

Ecommerce - Phase 1

**Program:**

***Course Code:* CSE411**

***Course Name:* Distributed Computer Systems**

***Examination Committee***

**Prof. Gamal Abdel Shafy Ebrahim**

**Ain Shams University**

**Faculty of Engineering**

**Fall Semester – 2021**

**Student Personal Information**

|  |  |
| --- | --- |
| 1700923 | عمرو ايهاب عبدالعزيز محمد عبدالعزيز |
| 1701040 | لؤي انور عبدالرازق حجازي |
| 1701048 | ليلي وائل سيد عبدالمنعم |
| 1701051 | لينة ايمن محمود |
|  |  |

Contents

[Introduction 4](#_Toc90075132)

[Target beneficiaries of the project 4](#_Toc90075133)

[Adopted programming languages 4](#_Toc90075134)

[System Architecture 5](#_Toc90075135)

[Application-Level Protocol 6](#_Toc90075136)

[Distributed Database Design 9](#_Toc90075137)

[Testing 12](#_Toc90075138)

[End-user guide 13](#_Toc90075139)

[Resources Needed 21](#_Toc90075140)

[Role of Each Member 22](#_Toc90075141)

[References 23](#_Toc90075142)

# Introduction

Our system is a distributed online marketplace that sellers can use to offer their products and get their profits. Sellers can browse all users’ products, view details of each product, and view all other users and stores. Sellers can view their stores, manage their products, and add other sellers’ products to his store to offer it. Sellers also can order and purchase other sellers’ products and provide their info to ship these products to him. Sellers can search for products for sale by other sellers and search for other sellers using username to browse their stores.

Each seller has to sign up and log into our system to benefit from our provided services and view his account to check his personal info, charge balance to purchase new products from other sellers, view his purchase history with the purchased items, and view his sold history with the sold items. In the users’ store, User can edit their products info to add more details or change the product image or price, add new product to his store to sell it later, and delete a specific product if he wants. Also, each seller can remove the other sellers’ products from his store or purchase it to edit, delete and sell into his store.

When the user purchases product from another user’s store, the money is transferred to the seller’s balance and the purchased product is transferred to the buyer store.

# Target beneficiaries of the project

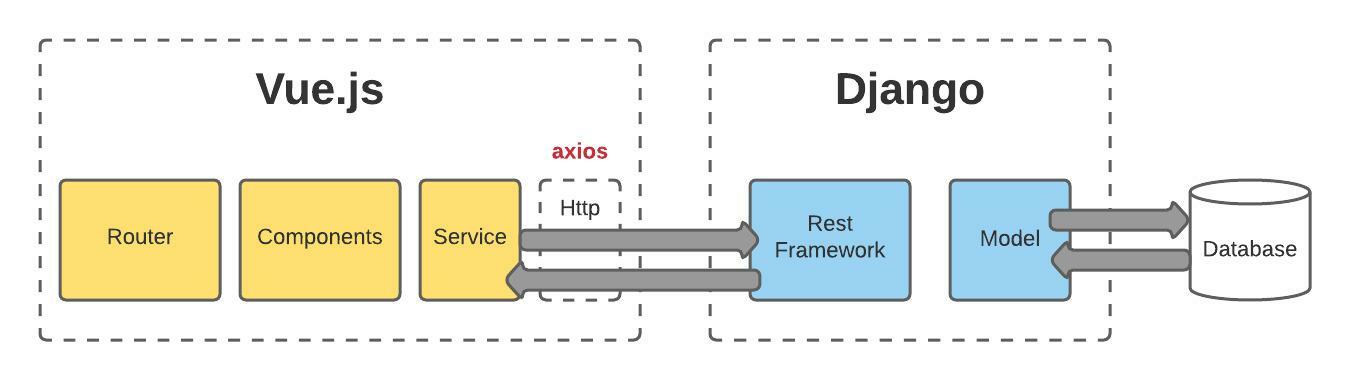
One of our marketplace system goals is to make the marketing of products much easier which helps small sellers to establish their business and get their first customers. We also aim to facilitate the communication among users to let them gain more experience dealing with other experienced sellers.

Customers can order products easily and support other junior sellers to pay attention or sell their own products and become our new client.

# Adopted programming languages

We used Django REST framework to build the API and Vue JS to build the front end of the project.

# System Architecture



We choose Django for our system which uses the Client-Server and MVC Architectures since it supports

**1. Rapid Development**

Actually, this Django architecture that separates in different components makes it easy for multiple developers to work on different aspects of the same application simultaneously. That is also one of the features of Django.

**2. Loosely Coupled Components**

This architecture of Django has different components which require each other at certain parts of the application, at every instant, that increases the security of the overall website. As the model file will now only save on our server rather than saving on the webpage.

**3. Ease of Modification**

This is an important aspect of development as there are different components in Django architecture. If there is a change in different components, we don’t have to change it in other components.

This is actually one of the special features of Django, as here it provides us with much more adaptability of our website than other frameworks.

# Application-Level Protocol

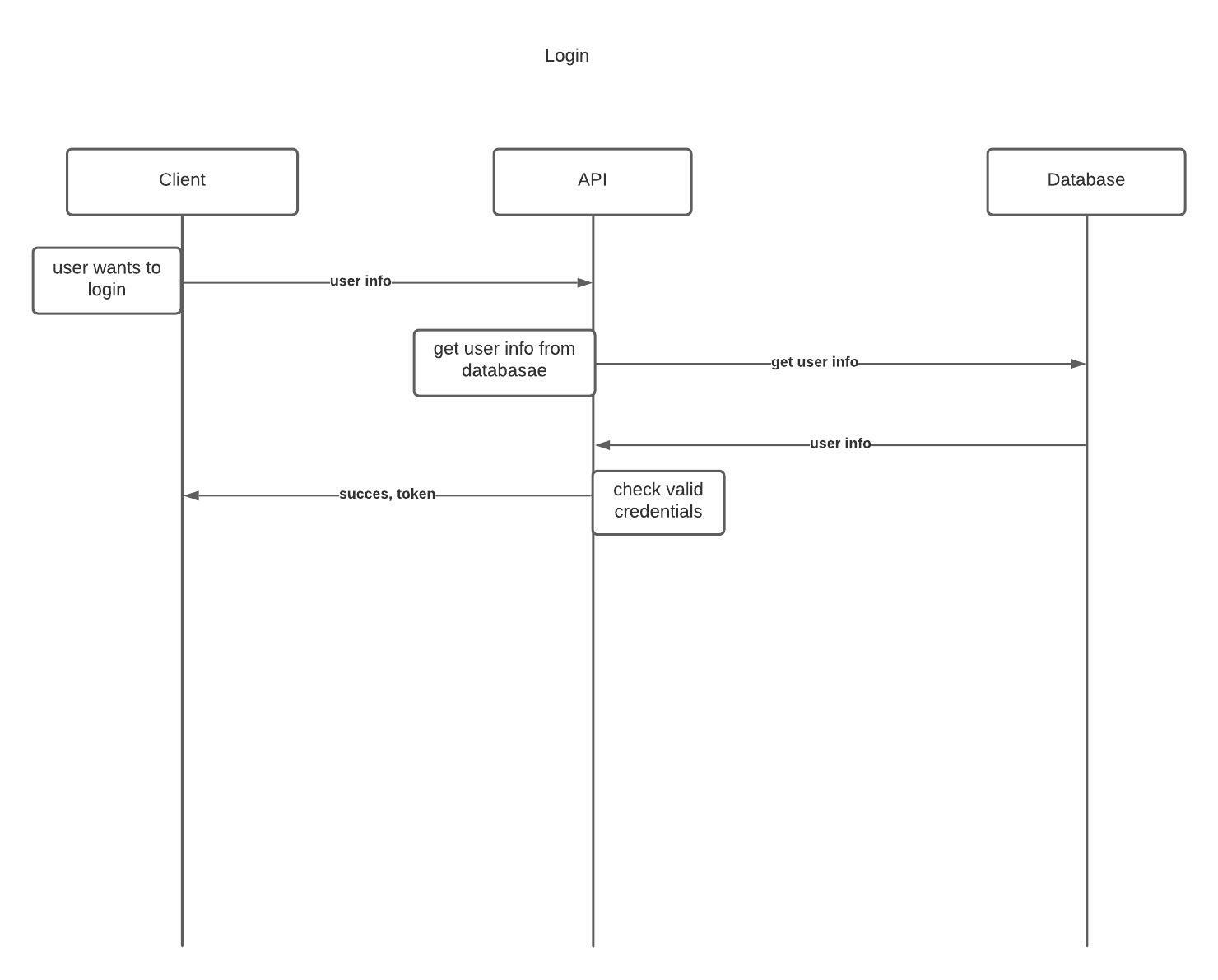


Figure 1 Login Scenario

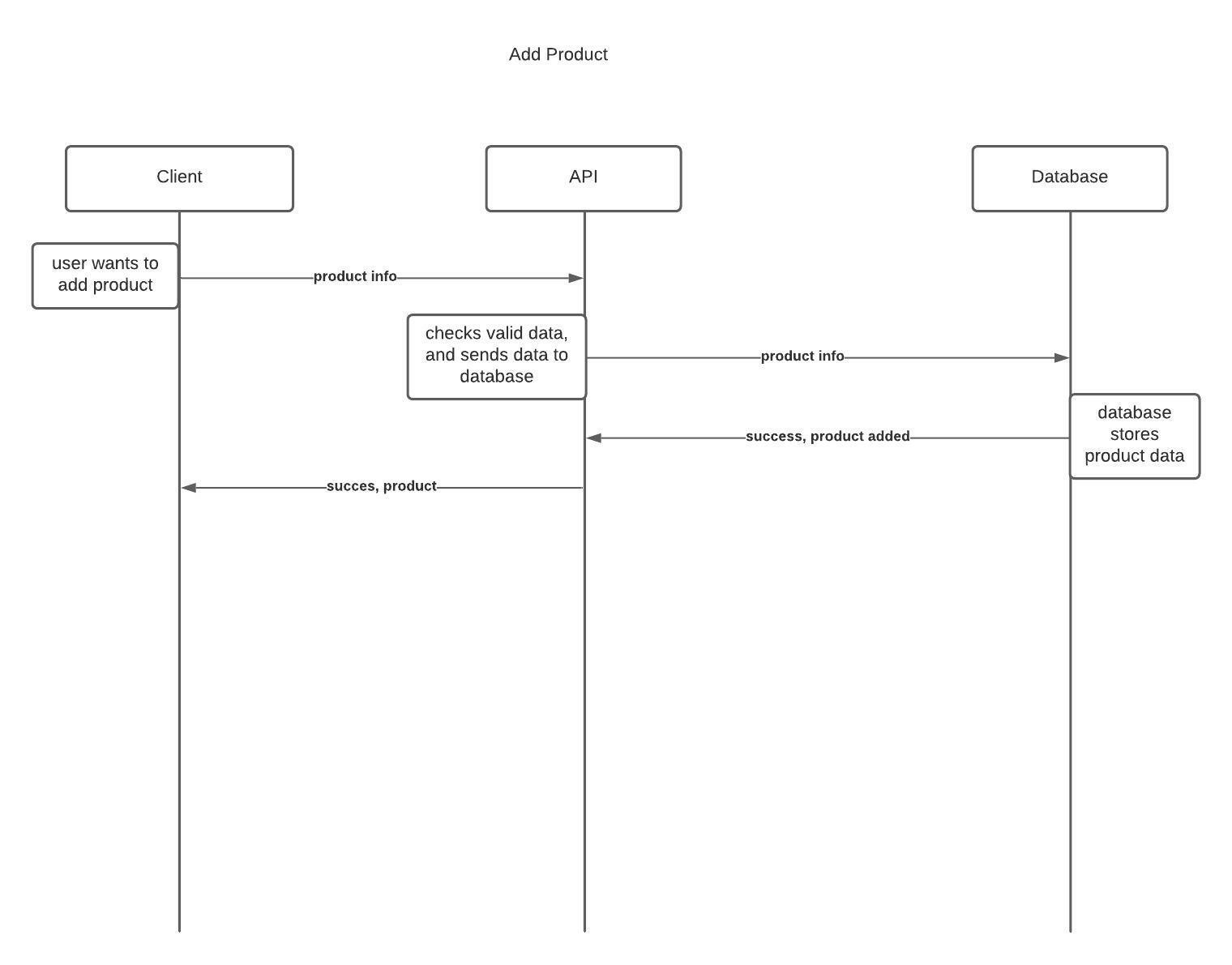


Figure 2 Add product scenario

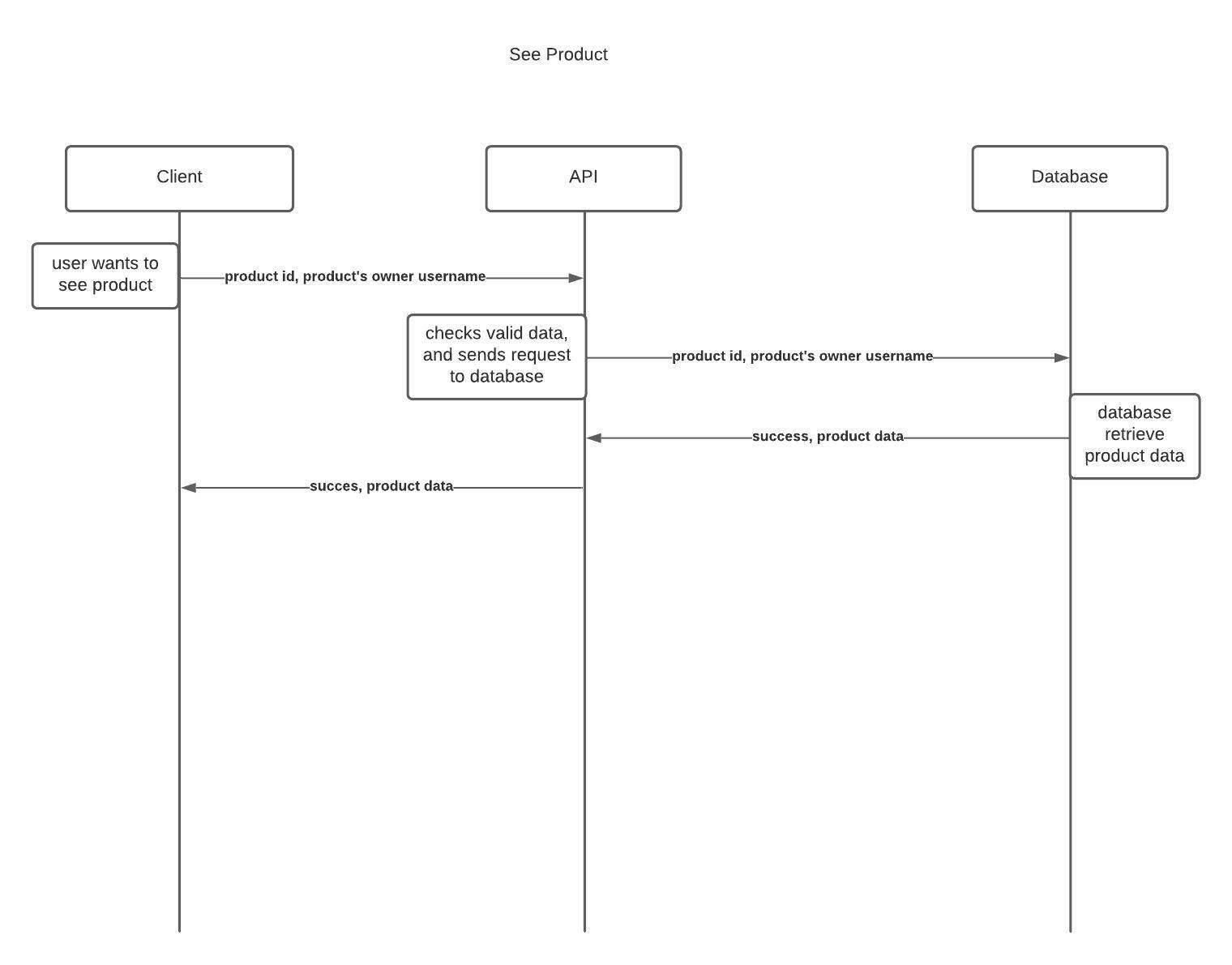


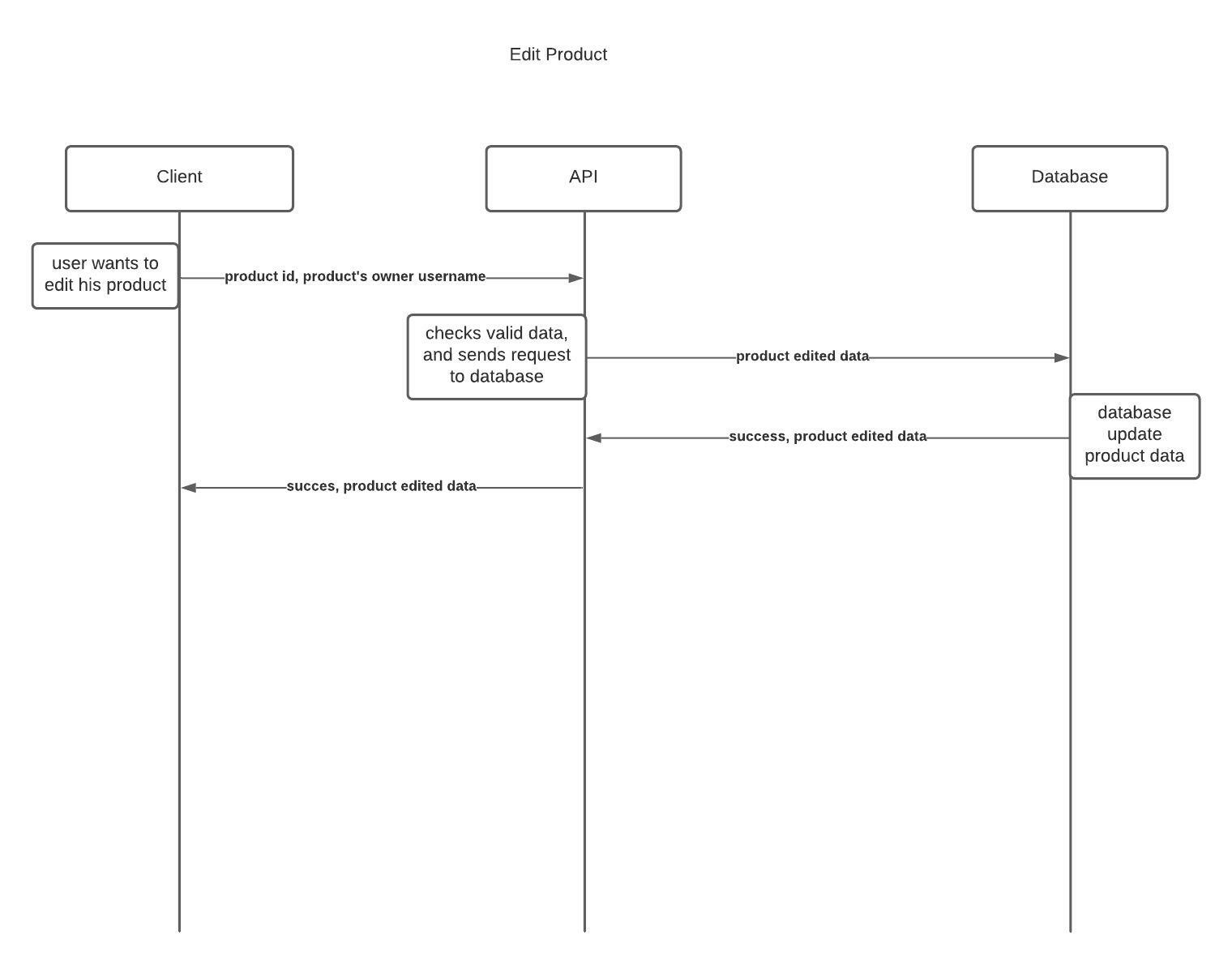
Figure 3 Get product details scenario

Figure 4 User editing their own product scenario

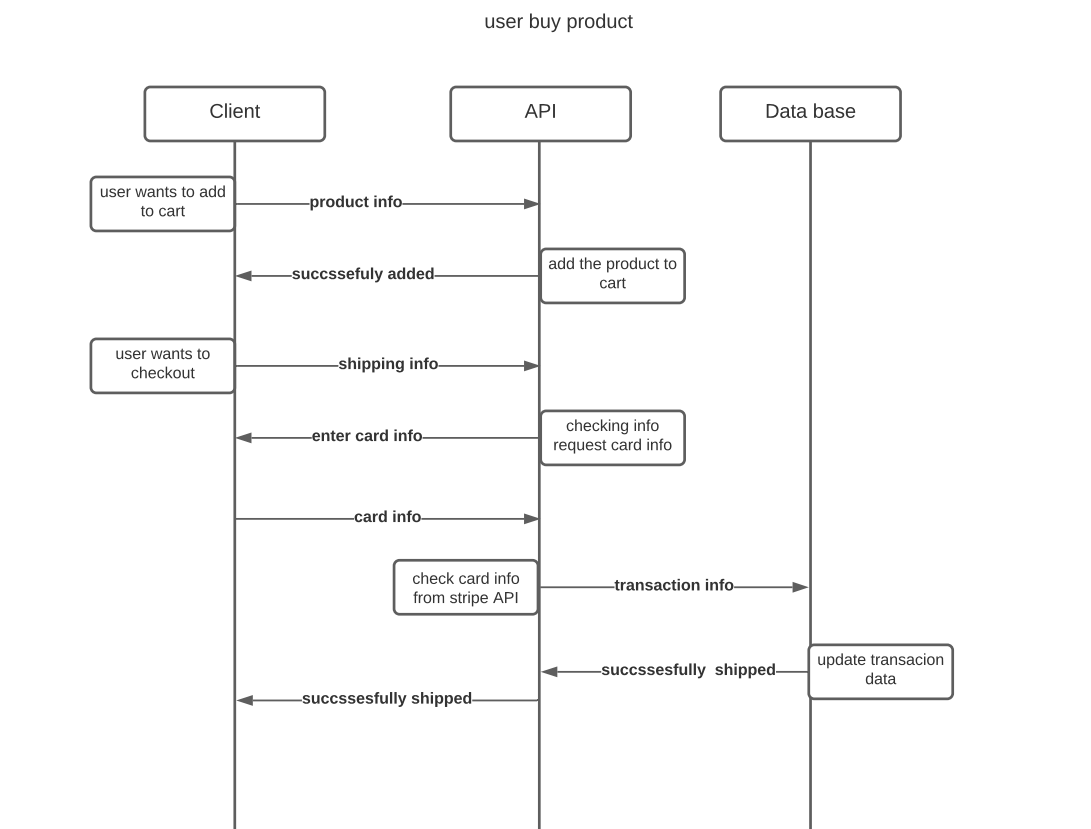


Figure 5 User buying a product scenario

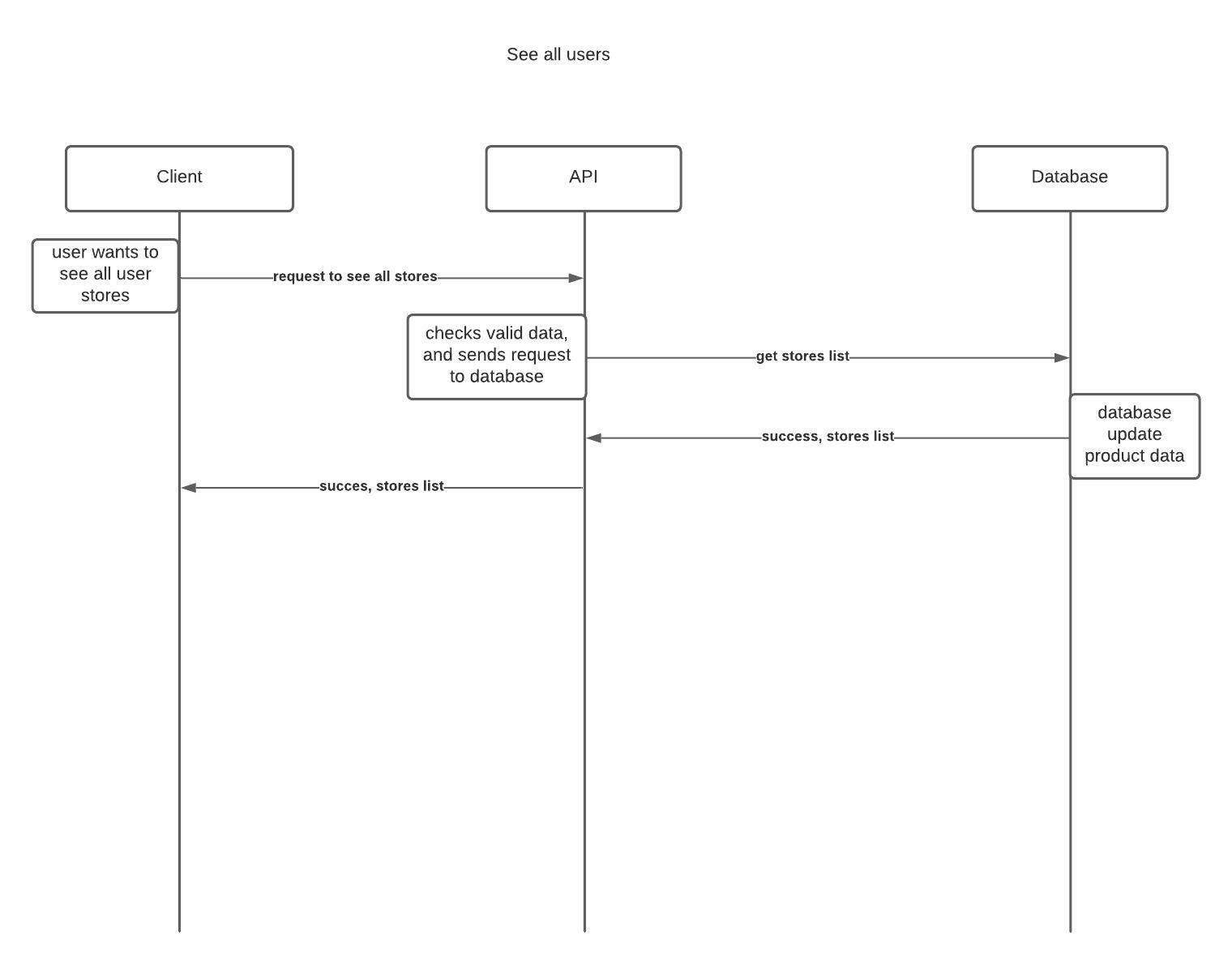


Figure 6 User browsing all stores list

# Distributed Database Design

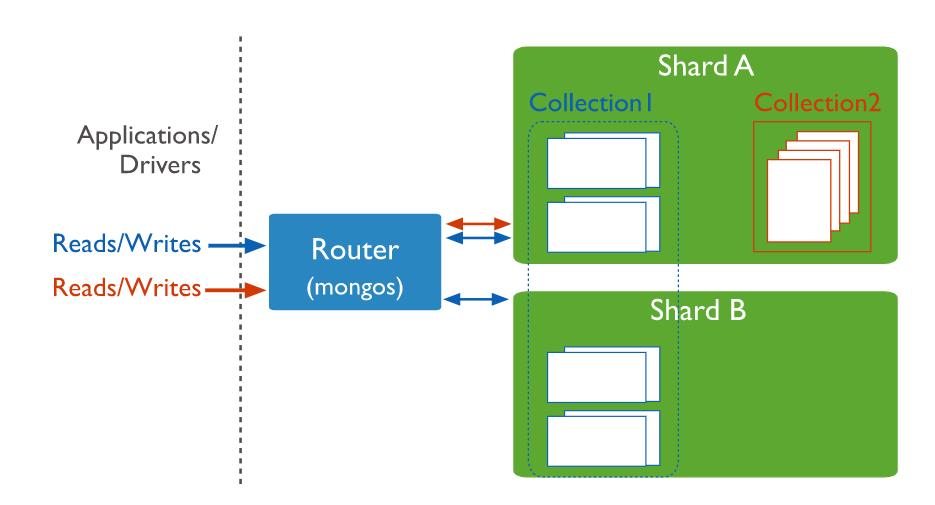
We are using MongoDB which doesn’t enforce a schema on the database.

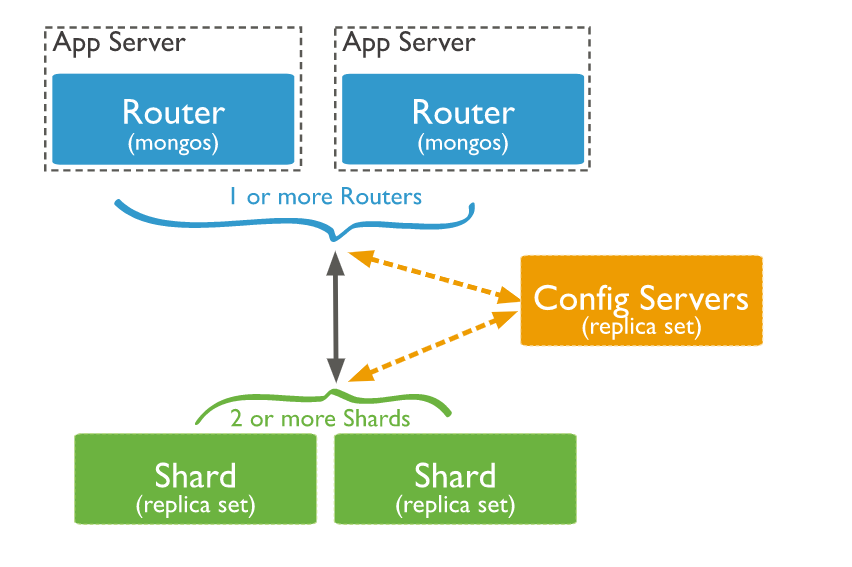
The database consists of collections which are stored in different “shards”. The database decides on which shard to place the record based on a chosen shard key for every collection.

In MongoDB when sharding is enabled, the collections are fragmented horizontally

The application connects to the different shards through a router which uses the info in the config server to find the records.

We used 2 shards and 2 replicas for every shard.





|  |  |
| --- | --- |
| Collection | Attributes |
| MyUser | first\_name  last\_name  email  username  not\_owned\_products many to many relation with a Product  balance  date\_joined  last\_login  is\_admin  is\_staff  is\_superuser |
| Product | owner -> foreign key to a MyUser  category  name  slug  description  price  image  thumbnail  date\_added |
| SoldProduct | seller -> foreign key to a MyUser  buyer -> foreign key to a MyUser  category  name  slug  description  price  image  thumbnail  date\_added |
| Order | user -> foreign key to a MyUser  first\_name  last\_name  email  address  place  phone  created\_at  paid\_amount |
| OrderItem | Order -> foreign key to an Order  Price |

# Testing

API Testing:

We used automated tests for views, models and urls in Django tests. We used postman to test all of the API endpoints.

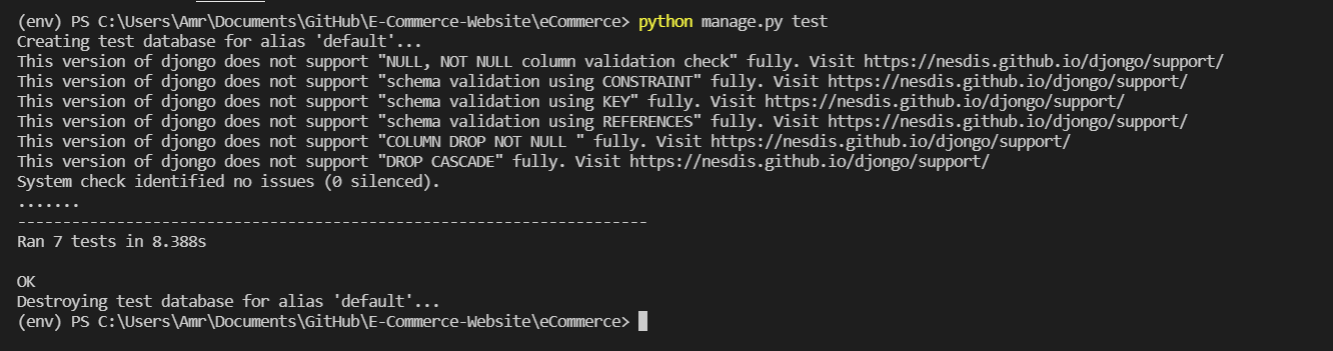


Figure 7 API Automated Testing

Sample of postman API end points testing

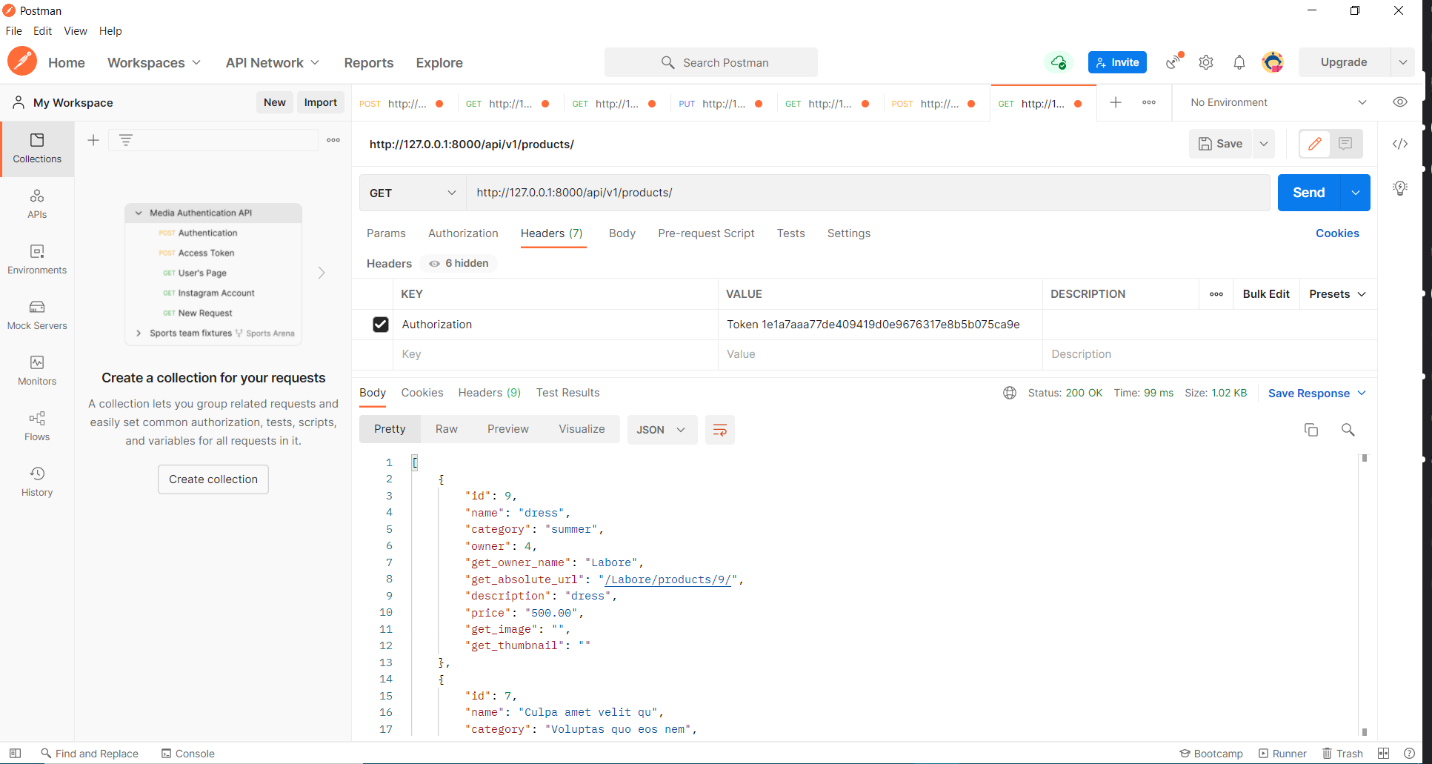


Figure 8 postman testing

# End-user guide

1. Signup:

First user needs to create an account with valid data.

Graphical user interface, application

Description automatically generated

1. Login:

Login into your account with the registered data.

Graphical user interface, application

Description automatically generated

1. Home Page:

Graphical user interface, website

Description automatically generatedYour home page that contains all the products that are sold by other users, where you can view any products to see more details.

1. Product Page:

In the view details page for each product, you will be able to choose whether to add the product to your card to buy it or add it to your store to display it inside your store page.

Graphical user interface

Description automatically generatedCart Page:

Table

Description automatically generatedYor cart will contain all the products you want to buy and when you are done adding products to your card you can go to checkout.

1. Checkout Page:

You need to enter a valid shipping info and click submit to successfully buy the products and it will appear in your store and will be delivered to the shipping address so you can sell it in your store.

Graphical user interface, application

Description automatically generated

1. Success Page:

Graphical user interface, text, application

Description automatically generatedThis page will appear after you buy a product to your store.

1. My Account Page:

My account page contains your personal info. and a button to charge your balance as well as your purchased history and sold history.

A picture containing graphical user interface

Description automatically generated

1. Charge Balance:

You can charge your balance using a valid card info.

1. Graphical user interface, text, application

   Description automatically generated Users Page:

Graphical user interface, application, website

Description automatically generatedUsers page contains all other users stores where you can view any store to see their products and buy it or add it to your store to be sold.

1. User Store Page:

Graphical user interface, website, timeline

Description automatically generatedIn view store page you can see all the products sold by this user.

1. My Store:

Graphical user interface, application, website, Teams

Description automatically generated My store page contains all the products that you sell where you can edit or delete them, and all products you sell for other users where you can buy them or remove them from your store. You can also add a new product.

1. Add Product:

Graphical user interface, application, website

Description automatically generatedAdd product page where you can add a new product to your store.

1. Search:

You can search for any product or user.

A picture containing shape

Description automatically generated

1. Admin Page:

Table

Description automatically generatedAdmin panel page contains reports about the transactions performed on the systems.

# Resources Needed

To run the server these requirements need to be installed.

asgiref==3.4.1

certifi==2021.10.8

cffi==1.15.0

charset-normalizer==2.0.9

coreapi==2.3.3

coreschema==0.0.4

cryptography==36.0.0

defusedxml==0.7.1

Django==3.2.9

django-cors-headers==3.10.1

django-rest-framework==0.1.0

django-templated-mail==1.1.1

djangorestframework==3.12.4

djangorestframework-simplejwt==4.8.0

djongo==1.3.6

djoser==2.1.0

idna==3.3

itypes==1.2.0

Jinja2==3.0.3

MarkupSafe==2.0.1

oauthlib==3.1.1

Pillow==8.4.0

pycparser==2.21

PyJWT==2.3.0

pymongo==3.12.1

python3-openid==3.2.0

pytz==2021.3

requests==2.26.0

requests-oauthlib==1.3.0

six==1.16.0

social-auth-app-django==4.0.0

social-auth-core==4.1.0

sqlparse==0.2.4

stripe==2.63.0

uritemplate==4.1.1

urllib3==1.26.7

To run the frontend, you need to install

Vue, axios, bulma, bulma-toast, moment using npm package manager

# Role of Each Member

All work was done together through online meetings and pair programming.

|  |  |
| --- | --- |
| Name | Contribution |
| عمرو ايهاب عبدالعزيز | API endpoints (backend)  Database  Report  Testing |
| لؤى انور عبدالرازق | API endpoints (backend)  Database  Report  Testing |
| لينة ايمن محمود | Database  Frontend  Report  Testing |
| ليلى وائل سيد | Database  Frontend  Report  testing |

# References

1. Django REST framework documentation
2. Bulma CSS documentation
3. Vue.js documentation