CAPSTONE PROJECT FINAL REPORT

Submitted to
Computer Science Department
College of Computing Sciences and Engineering
Kuwait University

Advisor

Dr. Hussain AlMohri

Group Members

Abdulrahman Fattah 2142129395 {Member}

Mohammed AlHaddar 2142127691 {Member}

Sheikha AlMishrie 2131110650 {Member}

Monday, 25th of November 2019

DataShop

Scalable Shopping System

Acknowledgements

Special thanks to Dr. Hussain AlMohri for his guidance and supervision on the project, in addition to holding our backs financially and spiritually. Without the guidance of Dr. AlMohri, we would not have reached this far and we would still struggling with the tiniest bugs and issues.

We would also like to thank Dr. Mohammad Samaoui for his efforts in explaining the basics of Amazon Web Services (AWS) and introducing us to making basic scalable cloud applications.

The team also appreciates Fatma AlKandari's efforts for suggesting and designing the system's logo.

We would also like to express our gratitude to all of our friends and colleagues at the computer science department for elaborating ideas with us and giving us support whenever possible.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Abstract

Begun in September 2019, the DataShop system is an online e-commerce website that features scalability, reusability, extensibility, and intelligence. The system is hosted on Amazon Web Services (AWS) to provide maximum scalability by using EC2's with the RDS database. The system is also designed to be reusable by relying mainly on components that support coupling and cohesion by crafting and dividing the functionality of the system carefully on its components. The component design of the system allows the system to be extensible in every manner, in addition to using an extensible framework which is Django and relying mainly on an open-source external component. Finally, the system uses machine learning to cluster users and products using hash maps to better categorize locations and offer products suggestions to the users of the system depending on their current location and browsing history.

Keywords: Scalability, Reusability, Machine Learning, Extensibility, E-Commerce, Data, Shop.

1. INTRODUCTION AND BACKGROUND

The e-commerce industry has changed the way business operations are carried out. E-commerce is a huge platform that is rapidly growing at unimaginable levels worldwide. Although a lot of online shopping systems are available, it is still reasonable for the team to build a new shopping system, called DataShop. As the name suggests, DataShop is a secure, usable, and above all portable and scalable online shopping platform. This report will go through a general description of the project, the motivations behind it, the deliverables with detailed features and design implementation, and the scope of work covered in the project.

First of all, Datashop is a website built by the Django framework using Python programming language for the backend, while using HTML, JavaScript, and CSS for the frontend. The system is deployed and hosted using Amazon Web Services to ensure maximum scalability and security. The system relies mainly on components and reusability concepts where every component was designed to be reusable and extensible by other developers of the system.

The system contains many features and functionalities that will be described in the coming sections, where each feature pours in the bowl of scalability, reusability, extensibility, and machine learning.

Datashop offers its users many features including suggesting products for shoppers using machine learning algorithms that are based on users' behavior and locations, while at the same time offers sellers with full control over their stores and products. The system was designed to be simple to use while offering straight forward functions for both types of users.

Many features and details regarding the system will be discussed in the coming chapters with full tests descriptions as a proof of concept that the system works as expected and it achieved all the requirements.

The main motivation behind this project is to create a system that can handle a huge number of requests and operations at a time to ensure the best user experience in addition to offering suggestions to users based on machine learning algorithms to enrich the usage of the system.

This report includes the discussion and description of the final deliverable of the project which includes this report, the full working code, testing to prove correctness, and a live demo to show how each function can be used and how it works.

1.1 BACKGROUND

The project is built mainly to serve the E-Commerce field which is focused on offering online shopping experience for users worldwide neglecting the location and obstacles between the seller and the shopper. E-Commerce is a system of purchases and sales of products and services, transfers of money and information via electronic media (Internet). This enables people to do business without distance and time barriers (Bhalla, 2019).

Different eCommerce business solutions may be required or used by different types of users, including online business models. Many different approaches and methods must be applied by different business types to ensure the full user experience. E-commerce systems include:

- B2B businesses
- B2C businesses
- Affiliate marketing business
- Google AdWords and AdSense marketing
- Online auction selling
- Web marketing

E-commerce system work by providing many functionalities that are pretty much like the offline (actual) stores and retails work, but with a broader and wider scale to include users from all over the world. Such systems rely mainly on three main components which are the browsing components, the order component, and the shipping component.

The browsing component enables shoppers to browse the digital stores and see the different types of products they offer; this includes adding media to present the product to a shopper and appropriate information for each product. Then, the shopper can choose a set of products to be added to an order and the order processing component takes place.

The order component handles the management of orders by shoppers where they can see the products they ordered, see proper information about the order such as total price and number of products, and proceed to checkout and pay to receive the order at their doorsteps.

The shipping component is responsible for transmitting the order to the sellers alongside with the shopper information so that sellers can prepare the order, deliver it to the shopper, and further communicate with the shopper regarding the order that connects the two entities (Bhalla, 2019).

Taking into account how E-commerce systems work, the project had to be built to provide the mentioned above functions with extra features to the potential users. The project had to follow a software engineering process to ensure that it can be constructed on time and delivered as expected. The chosen and followed software engineering process is Scrum with is wildly used and known to deliver a project in increments and allow multiple developers to work on the project effectively.

Taken from the official website of Scrum, “Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.” (ScrumTeam, 2019).

The scrum framework is heuristic; it is based on continuous learning and adaptation to factors. This recognizes that at the start of a project the team doesn't know everything and will learn through practice. With the re-prioritization embedded into the system and short release cycles, Scrum is designed to help teams to adjust themselves to changing conditions and user demands in a natural way.

Scrum is not completely rigid, although it is structured. It can be applied to fit each project’s requirements and fulfill its design and implementation. To use Scrum successfully, there are many hypotheses about the precise way scrum teams can work, but after more than a decade of helping agile teams get their work developers learned that the priority should always be clear communication, openness and a dedication to continuous change in any project, which ensures the balance.

Scrum has many artifacts that must be created to ensure the best results from it. The following list presents each one with its description:

- Product Backlog is the master list of work that needs to get done maintained by the product owner or product manager. This is a dynamic list of features, requirements, enhancements, and fixes that acts as the input for the sprint backlog.
- **Sprint Backlog** is the list of items, user stories, or bug fixes, selected by the development team for implementation in the current sprint cycle.
- Increment: (or Sprint Goal) is the usable end-product from a sprint (AtlassianTeam, 2019)

2. USER AND SYSTEM REQUIREMENTS

This section contains brief descriptions and details about the project's different features and requirements and the approach that the team followed to elicit them, in addition to some figures that show how did the team reach the point where everything is clear and applicable.

2.1 REQUIREMENTS ELICITATION APPROACH

The requirements elicitation took a set of approaches and methods to come up with the final list of requirements for the project. The methods used in this process were all shared by the stakeholders and each member/method contributed to the requirements discovered and agreed on. The main requirements elicitation methods used to gather requirements are:

Prototyping, which is the main requirements elicitation approach the team uses for eliciting requirements. This approach involves creating a prototype, exposing it to the stakeholders of the project, and taking feedback to modify/add requirements. The prototype created for such a job is not a through away one, the actual system is going to be built using increments on this prototype which contains the base requirements for the system. Figure 2 shows the prototyping requirements elicitation steps.

Brainstorming, which is done by the whole team in a way that best suits the project idea and the domain of e-commerce projects. The brainstorming was held at the start of the project by the team to try to figure out what requirements are mandatory and which of them can be achieved taking into consideration the team member's experiences, guts, and knowledge.

Document Analysis, which was conducted by the team to better visualize the domain of interest and have a general knowledge about the do and don’ts of the domain. The document analysis included reading and scanning of many scientific papers and researches focused on the domain of interest.

Interface Analysis, which each team member was exposed to a certain set of e-commerce website and each one extracted a set of requirements that are considered essential to these websites.

Interviews, which included the gathering of user stories which on the other hand were collected from a set of stakeholders interested or already involved in the domain of e-commerce. The users whom the stories were collected from were either admins, store owners, or regular

customers who use online shopping in their daily life. The team got a satisfiable set of user stories which will be presented in later sections.

Requirements Workshops, which were conducted partially by the team members to better understand the system and extract requirements.

Figure 1 shows a sequence diagram for how the requirements elicitation approach occurred.

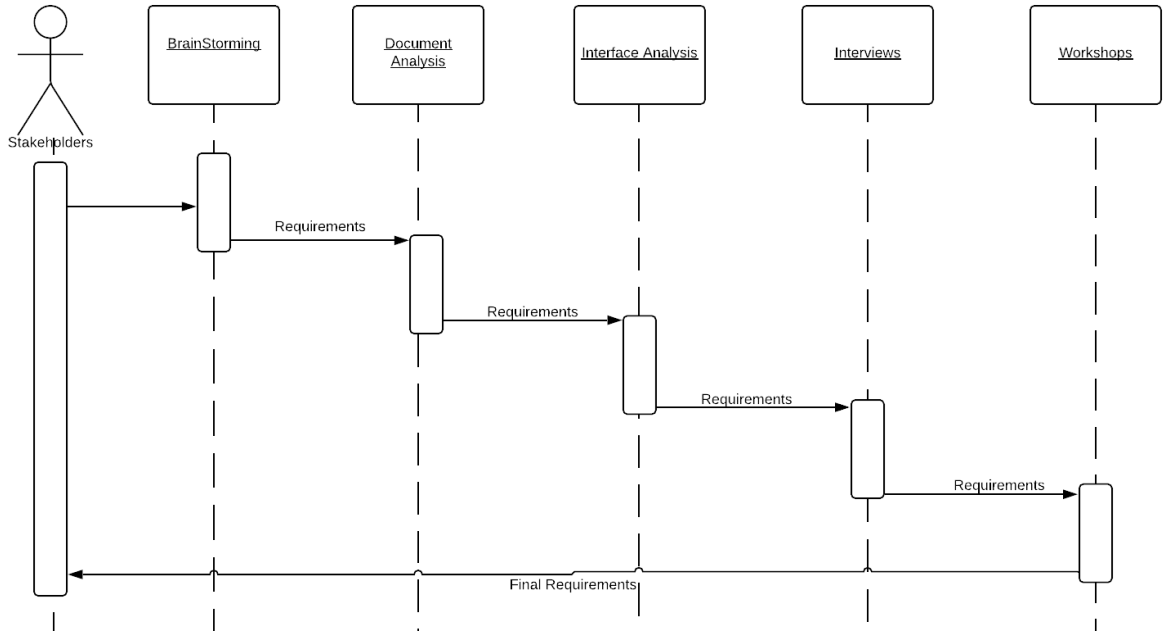


FIGURE 1REQUIREMENTS ELICITATION SEQUENCE DIAGRAM

2.2 REQUIREMENTS SPECIFICATION

The requirements specification is a process under which the description of what the system will do and how it is expected to perform is conducted. The SRS is also used to describe the functionality the users and the business going to use in the project. Taking this into consideration, an SRS must include a purpose, and overall description to be a reference for the project, and requirements specification of the system/software as a whole.

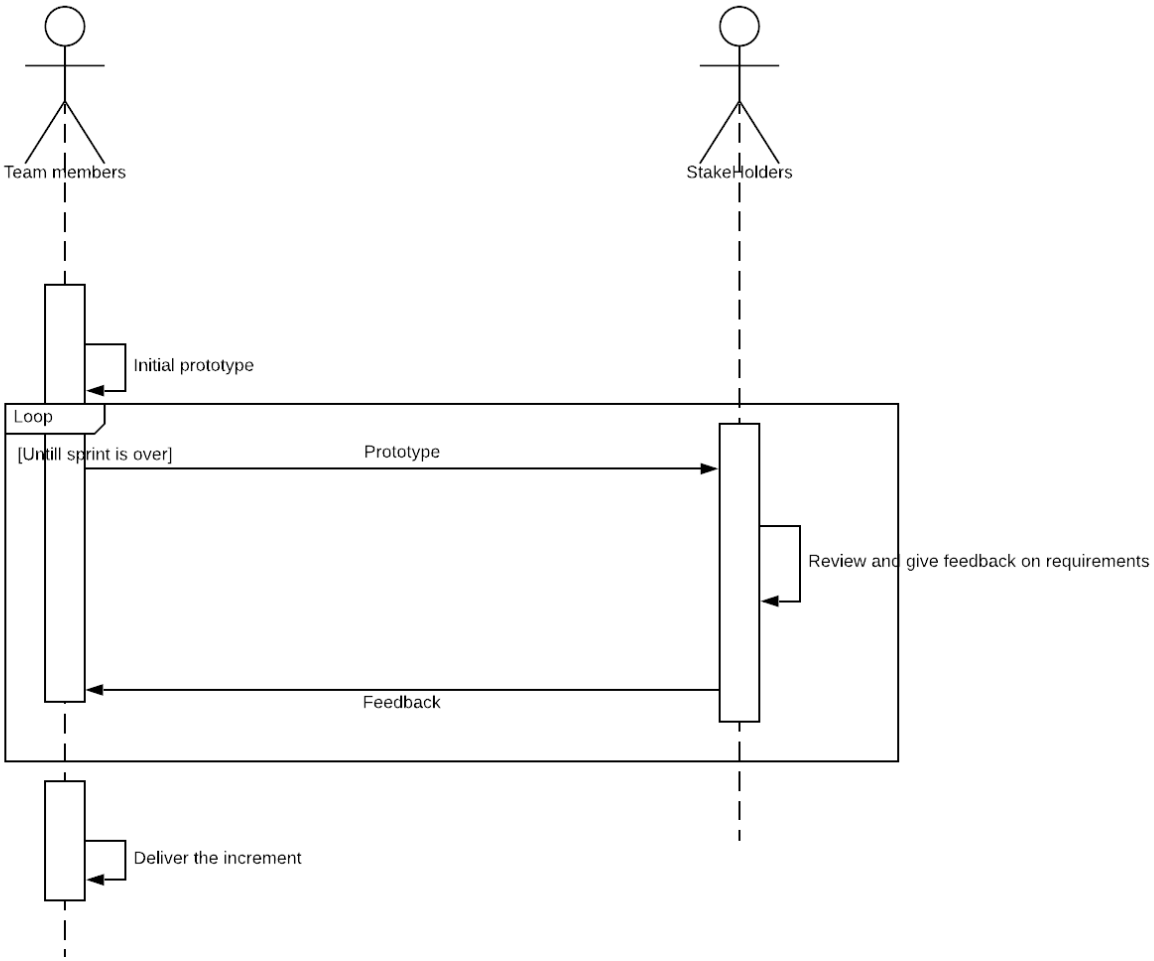


FIGURE 2 PROTOTYPE SEQUENCE DIAGRAM

i) Purpose of The Requirements Specification

The purpose of the requirements specification if to have a reference on the requirements the system needs and how the system is going to work.

ii) Description of The Requirements

The system is mainly described as a smart, reliable e-commerce shopping website that serves store owners and customers by following the B2C business model. The system is an interactive system, which means that customer behavior can be analyzed to give the best experience and suggests certain products to each customer. The system also has all the essential functionalities of an online shopping system which are going to be listed in the following sections. Providing such functionalities and features, the system is going to be hosted on Amazon Web Services which provides a secure and reliable environment for websites to run on and function safely.

Moreover, the system is going to be built to be used in two modes, plug-and-play, and configure-and-play modes. The plus-and-play mode is achieved by loading the system with default configuration that best suits the general use of such a system and which an admin can run in one click. The configure-and-play mode is achieved by providing a configuration tool for the system before it starts such that system administrators can toggle preferred settings and features before they run the system.

iii) Intended Audience of the Study

This SRS is intended to all the stakeholders involved in this project, especially to the Capstone Committee who are the main customer of this project.

iv) Intended Use of the Study

The use of this SRS is mainly as a reference and a grading sheet for the system. The developers are going to use it as a reference on what to implement and how to do so, while the project manager (the supervisor) and the customer (the CPC) will use it as a documentation and a grading sheet to mark progress on the project.

v) Scope of The Study

The scope of the SRS is the whole system, starting from the hardware specification, and ending with normal customers' requirements.

vi) System features and Requirements

a. System Features

- Search for e-stores, and furthermore search for products using the website. Meaning that the system users will be able to perform a search based on e-stores or products. The search for product functionality is performed across all stores, giving the view of the system to the user as one giant e-store.
- A safe and trusted payment method that connects all the e-store under one receipt which the user can pay once for all the stores. The system will have a gateway allowing other interfaces to be integrated to offer the functionality of the payment system and will be connected to more payment gateways if time allows.
- Generic store creation for the e-store owners under which they can customize their store pages and put their flavor of design. The generic store design will handle the look and feel of each e-store homepage and give the e-store owners the ability to customize as much as they want while they can integrate their contact information as easily as possible. The customization of the e-store page will be wizard-driven.
- Analysis of user behavior and displaying product recommendations. This feature is created to serve both customers and business owners in a way that shows recommended products to the customers while at the same time advertise for the e-store owners.
- Custom favorite list creation for users, in which users can add e-stores and products for private a list associated with their accounts. This list is per user and can be viewed using smart filters.

- Order status update and order panel for the store owners, under which each user will be able to follow the status of his orders while store owners can update the status of the products manually using the system.
- Provide direct contact with the e-store owners by serving contact information on the e-store page (email address, social media, etc...), in addition to ticket submission and tracking their status by the owners.
- Order history for a future revision, so that the user can revise and reorder the same order again.
- Display detailed information on each product so that users have no doubts about a product.
- Provide user accounts such users can create accounts and use the system. The user accounts hold user's information such as name, address, preferred products, and lists. The user accounts are also used for tracking orders and revising them.
- Has categorization on e-stores and products. In other words, the system provides browsing by categorizing depending on the e-stores or products.
- A scalable system can hold a variable number of user requests without the need for human interaction. The system will use smart and machine learning algorithms to adjust the system state so that it can handle the current number of user requests.
- It provides an email notification system to warn and communicate with users of the system. The notification system is built on the email service and will notify the customers of any update to their orders while notifying the store owners of any newly received order.
- System admin interface to configure and control the system. This interface will be for admins only and will provide them with the general settings and configuration facility to configure and adjust the system state.

b. System Requirements

The requirements of the whole system are going to be classified and listed in the section. Each requirement is going to be listed according to source (prototyping, brainstorming, interface/documents analysis, workshops), type (functional, nonfunctional), and category (security, interface, performance, network, data).

Number	1
Content	Email must be the login key to the system
Source	Brainstorming
Type	Functional
Category	Interface

Number	2
Content	Password must be strong or at least has medium strength
Source	Brainstorming
Type	Nonfunctional
Category	Security

Number	3
Content	Products must be able to have multi images
Source	Interface Analysis
Type	Functional
Category	Interface

Number	4
Content	Products must be categorized on a minimum of 2 levels
Source	Document Analysis
Type	Functional
Category	Interface

Number	*5*
Content	Multi-currency support can be configured at the admin initialization of the system
Source	Workshops
Type	Functional
Category	Interface

Number	*6*
Content	Each user must have a profile
Source	Prototyping
Type	Functional
Category	Interface

Number	*7*
Content	Shoppers and store owners share one module, a user can be both a shopper and a store owner
Source	Workshops
Type	Functional
Category	Interface

Number	*8*
Content	Cart must auto calculate the total price of the products
Source	Workshops
Type	Functional
Category	Interface

Number	9
Content	Each product must have its page that holds its information
Source	Interface Analysis
Type	Functional
Category	Interface

Number	10
Content	Only logged in shopper can use the cart
Source	Interface Analysis
Type	Functional
Category	Interface

Number	11
Content	Stores can have their contact information point to all popular social networks in addition to phone numbers.
Source	Brainstorming
Type	Functional
Category	Interface

Number	12
Content	When an order is shipped, it is automatically saved in history orders
Source	Workshops
Type	Functional
Category	Interface

Number	13
Content	The store owner can add, remove, and update products
Source	Prototyping
Type	Functional
Category	Interface

Number	14
Content	Shoppers can add/remove items to the cart
Source	Prototyping
Type	Functional
Category	Interface

Number	15
Content	Use Bootstrap to make the interface simple and elegant
Source	Prototyping
Type	Functional / Nonfunctional
Category	Interface

Number	16
Content	Sellers can register to the system by using a special signup page, the registration form includes description, new business or not, have physical store or not and a logo
Source	Prototyping
Type	Functional
Category	Interface

Number	17
Content	Users (shoppers and anonymous) can browse all the stores in the system
Source	Prototyping
Type	Functional
Category	Interface

Number	18
Content	A store owner can have multiple stores in one account
Source	Prototyping
Type	Functional
Category	Interface

Number	19
Content	Multi-currency support for each store
Source	Interface Analysis
Type	Functional
Category	Interface

Number	20
Content	Storeowners can see analytics of their products
Source	Workshops
Type	Functional
Category	Interface

Number	21
Content	Provide a backend API for integration with other systems
Source	Workshops
Type	Nonfunctional
Category	Interface

Number	22
Content	Shopper can contact sellers using a ticket system which opens a semi-discussion between the two entities.
Source	Workshops
Type	Functional
Category	Interface

Number	*23*
Content	System search can either be store-wise or product-wise.
Source	Workshops
Type	Functional
Category	Interface

Number	*24*
Content	Shoppers should be able to pay throughout the system to receive their orders
Source	Workshops
Type	Functional
Category	Interface

Number	*25*
Content	Anonymous users who don’t have an account can browse and use the system
Source	Workshops
Type	Functional
Category	Interface

Number	*26*
Content	User behavior should be analyzed for better product recommendations.
Source	Workshops
Type	Nonfunctional
Category	Interface

Number	*27*
Content	Shoppers can add products/stores to private favorite lists for easier access.
Source	Workshops
Type	Functional
Category	Interface

Number	28
Content	Both shoppers and sellers can track ordered by status updates using the system.
Source	Workshops
Type	Functional
Category	Interface

Number	29
Content	Past orders can be saved in order history and can be reordered again
Source	Workshops
Type	Functional
Category	Interface

Number	30
Content	Each shopper must have an address registered in the system before an order is initiated
Source	Prototyping
Type	Functional
Category	Interface

Number	31
Content	Allow predictive scaling on the backend (AWS) to handle system stress at peak times.
Source	Workshops
Type	Nonfunctional
Category	Performance

Number	32
Content	The system should be fast in terms of response to client requests.
Source	Workshops
Type	Nonfunctional
Category	Performance

Number	33
Content	The system interface should be simple and constructive
Source	Interface Analysis
Type	Nonfunctional
Category	Interface

Number	34
Content	Create an EC2 instance and host the system’s website on it.
Source	Workshop
Type	Nonfunctional
Category	System

Number	35
Content	Host the system on Docker
Source	Prototype
Type	Functional
Category	System

Number	36
Content	Host Docker on the EC2 instance
Source	Prototype
Type	Functional
Category	System

Number	*37*
Content	System requests will be logged using RDS by AWS, and the database engine that will be used is PostgreSQL
Source	Workshop
Type	Functional
Category	System

Number	*38*
Content	Stores can be deactivated using profile
Source	Prototype
Type	Functional
Category	System

Number	*39*
Content	Empty search queries are not allowed in the system
Source	Prototype
Type	Functional
Category	System

Number	*40*
Content	Make the UI mobile friendly
Source	Prototype
Type	Functional
Category	System

Number	41
Content	Create a view for the store owner to view the sales of the store in a chart
Source	Prototype
Type	Functional
Category	System

Number	42
Content	Add a basic authorization module to your system such that every single view calls the authorization
Source	Prototype
Type	Functional
Category	System

Number	43
Content	Restrict sellers to a fixed number of products added per hour. A user should not be allowed to add 1000 products an hour,
Source	Prototype
Type	Nonfunctional
Category	System

Number	44
Content	Unified database for all instances
Source	Workshops
Type	Nonfunctional
Category	System

Number	*45*
Content	Unified storage(S3) for all instances
Source	Workshop
Type	Nonfunctional
Category	System

Number	*46*
Content	Implement registration access code that can be used to restrict the registration of shop owners.
Source	Brainstorming
Type	Functional
Category	Security

Number	*47*
Content	Populate the database with multiple stores and with each store populated with multiple products.
Source	Workshop
Type	Nonfunctional
Category	Performance

Number	*48*
Content	Find a methodology for logging data, now when the system logs data, it's logging the UNIX timestamp, the path of the request, Http method(GET/POST), remote address of the user, the users' ID, and location of the request
Source	Workshop
Type	Nonfunctional
Category	Security

Number	*49*
Content	Create a load balancer to distribute traffic among servers and to increase the reliability of the system.
Source	Workshop
Type	Nonfunctional
Category	Performance

Number	*50*
Content	Simulate user behavior, by preparing a questionnaire to seek feedback on the experience of a group of people who tried using the system.
Source	Brainstorming
Type	Nonfunctional
Category	Usability

Number	*51*
Content	Use AWS RDS service to host the system database for more scalability
Source	Workshop
Type	Nonfunctional
Category	Performance

c. Activity Diagrams of The Requirements

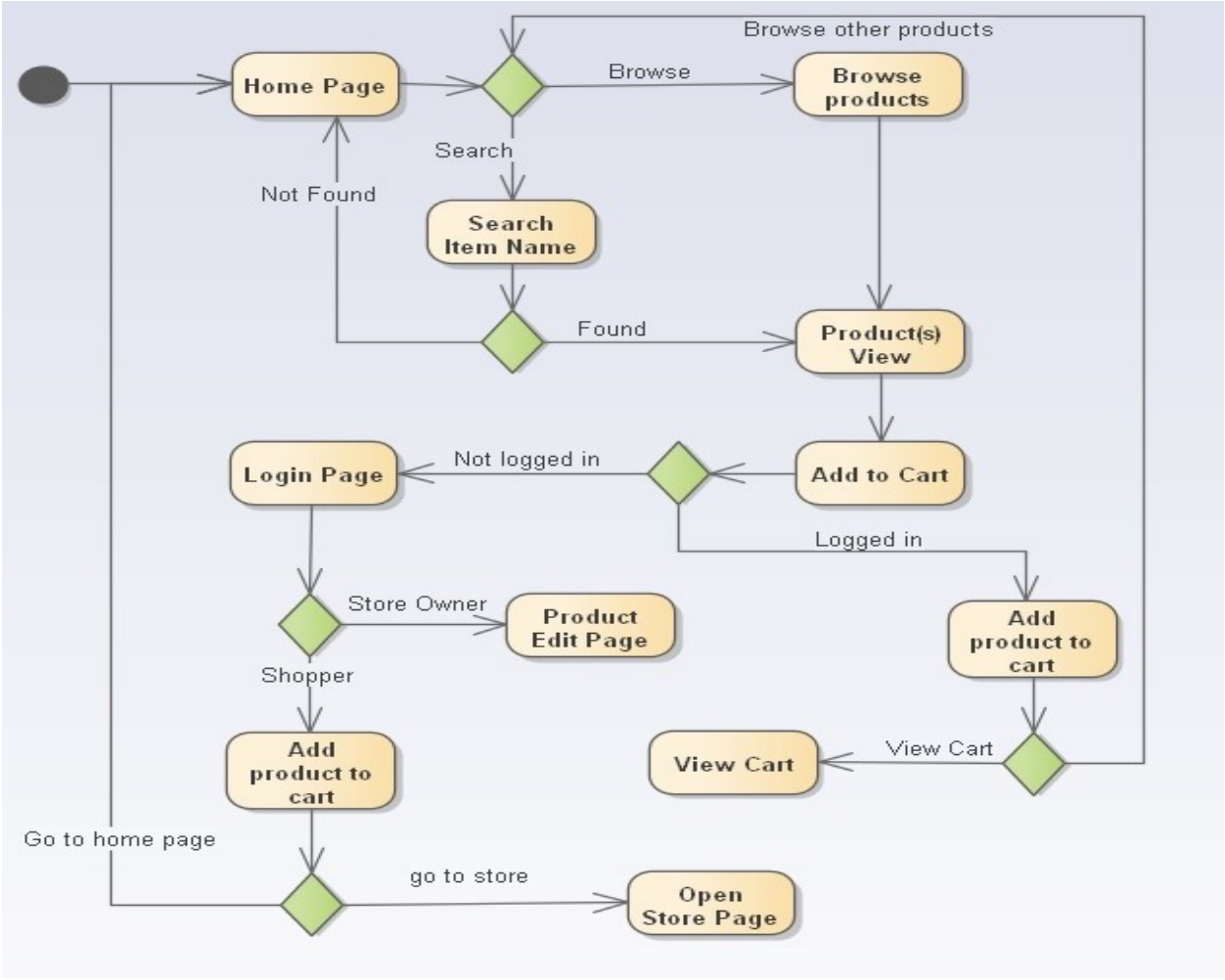


FIGURE 3 THE STORE ACTIVITY DIAGRAM

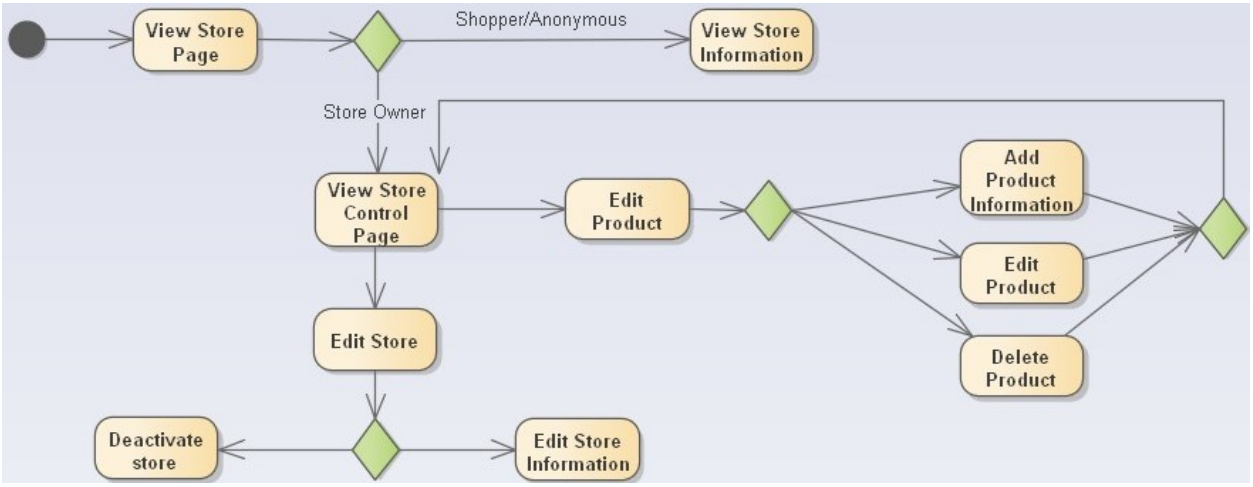


FIGURE 4 PRODUCTS LISTING AND VIEW ACTIVITY DIAGRAM.

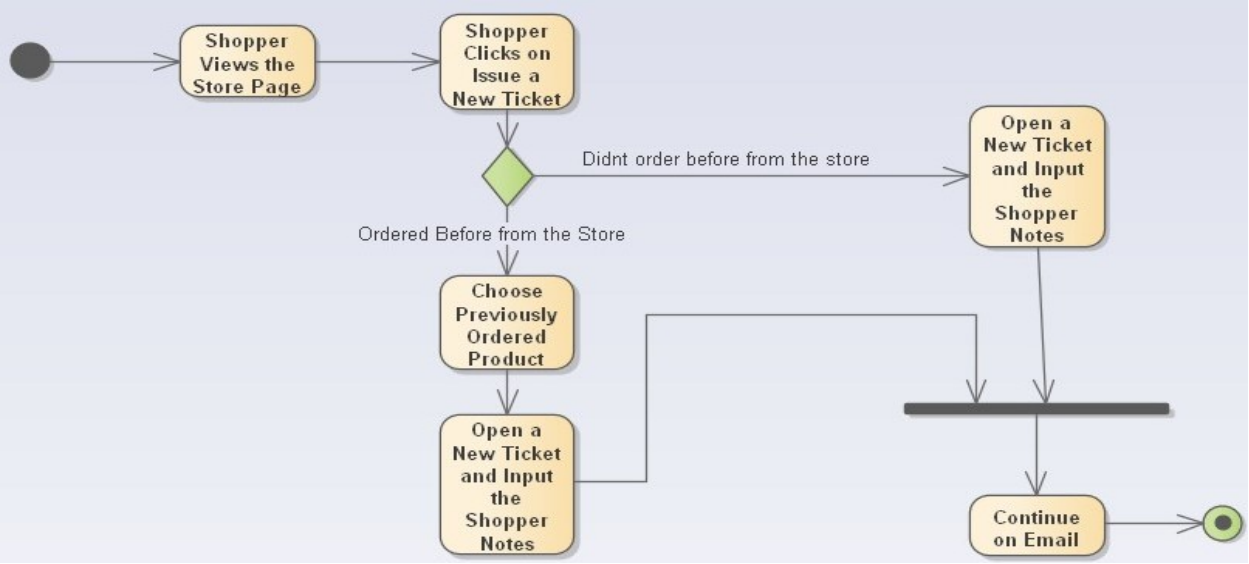


FIGURE 5 THE TICKET SYSTEM ACTIVITY DIAGRAM.

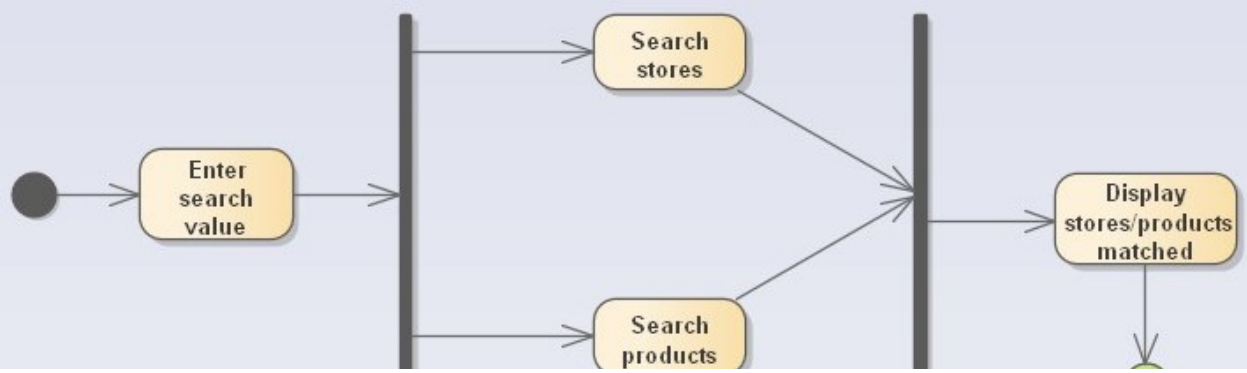


FIGURE 6 SEARCH STORES AND PRODUCT-WISE.

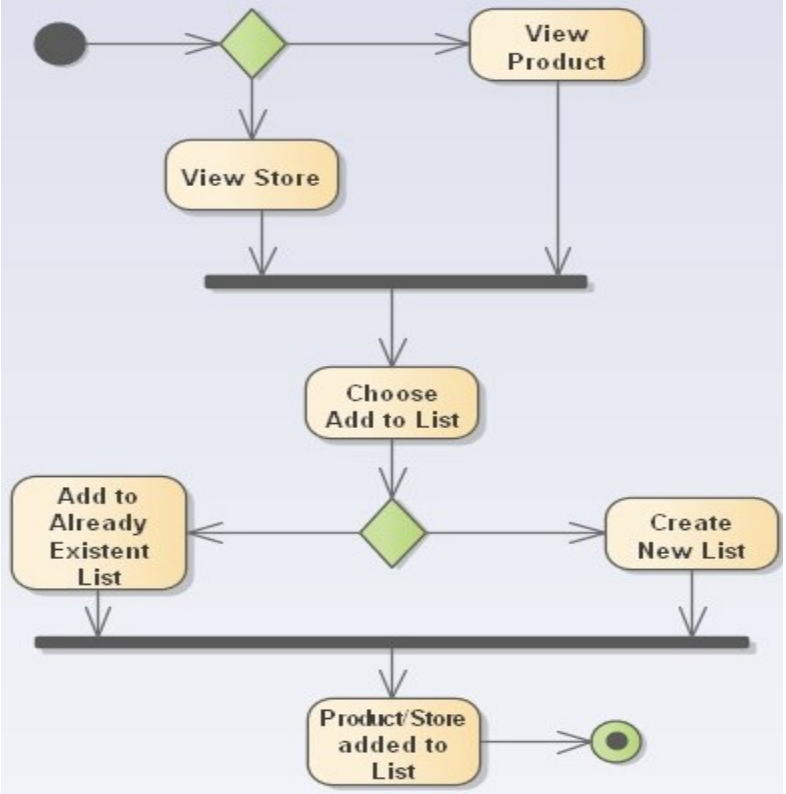


FIGURE 7 FAVORITE LIST IMPLEMENTATION ACTIVITY DIAGRAM.

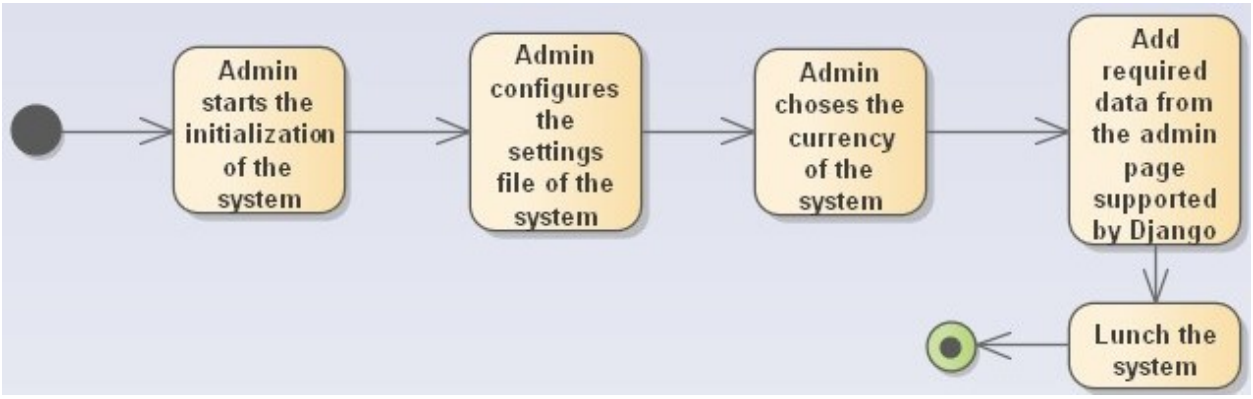


FIGURE 8 ADMIN CONFIGURATION ACTIVITY DIAGRAM.

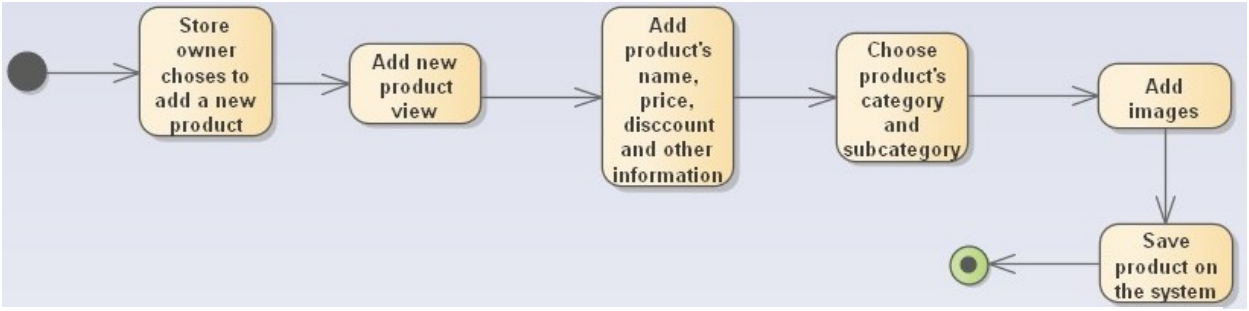


FIGURE 9 ADD PRODUCTS TO THE SYSTEM BY STORE-OWNER ACTIVITY DIAGRAM.

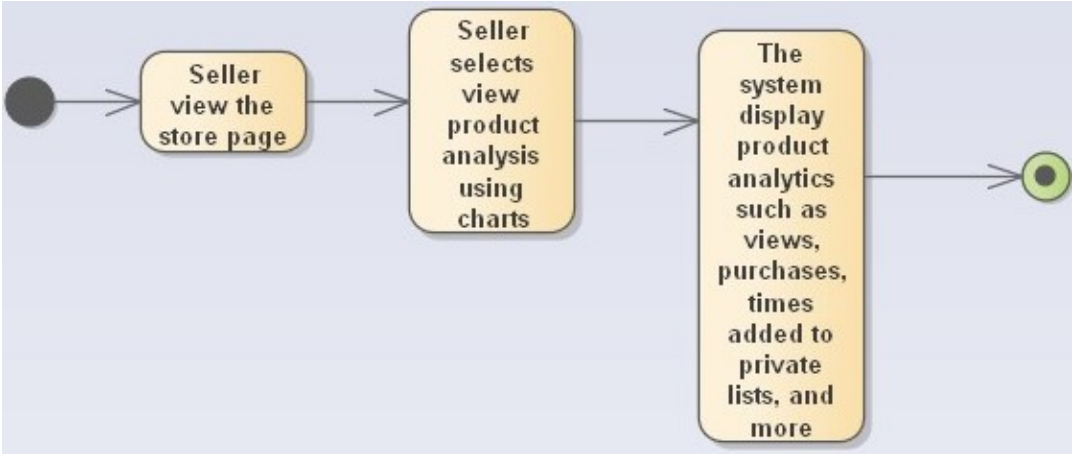


FIGURE 10 PRODUCT ANALYTICS ACTIVITY DIAGRAM

3. SYSTEM ARCHITECTURE

System architecture, the most solid part of the system was designed to suit the scalability needs. As the system was hosted on AWS to make use of the services the platform provides, the old system architecture was a bottleneck to the system scalability and the usage of scalability

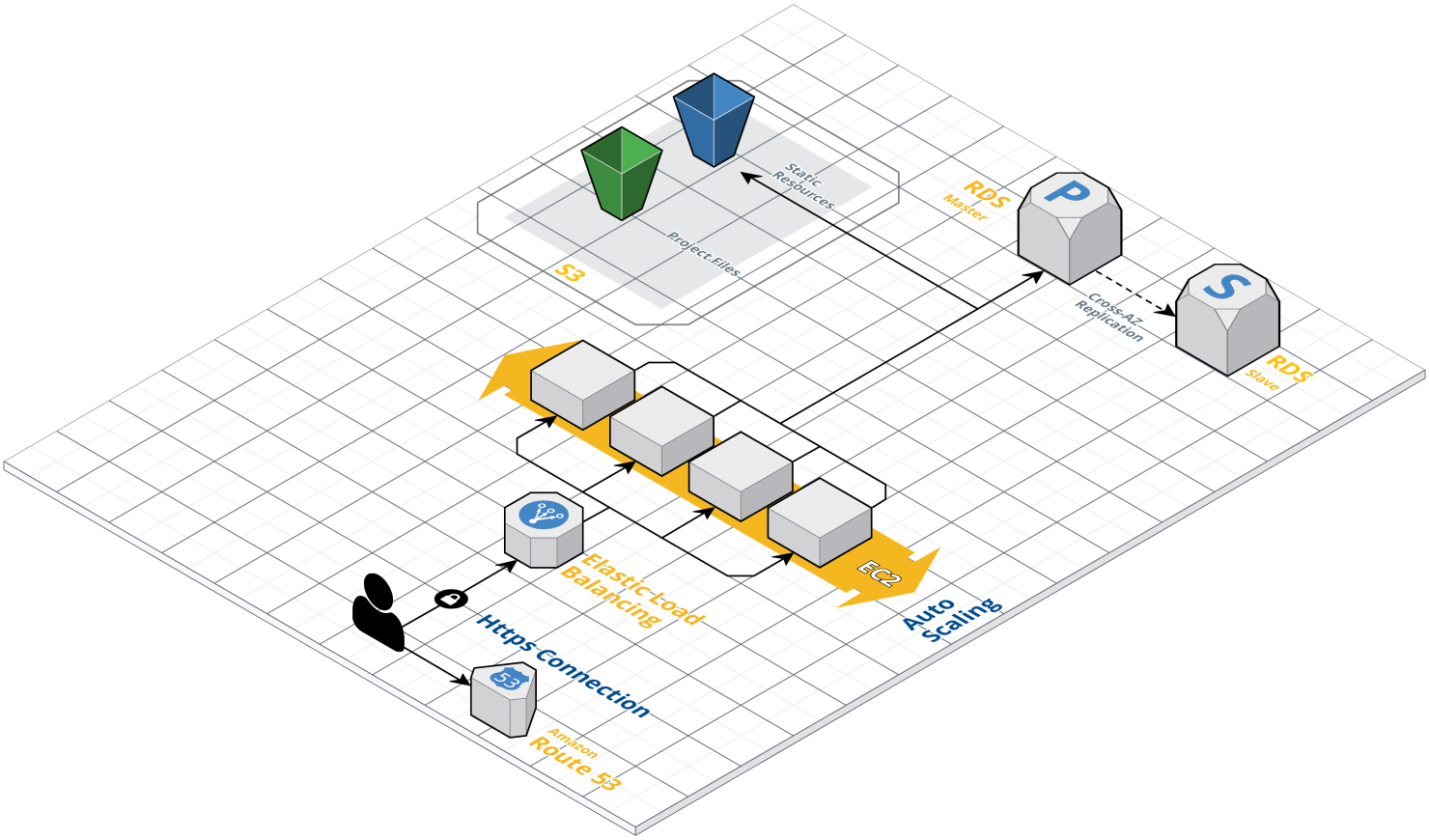


FIGURE 11 THE NEW SYSTEM ARCHITECTURE FROM A HIGH VIEW

features of AWS, thus the new system architecture is shown in Figure 1.

This system architecture makes use of the scalability services provided by AWS such as the load balancer, the RDS which is a relational database, and elastic computing scale the system horizontally and vertically.

AWS provides several services that we used in Datashop. Starting with Route 53 (Linkeit-Blog, 2019), it's a Domain Name System (DNS) service that gives the developers the ability to

register their domain name and insert DNS records allowing fast propagation inside the cloud infrastructure. The service was used to register the domain name for the project (https://datashops.io) by creating DNS type A record that points to the load balancer IP Address. A CNAME type record was also added for generating an SSL Certificate for securing the connection between the user and the load balancer.

The Elastic Load Balancer (ELB) is a service provided by AWS (Yadav, What is Amazon Elastic Load Balancer (ELB), 2019). It listens on specific ports allowing distributing the incoming traffic on those ports across a target group of Elastic Computing (EC2) instances. The result is balancing the computing and traffic load on those instances. This project is using ELB by setting the configuration to listen on 2 ports, one is for https (port 443), where each incoming request to https is forwarded to one instance in the target group of EC2 instances. The other port is for HTTP (port 80) which is used to redirect all the insecure connections to https mentioned earlier. The https listener is secured by an SSL certificate, the SSL certificate is generated using Amazon Certificate Manager (ACM) registered to our domain, making all requests to the load balancer secured and encrypted by default.

Elastic Computing (EC2) is a service that allows launching instances for computing purposes. For this project, scaling those instances is crucial (Yadav, Understanding Amazon EC2 Terminology, 2019). A target group was set for the instances. The ELB will then point to this target group. The target group is extended into an auto-scaling group, which is used to scale the target group in or out, as in adding new instances with a specific launch configuration, depending on a scaling policy.

The scaling policy in the current architecture is dependent on the average CPU usage of all the instances. If it exceeds or goes below a certain limit for a constant period, the scaling policy will be triggered adding or removing instances accordingly. The plan is to extend this scaling policy to use Machine Learning to make predictions based on historical logged data for the website users, launching new instances before the load occurs. The newly added instances are launched with a launch configuration. The launch configuration is used to specify the following details for the new instance:

- Amazon Machine Image (AMI)

- The instance type (Virtual CPU's, Instance Storage, Dynamic Memory, and network connection)
- Execution Script
- Permissions to AWS services.
- Security groups

In this case, the AMI (Rouse, Amazon Machine Image (AMI) , 2014) is set to be Amazon Linux distribution that comes with AWS CLI preinstalled and all other major libraries and binaries up to date. The instance type is set to a ‘micro' instance, that has 2 Virtual CPUs, 1 GB of dynamic memory and up to 5 gigabits of network connection. The execution script is executed whenever the instance is launched, it is used to pull the project data from the S3 project bucket (marked as green in *figure 1*), build the project and start serving incoming requests.

Simple Storage Service (S3) is a service that provides unlimited scalable storage (Rouse, Amazon S3 , 2018). S3 provides the concept of buckets, which are logical units of storage as volumes, used to store files or objects. Two buckets were created in this project, one is to serve static files and assets for the Django application and the uploaded media by the users and the other is to host the project data files to be pulled by the EC2 instances in the execution script as mentioned above.

Relational Database Service (RDS) (SumoLogic-Team, 2019) is used to host the database system. The selected database engine is PostgreSQL (postgreSQLTutorial, 2019). RDS provides the means to create replicas of the database in multiple availability zones, to scale and to avoid downtime. The database instance has 2 Virtual CPU's in it with 1 GB of RAM. The storage for the database is set to 100 GB, expandable to 1 TB.

The internal architecture of the instances has also changed to adhere to the new external architecture. MongoDB is now no longer used, as will be discussed in the design decisions section. The user no longer can access the instances directly, as the load balancer now is exposed to the web, while the internal instances are protected. Gunicorn (vsupalov, 2019) WSGI server is now forking multiple worker processes to handle the work concurrently. Each worker process has a running instance of the Django web application. *Figure 2* shows the current architecture inside the instances.

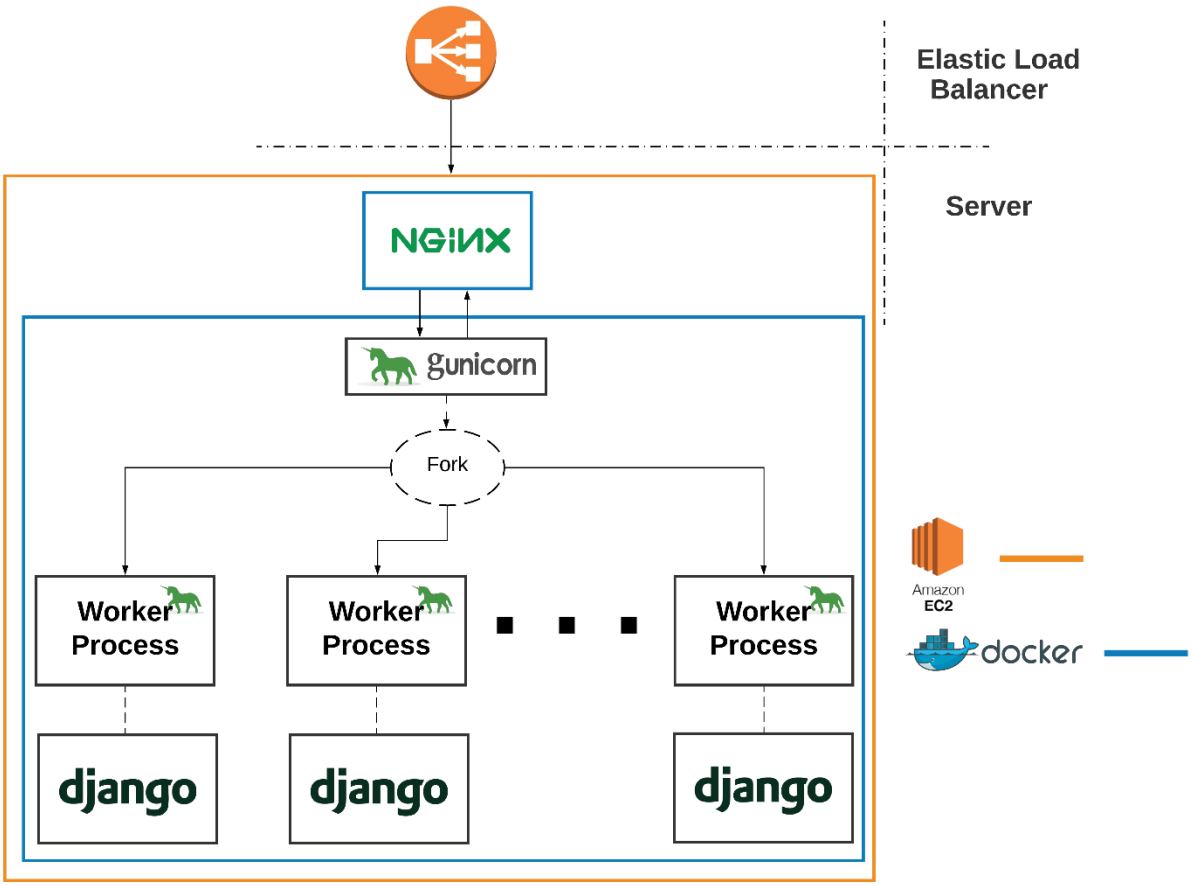


FIGURE 12 NEW INTERNAL ARCHITECTURE INSIDE EC2 INSTANCE

Overall, all these services are communicating with each other to serve requests, allowing lower response times with separation of concerns in mind. Such architecture can scale up and down in terms of computing power, database, and storage serving hundreds of requests per second with no performance issues.

On the other hand, Figure 17 shows how the inner components of the system are interacting and communicating together. The system discussed is mainly built on top of Django and uses applications to achieve the maximum modularity. The system consists of four apps in addition to the user app which classifies users into three categories. Each app is responsible for a certain set of data and its corresponding functionality. The product app is considered with the actual data of the product in addition to all the operations done on products like listing, adding, deleting, and so on. The cart product is associated with the cart manipulation and it handles all the operations done

on the cart for the shoppers. The store app is used to create, list, and modify the stores associated with the store owner's accounts. These five apps are all using data that can be retrieved from and saved to the sql3lite database which uses the model's manager interface provided by Django.

As mentioned at the introduction of this section, the system has five main apps that each is responsible for a set of data with the corresponding functionalities to manipulate the data.

4. SYSTEM DESIGN ARTIFACTS

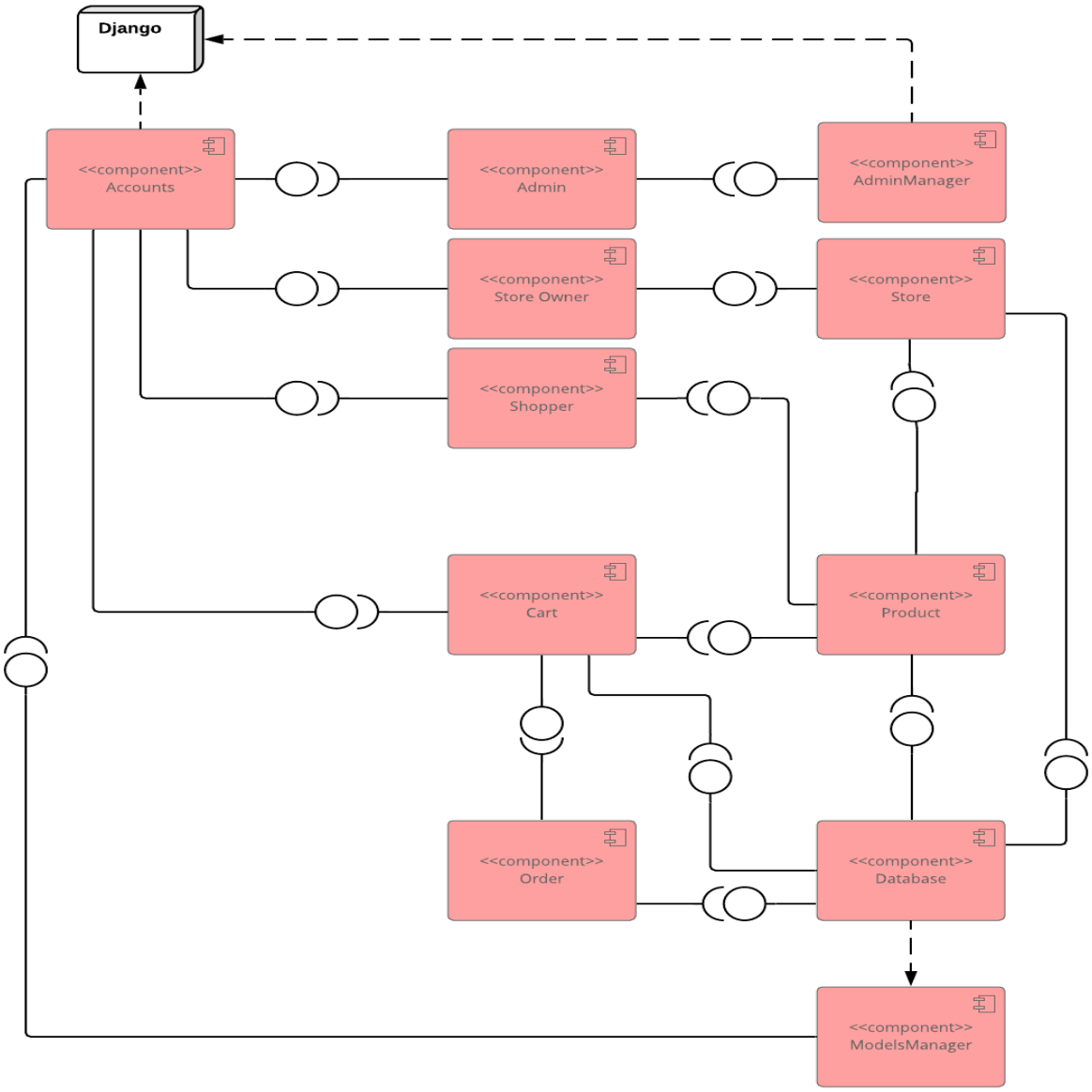


FIGURE 13 THE SYSTEM COMPONENT DESIGN.

This section discusses the system design regarding algorithms used to implement the required features and the design decisions with their effects on the total system design.

4.1. DESIGN DECISIONS

Figure 19 shows the models' relations in the whole system. Five big boxes are considered as applications, and each application has several models. The applications considered here are product, cart, store, order, and users. Each one of these contains one or more models and each model contains the related attributes to this model.

In this section, the intra-relations will be briefly explained followed by the inter-relations in the whole system.

i. Intra-relations:

Product:

- A category can contain many subcategories.
- A subcategory can contain many products.
- A product can contain many images.

Cart:

- A cart can be attached to many entries.

Store:

- A currency can be used in many stores.

Users:

- A country can contain many addresses.
- A custom user can have many addresses.
- A custom user can connect exactly one social media link.

ii. Inter-relations:

Product-Cart:

- A product can be added to many entries.

Product-Store:

- Many products can be added to one store.

Cart-Order:

- A payment method can be used in many orders.
- An order can be added to many entries

Cart-User:

- One cart is assigned to one user.

Store-User:

- A custom user can view many stores.
- A currency can be used by many countries

Order-User:

- Many orders can be added to one address.
- A custom user can do many orders.

Further design decisions include many aspects of the system and may change how the system behaves or acts in certain situations. To better classify design decisions, they will be based application-wise starting with users.

In the users' application, a user can convert the type from seller to shopper and vice-versa, enabling any shopper to have multiple stores, and any seller to have a cart and order product. The conversion is done using a function in the website page and related functions are provided depending on the current user type. This ability of conversion eliminated the need for an extra user model to model the two distinct types of users, and a flag is used to distinguish user type and provide corresponding functionality.

The cart implementation is based on entries that link each product with the user who put it in a cart and the cart of the user. These entries are used to calculate the cart information and link

each user with the preferred products. Although this design approach provides flexibility and efficiency in dealing with the carts, it provides overhead on the database storage where a lot of rows will be inserted into a database for each cart in the system. The cart may contain multiple products from different stores, and so when checking out, the cart will create multiple orders for each store to maintain the state of each separately without mixing the concerns in one order.

The order application uses the same technique the cart application uses to link products with users, but this time, an order is permanent in the system, thus order entries will generate huge data set over time and may be hard to manage, a fix for this would be a crown job that can clean up the database for very old entries and back them up in a secondary storage.

The order state should be always valid, so a Deterministic Finite Automaton (DFA) was constructed to manage the order state (*Figure 18*)

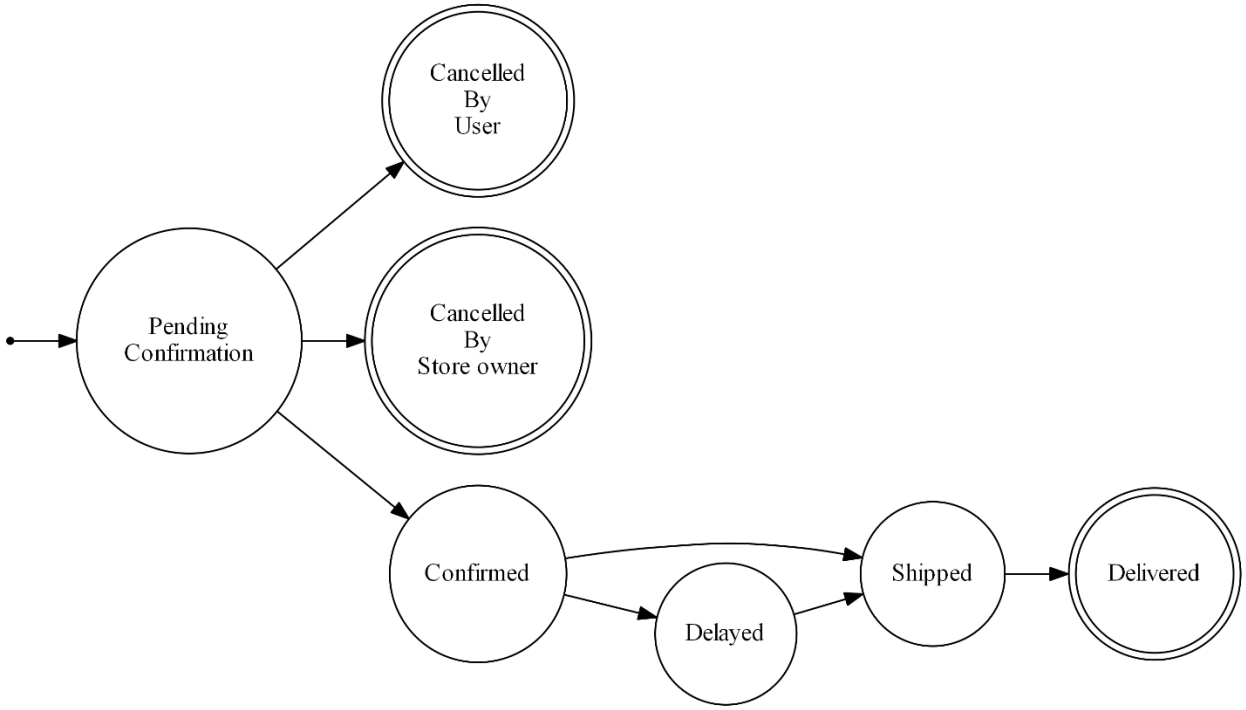


FIGURE 14 DFA FOR THE ORDER STATE

Each time the order state is changed, the validators will run an algorithm that simulates the DFA, to guarantee that the state transition from the old state to the new state is legal and can be done. If it's not a legal transition, an error message will be displayed to the user.

The search application takes a query string, which is then tokenized into a set of strings as keywords. Each keyword will be matched against all the product and store names, descriptions and categories. Such a design allows the user to look up the system easily.

The store application is now based on users, which offers the ability for a user to have multiple stores associated with the users' model. This design approach enables sellers to have multiple stores that each has a distinct list of products associated with it. This approach provides successful and is already manageable.

Finally, the product application uses the database as a main serving entity where all operation is done on products can be visualized as query retrieve, query update, and query save. The products' app also uses two distinct tables to link each product with its associated images to provide more flexibility for sellers to add multiple images to their products while adds overhead to retrieve the images of a certain product from the database.

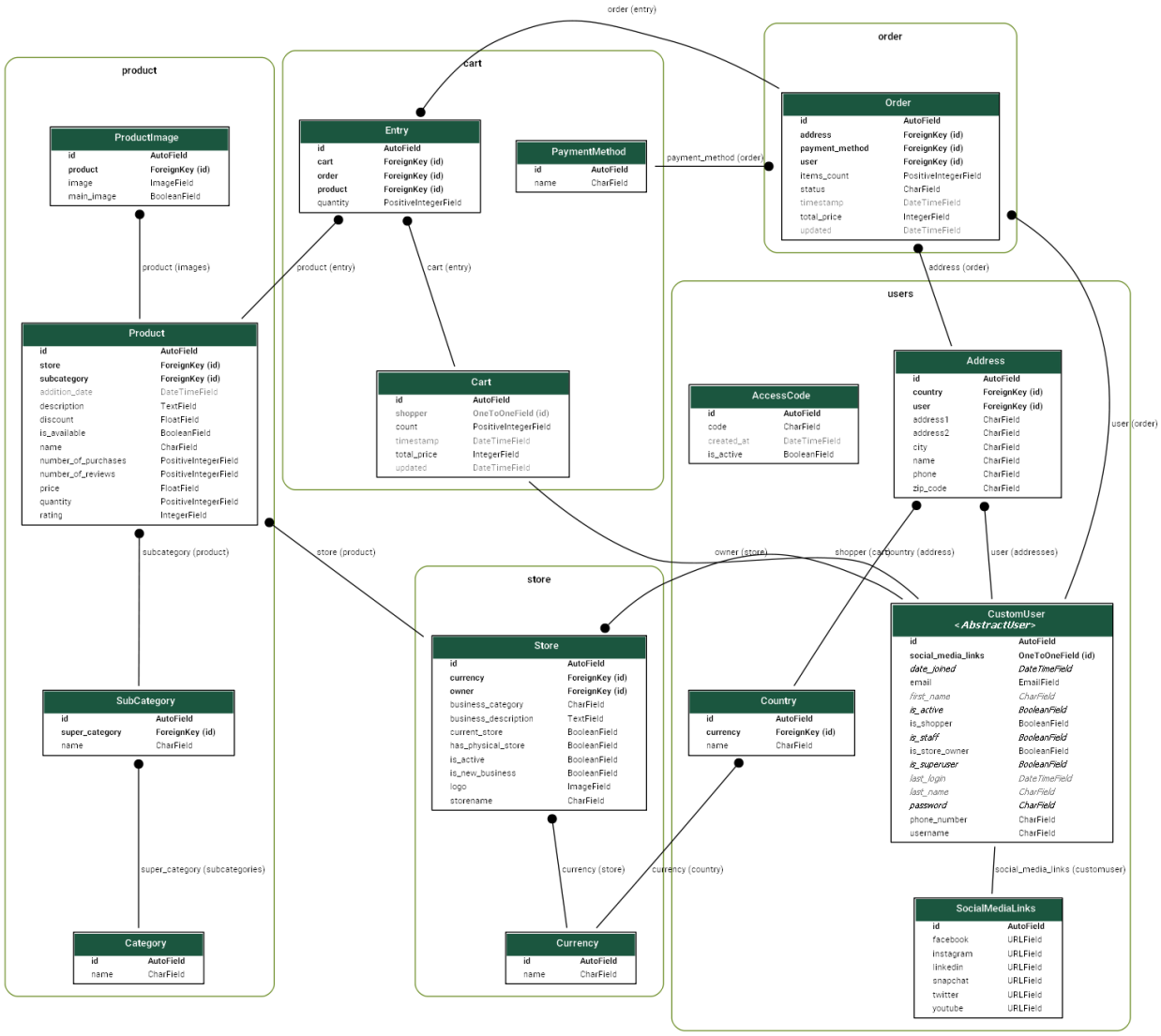


FIGURE 15 DATASHOP MODELS' RELATIONS

4.2. TECHNICAL DESCRIPTIONS

Some technical issues related to the system will be answered and the resolution will be mentioned in this section.

4.2.1 TECHNICAL ISSUES

Technical Issue	**Resolution**
How to handle DDOS attacks	The system prohibits users from being able to access a view a certain number of times during a time frame (1 minute for example)

Technical Issue	**Resolution**
How to assure that files uploaded by users are not harmed and won't get executed in the system	By limiting the files uploaded types to be only known types of images and nothing more

Technical Issue	**Resolution**
How to efficiently log users' behavior	By logging the request users send to the system, this way system can know exactly what they want and where are they at the current moment.

Technical Issue	**Resolution**
How to keep users' information safe and private	By encrypting users' passwords and databases' tables

Technical Issue	**Resolution**
How to ensure that the added products to the system don't contain inappropriate content	By using an API that will be implemented in the last sprint to check whether the uploaded images of products and stores are appropriate or not.

4.2.2 SEARCH ALGORITHM

Search functionality of the system was not made to be like a simple query. This includes the speed of the search, the type and quality of the results, and how the system dealt with the search queries.

The system's search is divided into two main categories, products search, and stores search. This division ensures that the results are more specific to the user query and that a user can be focused on what is he/she are searching for. When a user types a query in the search bar found on the website navigation bar, the template sends the query to the search view that first validate the query to not be empty, then the query gets into further modification.

Using PostgreSQL database engine, the database offers the tools and the function to enable full text search among the records inside the table. This functionality is achieved thanks to Vector Space Model (VSM). Vector space model is an algebraic model that transforms text into vectors, based on the lexemes generated in the tokenization process. The lexemes are based on a specific language, DataShop uses English lexemes. After generating the vectors, they get indexed using a special index named Generalized Inverted Index (GIN).

Combining VSM with GIN enables quick search, this is done by using the indexed vectors that were generated from the target table fields. The target fields for store search are business descriptions, category and store name, while the target fields for product search are both product name and description.

The search query from the user gets divided into keywords, and a VSM is created for each. Then using GIN, the query VSMs will be matched against the existing data. Each result will be annotated with a rank, the higher the rank, the more likely that the results are accurate against the search keywords. This algorithm enables the user to quickly search for specific products or stores, then getting the results sorted by descending rank.

4.3.USER INTERFACE

The user interface is flexible and can work on both mobile and desktop using responsive CSS directives. In this section, each desktop variant of the UI will be compared to mobile UI.

4.3.1 HOME

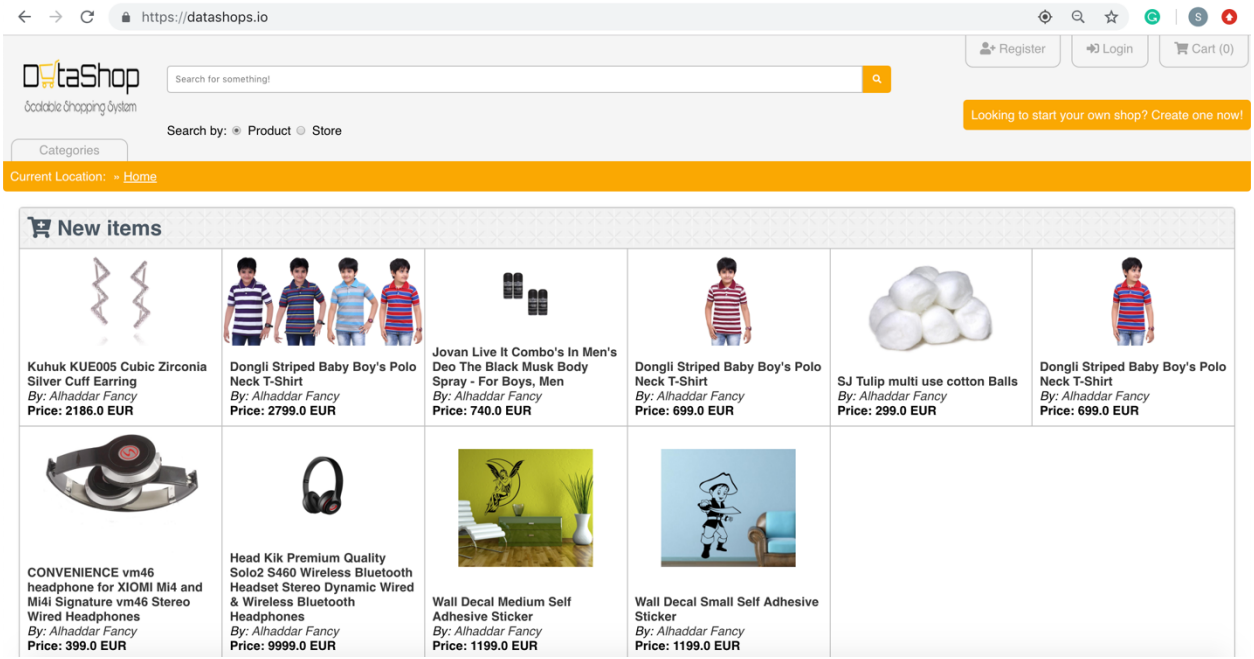


FIGURE 16 HOME PAGE

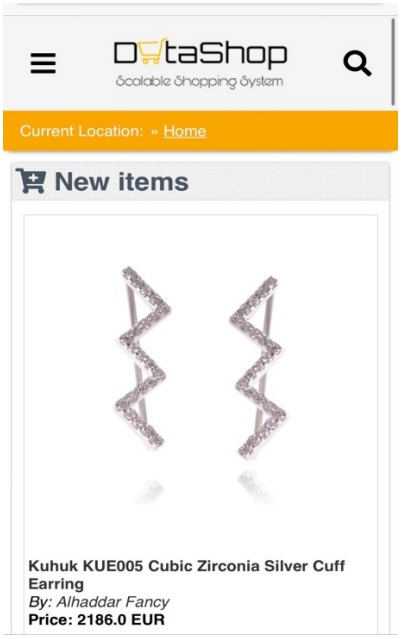


FIGURE 17 MOBILE HOME PAGE

4.3.2 LOGIN

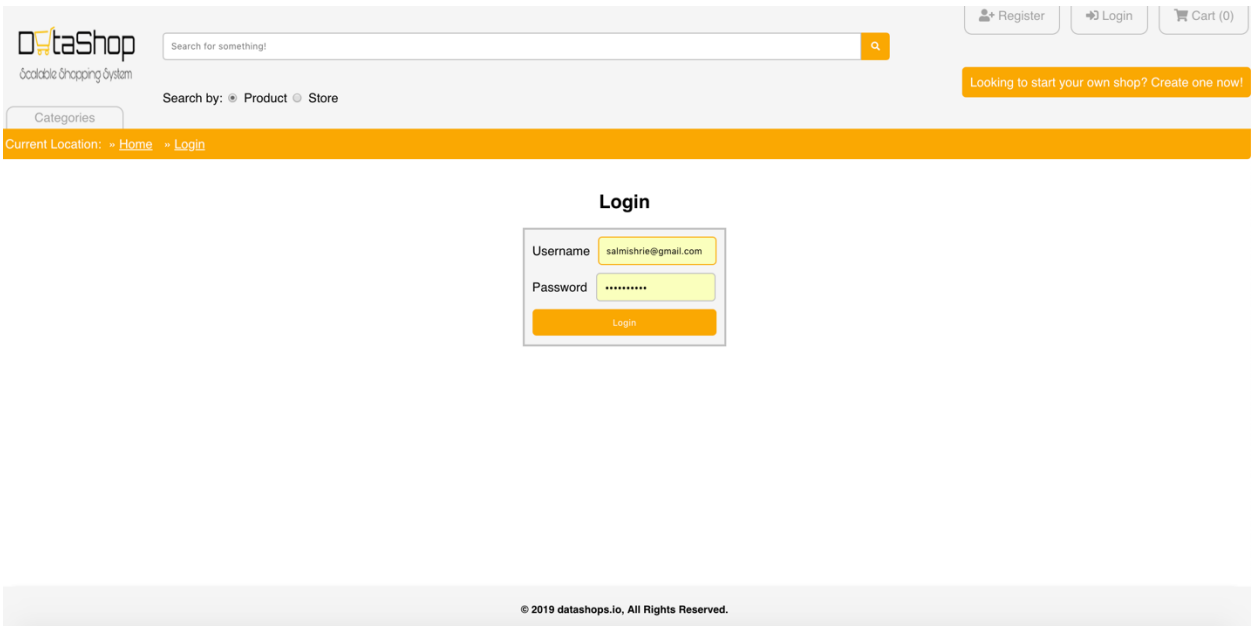


FIGURE 18 LOGIN

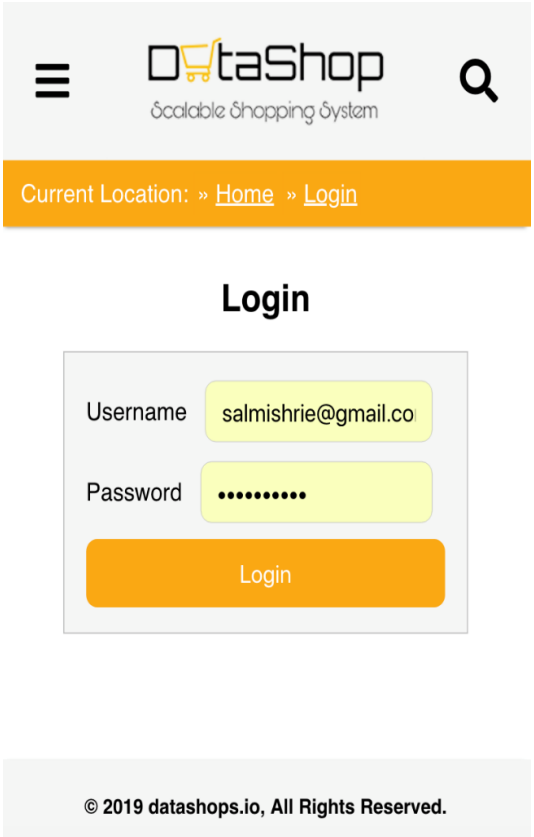


FIGURE 19 MOBILE LOGIN

4.3.3 REGISTERATION

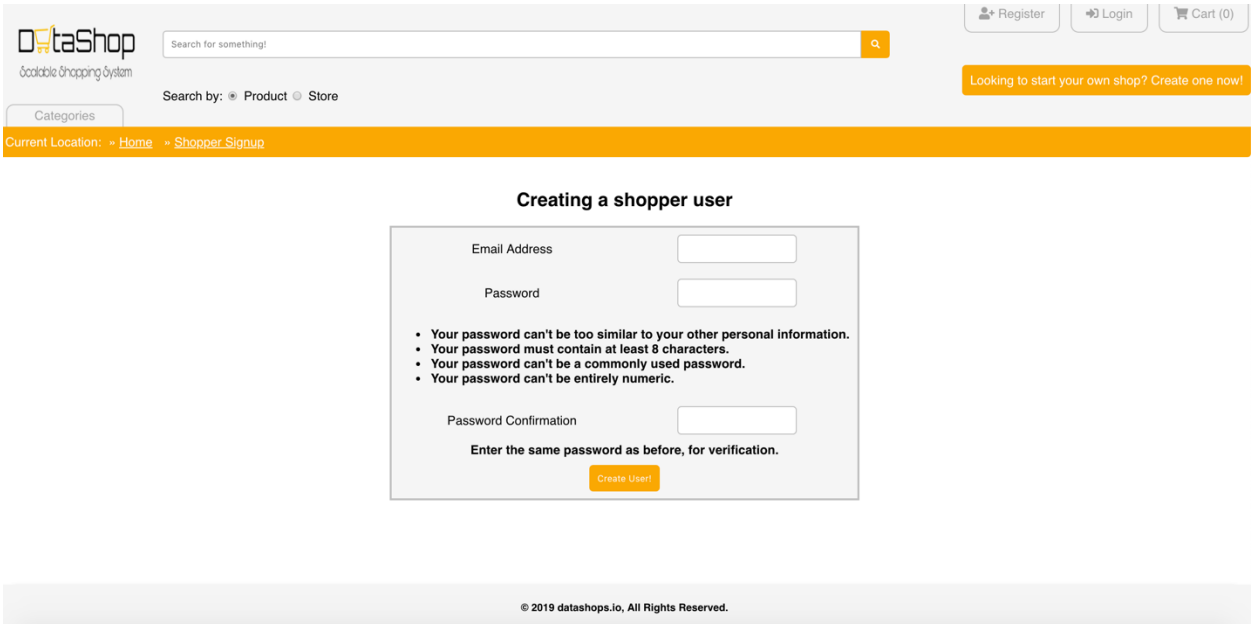


FIGURE 20 CREATE ACCOUNT

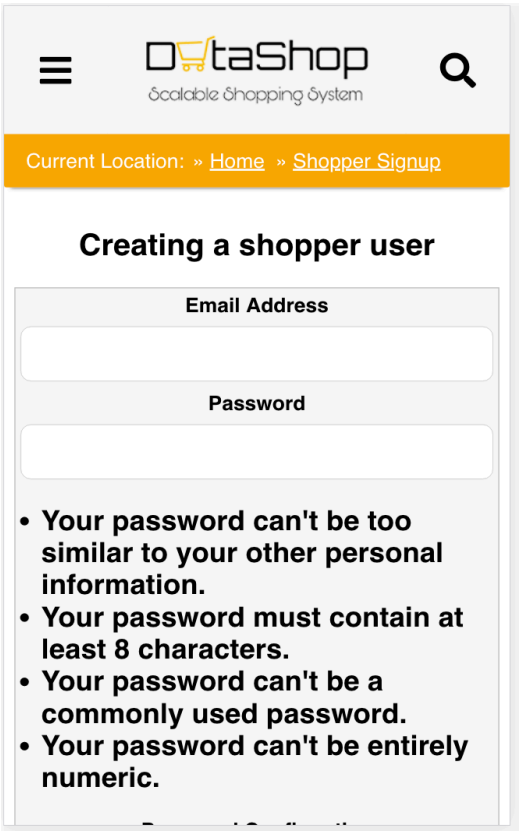


FIGURE 21 MOBILE CREATE ACCOUNT

4.3.4 CART

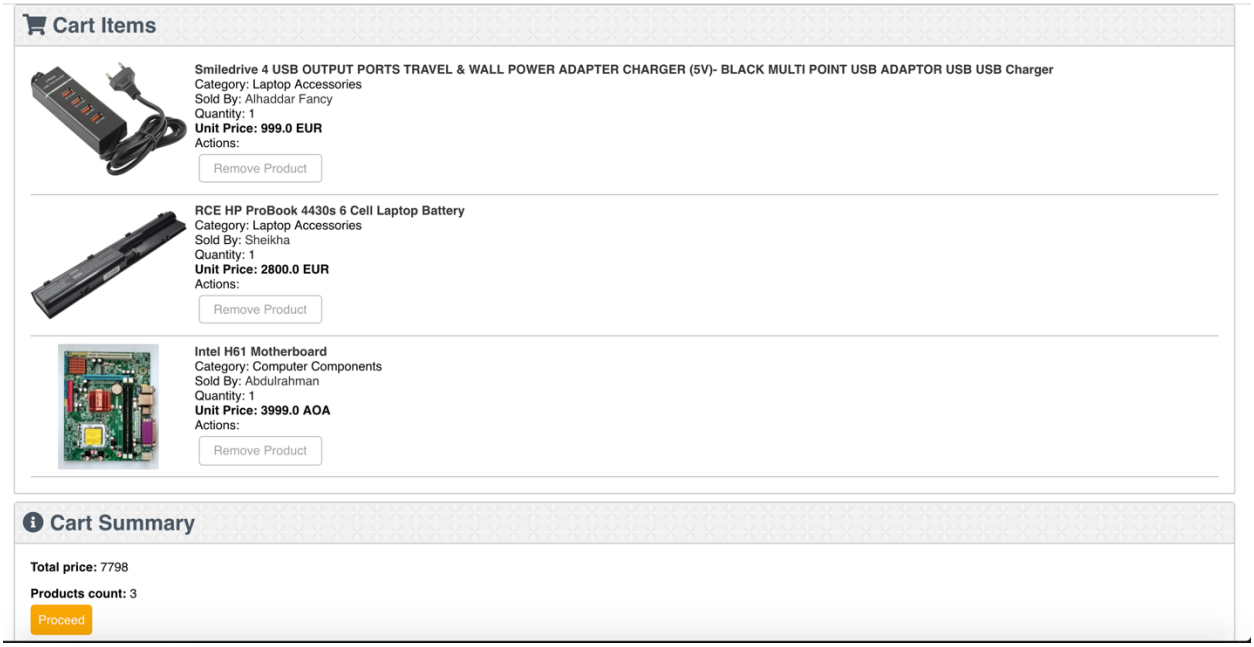


FIGURE 22 CART

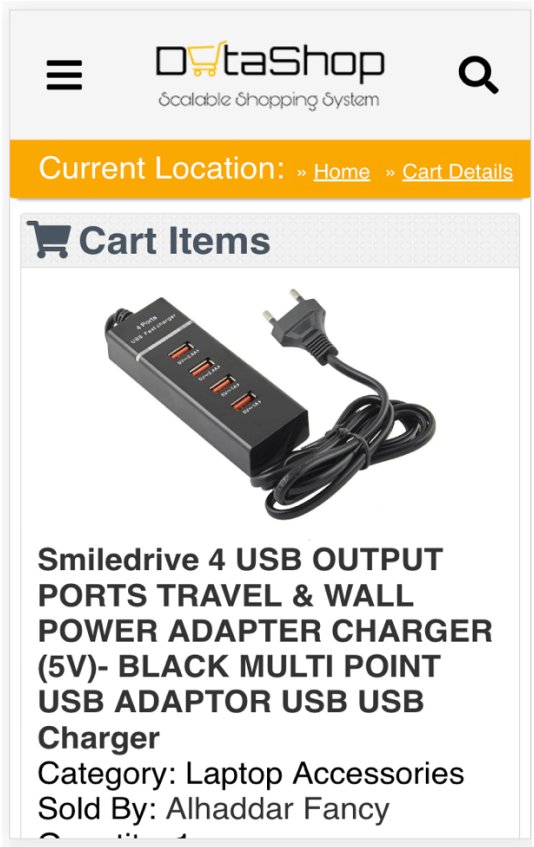


FIGURE 23 MOBILE CART

4.3.5 CHECKOUT

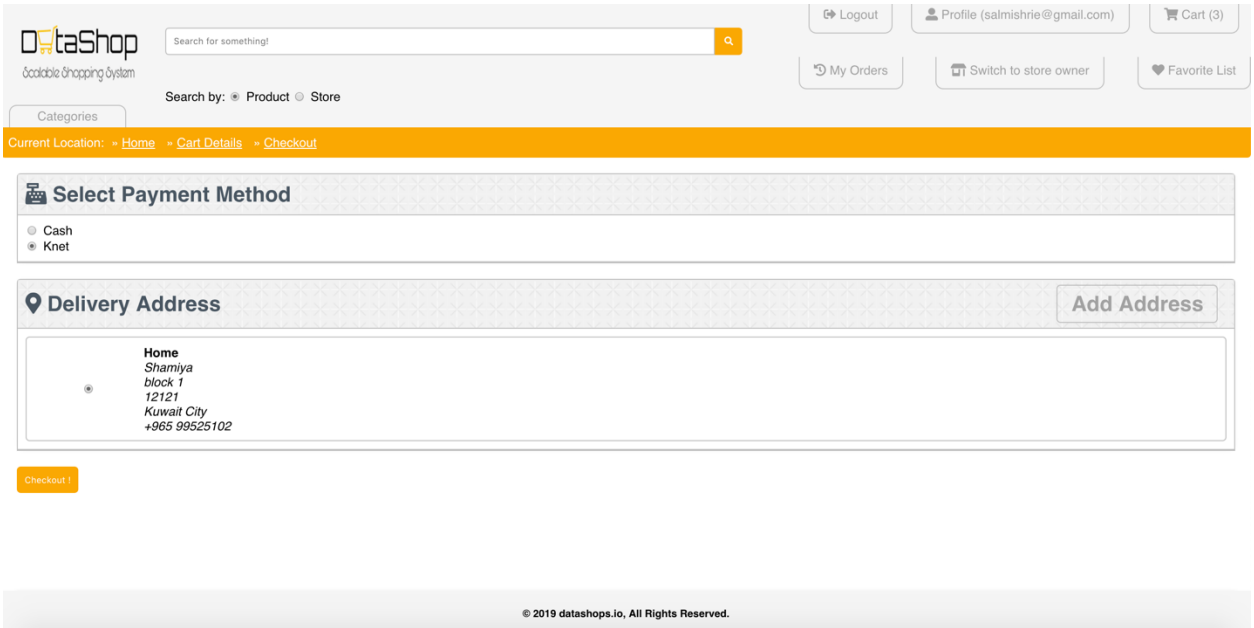


FIGURE 24 CHECKOUT PROCESS

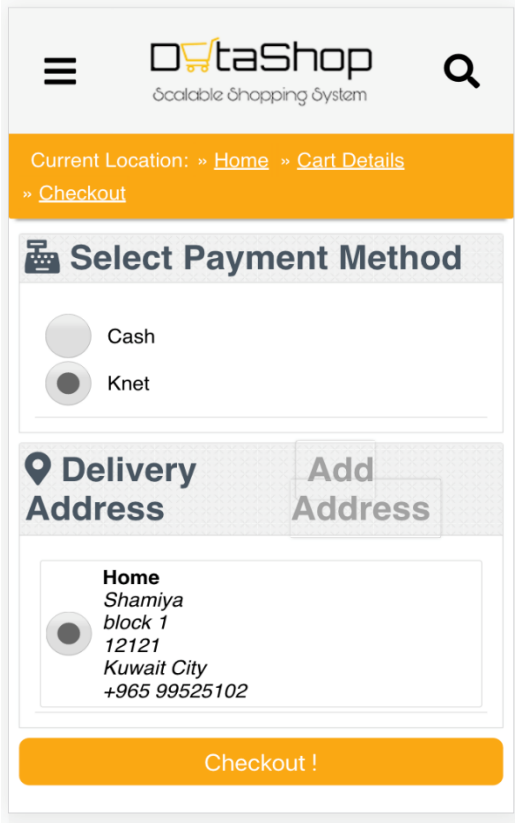


FIGURE 25 MOBILE CHECKOUT

4.3.6 PRODUCT DETAILS

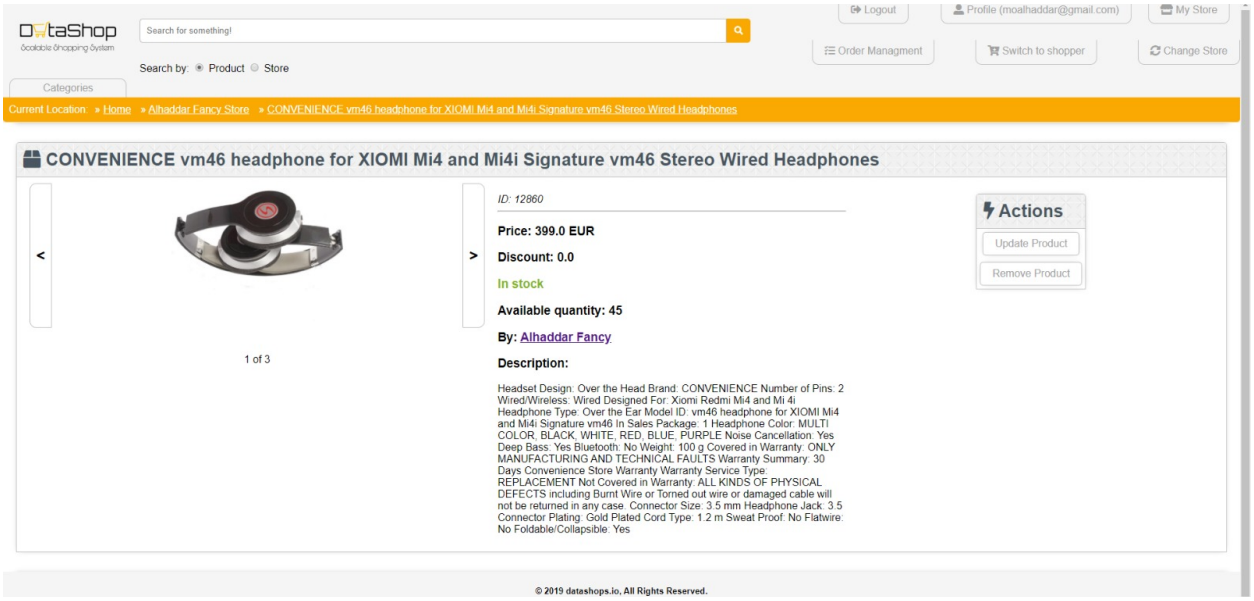


FIGURE 26 PRODUCT DESCRIPTION

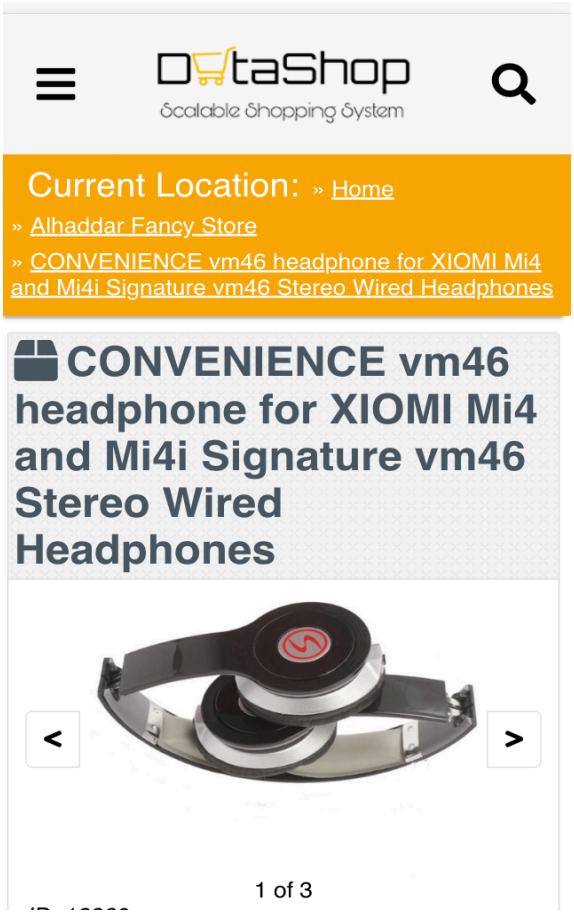


FIGURE 27 MOBILE PRODUCT DESCRIPTION

4.3.7 STORE

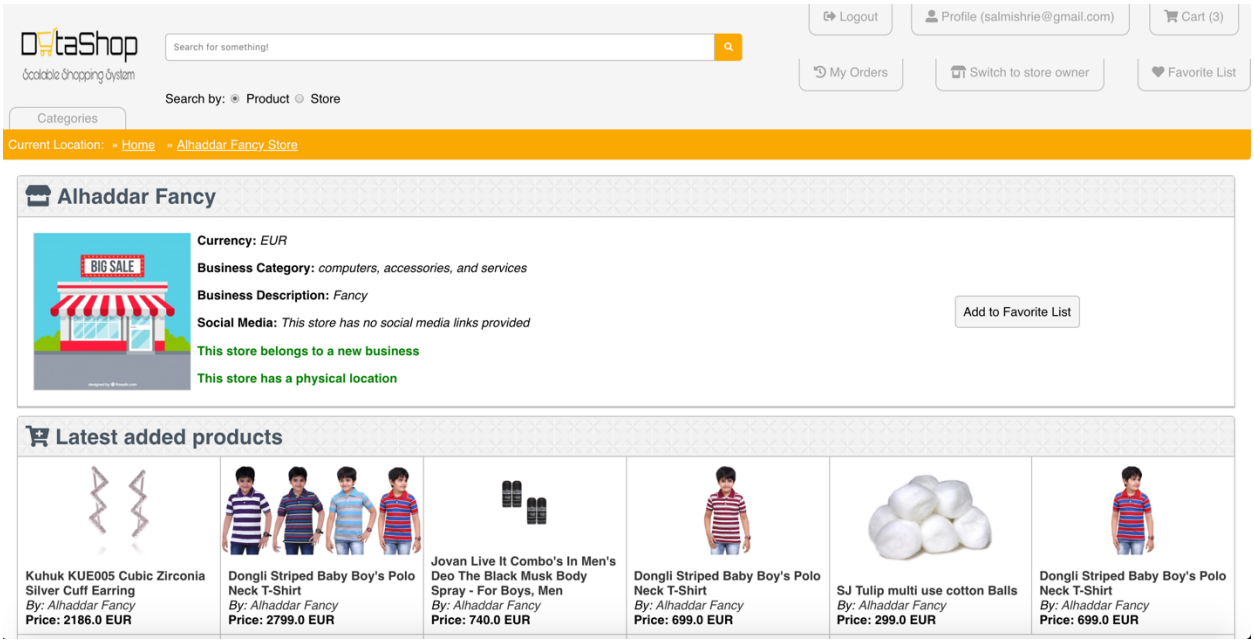


FIGURE 28 STORE DETAILS

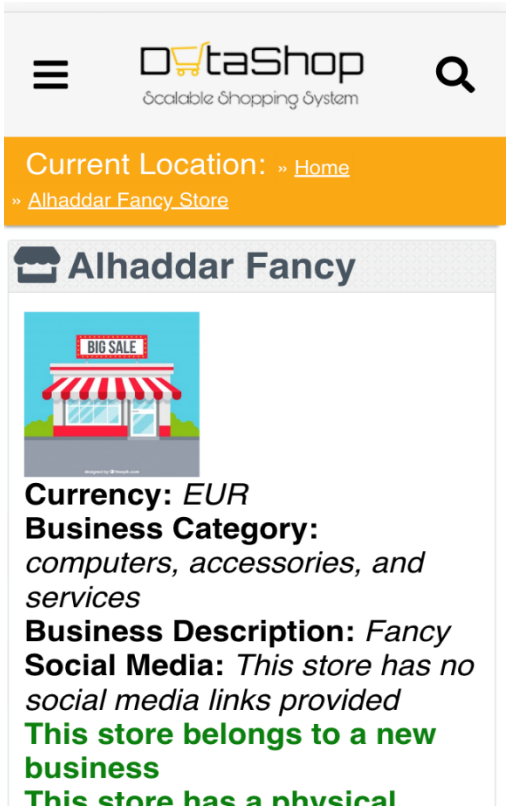


FIGURE 29 MOBILE STORE DETAILS

4.3.8 SEARCH

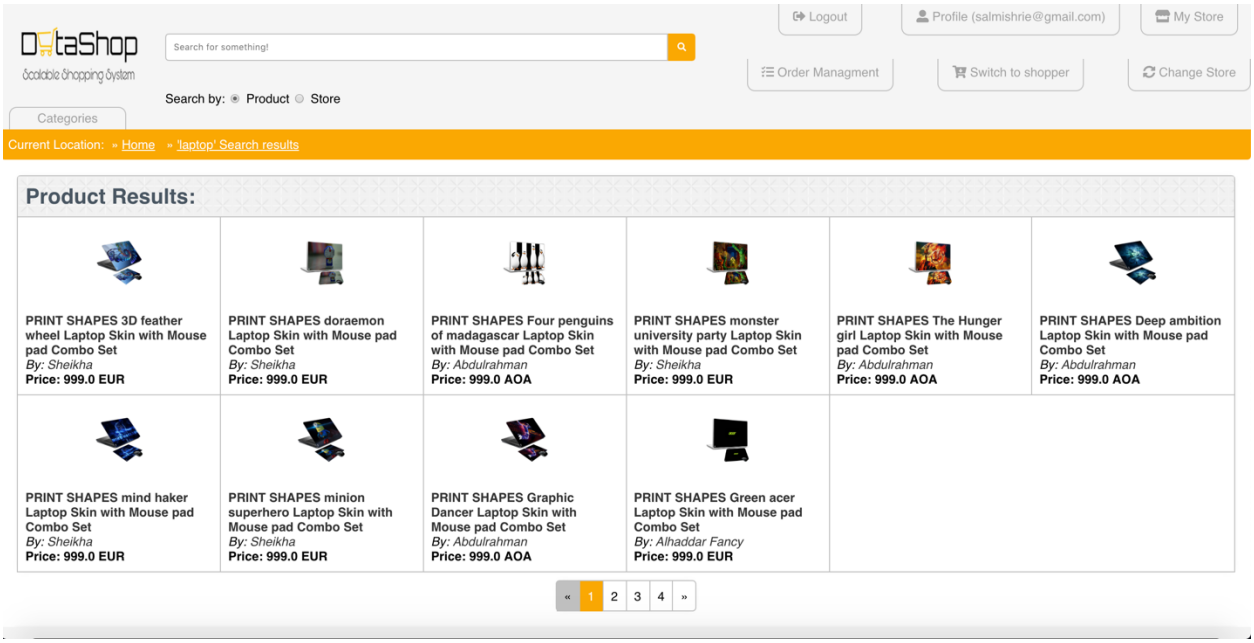


FIGURE 30 SEARCH

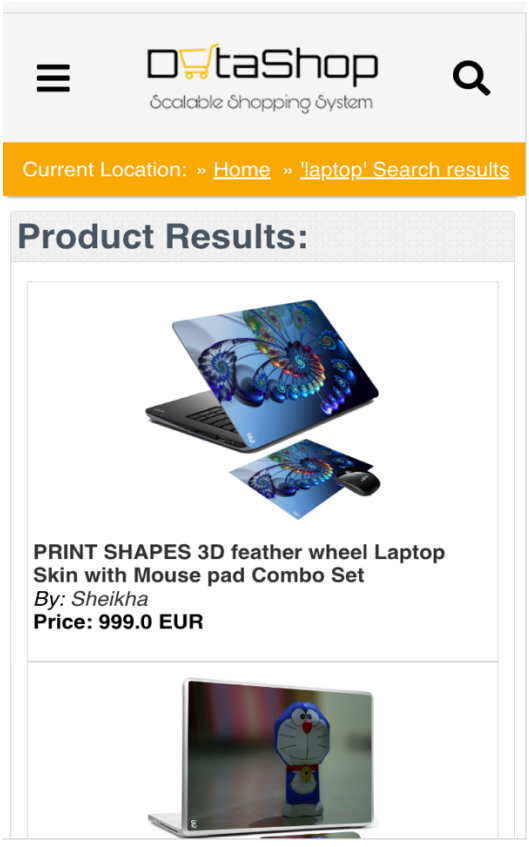


FIGURE 31 MOBILE SEARCH

4.3.9 CATEGORIES MENU

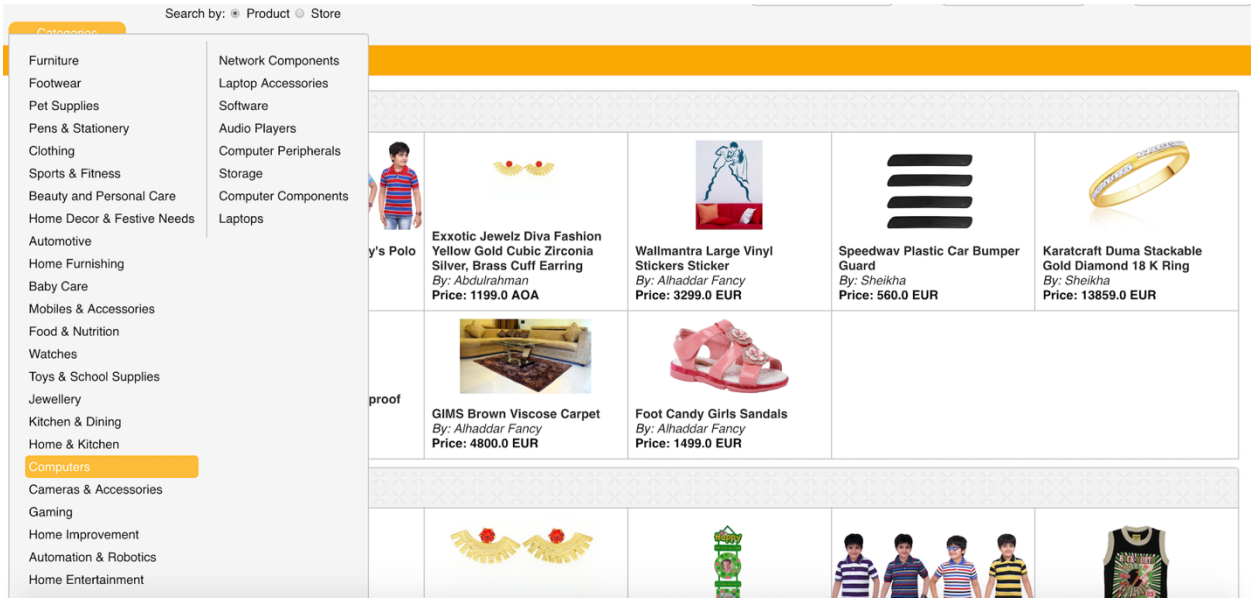


FIGURE 32 CATEGORIES MENU

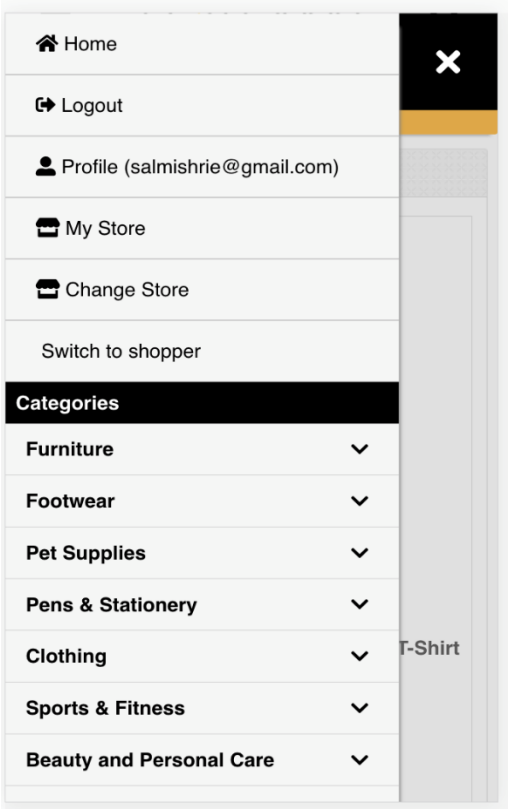


FIGURE 33 MOBILE CATEGORIES MENU

4.3.10 PROFILE

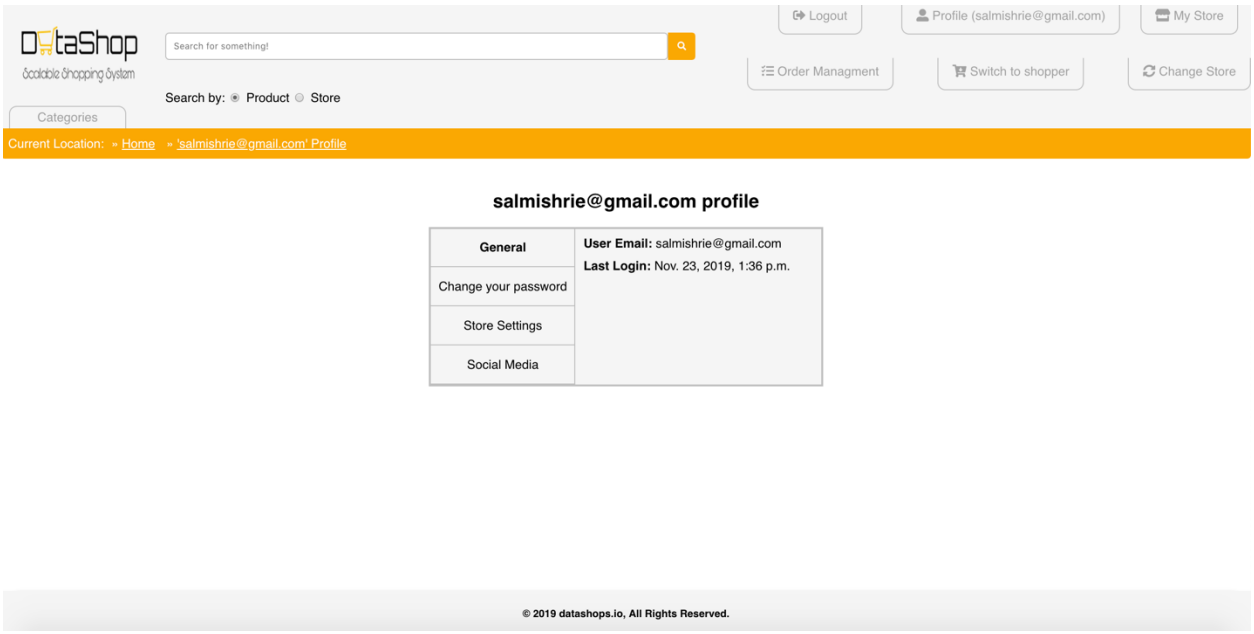


FIGURE 34 PROFILE PAGE

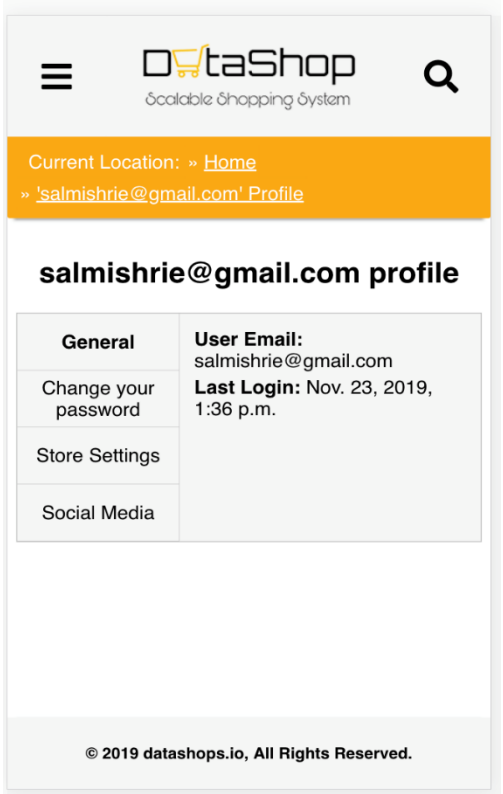


FIGURE 35 MOBILE PROFILE PAGE

4.3.11 ORDER HISTORY

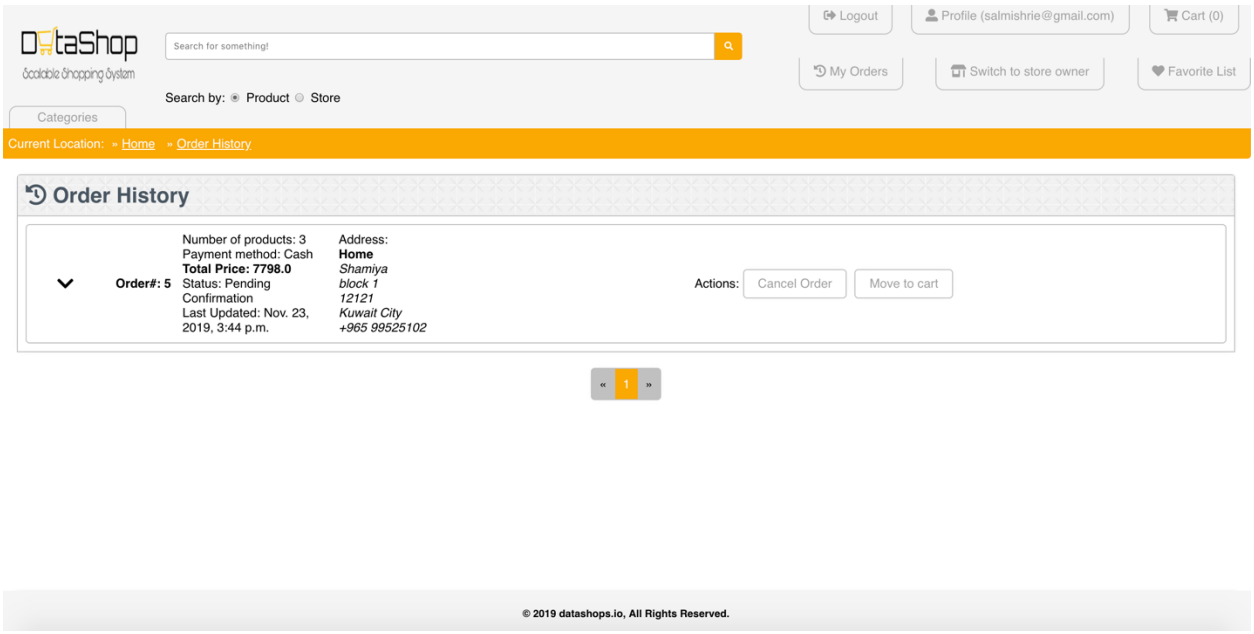


FIGURE 36 ORDER HISTORY

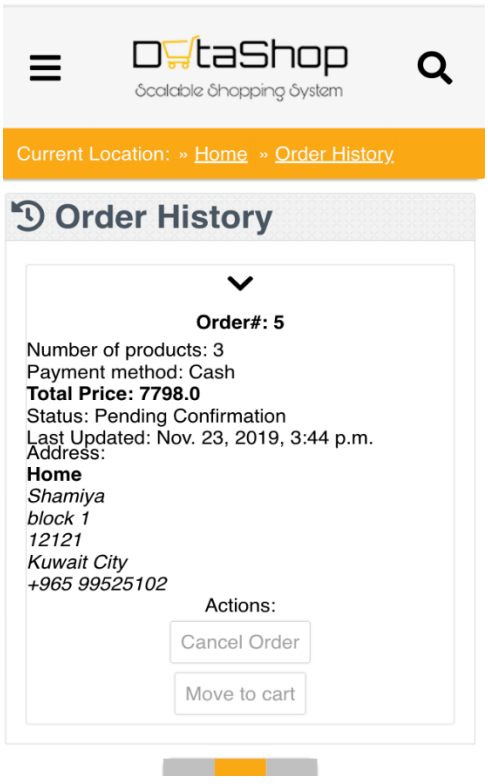


FIGURE 37 MOBILE ORDER HISTORY

4.3.12 PRODUCT ADD

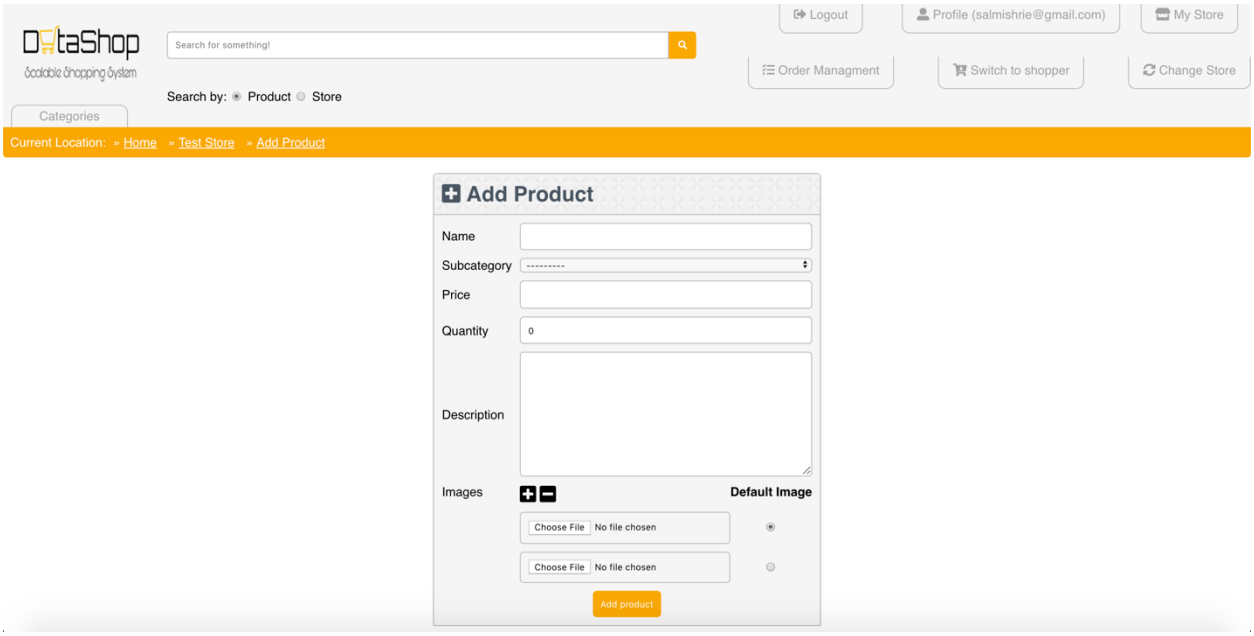


FIGURE 38 ADD PRODUCT

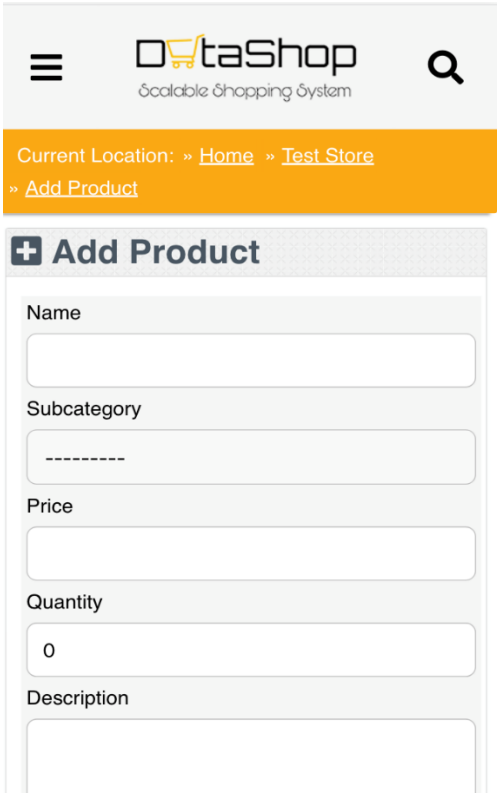


FIGURE 39 MOBILE ADD PRODUCT

4.3.13 STORE CREATION

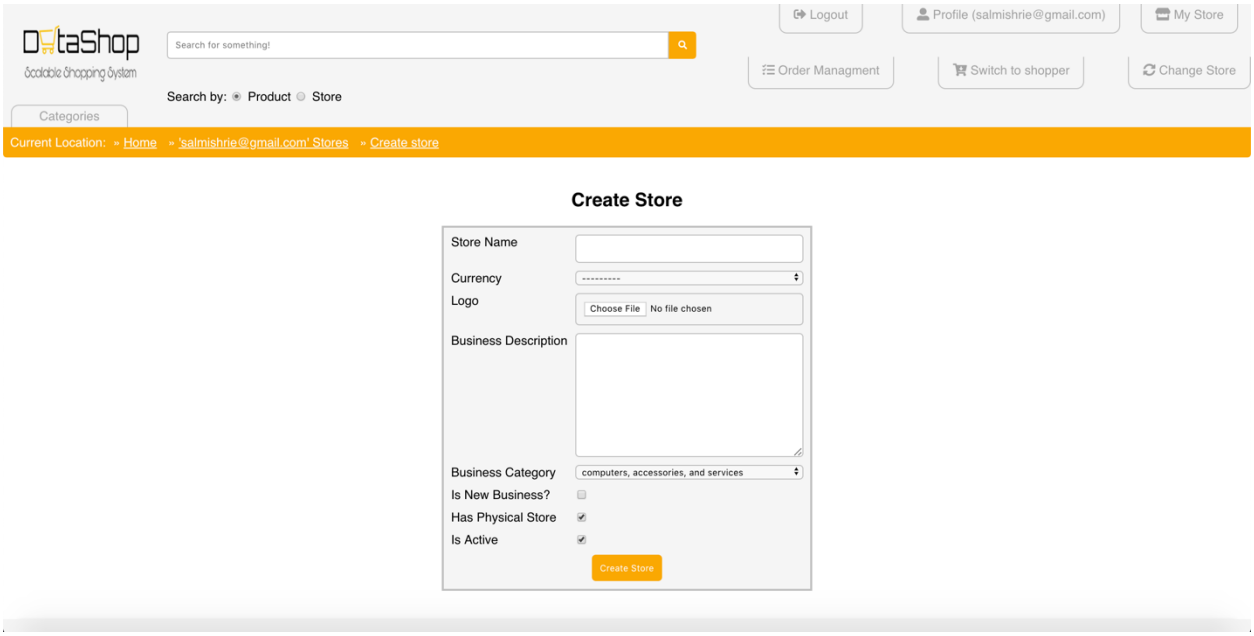


FIGURE 40 STORE CREATION

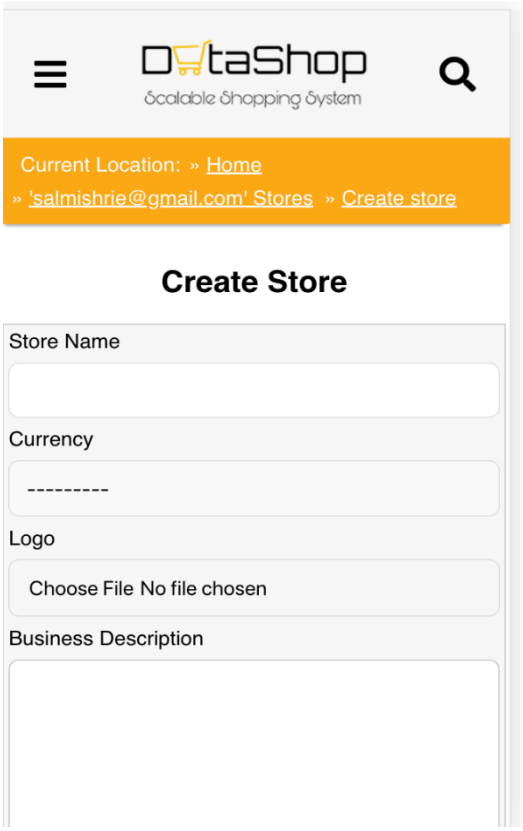


FIGURE 41 MOBILE STORE CREATION

4.3.14 CHANGE STORE

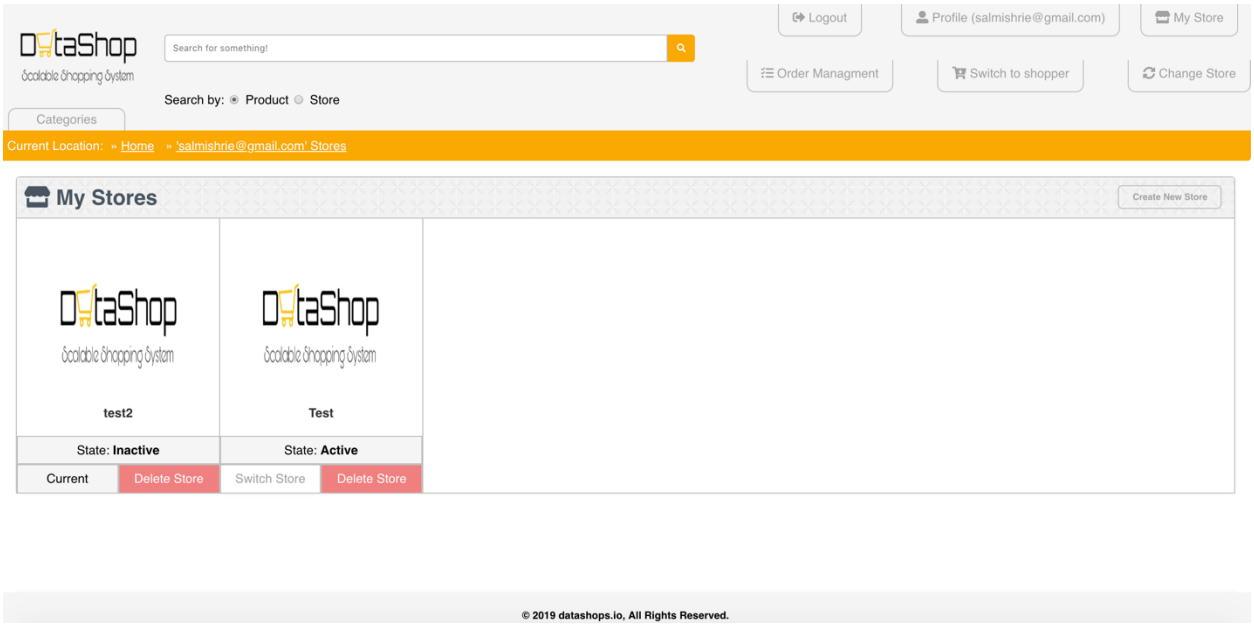


FIGURE 42 SWITCH STORES

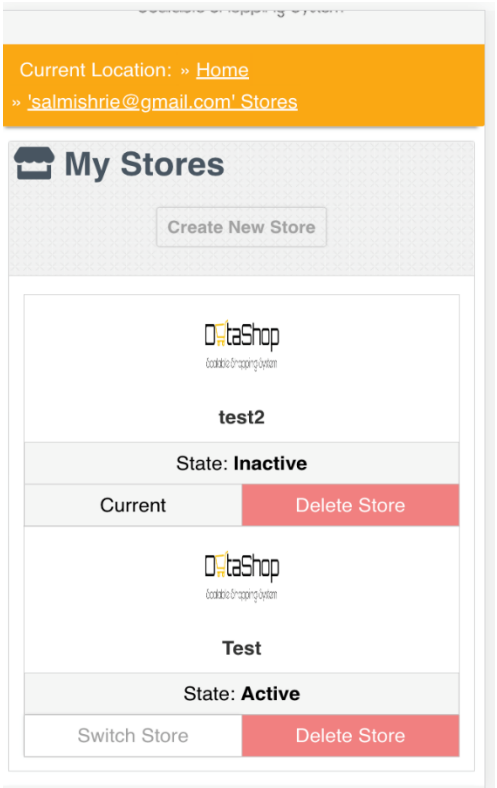


FIGURE 43 MOBILE SWITCH STORES

4.3.15 ORDER MANAGMENT

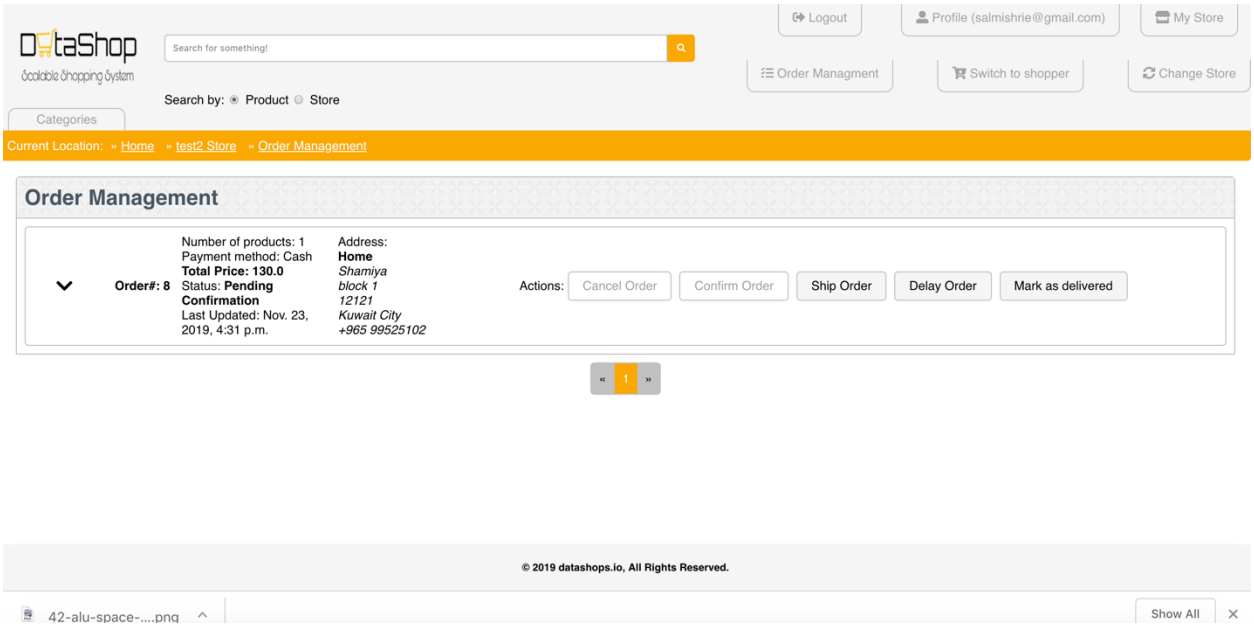


FIGURE 44 ORDERS MANAGEMENT

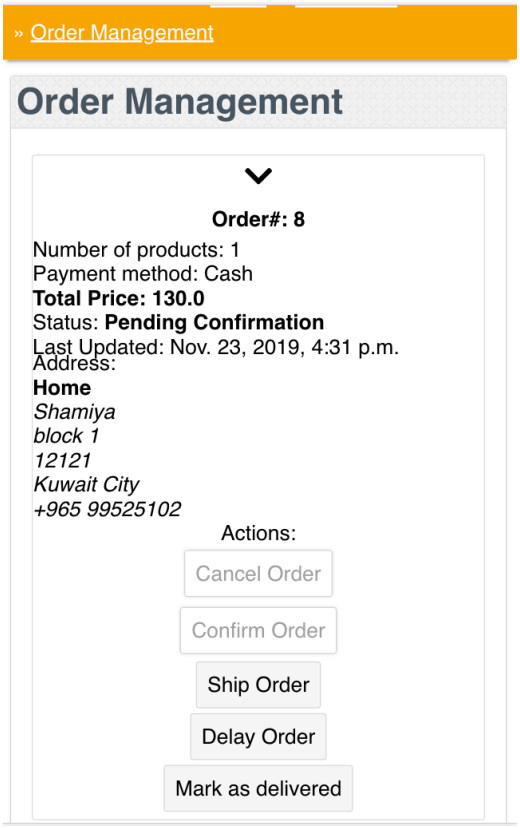


FIGURE 45 MOBILE ORDERS MANAGEMENT

4.3.16 FAVORITE LIST

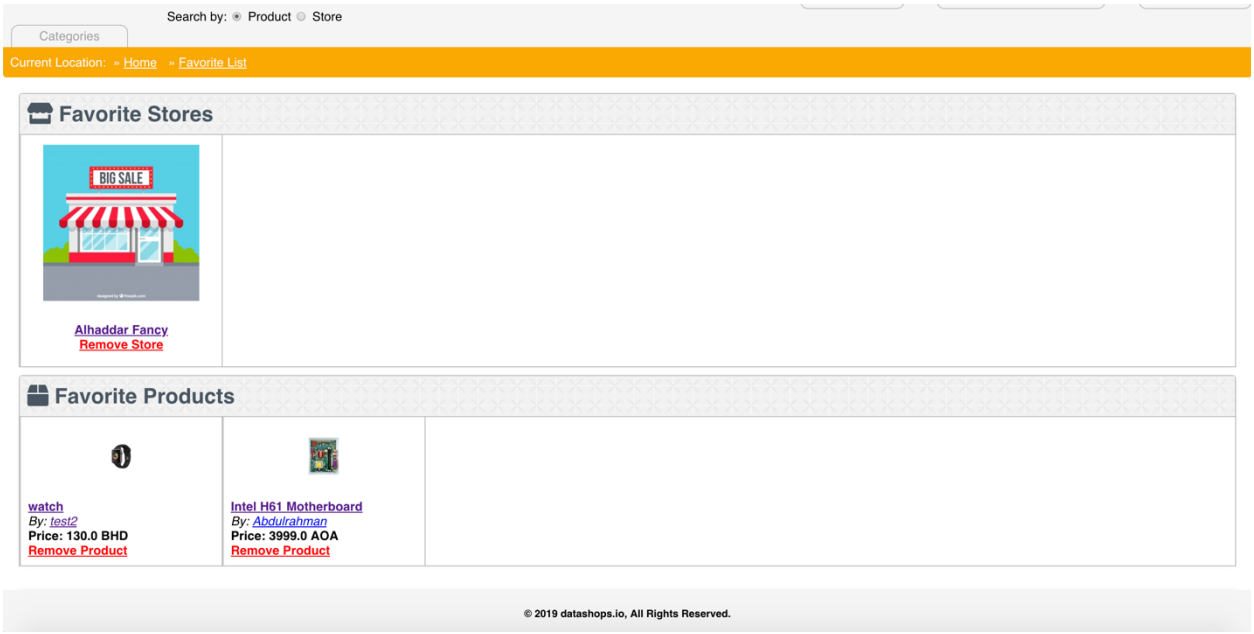


FIGURE 46 FAVORITE LISTS

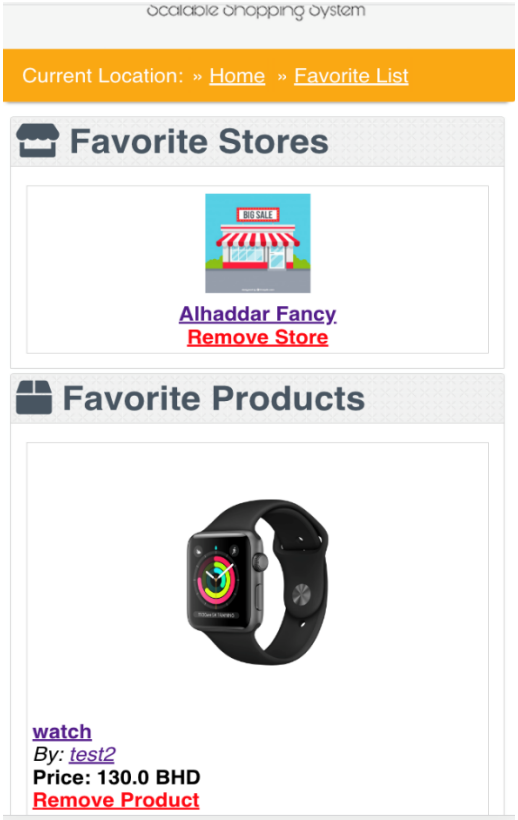


FIGURE 47 MOBILE FAVORITE LISTS

5. MODIFICATIONS OF THE ORIGINAL PLAN

This chapter includes all the changes made since the start of the project until the last phase and the final deliverable. The chapter is going to be divided into sections where each section holds the changes made in one field like requirements, design, models, etc. It is important to note that this chapter will only discuss the changes and will not get exposed to the whole plan to avoid repetition and to put emphasis on what has been changed only and for what reason.

5.1 REQUIREMENTS

Many requirements have been either added or removed to the original backlog of the project due to the wide range of the base requirements which either promoted new requirements or inhibited old ones.

The requirements which were removed from the project are:

Requirement Handle the database as a separate component for each instance.

Requirement Use a separate storage for each instance.

Requirement Log the system requests using MongoDB.

Requirement A user can be either a shopper or a store owner, but not both.

Requirement A store owner can have only one store registered in the system.

Requirement Registration form included email, password and a store name.

Requirement Allow store owners to customize their store pages using a wizard.

Requirement Usage of MongoDB as the main database engine of the system.

Requirement Store owners can export their stores' design to be latter used.

Requirement Allow guest users to use the system and do shopping.

Requirement	Allow users to find stores by categories.

On the other hand, many requirements were added to the project which are presented as follows:

Requirement	Each user must have a profile.
Reason for addition	To hold extra information required for the shopping.

Requirement	Stores can be listed and browsed.
Reason for addition	To give shoppers a general overview of what stores the system has and provide easy access to the store pages

Requirement	Password must be strong or at least has medium strength.
Reason for addition	To provide an extra layer of security and making password cracking harder.

Requirement	Products must be able to have multi images.
Reason for addition	To provide a better view of the products.

Requirement	Products must be categorized on a minimum of 2 levels.
Reason for addition	To make the products browsing easier and more convenient.

Requirement	Configure multi-currency support by the admin in the initialization of the system.
Reason for addition	To make the system more stable and provide a fixed price for the products, because the currency is always changing and unintentional loses and gains may occur.

Requirement	Cart must auto calculate the prices of the products.
Reason for addition	The products are going to have discount and shipment fields, which need extra calculation functionality.

Requirement	Each product must have its page that holds its information.

Reason for addition	Products will have much valuable information such as user rating and store name, these pieces of information cannot be listed in the products lists, thus a separate view is added for each product on the system.

Requirement	Only logged in shoppers can use the cart.
Reason for addition	The cart logic is going to need more information about the user to calculate the order status, thus only users with accounts can use the cart and proceed to checkout.

Requirement	When an order is shipped, it is automatically saved in order history.
Reason for addition	To make the order history specialized for shipped orders and make the user able to distinguish between shipped and unshipped orders.

Requirement	Shoppers can add/remove items to the cart.
Reason for addition	To make the cart more convenient to use.

Requirement	Use Bootstrap to make the UI simple and elegant.
Reason for addition	Bootstrap provides a ready to use design classes to can help to formalize the system design in a simple, yet elegant way.

Requirement	Seller can register in the system using a special signup page.
Reason for addition	Additional information is required for the seller account, and to distinguish between normal shoppers and sellers accounts.

Requirement	Each seller must have only one store registered in the system.
Reason for addition	To avoid the complexities of communications between the store owners and customers.

Requirement	Storeowners can see analytics of their products.
Reason for addition	To make the system smarter and more dynamic to use and to provide feedback to the storeowners about their products.

Requirement	Each shopper must have an address registered in the system before an order is initiated.
Reason for addition	To make the checkout process more automated and easier for the customers.

Requirement	The system interface should be simple and constructive.

Reason for addition	To make the system more likely-able to be used by different types of customers.

Requirement	Create an EC2 instance and host the system's website on it.
Reason for addition	To make the system reachable online and useable by users and testers.

Requirement	Host the system on Docker.
Reason for addition	To make every module of the system well separated and can run no matter what the underlying machine is.

Requirement	Host Docker on the EC2 instance.
Reason for addition	To make docker the main container that runes the system.

Requirement	Log the system request by using MongoDB for further inspection of user behavior.
Reason for addition	To better monitor the system and use the user behavior for future behavior analysis.

Requirement	Stores can be deactivated using profile.
Reason for addition	To ease the usability of the system by store owners.

Requirement	Empty search queries are not allowed in the system.
Reason for addition	To make the system more convenient to use.

Requirement	Make the UI mobile friendly.
Reason for addition	Because the system has no mobile application, mobile users should be able to use the system correctly.

Requirement	Create a view for the store owner to view the sales of the store in a chart.
Reason for addition	To make the system more usable by store owners.

Requirement	Add a basic authorization module to your system such that every single view calls the authorization.

Reason for addition	To make the system more secure and prohibit unlogged in users to use a certain function.

Requirement	Restrict sellers to a fixed number of products added per hour. A user should not be allowed to add 1000 products an hour.
Reason for addition	To protect the system against denial of service attacks.

Requirement	Implement registration access code that can be used to restrict the registration of shop owners.
Reason for addition	To regulate the registration process of sellers.

Requirement	Populate the database with multiple stores and with each store populated with multiple products.
Reason for addition	To allow the system to be tested effectively.

Requirement	Create a load balancer to distribute traffic among servers and to increase the reliability of the system.
Reason for addition	To allow the system to be fully scalable.

Requirement	Use AWS RDS service to host the system database for more scalability.
Reason for addition	To avoid the bottleneck of database access.

5.2 SYSTEM DESIGN

As can be seen in the System Design chapter of the report, the system design has changed to allow the scalability to take more place into the system architecture itself. The new system design relies on using the EC2 service of AWS to host the system files and code, while S3 is used as the system storage to store static files of the system, and finally the usage of RDS to hold the database of the system. Amazon's load balancer was used to receive the users' requests and perform load balancing to distribute the requests on the available machines. Finally, Route 53 was used to create and manage the DNS records.

6. IMPLEMENTATION FRAMEWORK AND DETAILS

This section will contain details about the frameworks and platforms that been used during the construction of the system. Besides, the coding decisions the team has made and the CASE tools that have been used during implementation.

6.1. FRAMEWORKS AND PLATFORMS

The project is aimed to be used in web browsers, targeted at mobile and desktop platforms. This was achieved by implementing responsive CSS stylings allowing different looks to make the system usable for both parties.

Django framework is used to construct the components related to the system. Django is a Python-based framework that follows the MVT (Model – View – Template) architectural pattern as shown in Figure 4.

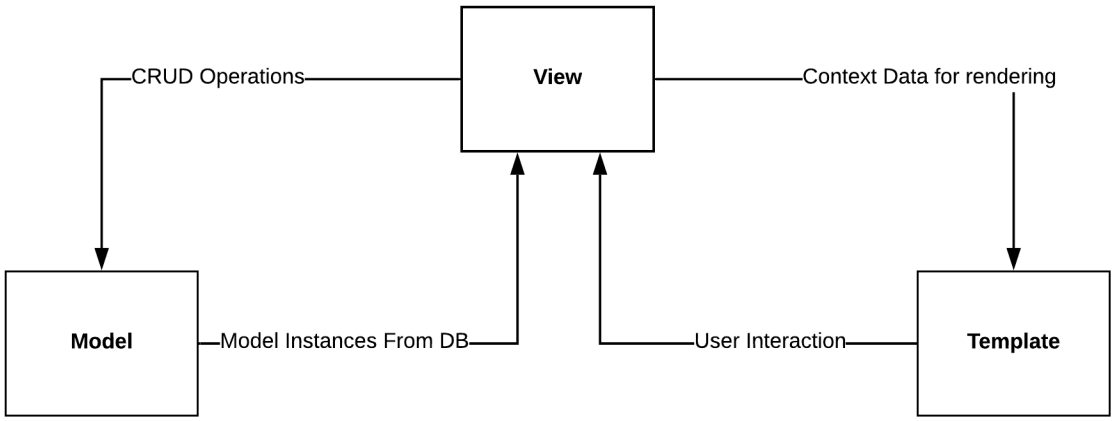


FIGURE 48 MODEL VIEW TEMPLATE PATTERN

The request starts its life cycle inside the framework by going through the view, where model fetching and business logic will be applied. After applying the logic, the view will return the HTML response that is rendered by the template rendering engine provided by Django.

6.2.COMPONENT AND CODE REUSE

Django formulates the concept of apps as a way to describe different components, providing separation of concerns. Some apps were implemented by the team members while others were library-like apps that had to be configured properly for usage, described in detail as follows:

- **Team made components**

Component	**Datashop**
Responsible functions	- Display the home page

Component	**Order**
Responsible functions	- Show order history - Show order details - Delete orders - Display the home page

Component	**Cart**
Responsible functions	- Cart Checkout. - Show cart contents.

Component	**Product**
Responsible functions	- Creating new products. - Show product details. - Update product details. - Delete product. - Adding product to cart. - Removing the product from the cart. - Show products in a specific category. - Product and store search.

Component	**User Profile**
Responsible functions	- Show user profile. - Update the user password. - Update the user store. - Add a delivery address. - Remove delivery address.

Component	**Store**
Responsible functions	- Show store details. - Creating new stores. - Changing between old and new stores. - Deleting stores.

Component	**Users**
Responsible functions	- User login. - User signup. - Switching between store owner and shopper.

- **Library components**

Component	**Django Storages**
Used functions	- Providing a custom storage backend to access AWS S3 directly from the app.

Component	**Django Extensions**
Used functions	- Generating model graph for the database.

Component	**AWS Boto3**
Used functions	- Call AWS API endpoints to access the resources of the project

Component	**Psycopg2**
Used functions	- PostgreSQL backend to communicate with the database.

Component	**Python Image Library**
Used functions	- Provide a means to open, manipulate and store images.

Component	**Django Rate Limit**
Used functions	- Limit users from accessing specific views more than X times in a limited period

6.3. CASE TOOLS

A set of CASE tools were used to aid the construction and engineering of the project including design and testing tools. These tools were all used since the start of the project and the team continued using them until this moment.

Sparx Enterprise Architect: This tool was used to create diagrams of the system design and architecture. It offers a huge set of tools that enable the user to create all types of diagrams and convert them into code when done. The team has mainly used this tool for diagraming purposes.

Lucid Charts: Lucid Charts is also a diagramming tool that is hosted on the cloud and has many features that aids to create fancy and elegant UML diagrams.

Git: Git and GitHub were used to share the project files between the team members and host the project files online for easy access and migration for AWS EC2 instance.

VSCode: This is the main IDE the team used to construct the project. It is integrated with GitHub and provides insight into the code reviews and code snippets. This tool is also integrated with SSH which provided easy access to the server and AWS services, and many more features that made the team depend on it only without the need to use any tool out of it.

Adobe Photoshop: Which was used to design the system's logo.

7. TESTING

Tests for the project were written to guarantee software reliability and to mitigate problems and bugs during future development. Different types of tests were conducted to assure software quality from different perspectives. The following sections will discuss different types of tests such as unit, integration and stress tests.

7.1. TESTING PLAN

Testing as a procedure takes a lot of time and effort and must be planned properly to avoid any waste of resources and cover as many cases as possible. Taking that into consideration, testing must be planned in a way that avoids redoing any type of test and covering all possible test cases to assure the system's reliability and ability to holds still when such cases occur.

Tests were divided by type as unit tests, integration tests, and stress tests, and each test type was conducted and created separately of the other types to avoid misconception and misleading test cases. The testing plan started by first doing system-wide tests that indicate the whole system's behavior under certain conditions as the system was treated as a solid and unbreakable part and tested accordingly. These tests were all under the stress tests umbrella and were conducted at the start of the testing phase. Then, tests regarding the user's interaction with the system were done to observe how the system will behave given a certain input from the user.

These tests dived a little deeper into the system details and started treating the system as separate functions or components that the user could interact with, and the tests were all under the integration tests umbrella. Finally, the tests that regard the small system's details were conducted to ensure each function/piece of the system was working as expected. These tests fall under the unit tests umbrella where the system is treated as small units that each unit can be tested separately from the others.

7.2. UNIT TEST CASES

Unit testing is a software level test that focuses on each working function of the code and tests its behavior in certain conditions. While the unit is the smallest testable part of a system, it can do a certain job and perform a certain task where the test is assigned to check for its success/failure when given a valid/invalid input.

The system was exposed to 151 different unit tests, and each was for a special case regarding a function in the system. The unit tests were conducted for the models of the applications inside Django to ensure that when models change they don't affect the views that use these models, for the forms to ensure that forms are behaving as expected and they only accept valid input from the user, and finally for the view to ensure each view is giving the right input when given a certain output. The views of the system were the most complex part of the system to test because each view has many test cases as will be shown in the test table. The following table presents each test's number, name, expected result, and test result as conducted from Django. The table will walk on the apps' tests one by one including all the three types of units.

TABLE 1 UNIT TEST RESULTS TABLE

Test Number	Test Name	Expected Result	Test Result
Test#1	Cart checkout by anonymous users	The view redirects the anonymous user to the login page	pass
Test#2	Cart checkout by logged in shopper users	The view works as expected and redirects the shopper to the checkout page	Pass
Test#3	Cart checkout by logged in store owners	The view redirects the user to the home page	Pass
Test#4	Get cart contents by anonymous users	The view redirects the users to log in page	Pass
Test#5	Get cart contents by actual shoppers	The view renders all the cart contents of the shopper	Pass
Test#6	Get cart contents by store owners	The view redirects the user to the home page	Pass
Test#7	Test the shopper field in the cart model	The field usage and appearance are consistent	pass
Test#8	Test the count field in the cart model	The field usage and appearance are consistent	Pass
Test#9	Test the price field in the cart model	The field usage and appearance are consistent	Pass
Test#10	Test the updated field in the cart models	The field usage and appearance are consistent	Pass

Test#11	Test the timestamp field in the cart model	The field usage and appearance are consistent	Pass
Test#12	Test the string representation of the cart model	The field usage and appearance are consistent	Pass
Test#13	Test the product field in the entry model	The field usage and appearance are consistent	Pass
Test#14	Test the cart field in the entry model	The field usage and appearance are consistent	Pass
Test#15	Test the quantity field in the entry model	The field usage and appearance are consistent	Pass
Test#16	Test the string representation of the entry model	The field usage and appearance are consistent	Pass
Test#17	Test the name field in the payment method model	The field usage and appearance are consistent	Pass
Test#18	Test the string representation of the payment method model	The field usage and appearance are consistent	Pass
Test#19	Test the home page view of the website	The view should work without the need to sign in and it uses the correct template	Pass
Test#20	Test the home page product pagination	The products are paginated by 10	Pass
Test#21	Test home page product number at the last page of pagination	The last page of products most contain (total number of products % 10)	Pass
Test#22	Add product to shopper favorite list by anonymous user	the view must redirect the user to login page	Pass
Test#23	Add product to shopper favorite list by store owner	The store owner must be redirected to the home page	Pass
Test#24	Add inexistent product to the shopper favorite list	The product shall not be added and the view must redirect user to the product page	Failed, the view gets into an infinite loop trying to add an inexistent product. # fixed
Test#25	Add product to the shopper favorite list	The product is added to the list and the shopper is redirected again to the product detail page	Pass
Test#26	Add store to the shopper favorite list by anonymous user	The user is redirected to the login page	Pass
Test#27	Add store to the shopper favorite list by a store owner	The store owner gets redirected to the home page	Pass
Test#28	Add an inexistent store to the shopper favorite list	the store is not added and the shopper is redirected to the store page	Failed, got into an infinite loop trying to add the store to the favorite list. # fixed
Test#29	Add store to favorite list by the shopper	The store gets added to the list and shopper is redirected back to the store page	Pass
Test#30	View the favorite list of the shopper by anonymous user	The user must be redirected to the login page	Pass
Test#31	View the shopper favorite list by a store owner	The store owner must be redirected to the home page	Pass

Test#32	View the shopper favorite list by the actual shopper	The favorite list is rendered to the shopper	Pass
Test#33	Remove a product from the favorite list by an anonymous user	The user must be redirected to the login page	Pass
Test#34	Remove a product from the favorite list by a store owner	The store owner must be redirected to the home page	Pass
Test#35	Remove inexistent product from the list by the shopper	The view redirects the user to the list page without doing anything	Pass
Test#36	Remove existent product from the list by a shopper	The product gets removed from the list and the shopper is redirected to the list page again	Pass
Test#37	Remove a store from the favorite list page by an anonymous user	The user gets redirected to the login page	Pass
Test#38	Remove a store from the list by a store owner	The store owner gets redirected to the home page	Pass
Test#39	Remove an inexistent store from the list by the user	The view removes nothing from the list and the user gets redirected to the list page	Pass
Test#40	Remove a store from the list by the shopper	The store is removed from the list and shopper is redirected to the list view again	Pass
Test#41	Test the user field in the favorite product model	The field usage and appearance are consistent	Pass
Test#42	Test the product field in the favorite product model	The field usage and appearance are consistent	Pass
Test#43	Test the user field in the favorite store model	The field usage and appearance are consistent	Pass
Test#44	Test the store field in the favorite store model	The field usage and appearance are consistent	Pass
Test#45	Create a product as anonymous user	The user should be redirected to the login page	Pass
Test#46	Create a product as a logged in shopper with get method	The shopper must be redirected to the home page	Pass
Test#47	Create a product as a logged in shopper with post method	The shopper must be redirected to the home page	Pass
Test#48	Create a product as a logged in seller with get method	The seller is redirected to the product creation page	Pass
Test#49	Create a product as a seller with empty data with post method	The seller is redirected to the product creation page and the product is not created	Pass
Test#50	Create a product as a seller with data and post method	The product is created	Pass
Test#51	View the products details as anonymous user	The user can see the product details	Pass
Test#52	View the product details as a shopper	The shopper can see the product details page	Pass
Test#53	View the product details as a seller	The seller can see the product details page	Pass
Test#54	Update a product as anonymous user with get method	The user is redirected to the log in page	Pass
Test#55	Update the product as anonymous user with post method	The user is redirected to the login page	Pass
Test#56	Update the product with shopper and get method	The shopper is redirected to the home page	Pass

Test#57	Update the product with a shopper and post method	The shopper is redirected to the home page	Pass
Test#58	update the product as a seller with get method	the seller is redirected to the product update page	Pass
Test#59	Update the product as a seller with post and no data	The seller is redirected to the product creation page and the product is not created	Pass
Test#60	Update the product as a seller with post method and correct data	The product is updated successfully	Pass
Test#61	Delete a product by anonymous user and get method	The product is not deleted	Pass
Test#62	Delete a product as anonymous user with post method	The product is not deleted	Pass
Test#63	Delete a product as a shopper with get method	The product is not deleted	Pass
Test#64	Delete a product as a shopper with post method	The product is not deleted	Pass
Test#65	Delete a product as a seller with get method	The product is not deleted	Pass
Test#66	Delete a product as a seller with post method	The product is deleted	Pass
Test#67	Add a product to cart by an anonymous user	The product is not added and the user is redirected to the login page	Pass
Test#68	Add a product to the cart by a logged in shopper	The product is added to the shopper's cart	Pass
Test#69	Add an inexistent product to the cart by a shopper	The product is not added and the shopper is redirected to the error page	Failed, the cart tried to add the invalid product anyway. # fixed
Test#70	Add a product to the cart by the shopper with 0 quantity	The product is not added to cart	Pass
Test#71	Add a product to the cart by a seller	The product is not added to the cart and the seller is redirected to the home page	Pass
Test#72	Remove a product from the cart by anonymous user	The product is not removed and the user is redirected to the login page	Pass
Test#73	Remove a product from the cart by a logged in shopper	The product is removed from the shopper's cart	Pass
Test#74	Remove an inexistent product from the shopper cart	The product is not removed	Failed, the cart tried to remove the product anyway. #fixed
Test#75	Remove a product from the cart by a seller	The product is not removed and the seller is redirected to the home page	Pass
Test#76	View a subcategory with its products by anonymous user	The user sees the products in that subcategory	Pass
Test#77	View a subcategory with its products by shopper user	The user sees the products in that subcategory	Pass
Test#78	View an inexistent subcategory with its products by shopper user	The shopper is redirected to the home page	Pass
Test#79	View a subcategory with its products by seller user	The user sees the products in that subcategory	Pass

Test#80	Create a product using the product creation form with valid data	The product is created and saved in the database	Pass
Test#81	Create a product using the product creation form with invalid data	The product is not created	Pass
Test#82	Test the name field of the product model	The field usage and appearance are consistent	Pass
Test#83	Test the subcategory field of the product model	The field usage and appearance are consistent	Pass
Test#84	Test the store field of the product model	The field usage and appearance are consistent	Pass
Test#85	Test the price field of the product model	The field usage and appearance are consistent	Pass
Test#86	Test the description field of the product model	The field usage and appearance are consistent	Pass
Test#87	Test the quantity field of the product model	The field usage and appearance are consistent	Pass
Test#88	Test the string representation of the product model	The string usage and appearance are consistent	Pass
Test#89	View the store page as an anonymous user	The user should be able to see the store page	Pass
Test#90	View the store page as a shopper	The shopper is allowed to see the store page	Pass
Test#91	View the store page as a seller	The seller is allowed to see the store page	Pass
Test#92	Change the current store as an anonymous user	The user should be redirected to the login page	Pass
Test#93	Change the store as a shopper	The shopper should be redirected to the home page	Pass
Test#94	Change the store as a seller	The seller should see the change store page with its functions	Pass
Test#95	Create a new store as an anonymous user	The user is redirected to the login page	Pass
Test#96	Create a new store as shopper with post method and no data	The shopper must be redirected to the home page	Pass
Test#97	Create a new store as a shopper with post method and valid data	The shopper must be redirected to the home page	Pass
Test#98	Create a store as a seller with invalid data and post method	The store is not created and the seller is redirected to the store creation page	Pass
Test#99	Create a store as a seller with valid data	The store is created and the seller is redirected to the store page	Pass
Test#100	Change the current store as an anonymous user	The user is redirected to the login page	Pass
Test#101	Change the current store as a shopper user	The user is redirected to the home page	Pass
Test#102	Change the store as a seller	The current store is changed	Pass
Test#103	Create a new store using the store form itself	A new store should be created and saved in the database	Pass
Test#104	Create a store with the store form as anonymous user	The store is not created and the user is redirected to the login page	Pass
Test#105	Create a store with anonymous user with post method and no data	The store is not created and the user is redirected to the home page	Pass
Test#106	Create a store with as a shopper with post method and no data	The shopper is redirected to the home page	Pass

Test#107	Create a store as a shopper with post method and valid data	The shopper is redirected to the home page	Pass
Test#108	Create a store as a seller with post method and no data	The store is not created	Pass
Test#109	Create a store as a seller with post method and valid data	The store is created and saved to the database	Pass
Test#110	Create social media object using the form with valid data	The object is created and saved in the database	Pass
Test#111	Create social media object using the form with invalid data	The object is not created	Pass
Test#112	Test the store name field in the store model	The field usage and appearance are consistent	Pass
Test#113	Test the string representation field in the store model	The field usage and appearance are consistent	Pass
Test#114	Test the currency field in the store model	The field usage and appearance are consistent	Pass
Test#115	Test the business description field in the store model	The field usage and appearance are consistent	Pass
Test#116	Test the business category field in the store model	The field usage and appearance are consistent	Pass
Test#117	Test the is new business field in the store model	The field usage and appearance are consistent	Pass
Test#118	Test the has physical store field in the store model	The field usage and appearance are consistent	Pass
Test#119	Test the logo field in the store model	The field usage and appearance are consistent	Pass
Test#120	Test the name field in the currency model	The field usage and appearance are consistent	Pass
Test#121	Test the country field in the currency model	The field usage and appearance are consistent	Pass
Test#122	Test the Instagram field in the social media links model	The field usage and appearance are consistent	Pass
Test#123	Test the Twitter field in the social media links model	The field usage and appearance are consistent	Pass
Test#124	Test the Snapchat field in the social media links model	The field usage and appearance are consistent	Pass
Test#125	Test the Facebook field in the social media links model	The field usage and appearance are consistent	Pass
Test#126	Test the LinkedIn field in the social media links model	The field usage and appearance are consistent	Pass
Test#127	Test the YouTube field in the social media links model	The field usage and appearance are consistent	Pass
Test#128	Test the phone number field in the social media links model	The field usage and appearance are consistent	Pass
Test#129	Test the general login in the system	The user should be logged in and redirected to the home page	Pass
Test#130	Sign up as a shopper in the system with anonymous user and get method	The user is redirected to the login page	Pass
Test#131	Sign up as a shopper with anonymous user and post method with no data	The user is redirected to the sign up page	Pass
Test#132	Sign up as a shopper with anonymous user and post method with valid data	The user is signed up, logged in, and redirected to the home page	Pass
Test#133	Sign up as a shopper with already logged in user	The user is redirected to the home page	Pass

Test#134	Sign up as a user in the system with anonymous user and get method	The user is redirected to the login page	Pass
Test#135	Sign up as a seller with anonymous user and post method with no data	The user is redirected to the signup page	Pass
Test#136	Sign up as a seller with anonymous user and post method with valid data	The user is signed up, logged in, and redirected to the home page	Pass
Test#137	Sign up as a seller with already logged in user	The user is redirected to the home page	Pass
Test#138	Convert shopper to seller as anonymous	The user is redirected to the login page	Pass
Test#139	Convert shopper to seller as seller	The seller is redirected to the home page	Pass
Test#140	Convert shopper to seller as shopper with no store	The shopper is converted to seller and redirected to the store creation page	Pass
Test#141	Convert shopper to seller as shopper with store	The shopper is converted to seller	Pass
Test#142	Convert seller to shopper as anonymous user	The user is redirected to the login page	Pass
Test#143	Convert seller to shopper seller	The seller is redirected to the home page	Pass
Test#144	Convert seller to shopper as seller	The seller is converted to shopper	Pass
Test#145	Test the shopper creation form with valid data	A new shopper user in created in the system	Pass
Test#146	Test the shopper creation form with invalid data	Nothing is created	Pass
Test#147	Test the seller creation form with valid data	A new seller user in created in the system	Pass
Test#148	Test the seller creation form with invalid data	Nothing is created	Pass
Test#149	Create an address as anonymous user	The user is redirected to the login page	Pass
Test#150	Test the address form with valid data as shopper	A new address is created and saved in the system	Pass
Test#151	Test the address form with invalid data as shopper	Nothing is created	Pass

Having listed all the unit tests, their number may be satisfying, but do they cover the most important parts of the system? By using Coverage, a Python tool that measures the effectivity of the unit tests and how much code they cover. This tool provides a detailed report about each code file in the system and shows the lines that were covered/not covered in the unit tests. The tool also shows a general report about how many code lines are covered and the total percentage of the unit tests coverage of the code. It is important to mention that the report includes files that are not testable like scripts and algorithms, thus the total percentage is higher than what is shown in the tool report. Figure 49 shows the general report generated by the tool. The test report is included in the project files and can be seen in detail for further inspection. The total amount of code covered

with the project's unit tests is 88% while the real number may be higher as mentioned that some code files that cannot be tested such as scripts were included in the report of the tool.

FIGURE 49 UNIT TEST COVERAGE.

In addition to the Coverage tool report, the total percentage of failed test compared to success tests is 2.6% where only 4 tests failed out of 151 total unit tests.

7.3. INTEGRATION TESTS

Integration tests are conducted to check whether different modules are interacting correctly or not. In the following test cases, each view in the system will be rendered in the browser to check for template issues, CSS and JavaScript code will be evaluated based on the interaction and the purpose of the page. In other words, integration tests will combine the view, HTML template, CSS and JavaScript to validate that the page completes its purpose on all cases.

The tests will be run using selenium web driver to simulate user's interaction in the page. The following table will include the test case and the expected behavior and the result of each.

TABLE 2 INTEGRATION TESTS RESULTS TABLE.

#	Test Case	Expected Results	Test Results
1	Home Page	Home page renders with list of products and pagination	Failed, the products from deactivated stores were showing
1.1	Home Page	Home page renders with list of products and pagination	Pass
2	Header	All buttons in the header are rendered, clickable, search box is activated, categories list is showing	Pass
3	Error page	Error page displays errors in case of redirection.	Pass
4	Search box	Search should show results for both store and products search, if any.	Pass
5	Get favorite list	The favorite lists for both stores and products should render the elements, if any.	Pass
6	Remove favorite list	User should be able to remove elements from both lists	Failed, confirmation message did not work as expected.
6.1	Remove favorite list	User should be able to remove elements from both lists	Passed
7	Insert favorite list	User should be able to add elements in both lists	Failed, duplicate values are allowed

7.1	Insert favorite list	User should be able to add elements in both lists	Passed
8	User Profile	The profile should render as expected with working javascript to switch between the profile tabs in both store owner and shopper views	Passed
9	Product Details	Product details are rendered, with appropriate actions for each type of user. The store owner can delete and modify the product, shopper can add to favorite list and to cart	Failed, having multiple stores gives update button without having the store as active, resulting in system error.
9.1	Product Details	Product details are rendered, with appropriate actions for each type of user. The store owner can delete and modify the product, shopper can add to favorite list and to cart	Passed
10	Product Update	Product details should be updated from the form	Passed
11	Product Create	Product created should appear in the store and details with the correct form data	Passed
12	Category	Products from a specific category should show with their pagination	Pass
13	Creating Users	Users should be created (both shopper and store owner)	Failed, no error message is shown under wrong data.
13.1	Creating Users	Users should be created (both shopper and store owner)	Failed, creating the user is working but the info message about the redirection is not being shown
13.2	Creating Users	Users should be created (both shopper and store owner)	Pass
14	Login	Users should be able to login with appropriate credentials	Error, no error message is shown under wrong data.
14.1	Login	Users should be able to login with appropriate credentials	Pass

15	Logout	User should be able to sign out from the system	Pass
16	User Profile Change password	User profile forms should all give expected results with user interaction	Failed, password did not change
16.1	User profile Change password	User profile forms should all give expected results with user interaction	Pass
17	User profile address	Shopper should be able to remove/add an address with valid values	Pass
18	User profile social media	Store owner should be able to add social media links	Failed, non URL input gives HTTP 500 error
18.1	User profile social media	Store owner should be able to add social media links	Pass
19	Store Details	Store should have all its data rendered correctly	Failed, logging middleware caused a crash
20	Store Details	Store should have all its data rendered correctly	Passed
21	Store creation	New stores should be created from the form with the valid user input	Failed, invalid checkbox that is meant to be an internal attribute is exposed in the template
21.1	Store creation	New stores should be created from the form with the valid user input	Passed
22	Store owner stores	The page should render correctly with all the stores visible to the user, plus their current status	Pass
23	Logging	The website should send an axios http request to send the location, if the user accepts the permissions	Pass

7.4. STRESS AND PERFORMANCE TESTS

A stress test was done using a script written in Python that creates bulk asynchronous HTTP requests using an event loop and then reads the response from the server. A special machine with a network connection of 20 Gbps is used to run the script. Several tests were conducted to measure how the system behaves under certain circumstances and the results were extracted for each test. It turned out the system was able to scale up and down depending on the load on the machines' CPUs and the network traffic. Moreover, each test took almost 30 minutes to complete, and each test was conducted on a different type of EC2 instances.

The first 4 tests show how the system behaves under stress with an undiscovered performance bug. The last test will show the final results with a discussion regarding the said bug and how it was fixed.

Each test will start with one running instance and one launch configuration mentioned in each test section title, scaling horizontally from 1 to 5 running instances. The change in configuration will show the performance gain under vertical scaling. Test summary can be found under each subsection followed by raw data table and diagrams to show the relation between the test parameters as throughput (requests per second), number of instances, the response time (ms) and database CPU utilization

As a general note on the set of tests, increasing the instance power decreases the response time to user requests. Scaling up the system by initiating more instances allowed the system to handle more requests per second (RPM) because the new instances can handle more work in a minute. The last test will show how a database related bug was fixed which was causing database bottleneck, limiting the response time due to the high utilization of the database resources.

7.4.1 T3.MICRO (2 CPU) EC2, T3.MICRO DB (2 CPU)

These test results show that the response time of the system is poor when a single instance is running, due to the resources shortage of the instance type, however, the response time gets lower when the number of instances increases until it reaches the lowest recorded response time when 5 instances are up.

It's clear from figure 16 that the response time decreased as the number of instances increases. In *figure 17*, the number of instances (red line) is directly proportional to the throughput (blue line) which is the number of requests per minute, up until midway of testing, the result is unexpected but it was due to a bug that affected the performance of the system that has been fixed and the results can be found in the last test. Figure 18 illustrates the relationship between the throughput and the database CPU utilization.

TABLE 3 T3.MICRO (2CPU) RAW TEST DATA

Throughput (RPM)	Current Number of Instances	Response time (ms)	Time	DB CPU
690	1	9149	10:29	8%
420	1	9762	10:30	15%
616	1	9428	10:31	15%
612	1	9208	10:32	15%
409	1	10004	10:33	15%
610	1	10974	10:34	10%
789	2	8179	10:37	4%
1081	2	5593	10:38	17%
1208	2	4788	10:39	27%
658	2	8934	10:40	25.00%
1113	2	5143	10:41	20.00%
1115	2	5371	10:42	24.00%
1215	2	4826	10:43	28.00%
1202	2	3525	10:44	27.00%
1220	2	4753	10:45	28%
1800	3	4860	10:46	27%
1762	3	4252	10:47	28%
1811	3	3283	10:48	36%
2362	3	3266	10:49	41%
1238	4	4712	10:50	36.00%
1223	4	4786	10:51	28.00%
1604	4	4125	10:52	33.00%
713	5	8171	10:53	23.00%
977	5	6251	10:54	17.00%
1794	5	3259	10:55	32.00%
2668	5	2496	10:56	46.00%
2411	5	2431	10:57	57.00%
2254	5	2583	10:58	55.00%
2403	5	2725	10:59	51.00%
2243	5	2537	11:00	54.00%
2416	5	2461	11:01	56.00%
2422	5	2415	11:02	56.00%
2748	5	2329	11:03	55.00%
2992	5	1963	11:04	68.00%
2979	5	1961	11:05	69.00%

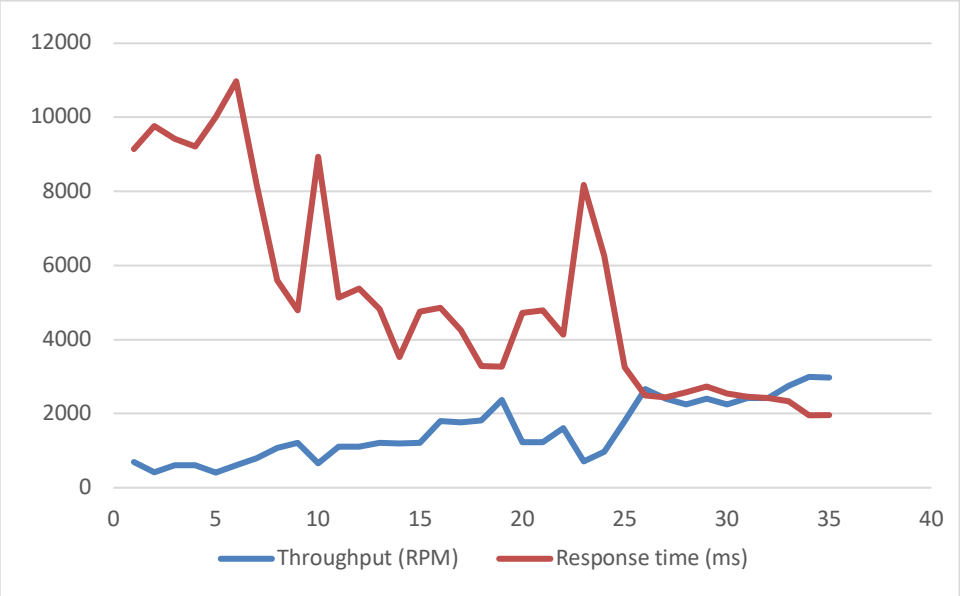


FIGURE 50 THE RELATIONSHIP BETWEEN THE THROUPUT AND RESPONSE TIME.

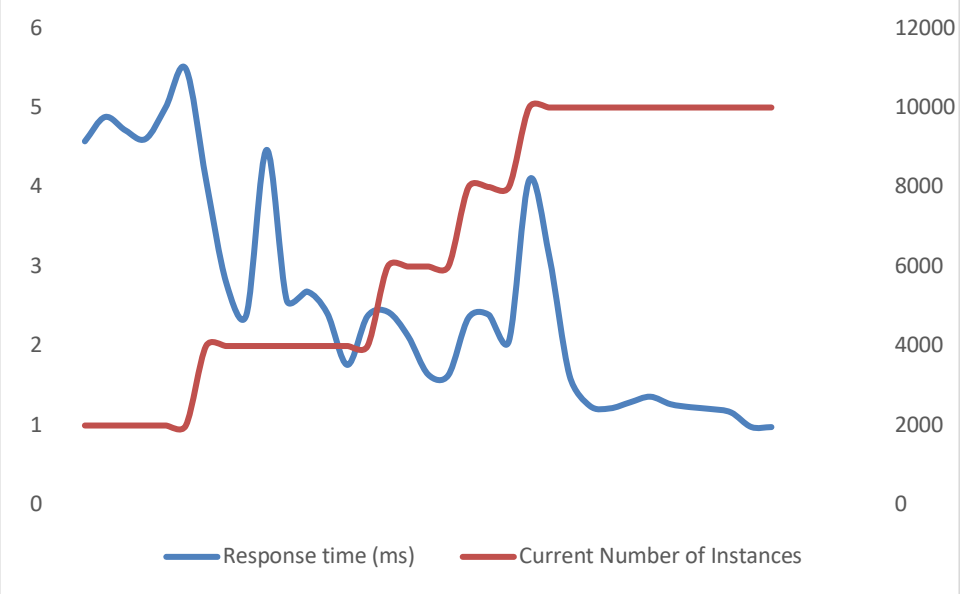


FIGURE 51 THE RELATIONSHIP BETWEEN THE RESPONSE TIME AND THE NUMBER OF INSTANCES.

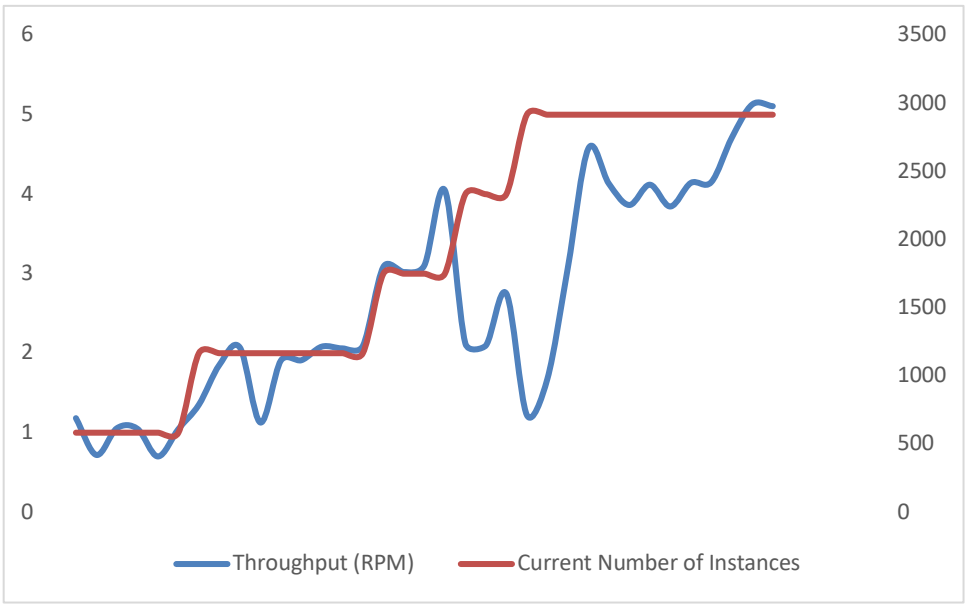


FIGURE 52 THE RELATIONSHIP BETWEEN THE THROUPUT AND THE NUMBER OF INSTANCES.

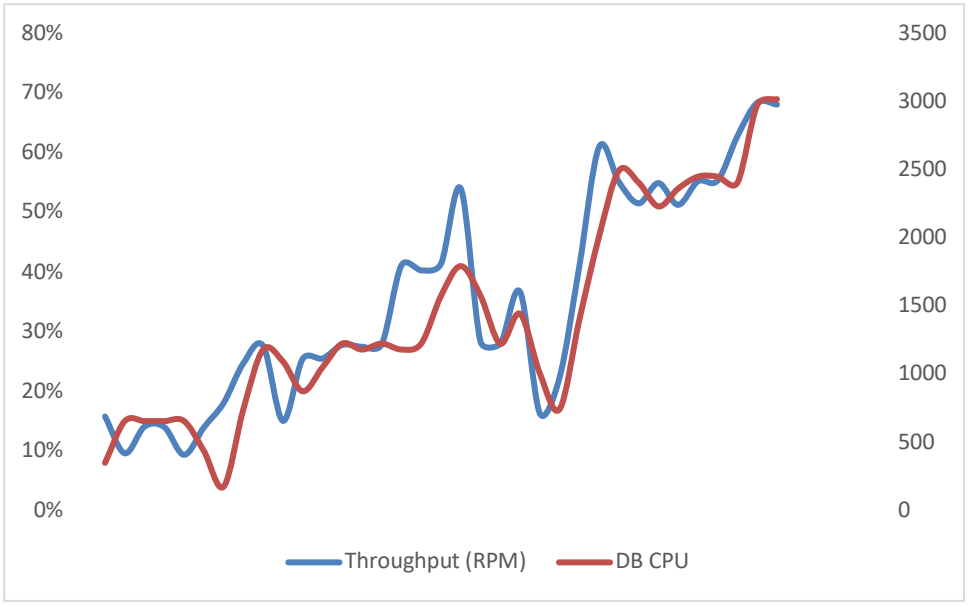


FIGURE 53 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE DATABASE CPU USAGE.

7.4.2 M5.XLARGE (4 CPU) EC2, T3.MICRO DB (2 CPU)

In this test, the instance configuration has changed to include 4 CPU cores and 8 GB of DRAM for each instance. Table 2 shows the raw test data.

Figure 19 shows the relationship between the throughput and the number of launched instances, where the number of launched instances now can handle more throughput than the previous instance type. Figure. 20 clearly shows the response time decreases with time while the throughput almost remains the same because overtime more instances get launched until the system reaches a steady-state. Figure. 21, on the other hand, shows the timeline of the throughput and the database CPU utilization which increases over time due to the fact that the database is a bottleneck in the system which shall be resolved in later tests.

TABLE 4: M5.XLARGE (4CPU) EC2, T3.MICRO DB(2CPU) RAW DATA

Throughput (RPM)	Current Number of Instances	Response time (ms)	Time	DB CPU
1223	1	4611	11:40	12.00%
1228	1	4632	11:41	27.00%
1219	1	4701	11:42	28.00%
1227	1	4665	11:43	28.00%
1227	1	4649	11:44	28.00%
1221	1	4611	11:45	28.00%
1926	2	4737	11:46	28.00%
2016	2	3656	11:47	28.00%
2020	2	2703	11:48	42.00%
2003	2	2869	11:49	46.00%
1621	3	3484	11:50	42.00%
1632	3	3528	11:51	37.00%
2398	3	2403	11:52	44.00%
1900	3	3027	11:53	54.00%
2418	3	2359	11:54	44.00%
2662	3	2122	11:55	55.00%
3176	3	1785	11:56	70.20%
3191	3	1774	11:57	67.00%
3528	3	1604	11:58	73.00%
3541	3	1589	11:59	80.00%
3300	3	1732	12:00	78.00%
3477	3	1625	12:01	80.00%
3242	3	1625	12:02	80.00%
3335	3	2678	12:03	70.00%
3550	3	3163	12:04	81.00%
3729	5	2903	12:05	80.00%
3801	5	2976	12:06	90.00%
3670	5	3108	12:07	89.00%
3880	5	2931	12:08	82.00%
4209	5	2614	12:09	95.00%
4269	5	2716	12:10	98.00%
4227	5	2746	12:11	99.00%
4215	5	2697	12:12	96.00%
4262	5	2597	12:13	96.00%
4274	5	2441	12:14	98%

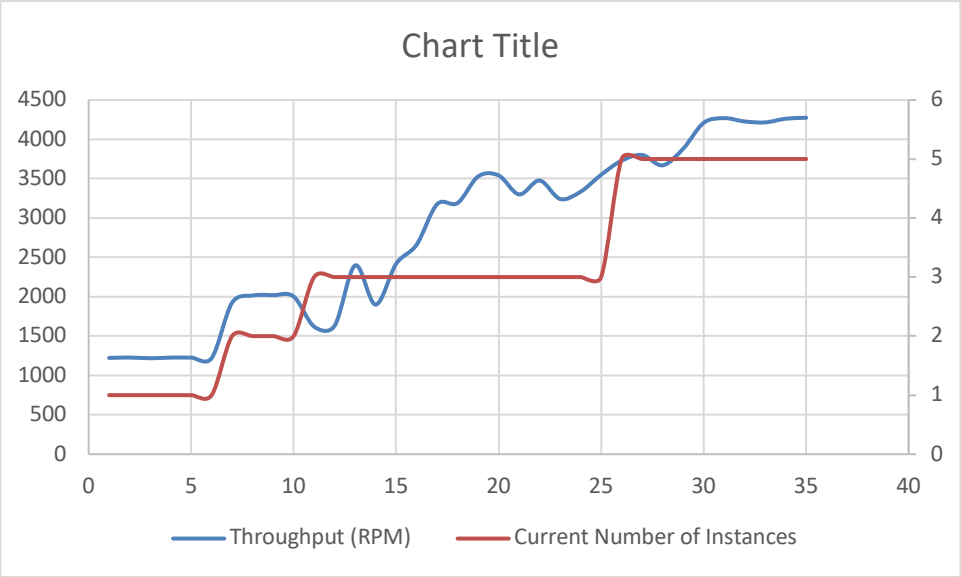


FIGURE 54 THE RELATIONSHIP BETWEEN THROUGHPUT AND NUMBER OF INSTANCES.

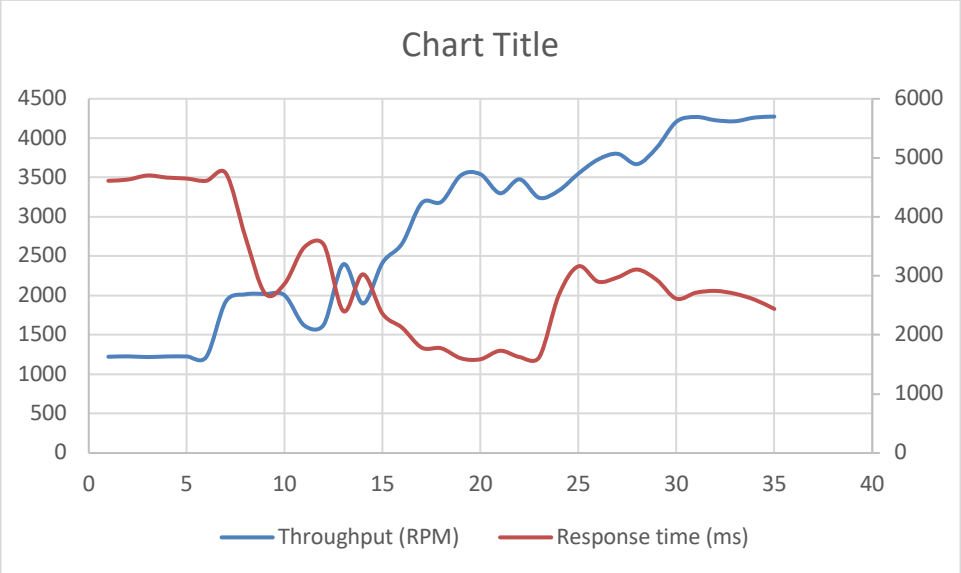


FIGURE 55 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE RESPONSE TIME

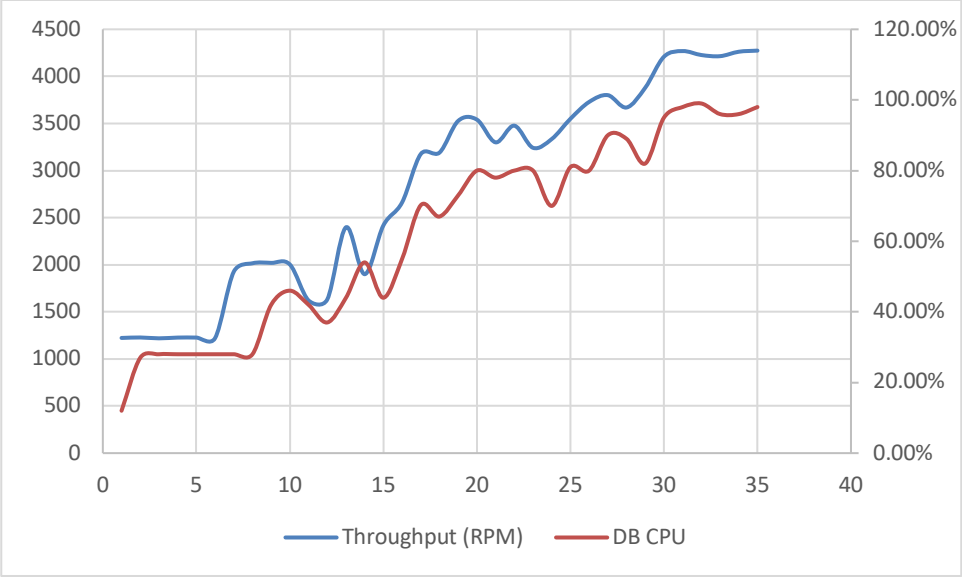


FIGURE 56 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE DABASE CPU UTILIZATION.

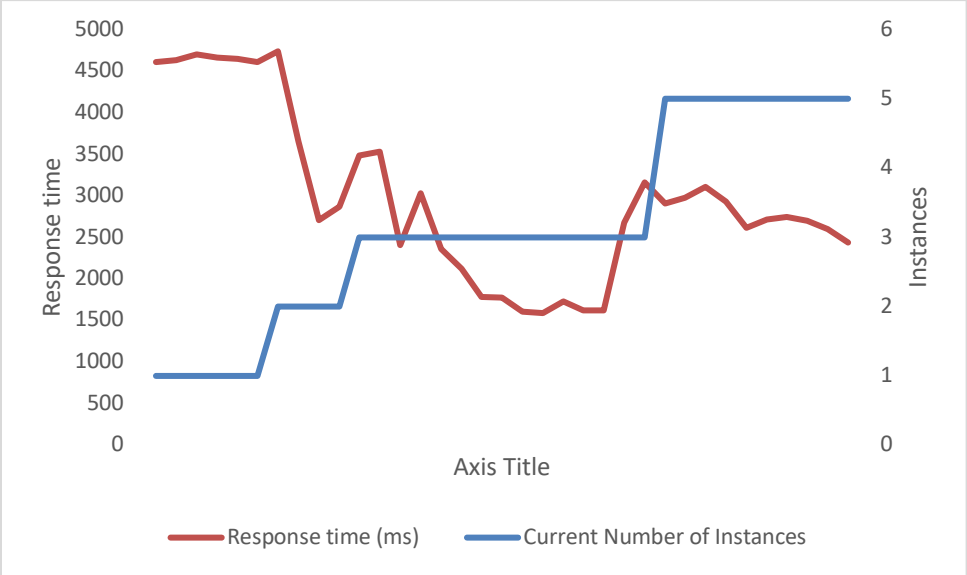


FIGURE 57 THE RELATIONSHIP BETWEEN NUMBER OF INSTANCES AND RESPONSE TIME

7.4.3 M5.4XLARGE (16 CPU) EC2, T3.2XLARGE DB (8 CPU)

This set of tests is done on an even more powerful EC2 instance and RDS instance, the general response time gave the throughput and the number of instances has generally decreased but the database access still introduces a bottleneck.

The response time and throughput start from 2155ms, 4705 rpm respectively when one instance is running. Under full load the system scales out to 5 instances giving response time and throughput equal to 314ms, 19101 rpm respectively.

Figure. 22 shows the relation between the throughput and the number of instances in a manner that shows the ability of the system to scale up when the throughput increases. It can be clearly seen that the system is doing well scaling up with the increase of the throughput.

Figure.23 shows great results of how the response time decreases when the throughput increases due to the fact that the current number of instances increases over time.

Figure. 24 graphs the relation between the throughput and the database CPU utilization, the database is still a bottleneck and the problem is not solved yet.

TABLE 5:M5.4XLARGE (16 CPU) EC2, T3.2XLARGE DB (8 CPU) RAW TEST DATA

Throughput (RPM)	Current Number of Instances	Response time (ms)	Time	DB CPU
1075	1	2155	1:03	12%
4705	1	2542	1:04	22%
4698	1	2425	1:05	22%
4700	1	2393	1:06	22%
4697	1	2428	1:07	22%
4699	1	2426	1:08	22%
4692	1	2410	1:09	22%
4709	1	2457	1:10	22%
6736	2	2430	1:11	22%
6214	2	2138	1:12	22%
6239	2	1819	1:13	28%
6914	2	1662	1:14	29%
8462	3	1366	1:15	35%
9391	3	1227	1:16	44%
9372	3	1234	1:17	45%
10649	3	1825	1:18	45%
13977	4	934	1:19	61%
13982	4	827	1:20	72%
13980	4	824	1:21	72%
13981	4	825	1:22	72%
16973	5	964	1:23	72%
18181	5	950	1:24	95%
18177	5	1130	1:25	96%
18786	5	1044	1:26	99%
19035	5	780	1:27	99%
19069	5	892	1:28	99%
19114	5	891	1:29	99%
19049	5	898	1:30	99%
19072	5	666	1:31	99%
19051	5	498	1:32	99%
19101	5	314	1:33	99%

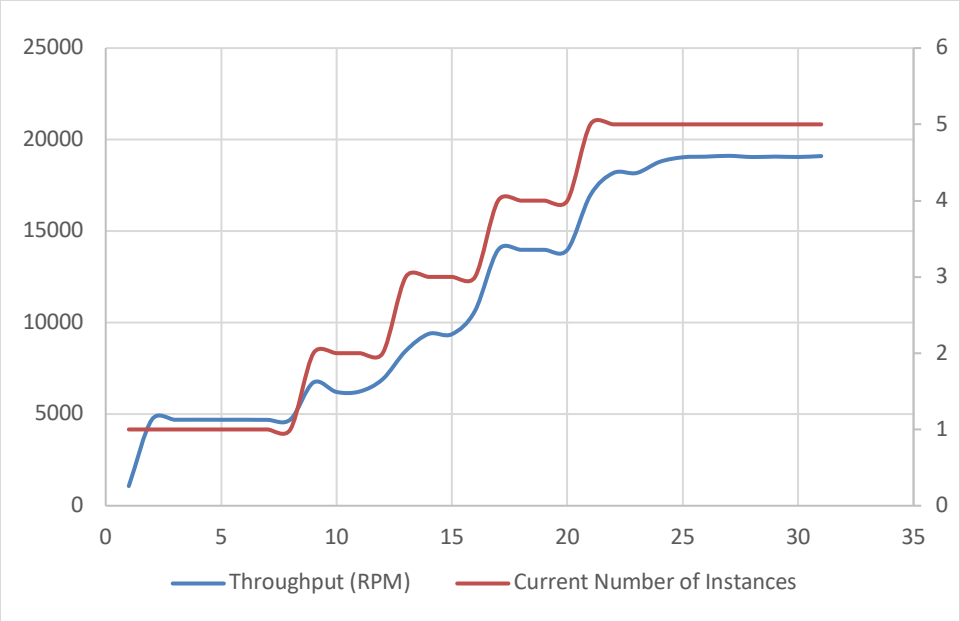


FIGURE 58 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE CURRENT NUMBER OF INSTANCES.

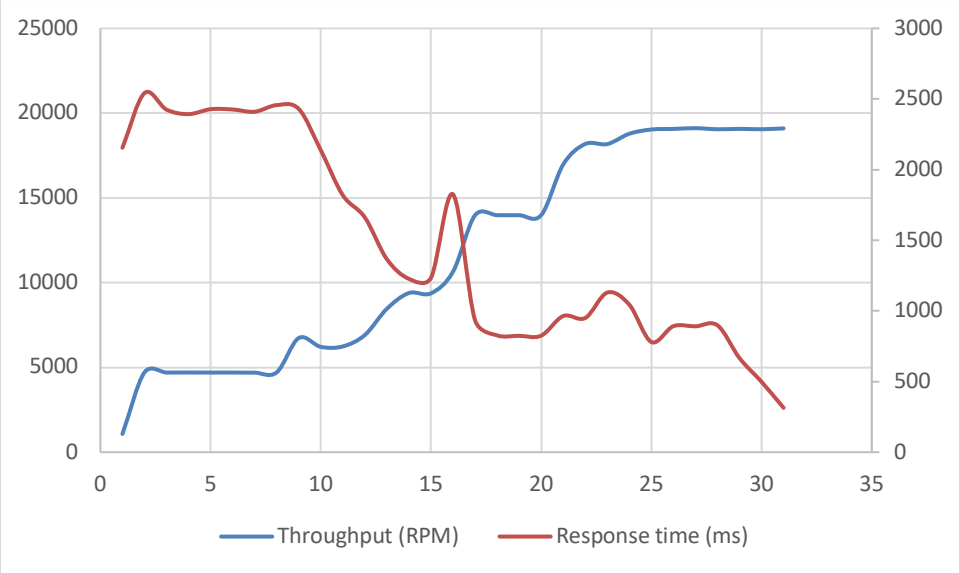


FIGURE 59: THE RELATIONSHIP BETWEEN THROUGHPUT AND RESPONSE TIME.

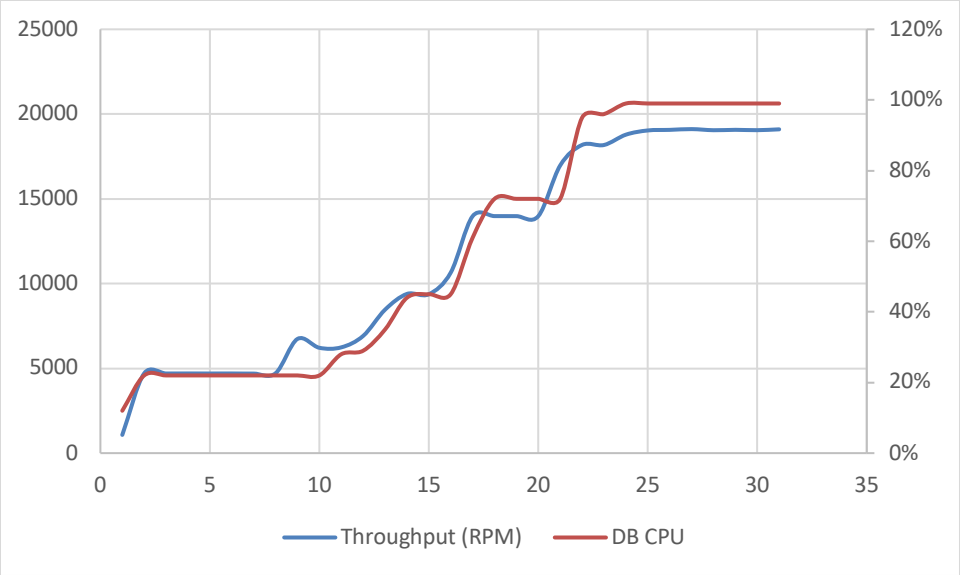


FIGURE 60 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE DB CPU UTILIZATION.

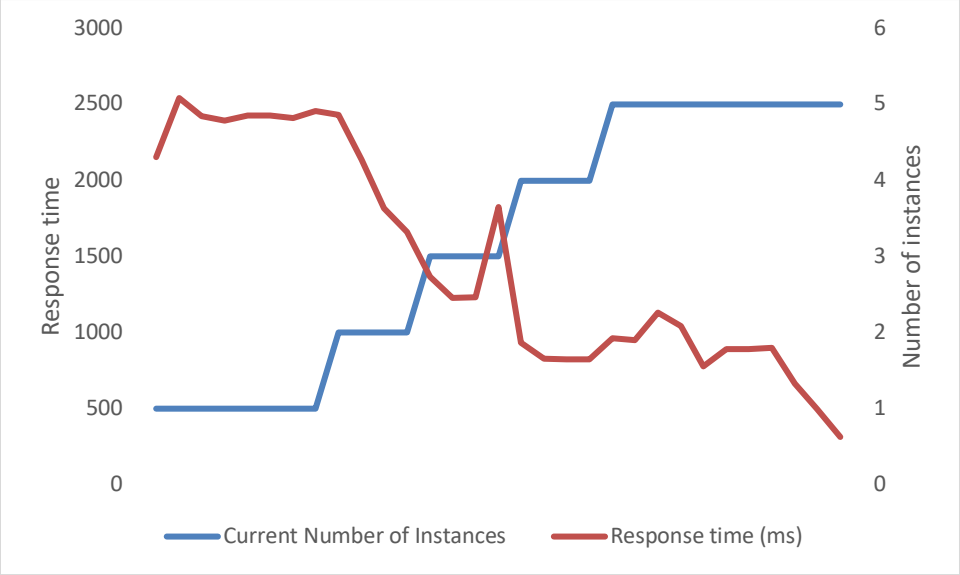


FIGURE 61 THE RELATIONSHIP BETWEEN THE CURRENT NUMBER OF INSTANCES AND THE RESPONSE TIME

7.4.4 M5.8XLARGE (32 CPU) EC2, M5.4XLARGE DB (16 CPU)

This test is using the most powerful EC2 instance type chosen by the team members, with 32 CPU cores per instance, 128 GB of RAM and a 16 CPU cores database.

Starting with one instance, the response time and throughput are equal to 2694ms, 9046 rpm respectively. Under full load, the system scales out to 5 instances with the configuration mentioned above, reaching response time and throughput equal to 1577ms, 35571 rpm respectively.

Figure 25 shows the relation between the throughput and the number of instances in the system, the system can scale up when the throughput increases and is also able to keep up good with high throughput.

In *Figure 26*, it can be noticed that there is an inconsistency in the response time, due to the database bug that was unknown during conducting this test, which also affected the throughput and caused some performance drops.

Figure 27 relates the database CPU usage with the response time. Notably, that the database high CPU usage at the end of the test caused a huge response time spike.

TABLE 6:M5.8XLARGE (32 CPU) EC2, M5.4XLARGE DB (16 CPU) RAW TEST DATA

0	Current Number of Instances	Response time (ms)	Time	DB CPU
1939	1	2129	1:55	10%
9046	1	2694	1:56	15%
9039	1	2576	1:57	18%
9043	1	2563	1:58	18%
9050	1	2542	1:59	18%
9038	1	2552	2:00	18%
9028	1	2337	2:01	18%
9046	1	2546	2:02	18%
9045	1	2527	2:03	18%
9027	2	2834	2:04	18%
9054	2	3271	2:05	18%
9093	2	3152	2:06	18%
9094	2	3332	2:07	18%
14789	2	2687	2:08	18%
16753	3	2539	2:09	32%
19689	3	1804	2:10	27%
20662	3	1698	2:11	54%
17972	3	2189	2:12	36%
18071	3	2024	2:13	35%
23084	3	1973	2:14	44%
26897	4	1789	2:15	61%
31781	4	1224	2:16	82%
25763	4	1124	2:17	67%
28581	4	1018	2:18	67%
34237	4	852	2:19	91%
33753	4	743	2:20	98%
34040	4	847	2:21	99%
34021	4	780	2:22	98%
33699	4	394	2:23	97%
25793	4	289	2:24	92%
26913	4	1515	2:25	56%
33867	5	1677	2:26	99%
34061	5	1659	2:27	99%
34404	5	1655	2:28	99%
35230	5	1611	2:29	99%
35737	5	1583	2:30	99%
35571	5	1577	2:31	99%

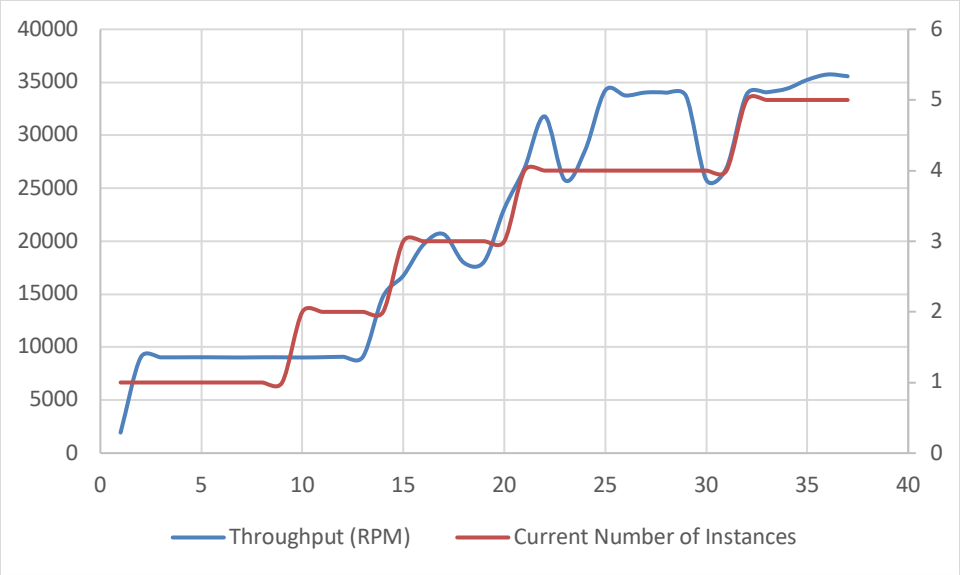


FIGURE 62 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE NUMBER OF INSTANCES.

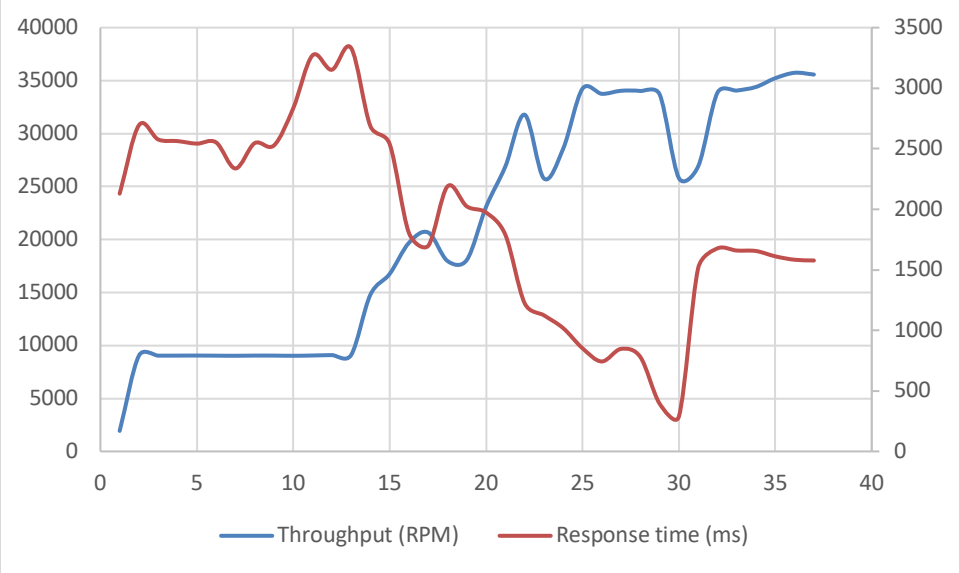


FIGURE 63: THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE RESPONSE TIME.

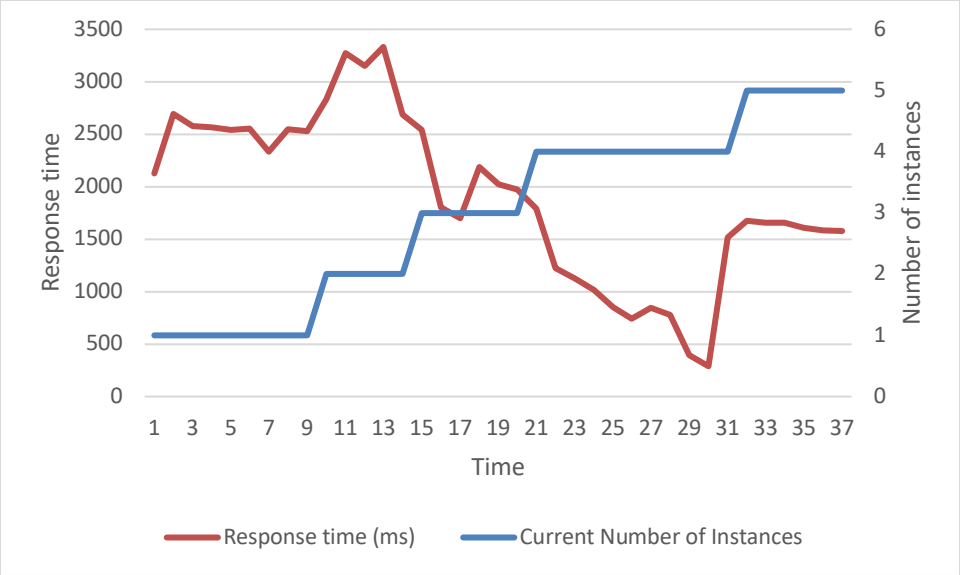


FIGURE 64 THE RELATIONSHIP BETWEEN THE RESPONSE TIME AND THE CURRENT NUMBER OF INSTANCES

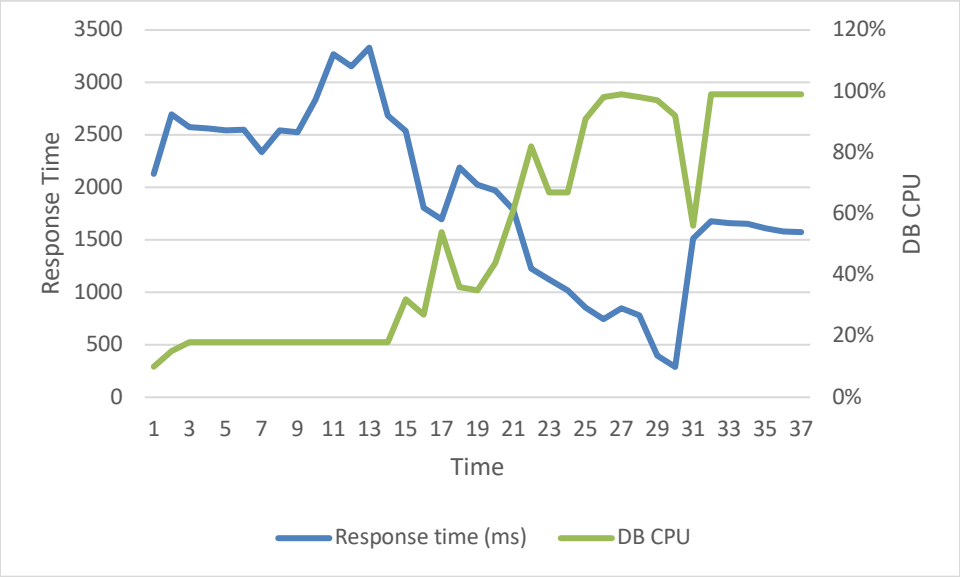


FIGURE 65 THE RELATIONSHIP BETWEEN THE RESPONSE TIME AND THE DB CPU

7.4.5 M5.4XLARGE (16 CPU) EC2, M5.4XLARGE DB (16 CPU) (4 READ REPLICAS)

It can be noted that the previous tests had a really bad performance in terms of response time. A webserver typically should aim for lower than 200ms response time to be considered reliable. After investigating and debugging, the team was able to locate and fix the issue.

The system is using Django Object Relational Mapping (ORM) to access the database. Due to misconfigured queries, the system was calling the database 160 times for a single page. This caused the system to overwhelm the database with unnecessary database hits and as a result, the performance hit was really high.

Even when the database queries were fixed, the database usage was still high and causing some issues, so horizontal scaling for the database was done.

Four databases read replicas were created along with the master database. The system configuration is the same as in the test 7.5.3. The performance gain from the fixes are very noticeable.

The raw data is shown at *Table 4.* The response time in the beginning was starting from 800ms. After fully scaling out to 5 instances, the response time got down to 187ms. This huge improvement is related to the fixes mentioned previously. In the next figures, it's interesting to observe that the number of instances is proportional to the throughput (1:1 relation). Thanks to reading replicas, the database is no longer a bottleneck and average CPU utilization reaches 30% under full load.

TABLE 7: M5.4XLARGE (16 CPU) EC2, M5.4XLARGE DB (16 CPU) (4 READ REPLICAS) RAW TEST DATA

Throughput (RPM)	Current Number of Instances	Response time (ms)	Time	Average DB CPU
6272	1	891.854532	1:04	6%
6307	1	883.8143644	1:05	7%
6349	1	855.3223448	1:06	6%
6358	1	858.1405661	1:07	8%
6372	1	849.3012853	1:08	5%
6279	1	862.4880957	1:09	8%
6308	1	857.3266921	1:10	7%
6363	1	857.4333091	1:11	9%
9897	2	577.7220209	1:12	10%
12725	2	443.8271742	1:13	9%
12575	2	443.0477578	1:14	11%
12591	2	442.8021668	1:15	11%
12681	2	442.8311896	1:16	10%
12715	2	443.0913081	1:17	11%
12708	2	443.6010511	1:18	11%
12544	2	447.4306824	1:19	10%
12662	2	440.606537	1:20	10%
18876	3	307.4983206	1:21	17%
18925	3	303.7387376	1:22	15%
18945	3	303.8823518	1:23	16%
18827	3	305.4328165	1:24	16%
17699	3	279.1139371	1:25	14%
25197	4	232.5979481	1:26	21%
25200	4	232.3260859	1:27	22%
26948	4	217.1942588	1:28	24%
30827	5	188.6309186	1:29	27%
30903	5	188.1206039	1:30	28%
30963	5	187.5250058	1:31	27%
30938	5	187.6114712	1:32	30%
30951	5	187.2296994	1:33	30%

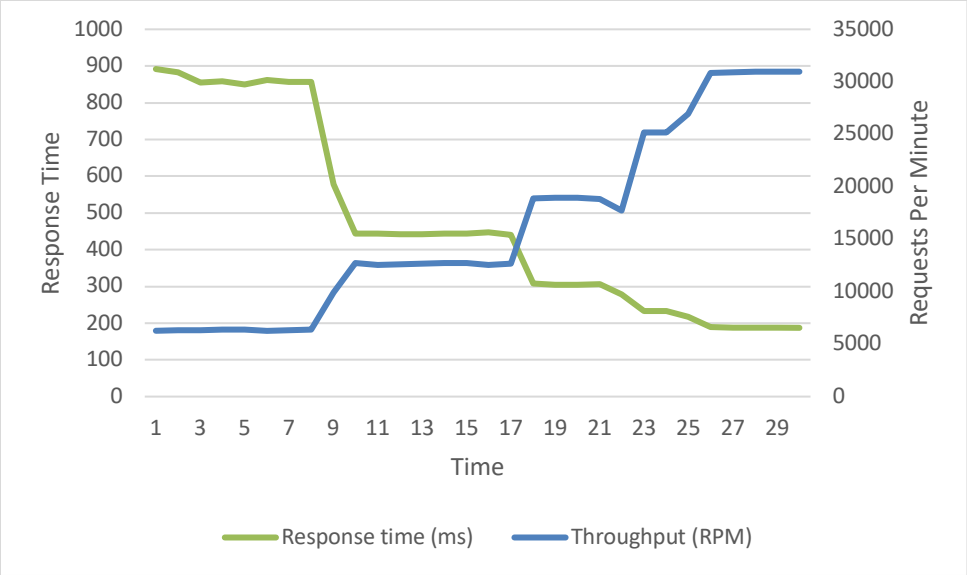


FIGURE 66 THE RELATIONSHIP BETWEEN RESPONSE TIME AND THROUGHPUT

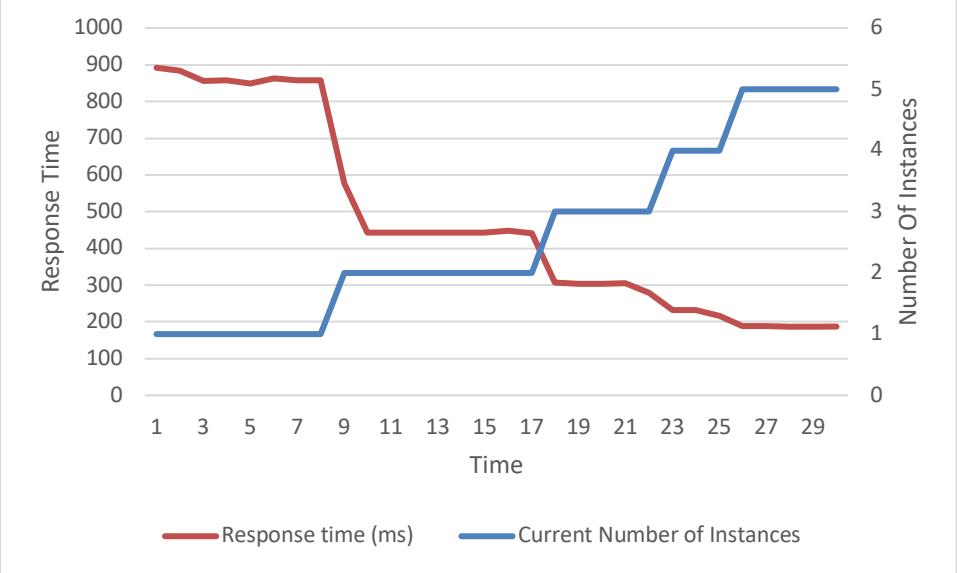


FIGURE 67 RELATIONSHIP BETWEEN THE RESPONSE TIME AND THE CURRENT NUMBER OF INSTANCES

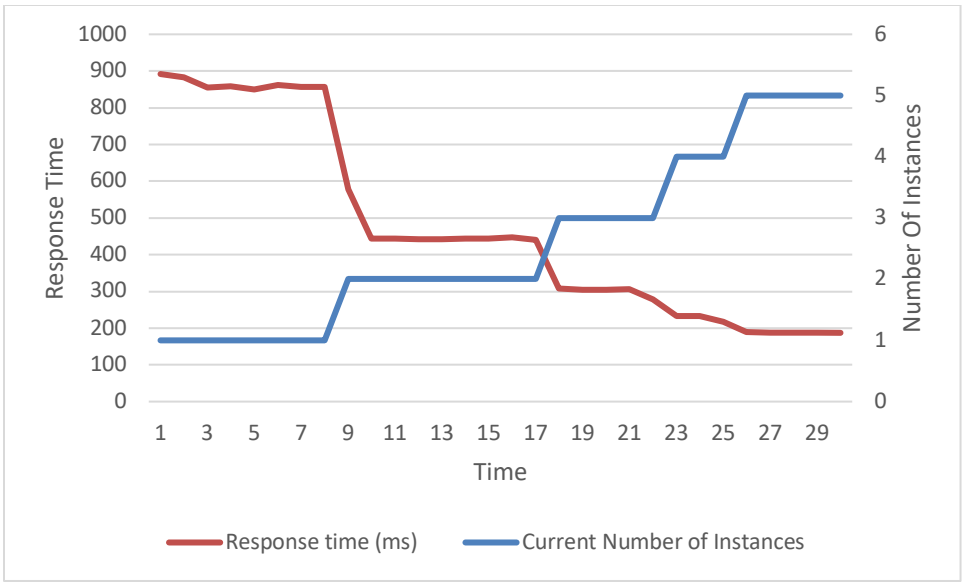


FIGURE 68 RELATIONSHIP BETWEEN THE RESPONSE TIME AND THE CURRENT NUMBER OF INSTANCES

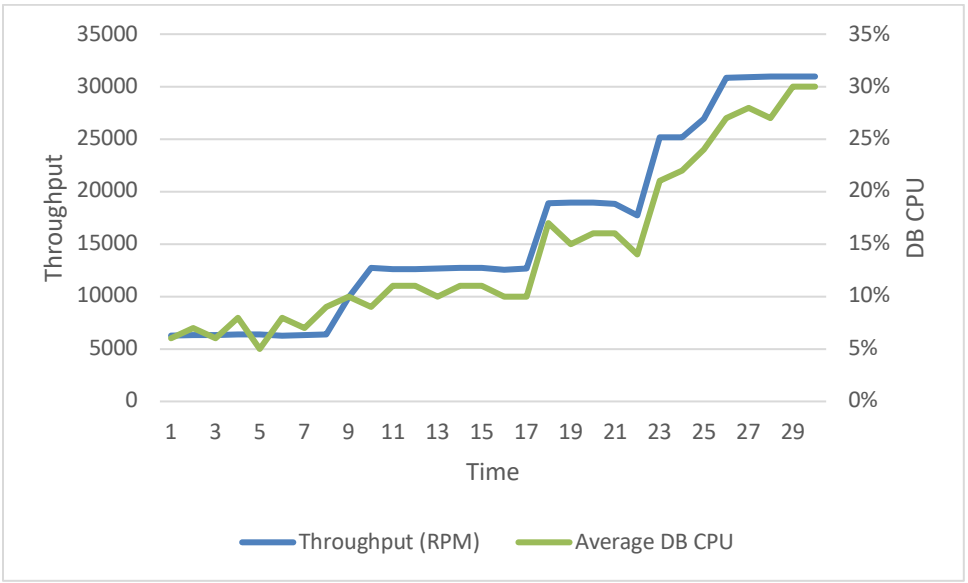


FIGURE 69 THE RELATIONSHIP BETWEEN THE THROUGHPUT AND THE AVERAGE DB CPU

7.5 END-USER TESTS

An end-user test was done to test the system from higher-levels. The team starts simulating end-users' behavior by asking users to do virtual shopping and store-owners to create their own stores in the system. The team has created a questionnaire that seeks end-users feedback on the experience and have them fill it up. This questionnaire was divided into 3 sections:

- System Usability Scale
- Use Case Usability Scale
- Further Questions (advanced questions that were asked to more experienced users of the system, where their answers will reflect exactly how the system behaves and how does it work).

The team has collected 62 end-users' responses, and the statistics of their answers are as follows:

7.5.1 SYSTEM USABILITY SCALE

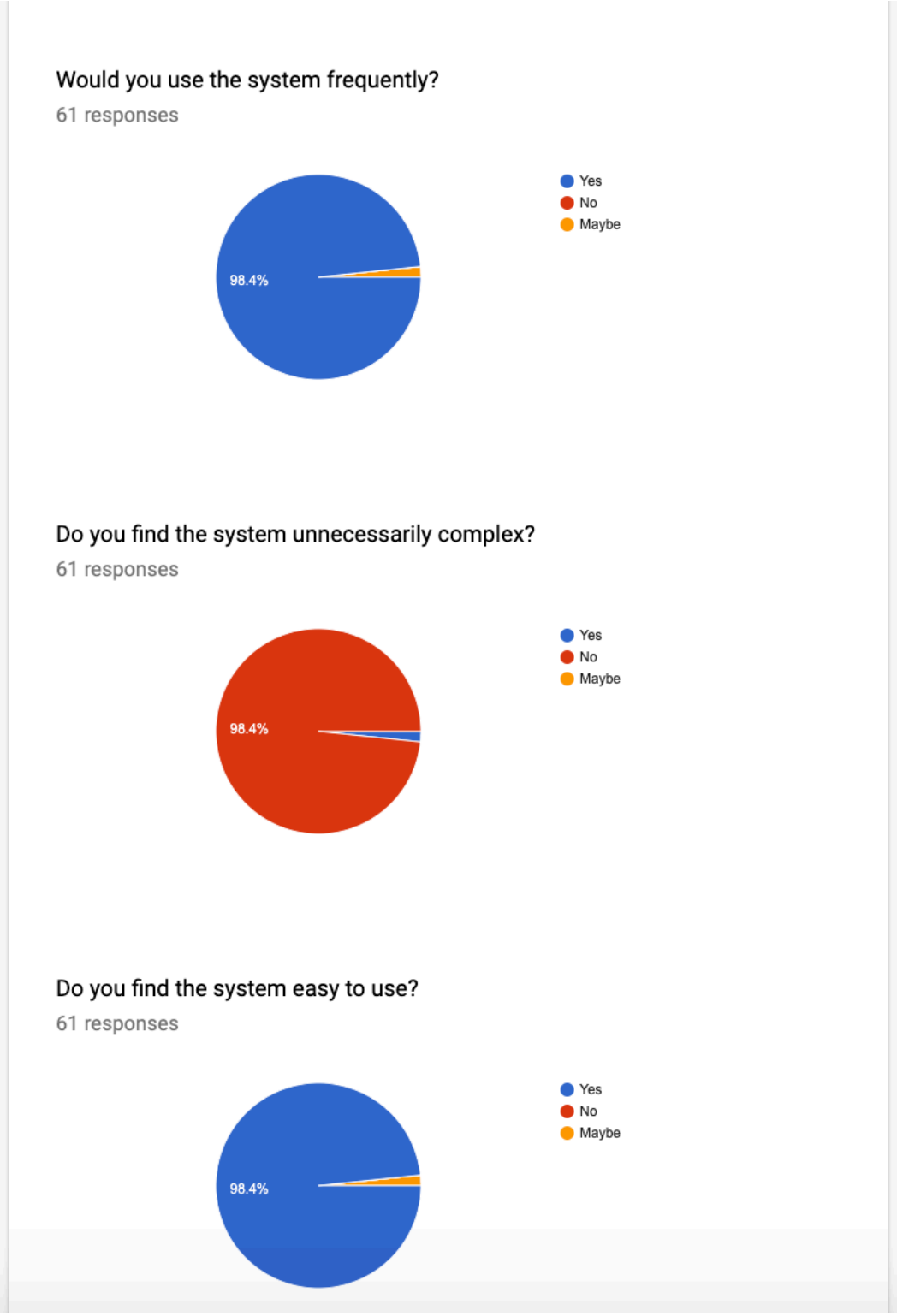


FIGURE 70 SYSTEM USABILITY SCALE TESTING STATISTICS

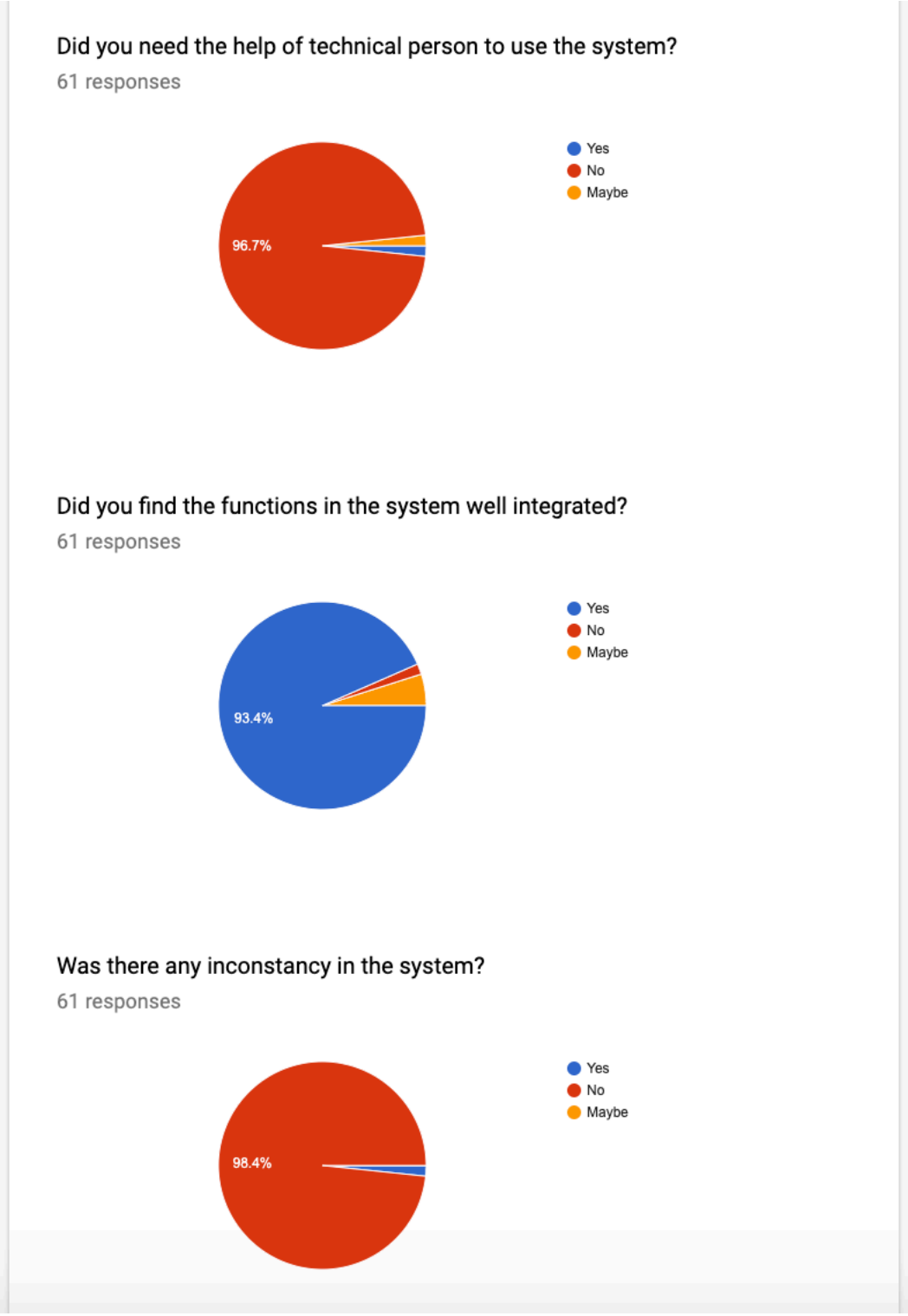


FIGURE 71 SYSTEM USABILITY SCALE TESTING STATISTICS

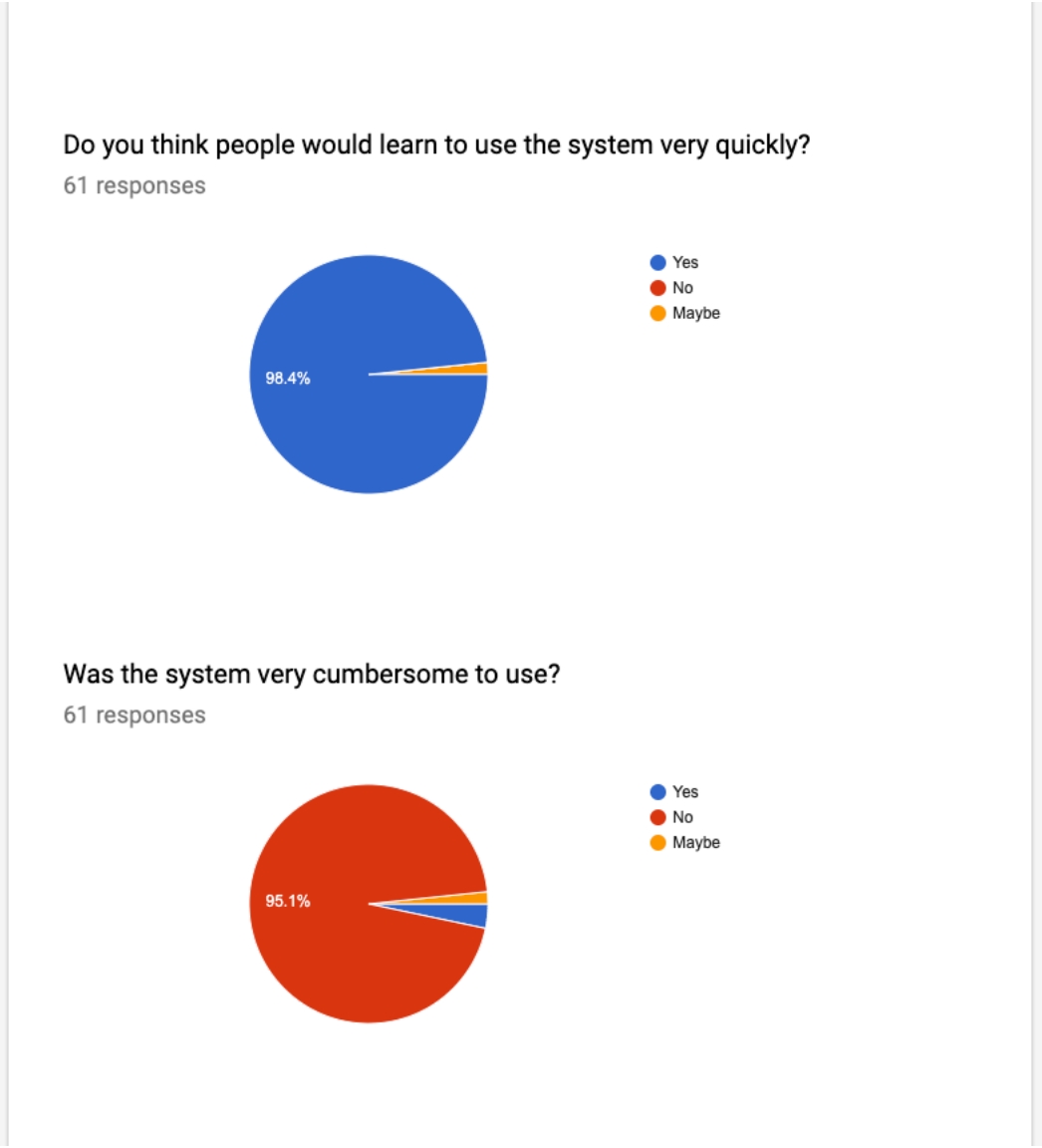


FIGURE 72 SYSTEM USABILITY SCALE TESTING STATISTICS

7.5.2 USE CASE USABILITY SCALE

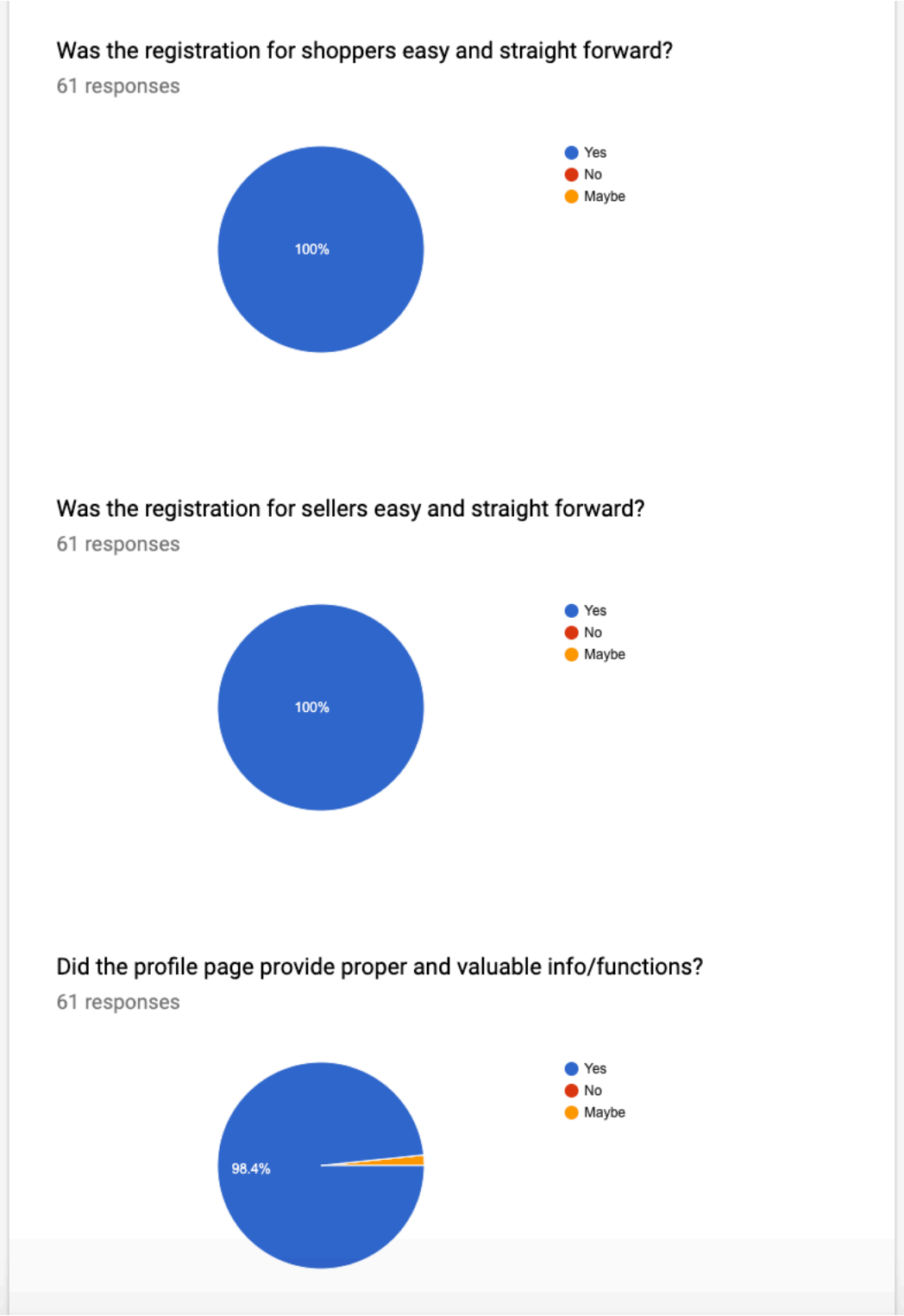


FIGURE 73 USE CASE USABILITY SCALE TESTING STATISTICS

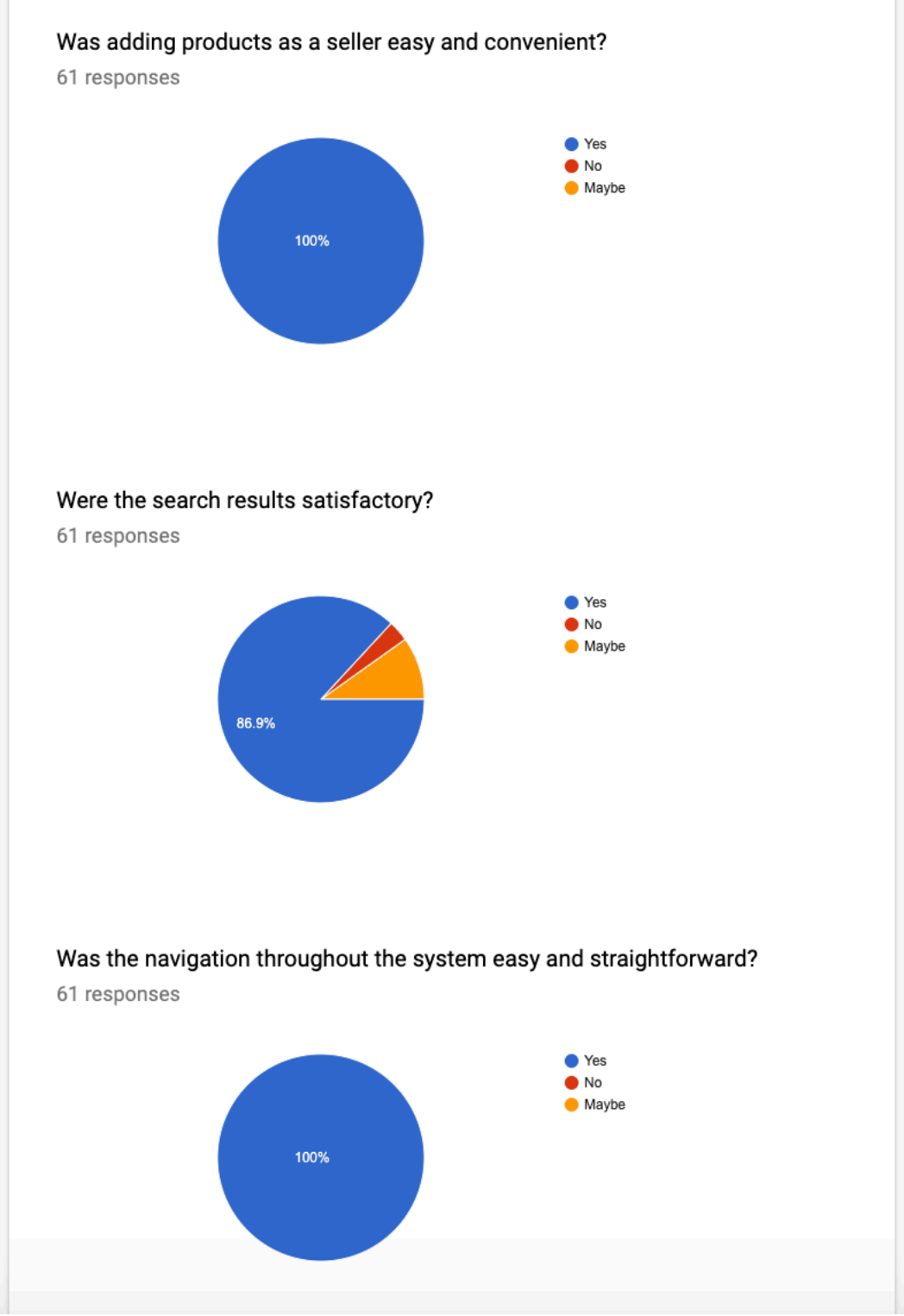


FIGURE 74 USE CASE USABILITY SCALE TESTING STATISTICS

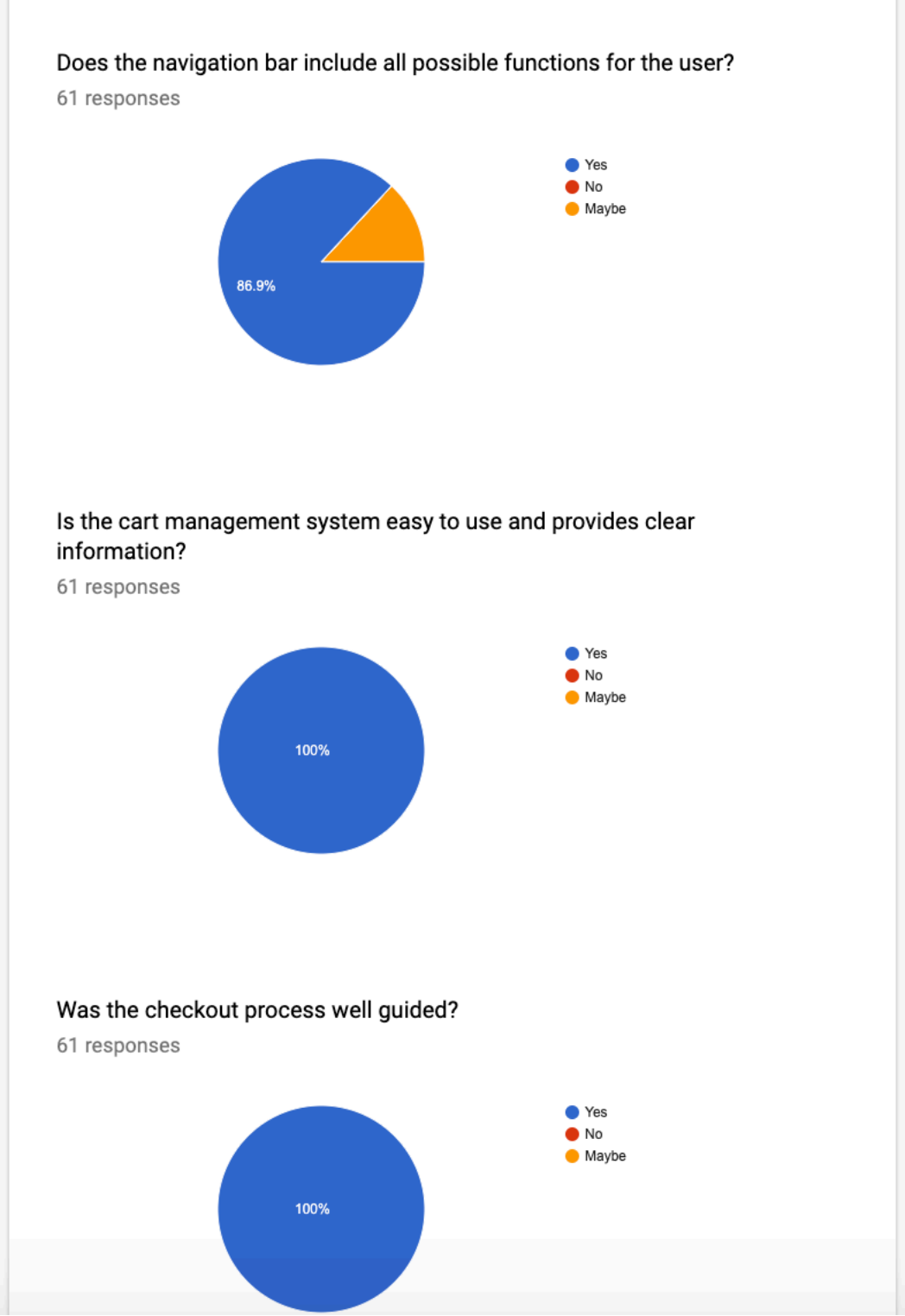


FIGURE 75 USE CASE USABILITY SCALE TESTING STATISTICS

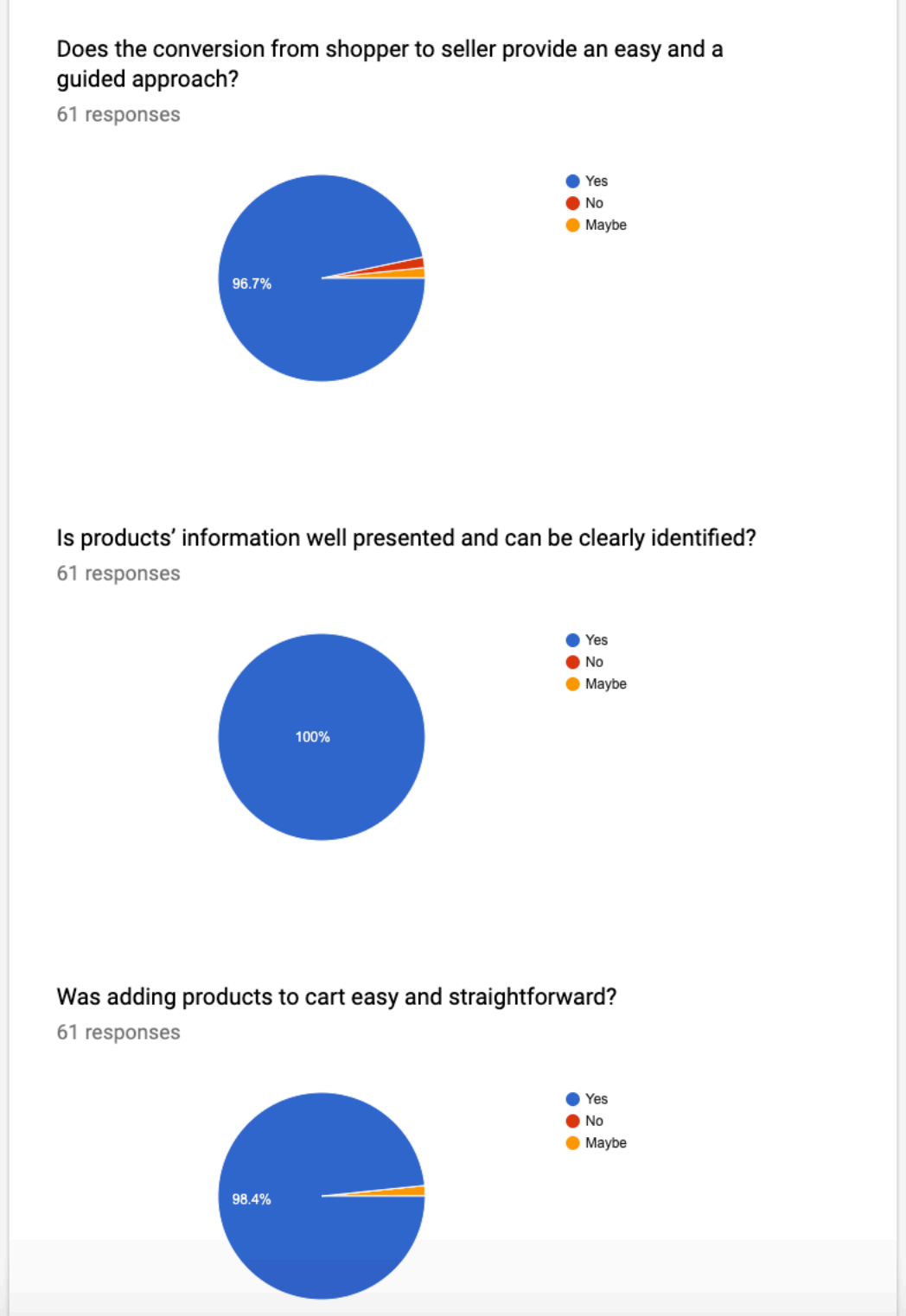


FIGURE 76 USE CASE USABILITY SCALE TESTING STATISTICS

7.5.3 FURTHER QUESTIONS

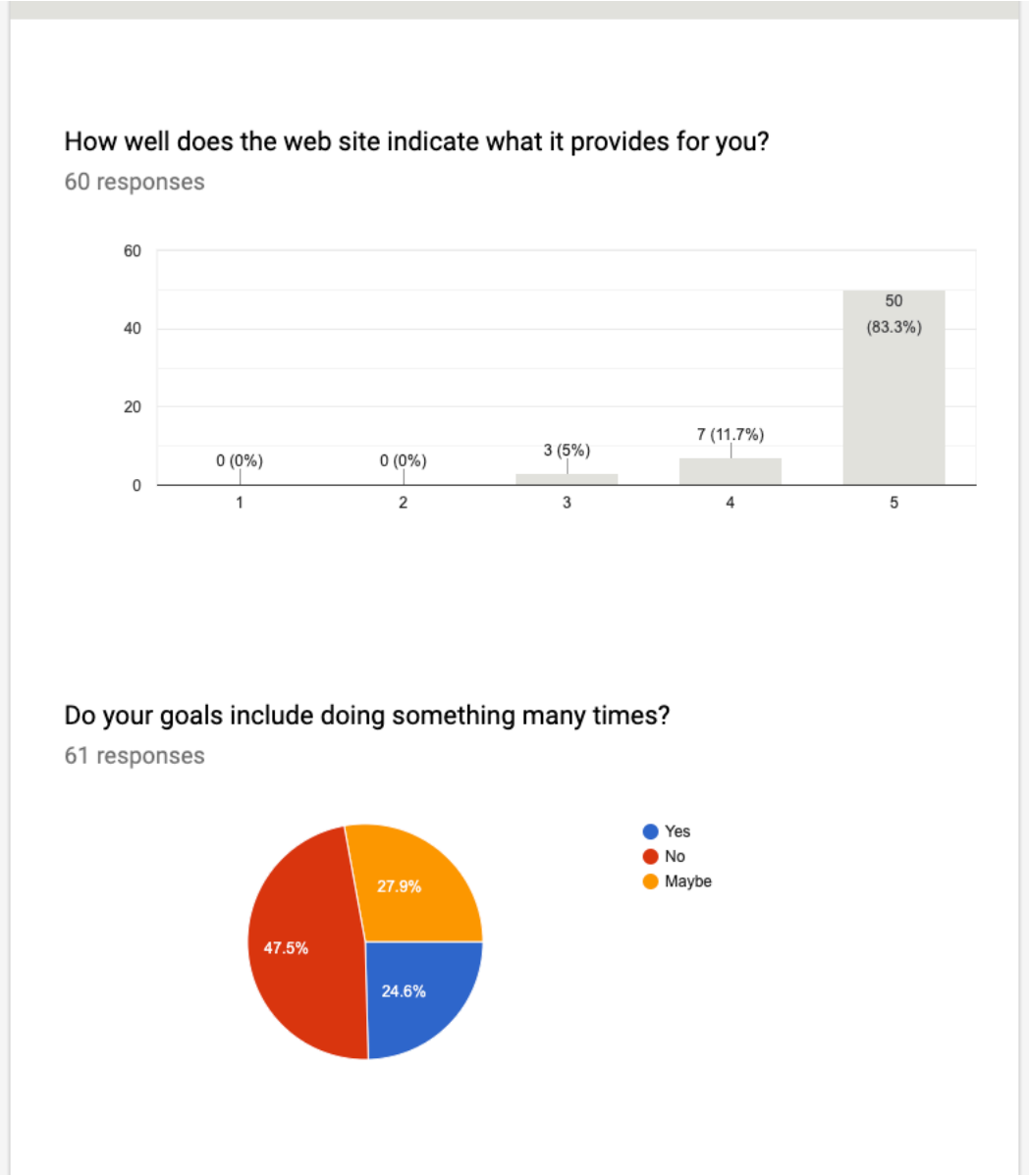
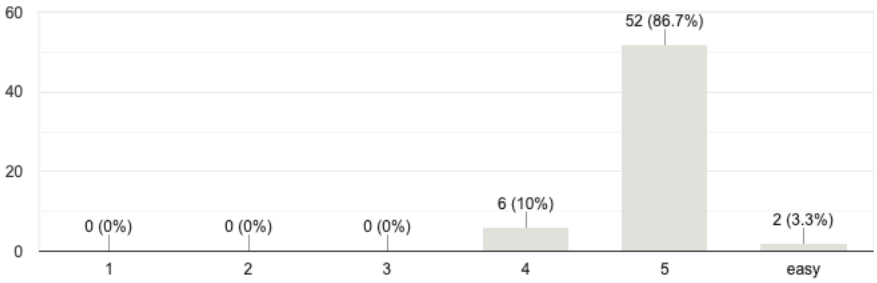


FIGURE 77 FURTHER QUESTIONS TESTING STATISTICS

How clear is the structure of the site, and the place of the current page within it? How easy is it to move to another part of the site if you're in the wrong place?

60 responses



How, and how well, does the web site enable you to find the products/functions you need?

61 responses

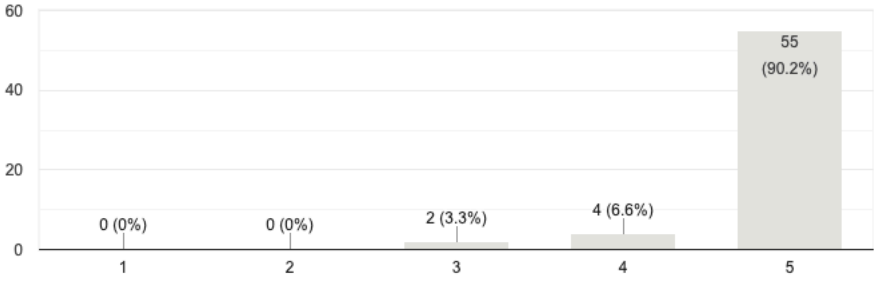
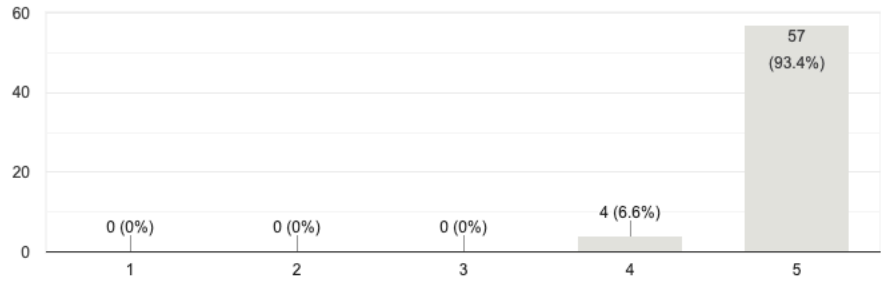


FIGURE 78 FURTHER QUESTIONS TESTING STATISTICS

How, and how well, does the layout and style signal the structure of each page and of the site?

61 responses



How clear is the sequence of pages and page elements you need to follow to achieve your aims?

61 responses

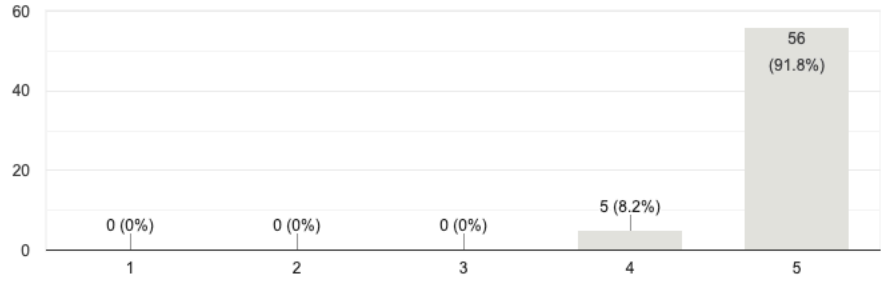
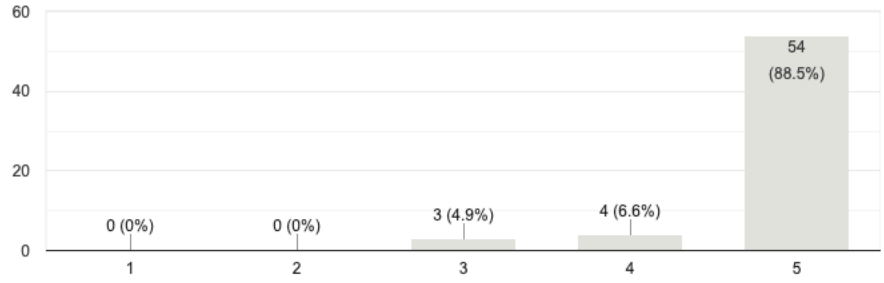


FIGURE 79 FURTHER QUESTIONS TESTING STATISTICS

How well does the web site support quick scanning to find out what the web site is about and what is and isn't there?

61 responses



How easy is it to tell that unsupported use cases are unsupported?

60 responses

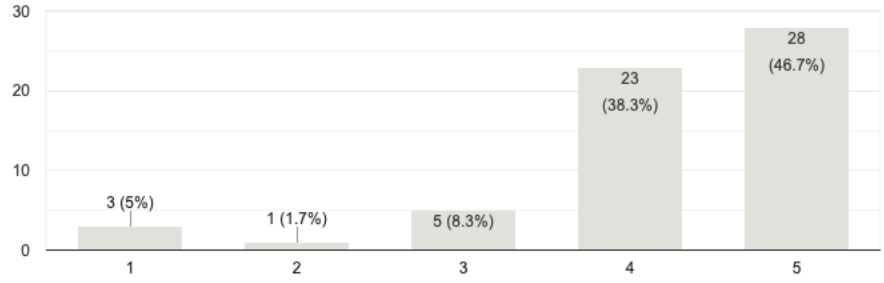
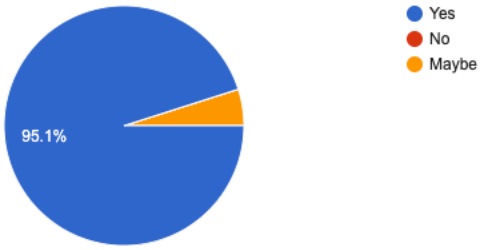


FIGURE 80 FURTHER QUESTIONS TESTING STATISTICS

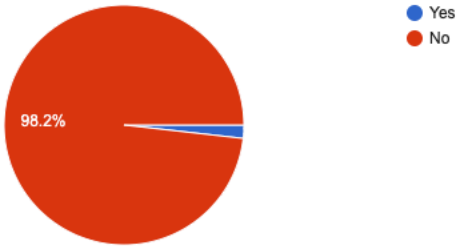
Are the contents of pages clearly indicated by their links in other pages?

61 responses



Does the web site open additional browser windows? If so, how clearly does it signal what is going on? Is opening additional windows justified?

56 responses



How up-to-date is the content? Do you have confidence that the information is current?

60 responses

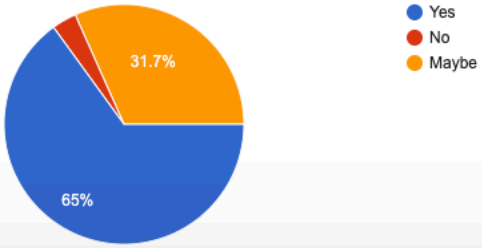
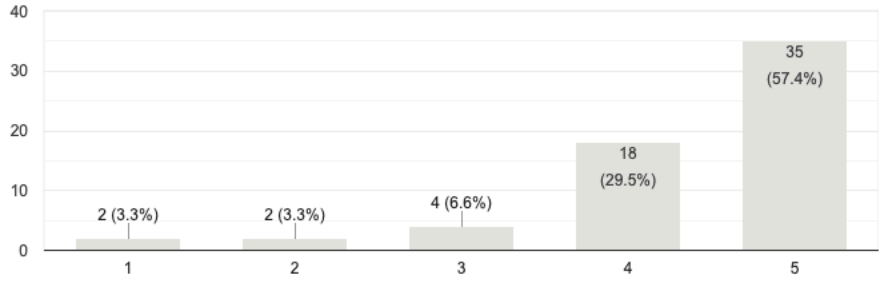


FIGURE 81 FURTHER QUESTIONS TESTING STATISTICS

How reliable is the content? If some items of information should be treated with caution, is the site clear and honest about this?

61 responses



Does the content of the web site include unwelcome content, such as advertisements in banners or pop-up browser windows? If so, how much do these obtrude?

59 responses

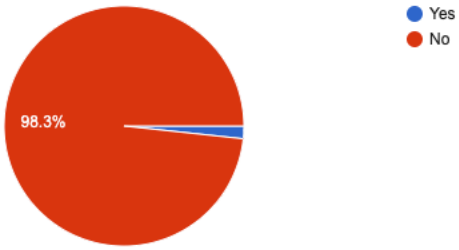


FIGURE 82 FURTHER QUESTIONS TESTING STATISTICS

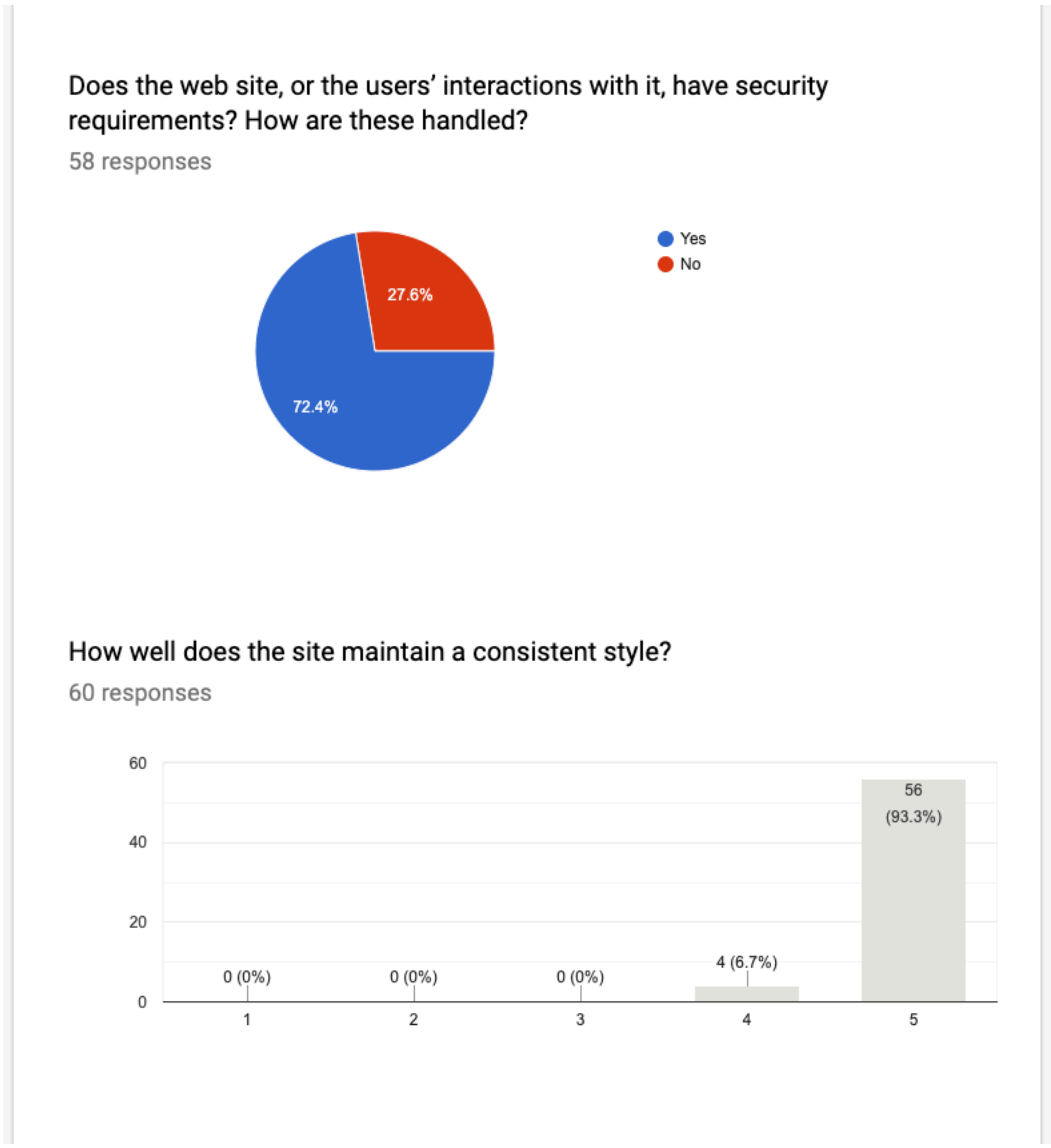


FIGURE 83 FURTHER QUESTIONS TESTING STATISTICS

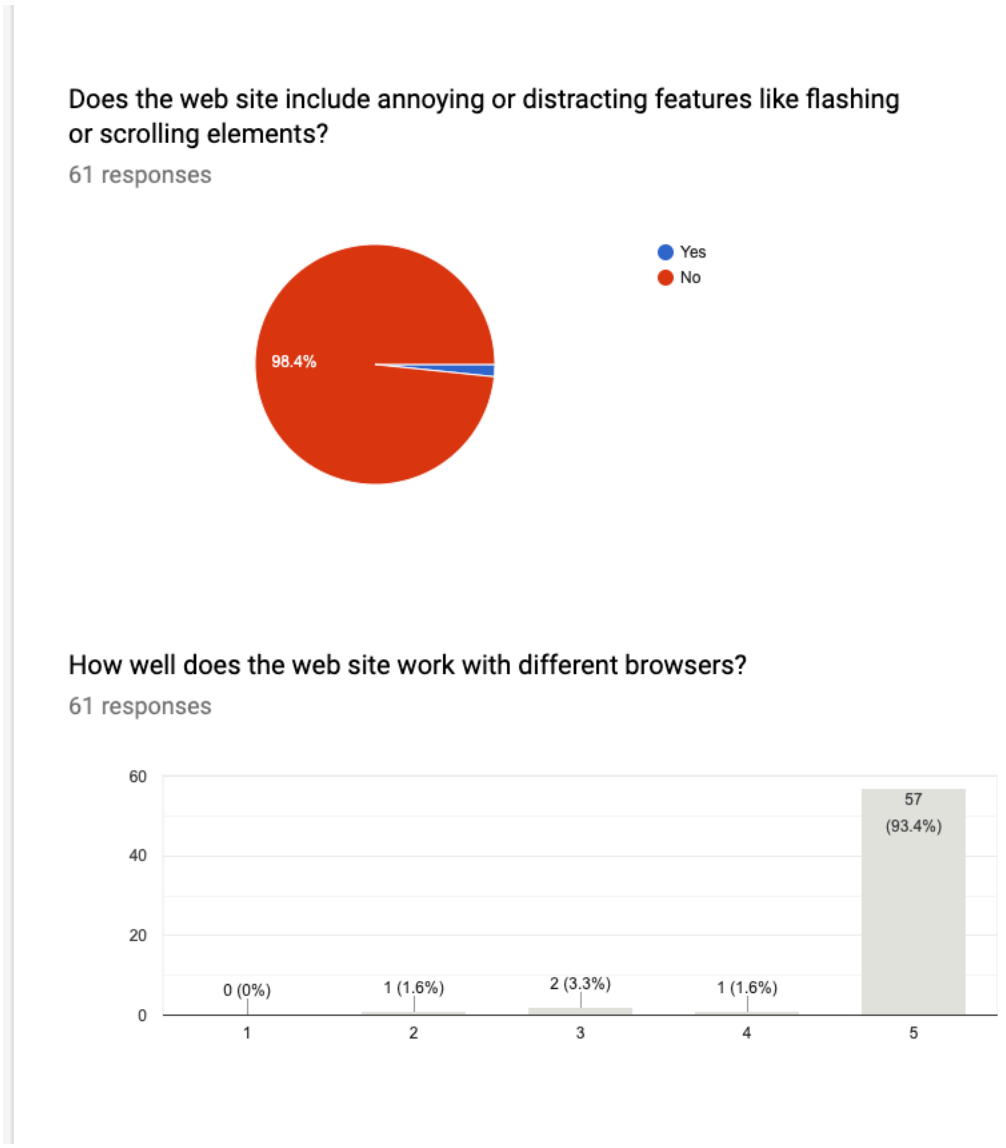
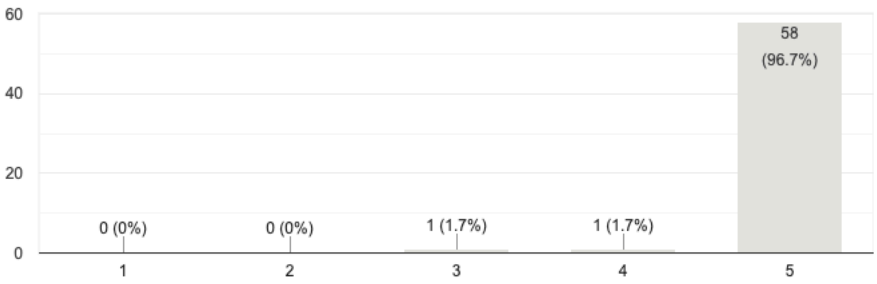


FIGURE 84 FURTHER QUESTIONS TESTING STATISTICS

How well does the web site work with different sized screens?

60 responses



Does the web site require plug-ins that many users won't have? If so, are these signaled and easy to avoid?

61 responses

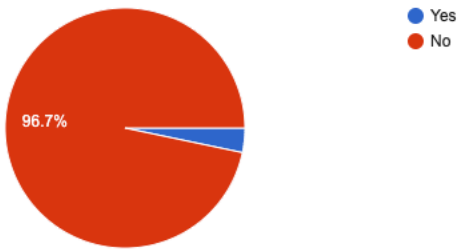


FIGURE 85 FURTHER QUESTIONS TESTING STATISTICS

8. TOOLS AND COMPONENT REUSE

During working on the project, a method to develop reusable components had to be constructed. The team has found a library that offered a framework to construct reusable template components for the user interface. Unfortunately, the library was deprecated since 2015. Team members, however, have decided to fork the library and refactor the codebase to meet the new requirements and make the work with the user interface intuitive.

Django-components (Mohammed Alhaddar, 2019), is the name of the library that the team refactored, allowing it to function again for new versions of Django and adding more features to keep up with the requirements. a new parser and compiler for the template engine had to be written to make the library work again.

The reusable UI components the team constructed are:

- Button.
- Expandable Card.
- Store and product list.
- Pagination buttons.

Each component can have separate HTML, CSS and JavaScript files, with separate attributes to allow modifications in all the user interfaces both mobile and desktop, all can be reused with one line of code, following the “don’t repeat yourself” (DRY) principle.

9. MACHINE LEARNING

Machine learning (ML) algorithms were implemented in the system to enhance the user experience while using the shopping system. In the following sections, two algorithms will be discussed. One is implemented using a recommendation model, and the other is built using the k-means clustering model. Both algorithms are aiming to bring product recommendations to the user.

9.1 COLLECTING DATA

The first step in machine learning is gathering the correct data. The team has constructed a logging system that logs the user's interactions with the website in specific views, then saving the log into the database, allowing it to be extracted later when building ML models.

The logging behavior is implemented in a middleware that every request in the system has to go through, extracting the important features to be saved, such as:

- Request Method
- Path of the request
- User ID
- Is an anonymous user
- Visited product, category, store if any
- Longitude and latitude coordinates
- City
- IP Address

The user city is determined by the IP Address, using a special lookup table in the database that contains all the IP address ranges and their information. The longitude and latitude, however, are based on two methods:

1. IP Address
2. GPS using Geolocation browser API.

Using Geolocation API, the user will receive a prompt to allow accessing location permissions from the browser. If the user agrees on sharing the location, then the exact location coordinates from the GPS will be stored in the database with any subsequent calls. Otherwise, the longitude and latitude will be determined using the IP lookup table in the database as an approximation to the exact location.

9.2 GEOHASH CLUSTERING

The first ML algorithm is based on the users' coordinates to get the popular products in the current geographic cluster. Formally, Given Longitude **X** and latitude **Y**, retrieve the top ten popular products in the current geographic cluster, popularity is based on product page visits. The log data is sufficient as a dataset to build this model.

Geohashes are used to divide the entire world map into regions/clusters. it encodes geographic regions into short strings consisting of both alphabets and numbers, ranging between length of 1 up to 12 characters. Each level consists of a 4*8 grid, totaling 32 clusters per level (*Figure 27*). For example, given a string with a single character "t", this will specify the entire cluster marked with letter t in *Figure 27.* Implying that the longer the string, the more exact the location will be.

The team has decided to split the entire world map into clusters of area that is equal to 25 KM^2. To reach this precision in geohashs, a string of 5 characters is used as in *Figure 28*.

Given that the logs are storing the exact longitude and latitude, special libraries were used to convert the given coordinates and get the 5^{th} level of the geohash cluster. After that, a process to generate the model will be executed as follows:

1. Extract the system logs.
2. Build a list of clusters that has active users.
3. Get product-specific logs.
4. Build a pivot table concerning clusters and all the logged products in the system. The cells will contain the number of times product X has been visited in cluster Y. (See Table below)

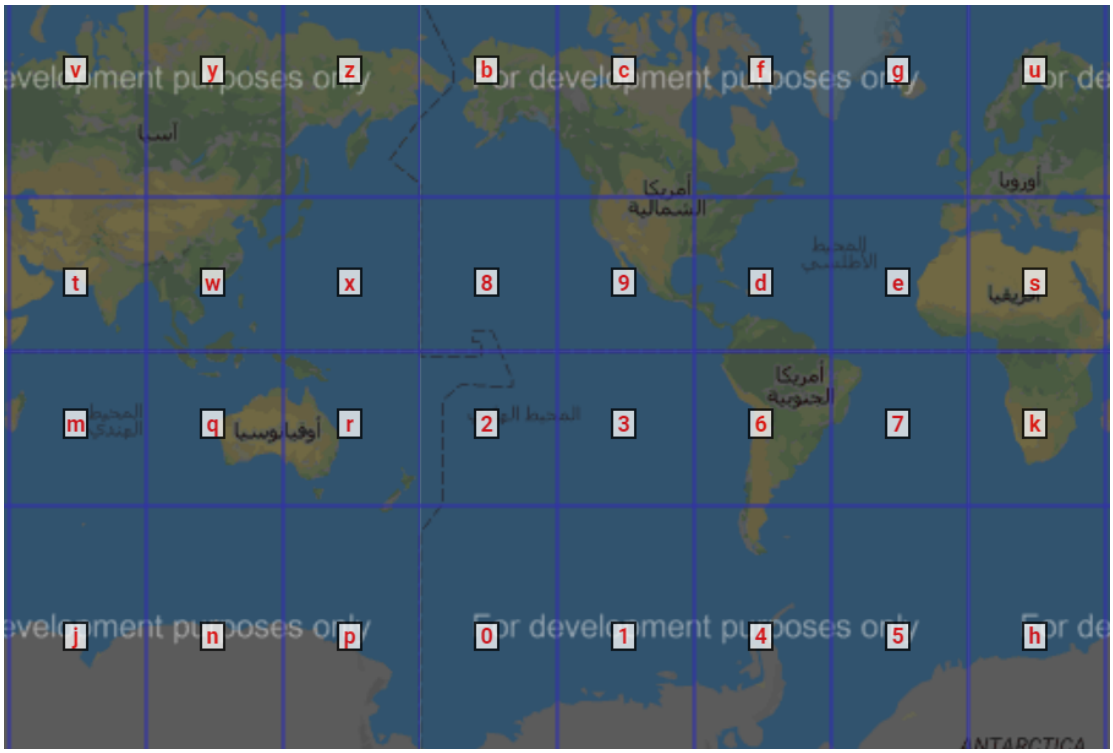


FIGURE 86 FIRST LEVEL OF GEOHASH

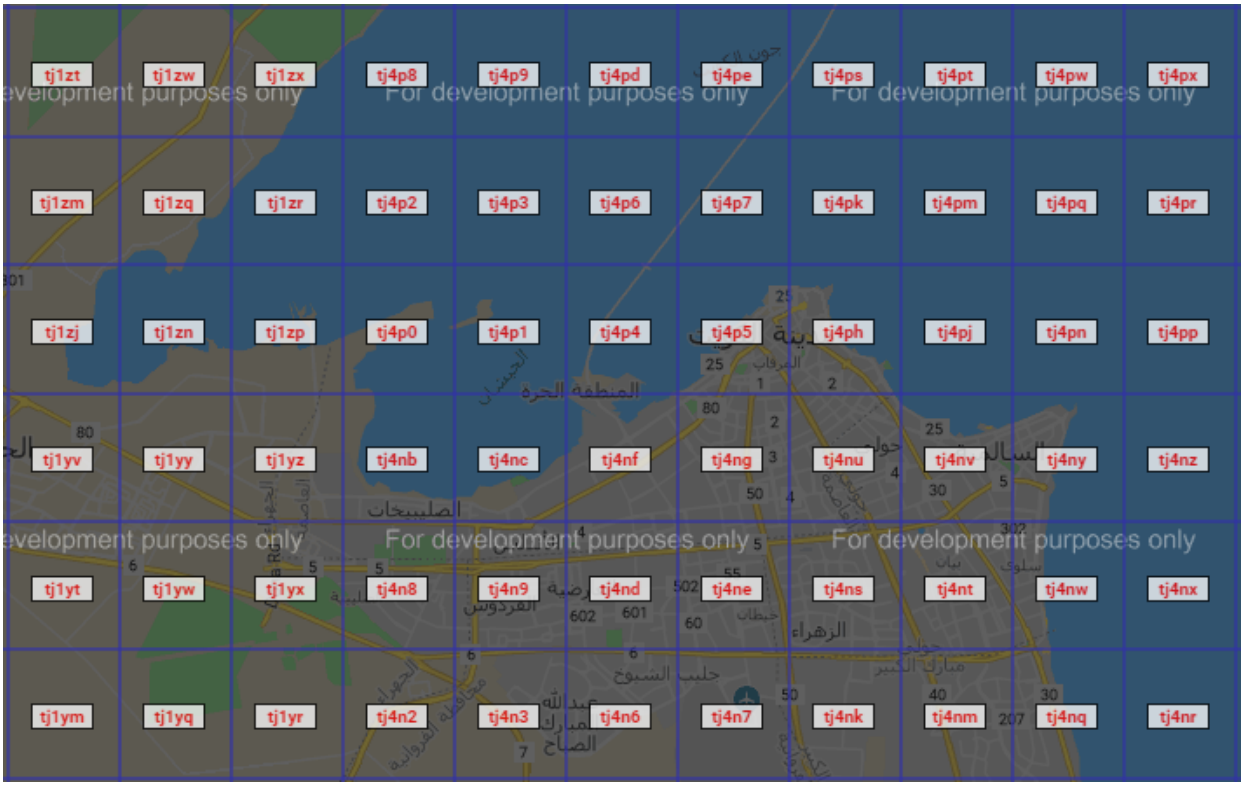


FIGURE 87 5TH LEVEL OF GEOHASH

5. Normalize the cells
6. Rank the products in each cluster based on popularity (visit count)
7. For each region, Insert top 10 product recommendations inside the region model.

visited_product	10	11	27	30	44	63	...	12890	12891	12892	12893	12894	12887414
geohash							...						
109	NaN	NaN	0.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
110	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.5	0.227273	NaN	NaN
111	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
112	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.5	0.227273	NaN	NaN
113	NaN	NaN	NaN	NaN	NaN	NaN	...	0.272727	0.636364	0.5	0.227273	0.181818	0.0
114	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
115	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.636364	0.5	0.227273	NaN	NaN
116	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
117	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.5	NaN	NaN	NaN
118	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
119	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.5	NaN	NaN	NaN
120	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
121	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.5	NaN	NaN	NaN
122	NaN	NaN	NaN	NaN	NaN	NaN	...	0.272727	NaN	NaN	NaN	NaN	NaN
123	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
124	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	0.181818	NaN
125	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
126	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
127	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
128	0.0	NaN	NaN	NaN	NaN	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN
129	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
130	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
131	NaN	NaN	NaN	NaN	0.090909	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
132	NaN	0.045455	NaN	0.045455	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
133	NaN	NaN	NaN	NaN	NaN	NaN	...	0.272727	NaN	0.5	0.227273	NaN	NaN
134	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
135	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN

FIGURE 88 GENERATED PIVOT TABLE

Once the data is populated, the system will try to get the location and find the top products that are in the cluster populated by the recommendation model mentioned above.

To test the model, random product visits with random users are simulated. The coordinates of the random visits were scattered uniformly around Kuwait City and nearby governorates. A script was written to generate a heat map from the logs, to observe the user's activity from the log data. Figure *30* shows a heat map of the generated product visits by the simulation.

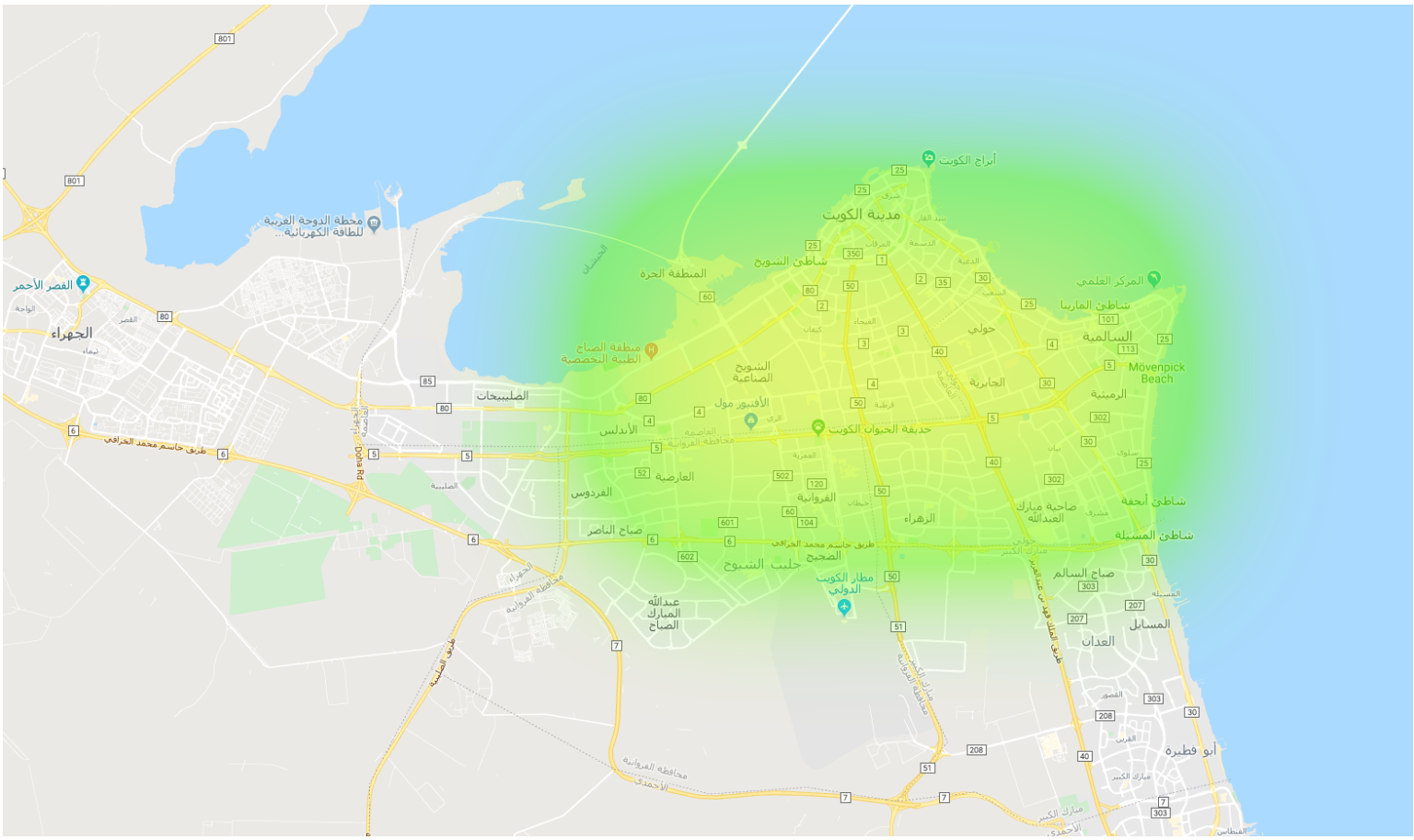


FIGURE 89 HEATMAP OF THE LOG ACTIVITY

The users' location is faked to a random point inside the heat map, giving different recommendations based on the user's location:

Longitude, Latitude	**Geohash**	**Area**	**Recommended Product IDs'**
29.332326,48.065551	tj4nv	Salmiya	12873, 12615, 7599, 8811, 9120, 9531, 12340, 6682, 7909, 7916
29.297987,48.043971	tj4nt	Bayan	12892, 6668, 12615, 12874, 6396, 7599, 8811, 9531, 6682, 7064
29.281205, 48.004221	tj4ns	Alzahra	6396, 7270, 7599, 9531, 12340, 12728, 7909, 9054, 9356, 12892
29.368715, 47.991339	tj4ph	Kuwait City	6396, 7270, 9120, 9531, 7909, 7916, 10579, 11216, 11389, 12114

9.3 K-MEANS

K-means (geeksforgeeks, 2019) as a machine learning algorithm relies on clustering similar data units together and provides a prediction methodology that when given a data unit returns the proper cluster the data most likely belongs to. The project used K-means clustering to cluster products as similar products that have the same name/description in clusters and used these clusters to predict and recommend products for a certain user depending on his/her product browsing history. That being said, any user who uses the system, the system collects browsing history for that user to gather information about what products the user is interested in, then these products are fed to the algorithm which recommends other products that are similar in name and description to the ones that the user already saw. Doing this enables the algorithm to produce more appropriate results and give better recommendations to each user separately. Moreover, the algorithm also ranks the browsed products by the user using a scoring technique that ranks products depending on how many times the user saw, thus the products that score the most are the ones that are fed to the algorithm and recommendations are proposed according to them.

After running the K-means algorithm and getting clusters as a result, users' profiles are created which hold the most common words extracted from the most products a user is interested in. These words are then used against the clusters to extract products that are similar or include the words the user is interested in and the result will be displayed to the user as recommended products.

The algorithm works following these steps in order:

1. Read and normalize all the logs in the system, thus get the users browsing history.

   This step enables the products to be discovered and be fed to the Kmeans algorithm.

2. Read and normalize all product data in the system to be then fed to the algorithm when the users' most popular products are determined.
3. Count the logs (browsing history) of each user in the system and eliminate users who have less than 5 logs.

This step ensures that the algorithm has enough data about each user to parse and improves the quality or recommended products.

4. Calculate the popularity of the product for the users who passed step 3.

This step assigns a score number to the products which are most visited in the system to be later used in general recommendations based on the geolocation of a user.

5. Normalize the products' names and descriptions to be fed to the algorithm.

This step ensures that products are converted to values that can be fed to the algorithm.

6. Create user profiles for all the users who passed step 3.

These profiles are then used to predict products a user is most interested in.

7. Cluster products and predict users' interests by using their profiles.

This is the final step that gets the actual results of the algorithm were each user profile is used to calculate the most popular words (taken from browsed products names and descriptions) a user is interested in and are given to the K-means clusters to predict similar products a user may also be interested in.

The following diagrams show how the results of the algorithm are presented to the user. When a user tries to see any product details, he/she will be offered product recommendations based on the past products the user already saw, this ensures that the user style and type in products get improved whenever a user clicks on a recommended product. Collecting data about a user browsing history in the system ensures that the future results for the user are going to be more accurate as more data about the user is going to be included in his/her profile that will produce more accurate recommendations for the specific user. Moreover, as can be clearly seen throughout the diagram, a subset of the recommended products for a user is shown, while the rest of the set is kept to be recommended in other places to avoid the static look and feel of the recommendation system and to give the user a broader set of products to choose from. The bellow diagram was taken from a test user that is interested in boy's clothing; it can be clearly seen that all the recommended products are for boy's clothing as all the user's browsing history is focused on this type of product. Multiple refreshes/products pages offer the same type of recommended product as well. This proves that the algorithm is working as expected and can be tested with almost no effort using a dummy user.

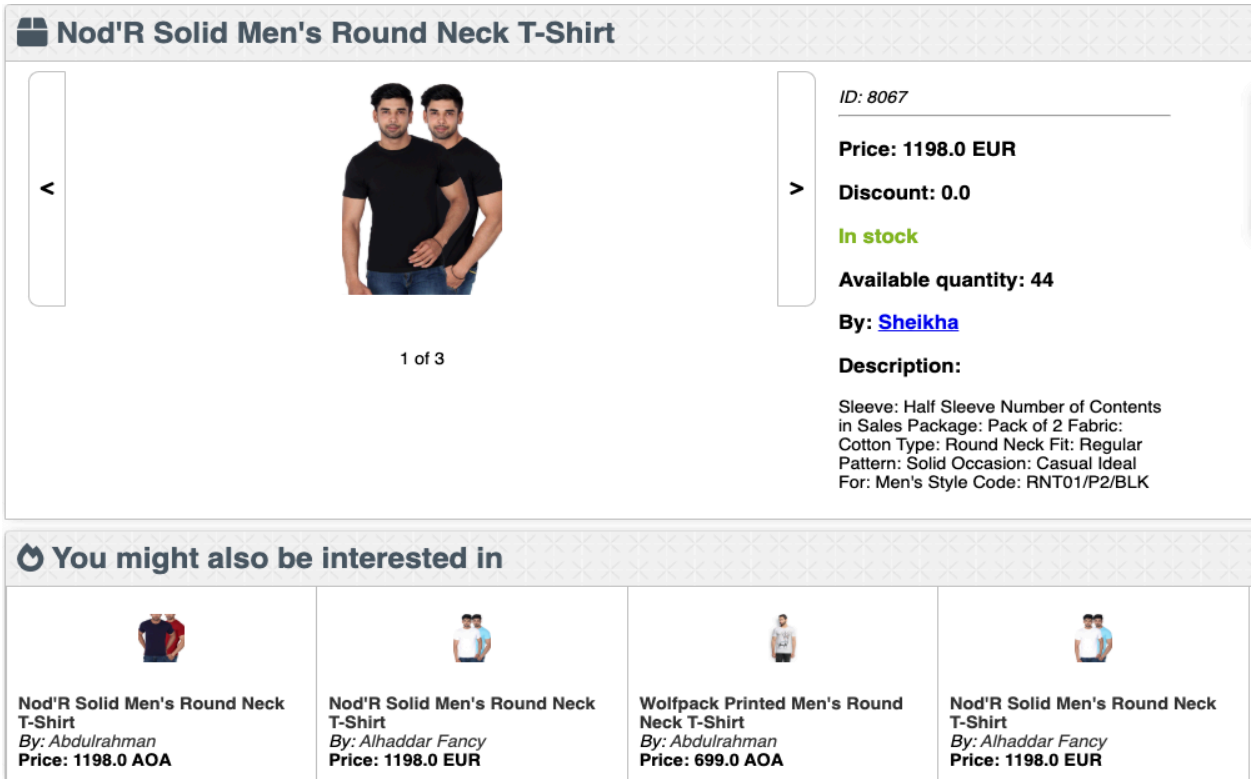


FIGURE 91 ANOTHER PRODUCTS RECOMMENDATIONS FOR THE SAME USER.

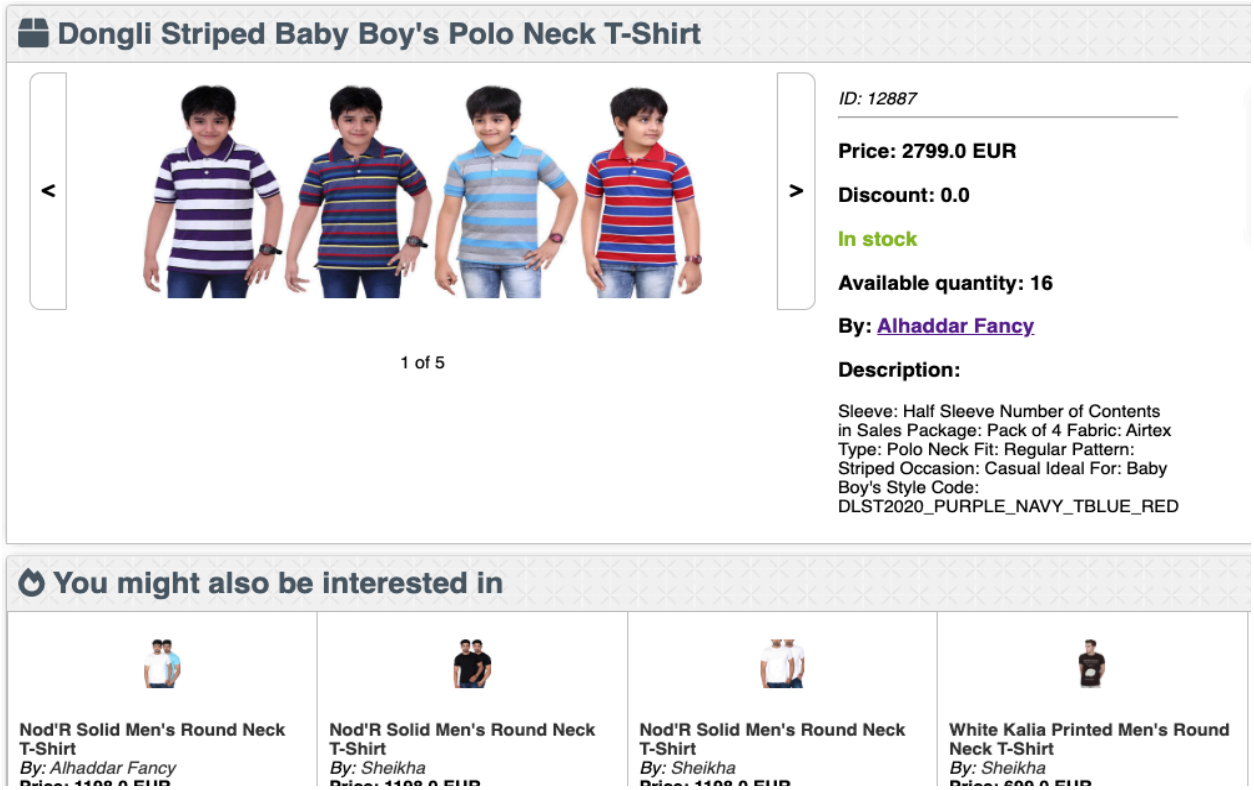


FIGURE 90 PRODUCTS RECOMMENDATION INSIDE PRODUCT DETAIL VIEW.

9.4 MORE ON MACHINE LEARNING

As a bonus on the machine learning using the K-means clustering and grouping products by scores they make given how many times each product is visited, it was reasonable and almost proofed that users who belong to a geo-region could be interested in the same set of products, thus the following was done. When collecting the logs of the users' browsing history on the website, and when each product gets its score as the number of visits, sort the products on the store in descending order to get the most popular products in the system. After getting the set of the most popular products in the system, this set is saved and is displayed to users depending on their location. For example, is product X scored the highest score given the region is Y, then each user logs into the system from region Y will be offered product X as a recommendation. As can be clearly seen that X and Y are sets of products and users respectively, thus it can make sense to offer users from the same region the most popular products in their region. However, this algorithm cannot be classified as a machine learning algorithm in any means but doing so proved a success and it adds more dynamic behavior to the website. Moreover, these products are only recommended to the users on the homepage of the website, which means they will not be recommended to users specifically and will not be shown as similar products in any means.

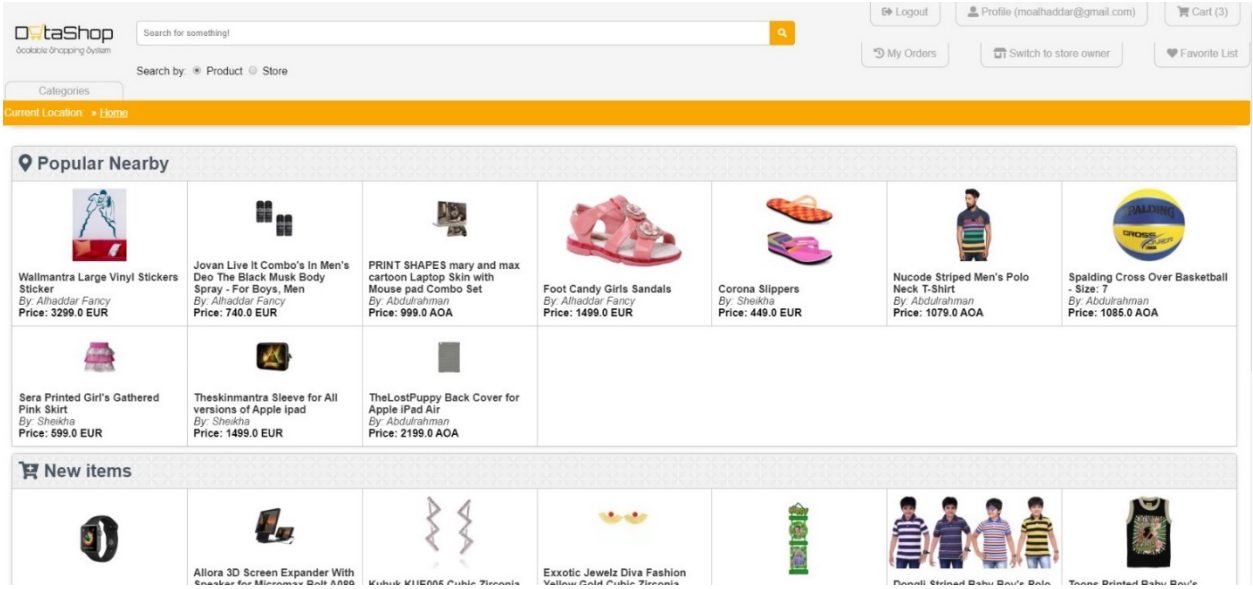


FIGURE 92 GENERAL RECOMMENDATIONS BASED ON GEOLOCATION.

10. CONCLUSIONS AND LESSONS LEARNED

To conclude, Datashop offered E-commerce services alongside other features such as scalability, reusability, extensibility, and machine learning techniques to enrich the user experience in the system. The system has implemented many requirements and features to allow users to enjoy using it while the architecture of the system was designed to allow for more requirements and features to be implemented. Many design decisions were made to drive the system into the reusability field and enable the system to be more reusable and extensible. The system is based on a website that offers an interface for all types of users both using a PC or a Mobile phone to use and interact with the system. The system is mainly built using Python programming language and it uses several frontend programming languages and techniques to deliver the desired user experience. Many tests were done on the system to verify and validate it, and full documentation was presented in this report.

Many technical lessons were learned, but all pour into the work on the design of web apps and how to manage risks and challenges. The team has learned to manage time effectively, prepare plans for work, and make communication shorter and more informative. The team has also learned to control and prepare for challenges that may stop the development of the system and think deeply to avoid or eliminate risks which may arise from design decisions that were, or about to be taken. One of the vital lessons learned from working on such a project is that nothing comes easy, and quick solutions may prove to be invalid sooner than expected, and detailed elaboration and search methodology have to be developed to find the optimal solutions for each problem faced.

11. TEAM MEMBERS TASKS AND CONTRIBUTIONS

Team members worked equally and effectively to make this project alive, this includes working on all the planning tasks, programming tasks, and testing tasks. It is hard to measure each member's contribution as tasks were divided according to experience and *bravery to handle a new task,* while the whole team participated in solving problems that a team member is facing. However, some tasks were assigned to a single team member due to his/her knowledge in the field, while the other members kept their focus on other tasks, as for example, Mohammad Alhaddar took the lead in dealing with AWS and designing the system architecture, while Sheikha took the most planning parts and interactions with outer personas, and finally Abdulrahman was focused on creating views and user experience. A detailed percentage cannot be determined well, but for approximation, the team agreed that Alhaddar has done 40% of the work, while both Abdulrahman and Sheikha both did 30% of the total work. Moreover, the team members also overlapped each other's tasks either offering refinement or fixing bugs, thus as for the team opinion the project is one unit that was created equally by the whole team members.

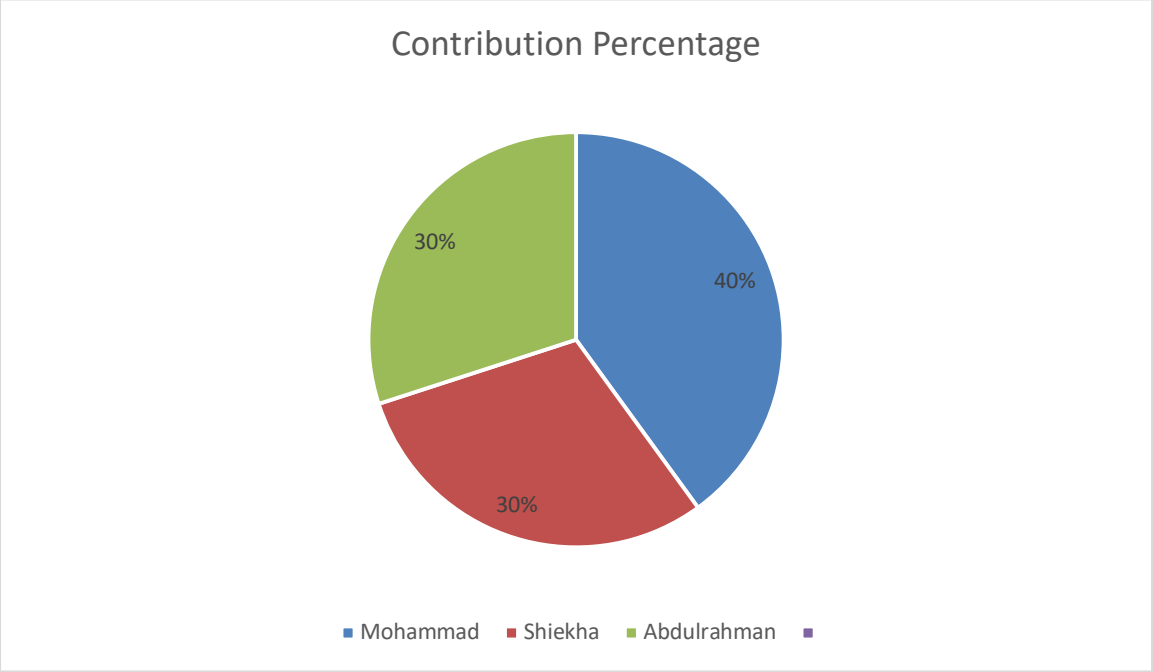


FIGURE 93 TEAM CONTRIBUTIONS CHART.

REFERENCES

AtlassianTeam. (2019, 11 25). *What is Scrum?* . Retrieved from Atlassian: https://www.atlassian.com/agile/scrum

Bhalla, P. (2019, 8 2). *What is eCommerce Business and How does it Work*. Retrieved from shiprocket: https://www.shiprocket.in/blog/what-is-ecommerce-how-it-operates/

geeksforgeeks. (2019, 11 25). *K means Clustering*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/k-means-clustering-introduction/

Linkeit-Blog. (2019). *What is Amazon Route 53?* Retrieved from Linkeit Blog: https://www.linkeit.com/blog/what-is-amazon-route-53

Mohammed Alhaddar, S. A. (2019, 11 1). *Django Components*. Retrieved from Github: https://github.com/mohammedalhaddar/django-components

postgreSQLTutorial. (2019). *What is PostgreSQL?* Retrieved from postgresqltutorial: http://www.postgresqltutorial.com/what-is-postgresql/

Rouse, M. (2014, 4). *Amazon Machine Image (AMI)* . Retrieved from SearchAws: https://searchaws.techtarget.com/definition/Amazon-Machine-Image-AMI

Rouse, M. (2018, 11). *Amazon S3* . Retrieved from SearchAWS: https://searchaws.techtarget.com/definition/Amazon-Simple-Storage-Service-Amazon-S3

ScrumTeam. (2019, 11 25). *What is Scrum?* . Retrieved from Scrum: https://www.scrum.org/resources/what-is-scrum

SumoLogic-Team. (2019). *What is Amazon RDS?* Retrieved from Sumo Logic: https://www.sumologic.com/insight/what-is-amazon-rds/

vsupalov. (2019). *vsupalov*. Retrieved from What Is Gunicorn, and What Does It Do?: https://vsupalov.com/what-is-gunicorn/

Yadav, K. (2019, 10 14). *Understanding Amazon EC2 Terminology*. Retrieved from HackerNoon: https://hackernoon.com/understanding-amazon-ec2-terminology-85be19d0af28

Yadav, K. (2019, 10 14). *What is Amazon Elastic Load Balancer (ELB)*. Retrieved from HackerNoon: https://hackernoon.com/what-is-amazon-elastic-load-balancer-elb-16cdcedbd485