integration to Flutter. As a final decision, we switched to YOLOv5 for our object recognition model. YOLOv5 has faster model training and we used an open source image labeling tool for labeling the data that is needed for training.

For the database of the application, we used Firebase which is a platform developed by Google for creating mobile and web applications. Since both Firebase and Flutter are developed by Google, integrating these two platforms is easier. Again follow the manuals of these two platforms to learn the integration and management of the database. The free version of firebase is enough to store the limited user and product information for the Final Demo of InPackt.

8. Conclusion and Future Work

In the final report of the CS491/492 senior design project, which we made during our senior year at Bilkent, as 5 computer engineering students, we represent in detail about the final state of our project, which we worked on for two semesters. After briefly explaining our project that provides easy access to the contents of packaged products related to market shopping, we named InPackt, and it enables personalized shopping according to user preferences. The requirement details about the project are shown in 3 sections: functional, non-functional and pseudo-requirements. The design parts of the InPackt project, which were mentioned separately in previous reports, were combined and updated, and the final architecture and design were determined as different diagrams, packages and layers. The development and implementation of the InPackt application was explained in detail, including tools, programming languages, APIs and libraries. The tests performed to control the operation of the application as the strategies and tools used for these tests were examined. The maintenance plan of the application is explained under different headings, including the server, new versions, and the store which app can be downloaded. Titles such as Consideration of various factors in engineering design, ethics and professional responsibilities, judgments and impacts to various contexts were written in detail as a result of research conducted throughout the two semesters. The importance of teamwork, the areas in which the group members lead and their contributions to the project were mentioned. The future work of the InPackt project is to expand the database for worldwide products, improve the object recognition model, and enable many more products to be recognized with high accuracy. If the application reaches the desired level, it will be installed in the Google play store.

9. References

[1] “Yolov5 Tutorial” Github. Accessed May 2, 2022.

https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb

[2] “Python” Python. Accessed May 2, 2022.

https://www.python.org

[3] “Firebase” Firebase. Accessed May 2, 2022.

https://firebase.google.com/?gclid=CjwKCAjwjtOTBhAvEiwASG4bCNj5rlfBrxEtCc50QTWF9ZWL1M8m32PSrGAbs67JKndd4Hzwv4BxtRoCLX8QAvD_BwE&gclsrc=aw.ds

[4] “You Asked: Should We All Go Vegetarian or Vegan to Reduce Our Carbon Footprint?” Colombia Climate School. Accessed May 2, 2022.

https://news.climate.columbia.edu/2019/09/26/vegetarian-vegan-diets-climate-change/