Bilkent University

Department of Computer Engineering

Senior Design Project

InPackt

Low-Level Design Report

February 28, 2022

Şükrü Can Erçoban - 21601437

Kadir Mert Laleci - 21601960

Vural Doğan Akoğlu - 21602479

Kutsal Bekçi - 21602163

Fadime Selcen Kaya - 21801731

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

1.0 Introduction

In today’s world, packaged foods have an important place in the food industry. Due to the institutionalization of the markets and the mass production of food products, many products have become accessible from everywhere in a packaged way. Access to many products in one place also significantly reduced shopping time. In order to keep up with the speed of today's world, the shopping time has been shortened. Therefore, it is a waste of time and difficult to browse the contents of packaged products written in small fonts during shopping, when many products are purchased at the same time. If the customer is allergic to a particular ingredient, shopping becomes a more demanding and time-consuming task and finding a product that is similar but without unwanted ingredients is like looking for a needle in a haystack.

InPackt has emerged to access the contents of the packaged product more easily and in a shorter time. The application allows users to scan the product using the phone camera so that in real time the application recognizes the product and lists its contents. Thus, the contents of the packaged product can be easily accessed without reading the small fonts one by one. The user can list the contents that need attention, and if these contents are in the scanned product, the application can immediately give a warning and suggest a new product specific to the user according to the previous preferences of the user. This application will make shopping easier and will allow you to buy by knowing what's inside the products.

InPackt also provides personalized recommendations to the user. Thus, products are listed according to the user's preferences and user satisfaction increases.

In this report, there are design trade-offs, high level explanations of packages of InPackt and class interfaces. All of these are explained well.

1.1 Object Design Trade-Offs

While developing InPackt, it was thought that some features should be more prominent and more important. Some other features were sacrificed to improve certain features of Inpackt. Every trade-off had a reason and we've discussed it in detail below.

1.1.1 Compute Time - Accuracy Rate

The model created for the object recognition part of the application is trained with many photos of the products. The number of photos and products is extending the training period and the model is growing. To avoid this, we trained the model with a certain amount of photos. We have reduced the computation time with little compromise in accuracy.

1.1.2 Cost -Performance

The application uses cloud services for performance and size. In addition, the performance of object recognition may vary depending on the specifications of the smartphone used by the user. As the number of products increases, the need for server performance increases. For now, it is enough for 200 - 300 products, but as the number of products increases, it will not be enough and the cost will increase. In this case, cost will be sacrificed for performance.

1.1.3 Space - Time

The application needs to have a large space for object recognition and customized recommendations and their databases, but this increases the size of the application and affects performance. When these big features of the application need to compare using a database, these processes are done in the cloud. Thus, while the application size is reduced, the uptime is increased as database response time is included. Application size matters, but response time is just as important. It is aimed to reach an acceptable rate of application size without prolonging the response time of the application. There is a balanced trade-off.

1.2 Interface Documentation Guidelines

The guidelines of interface documentation is below.

Class Sample
Class explanation
Attributes
● attributeName1: attributeType1 ● attributeName2: attributeType2
Functions
● function1(): Function explanation and return type if exists. ● function2(): Function explanation and return type if exists.

1.3 Engineering Standards

For the diagrams in this report, UML Diagrams are used for diagrams and standards of UML Diagrams are followed. For citation, IEEE citation sta

ndards are used.

1.4 Definitions, Acronyms and Abbreviations

1.4.1 Definitions

Training: For object detection, Yolov4-tiny model is used as a pre-trained model and it is trained in darknet framework.

Machine Learning Model: Machine learning is a model that learns to reach certain results by using a set of data with an algorithm provided to it.

1.4.2 Acronyms and Abbreviations

UML: Unified Modeling Language.

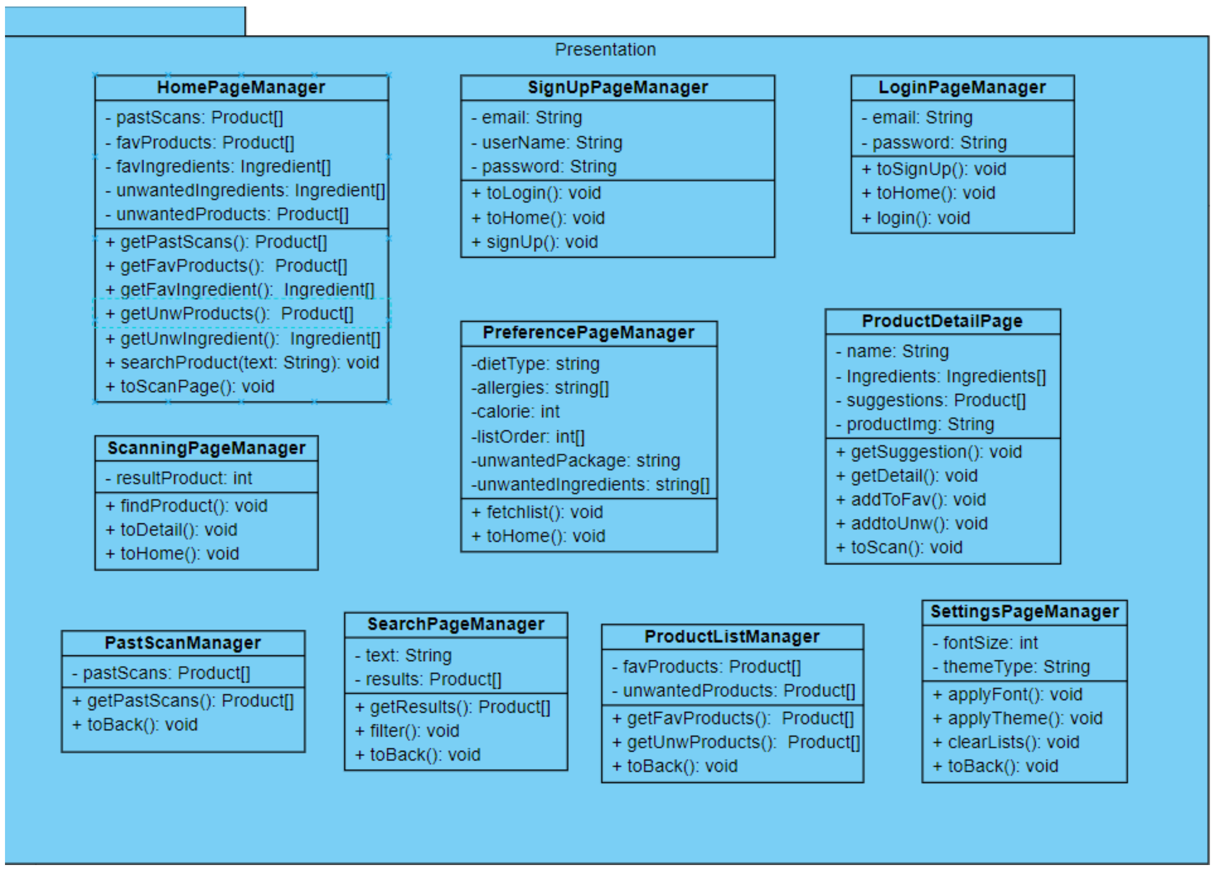
IEEE: Institute of Electrical and Electronics Engineers.

2.0 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.

2.1 Presentation Package

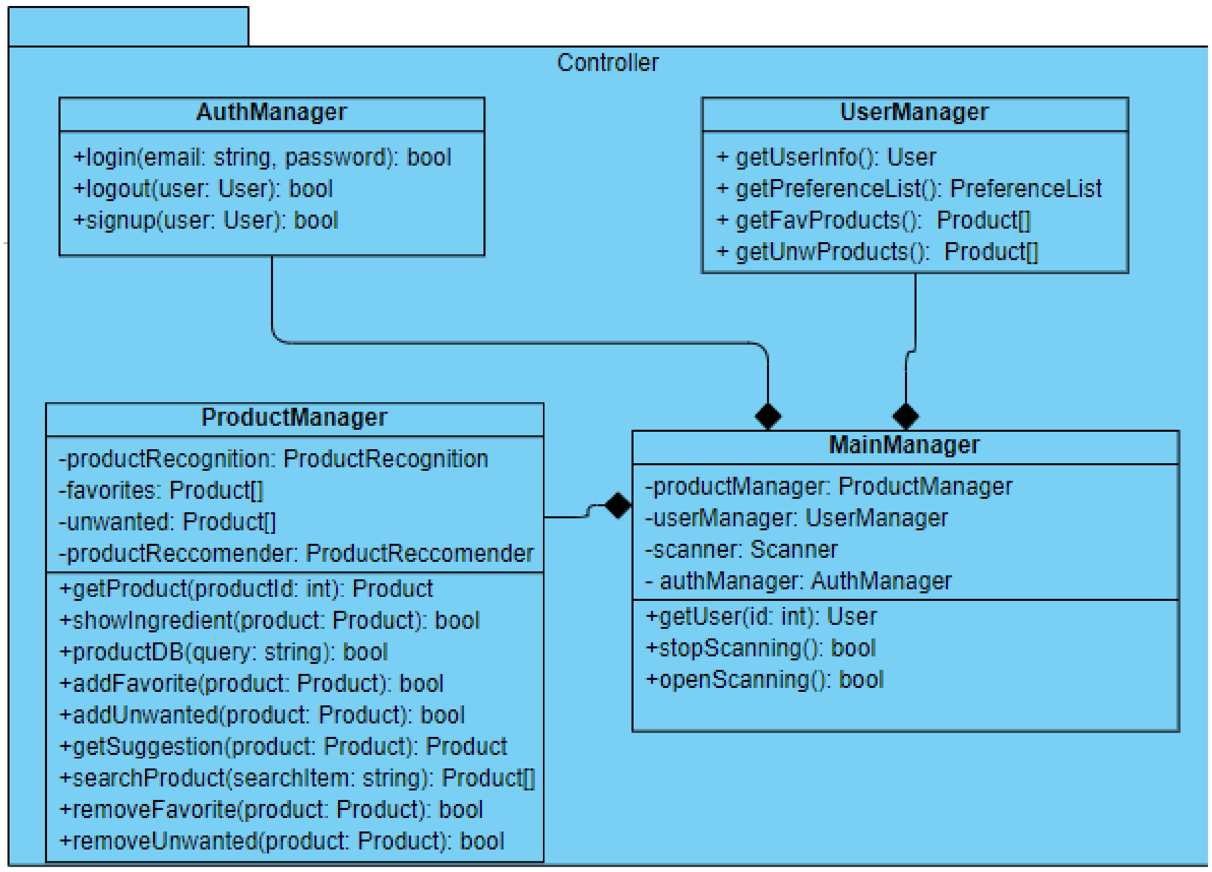
The presentation package is responsible for receiving and displaying the inputs which come from the logic layer and sending the actions and requests to the logic layer.



2.2 Logic Package

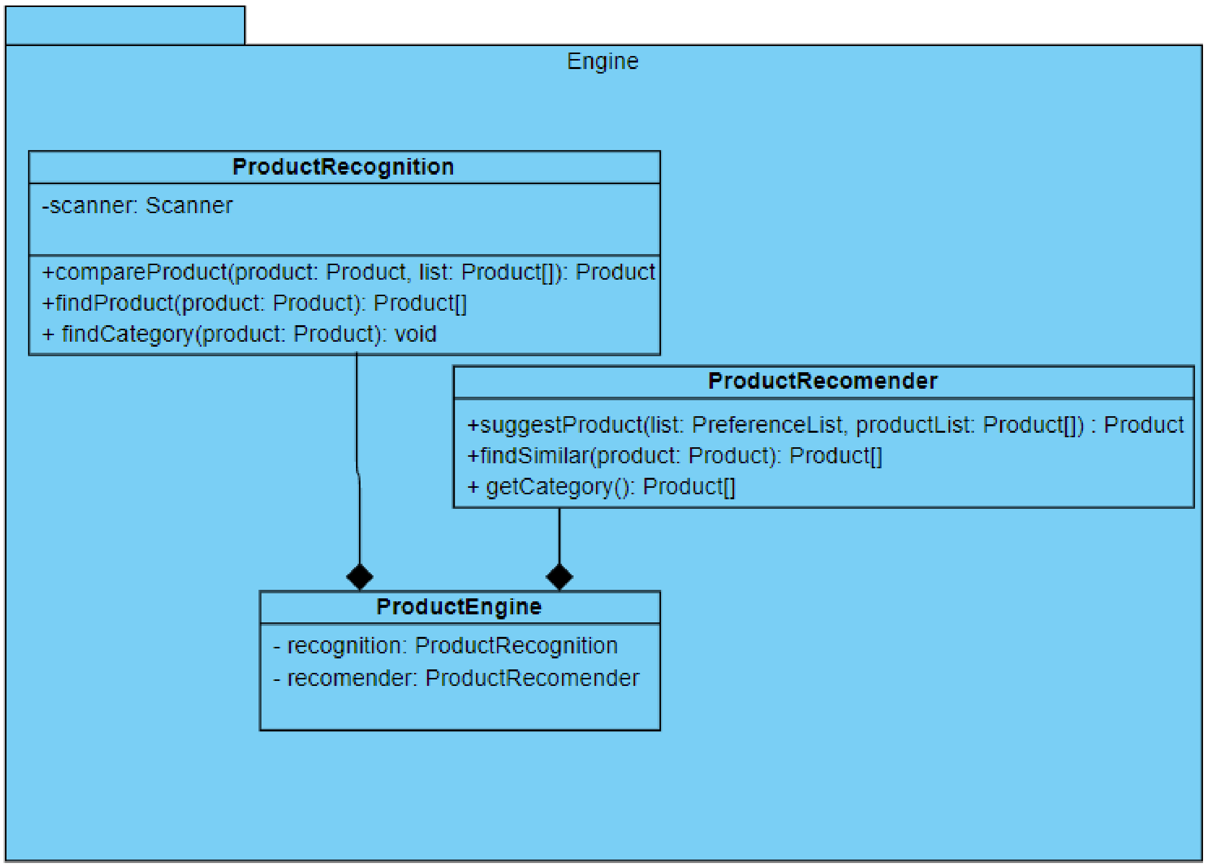
2.2.1 Controller Package

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

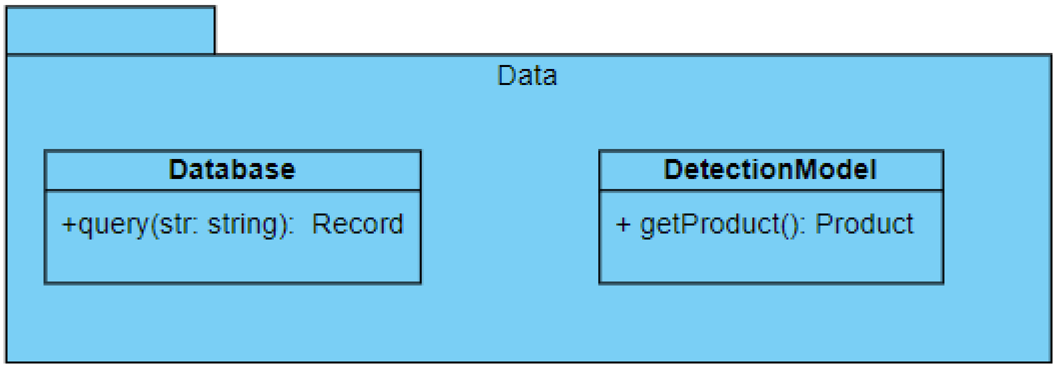
2.2.2 Engine Package

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, ProductRecomender 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

2.3 Data Package

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



3.0 Class Interfaces

3.1 Model

3.1.1 Presentation

3.1.1.1 Home Page Manager

Class HomePageManager
This class allows the user to switch between pages.
Attributes
● pastScans: Product[] ● favProducts: Product[] ● favIngredients: Ingredient[] ● unwantedIngredients: Ingredient[] ● unwantedProducts: Product[]
Functions
● getPastScans(): This function returns the list of the past scanned products of the user as Product[]. ● getFavProducts(): This function returns the user's favorite products as Product[]. ● getFavIngredient(): This function returns the user's favorite ingredients as Ingredient[]. ● getUnwProduct(): This function returns unwanted products of the user as Product[]. ● searchProduct(text: String): This function searches for a specific product and returns the product. ● toScanPage(): Opens the scanning page of the application.

3.1.1.2 SignUp Page Manager

Class SignUpPageManager
This class allows users to register the system of InPackt application

Attributes
<ul><li>email: string</li><li>userName: string</li><li>password: string</li></ul>
Functions
<ul><li>toLogin(): This function opens login page</li><li>toHome(): This function opens the home page</li><li>signUp(): This function registers user to system</li></ul>

3.1.1.3 Login Page Manager

Class LoginPageManager
This class allows users to enter the system of ınpackt application
Attributes
<ul><li>email: string</li><li>password: string</li></ul>
Functions
<ul><li>toSignUp(): This functions opens signup page</li><li>toHome(): This functions opens the home page</li><li>login(): This function allows users to login the system</li></ul>

3.1.1.4 Scanning Page Manager

Class ScanningPageManager
This class allows the user to scan a product to see its ingredients
Attributes
<ul><li>resultProduct: int</li></ul>
Functions
<ul><li>findProduct(): This functions find scanned product from database</li></ul>

● toDetail(): This function opens product detail page ● toHome(): This function opens the home page

3.1.1.5 Preference Page Manager

Class PreferencePageManager
This class shows the preference page of the user.
Attributes
● dietType: string ● allergies: string[] ● calorie: int ● listOrder: int[] ● unwantedPackage: string ● unwantedIngredients: string[]
Functions
● fetchlist(): This function updates the data and sends current data to the backend application. ● toHome(): This function opens the home page.

3.1.1.6 Product Detail Page

Class ProductDetailPage
This class shows ingredients of scanned products
Attributes
● name: string ● ingredients: Ingredients[] ● suggestions : Products[] ● productImg: string

Functions
<ul><li>getSuggestion(): This function takes alternative products to the scanned product that do not match the user profile.</li><li>getDetail(): This function takes the ingredients of products from the database.</li><li>addToFav(): This function allows users to make a product to their favorite list.</li><li>addToUnw(): This function allows users to make a product to their unwanted list.</li><li>3toScan(): This function opens the scanning page.</li></ul>

3.1.1.7 Past Scan Manager

Class PastScanManager
This class shows the preference page of the user.
Attributes
<ul><li>pastScans: Products[]</li></ul>
Functions
<ul><li>getPastScans(): This function returns the past scans of the user as Product[].</li><li>toBack(): This function opens the previous page.</li></ul>

3.1.1.8 Search Page Manager

Class SearchPageManager
This class allows users to search products from database
Attributes
<ul><li>text: string</li><li>results: Product[]</li></ul>

Functions
● getResults(): This function shows the filtered search results to the user. ● filter(): This function filters the search results according to the user preferences. ● toBack(): This function opens the previous page.

3.1.1.9 Product List Manager

Class ProductListManager
This class shows users to their favorite and unwanted products' lists
Attributes
● favProducts: Product[] ● unwantedProducts: Product[]
Functions
● getFavProducts(): This function returns the user's favorite products as Product[]. ● getUnwProducts(): This function returns the user's choice of unwanted products as Product[]. ● toBack(): This function opens the previous page.

3.1.1.10 Settings Page Manager

Class SettingsPageManager
This class shows the settings page of the application.
Attributes
● frontSize: int ● themeType: String

Functions
● applyFront(): This function returns the past scans of the user as Product[]. ● applyTheme(): This function applies the theme on the pages. ● clearLists(): This function clears the lists to reset user specifications. ● toBack(): This function opens the previous page.

3.1.2 Controller

3.1.2.1 Auth Manager

Class AuthManager
This class handles the authentication functions of the user.
Functions
● login(email: string, password): This function returns true or false based on the success of logging in. ● logout(user: User): This function returns true or false based on the success of logging out. ● signup(user: User): This function returns true or false based on the success of signing up.

3.1.2.2 User Manager

Class UserManager
This class handles the user-specific functionalities and requests.
Functions
● getUserInfo(): This function returns the user information requested as User.

● getPreferenceList(): This function returns the preference list of the user as PreferenceList. ● getFavProducts(): This function returns the favorite products of the user as Products[]. ● getUnwProducts(): This function returns the unwanted products of the user as Products[].

3.1.2.3 Product Manager

Class ProductManager
This class handles the requests related to products sent by the user.
Attributes
● productRecognition: ProductRecognition ● favorites: Product[] ● unwanted: Product[] ● productRecommender: ProductRecommender
Functions
● getProduct(productId: int): This function returns the product with the specified id as Product. ● showIngredient(product: Product): This function shows the ingredients of the specified product and returns whether it was successful or not as a bool. ● productDB(query: string): This function executes the query given as string and returns whether it was successful or not as a bool. ● addFavorite(product: Product): This function adds the specified product to the user's favorites list and returns whether it was successful or not as a bool. ● addUnwanted(product: Product): This function adds the specified product to the user's unwanted list and returns whether

it was successful or not as a bool. ● removeFavorite(product: Product): This function removes the specified product from the user's favorites list and returns whether it was successful or not as a bool. ● removeUnwanted(product: Product): This function removes the specified product from the user's unwanted list and returns whether it was successful or not as a bool. ● getSuggestion(product: Product): This function gets the suggestion from the engine and returns it as a Product. ● searchProduct(searchItem: string): This function searches among the products for the given search and returns the result as Product[].

3.1.2.4 Main Manager

Class MainManager
This class manages all of the other managers and the scanner.
Attributes
● productManager: ProductManager ● userManager: UserManager ● scanner: Scanner ● authManager: AuthManager
Functions
● getUser(id: int): This function returns the user with the specified id as User. ● stopScanning(): This function stops the scanning functionality and returns whether it was successful or not as a bool. ● openScanning(): This function opens the scanning functionality and returns whether it was successful or not as a bool.

3.1.3 Engine

3.1.3.1 Product Recognition

Class ProductRecognition
This class is tasked with recognizing a product.
Attributes
● scanner: Scanner
Functions
● compareProduct(product: Product, list: Product[]): This function gets the products which have similarity rates and returns the highest similarity.   ● findProduct(product: Product): This function finds the similar products for searching action and returns the list of products.   ● findCategory(product: Product): This function finds the category of the specified product.

3.1.3.2 Product Recommender

Class ProductRecommender
This class is tasked with the recommendation functionalities.
Functions
● suggestProduct(list: PreferenceList, productList: Product[]): This function finds the suggested product based on user preferences and returns it as Product.   ● findSimilar(product: Product): This function finds similar products to the specified product and returns Product[].   ● getCategory(): This function gets the category of the scanned product and returns Product[].

3.1.3.3 Product Engine

Class ProductEngine
This class hadles the functionalities of recognition and recommendation.
Attributes
● recognition: ProductRecognition ● recommender: ProductRecommender