

EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

Lefke Marketplace

Umutbek Abdimanan uulu

174568

“Lefke Marketplace (Responsive Web application)” is a project that allows to unite in one information and trade space suppliers and consumers of various goods and services and provides the participants of the united trading platform with a number of services that increase the efficiency of their business. For this, I launched a website that allows online sales and purchases of goods / services from business sectors in North Cyprus(Lefke). Nowadays, the business system is highly valued by the relationship between the customer and the seller.

Interactive communication plays a very important role in finding out the buyer's preferences. The most suitable and highly effective option is the Internet. The goal of my project is to convey this system to the target audience as reliably as possible.

Supervisor

Vesile Evrim

Publish Date

17/01/2022

Table Of Contents

Your Name Surname	1
Your Student Number	1
Your Supervisor Name	1
Publish Date.....	2
1.Introduction.....	3
1.1 Goals	3
1.2 Use Case Diagram	4
2. Literature Survey.....	5
3. Background Information	7
3.1 Required & Used software	7
3.1.1 Frontend	7
3.1.2 Backend.....	11
3.2 Other software	16
4. Design Documents	17
4.1 User Flow Diagram.....	17
4.2 Entity Relationship diagram	19
5. Methodology.....	20
6. Conclusion.....	45
6.1 Benefits.....	45
a. Benefits to stores :.....	45
b. Benefits to clients :.....	45
c. Benefits to me :.....	46
6.2 Ethics.....	46
6.3 Future Works	47
7. References	48

1.Introduction

1.1) Goals

The goal of this project is to launch a responsive website that allows online sales and purchases of goods / services from business sectors connected to this system.

Marketplace is a general set of activities related to the market economy, an online database. This is the best assistant or partner project for small medium-sized businesses, large retailers, wholesalers and domestic manufacturers to improve their business and solve the problem of customer policy.

I searched from internet and the global e-commerce turnover reached \$ 29 trillion in 2021. The recognized leaders in this area are the United States, the countries of the European Union and China. There is no digital value for North Cyprus today, I found only one marketplace in North Cyprus, although electronic trading systems are widely used in various spheres of the North Cyprus's economy, affecting the activities of small and medium-sized businesses in the field of retail trade and services, as well as banking. What can I say, today it has become customary for many North Cyprus people including international students to order goods on Amazone, Alibaba, Wildberries.

The main drivers of e-commerce growth are e-marketplaces. They help small and medium-sized businesses to enter the market with minimal investment and skills. And these same sites ensure the security of data and payments, as well as consumer protection. And also my project is guided by the implementation of the following tasks;

- 1) Ensure active business conduct of all sales services.
- 2) Create a modern and reliable environment for entrepreneurs.
- 3) Creation of an expanded online market that includes all business areas.
- 4) An open online marketplace that will be available to lefke people.
- 5) It will give the right to choose and compare prices to lefke people.

"Lefke Marketplace" → STORE → product / service → sale → customers

“Lefke Marketplace” → CLIENT → product / service → purchase → buy / order with delivery

→ Online store → online sale → online purchase

→ Adding/Updating Products / Services

→ Phone calls

→ Basket

→ Filter (specific product)

→ Sort product

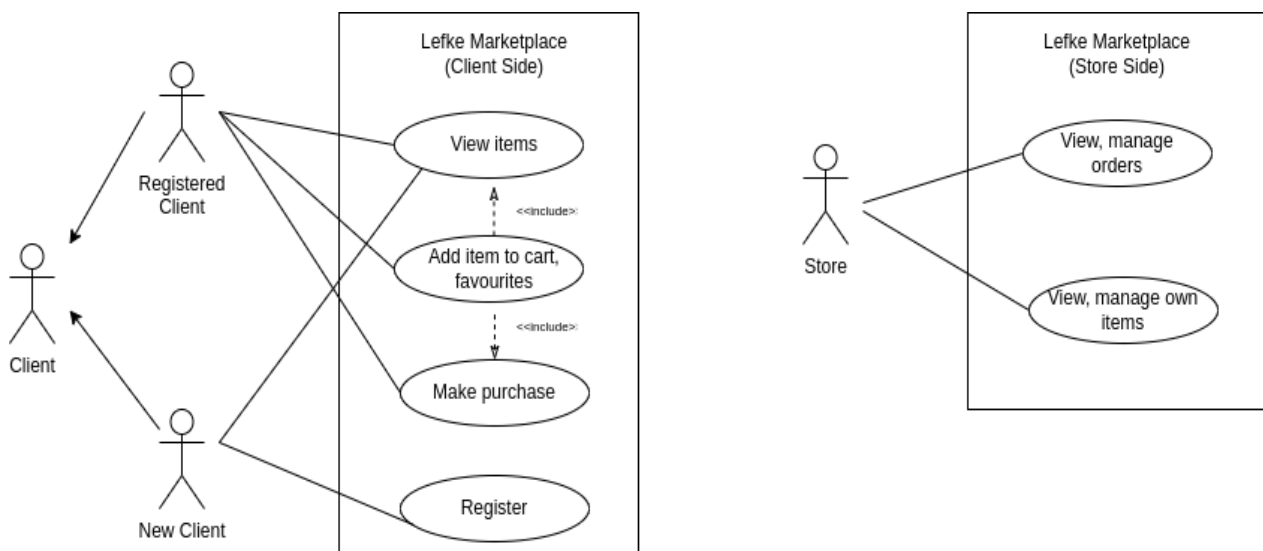
→ Search product

→ Banner advertising

→ Favourite products

→ Order history

1.2) USE CASE DIAGRAM



2. Literature Survey

My project for now only for lefke region, because I did not added geolocation or region category. But in future I am planning do it. It will be the first marketplace in lefke region. In North Cyprus I can found only one marketplace – cypazar.com. Five-six month ago there was more and one of them was inncy.com. But they stopped working.

Cypazar.com online platforms which you can sell and buy goods/services, this websites dedicated only North Cyprus. Unlike my project, there is no store side. From same account user either can buy product and sell product. If you want to sell your chair, you need just take it picture and upload, any user can do it. There is no geolocation or region filter, goods from Karpaz visible to people who lives in Lefke.

In my project clients can't sell their goods, only stores have access to it and they can't register themselves. Only admin register stores to website. There is a huge difference between my project and Cypazar.com

The similarities of our projects is user can filter, sort and search product, select it as a favourite item and order it online.

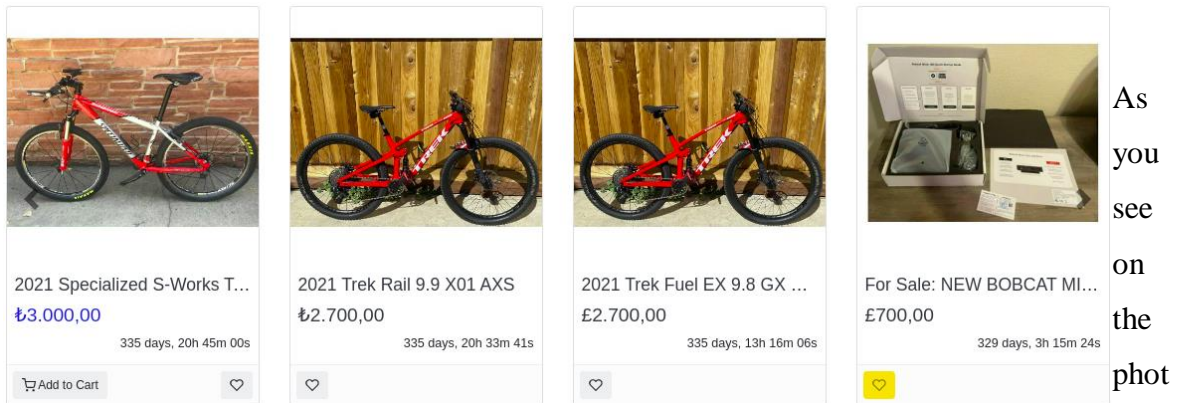
Compare1 : Store page.

As I already said there is no stores in cypazar. Any user can be store, can sell his goods. In my project I have a stores page, where user can see list of stores. Store's own category, goods. User can select favourite or any store which he/she can see all detail about store. Also user has ability to compare products from different stores.

Compare2: Adding item to basket.

In my project when user adds item to basket, for every stores will be created seperate basket. Name of store, items, quantity of item, ability to increase or decrease it, total cost, add more and order. Below, basket for another store.

In cypazar this function is very strange, you can add some goods to basket but for some goods there is no this function.



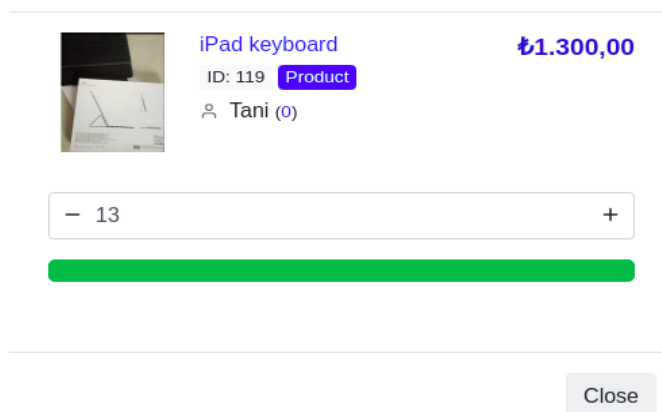
o above, “Add busket “ function has only on first item. And I don’t have any idea about this. Is it a feature or bug...

Compare3: Bid for product

In cypazar you buy products one by one, you can’t select several items from one person or store and buy it all. And they added a function bidding. I really liked this function, because buyer can bargain with seller. In my project there is no this function and I’m not going to add it. Because in my project there is official stores, if people want to bargain they can call to store directly.

All in all, as a user I didn’t like user interface of cypazar, it’s too complicated and too many bugs.

Add to Cart



Unlike my project, here firstly you need to specify count of item, then add to basket. Here I'm trying to add item to basket, I specified count as 13 and there is no any button with text add to cart. Yes, to add it I need to click green button.

3. Background Information

In this project I used React.JS and Next.JS for frontend and Python for Backend.

ReactJS is very popular javascript library that is used for building user interfaces. It is very convenient to use the React library when creating user interface and developing pages.

This is because JSX is used instead of HTML to create the structure of the pages. In JSX, we can simultaneously describe the appearance of the page. And for client part I used Next.Js, it is almost similar to React, because it is React's framework. It is very easy and fast. Next.js is widely used by the biggest and most popular companies all over the world like Netflix, Uber, Starbucks. It is also considered as one of the fastest-growing React frameworks, perfect to work with static static files – which was the hottest topic in the web development world lately

For Backend I used Python's popular framework – DRF(Django Rest Framework)

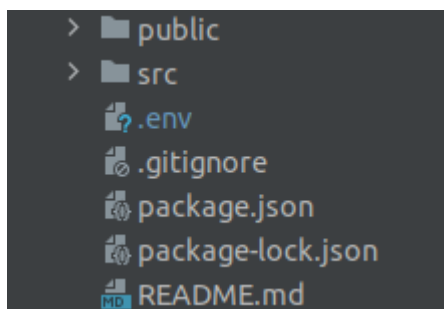
Why I used DRF because it makes serialization so easy. You define your models for your database using Python and it handles all the database migrations and queries. Just a few lines of code using Django Rest Framework, and you can serialize you database models to REST-ful formats.

3.1 Required & Used software

3.1.1 FRONTEND:

Firstly let's talk about Structure of Frontend

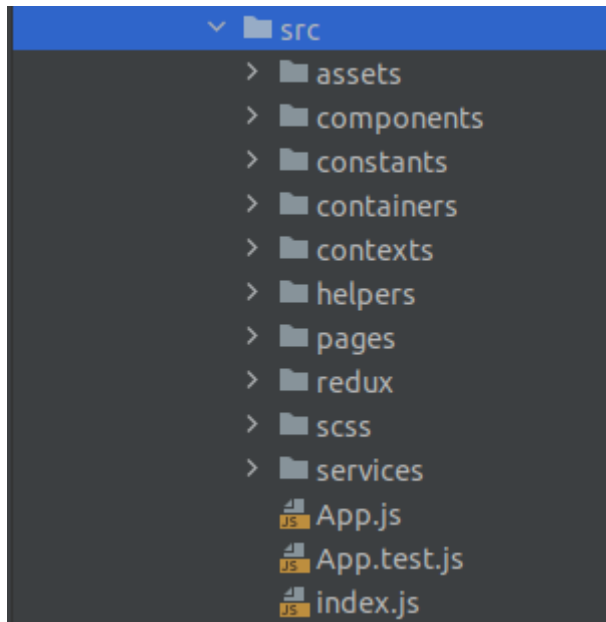
When creating a web application, it is recommended to logically separate the codes into directories.



public - the index.html file and general information about the project.

package.json file contains information about the external libraries included in the project.

src - writes executable code and saves media files (shown below)



assets - media files are stored.

components - records components (folders and keys) that have been used more than once.

constants – constant variables like in status(New – 1, Packing – 2, On_way - 3)

containers - write logical codes (filter, search).

helpers - for helper functions. (used for date-time)

pages - contains pages. Uses components and containers inside.

redux - Writes the logic for saving, modifying, disabling and accessing a web application.

scss - contains scss style files

services - the logic and functions of creating requests to the server.

- For example: Update product


```

updateProfile = async (formData) => {

  const options = {

    method: 'PATCH',

    headers: {

      'Authorization': `Token ${getToken()}`,

    },

    body: formData

  }

  return _doRequestAndParse(`${_baseApi}/item/{item_id}`, options)

}

_doRequestAndParse = async (url, options = { method: 'GET' }) => {

  try {

    const response = await fetch(url, options)

    const data = await response.json()

    if (!response.ok){

      return { success: false, data }

    }

    return { success: true, data }

  } catch (e) {

    return { success: false, data: { detail: e.response?.data ||
      e.message.toString() } }

  }

}

```

index.js file describes the integration of the `responsex` and `il8next` libraries and the `react` library.

Used software and tools

3.1.1.1 - React.JS and Next.JS frameworks

I already said about this popular frameworks above. React – developed and supported by Facebook and a team of private programmers. Next.JS – owned by company Vercel, which also maintains and leads its open-source development. They both used for creating user interfaces and developing pages.

3.1.1.2 - NPM package manager

NPM is a package manager for the JavaScript language. This will help you download the required packages easily and conveniently. NPM also creates its own package.json file. This file contains the names and versions of the required packages. With this file, other users can install all packages using the same npm install command.

3.1.1.3 -Bootstrap

Bootstrap is a CSS framework developed by Facebook for all devices. It includes ready-made components such as forms, buttons, and navigation. This structure simplifies the process of developing HTML pages and eliminates the need to write any CSS or JavaScript code. It is by far the most popular CSS framework.

3.1.1.4 -Redux framework

Redux is a status management library for JavaScript applications. It helps to write stable applications that run in different environments (client, server) and are easy to test. Redux is ideal for medium to large applications. It should only be used when it is not possible to manage a standard situation in React or another library with a manager.

3.1.1.5 - React-router-dom framework

React Router is the React standard routing library. This will sync the web app interface with the URL in the browser. React Router allows you to intelligently control the flow of information in your web application. In other words, the URL can be manipulated from a non-server application.

3.1.1.6 - SASS

SASS (Syntactically Awesome Style Sheets) is a scripting language that extends the capabilities of CSS. The page's CSS styles are written in SASS syntax and compiled to CSS. The main goal of SASS is to make the coding process simpler, more efficient, and easier to understand. SASS advantages:

- Using Variables
- Import
- worn out rules
- mixins (sometimes called syntactic sugars)

3.1.1.7 – Webstorm

WebStorm takes full advantage of the modern JavaScript ecosystem. Uses smart code completion, input error checking, fast code navigation, and refactoring for JavaScript, TypeScript, style languages, and popular frameworks.

- Correct mistakes
- Integration with version control systems

3.1.2 Backend

Some information and structure about backend

The most important thing for Backend is the creation of the endpoint. The endpoint has three parts:

URL are the way that kind should evolve when it comes to demand.

Views - get data from the request and respond to the customer.

Serializers - data validation and transformation.

```

class LoginAPI(APIView):
    """Create a new auth token for user"""
    serializer_class = serializers.LoginSerializer

    def post(self, request):
        serializer = serializers.LoginSerializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data['user']
        info = models.User.objects.filter(login=user)
        userdata = serializers.UserSerializer(info, many=True)
        token, created = Token.objects.get_or_create(user=user)
        return Response({"token": token.key, 'data': userdata.data}, status=200)

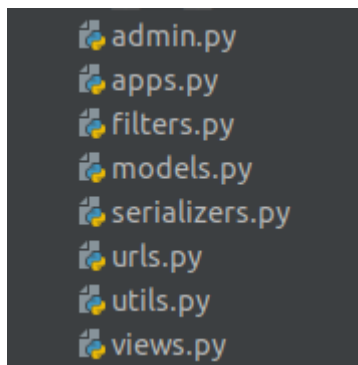
```

Image(3.1.2.0). The class that designed the endpoint for login

As shown in Image 3.1.2.0, this class only works with the POST method, and the data is retrieved using the LoginSerializer.

Backend structure

Looking at the Backend structure by code, looking through the directory and file, it becomes clear.



admin - This file writes classes and functions for making changes to the Django admin panel.

filters – filter logic of project

models - the classes used to write the database structure are written to this file.

serializers - This file is used to write the serializer used to validate and input data on demand

urls - from a back-end perspective, to identify each endpoint and which class or method develops it - they are all written in a single file.

utils - additional functions are written to this file, in addition to the endpoint. It can be anyone, send a letter, send a request to the verification system.

views - The classes that run and respond to each endpoint are written to this file.

3.1.2.1 -PIP

Pip - package manager for the Python language. This will help you download the required packages easily and conveniently. Downloaded packages are loaded into the current Python interpreter. Every Python programmer can contribute and download various packages to the pip system, and this system has become the standard Python system.

3.1.2.2 - Django Rest Framework

DRF (Django Rest Framework) - written in Python. It downloads packages using pip. It is used to create REST API and is multifunctional, most importantly:

- **Serialization** - casting data to the desired type (json, multipart / form-data, etc.).

Json is often used

- **Routing** is the method that responds to a request to the backend.

- **Authorization** - authorization there are different methods for connecting different fronts and backend (TokenAuthentication, SessionAuthentication, JWTAuthentication). This method is the most convenient from a security point of view, since I am used TokenAuthentication technology.

- **Documentation** - Clear documentation is written for each endpoint which method is HTTP and which data is sent or received. To open the documentation, you need to go to the link **main_url/swagger/**

Marketplace API ^{v1}

[Base URL: `lefkemarketbackend.herokuapp.com/api`]
<http://lefkemarketbackend.herokuapp.com/swagger/?format=openapi>

Marketplace API

[Terms of service](#)

[Contact the developer](#)

[BSD License](#)

Schemes

HTTP

Django

Umutbek

Django Logout

Authorize

Filter by tag

item

GET /item/cart/ item_cart_list

POST /item/cart/ item_cart_create

GET /item/cart/{id}/ item_cart_read

PUT /item/cart/{id}/ item_cart_update

PATCH /item/cart/{id}/ item_cart_partial_update

DELETE /item/cart/{id}/ item_cart_delete

Documentation screenshot

- POSTGRESQL (DATABASE)

PostgreSQL is one of the most popular relational database management systems on the market. It's also free and open source. PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set and innovative solutions.

I used postgresQL because using it with Django offers many benefits.

For example:

- Django has `django.contrib.postgres` to make database operations on PostgreSQL.
- PostgreSQL has the richest set of features that are supported by Django(Full-text search, Database migration operations)

- PYCHARM (editor)

PyCharm is one of the widely used Python IDE which was created by Jet Brains. It is one of the best IDE for Python. I am using professional PYCHARM edition. It gives me access to write code not only in pure python but also in React and Next.JS. And can deploy and debug code running on Docker container.

-Heroku Server

I am saving my project into Heroku server. It is a containerized cloud platform as a service (PaaS). Developers use Heroku to deploy, manage and scale modern applications. This platform is elegant, flexible, and easy to use. It's a great place to practice popular architectural patterns. When you run into issues, heroku robust documentation, support, and community resources are available to help you understand the root cause and troubleshoot quickly.

There is a free services to experiment and learning, for my project I am using free services

- Firebase

I already said about heroku server which I deployed my project, it is free and not great server for this project. Because in getting order part it must be real time update(asynchronous), heroku can't handle it. To solve it I used Firestore database which is asynchronous(Why I used it? I explained it in detail below). Firebase Realtime Database is a cloud hosted database. The data is stored in JSON format and synchronized in real time with every connected client. I built it cross-platform apps using JavaScript SDK, and list of my orders page is automatically updated with the latest data.

- Docker

I runned my backend part of project in Docker. Docker is a container management system. It allows you to package a website with all its environment and dependencies into a container, which can be easily and easily managed in the future: transferred to another server, scaled, updated. Because of I'm deployed project to Heroku, I didn't use a docker in deployment. Because in heroku server deploying with gunicorn easy and very fast

3.2 Other software

- Figma:

I used FIGMA for designing my project user interface and logo. It is a web-based graphics editing and user interface design app. You can use it to do all kinds of graphic design work from wireframing websites, designing mobile app interfaces, prototyping designs, crafting social media posts, and everything in between.

- Bitbucket

I am storing my source code into bitbucket. It is a cloud-based service that helps developers store and manage their code, as well as track and control the changes to their code. BitBucket provides a cloud-based Git repository hosting service. Its interface is user-friendly enough so even novice coders can take advantage of Git. We generally require a bit more technical knowledge and use of the command line to use Git alone.

- Postman

In this project I worked on backend and frontend parts myself and I used postman for API testing. Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.

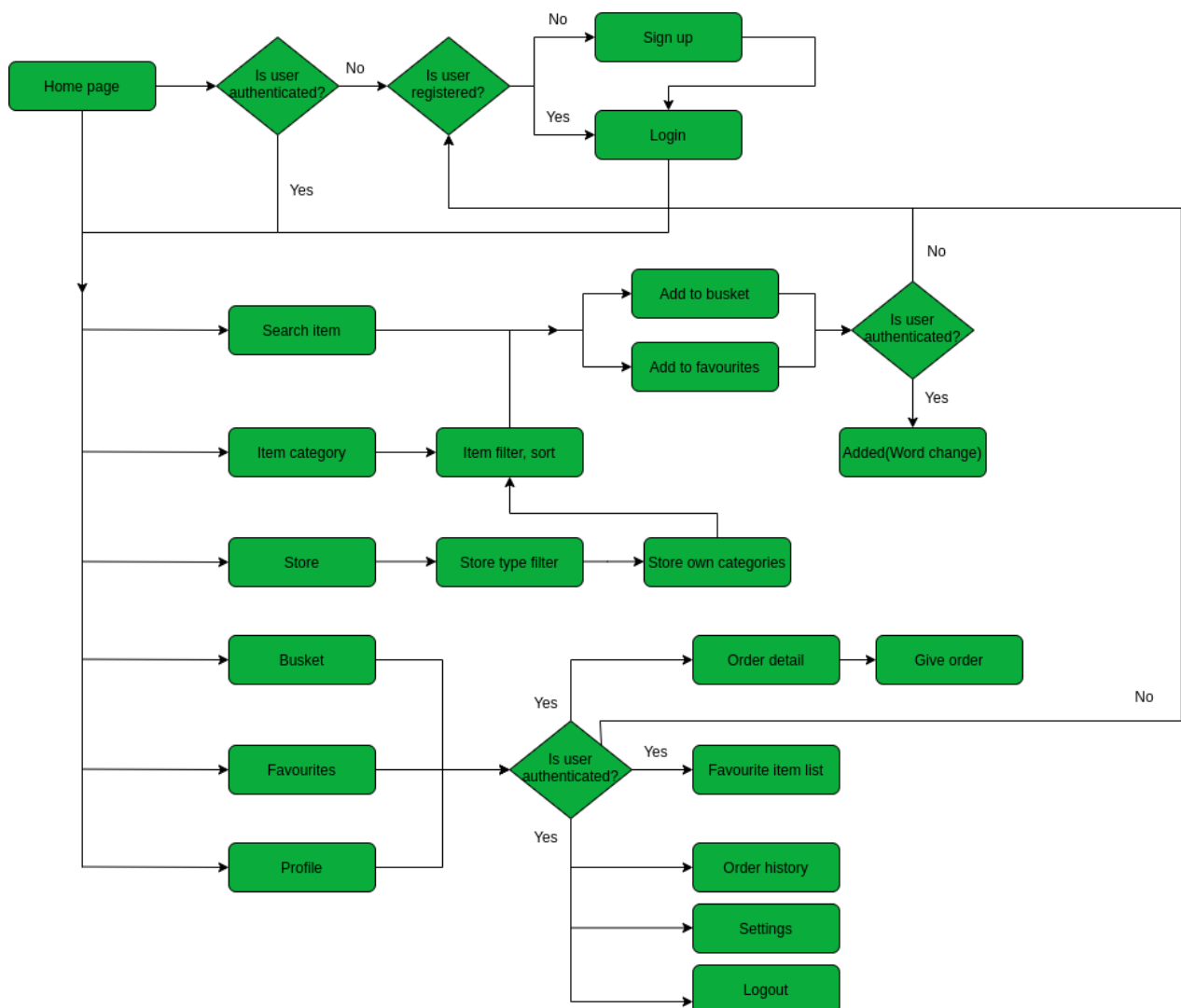
Postman offers many endpoint interaction methods. The following are some of the most used, including their functions:

- GET: Obtain information
- POST: Add information
- PUT: Replace information
- PATCH: Update certain information
- DELETE: Delete information

4. Design Documents

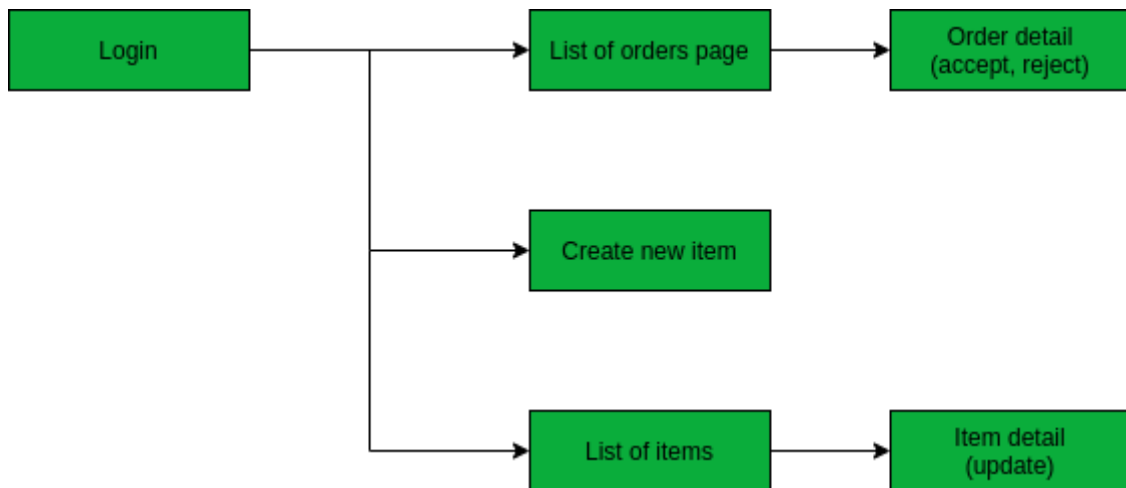
4.1 User flow diagram

4.1.1 Client



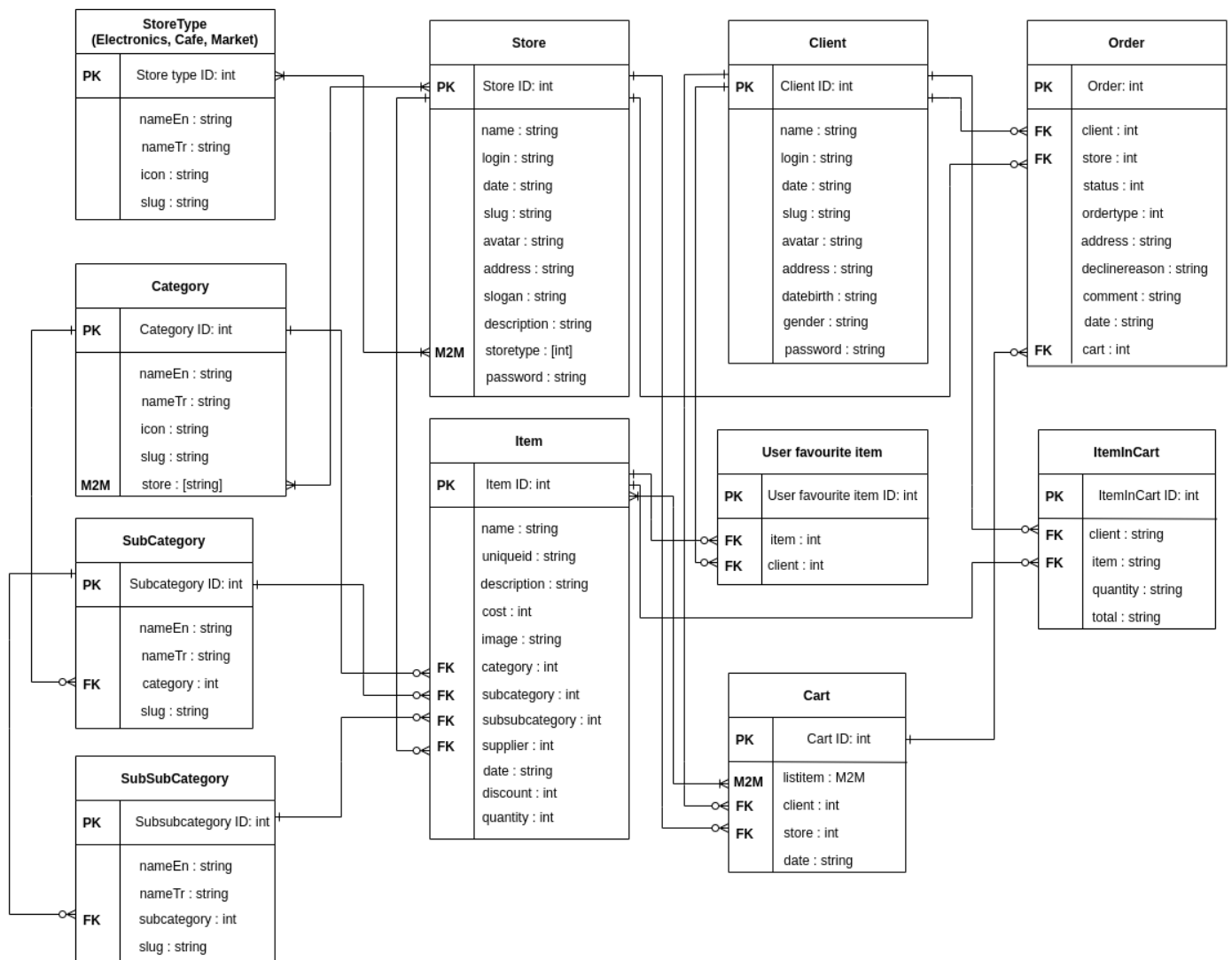
In the diagram above, you can see a user flow. Shortly, in the project we have authenticated and non-authenticated users. Non authenticated users can view items, filter, sort and search it, but they can not do main functions like: Adding to basket and order, select it as a favourite item. To do it, user needs to authenticate.

4.1.2 Store side



This one is store's user flow diagram. Store don't have ability to register themselves. Only admin can register the store. They login to system using given username and password. They can see list of orders and manage it (accept, reject). And add new item, see other items and update it.

4.2 Entity Relationship Diagram



Architecture of project. It consists 11 tables

1) StoreType – The store type table in the system (Electronics, Cafe, Supermarket)

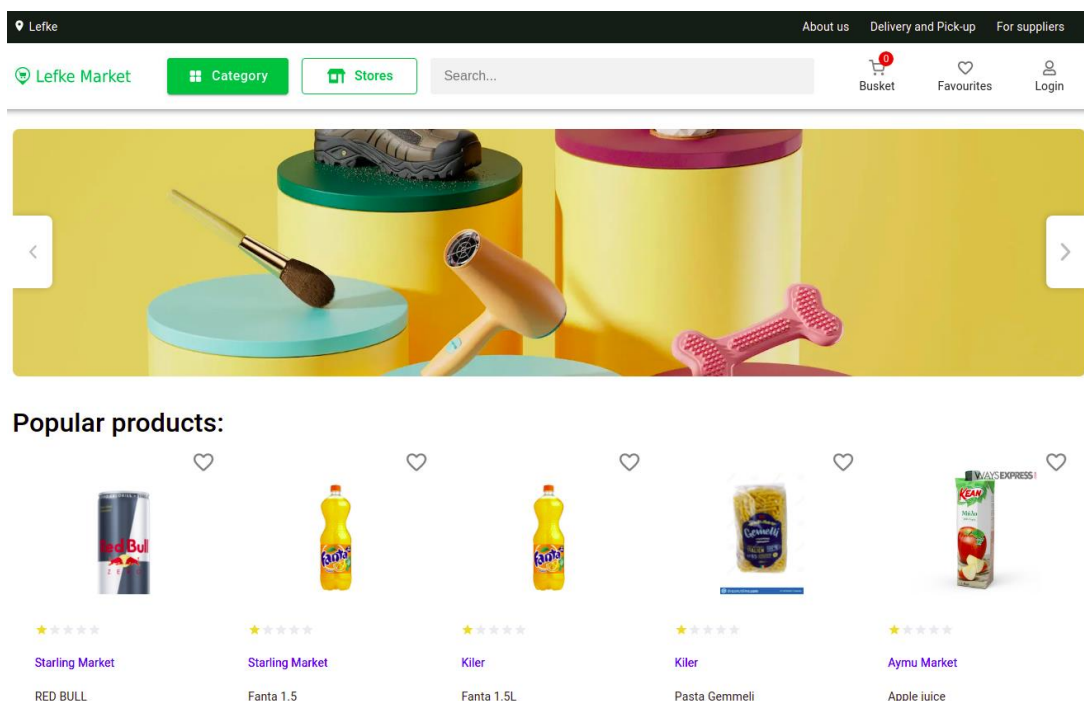
- 2) **Store** - The store table in the system
- 3) **Client** – The client table in the system
- 4) **Category** – The Category table in the system
- 5) **Subcategory** – The Subcategory table in the system
- 6) **SubSubCategory** – The SubSubCategory table in the system
- 7) **Item** – The Item table in the system
- 8) **UserFavouriteItem** – The user favourite items table in the system
- 9) **Cart (Basket)** – The Basket table in the system
- 10) **ItemInCart** – The items in the cart with quantity table in the system
- 11) **Order** – The client order table in the system

5. Methodology

In this project, I worked on two different websites, responsive website for clients and website for stores.

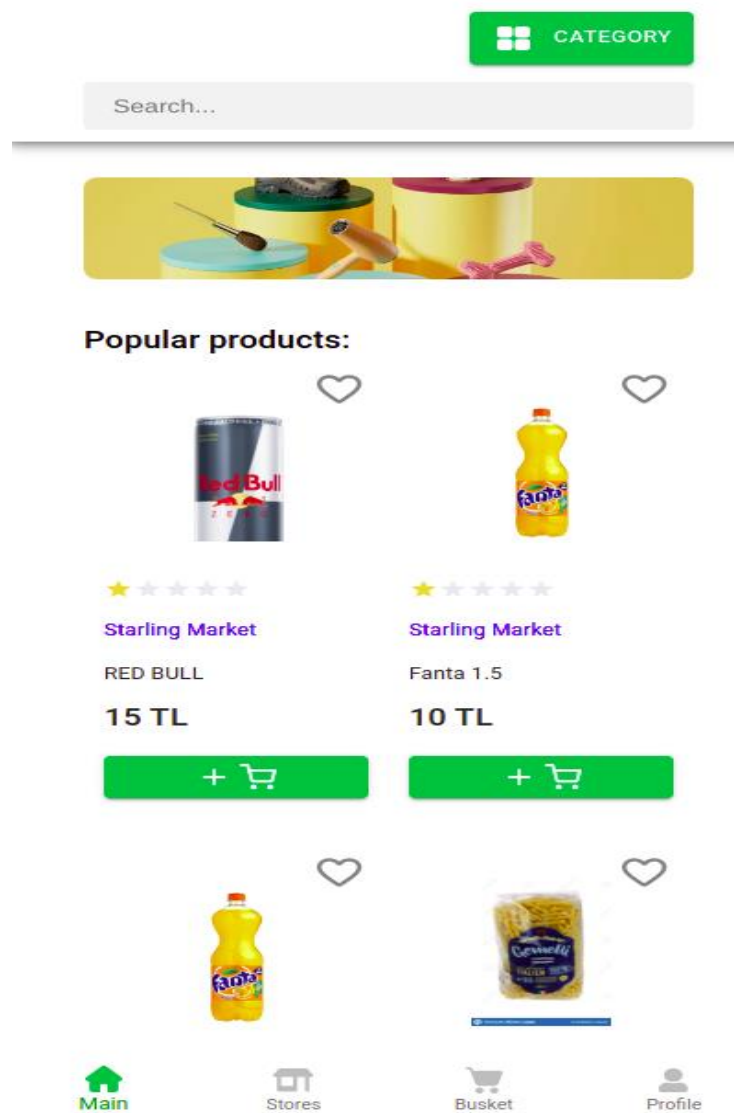
Responsive website for clients:

Client typing the domain of the website in the browser and will be redirected to the main page of the site. It does not matter, with laptop, tablet or phone it is a user friendly website. Even if user not registered, he will be redirected to the main page and can see list of products, stores, can filter, sort and search products. But they can't do functions like adding item to basket and order it, select item as a favourite.



10 TL	15 TL	6 TL	4 TL	15 TL
ADD TO BASKET	ADD TO BASKET	ADDED	ADDED	ADD TO BASKET

Main page. (Image 5.1.1)



Main page mobile version

(Image 5.1.2)

Popular products

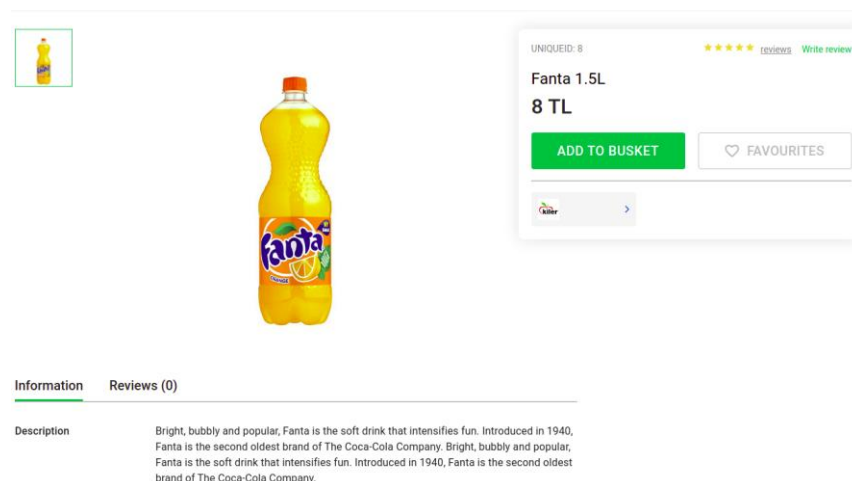
As you can see from the image (5.1.1 and 5.1.2) the user will be redirected to page. And can able to see most popular products in the marketplace. So how this popular products is working:

I have a table IPAddresses and I'm connecting it with table Item as a many to many field and called it views. So when a user sees an item in detail, using `request.META.get('HTTP_X_FORWARDED_FOR')` I'm catching user IP ADDRESS and adding it to field views, if it does not exist in the field yet. And in the final, I am calculating number of ip addresses inside views field and by this way we can calculate number of views in item.

When I'm showing list of popular products, I'm just ordering number of views in descending order.

Item in detail

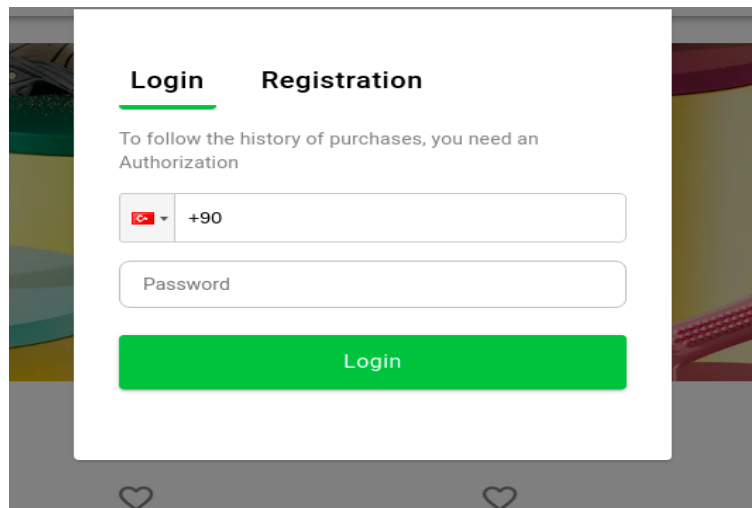
If a user clicks to the item, user will be redirected to item in detail page. In this page (Image 5.1.3) user can see Item in detail. Description of item, all images and write review for an item(for now review is not reliaed)



Item in detail (Image 5.1.3)

Non authenticated user

As you can see images above, user can select item as a favourite and add it to basket. But what if user not authenticated? So in this situation, user will be redirected to login page.

The image shows a mobile application interface for a login page. At the top, there are two tabs: 'Login' and 'Registration'. The 'Login' tab is selected, indicated by a green underline. Below the tabs, a message states: 'To follow the history of purchases, you need an Authorization'. The form includes a phone number input field with a dropdown menu showing a Turkish flag and '+90'. Below this is a 'Password' input field. At the bottom of the form is a large green button labeled 'Login'. The background of the app shows a blurred image of a grocery store aisle with various products.

Login **Registration**

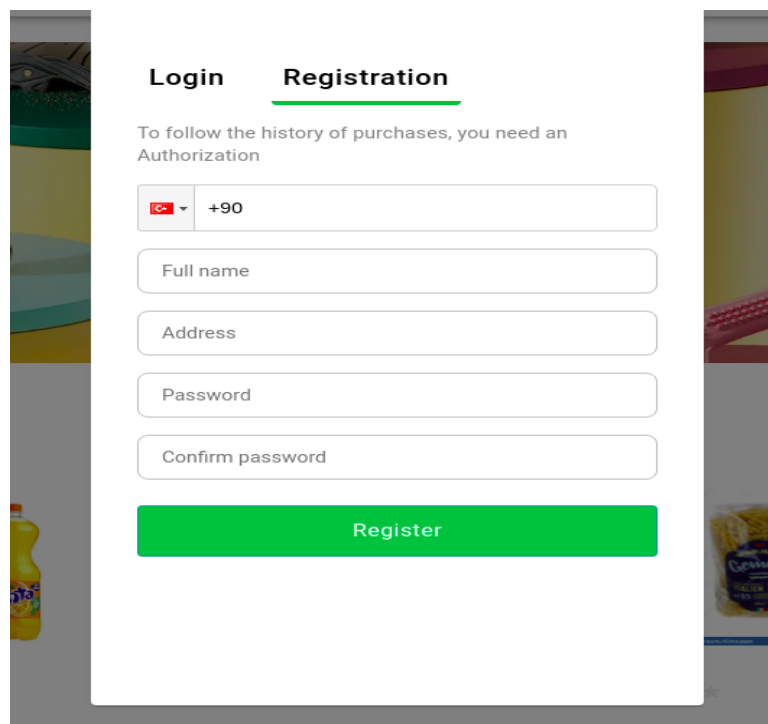
To follow the history of purchases, you need an Authorization

+90

Password

Login

Login page (Image 5.1.4)

The image shows a mobile application interface for a registration page. At the top, there are two tabs: 'Login' and 'Registration'. The 'Registration' tab is selected, indicated by a green underline. Below the tabs, a message states: 'To follow the history of purchases, you need an Authorization'. The form includes a phone number input field with a dropdown menu showing a Turkish flag and '+90'. Below this are input fields for 'Full name', 'Address', 'Password', and 'Confirm password'. At the bottom of the form is a large green button labeled 'Register'. The background of the app shows a blurred image of a grocery store aisle with various products.

Login **Registration**

To follow the history of purchases, you need an Authorization

+90

Full name

Address

Password

Confirm password

Register

Registration page (Image 5.1.5)

If user is not registered yet, he can go to register page(Image 5.1.4). After registration will be redirected to login page(Image 5.1.4) automatically.

So how it is working :

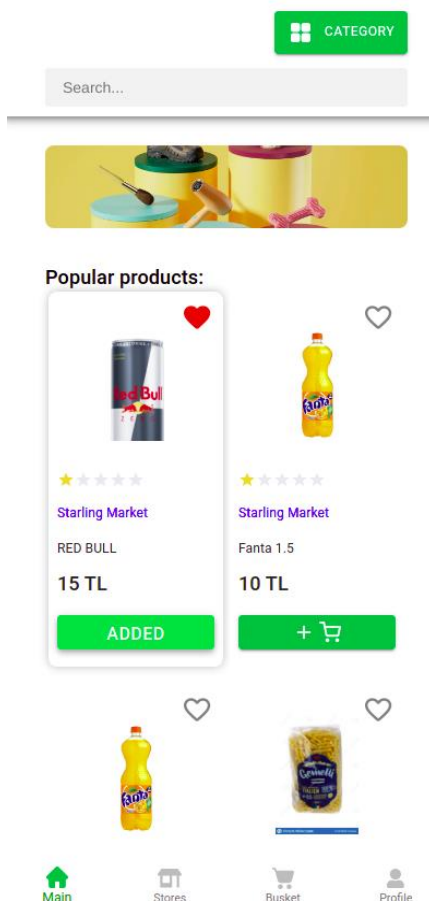
When user trying to register, he need to write only turkish number as a username. It needs to contain 12 characters and starts with +90. Otherwise, it gives error. New password and confirmation should match otherwise it gives error too (Passwords do not match!!!)

During login, if user credentials are true and exists in the DB user will be authenticated and redirected to main page. Otherwise, gives error (Incorrect Username or Password || User does not exist)

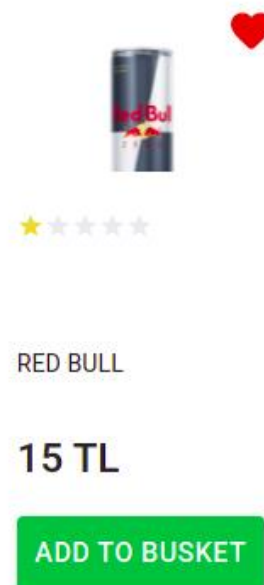
In django there are different methods for authorization such as TokenAuthentication, SessionAuthentication, JWTAuthentication. I used TokenAuthentication technology, this type of authentication is suitable for client- server applications, where the token is safely stored. You should never expose your token, as it would be (sort of) equivalent of a handing out client username and password.

Authenticated user and favourite items

After successful authentication, user will be redirected to main page. Now user has access to all functions in website. As you can see from Image 5.1.6, user selected RED BULL as a favourite item. And a heart's color became a red. So now from profile user can find My favourites, if you click it you can see Image 5.1.7



Favourites



Favourite items page

(Image 5.1.7)

Item selected as favourite

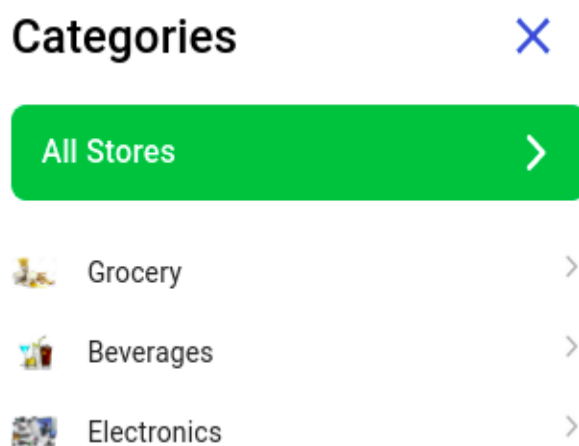
(Image 5.1.6)

Logic of this function: I have a table – UserFavouriteItems. In this table have 2 columns (1.user – FK of Client table and 2.item – FK of Item table). So when user clicks to heart(select it as a favourite), post request will work, where column item – selected item, user – Token of authenticated user.

In get request, I am filtering selecting item by TOKEN. So authenticated user can get only their selected items

Category

User can filter items by their category clicking category button (Image 5.1.1 – 5.1.2) and you will see Image(5.1.8). If user clicks to category he can see subcategory of this category and by this way user can filter items by category, subcategory and subsubcategory.



(Image 5.1.8)

How this filter is working? I added three fields to my ITEM table: category, subcategory and subsubcategory and connected it with tables category, subcategory and subsubcategory as FK field. So now I can assign item to it's category and using

django-filter library I can filter item by it's category, subcategory and subsubcategory. It's url looks like:

mainURL/item/? category_id={id}&subcategory_id={subcategory_id}

It gives list of filtered items in JSON format

Filter and sort items

After filtering item by it's category you will be redirected to this page(Image 5.1.9). Here you can sort items by it's popularity(most viewed items), cost and rating(not realised yet) in ascending and descending orders. Django_filters library includes OrderingFilter class that supports simple query parameter controlled ordering of results. I used this class to order items. It's url looks like:

All / Beverages

Beverages

Categories

Carbonated drinks

Natural Juices


Energetic drinks

Filter

Cost

From ₺ To ₺

Sort by: Popularity Cost Rating




★ ★ ★ ★ ★

Aymu Market

RED BULL

18 TL

ADD TO BUSKET




★ ★ ★ ★ ★

Aymu Market

Apple juice

6 TL

ADD TO BUSKET




★ ★ ★ ★ ★

Kiler

Apple juice 1L

5 TL

ADD TO BUSKET




★ ★ ★ ★ ★

Starling Market

Fanta 1.5

10 TL

ADD TO BUSKET



★ ★ ★ ★ ★

Kiler

Fanta 1.5L

8 TL

ADD TO BUSKET

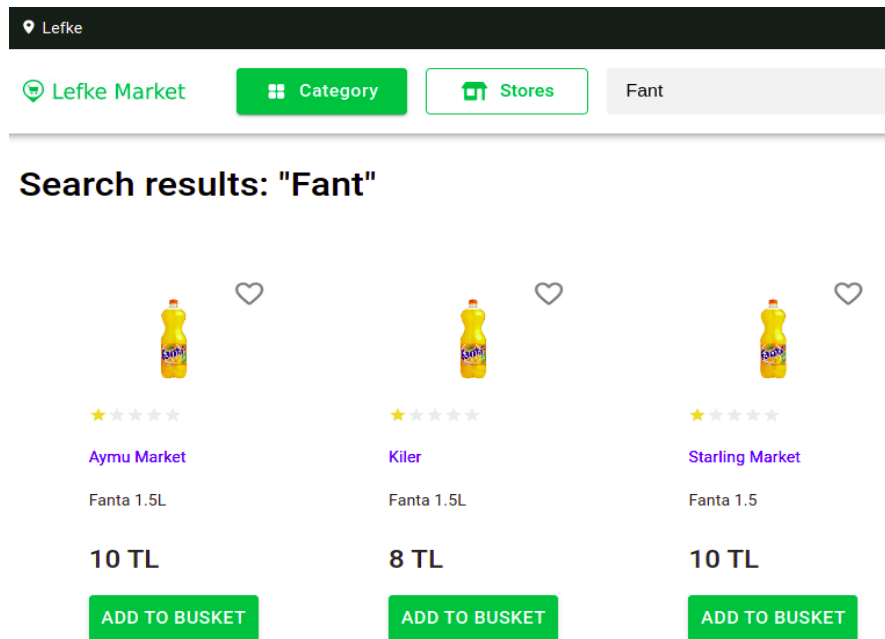
main URL/
item/?
orderi
ng=vi
ews(a
scend
ing
order)
-
views
(desc

(Image 5.1.9)

Search item

User also can search an item (Image 5.2.0).

26



(Image 5.2.0)

As you see from the picture (Image 5.2.0) all searched items from different stores will be listed. Here user can compare item cost from different stores. For this function I used django-filter library's SearchFilter class that supports simple search by one query parameter. It searches item by it's name and description.

When user gets list of products, what if number of products more than 1000000 and how it works? How much time it takes to load large number of items,

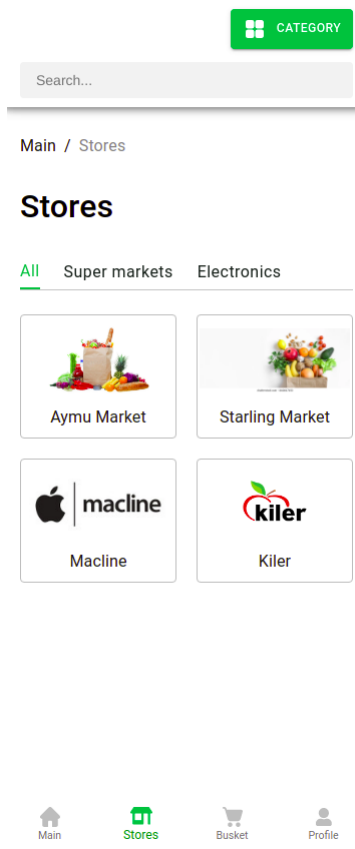
In this situation I used Pagination. This allows you to modify how large result sets are split into individual pages of data. I used DEFAULT_PAGINATION_CLASS and PAGE_SIZE setting keys.

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 20,
```

So now page_size is 20, it means when user gets large number of items it splits into individual pages which is 20. When you come to end of first 20 items, other 20 will be loaded and until end it works in this way

Stores page

When user clicks to store button in image(5.1.1-5.1.2) he will be redirected to this page(Image 5.2.1 – Image 5.2.2)

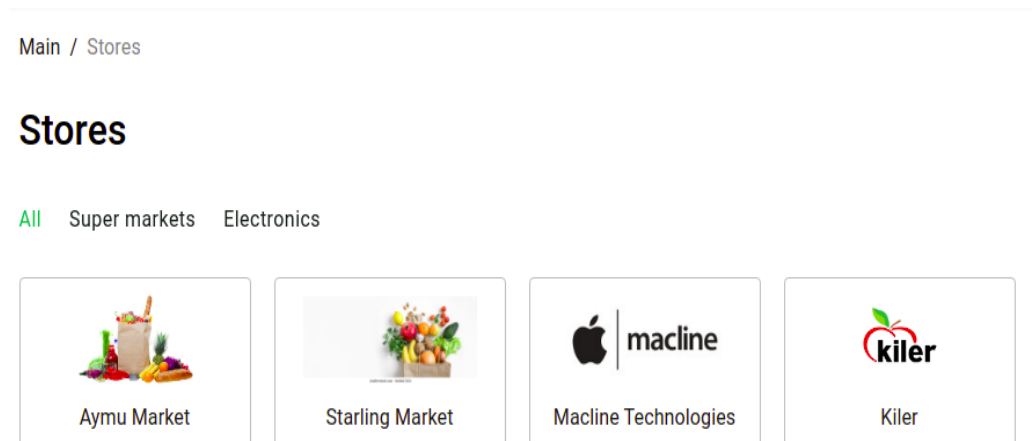


(Image 5.2.2)

mobile version

In this page you can see the list of stores and their types. You can filter it by their types. For example, If there are 50 stores with different types and you want only electronic stores, you need to just select types(categories) above the store

(Image 5.2.3)



(Image 5.2.1)

Main / Stores

Stores

All Super markets Electronics



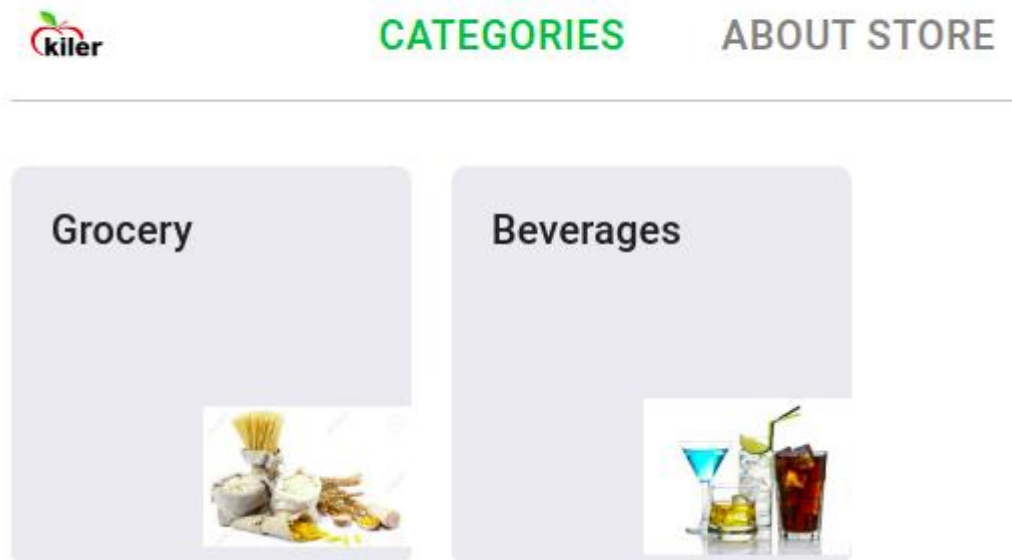
(Image 5.2.3)

Store own categories

If a user clicks to favourite or any store, he will be redirected to this page

(Image 5.2.4)

Main / Stores /



(Image 5.2.4)

As you see inside the store you can see store own categories. In our list categories we have electronics, but we can't see it inside kiler market. Because kiler do not sell electronics. So how it is released:

I have a category table and I added stores fields to this table. It is connected to Store table as a Many To Many Field.

So now if I create a category beverages, to store field I can add list of stores who sells beverages.

Beverages

NameEn:	<input type="text" value="Beverages"/>
NameTr:	<input type="text" value="Beverages"/>
Icon:	<input type="text" value="https://online.gipermarket.kg/upload/resize_"/>
Slug:	<input type="text" value="beverages"/>
Supplier:	<input type="text" value="kiler@gmail.com"/>

In main page I'm getting list of categories by this url:

- mainURL/item/category - it returns all categories

To get store categories I am filtering it by store:

- mainURL/item/category/?supplier={store_id} - it returns exact store categories

About store:In the right side of the image (5.2.4) you can see About store. If you click it, you will be redirected to this page(Image 5.2.5)

Main / Stores /



CATEGORIES

ABOUT STORE

+905488533853



LAU YOLU

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqu Lorem ipsum dolor sit amet,. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqu Lorem ipsum dolor sit amet,.Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqu Lorem ipsum dolor sit amet,.

(Image 5.2.5)

Here user can see all information about Store. In future I am planning to add a map, user will be able to see the exact location from map.

Cart

To buy a product user firstly need to add item to busket. We have different stores and cart for every stores will be created seperately as shown(Image 5.2.6)

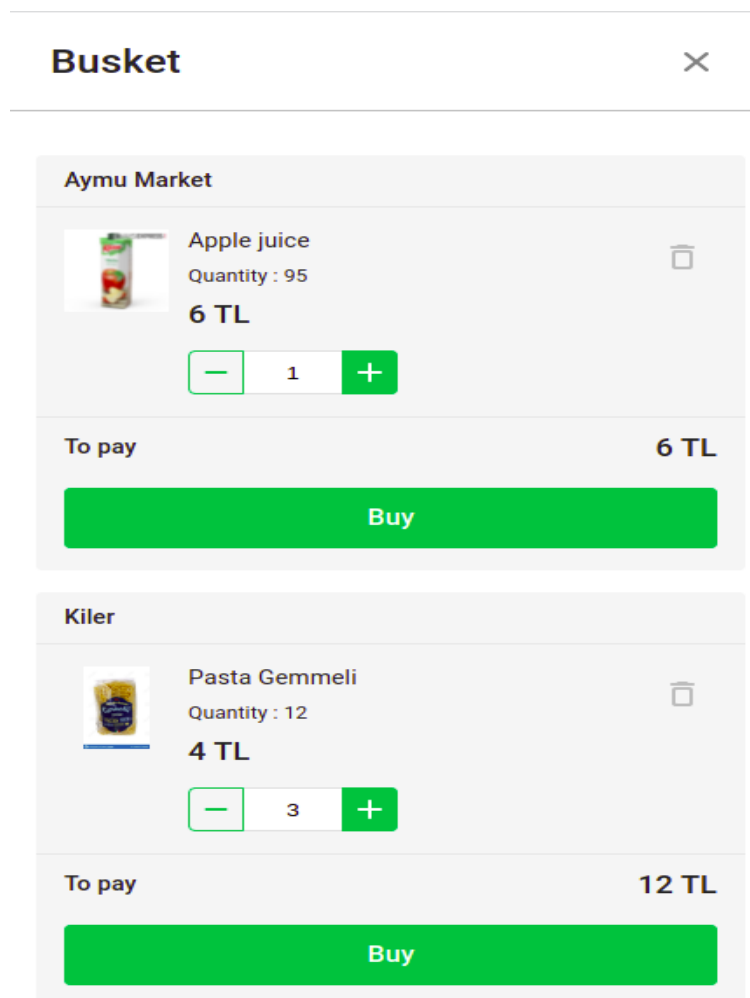


Image 5.2.6

As you see image 5.2.6, for two stores created separate baskets. You can increase and decrease quantity of item in the basket. If quantity is equal to 0 item will be deleted automatically. In increasing when the quantity of item is exceed the actual quantity in the store it gives error messages('Quantity' must be smaller or equal to actual quantity in the store)

If user wants to remove item from the basket, there is a delete button in right side.

In the button of each basket you can see a total cost of all items, every time when user increase or decrease or add an item it changes.

Logic of basket.

When user adds item to basket, he selects item of exact store and press button add to basket. At this time post request will be sent for my 3 fields: Item-selected item, store-selected item's store and client-who selected that item. Using django signals two tables in database will be created automatically.

1) ItemCart(Basket) table

- selected item – integer (FK of item)
- quantity – float (initially will be 1)
- total – float (total price of one item)

When user increase the quantity of item in the basket post request will work and quantity of item in ItemCart table will increase. If user adds new item to basket, signal checks item in db, if it does not exists new item with quantity will be created in ItemCart table.

2) Cart table

- itemcart - [] (Many to many field with ItemCart table)
- store – int (FK of Store – selected item's store)
- client – int (FK of Client – who selected that item)
- totalprice

Selected items with quantity will be added to itemcart table. Before adding I'm checking either store and client. I have a store in my Cart table, using it I'm separating baskets, for each store own basket. And client field to get authenticated user's cart. Totalprice – all selected items total prices.

If in basket has 2 items.

1) Item 1 → quantity 3 → cost 12TL - total field in ItemCart table will be 36 TL

2) Item 2 → quantity 1 → cost 20 TL – total field in ItemCart table will be 20 TL

Here total price field in Cart table will be 56TL


```
[
  {
    "id": 15,
    "listitem": [
      {
        "id": 18,
        "item": {
          "id": 15,
          "name": "RED BULL",
          "uniqueid": "12349",
          "description": "",
          "cost": 15.0,
          "slug": "red-bull-2nw0",
          "issale": false,
          "discount": null,
          "quantity": 15,
          "publishDate": "2021-12-11T21:23:28.059128Z",
          "image": "https://images.ctfassets.net/lcr8qbvxj",
          "category": 7,
          "subcategory": 9,
          "subsubcategory": null,
          "supplier": 15,
          "item_views": [
            1,
            6,
            7
          ]
        },
        "total": 15,
        "quantity": 1
      }
    ],
    "clientid": {
      "id": 9,
      "username": "Umutbek",
      "login": "+995488533853",
      "phone": "+995488533853",
      "email": "karimovumutbek@gmail.com",
      "avatar": "",
      "date": "2021-12-05T22:19:40.929185Z",
      "slug": "",
      "address": "LAU YOLU",
      "type": 1,
      "datebirth": "1999-08-14",

```

-->

```

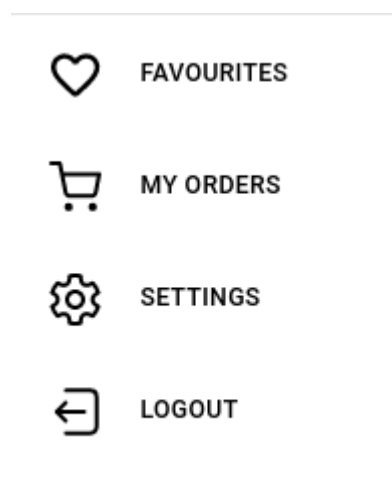
    "gender": 1,
    "vcode": null
  },
  "storeid": {
    "id": 15,
    "username": "Starling Market",
    "login": "starling@gmail.com",
    "phone": "+905488533859",
    "email": "aymu@gmail.com",
    "slug": "starling",
    "avatar": "https://image.shutterstock.com/ima",
    "type": 2,
    "address": "LAU YOLU",
    "slogan": "Lorem ipsum dolor sit amet, consec",
    "description": "Lorem ipsum dolor sit amet, c",
    "storecategory": [
      2
    ]
  },
  "check": 24,
  "total_price": 15

```

The Cart will be given in JSON format like picture above

Profile

When user clicks to own profile will be redirected to this page (Image 5.2.7)



Image(5.2.7)






Favourite – List of user favourites items

My order – User orders

Settings – There will be language changes, terms of use and

Logout - Logout from website

My orders

Not confirmed	Not confirmed	Delivered	Declined	On way
				
Starling Market	Aymu Market	Kiler	Starling Market	Kiler
Fanta 1.5	Apple juice	Pasta Gemmelli	Fanta 1.5	Fanta 1.5
10 TL	6 TL	4 TL	10 TL	10 TL
Ordered : 2022-01-16T11:03:57.755768Z	Ordered : 2022-01-15T01:34:54.547352Z	Ordered : 2022-01-15T01:33:44.916541Z	Ordered : 2022-01-15T01:29:15.792952Z	Ordered : 2021-12-29T13:08:50.678780Z

Here client can see own orders with statuses and date

Client Order

When user clicks to BUY button in Image(5.2.6), he will be redirected to order page(Image(5.2.8))

Give order

Phone number

 +90 (548) 853-3853

Delivery type

☒ Delivery

☐ Pick up

Tufekchi E blok, 2 daire

Total cost

15 TL

Confirm order

Image(5.2.8)

Here client needs to write phone number, which courier or market workers can call. Can select order type(delivery or pick up). Market administrators can see it, if order type is pick up they will make your order ready and clients go to market, pay and can take it. If type is delivery, market courier deliver your order to address which you wrote under the delivery type. For now there is no online payment, Client gives money to courier

Responsive website for Stores

In this project stores can't register themselves and sell their products. To be partner they firstly need to make contract with administrator of this website. After contract, admin will register the store in DjangoAdmin panel and gives credentials to store:

Select Store to change ADD STORE +

Action: 0 of 4 selected

<input type="checkbox"/>	STORE
<input type="checkbox"/>	aymu@gmail.com
<input type="checkbox"/>	starling@gmail.com
<input type="checkbox"/>	macline@gmail.com
<input type="checkbox"/>	kiler@gmail.com

4 Stores

So what is the Django Admin Panel:

Django provides a default admin interface which can be used to perform create, read, update and delete operations on the model directly. It reads set of data that explain and gives information about data from the model, to provide an instant interface where the user can adjust contents of the application . This is an in-built module and design to execute admin related work to the user. There will be adminuser who can work with all models. I have a model store(table store) and admin can create, read, update and delete it.

Using given credentials by us, store can login to website:

The screenshot shows the login interface for Lefke Market. On the left, there is a 'Login' section with the text 'Sign In to your account'. Below this are two input fields: 'Username' and 'Password', each with a corresponding icon (a person for username and a lock for password). There is a 'Login' button and a 'Forgot password?' link. On the right, there is a large blue box labeled 'Admin Panel' with the text 'Welcome to the Lefke Market control panel'.

If store writes incorrect username or password, system gives error(Incorrect username or password). Otherwise login success!!

List of orders

After successful login, store will be redirected to main page, which is list of orders.

The screenshot shows the Admin Panel interface for Lefke Market. On the left, there is a sidebar with 'Orders' and 'Products' links. The main area displays a table of orders. The table has columns for '#', 'Client name', 'Address', 'Status', 'Date', and 'Total cost'. The orders are filtered by 'All' status. The table contains 4 rows of data.

#	Client name	Address	Status	Date	Total cost
# 1	+905488533853	Kyrgyzstan, City Bishkek	New	Today 6 AM:12	17 TL
# 2	+905488533853	Lau yolu Merhale 6	New	Today 6 AM:11	4 TL
# 3	+905488533853	Lau yolu test address	On way	29 Dec 2021 7 PM:08	27 TL
# 4	+905488533853	Tufekchi e blok	Packing	29 Dec 2021 6 PM:53	23 TL

Image(5.2.9)

As you see from the image(5.2.8) there are 6 types of order statuses.

- 1) All – you can see all your orders
- 2) New – New orders
- 3) Packing – If store accepts order, status will be Packing
- 4) On the way – If order type is delivery, and courier on the way, status of the order will be on the way
- 5) Delivered – After successful delivery

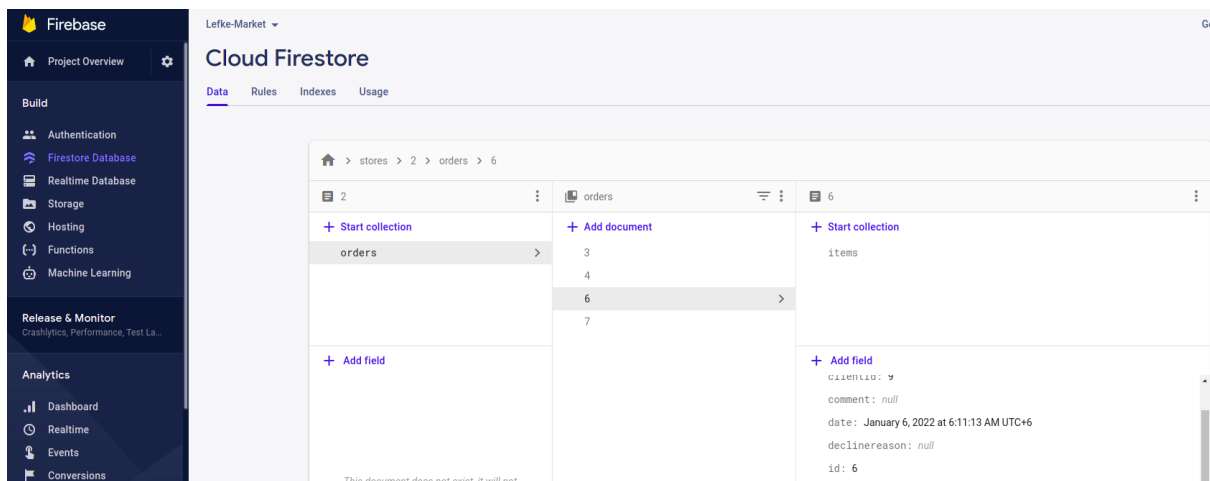
6) Declined – Declined order(there is a field decline reason, user and store can see it)

You do not need to update your page to get new orders of the store, all orders updating real time(it is asynchronous). But I am using Django Rest Framework which is synchronous. So how it is updating real time?

Django is synchronous, but it handles asynchronous protocol like WebSockets. Before starting project I thank to use DJANGO CHANNELS(WEB SOCKET) to update orders in real time.

Django channels - Channels preserve Django's synchronous behavior and add an asynchronous protocol layer that allows users to write fully synchronous, asynchronous views, or a combination of both. Channels basically allow an application to maintain "long-term connections". It replaces Django's standard WSGI with its ASGI.

But as a server, I am using HEROKU server which is free. Deploying project with websockets to HEROKU server is impossible. This server can not handle asynchronous projects. For now I can't buy a good server and pay each month for that. And I looked for another options, and decided to use **FIREBASE**(Firestore database) which is asynchronous. I already said about Firebase in Background information.



Image(5.3.0)

I used Firebase only for order part... Two backend for one project may be not good idea, but for my situation it was the best option. Orders will be saved either in my backend and Firebase. So how it is working?

Front side sends post request to the django backend (Image 5.2.8 Confirm order)

URL – mainURL/item/clientorder

DATA:

client – int (FK of client table) whose order

store – int(FK of store table) which store
status – int(default status is 1)
ordertype – int(1-delivery, 2- pick up)
address – string (address of client)
date – string (auto)
cart – int (FK of cart table) which cart
declinereason – string (initially it will be NULL)

Firstly, I am saving this order in my Django backend then inside django I am creating this order in Firebase. So as a Frontend developer I didn't send a post request to firebase, all this logic I did it in Django Backend. I just get orders from Firebase.

Create order for Firebase inside django backend

```
def create_order_in_firebase(saved_data, currentuser):  
  
    data = {  
        u'id': saved_data.id, u'clientId': saved_data.clientId,  
        u'phone': saved_data.clientId.phone, u'status': saved_data.status,  
        u'totalcount': saved_data.totalcount, u'ordertype': saved_data.ordertype,  
        u'address': saved_data.address, u'comment': saved_data.comment,  
    }  
  
    firestore.db.collection(u'stores')\br/>        .document(str(saved_data.storeId.id))\br/>        .collection(u'orders').document(  
            str(saved_data.id)).set(data)
```

Image(5.3.1)

As you see image above, I have a function - create_order_in_firebase and inside firstly I am collecting all client order data inside data dictionary then creating it in Firebase project using SET.

So when I am changing status of order in project(Ex: From New to Packing), I am sending patch(partial update) request to the django backend and from there changing order status in firebase same as create order.

```
def save(self, *args, **kwargs):
    if self.status == 2:
        firestore.db.collection(u'stores').\
            document(str(self.storeId.id))\
                .collection(u'orders').document(str(self.id))\
                    .update({"status": 2})
```

Image(5.3.2)

As you see picture above, when status of order become 2(packing) in django backend, in firebase it also changes.

Connecting Django backend with Firebase

To create order in firebase and update it we need to connect our django project with firebase project. So when we create a new project in firebase, they give us credentials in json file:

```
{
  "type": "service_account",
  "project_id": "lefke-market",
  "private_key_id": "370304738ec58d8050dfe2a7e6f6a9357126725b",
  "private_key": "-----BEGIN PRIVATE KEY-----\nmIIIEugIBADANBgkqhkiG9w0BAQEFAASCBAQwggSgAgEAAoIBAQC8Q1yWoQyIToFZ\nbbj\n-----",
  "client_email": "firebase-adminsdk-qqbcm@lefke-market.iam.gserviceaccount.com",
  "client_id": "104715324519952595938",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-qqbcm%40lefke-market"
}
```

Image(5.3.3)

Here, I created a serviceAccountKey.json file inside my django project and copied-pasted keys in the picture above(Image 5.3.3). After created a firestore.py python file and using firebase_admin library initialized the firebase project(Image 5.3.4). Firebase project connected successfully.

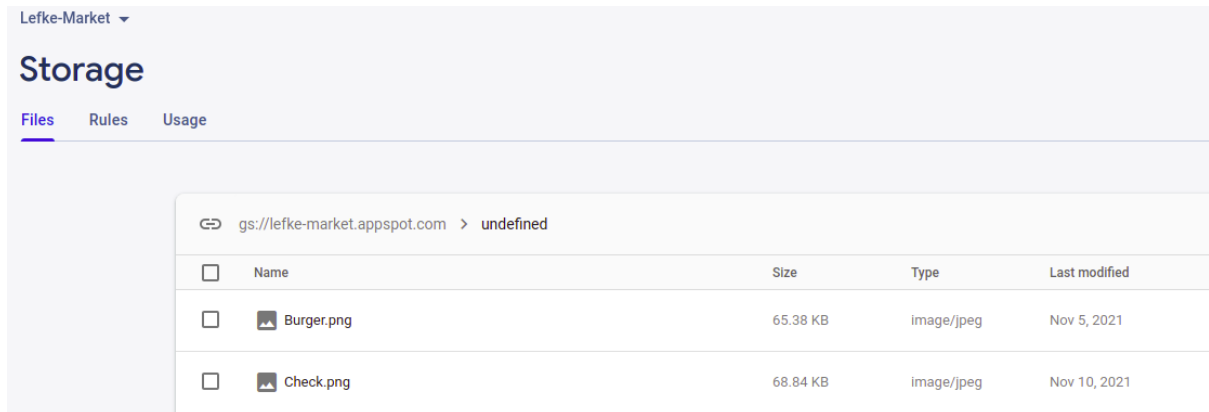
```
import firebase_admin
from firebase_admin import credentials, firestore

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred)
db = firestore.client()
```

Image(5.3.4)

If you look Image(5.3.1) in creating and updating order in firebase, I'm always starting with **firestore.db....** So here firestore is python file (firestore.py)

Also I'm storing all my media into firebase storage, because I already said my testing server is too small and works very slow



if store wants to see more detail about order and change status of order, they need to click to order. They will be redirected to order detail page (Image 5.3.4)

Packed ×

Phone number	Order ID	Address
<input type="text" value="+905488533853"/>	<input type="text" value="3"/>	<input type="text" value="Tufekchi e blok"/>
Order Date	Total sum	
<input type="text" value="29 Dec 2021 6 PM:53"/>	<input type="text" value="23.00"/>	
Client name	Order type	
<input type="text" value="Umutbek"/>	<input type="text" value="Delivery"/>	

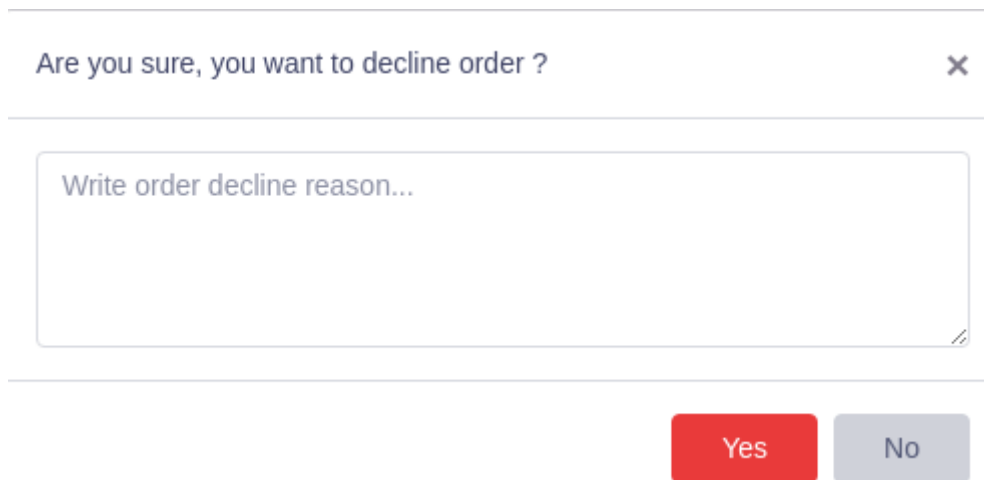
Items

#	Name	Cost	Quantity	Total sum
1	Red Bull	15	1	15
2	Fanta 1.5L	8	1	8

Image(5.3.5)

As you see from the picture (Image 5.3.5) there is all information about order and client. Now we are in order which is packing. In the right bottom of the picture you can see a two buttons(Ready for delivery and reject order). So if a order is already packed, they need to press button Ready for delivery, and courier needs to take the

order, after courier taking it will change to on way. If there is some problem with order, and they want reject it, they need to press button Reject order:



Image(5.3.6)

As you see from picture above(Image(5.3.6) when store declines order they need to write reason and after can do it.

So changing statuses and writing decline reason, how it is released?

In my Order table I have status and decline reason fields.

Status – initially will be 1 which is New

Decline reason – initially will be NULL

When a store tries to change order status:

I am sending PATCH(partial update) request to URL

mainURL/item/clientorder/{order_id} (order in detail now)

```
{  
  "status": 3  
}
```

if store wants to reject order, same request to same URL will be sending

```
{  
  "status": 5  
  "declinereason": "Test reason"  
}
```

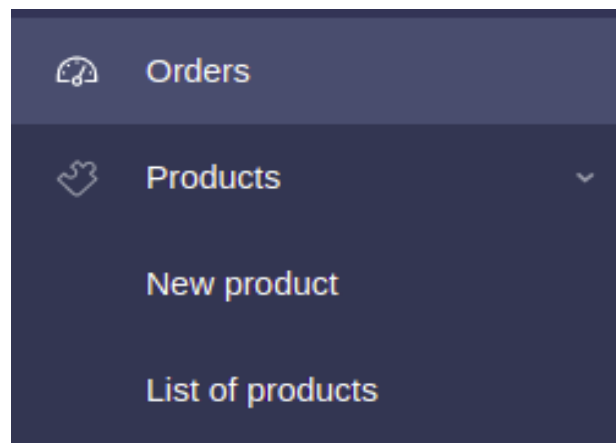
Order statuses in code:

```
class OrderStatuses(models.IntegerChoices):
    New = 1, 'New'
    Packing = 2, 'Packing'
    Delivering = 3, 'On way'
    Delivered = 4, 'Delivered'
    Rejected = 5, 'Declined'
```

I'm changing order not only in Django backend, in firebase too, I said about it in detail in (Image 5.3.2)

Create new product

If store admin clicks to product(Image 5.2.9) he can see:



Image(5.3.7)

Here admin can see list of own products and create new product. When you click to New product you will be redirected to new product create page (Image 5.3.8). But in this function user can add only one item. Of course adding products one by one is impossible for markets which has about 1000000 products. For them I have several options:

1) If they use an EXCEL, I will create them a page where they can upload all list of their

products in EXCEL file (I will convert it to CSV). They need to write product changings to their excel file (added new products, quantity updates etc....)) and upload it to given page.

I will set a time and using FTP(File transfer protocol) and django update_create function update website data. For example: If I set a time at 04:00 every day, the script will check products name and uniqueid which is in CSV file, if it exists in website db it will update the cost and quantity of product. Otherwise, script creates new product in website db.

2) If they use any system and have api's I can make integration with them, for now I don't have any idea, because I don't know what kind of system they use

A stores who want to add not too many products, they can add it one by one...

← Back

Item create

Name <input type="text" value="Enter name of product..."/>	Category <input type="text" value="Select..."/>
Description <input type="text" value="Enter product description..."/>	Subcategory <input type="text" value="Select..."/>
Cost <input type="text" value="Enter product cost..."/>	Subsubcategory <input type="text" value="Select..."/>
Quantity <input type="text" value="Enter product quantity..."/>	
Photo <input type="button" value="Choose File"/> No file chosen	

Reset Create

Image (5.3.8)
















List of products

If store admin clicks to List of products(Image 5.3.7) he will be redirected to List of store items page:

List of products

Search

New product

#	Name	Photo	Cost	Quantity		
1	Apple juice 1L		5	20		
2	Pasta Gemelli		4	12		
3	Apple juice (1Litre)		5.5	18		
4	Fanta 1.5		10	10		
5	Apple juice		6	95		










Image(5.3.9)

As you see image(5.3.9) admin can see list of store own products with quantity. If there is too many products system gives it 20 by 20 (pagination used, I explained it above). Admin can **search** their products by name, if there is a time, I will add search by item uniqueID. It searches by item name and decription. If you write app all items which consists app word will be shown (In image below, app – apple, apple juice)

List of products

Search

New product

#	Name	Photo	Cost	Quantity		
1	Apple juice 1L		5	20		
2	Apple juice (1Litre)		5.5	18		
3	Apple juice		6	95		

< > 1 < >

Item search (Image 5.4.0)

Admin can also update product and delete it

Are you sure, you want to Delete product
"Fanta 1.5" ?

Delete Cancel

Item delete(Image 5.4.1)

Update item

If admin clicks to update icon in Image(5.4.0) he will be redirected to update item

Item change "Apple juice 1L"

Name

Apple juice 1L

Description

Enter product description...

Cost

5


Quantity

20

Photo

Choose File

No file chosen



Category

Beverages

Subcategory

Natural Juices

Subsubcategory

Select...

Reset

Change

page:

As you see from picture above old datas are auto filled. Admin can change any of them and can reset it. To update it I'm sending PUT/PATCH(partial update) request to URL.

6. Conclusion

In conclusion, this project is very ambiguous. It is 21th century and it is time to move on online shopping in North Cyprus. It is beneficial for all and there are list of benefits for users and me:

6.1 Benefits

a) Benefits for stores

- Fast attraction of new clients;

45

- Increase in the number of orders and purchases;
- Organization of online sales;
- Reduction of transaction costs;
- Healthy competition;
- Minimization of fraudulent transactions;
- Modern and reliable business management;
- Rapid business expansion.

b) Benefits for clients :

- Optimal prices;
- Saving time;
- Preservation of nerve cells;
- Wide selection;
- Convenience in shopping.
- Make purchases without leaving your home
- Offer a convenient way to compare prices and products from a single source

c) Benefits to me :

- As I said already, it is very usefull and ambigious project, it will be good work experience for me.
- If I continue working on this project after graduation, I can make a good money.

For example:

- There is a banner advertising on the main page, from there you can also make good money

- Stores can't register themselves, if users start to work, and become popular website, stores want to join. Before registering them, I will take my share.

6.2 Ethics

- No single seller or buyer will so dominate the market that he is able to force the offers to accept his terms or go without
- All exchanges are fully voluntary. That is, participants are not forced to buy or sell anything other than what they freely and knowingly consent to buy or sell
- Buyers and sellers are free to enter or leave the market as they choose
- No external parties regulate the price, quantity, or quality of any of the goods being bought and sold in the market
- Costs and benefits of production or use exchanged goods fully bear those who buy or sell goods and not by anyone other external parties.

Why did I choose this project?

Before coming to North Cyprus, I often did an online shopping in my own country. When I came to first time in Cyprus, there was not online shopping system, and it was so strange. Several times I ordered from Aliexpress, but it takes 2 month which is too long time. That time I thank about this project, but not too serious.

Corona started, 2020 February, In quarantine time for 4 months I did not leave the house in Lefke. I ordered groceries to home. I did all this through a phone call, not knowing nether the price nor the photo. Every time you ask from store employee the price, is the product fresh. And I was sure about not only me calling them.

Sometimes you want to eat something unusual, you don't know what exactly, you ask something, it does not have in the store. I said then, God, this is the 21st century, and we live in which century. We urgently need a solution to the problem, and I intended to do this project.

I wanted to facilitate the work of both the store employee and the customer.

6.3 Future Works

After graduation I am planning to study masters degree and I am not sure for now can I continue it or no in future. But this project is very ambitious and I want to continue working on it in future. If I continue, I will add several new functions, some of them:

1) Chat

Direct chats inside web app. Client don't need to call store for any questions. Just write from website.

2) Write review for item, store

I could not finished this part on my first version, but in future I will add it. User can review item, store and rate it. Other user's can see it. It is very useful function, to get good reviews stores will work on their products and services

3) Online payments

Actually this task is not complicated, I just need contract with banks and their API's. But for now it doesn't have in project

4) Geolocation and region filter

For now this website only for Lefke region. Because I did not added region filter and geolocation. If I add this function, it will be useable in all regions, user can filter by region and see stores/items which is only on this region.

7. References

<https://www.cypazar.com/> all information about marketplace cypazar.com

<https://ftms.edu.my/> : Some ethics similar to my project

<https://www.geeksforgeeks.org/>: I get some used softwares, actually I know what I used and it's functionality, but I wanted to write it with academic words