Bilkent University

Department of Computer Engineering

Low Level Design Report

ShopCart

Ahmet Kaan Uğuralp, Ahmet Işık, Furkan Ahi, Mehmet Yaylacı, Ravan Aliyev

Supervisor: Asst. Prof. Dr. A. Ercüment Çiçek

Jury Members: Asst. Prof. Dr. A. Ercüment Çiçek, Asst. Prof. Dr. Shervin Arashloo, Asst. Prof. Dr. Hamdi Dibeklioğlu

Low Level Design Report February 28, 2022

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

Abstract

ShopCart is a mobile application to help users to improve their shopping habits. People run out of products eventually they use these products and forget to buy them before they run out. Another problem is finding appropriate prices for their budget. Our application helps to solve this problem by allowing them to create shopping lists and add products to the list easily in 4 different ways: scanning barcode, scanning image, voice search and typing product name. Users can share these lists with their communities and the application notifies them when they leave home to remind them that they have to buy products which they are about to run out of. This mobile application makes people's life easier by improving their smart shopping practices.

1. Introduction

People have been going shopping in order to meet their needs. We all need to purchase food and beverages to survive. Storing the purchased food in refrigerators, cooking and ordering meals are all common activities. When ingredients are used for these purposes they run out eventually. People make the shopping list when the refrigerator is nearly empty. However, they may forget some of the products and this causes them to go shopping again. Since food prices keep rising, smart shopping becomes more challenging. Smart shopping means knowing what to buy and when. [1] Keypoint here is knowing the basics before going shopping, like what you need, the amount of each food needed and where you can find these products with the lowest price.
ShopCart is a mobile application which helps users to keep track of the products that run out and purchase reminders. The information about these products includes their monthly consumption frequency, the number of the days that it has been run out, and so on. So this application helps consumers to be a smart shopper by showing when each product is out of stock, amount of the needed product, which market offers the lowest price for each product and calculating the total spending on food. Before throwing away the depleted product, users scan its barcode, enter its name manually or scan a photo in order to create a shopping list. After the product is added to the shopping list, its numeric data depletion time and other information will be preserved

as a result of these methods. Users can create a “community” on the app if they live together. By the help of this feature users living together can create a shopping list of the community and each member can add and remove products from this list. Users can also edit the list after purchasing it to choose not to purchase it again. The goal of this app is to make people’s lives easier by improving their smart shopping practices. Our shopping practices have big implications for our health, our society and the environment and this app helps to raise living standards by improving consumption practices. [2]

In this paper, we provide the low level design of the paper. Firstly, object trade offs will be discussed and interface documentation guidelines and engineering standards which are followed while writing the report will be given. Later on, packages of the *ShopCart* application will be described. Lastly, this will be followed by the description of the class interfaces of our system.

1.1. Object design trade-offs

1.1.1. Understandability vs. Complexity

Our *ShopCart* application is designed to provide the users an opportunity to create different shopping lists, calculate the approximate prices, find the cheapest prices for each product and share these lists with different communities that are involved with only one application.

For this reason we have included multiple functions in order to achieve these goals which can be problematic for the users to learn how to use the application easily. For this reason we have limited the functionalities and design a user friendly user interface to find the appropriate balance between understandability and complexity.

1.1.2. Portability vs. Development Time

We want to increase the number of the users that we target by making a portable application on different operating systems such as iOS and Android. For this reason we decided to use React/React Native framework where this framework allows us to deploy the final product on different operating systems. Our decision will increase the development time when the trade off between portability and development time is considered, but we choose portability over development time.

1.1.3 Reliability vs Rapid Development

Our strategy is to start a rapid development process by the help of agile model development. This will help us to have some functions working in our hands which are not reliable to meet the project requirements. But, later on we will continue the development process step by step to increase its reliability and other functionalities. To conclude, for the early stages of the development process, rapid development is chosen but later on reliability will be favoured.

1.2. Interface documentation guidelines

The generic guidelines for class interfaces will be followed in this report. “ClassName” will show the class names. For the variables “variableName” standard name will be used and “methodName()” will be used for the method names. Constructors will be shown under “Methods and Attributes” and “ClassName()” will be used for naming. For the class interface, firstly class name and its description then methods will be written which will be followed by the attributes of that class. The class interface is shown below:

Class: ClassName	
Class description will be written here	
Methods and Attributes	
Methods and Attributes listed here: public methodName1() private methodName2() private int variableName1 public String variableName2	The functionality of the method 1 The functionality of the method 2

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

We will follow the IEEE standards for the citations and references in this report, since it is widely used in the software engineering field for the reports[3]. We will use UML guidelines for the diagrams, where UML is a general-purpose modelling language often used in computer science[4,5].

1.4. Definitions, acronyms, and abbreviations

- **UI:** User Interface
- **API:** Application Programming Interface
- **Server:** Houses the database and is the backend of the system. All logical operations on the data are done here.
- **Class:** “A generalised description of a project”
- **Controller:** Controller of the Model and View where gets input from the user and updates the data if needed [6].
- **Method:** Behaviour of the objects attributed to the class .
- **Model:** Data used by program, i.e. database, file, or a simple object [6].
- **Server:** A computer which provides information and services needed to the client..
- **View:** Displaying an object in the system to the user[6].

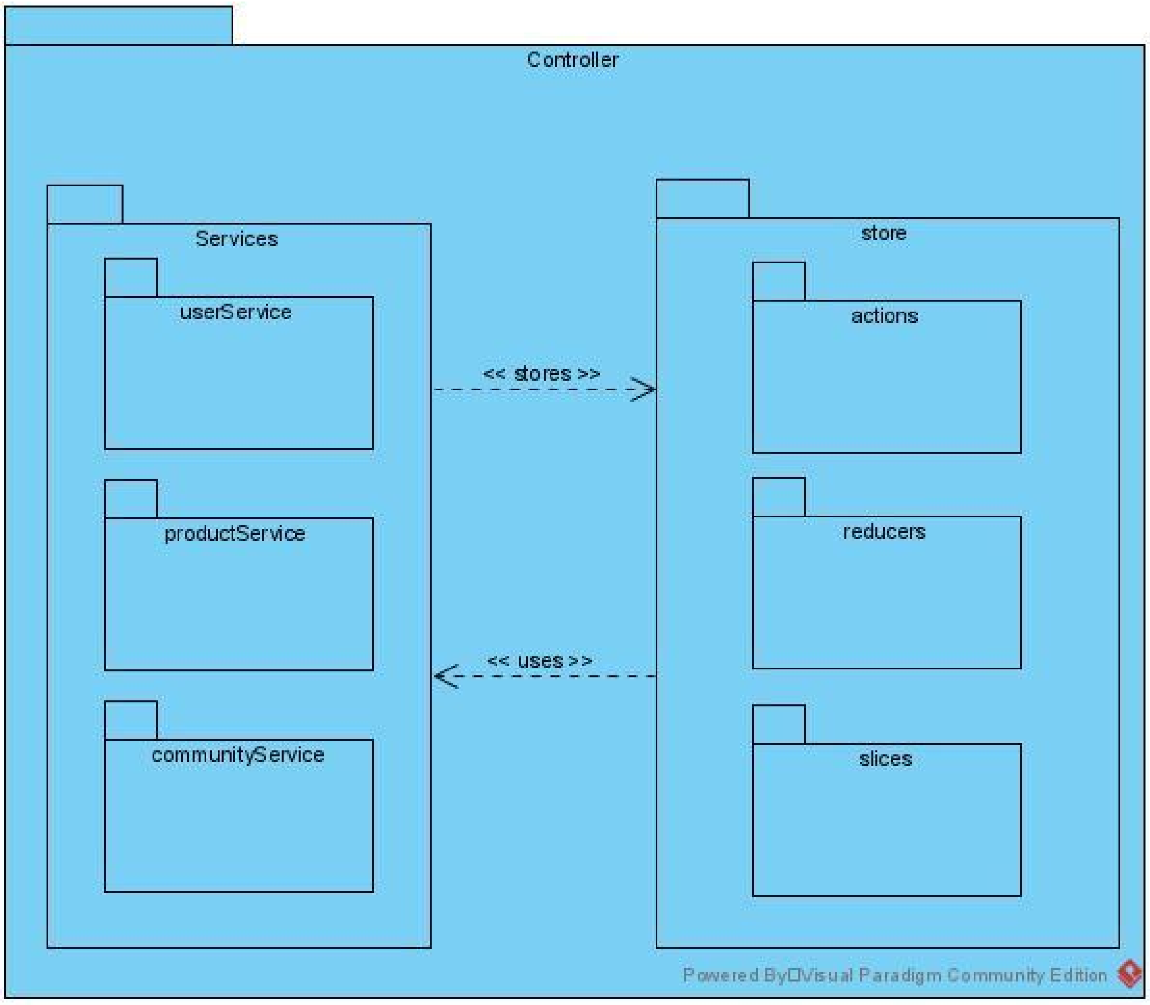
2. Packages

This section contains the package diagram of the *ShopCart* system. *ShopCart* is composed of two parts at the topmost level, Client and Server. Client has two components: Controller and View. Server also has three components: Models, Controller, and Data Fetcher. In this section, both the Client and the Server sides of our application will be generally analysed. How the classes are divided into different packages will be discussed. Individual explanations for the classes are provided in Class Interfaces section.

2.1. Client

2.1.1. Controller

The controller package is responsible for managing the operations on the client part, creating a set of objects and communication between services. There are two sub-packages in this part. The first of these is the services package, and the second is the store package. The controller package is as follows.



Services:

userService: used to create, manipulate and delete data about the users and send them to the store subpackage

productService: used to manage any current product(s) actions and sent them to store.

communityService: used to manage communities created by users and sent them to store.

Store:

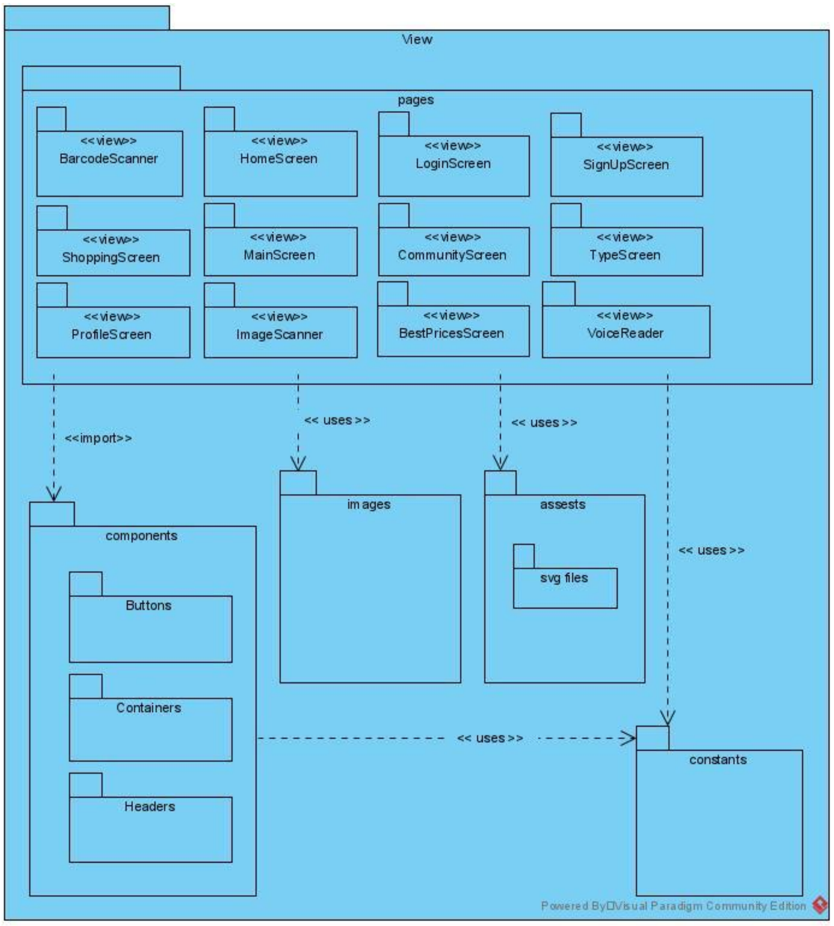
actions: performs login, logout and sign up operations using reducer instances and slice functions. It also stores data to the reducer using the methods in the slice while updating the states with slice methods.

reducers: contains functions to keep current state and reaches them via initialising in components

slices: contains the instances and functions that are kept reaching the store package.

2.1.2. View

To summarise the View package in the simplest way, it is responsible for the visualisation and operations between pages. It contains images, assets, page designs and constants of the front-end. It has five subpackages: pages, components, images, assets, constants. The view package is as follows.



pages: contains screens of the application. It uses the other subpackages' data below.

components: has three subpackages; buttons, containers, headers. This data is imported by pages.

images: has image data of the app.

assets: has .svg files of the app.

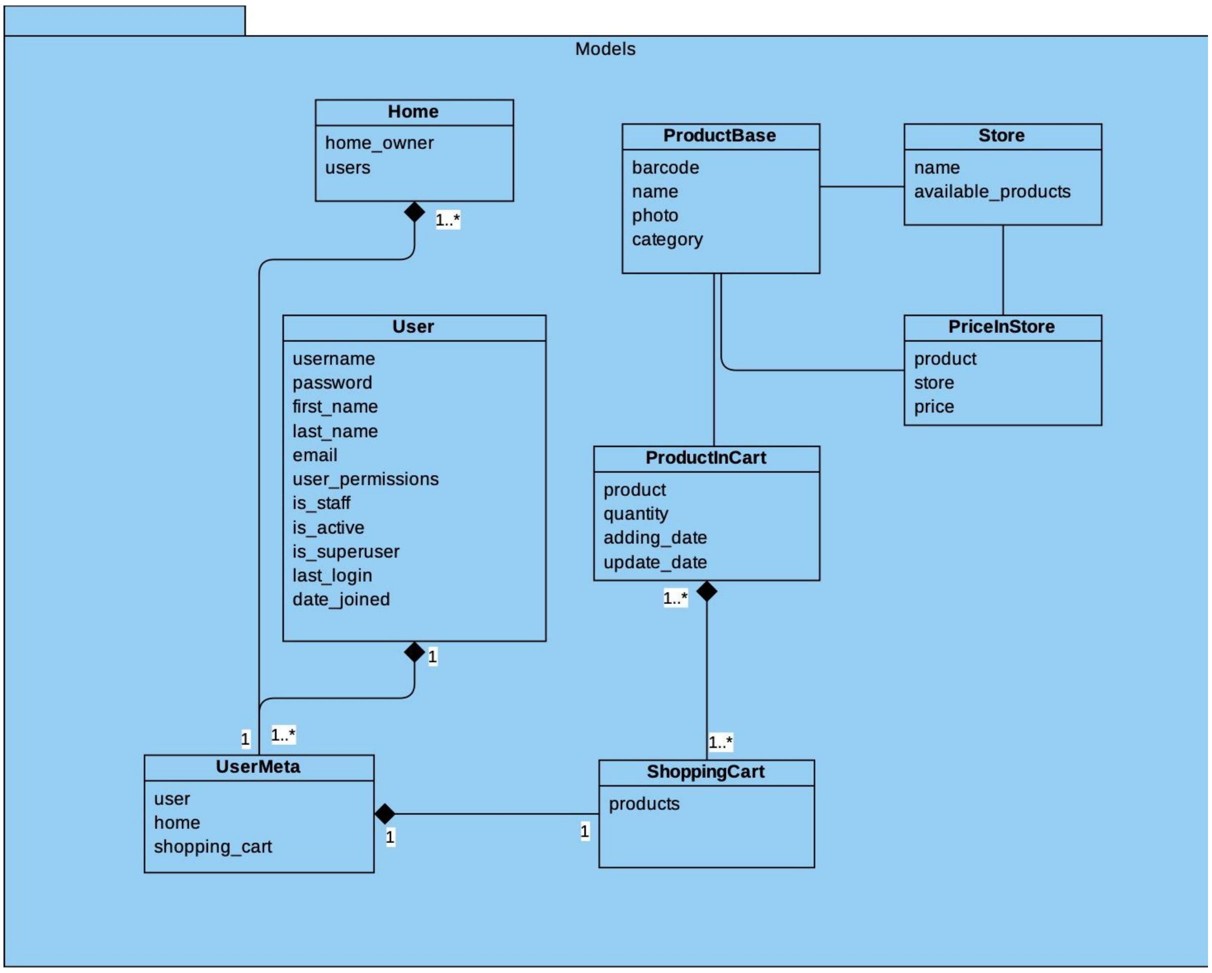
constants: contains the app constants.

2.2. Server

Server package contains three packages: Models, Controller and DataFetcher.

2.2.1. Models

This subpackage keeps the information about the database.



Home: keeps the information of the owners and the users connected to a home.

UserMeta: connects the users, homes and shopping carts.

ShoppingCart: keeps which ProductsInCart objects are in a given user's cart.

ProductsInCart: keeps the products and the amount of the products.

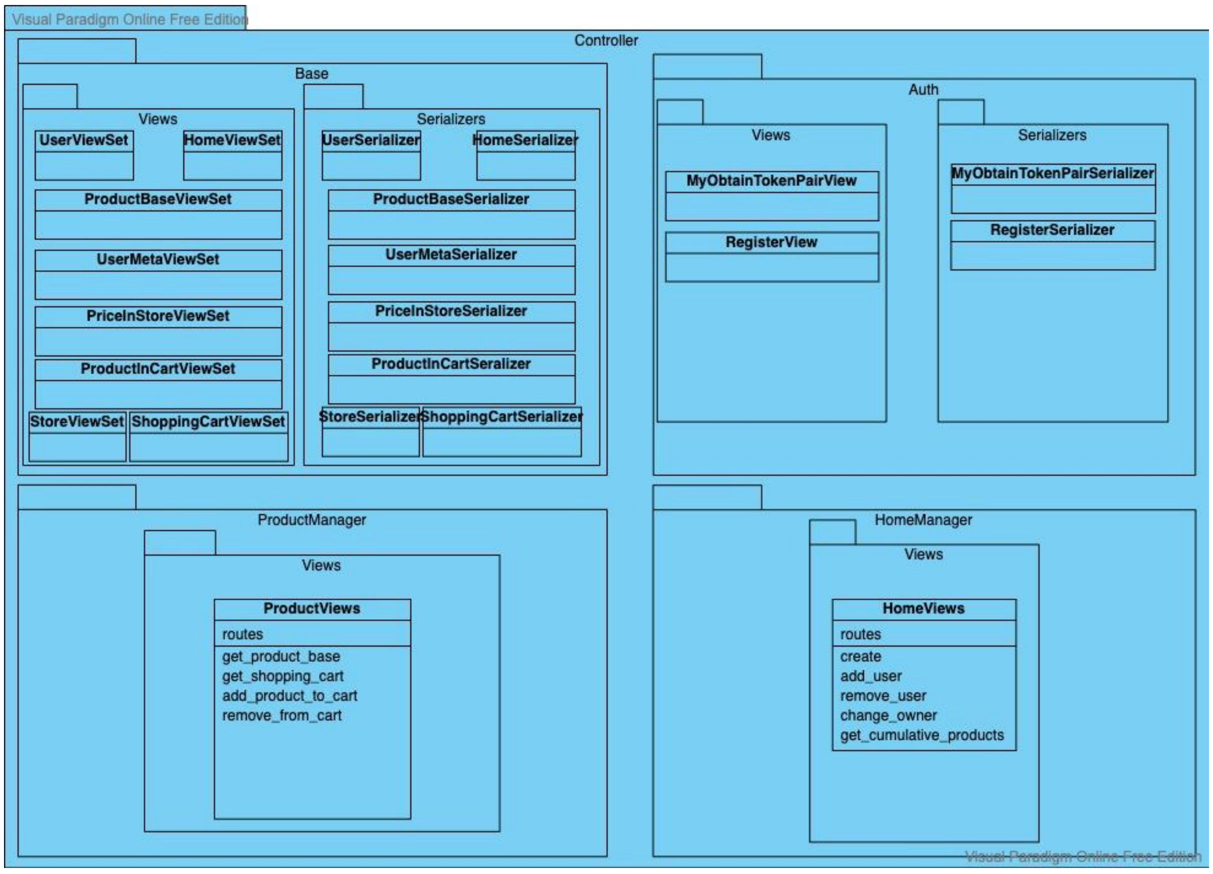
PriceInStore: keeps the prices and the stores of the products.

Store: keeps the information of the stores that sell the products.

ProductBase: is connected with the database our system keeps. This class connects barcode numbers, names and prices of a given product.

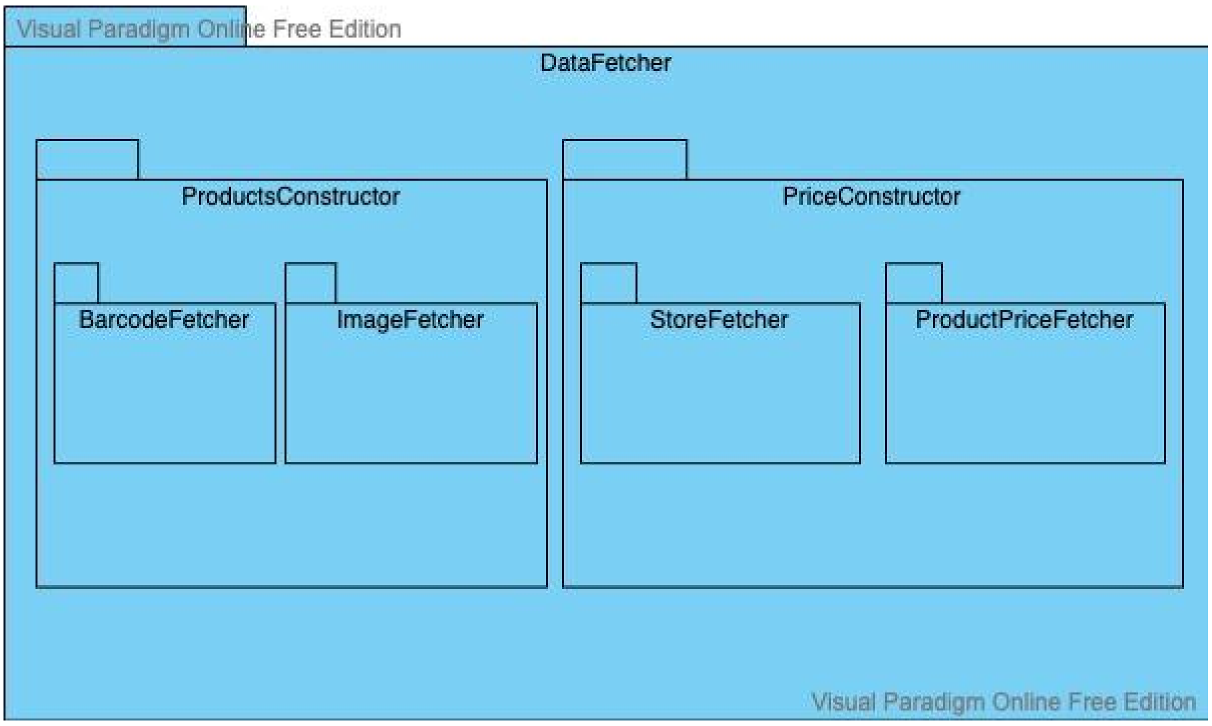
2.2.2. Controller

This subpackage keeps the bases of the database and also communicates with View and Model.



2.2.3. Data Fetcher

This subpackage scrapes data from “cimri.com” and “barkodoku.com”.



3. Class Interfaces

3.1. Client

3.1.1. Pages

Class: View	
This class is used to manage the pages on the server side.	
Methods and Attributes	
<<view>> BarcodeScanner	A screen with camera interface to scan any barcode of a product.
<<view>> HomeScreen	
<<view>> LoginScreen	Classic login screen with the Google login option and create an account tab.
<<view>> SignUpScreen	Sign up page.
<<view>> ShoppingScreen	Shows products and total amount of your cart.
<<view>> MainScreen	Main screen with some sections such as categories, featured products.
<<view>> CommunityScreen	Shows your communities and has a section to create a new community.
<<view>> TypeScreen	Product searching with manual typing screen.
<<view>> ProfileScreen	

<<view>> ImageScanner <<view>> BestPricesScreen <<view>> VoiceReader	Profile screen for users that has personal information A screen for searching with an image of a product. A screen Shows best prices of current added products. A screen for searching with a user voice.

3.2. Server

3.2.1. Models

Class: User	
This class is used to manage the users on the server side.	
Methods and Attributes	
private String username	This is a unique username for every user.
private String password	
private String first_name	
private String last_name	
private String email	This is a unique email for every user.
private boolean is_staff	If a user has staff permissions.
private boolean is_active	If the user is online.
private boolean is_superuser	If the user has superuser permissions.
private String last_login	
private String date_joined	

3.2.2. Controller

Class: ProductManager	
This class is used to manage the products on the server side.	
Methods and Attributes	
get_product_base(String barcode): ProductBase	Returns the ProductBase of a product matching the barcode
get_shopping_cart(User user): List<Product>	This will return the ProductBase list the user has in their cart.
add_product_to_cart(ProductBase)	Add a product to cart matching the ProductBase.
remove_from_cart(ProductBase)	Remove a product from a cart matching the ProductBase.

Class: HomeManager
Manages the home. Users with necessary permission can add and remove users. This class can also return the products in a home.
Methods and Attributes

public create(String home_name)	This method adds a home to the database.
public add_user(String username)	This method adds a user to the home. The user calling this method should be the owner of the home.
public remove_user(String username)	This method removes a user to the home. The user calling this method should be the owner of the home.
public change_owner(String username)	This method changes the user of the home. The user calling this method should be the owner of the home or have the necessary authorization.
public get_cumulative_products(String username): List<Product>	Returns the products of the home.

Class: AuthenticationManager	
This class concerns the authentication of the users.	
Methods and Attributes	
public login(String username, String password)	This method tries to login to the system. An error is raised if the user does not exist.

public signup(String username, password, String first_name, String last_name, String email)	This method adds another user. An error is raised if the given username or email matches to another user.

3.2.3. Data Fetcher

Class: DataFetcher	
This class is used for getting data from various sites using scraping libraries.	
Methods and Attributes	
private getSiteText(String url): String	This method returns the HTML text of a site and takes the URL.
public getProductData(String barcode): List<String>	Returns the data found scraping matching the barcode given.
private List<String> productData	The data found scraping “cimri.com”
private bs4.element productSoup	BeautifulSoup element with the product site.
private bs4.element photoSoup	BeautifulSoup element with the product photo.
private String storeName	A store URL found to be selling the product.
private int price	Price of the product.
private bs4.element barcodeSoup	BeautifulSoup element with the URL of “barkodoku.com” added with the barcode number.
private String productName	Name of the product matching the barcode.

4. Glossary

Machine Learning: “Study of computer algorithms that can improve automatically through experience and by the use of data [8]”

Computer Vision: “a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images [9]”

API: Application Programming Interface.

Third Party Application

Tensorflow: E2E open source Machine Learning (ML) framework, used to architect and deploy ML applications [16].

MySQL: World’s most popular open source database [10].

DLib: C++ library containing machine learning algorithms and tools [11].

Django: High-level Python web framework used for both frontend and backend development [12].

React Native: A popular JavaScript-based mobile app framework that allows you to build natively-rendered mobile apps for iOS and Android. The framework lets you create an application for various platforms by using the same codebase.

SQLite: Claims to be the “most used database engine in the world” [13].

GitHub: A code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

OpenCV: Open source library used for computer vision applications.

Agile Development: software development methodologies centred round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organising cross-functional teams.

External Api services: (hepsiburada.com/api and trendyol.com/api)

AWS: Amazon Web Services. Claimed to be the world's most comprehensive and broadly adopted cloud platform [14].

S3 Bucket: Simple storage service available in AWS.

Expo-barcode-scanner: An advanced barcode-scanner written in Javascript and TypeScript [15].

References

[1] “Grocery Guide”, American Heart Association, 2020, [Online] Available: https://www.heart.org/-/media/aha/recipe/pdf-files/grocery-guide-english-shopping-budget.pdf?la=e. [Accessed: 24-Feb-2022].

[2] “Food consumption in UK”, Rand Corporation, 2020, [Online] Available: https://www.rand.org/content/dam/rand/pubs/research_reports/RR4300/RR4379/RAND_RR4379.pdf. [Accessed: 24-Feb-2022].

[3] “1016-1987 - IEEE Recommended Practice for Software Design Descriptions,” IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/document/565312. [Accessed: 07-Feb-2021].

[4] Opennetworking.org, 2021. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/UML_Modeling_Guidelines_V1.0.pdf.[Accessed: 24-Feb-2022].

[5] Amit, “All You Need to Know About UML Diagrams: Types and 5+ Examples,” Tallyfy, 26-Feb-2020. [Online]. Available: https://tallyfy.com/uml-diagram/#what_is_the_use_of_uml. [Accessed: 24-Feb-2022].

[6]“MVC,” MVC (Model-View-Controller) Definition. [Online]. Available: https://techterms.com/definition/mvc#:~:text=Stands%20for%20%22Model%2DView%2D,for%20developing%20modern%20user%20interfaces. [Accessed: 25-Feb-2022].

[8] “Machine learning,” Wikipedia, 04-Nov-2021. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Accessed: 24-Dec-2021].

[9] “What is Computer Vision?,” IBM. [Online]. Available: https://www.ibm.com/topics/computer-vision. [Accessed: 24-Dec-2021].

[10] “Tensorflow,” TensorFlow. [Online]. Available: https://www.tensorflow.org/. [Accessed: 24-Dec-2021].

[11] “MySQL documentation,” MySQL. [Online]. Available: https://dev.mysql.com/doc/. [Accessed: 24-Dec-2021].

[12] Django. [Online]. Available: https://www.djangoproject.com/. [Accessed: 24-Dec-2021].

[13] Sqlite. [Online]. Available: https://www.sqlite.org/index.html. [Accessed: 24-Dec-2021].

[14] “What is AWS”. [Online]. Available: https://aws.amazon.com/what-is-aws/. [Accessed: 24-Dec-2021].

[15] “BarCodeScanner.” Expo Documentation, https://docs.expo.dev/versions/latest/sdk/bar-code-scanner/. [Accessed: 24-Dec-2021].

[16] “Official Legal Text,” General Data Protection Regulation (GDPR), 2019. [Online]. Available: https://gdpr-info.eu/. [Accessed: 24-Dec-2021].