CS6P05 Final Year Project Computing

Final Report

Samanghar – An auction and shop based Ecommerce solution

Student Name: Kushal Bhattarai

London Met ID: 17031137

College ID: NP01CP4A170221

Internal Supervisor: Roshan Tandukar

External Supervisor: Amulya Lohani

Due Date: 5th June 2020

Submission Date: 5th June 2020

Word Count: 8083

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and marks of zero

SAMANGHAR – AN AUCTION AND SHOP BASED ECOMMERCE SOLUTION

Abstract

In the introduction of topic section the topic of eCommerce was discussed , problems were stated and solution were described as well as current scenario and motivation for the project to be done was included . In Background part of the report includes the background information about the project like what is it how it works and how was the logic and architecture how it is being made is described. The background portion contains review of similar systems, survey review. The development part includes the process of choosing methodology. Three methodologies were considered and one of them was selected and described how and why it was selected. In the Progress section the overall progress was explained. Use case, wireframes, ER as well as activity diagram was include in the progress section. In Analysis of Progress, the progress of the project was analysed and a new plan was considered to continue the project. In Further Work what is to be done was discussed and reported. In Appendix portion, the some contents of proposal are included like, previous plan and project Gantt chart. Milestones of the project Risks and contingency plans were included.

Acknowledgment

I would like to express their sincere gratitude to our module teachers Mr. Roshan Tandukar and Mr. Amulya Lohani supporting and guiding the outcome of this coursework. The tasks on hand were challenging and often confusing but due to their insightful narrative of the scenario, I have been able to accomplish our final year project. Their opinions and constructive criticism were very valuable to do all the tasks successfully. Furthermore, I am thankful and fortunate enough to get constant encouragement, support, and guidance from all the staff and teachers who took the time to review our work and provide us with their genuine suggestions.

Table of Contents

Table of figures

Table of tables

START OF THE REPORT

CHAPTER 1: INTRODUCTION

1.1 PROJECT DESCRIPTION

The expectation is to develop a website based on the process of the C2C model. My website created will have a connection between the two users through a site where the details are present on the website, and the communication of users is conducted. The login system for all types of users will be created. A proper Dashboard for admins for the customization of different products and managing the users will be provided. The users can buy and sell items through the bidding system. A proper communication system for the buyer and seller will be implemented. Full Transparency will be provided about the bidders, seller's products and final buyer. History of the products sold and bought products can be viewed. Messaging system for the Users when the auction has started or any items that have been added to the selling section of the website is a feature that will be present on the site. The seller will provide all the required details about the products and will upload a video or picture on the website. The Website will have a proper checkout system where the winning bidder can check out. A different section where the products can be bought directly from a seller without bidding and the seller has agreed not to sell the item through a bidding system. The registered users can bid on the item and compete with other bidders. The payment system can be cash upon delivery.

1.2 CURRENT SCENARIO

The e-commerce market in Nepal is worth around $25 million. The market is seeing 300 percent growth annually. Customers are trusting e-commerce more, and sellers are seeing the value of e-commerce. **(The Kathmandu Post, 2020)**

As of May 2019, there were 31 private ISPs in Nepal, with about 200,000 subscribers, and nearly 16.67 million internet users nationwide that are based on E-commerce. Roughly 40 percent of these accounts are commercial, with businesses promoting their products and services and communicating with foreign companies via the internet. Ecommerce is still in its infancy in Nepal. **(export.gov, 2019)** The country's challenging terrain and lack of street addresses make deliveries a challenge. My project is based on Consumer to consumer deals with used things or products. Here a consumer can post his products, goods, services and another consumer, who is interested, can purchase it by using this E-commerce system with the addition of bidding. This web technology helps us to sell our used properties or assets like bike, car, house, any electronic items or kitchen items by using this model as well as new products. Hamrobazar, Sastoramro, and many other sites can provide you this service, but they do not have to buy now feature and bidding features. In those websites only the information is provided, and the users must communicate with their phone and make specific deals. I want to create a bidding system where the users can bid, and the seller can accept the bid and the potential buyers can not only have information about the products but can buy them with a proper payment system. People use social media sites like Facebook and Instagram to sell their products through Facebook groups and Instagram stories. The problem is that people can‟t know who to trust and will the person be able to give money. "A recent survey conducted by LocalCircles shows that 38 percent of respondents out of 6,923 have been sold counterfeit products from an eCommerce site in the past year. Among the leading eCommerce sites, 12 percent of the respondents said they received fake products." **(Devansh Sharma, 2018)** There will be spammers who would spam the owner of the product, with that problem in mind I want to create a website where people can display the information of products and place an offer for the product owner.

Industry insiders and government officials said that the e-commerce market would grow manifold due to a swelling middle class and a surge in the number of internet users. The Nepal Telecommunications Authority claimed that internet penetration in Nepal had reached 63 percent of the total population as of 2017 following a smartphone boom. Another reason behind the burgeoning e-commerce market is the large number of Nepalis living abroad—for work, study and business—where they gain exposure to international brands. When they return home, they prefer to purchase international products online. Toya Narayan Gyawali, joint secretary of the Ministry of Industry, Commerce and Supplies, told the Post that the draft regulations would be given final shape within a month. "After the draft is finalized, it will be submitted to the Cabinet for its approval to implement it." The absence of regulations has led to an increasing number of consumer complaints. One of the key features of the regulation would be consumer protection. E-tailers and marketplaces will be made liable for fraudulent sales and substandard or defective products, Gyawali said. Besides checking fraud and business misconduct, the regulations will properly manage electronic transactions, regulate foreign currency transactions and promote Nepali products through online stores on the global market, said Gyawali. There are no exact statistics of the size of the e-commerce business in Nepal, but market insiders estimate annual turnover to be in the neighbourhood of $25 million. They said that the number of online buyers had been increasing rapidly in recent years, particularly in the Kathmandu Valley, as they can buy a wide range of national and international products with a few clicks of the mouse. Normally, it's a cash on delivery business model that reduces hassles and saves time for customers. The government doesn't have the exact date for online marketplaces. According to a government official, there are a large number of unregistered online shops. Daraz, Sastodeal, eSewaPasal, Metro Tarkari, Bhatbhateni Online, Mero Kirana, Foodmario, Foodmandu and Urban Girl are some of the popular online stores in the country. Several rounds of discussions have been conducted with the private sector to prepare the draft, said Gyawali. The proposed regulation focuses on promoting e-commerce in Nepal by easing the process of doing online business. "Our neighbours India and China have made great strides in the development of e-commerce, and Nepal also needs to move ahead by improving services," he said. India's e-commerce market is expected to grow more than fourfold to $150 billion by 2022, fuelled by rising incomes and a surge in internet users, according to a report by software industry lobby group

Nasscom and consulting firm PwC India. The e-commerce market was valued at $36 billion in 2017. China's online retail market is expected to hit $1.8 trillion in 2022, buoyed by local tech titans Alibaba and JD.com, according to a report by Forrester. "E-commerce has massive growth potential, so proper regulations are necessary for the development of this sector," said Gyawali. Nepal lags behind in e-commerce because of a poor information and technology infrastructure, he said. "With regard to IT friendly people, Nepal falls in the average category among the least developed countries." https://kathmandupost.com/money/2019/03/06/government-to-introduce-e-commerce-regulations

1.3 PROBLEM DOMAIN AND PROJECT AS A SOLUTION

The Ecommerce website is pretty much everywhere in Nepal but has not been too much profitable as well as the auction system in these platforms have not been implemented. A proper auction website has not been available in Nepal in present context. A competitive e-commerce platform between the users is not available in the country. In present eCommerce websites sometimes only information is provided to customer. The users must communicate with their phones and make specific deals according to the information on the website. I wanted to create a website where the user doesn't have to communicate through their phone but can communicate from my website and buy and bid in the products. I want to create a bidding system where the users can bid, and the seller can accept the bid and the potential buyers can not only have information about the products but can buy them with proper payment system. Even though people use Ecommerce system but have People still use social media sites like Facebook and Instagram to sell their products through Facebook groups and Instagram stories.

The following are some of the disadvantages of an existing auction system

- The traditional method is a time-consuming process
- Date and time play an important role, as they operate for a few hours only.
- There is no separate module for sellers to upload their own Product in an online auction system.

Auctions are used to sell many things in addition to antiques and art. All-round the world there are auctions of commodities such as cattle, racehorses and just above anything elsewhere, there's a market of multiple people interested in buying the same thing that's the key to auction-a bunch of people who are interested in buying the same object and taking turns offering bids on the object. The right to buy that object will go to the highest bidder; It is called traditional auction

While the update of the traditional auction is online auction, companies from various industries are moving to online auction say eBay, on sale provided a worldwide platform for bidder by getting it to masses. In these websites they create their own auction by adding their own product to auction. The buyer can bid the product which he/she can win the product if he applies the winning bid. In my system Date and time will also play an important role, but the auction can be fast slow according to the user. My system has a separate module for sellers to upload their own Product in an online auction system.

1.4 AIM AND OBJECTIVES

The project aims to create a C2C eCommerce creating an opportunity for people to sell, buy products with auction like bidding system implemented to the website. The project aims to sell some rare products quickly. The project's objective is to create shorter and lessened channels of distribution system which reduces operating costs in the project with cheap and quicker communication between buyers and sellers. The project's objective to target is to reach relevant markets.

1.5 STRUCTURE OF THE REPORT

1.5.1 BACKGROUND

This part contains the background portion of the system.

1.5.2 DEVELOPMENT

This part contains methodology, implementation, and screenshots of the system.

1.5.3 TESTING AND ANALYSIS

This part contains testing and critical analysis of the test.

1.5.4 CONCLUSION

This part contains the conclusion of the report advantage, limitations, issues and future word is discussed.

CHAPTER 2: BACKGROUND

2.1 ABOUT THE END USERS

There are there types of users in the system. Superadmin, Admin and User.

Guest User:

The type of users who don't have account and can only view the products and search the products but cannot add products to cart or place a bid on any auction.

User:

The type of user who has a account where they can buy , bid, add to favourite, watch history about watched products, search and add products .

Admin:

The type of users who can view the analytics portion of the system like, who manages disputes and check the order, and can view the reports of the type of users, products added, orders but cannot bid on a project and cannot buy a products. The admin user has the capability to list the products.

SuperAdmin:

The type of uses who have absolutely control over the products and can view, edit, delete anything from the system. The type of users who can view the analytics portion of the system like, who manages disputes and check the order, and can view the reports of the type of users, products added, orders but cannot bid on a project and cannot buy a products. The admin user has the capability to list the products.

2.2 UNDERSTANDING THE SOLUTION

Django is a high-level Python Web framework that helps in rapid development and clean, pragmatic design. It's free and open-source, fast and exceedingly scalable. Django was intended to help developers take applications from concept to completion as quickly as possible. Django has a great security measure and it is reassuringly secure. One of the most important aspects that have made Django popular is because of its ability to quickly and flexibly development of the web. **(Django Software Foundation , 2020)**

Django with authorization supports the databases: PostgreSQL, MariaDB, MySQL, Oracle SQLite. Django supports many features as possible on all database backends. Django needs mysqlclient, mysqlconnect-c to connect to the MySQL database. (Django Software Foundation , 2020) The tables in the database are created automatically by Django using Django Models and but first, a database must be created to store the tables in Database. MySQL is an Oracle-backed open-source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL is based on a client-server model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). **(TechTarget, 2020)**

JetBrains is a global software vendor specializing in the creation of intelligent, productivity-enhancing tools for software developers and teams. JetBrains was founded in February 2000. Jet Brains has been continuously updating by automating routine checks and corrections, our tools speed up production, freeing developers to grow, discover, and create. The Python IDE for professional developers. PyCharm offers a unique coding experience for productive Python, Django, and web development. (JetBrains s.r.o., 2018) PyCharm offers great framework-specific support for modern web development frameworks such as Django, Flask, Google App Engine, Pyramid, and web2py, including Django templates debugger, manage.py, and appcfg.py tools, special auto completion and navigation, just to name a few. **(JetBrains s.r.o., 2020)**

The system is defined below: A client places an order or wins the bid. The order placed can directly go to the cart . The bid directly goes to order and is charged through an online payment system the order is completed. If the online payment system is not acceptable to the user can select cash on delivery service. The order is completed an email is sent to both seller and buyer. The placed order is carried out through the buyer and seller. The shipment is collected by the buyer who will collect it. My system user 3 tier application architecture where client interface, application layer and database layer. The client/ user interface includes webserver, browser and internet. Suppose a user wants to buy a product form the website. A user uses a web browser to visit the website connects through webserver to communicate with the e-commerce customer to the seller. The application layer includes the Application layer and the backend layer. The purpose of the application layer is to log in to the shopping system and check products that are available on the site. The database layer is the storehouse of data where all the data are stored in the database like product data, seller orders. The layer is accessible only through the application layer.

2.3 SIMILAR PROJECTS

One of the systems similar to my project is eBay, the company was founded in 1995. eBay is where the world goes shopping, sell, and give items basically a trading spot for people. In 2019 there have been 183 million active buyers. Sellers have the platform to grow their businesses through eBay. eBay measures success by our customers' success. eBay is in the top 10 global retail brands. (eBay Inc., 2020)

There are two types of listings on eBay that are auction-format listings and fixed-price listings. Fixed price listings are the same as online shopping. the user clicks a button to make a purchase, then pay for it. Fixed-price items have a "Buy It Now" button with a price listed next to it. The users can buy after clicking the "Buy It Now" button paying the price listed next to the button. Auction items have a "Place Bid" button. The box for entering bids and shows "current bid" prices are also available. Auction items are open to bids for a fixed amount of time. When time is up, the item is declared "sold" to the highest bidder. eBay auctions accept bids only for a specific amount of time. Users must place a bid that is higher than the current bid. The bid winner can pay less than your bid if the user wins but not less than the 2nd bidder’s amount. If the user's bid wins, the payment must be done. Functionality like Use 1-Click Bid can be used to place small bids and 3rd party programs can also be used to bid for automatic bidding system. (Hsiao, 2018)

eBay has built an online person-to-person trading community on the Internet. In the website sellers list items for sale, buyers to bid on items of interest as well as can buy the items. eBay users can browse the items. The items are arranged by topics, where each type of auction has its own category. This facilitates easy exploration for buyers and enables the sellers to list an item for sale immediately after registering. eBay notifies the buyer and seller via e-mail at the end of the auction if a bid exceeds the seller's minimum price, and the seller and buyer finish the transaction independently of eBay. The binding contract of the auction is between the winning bidder and the seller only. (brandeis.edu, 2016)

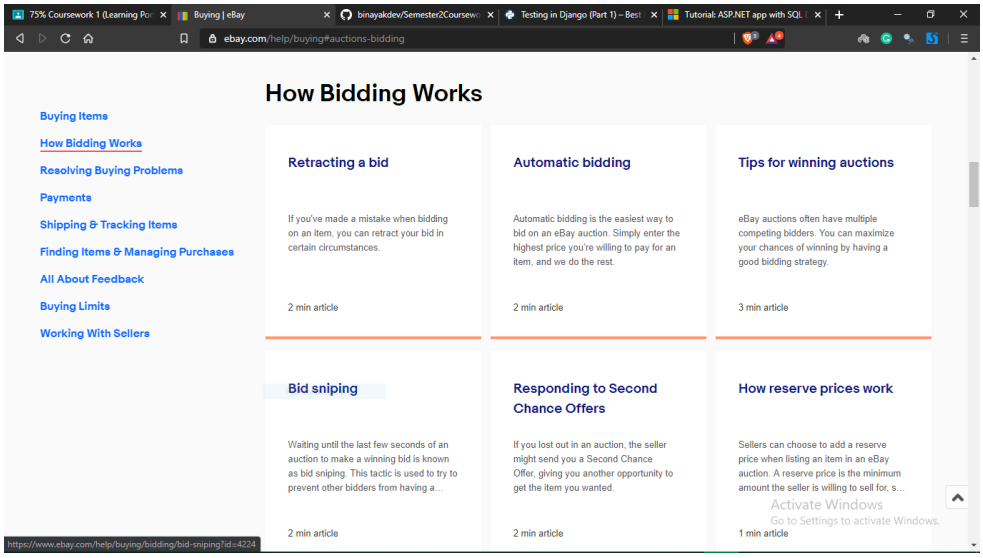


Figure 1:Ebay workings

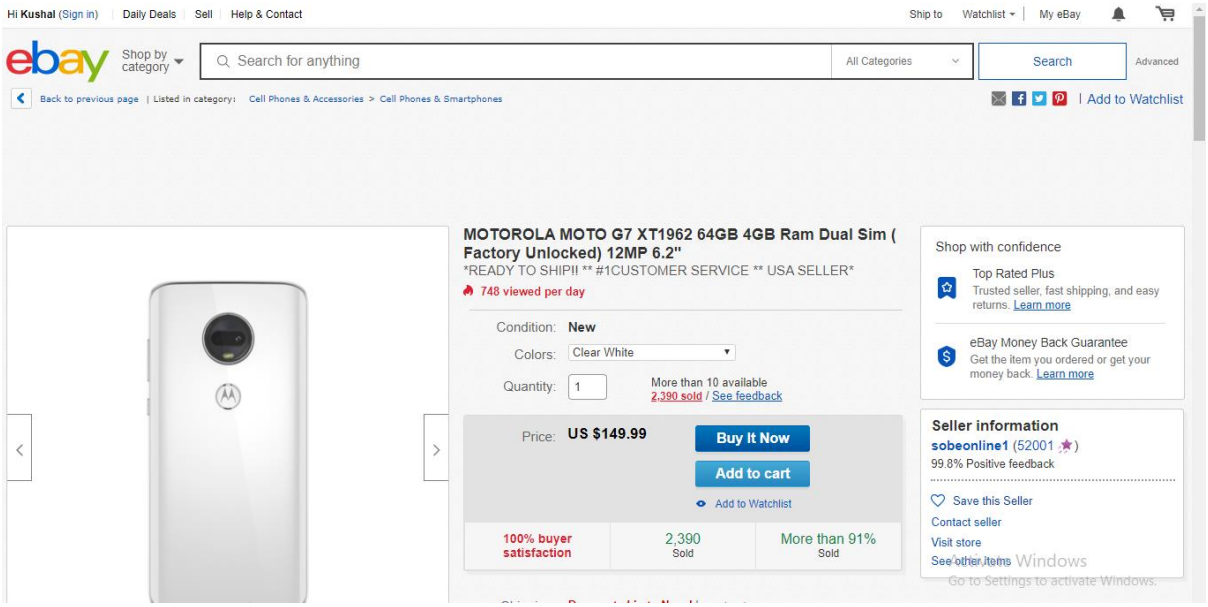


Figure 2:Ebay

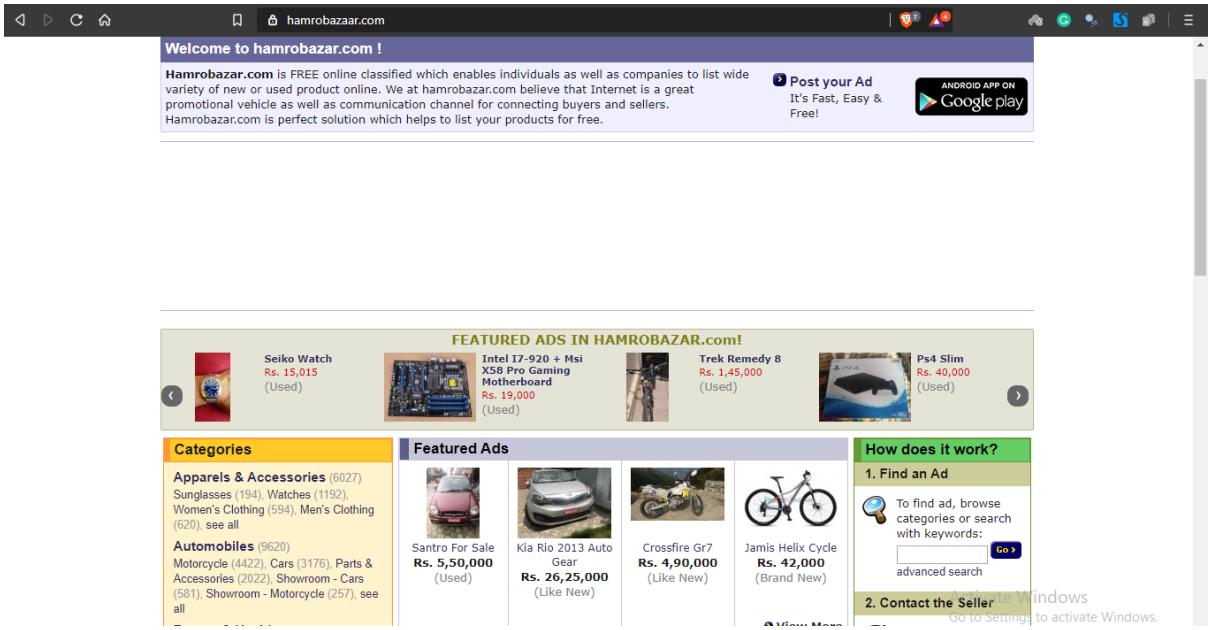


Figure 3:Hamrobazar site (hamrobazaar, 2020)

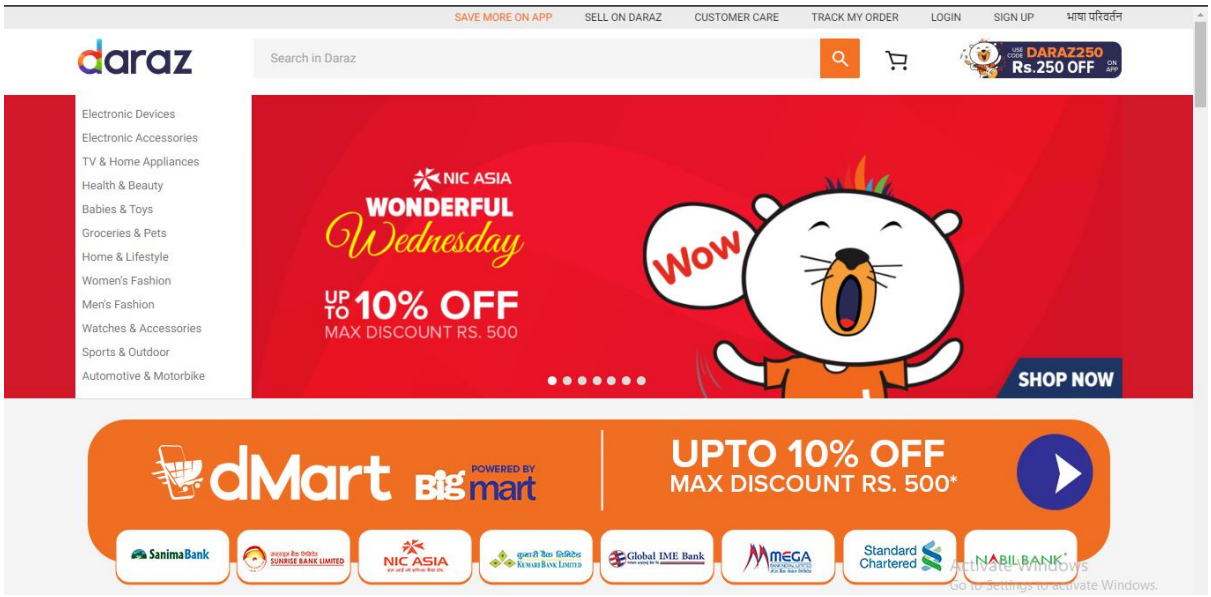


Figure 4:Daraz (Daraz, 2020)

2.4 COMPARISONS

Feature Desc	Samanghar(my project)	Ebay	Daraz	Hamrobazar
Add products	Yes	Yes	Yes	Yes
Add auction/listings	Yes	No	No	Yes
Time based Listings/ auction	Yes	No	No	Yes
Online Payment	Yes	Yes	Yes	Yes
Cash On delivery	Yes	Yes	Yes	Yes

Table 1: Table for product comparisons

Hamrobazar.com is FREE online classified which enables individuals as well as companies to list wide variety of new or used product online. We at hamrobazar.com believe that Internet is a great promotional vehicle as well as communication channel for connecting buyers and sellers. As per NTA April 2019 report, the internet users in Nepal has reached 18.24 million (including 14.24 million mobile data users) which itself indicates that the market for internet advertising is highly lucrative. Hamrobazar is visited by around 800,000 unique visitors monthly who use the site for buying and selling purpose. (hamrobazaar, 2020)

Daraz is the leading online marketplace in South Asia connecting thousands of sellers with millions of customers in Pakistan, Bangladesh, Sri Lanka, Nepal and Myanmar. Daraz has built a robust payment infrastructure that is localised to each market. Daraz have integrated with leading banks and local opened-loop wallets in the five countries to ensure that customers have a number of digital payment options. In Pakistan, the closed-loop Daraz Wallet has been launched as the smartest payment option. Daraz is known to be the best online selling website in Nepal. (Daraz, 2020)

CHAPTER 3: DEVELOPMENT

3.1 CONSIDERED METHODOLOGIES

3.1.1 Methodology 1- RUP

The RUP project is iterative and is measured in milestones. The iterations are used to manage the project by the Project manager. Here the RUP Planning process is developed in a structured way. The plan is made for each step and helps to reduce waste of resources. Inception Elaboration, Construction, and transition are the phases that are repeated in many iterations in this methodology. Each of the four stages has the main objective, which has to be completed before the project can progress to the next step. Each of the life cycle phases can be repeated if needed until the main objectives are met. Once the transition stage is completed successfully, the project is finished. (Study.com, 2019) (Sharpened Productions , 2019)

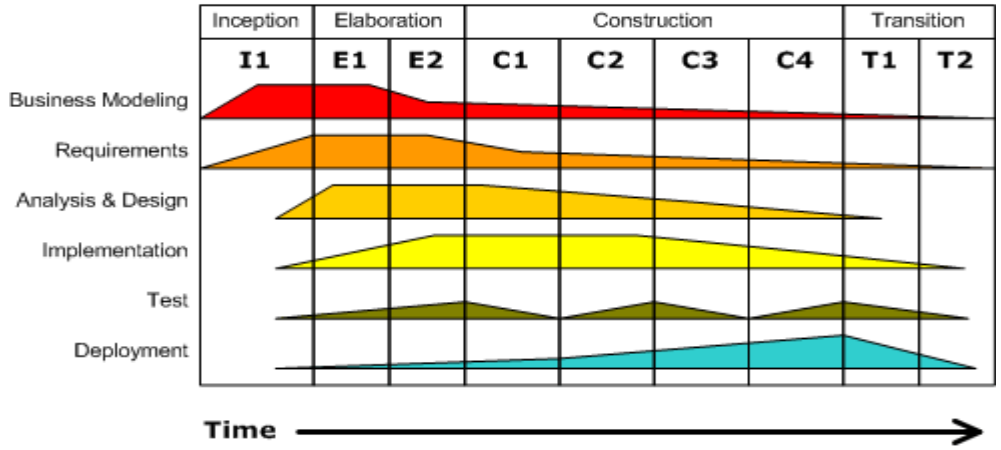


Figure 5: RUP development framework cycle (Study.com, 2019)

Advantages of RUP

1. This is a complete methodology in itself with an emphasis on accurate documentation
2. It is proactively able to resolve the project risks associated with the client's evolving requirements requiring careful change request management
3. Less time is required for integration as the process of integration goes on throughout the software development life cycle.
4. The development time required is less due to reuse of components.

3.1.2 Methodology 2- Spiral

The Spiral model is Meta-model as a result of it subsumes all the initial models. This model is named from its diagrammatic appearance that looks like a spiral with many loops. The exact number of loops of the spiral is not fixed and can vary from project to project. Each loop of the spiral is called a phase of the software process. The exact number of phases through which the product is developed can differ in according to the risk factor. The model helps the developer to understand and resolve the risks at each iteration along the spiral loop. The spiral model uses prototyping is used to produce a risk reduction mechanism. The methodology also uses the systematic stepwise approach of the waterfall model in the spiral loops. Disadvantages of the Spiral Model is that the process is risky, time-consuming costly, and because of many strict protocols that need to follow, the risk factor of the process is high. A risk analysis expert is required and could be costly which could again affect the project budget. Even though it is used in large scale projects Because of its disadvantages and risk factors affect it, I did not choose this methodology. (Learntek, 2019)

3.1.3 Methodology 1- Waterfall

The classical waterfall model divides the SDLC life cycle into a set of phases which is only started after the previous phase is completed. The output of one phase will be the input to the next phase. The phase in the methodology is as follows: Feasibility Study, Requirements analysis and specification, Design, Coding and Unit testing, Integration and System Testing and Maintenance. The development process is in a sequence that flows like a waterfall where the phases do not overlap over one another and don’t go from down to up. The waterfall model is not practicable as there are not any feedback path. The process flows one after another as if an error is not committed in any phases and the project was done in 100% accuracy.

This doesn't incorporate any mechanism for error correction sot it's not practicable. The requirements can change during the process and the change requests can be difficult to accommodate after the requirement specifications. this can't be maintained. To increase the efficiency and reduce the cost, phases may overlap but in the waterfall the phases do not overlap so even though I considered doing the project in waterfall methodology because of its disadvantages I did not choose this methodology. (geeksforgeeks, 2019)

3.2 SELECTED METHODOLOGY

The methodology that is going to be used is the Rational Unified Process RUP. Each phase is finalized with a milestone. A milestone is a point in time where decisions of critical importance must be made. In order to be able to make those decisions, the objectives must have been accomplished. For example, a milestone from the first two phases is the progress of the use case. A use case is a description of a system's behaviour and describes who can do what using a system. This is an important component in the development of software. Normally, a product will already have to be completed by then. That is because this involves prototypes and test models.

Resource	Worker	Activities
Kushal	Manager	Plans iteration
	Designer	Define prototype and operations
	Use Case Specifier	Details the use case
	System Analyst	Finds actor and Use cases
	Architect	Idnetify design mechanisms
	Coder	Builds the system
	Implementer	Perform tests

Table 2: Resourcce Planning

3.3 PHASES OF METHODOLGY

The Stages/Phases of RUP are as follows:

1. **Inception**

   The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed. During the first phase, the basic idea and structure of the project are determined. In this phase, the team meets regularly to determine the project's necessity, but also its viability and suitability. Viability and suitability also include the expected costs and the means needed to complete the project after the green light has been given (Rational, 2011) (Sharpened Productions , 2019)

2. **Elaboration**

   The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development. During the elaboration phase, the system's requirements and its required architecture are assessed and analyzed. This is where the project begins to take shape. The objective of the elaboration phase is to analyze products and to lay a foundation for future architecture. (Rational, 2011) (Sharpened Productions , 2019)

3. Construction

The project is developed and completed. The software is designed, written, and tested. In the construction phase of the Rational Unified Process (RUP), the software system is constructed in its entirety. The emphasis is on the development of components and other features of the system. The majority of coding also takes place in this phase. In this production process, the emphasis is on managing costs and means, as well as ensuring quality. Results from the production phase include Fully completed software system User manuals. (Rational, 2011) (Sharpened Productions , 2019)

4. Transition

The software is released to the public. Final adjustments or updates are made based on feedback from end-users. The objective of the transition phase is to transfer the product to its new user. As soon as the user starts using the system, problems almost always arise that require changes to be made to the system. The goal, however, is to ensure a positive and smooth transition to the user. Results and activities in the last phase: Beta testing Conversion of existing user databases Training new users Rolling out of the project to marketing and distribution Input from the new users should guide the assessment here. (Sharpened Productions , 2019) (Rational, 2011)

AS seen, RUP methodology has a highly flexible development path. The best practice in RUP is Developing iteratively, managing proper requirements, use components of large projects to reduce production time, visualizing models using UML diagrams. In the methodology the software requirements specification (SRS) keeps on evolving throughout the development process and loops are created to add them without affecting the cost of development which helps in testing and using different components in development as well as testing. (My-Project-Management-Expert, 2009)

Phase	Inception	
Iteration Number	Iteration1	Iteration2
Objective	Research	Research
Implemented	Basic idea of the project was created	Researched about the project viability and necessity an research completed

Table 3:Inception Phase

Phase	Elaboration			
Iteration Number	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Objective	Create	Analyse	Update	New create
Implemented	Find System requirements	Analysing the system requirements	Updating the requirements	Create new and updated requirements after update

Table 4:Elaboration

Phase	Construction				
	Iteration 7	Iteration 8	Iteration 9	Iteration 10	Iteration 11
Objective	System update	System update	System update	System update	System update
Implemented	Logins, register, adding products	Authentication, dashboard	Updating UI system,	Cart , orders,	Search

Table 5:Construction

Phase	Construction				
	Iteration 12	Iteration 13	Iteration 14	Iteration 15	Iteration 16
Objective	System update	System update	System update	System update	System update
Implemented	cash on delivery payment	History fucniton	Notification	Updating searching	Admin Panel

Table 6: Construction pasrt2

Phase	Transition		
Iteration Number	Iteration12	Iteration 13	Iteration 14
Objective	Test	Update	Finalize
Implemented	Tested using unit testing coverage	Updation bugs	Finalized project

Table 7:Transiton fucntion

3.4 SURVEY RESULTS

3.3.1 PRE-SURVEY RESULTS

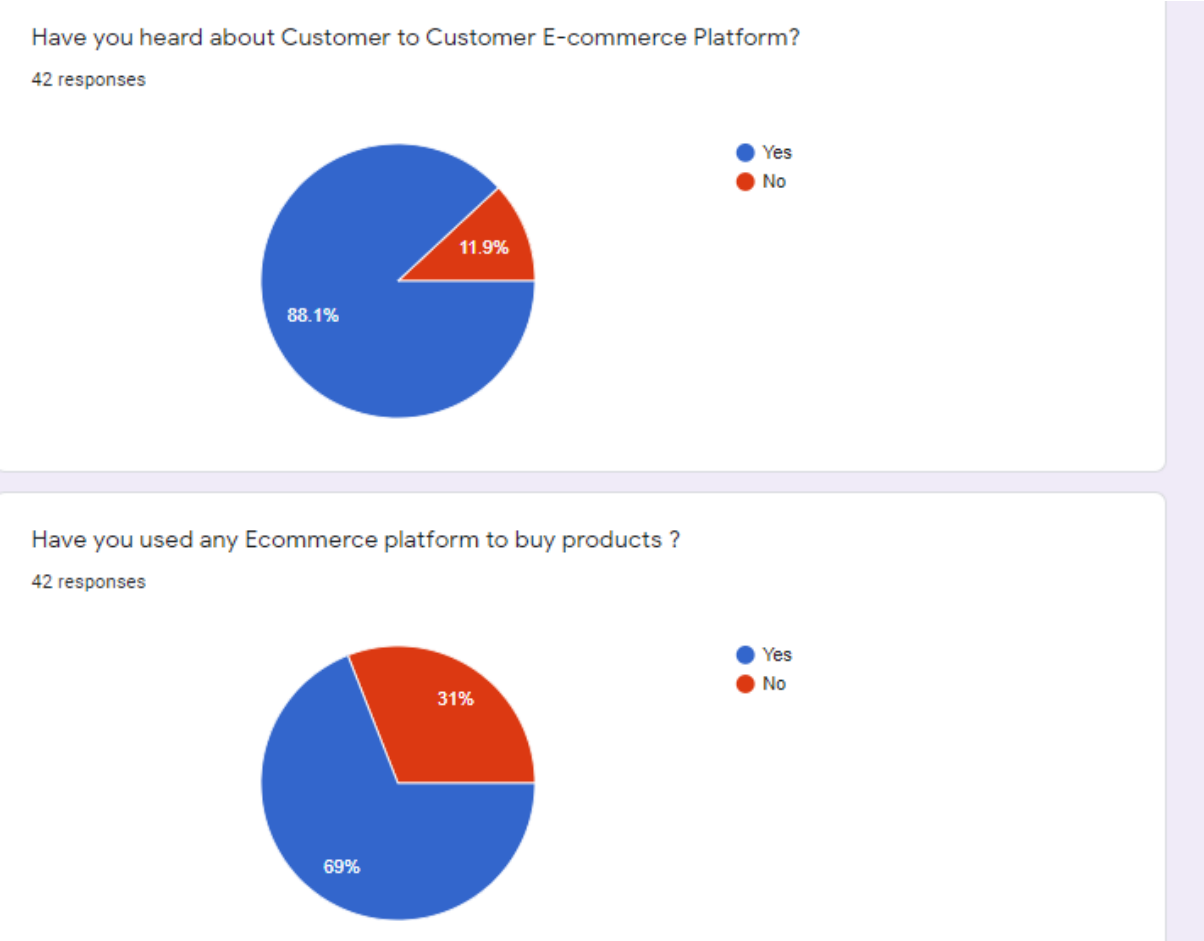


Figure 6:Survery question 1 & 2

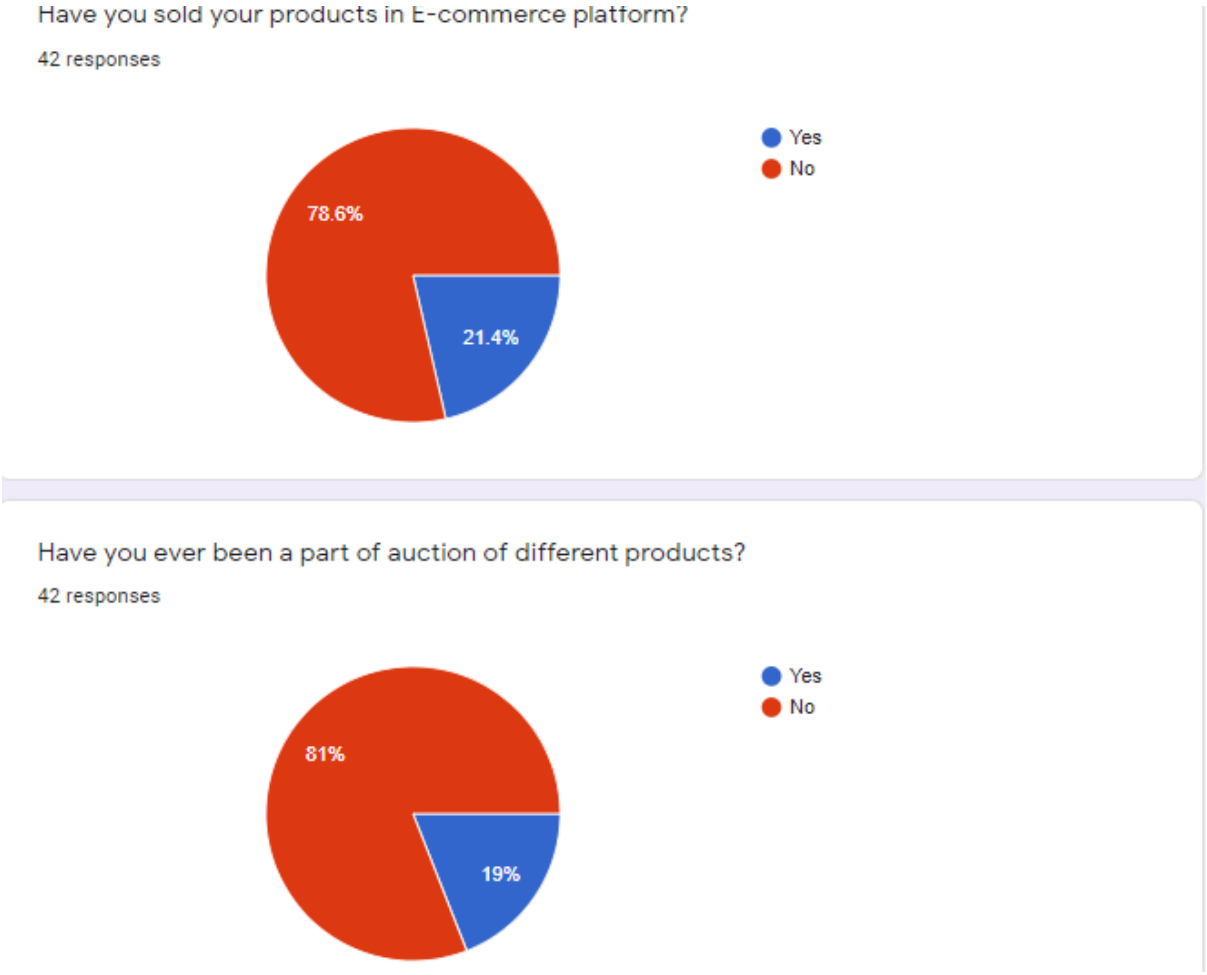


Figure 7:Server Question 3 and 4

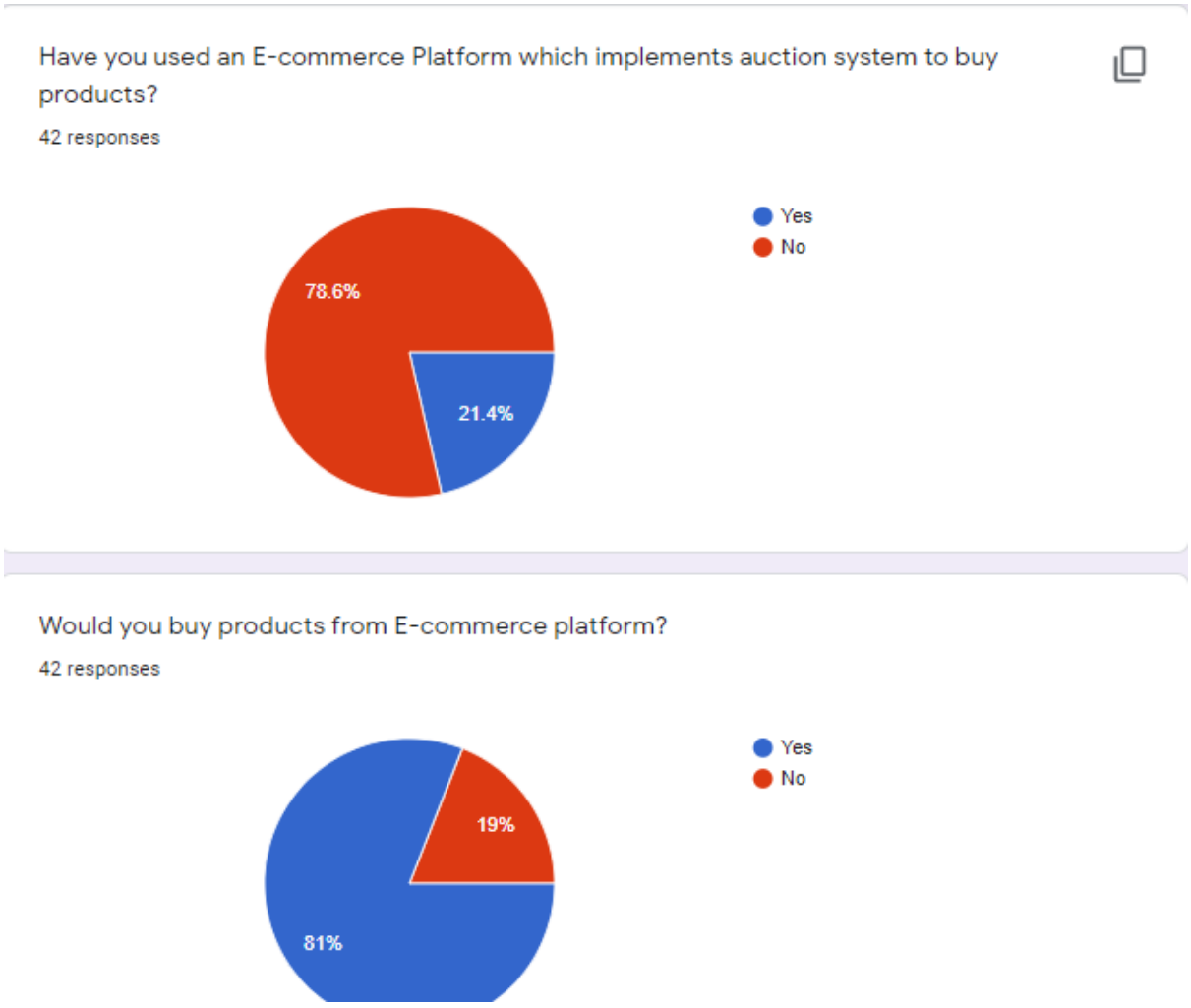


Figure 8:Survey question 5 and 6

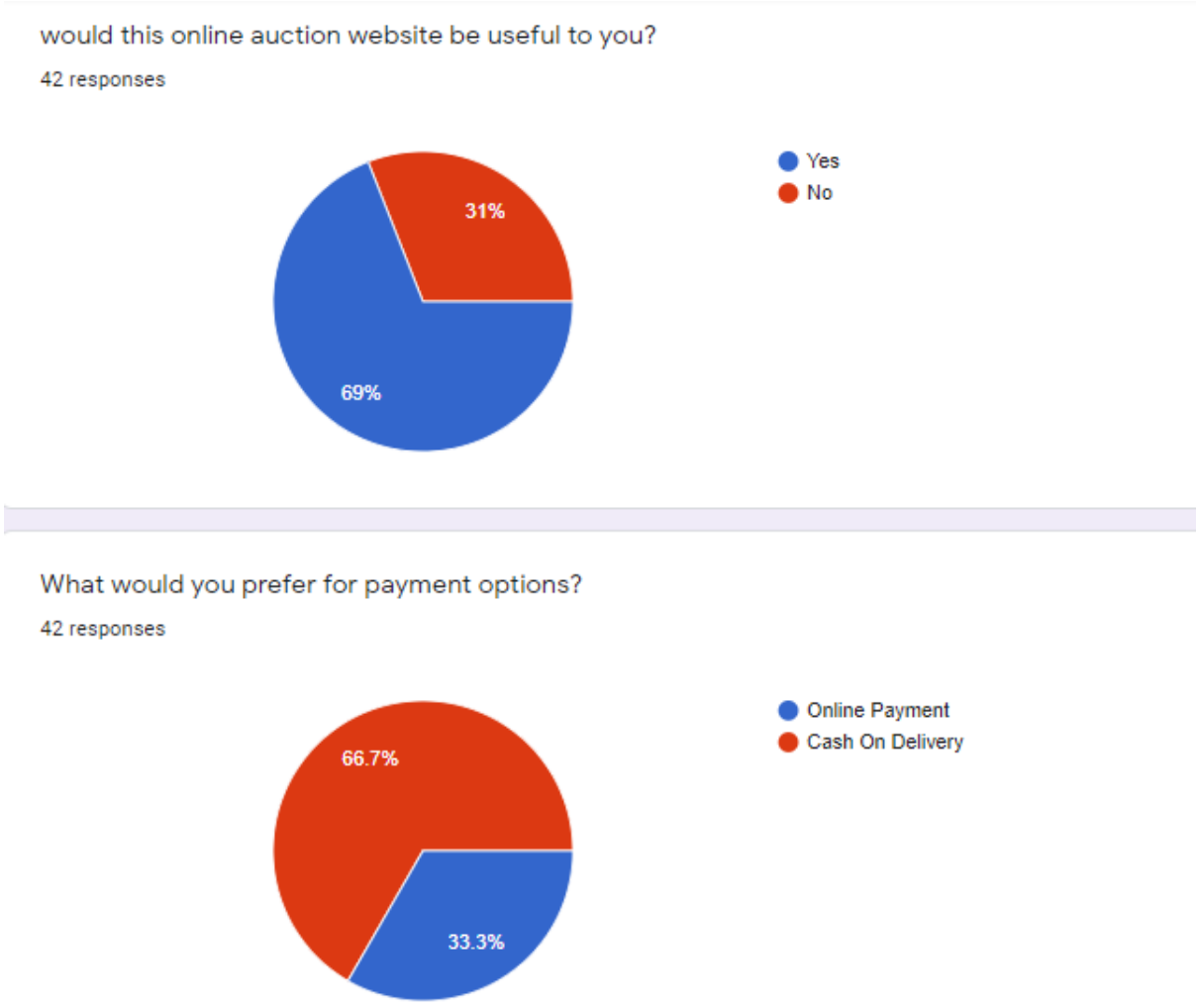


Figure 9: Survey Question 7& 8

Figure 10: Survey Question 9& 10

Mainly the survey gained positive reflection and encouraged to add certain features and gave idea about how to progress and create the system

3.3.2 POST-SURVEY RESULTS

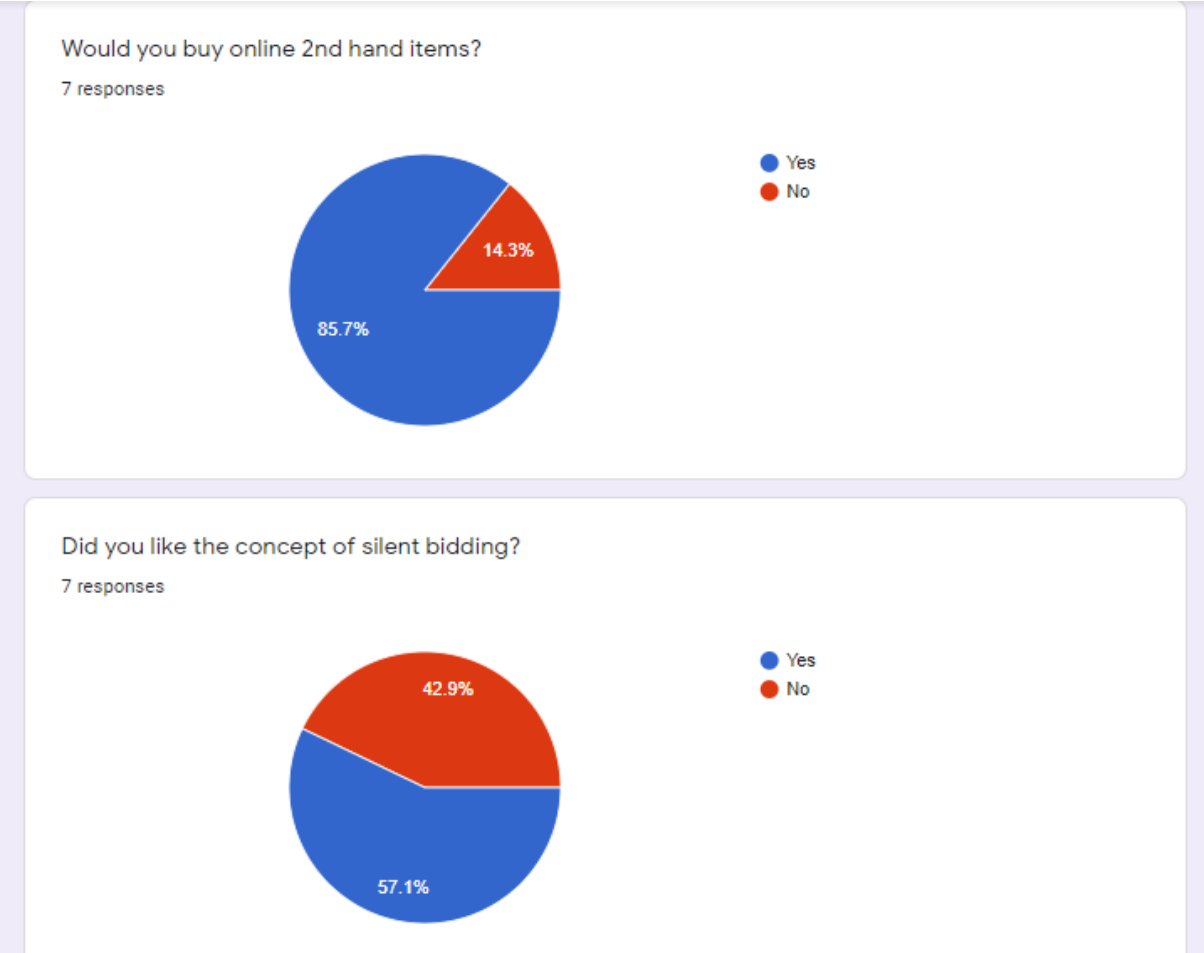


Figure 11: Post survey result 1

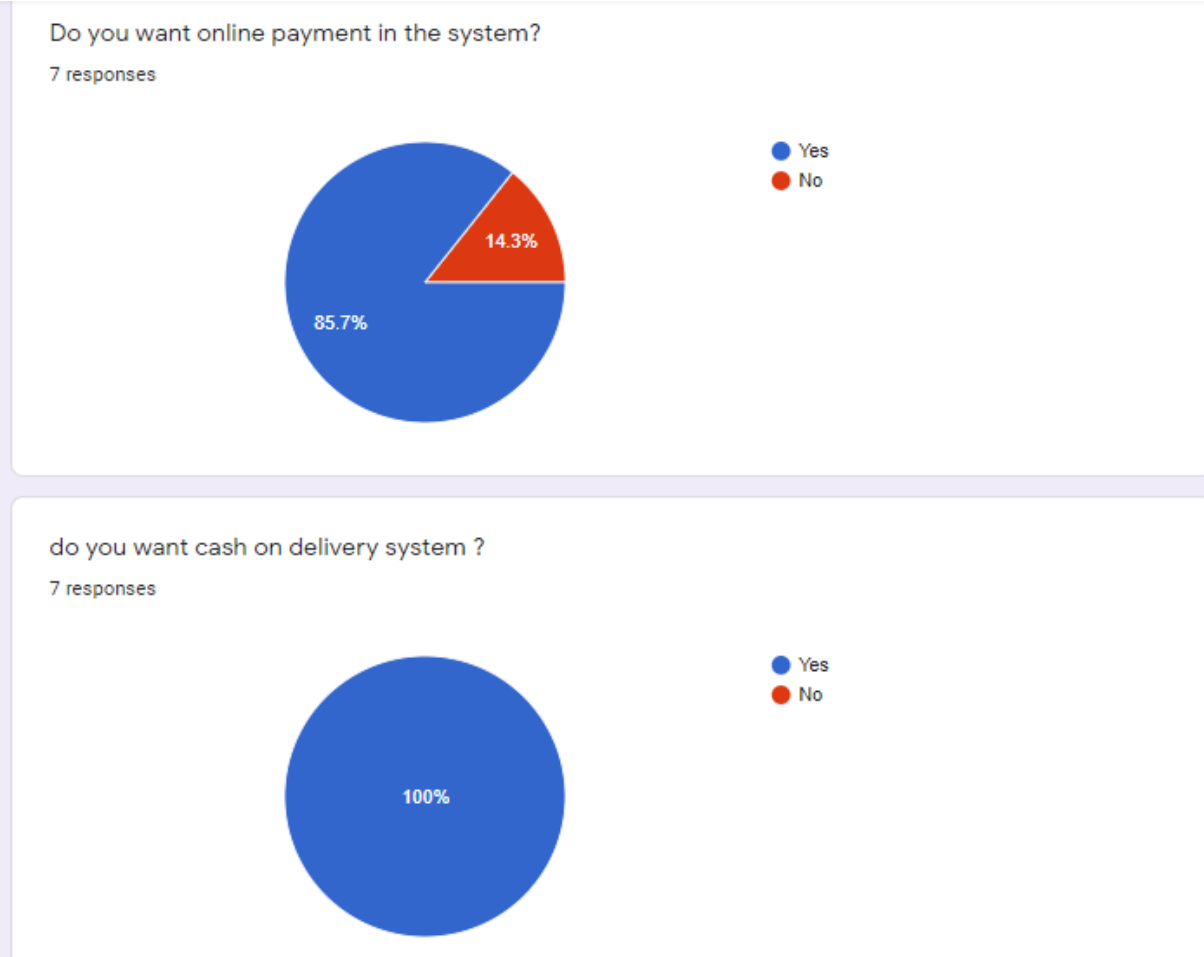


Figure 12: Post survey result 2

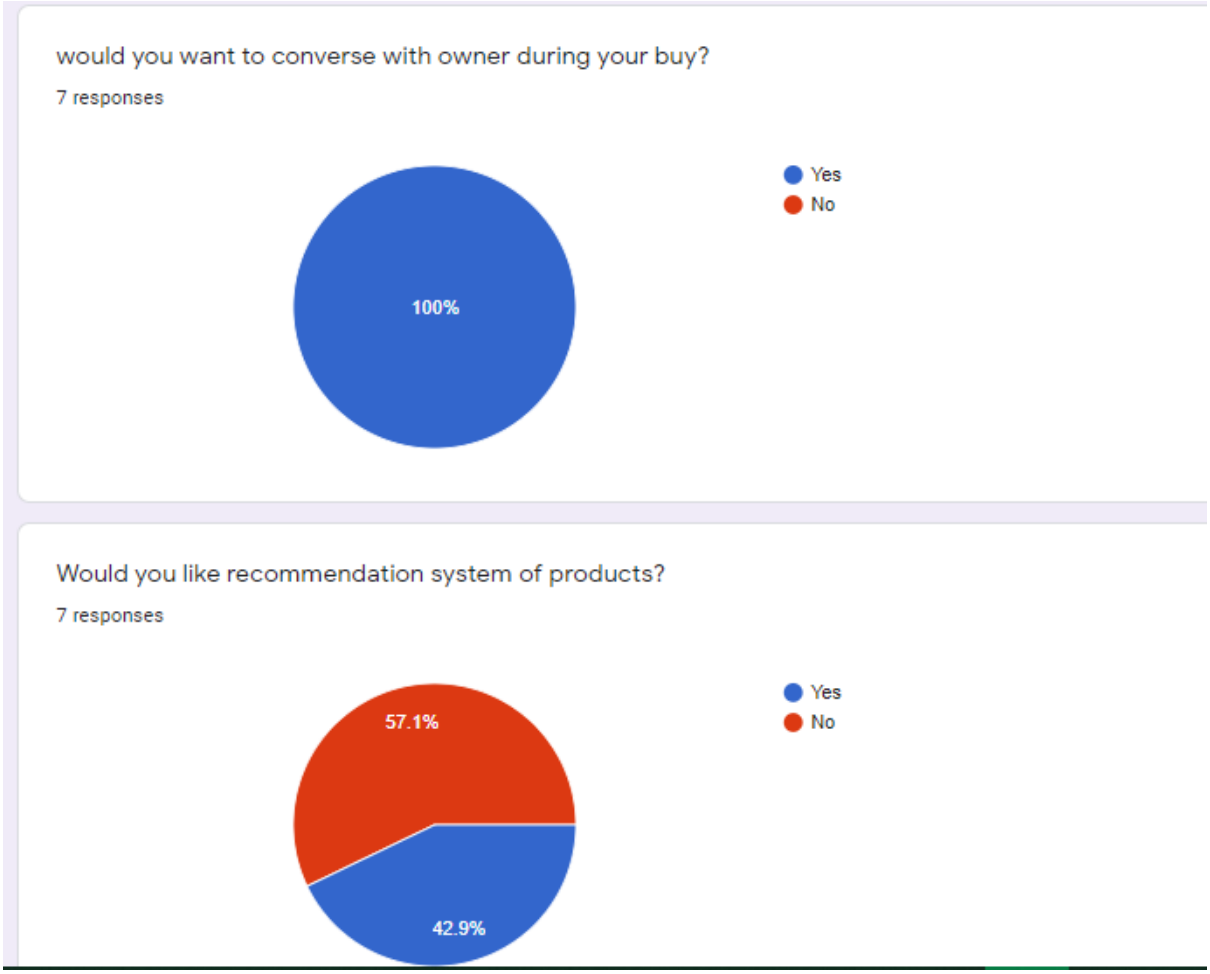


Figure 13:Post survey result 3

3.5 REQUIREMENT ANALYSIS

MoSCoW prioritization, also known as the MoSCoW method or MoSCoW analysis, is a popular prioritization technique for managing requirements. The method is commonly used to help key stakeholders understand the significance of initiatives in a specific release. The acronym, MoSCoW, stands for 4 different categories of initiatives: must-haves, should-haves, could-haves, and will not have at this time. Sometimes, the "W" in MoSCoW is used to stand for "wish" instead of "will not have right now." (ProductPlan , 2020)

MoSCow Prioritization parts	Functionality
Must Have (M)	Registration Login portal View products without login Add products Bid in products Fill out the shipping and billing details Payment through cash on delivery option
Should Have (S)	Shop - Cart system Update products Add time frame for auction Auction cart system
Could Have (C)	Add favourite products View history Notification system Online payment Conversation about the products in product page.
Will Not Have (W)	Overall messaging system for all the user Forum to request products Buy/ products as guest user

Table 8: MoSCow Prioritization parts

3.6 DESIGN TECHNIQUES

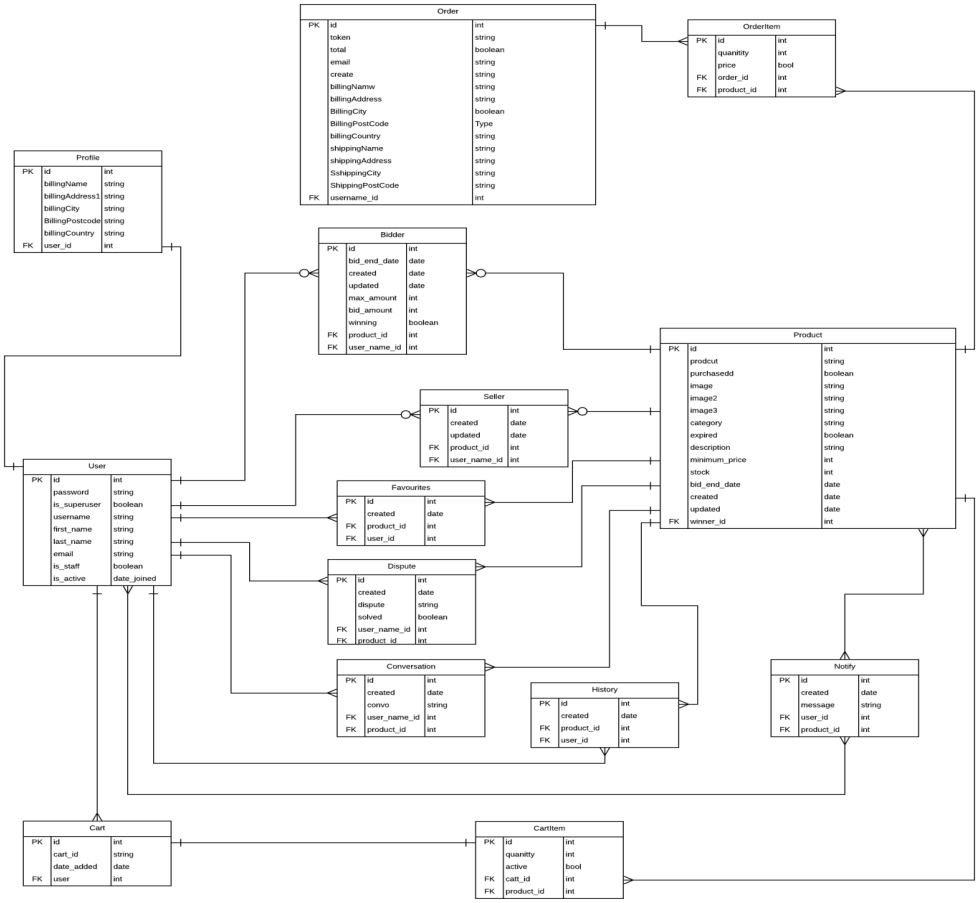


Figure 14: ER diagram

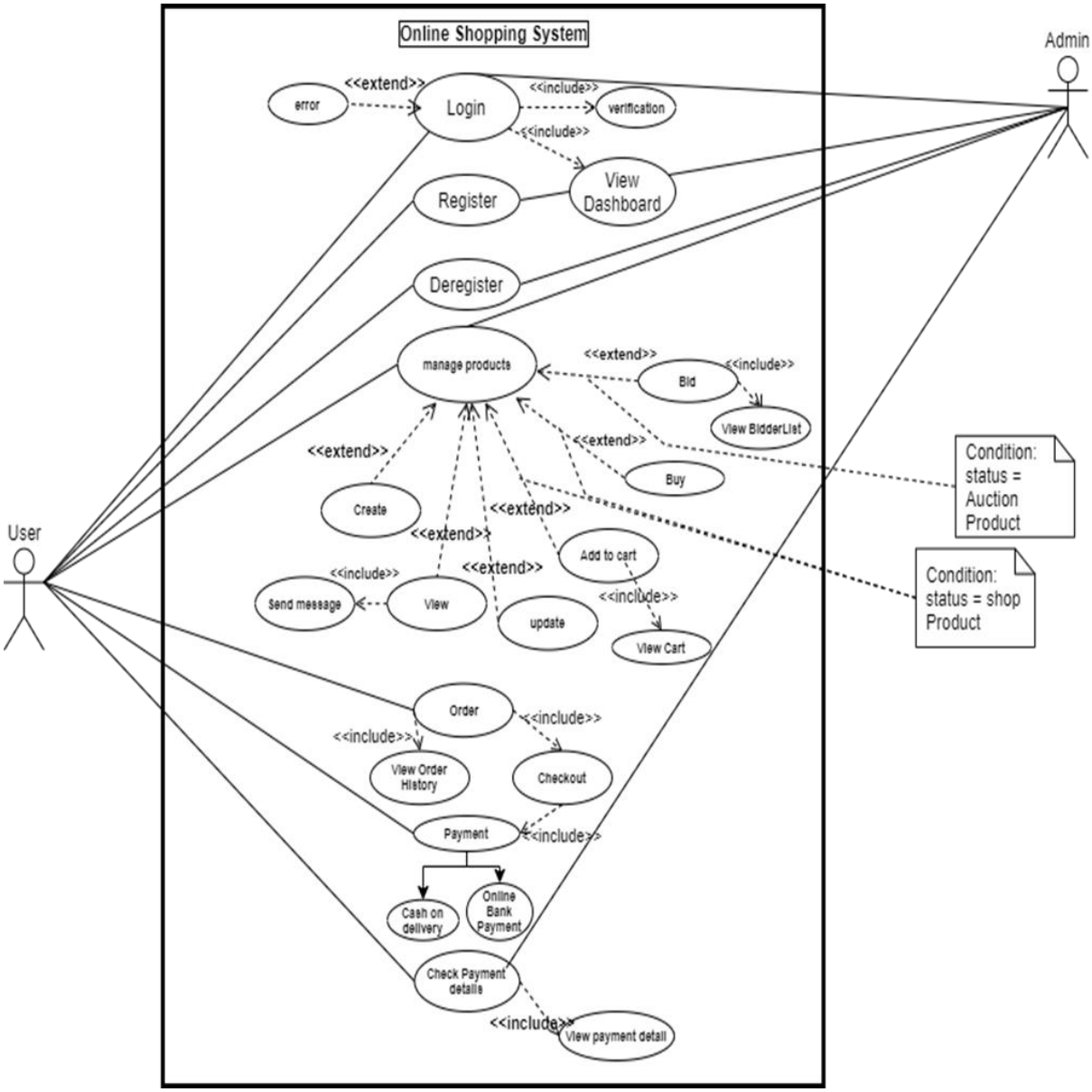


Figure 15: Use Case Diagram

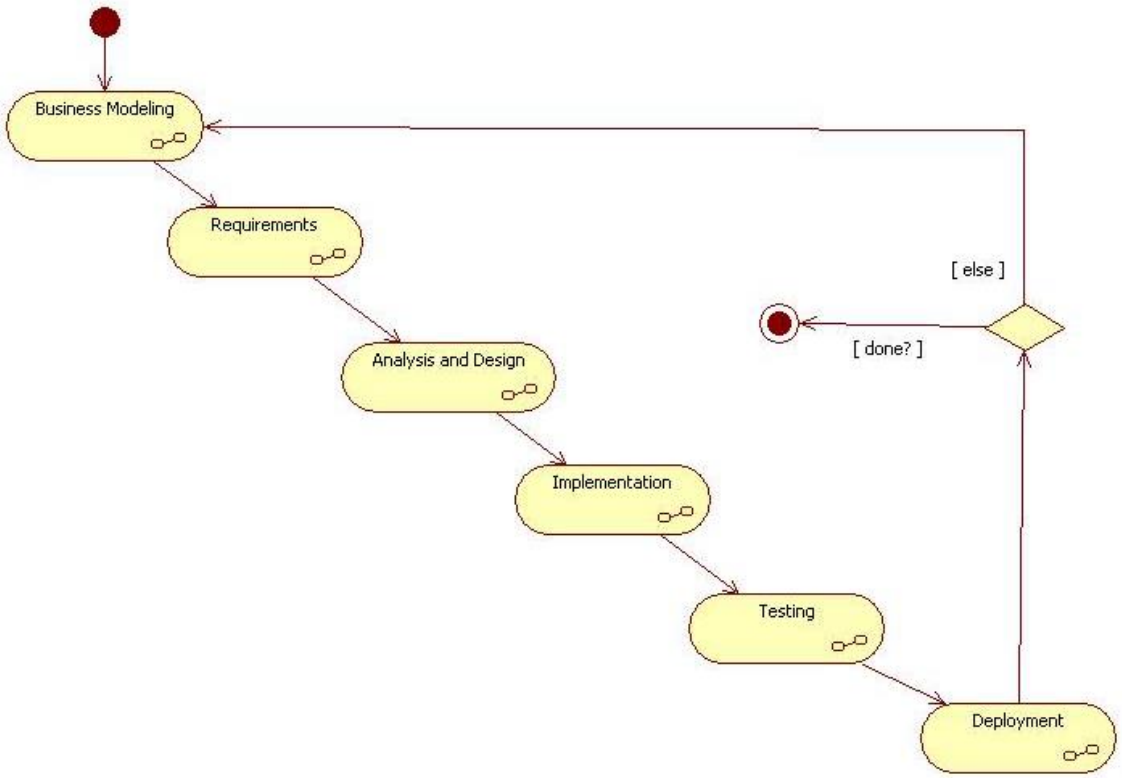


Figure 16: Iteration Worklow

3.7 IMPLEMENATION

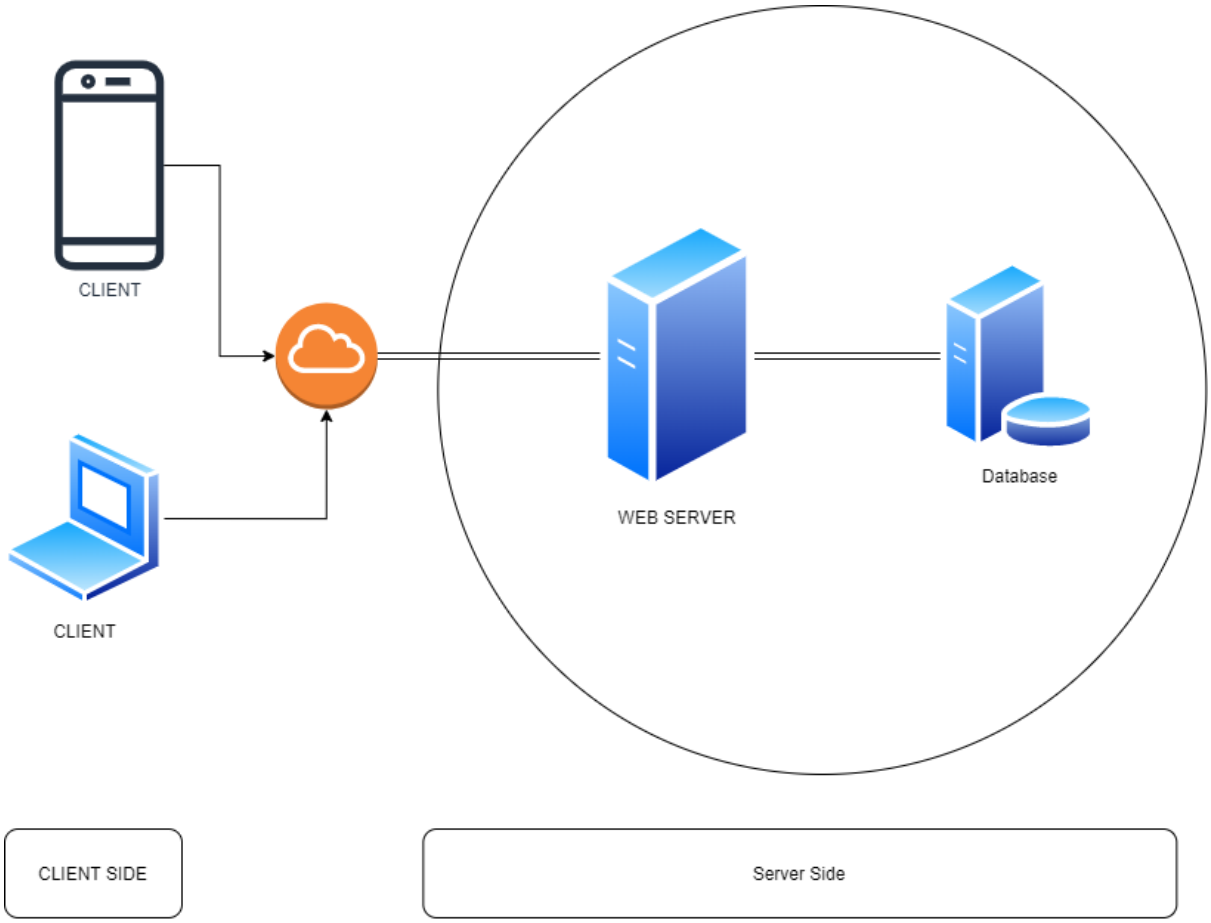


Figure 17:System architecture

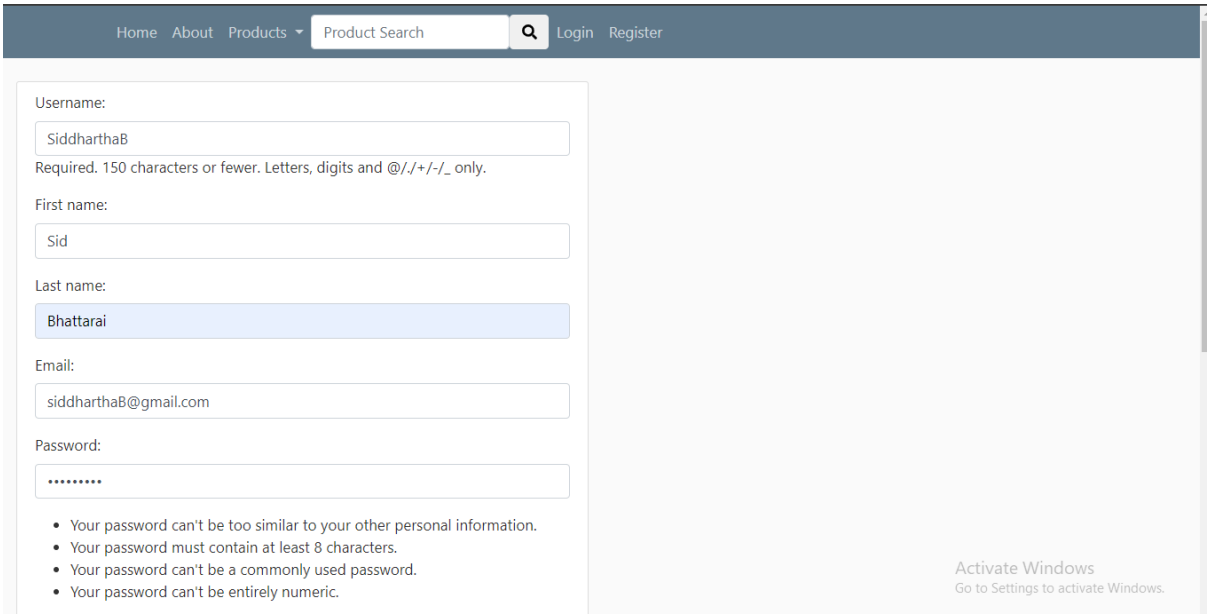


Figure 18: Login

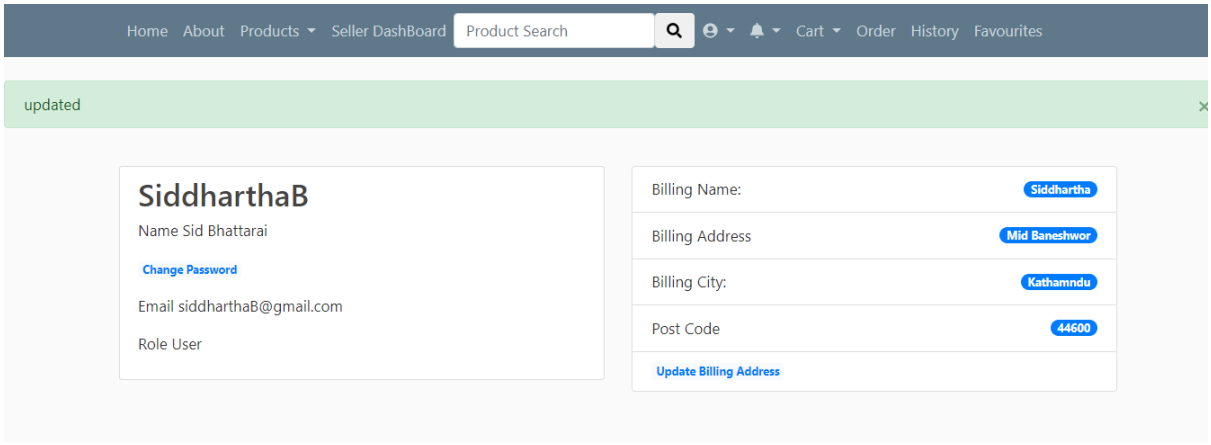


Figure 19:Profile

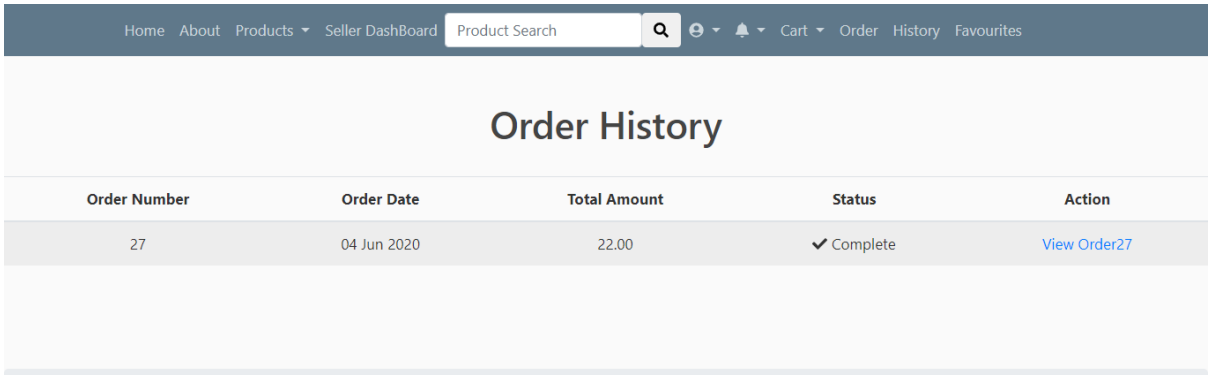


Figure 20:Order

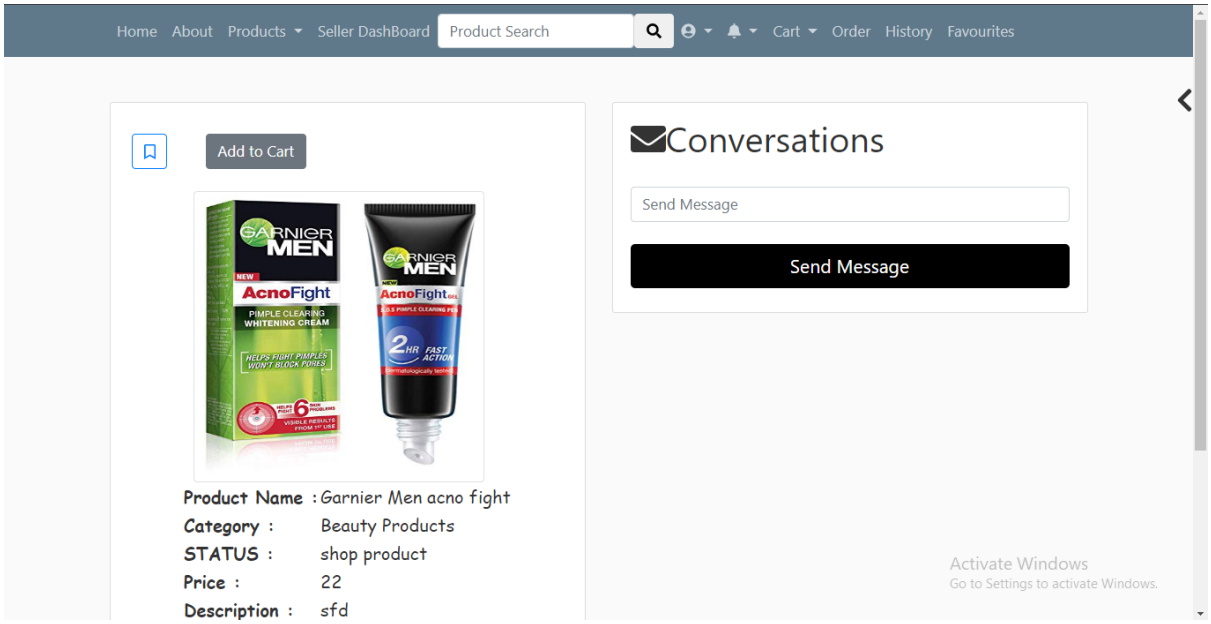


Figure 21: Shop product

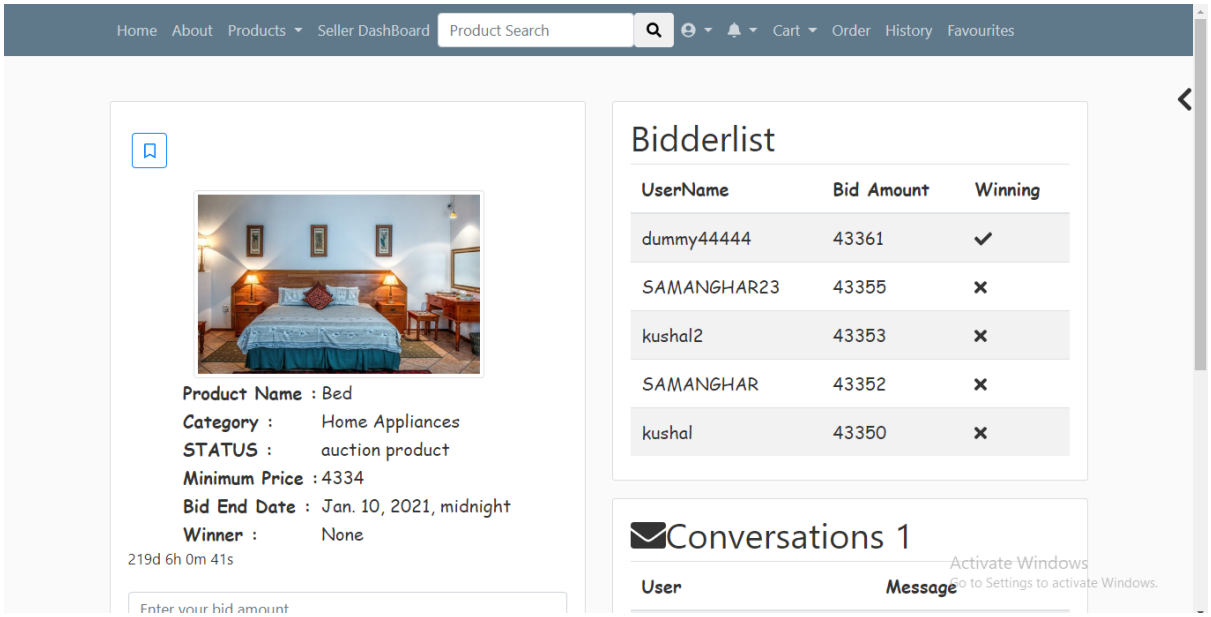


Figure 22:Auction Product detail

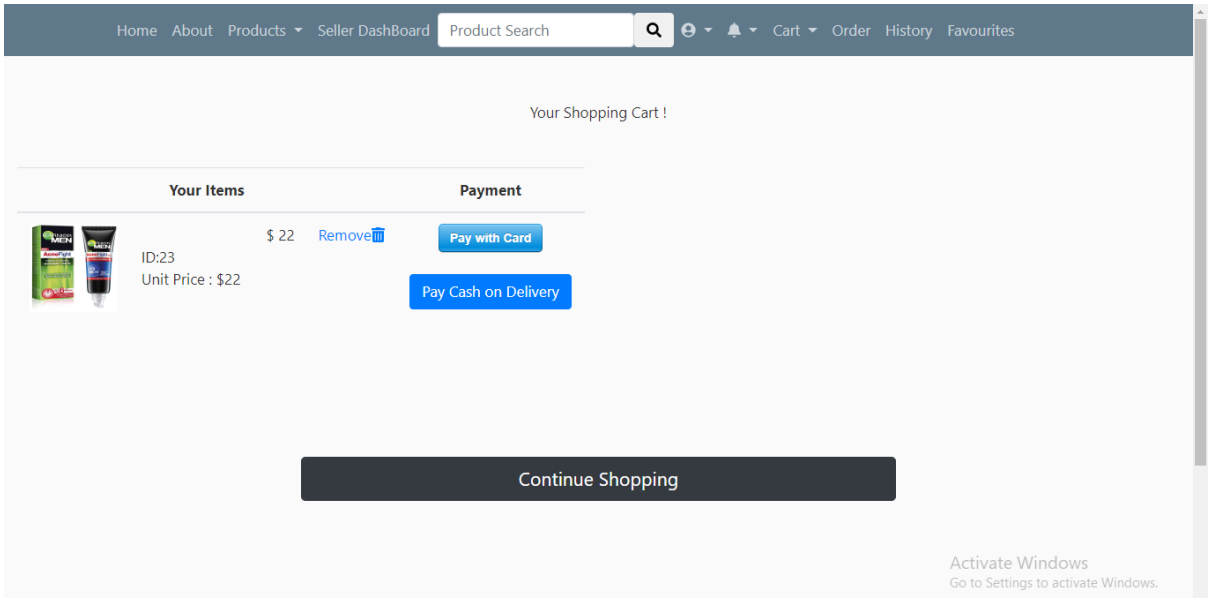


Figure 23:Cart

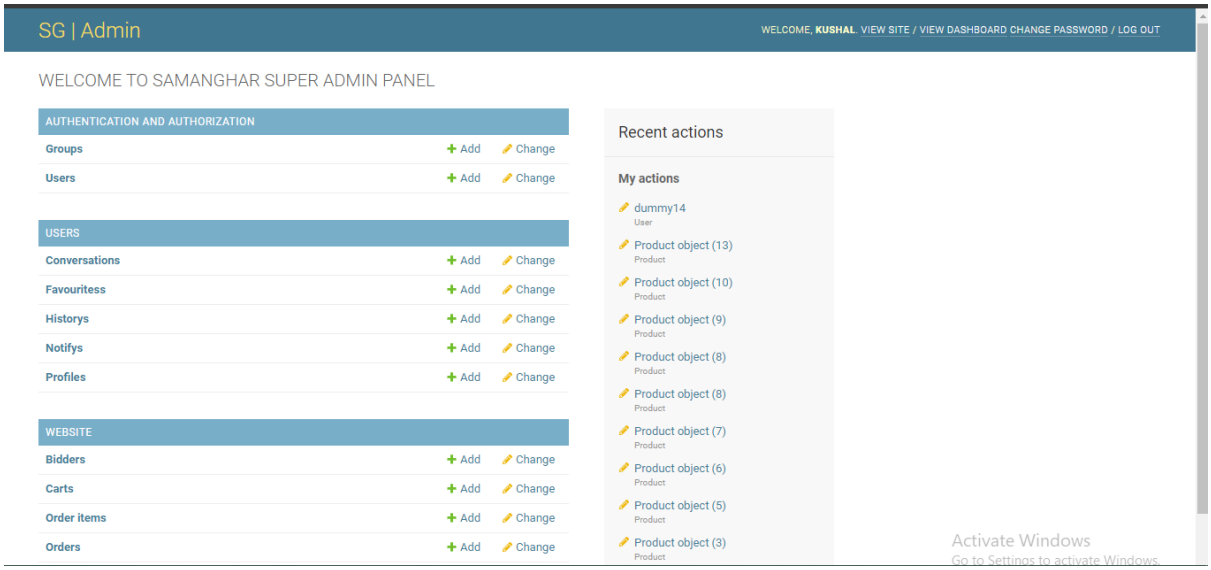


Figure 24:Admin page

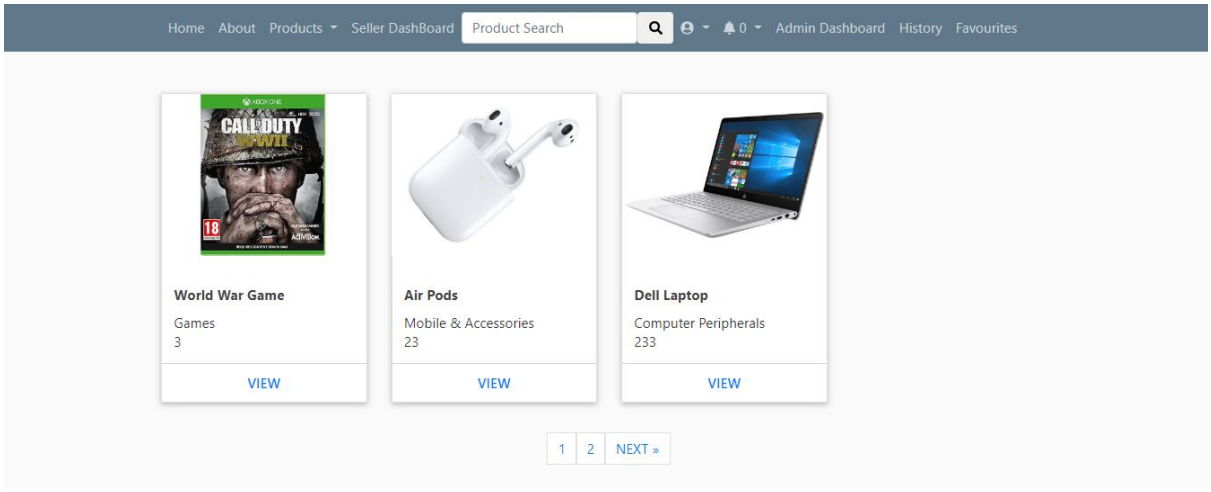


Figure 25:Shop

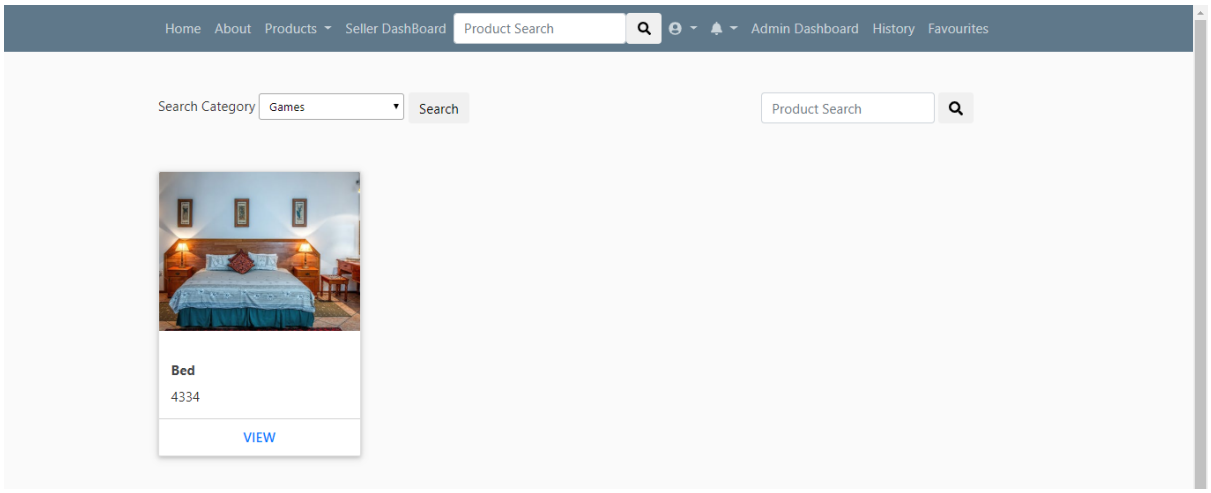


Figure 26:Search

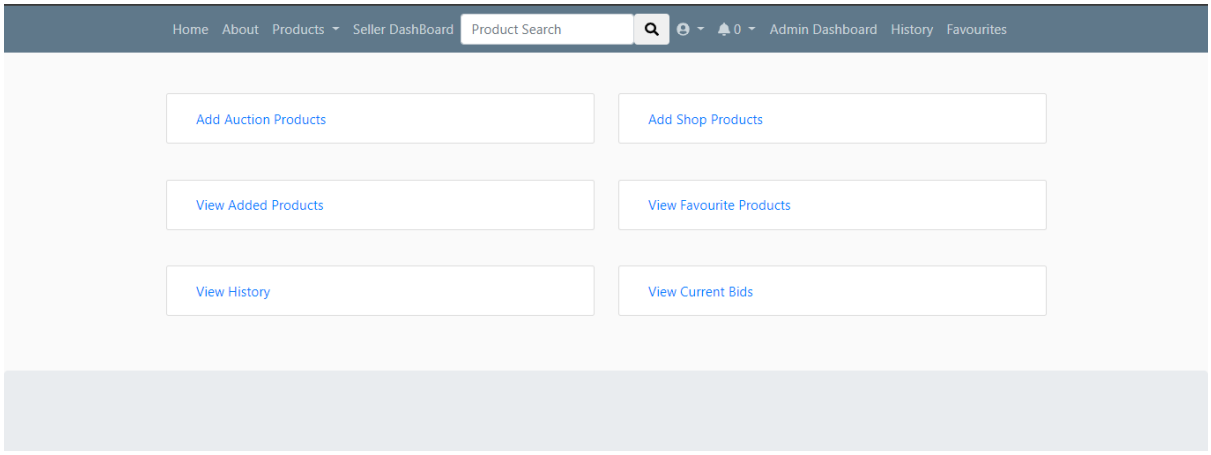


Figure 27:Functions

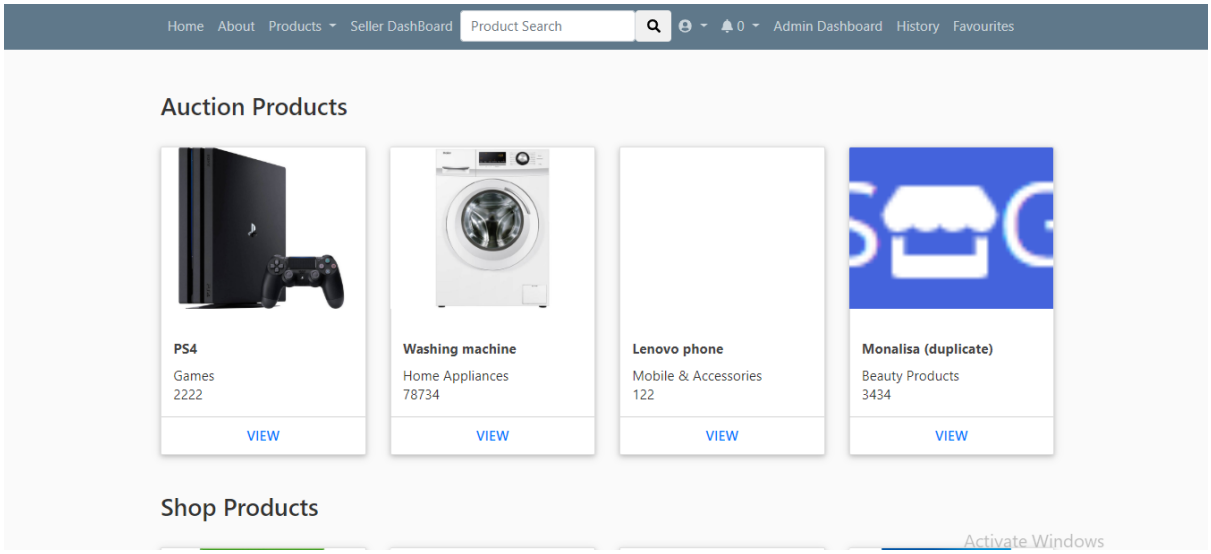


Figure 28:Expired/ purchased

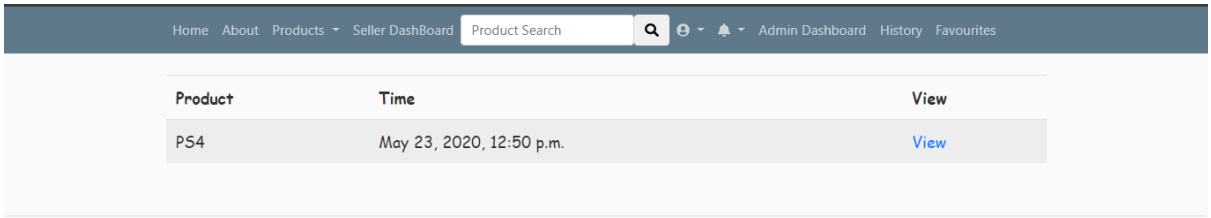


Figure 29: Favourite

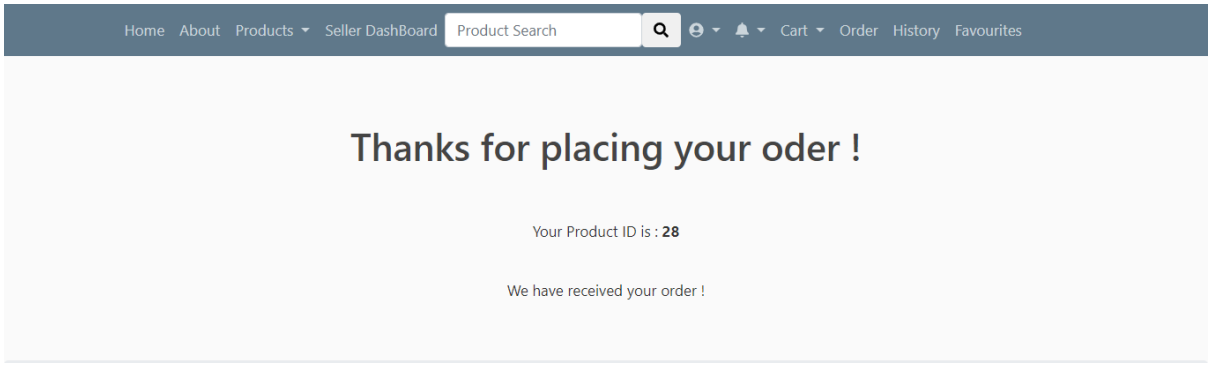


Figure 30:Placed order

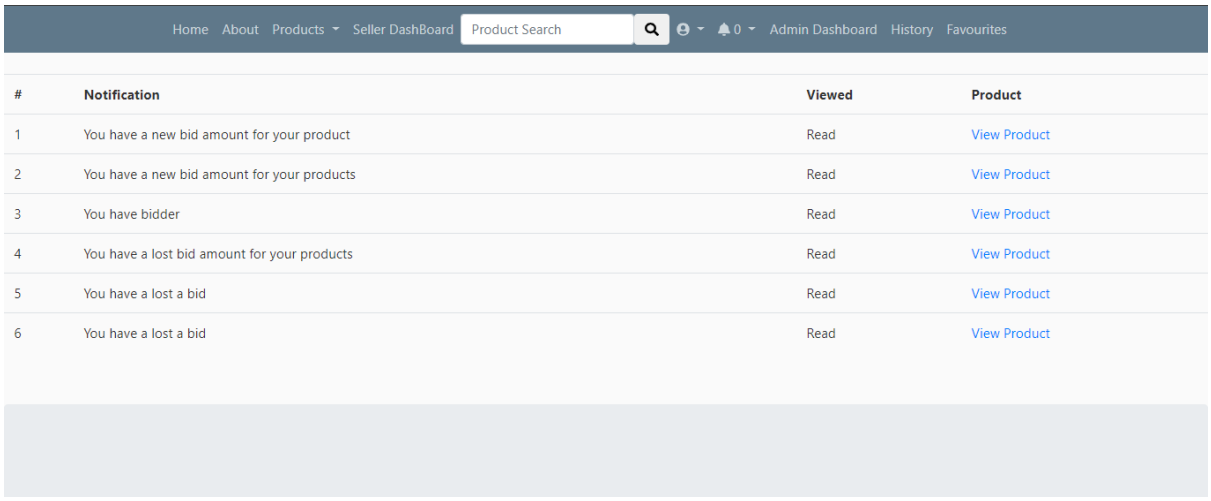


Figure 31:Notifications

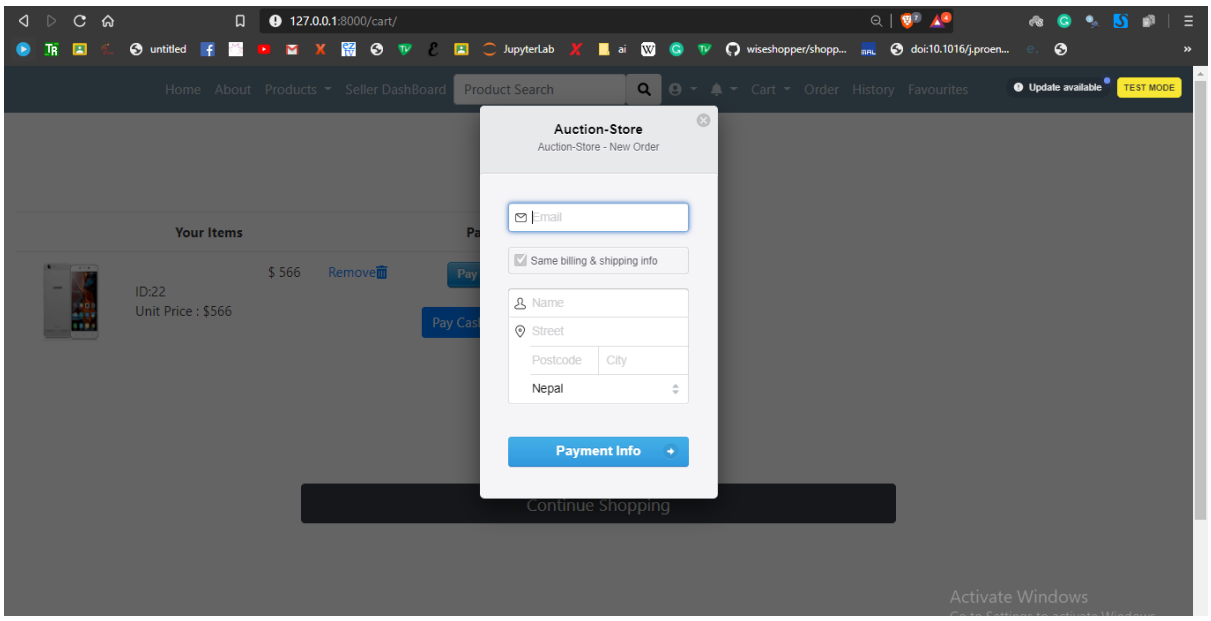


Figure 32:Stripe payment

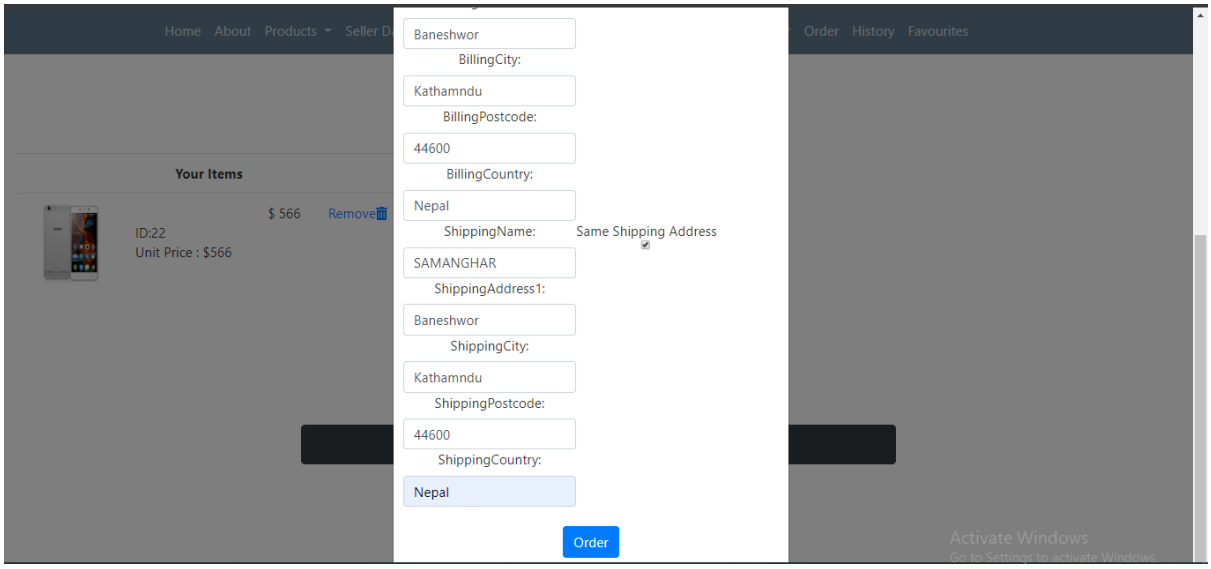


Figure 33:Payemnt cash on delivery

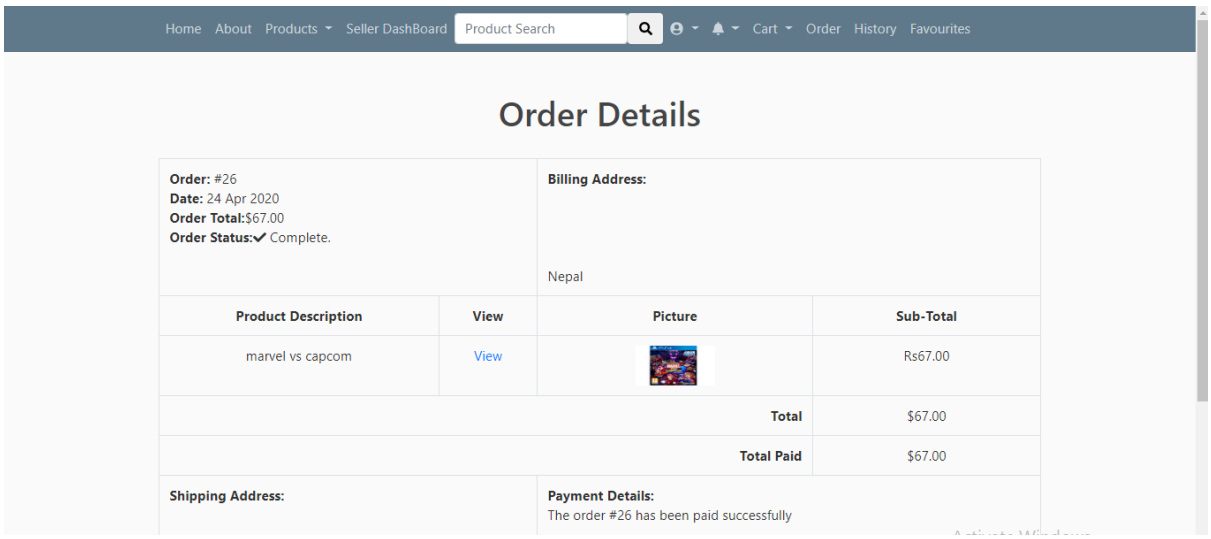


Figure 34:Order details

Develop iteratively

It is best to know all requirements in advance; however, often this is not the case. Several software development processes exist that deal with providing solutions to minimize cost in terms of development phases.

Manage requirements

Always keep in mind the requirements set by users.

Use components

Breaking down an advanced project is not only suggested but in fact unavoidable. This promotes ability to test individual components before they are integrated into a larger system. Also, code reuse is a big plus and can be accomplished more easily through the use of object-oriented programming.

Model visually

Use diagrams to represent all major components, users, and their interaction. "UML", short for Unified Modeling Language, is one tool that can be used to make this task more feasible.

Verify quality

Always make testing a major part of the project at any point of time. Testing becomes heavier as the project progresses but should be a constant factor in any software product creation.

Control changes

Many projects are created by many teams, sometimes in various locations, different platforms may be used, etc. As a result, it is essential to make sure that changes made to a system are synchronized and verified constantly.

(IBM Developer, 2020)

During the start of the project, I researched about many ecommerce platforms of world like , ebay, amazon, olx. I took surveys about the ecommerce system , researched about the bidding application topic, so that I would gain information about the system and how can it be implemented.

Firstly I learned how to create a django application searching tutorials and book and seeing the documentation.

Django

Views

A view function, or view for short, is a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really. The view itself contains whatever arbitrary logic is necessary to return that response. (Django Software Foundation, 2020)

I have created 3 views.py file where the view function are created one in classviews.py, website/view.py and users/view.py. class views have class based views whereas other two ivews are function based views. Generic views was used to adding and updating the products . the genric views that was used was CreateView and UpdateView. Class based view is a callable which takes a request and returns a response. This can be more than just a function. Django provides an example of some classes which can be used as views. These allow you to structure your views. Function based Views : The view returns an HttpResponse object that contains the generated response. Each view function is responsible for returning an HttpResponse object. this view function returns an HTML page that includes the current date and time. To display this view at a particular URL. That is where urls.py file is needed

Models

A model is the single, definitive source of information about your data. It contains the essential fields and behaviours of the data you're storing. Generally, each model maps to a single database table. Each model is a Python class that subclasses django.db.models.Model. Each attribute of the model represents a database field. (Django Software Foundation., 2020)

Template

Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. (Django Software Foundation, 2020)

The RUP uses four elements to describe processes
Workers ➔ people with roles
Activities ➔ tasks that need to be done
Artefacts ➔ use cases , plans, code, test case, test result
Workflows ➔ sequence of activities

Since I am only the developer the worker is always me 'Kushal Bhattarai' whether it be database developer, frontend developer , backend developer , tester. The activities that need to be done are function based using the artefacts like of use case diagram.

The workflows of RUP in each iterations are:
Business Modeling:
Requirements,
Analysis and design
Implementation
Test
Deployment
(Jacko & Stephanidis, 2003)

CHAPTER 4: TESTING AND ANALYSIS

4.1 TEST PLAN

4.1.1 UNIT TESTING, TEST PLAN

Features to be tested

1. Login
2. Register
3. Adding Products
4. Updating products
5. Adding Auction time
6. Ending Auction Products
7. Adding to carts
8. Buying Products from cash on delivery method
9. Ordering
10. Buying Products from stripe
11. Searching
12. Reading notification
13. Buying
14. Filing Dispute
15. Ordering won auction products
16. Adding conversation
17. Logging out

Test type Unit testing

Test objective The test objectives are to verify the Functionality of the system.

Test environment Anaconda environment with shell using converge

Test deliverables

Before testing phase

1. Test plans document.

2. Test cases
3. Test Design specifications.

During the testing

1. Test Tool
2. Simulators.
3. Test Data
4. Error logs

After the testing cycles is over

1. Test Results/reports

4.1.2 SYSTEM TEST PLAN

Features to be tested

18. Login
19. Adding Products
20. Updating products
21. Adding Auction time
22. Ending Auction Products
23. Adding to carts
24. Buying Products from cash on delivery method
25. Ordering
26. Buying Products from stripe
27. Searching
28. Reading notification
29. Buying
30. Filing Dispute
31. Ordering won auction products
32. Adding conversation
33. Logging out
34. Updating profile

Test type System testing

Test objective The test objectives are to verify the system.

Test environment

Objectives , Expected output , Gained output and test result will be created to store the testing document while the testing is done in debug mode.

TEST deliverables

Before testing phase

Test plans document.

During the testing

Test Data stored in form of table

After the testing cycles is over

Test Results/report

4.2 UNIT TESTING

The plan is to use Using coverage to test the system. First models are to be tested and the functions of the models need to be tested. Then the views are tested. The test that were run using shell.

```
(project) C:\Users\Dell\Desktop\FYP\django_project>coverage run manage.py test website
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
```

Figure 35: running unit test in system

```
----------------------------------------------------------------------
Ran 17 tests in 26.131s

OK
Destroying test database for alias 'default'...
```

Figure 36: test completed

```
Terminal:  Local  Local (2)  +
  File "C:\Users\Dell\Desktop\FYP\django_project\website\views.py", line 223, in save_bid
    max = Bidder.objects.get(product_id_id=request.POST.get('product_id')).first()
  File "c:\users\dell\anaconda3\envs\project\lib\site-packages\django\db\models\manager.py", line 82, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
  File "c:\users\dell\anaconda3\envs\project\lib\site-packages\django\db\models\query.py", line 406, in get
    raise self.model.DoesNotExist(
website.models.Bidder.DoesNotExist: Bidder matching query does not exist.

----------------------------------------------------------------------
Ran 16 tests in 5.959s

FAILED (errors=1)
Destroying test database for alias 'default'...

(project) C:\Users\Dell\Desktop\FYP\django_project>
```

Figure 37: testing error of save bid because of

```
if request.method == 'POST':
    if int(request.POST['bid_amount']) <= Product.objects.get(id=request.POST['product_id']).minimum_price:
        messages.error(request, "bid amount is low")
        return HttpResponseRedirect(request.META.get('HTTP_REFERER', '/'))

    max = Bidder.objects.first(product_id_id=request.POST.get('product_id')).first()
    if int(request.POST['bid_amount']) >= max.max_amount:
        pass
    else:
        messages.error(request, "bid amount must be higher than the latest bid")
        return HttpResponseRedirect(request.META.get('HTTP_REFERER', '/'))
```

Figure 38 :error code

```
1   from django.test import TestCase
2
3   # Create your tests here.
4   from django.urls import reverse
5
6   from users.forms import UserRegisterForm
7   from .models import *
8
9   from users.models import Profile, Notify
10
11
12  class RegisterProfile2Test(TestCase):
13
14      def create_user(self):
15          return User.objects.create(username='kushal')
16
17      def test_create_user(self):
18          a = self.create_user()
19          b = Profile.objects.last()
20          self.assertTrue(isinstance(a, User))
21          self.assertTrue(isinstance(b, Profile))
22          self.assertEqual(a, b.user)
23
```

Figure 39:unit test code for testing user creating

Here testing of creating users and profile are done.

```python
class RegisterProfileTest(TestCase):

    def create_user(self):
        return User.objects.create(username='kushal')

    def test_create_user(self):
        a = self.create_user()
        b = Profile.objects.last()
        self.assertTrue(isinstance(a, User))
        self.assertTrue(isinstance(b, Profile))
        self.assertEqual(a, b.user)

    def test_object_name(self):
        user = self.create_user()

        b = Profile.objects.last()
        expected_object_name = f'{user.username} Profile'
        self.assertEquals(expected_object_name, str(b))
        print(expected_object_name)
```

Figure 40: register user and checking object name

```python
class AddProductUserTest(TestCase):

    def create_shop(self):
        return Product.objects.create(id=33,
                                      product_name="abc2", category="Electronics", minimum_price="12", status="shop",
                                      image="abc.png",
                                      description="abc2"
                                      )

    def create_auction(self):
        return Product.objects.create(id=23,
                                      product_name="abc", category="Electronics", minimum_price="12", status="auction",
                                      image="abc.png",
                                      description="abc"
                                      )

    def test_get_image(self):
        filename = "abc.png"
        path = "auction_system/image_{0}_{1}".format(str(time()), filename)
        created_path = getImage(self, filename)
        self.assertEqual(path, created_path)

    def test_create_shop(self):
        a = self.create_shop()

        self.assertTrue(isinstance(a, Product))

    def test_create_auction(self):
        a = self.create_auction()
        self.assertTrue(isinstance(a, Product))
```

Figure 41: Testing adding product in both shop and auction

Here the testing of creating shop products and auctions

```
88  class HomeViewTest(TestCase):
89
90      def test_home(self):
91          url = reverse("home")
92          resp = self.client.get(url)
93          print(resp, 'a')
94          self.assertEqual(resp.status_code, 200)
95
96      def test_website_home(self):
97          url = reverse("website-home")
98          resp = self.client.get(url)
99          print(resp, 'b')
100         self.assertEqual(resp.status_code, 200)
101
102
103 class ShopViewTest(TestCase):
104     def test_shop(self):
105         url = reverse("shop")
106         resp = self.client.get(url)
107         print(resp, 'c')
108         self.assertEqual(resp.status_code, 200)
109
```

Figure 42: HomeView and ShopView test without login

The test of home, website-home, shop without login.

```python
class AuctionViewTest(TestCase):

    def test_auction(self):
        url = reverse("auction")
        resp = self.client.get(url)
        print(resp, 'd')
        self.assertEqual(resp.status_code, 200)


class ExpiredHomeViewTest(TestCase):

    def test_expiredHome(self):
        url = reverse("expired")
        resp = self.client.get(url)
        print(resp, 'e')
        self.assertEqual(resp.status_code, 200)


class LoginViewTest(TestCase):

    def test_login(self):
        url = reverse("login")
        resp = self.client.get(url)
        print(resp, 'f')
        self.assertEqual(resp.status_code, 200)
```

Figure 43: Auction , expired without login and login testing

```python
class NotificationViewTest(TestCase):

    def test_notification(self):
        url = reverse("viewnotify")
        resp = self.client.get(url)
        print(resp, 'h')
        self.assertEqual(resp.status_code, 302)


class MyBidsViewTest(TestCase):

    def test_mybids(self):
        url = reverse("mybids")
        resp = self.client.get(url)
        print(resp, 'i')
        self.assertEqual(resp.status_code, 302)
```

Figure 44: Notification, and bids without login test

Notification, and bids without login test are being tested.

```python
class LogInTest(TestCase):

    def setUp(self):
        self.credentials = {
            'username': 'testuser',
            'password': 'secret'}
        User.objects.create_user(**self.credentials)

    def test_login(self):
        # send login data
        response = self.client.post('/login/', self.credentials, follow=True)
        # should be logged in now
        print(response, 'j')
        self.assertTrue(response.context['user'].is_authenticated)
```

Figure 45: Login testing with credentials and with login url

Login in tested here while setting credentials.

```python
class AllTestViewTest(TestCase):

    def setUp(self):
        self.credentials = {
            'username': 'testuser',
            'password': 'secret'}
        User.objects.create_user(**self.credentials)

    def test_login(self):
        # send login data
        response = self.client.post('/login/', self.credentials, follow=True)
        # should be logged in now

        self.assertTrue(response.context['user'].is_authenticated)
        url = reverse('home')
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("website-home")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("expired")
        resp = self.client.get(url)
```

Figure 46: testing login , home website home url in same session after logged in

Testing of expired, home, website-home and authenticated is done.

```
        self.assertEqual(resp.status_code, 200)
        url = reverse("shop")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("auction")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("home")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("viewnotify")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("mybids")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("all-product")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("won")
        resp = self.client.get(url)
```

Figure 47:testing in same logged in session

Testing of shop, auction, home, viewnotify,mybids, all-prodcut,won is done here.

```
        url = reverse("profile")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("post-create")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("shopcreate")
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        Product.objects.create(id=1,
                               product_name="abc2", category="Games", minimum_pr
                               image="abc.png",
                               description="abc2"
                               )
        Product.objects.create(id=2,
                               product_name="abc23", category="Games", minimum_p
                               status="auction",
                               image="abc3.png",
                               description="abc23"
                               )
        Product.objects.create(id=3,
                               product_name="abc23", category="Games", minimum_p
                               status="auction",
                               image="abc3.png",
```

Figure 48:same logged in session with different url and creating product test

Adding product testing in login.

```
        Seller.objects.create(product_id_id=1, user_name_id=1)
        Seller.objects.create(product_id_id=2, user_name_id=1)
        Seller.objects.create(product_id_id=3, user_name_id=1)

        url = reverse("product_detail", args=[1])
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("updateAcc", args=[1])
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("search")
        resp = self.client.get(url, data={'data': 'category', 'q': 'Games'})

        self.assertEqual(resp.status_code, 200)
        url = reverse("search")
        resp = self.client.get(url, data={'data': 'product', 'q': 'abc2'})

        self.assertEqual(resp.status_code, 200)
        url = reverse("search")
        resp = self.client.get(url, data={'q': ''})

        self.assertEqual(resp.status_code, 200)
        url = reverse("addtime", args=[2])

        resp = self.client.post(url, data={'time': 'twelve', 'pk': 2})
```

Figure 49: testing other urls in same logged in session

Testing of viewing of product. Updating account is tested, searching category games as well as abc2 product as well as empty search. The testing of adding time is also done.

```
        resp = self.client.post(url, data={'time': 'twelve', 'pk': 2})

        self.assertEqual(resp.status_code, 302)
        url = reverse("addtime", args=[2])
        resp = self.client.post(url, data={'time': 'one', 'pk': 2})

        self.assertEqual(resp.status_code, 302)
        url = reverse("addtime", args=[2])
        resp = self.client.post(url, data={'time': 'two', 'pk': 2})

        self.assertEqual(resp.status_code, 302)
        url = reverse("message")
        resp = self.client.post(url, data={'p_id': 2, 'convo': "hi please bid faster i need money ASAP"})

        self.assertEqual(resp.status_code, 302)
        url = reverse("update", args=[2])
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)
        url = reverse("updateshop", args=[1])
        resp = self.client.get(url)

        self.assertEqual(resp.status_code, 200)

        url = reverse("addfavourite")
        resp = self.client.post(url, data={'pk': 2})
        self.assertEqual(resp.status_code, 302)
```

Figure 50: testing other urls, and updating sending message in same logged in session

Testing of updating sop, add-favourite, orderhistory, addcart, cart remove sending message is done here.

```
        self.assertEqual(resp.status_code, 302)
        url = reverse("order_history")
        resp = self.client.post(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("cart")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("add_cart", args=[1])
        resp = self.client.post(url)
        self.assertEqual(resp.status_code, 302)
        url = reverse("cart")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("cart_remove", args=[1])
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 302)

        url = reverse("logout")
        resp = self.client.get(url)
```

Figure 51: testing urls in the same logged in session and login out

```
self.credentials2 = {
    'username': 'testusess',
    'password': 'secretss',
    'is_staff': True,
    'is_superuser': True,

}
User.objects.create_user(**self.credentials2)
resp = self.client.post('/verifylogin/', self.credentials2, follow=True)
self.assertEqual(resp.status_code, 200)
url = reverse("add_cart", args=[1])
resp = self.client.post(url)
self.assertEqual(resp.status_code, 302)
url = reverse("sellDashboard")
resp = self.client.post(url)
self.assertEqual(resp.status_code, 200)
url = reverse("logout")
resp = self.client.get(url)
```

Figure 52: Testing Adding to cart

Testing logged in with superadmin adding cart and sellerdashboard. Logging in with simple credentials and doing save_bid.

```
    self.assertEqual(resp.status_code, 302)
    self.credentialsne = {
        'username': 'testy',
        'password': 'secretss',
    }
    User.objects.create_user(**self.credentialsne)
    resp = self.client.post('/verifylogin/', self.credentialsne, follow=True)
    self.assertEqual(resp.status_code, 200)
    url = reverse("save_bid")
    resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 4000})
    self.assertEqual(resp.status_code, 302)
    url = reverse("save_bid")
    resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 3000})
    self.assertEqual(resp.status_code, 302)
    url = reverse("logout")
    resp = self.client.get(url)
```

Figure 53: Testing bidding

```
self.credentials4 = {
    'username': 't',
    'password': 'secretss',

}
User.objects.create_user(**self.credentials4)
resp = self.client.post('/verifylogin/', self.credentials4, follow=True)
self.assertEqual(resp.status_code, 200)
url = reverse("save_bid")
resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 4000})
self.assertEqual(resp.status_code, 302)
url = reverse("save_bid")
resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 4009})
self.assertEqual(resp.status_code, 302)
url = reverse("save_bid")
resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 6000})
self.assertEqual(resp.status_code, 302)
url = reverse("save_bid")
resp = self.client.get(url, data={'product_id': 2, 'bid_amount': 6000})
self.assertEqual(resp.status_code, 302)
url = reverse("logout")
resp = self.client.get(url)

self.assertEqual(resp.status_code, 302)
```

Figure 54: Test rebidding

```
        self.assertEqual(resp.status_code, 302)
        url = reverse("register")

        resp = self.client.post(url,
                                data={'email': "samanghar@gmail.com", 'username': "saman", 'first_name': "kushal",
                                      'last_name': "bhattarai", 'role': "Admin", 'password1': "asid1234",
                                      'password2': "asid1234", 'is_staff': True}, follow=True)
        self.assertEqual(resp.status_code, 200)
        url = reverse("register")
        resp = self.client.post(url,
                                data={'email': "samanghar3@gmail.com", 'username': "saman3", 'first_name': "kushal",
                                      'last_name': "bhattarai", 'role': "SuperAdmin", 'password1': "asid1234",
                                      'password2': "asid1234", 'is_staff': True}, follow=True)
        self.assertEqual(resp.status_code, 200)
        url = reverse("register")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("register")
        resp = self.client.post(url,
                                data={'email': "samanghar2@gmail.com", 'username': "saman2", 'first_name': "kushal",
                                      'last_name': "bhattarai", 'role': "User", 'password1': "asid1234",
                                      'password2': "asid1234", 'is_staff': False}, follow=True)
        self.assertEqual(resp.status_code, 200)
        resp = self.client.post(url,
                                data={'email': "samanghar@gmail.com", 'username': "saman", 'first_name': "kushal",
                                      'last_name': "bhattarai", 'role': "User", 'password1': "asid1234",
                                      'password2': "asid123", 'is_staff': False}, follow=True)
        self.assertEqual(resp.status_code, 200)
```

Figure 55:Register test or user Admin, SuperAdmin ,Admin and wrong password

Testing of registration of different roles like user, superadmin and admin. Testing of verifylogin function.

```
        self.credentials55 = {
            'username': 'samam3',
            'password': 'asid1234',

        }
        resp = self.client.post('/verifylogin/', self.credentials55, follow=True)
        self.assertEqual(resp.status_code, 200)
        resp = self.client.get('/verifylogin/')
        self.assertEqual(resp.status_code, 302)
        self.credentials4 = {
            'username': 'kushalnew',
            'password': 'kushal1234',

        }
        resp = self.client.post('/verifylogin/', self.credentials4, follow=True)
        self.assertEqual(resp.status_code, 200)
```

Figure 56: Test for verifying login

```
self.credentials5 = {
    'username': 'testusersss',
    'password': 'secretss',

}
User.objects.create_user(**self.credentials5)
resp = self.client.post('/verifylogin/', self.credentials5, follow=True)
self.assertEqual(resp.status_code, 200)
url = reverse("save_bid")
resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 5000})
self.assertEqual(resp.status_code, 302)
url = reverse("winning")
resp = self.client.get(url)
self.assertEqual(resp.status_code, 200)
url = reverse("cart")
resp = self.client.get(url)
self.assertEqual(resp.status_code, 200)
url = reverse("add_cart", args=[1])
resp = self.client.post(url)
self.assertEqual(resp.status_code, 302)
url = reverse("admin-home")
resp = self.client.get(url)
self.assertEqual(resp.status_code, 302)
url = reverse("logout")
resp = self.client.get(url)
```

Figure 57: Testing different urls

Testing savebid, winning, cart, adiing to cart ,admin- home functions.

```
        resp = self.client.post('/verifylogin/', self.credentials2, follow=True)
        self.assertEqual(resp.status_code, 200)
        url = reverse("end", args=[3])
        resp = self.client.post(url)
        self.assertEqual(resp.status_code, 302)
        url = reverse("end", args=[2])
        resp = self.client.post(url)
        self.assertEqual(resp.status_code, 302)
        url = reverse("message")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 302)
        url = reverse("favourites")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("history")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("admin-home")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("winner")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("home")
        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 200)
        url = reverse("logout")
        resp = self.client.get(url)
```

Figure 58: Testing funcitons like end, history, favourites

Testing of ending produts auction, sending empty message. Histpry , favoutire, winner, admin home and home and testing of logout.

```
        self.assertEqual(resp.status_code, 302)
        self.credentialsnew = {
            'username': 'testuses',
            'password': 'secretss',
        }
        response = self.client.post('/verifylogin/', self.credentialsnew, follow=True)

        resp = self.client.get(url)
        self.assertEqual(resp.status_code, 302)
        self.credentials3 = {
            'username': 'test',
            'password': 'secretss',
            'is_staff': True,
        }
        User.objects.create_user(**self.credentials3)
        resp = self.client.post('/verifylogin/', self.credentials3, follow=True)
        self.assertEqual(resp.status_code, 200)
        url = reverse("save_bid")
        resp = self.client.post(url, data={'product_id': 2, 'bid_amount': 5000}, follow=True)
        self.assertEqual(resp.status_code, 200)

        url = reverse("logout")
        resp = self.client.get(url)
```

Figure 59: testing login, and testing if staff can bid or not.

Testing of credentiils and saving bid of staff where the staff cannot bid.

4.3 SYSTEM TESTING

TEST 1	
Action	**Enter invalid username and password**
Expected output	**Error and not being able to login. Redirects to login page**
Actual output	**Error and not being able to login. Redirects to login page**
Test Result	**Pass**

Table 9: Test 1

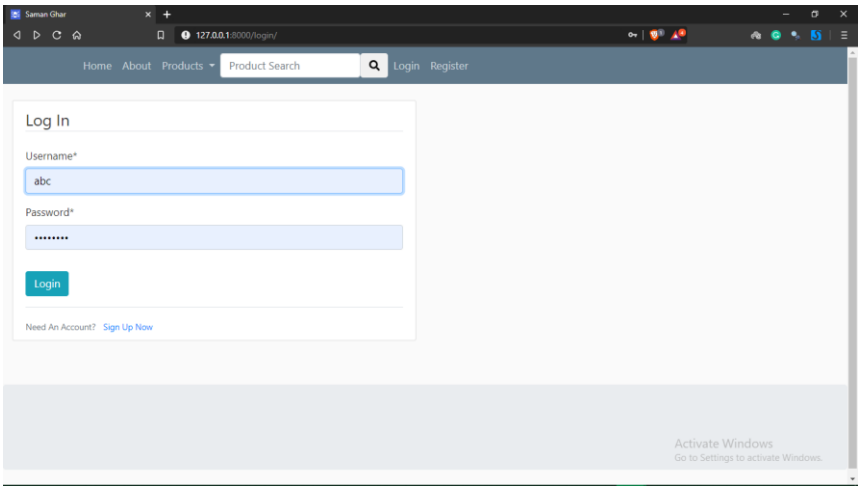


Figure 60: login test

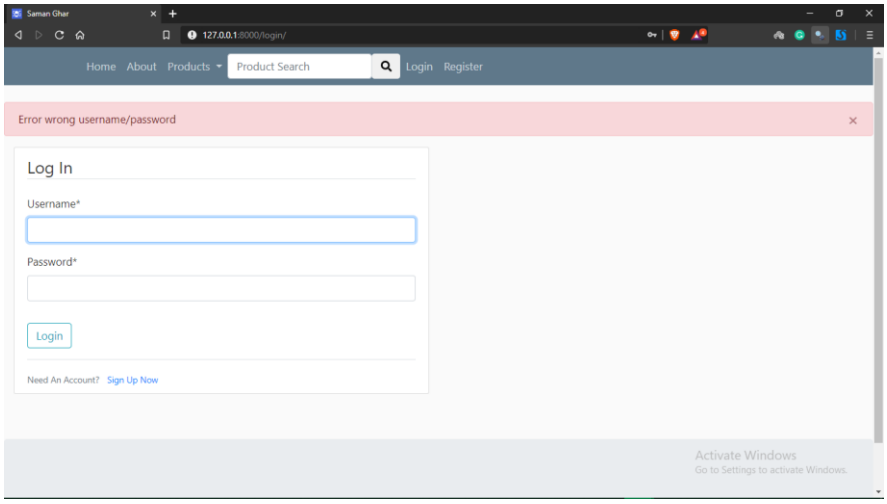


Figure 61: test result

TEST 2	
Action	**Enter deactivated profile username and password.**
Expected output	**Not being able to login and redirect to login page**
Actual output	**Not being able to login and redirect to home page**
Test Result	**Pass**

Table 10: Test 2

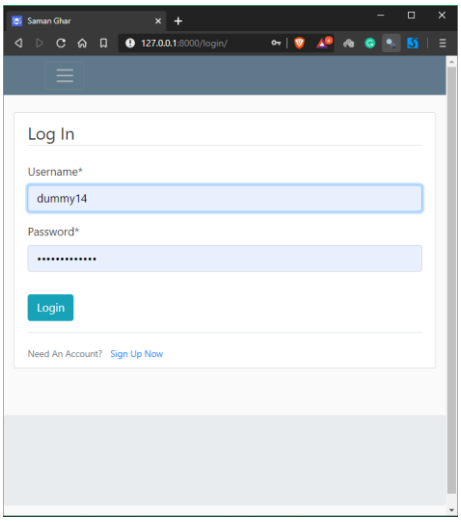


Figure 62: testing 2

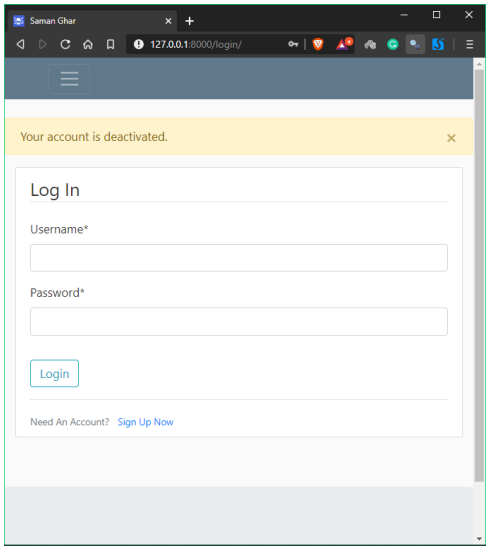


Figure 63: test result -2

TEST 3	
Action	**Enter valid username and password**
Expected output	**Being able to login. Redirects to login page**
Actual output	**Being able to login. Redirects to home page**
Test Result	**Pass**

Table 11:Test 3

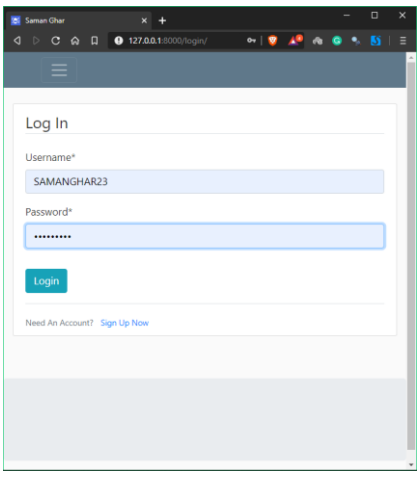


Figure 64 : login form

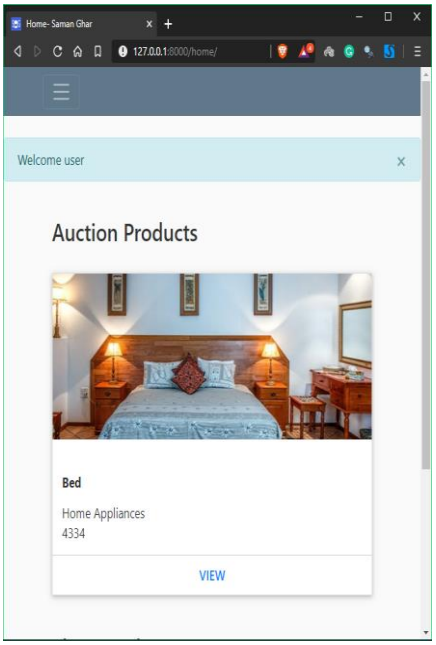


Figure 65:test result

TEST 4	
Action	**Add products**
Expected output	**New Product Added**
Actual output	**New Product Added**
Test Result	**Pass**

Table 12: Test 4

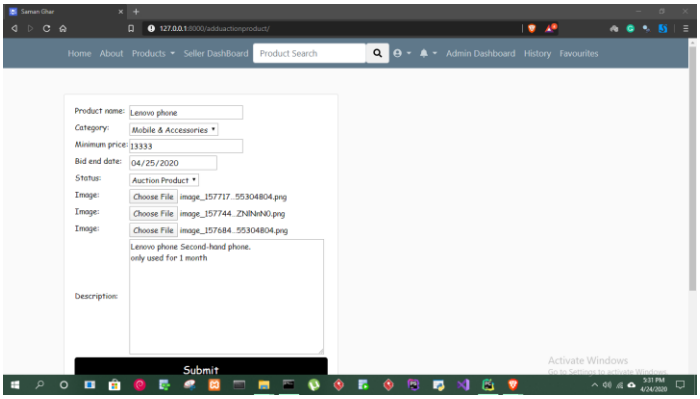


Figure 66: add product form

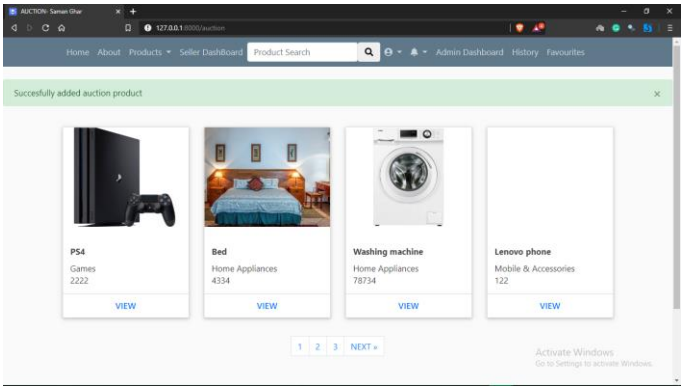


Figure 67:added products

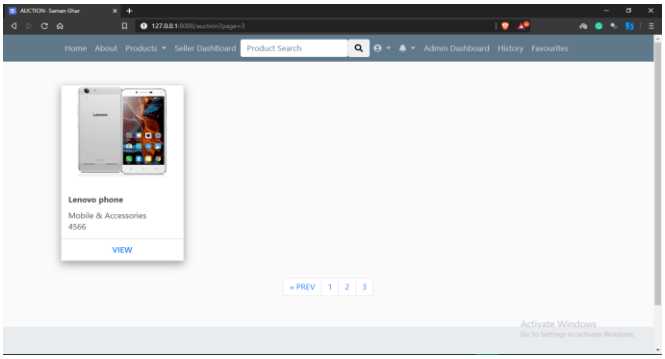


Figure 68:displayed new products

TEST 5	
Action	**Update products**
Expected output	**Product updated**
Actual output	**Product updated**
Test Result	**Pass**

Table 13: Test 5

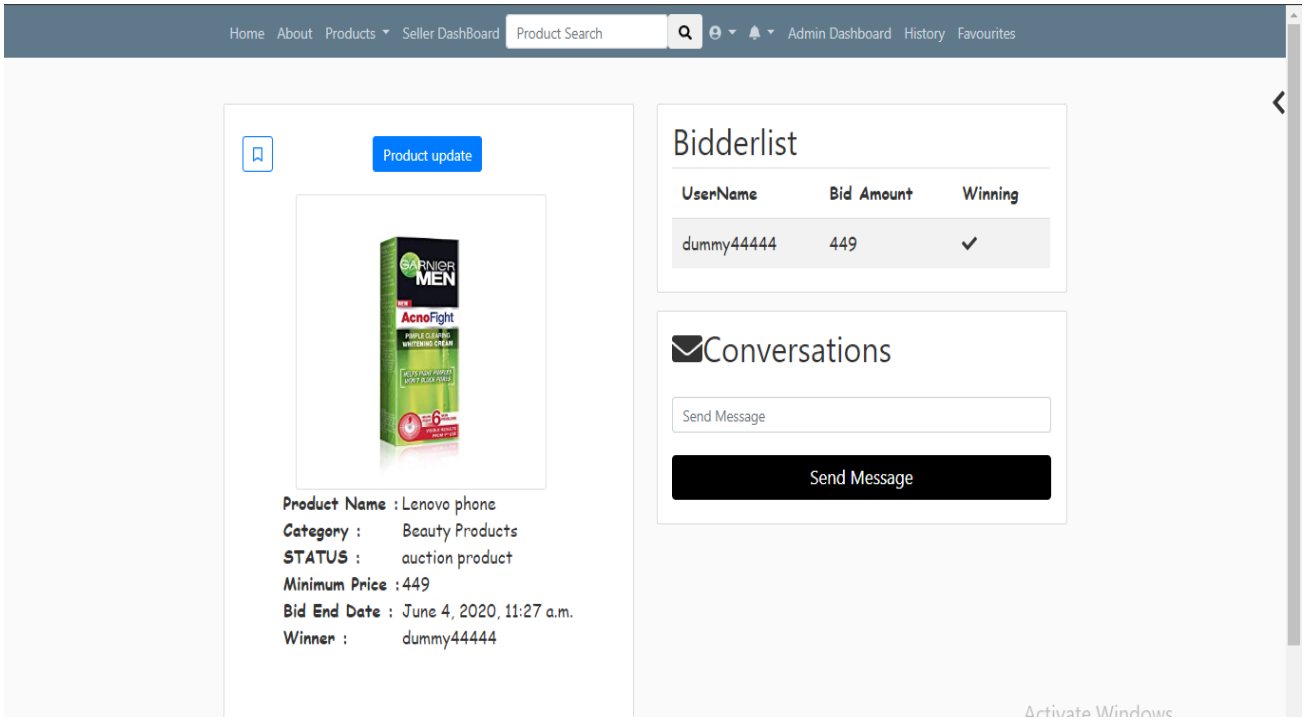


Figure 69:Product update 1

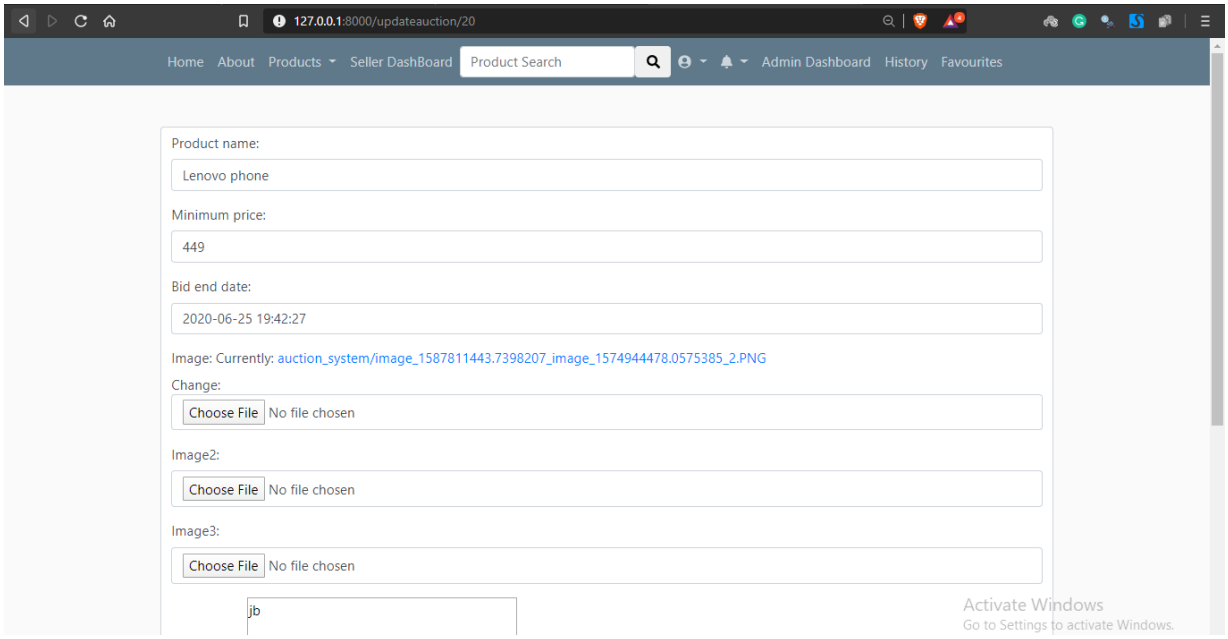


Figure 70:Product update 2

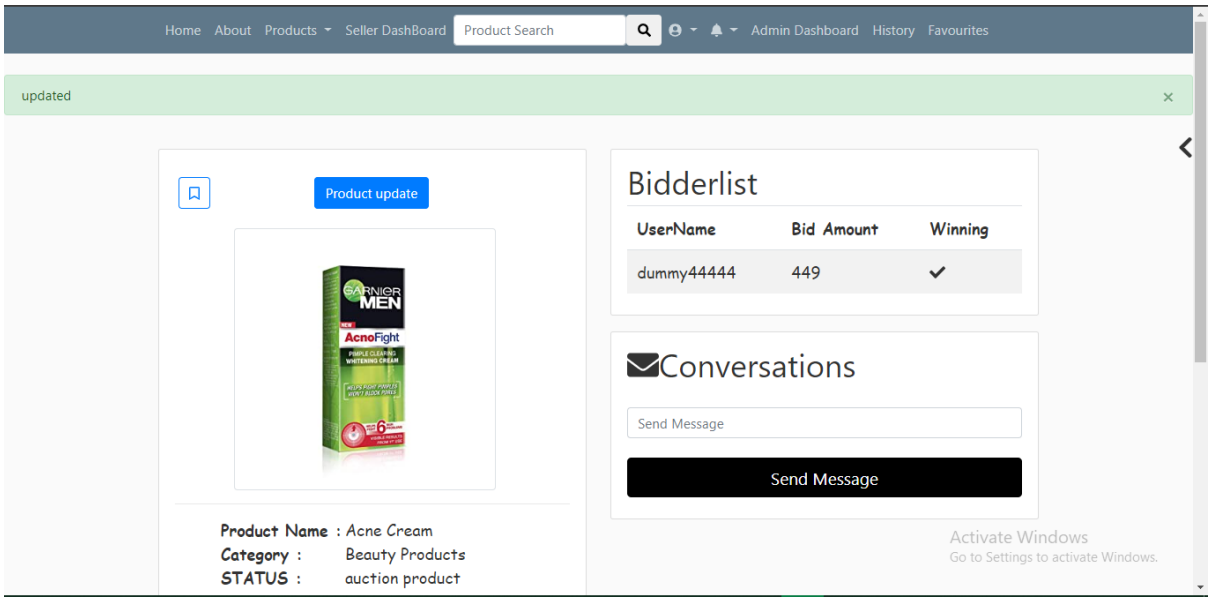


Figure 71:Product update 3

TEST 6	
Action	**Add Auction time**
Expected output	**Auction time added**
Actual output	**Auction time added**
Test Result	**Pass**

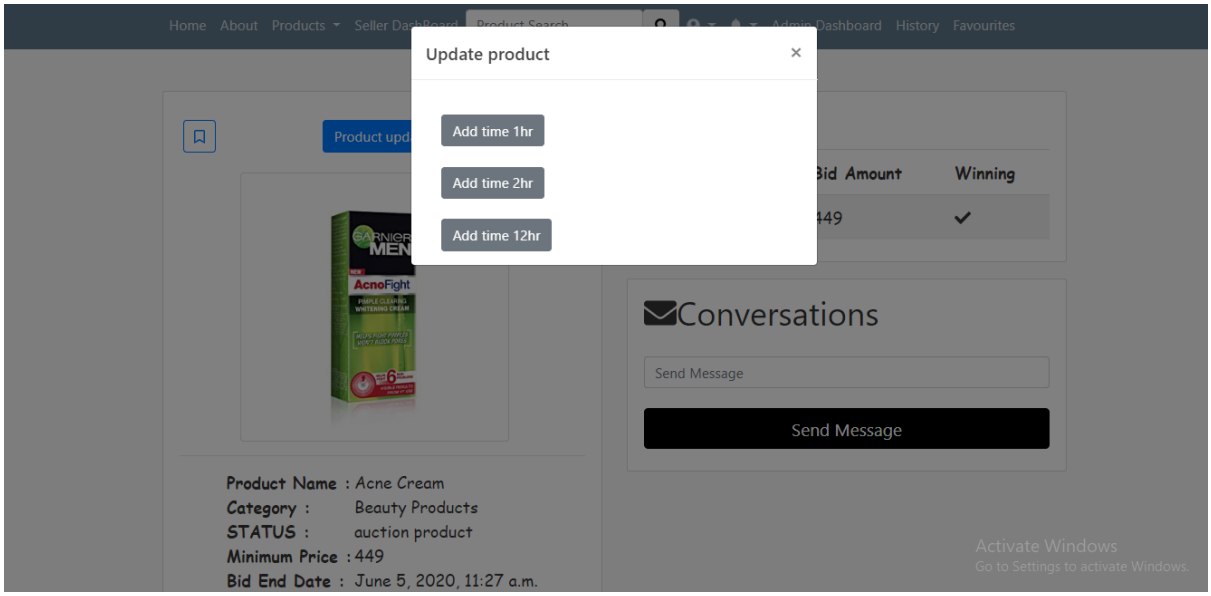


Figure 72:add 12 hr

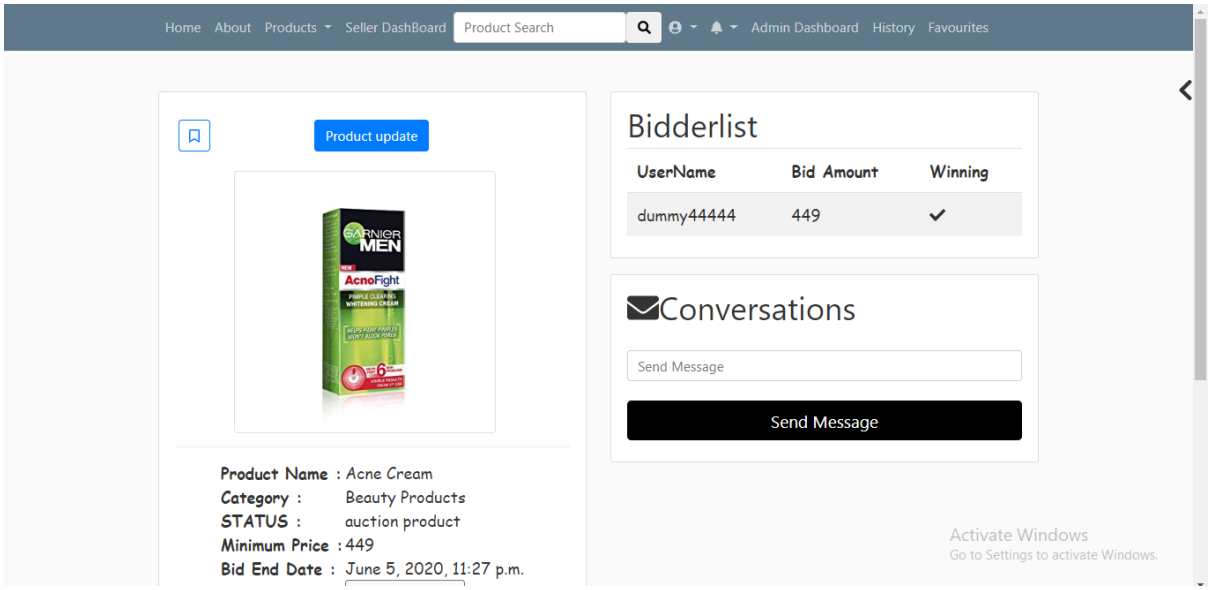


Figure 73: added 12 hr 11:27 am to 11:27 pm

TEST 7	
Action	**End auction products**
Expected output	**Auction ended**
Actual output	**Auction ended**
Test Result	**Pass**

Table 14:Test 6

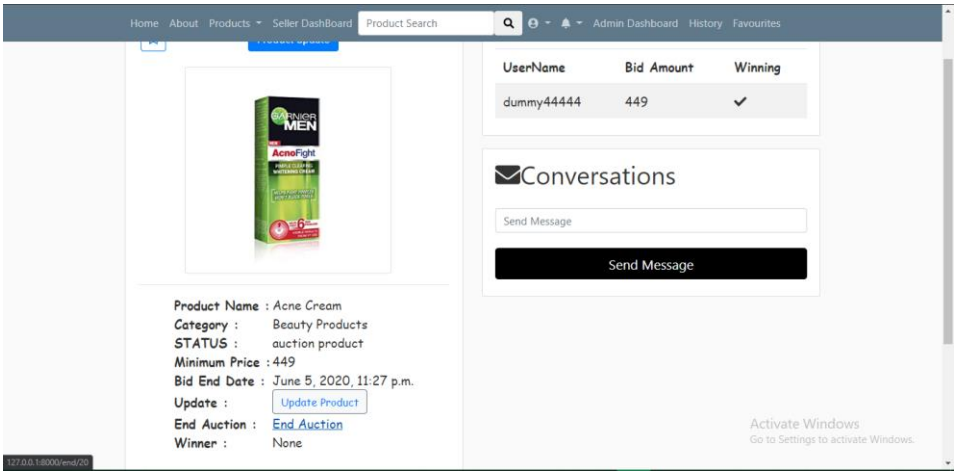


Figure 74: end auction

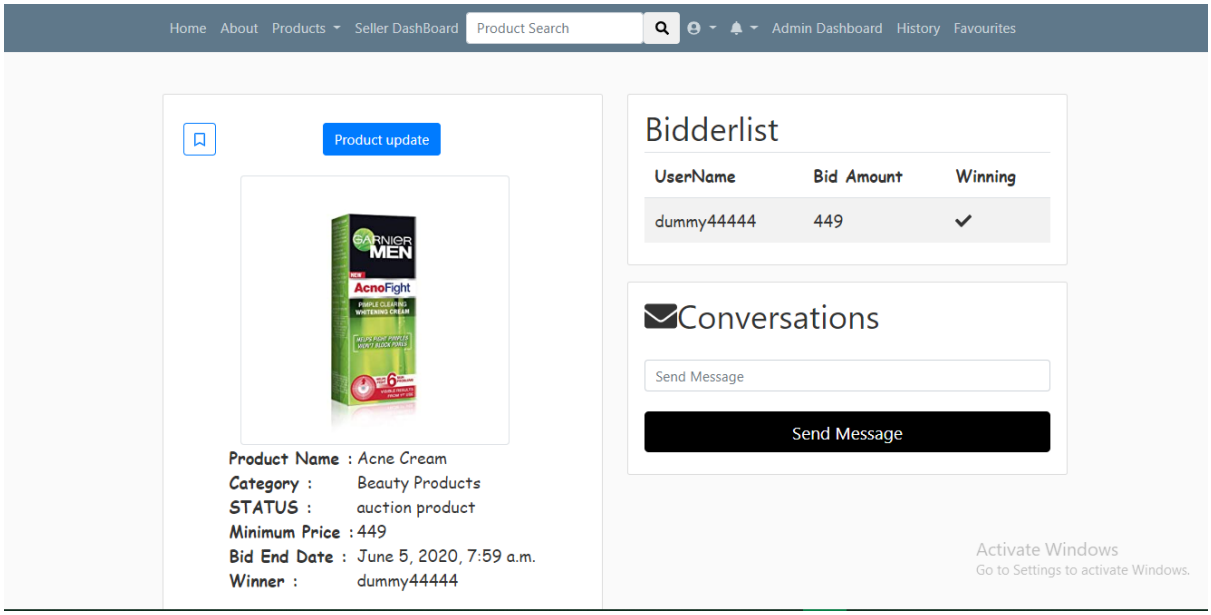


Figure 75:Auction ended

TEST 8	
Action	**Add to cart products**
Expected output	**Product added to cart**
Actual output	**Product added to cart**
Test Result	**Pass**

Table 15: Test 8

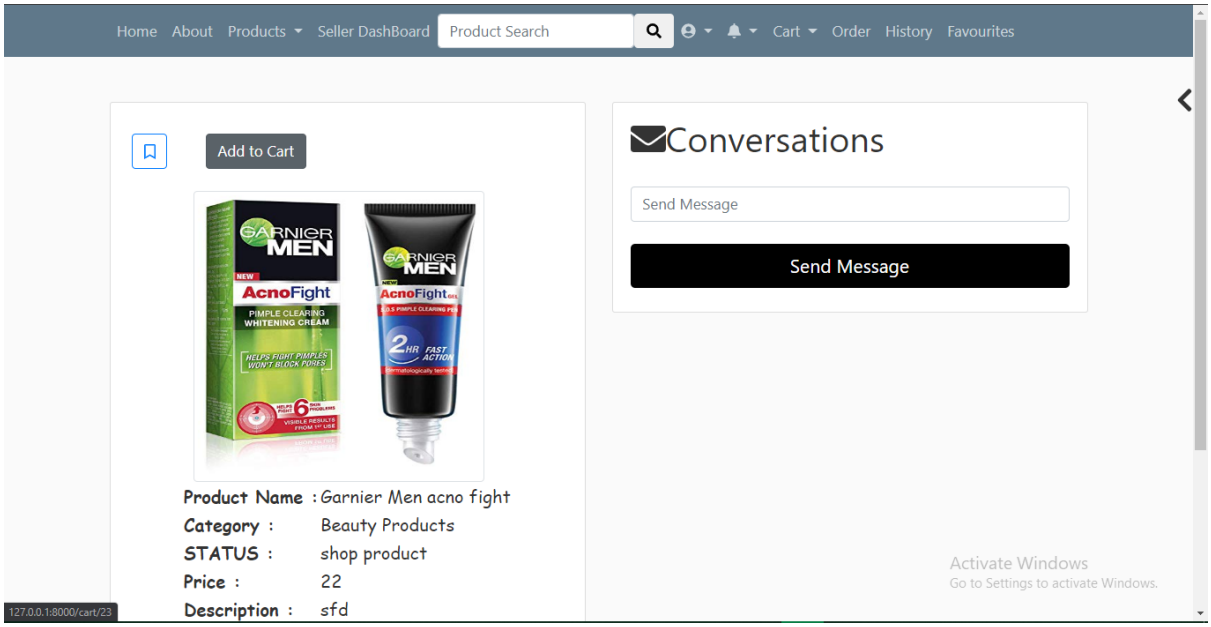


Figure 76: Add to Cart

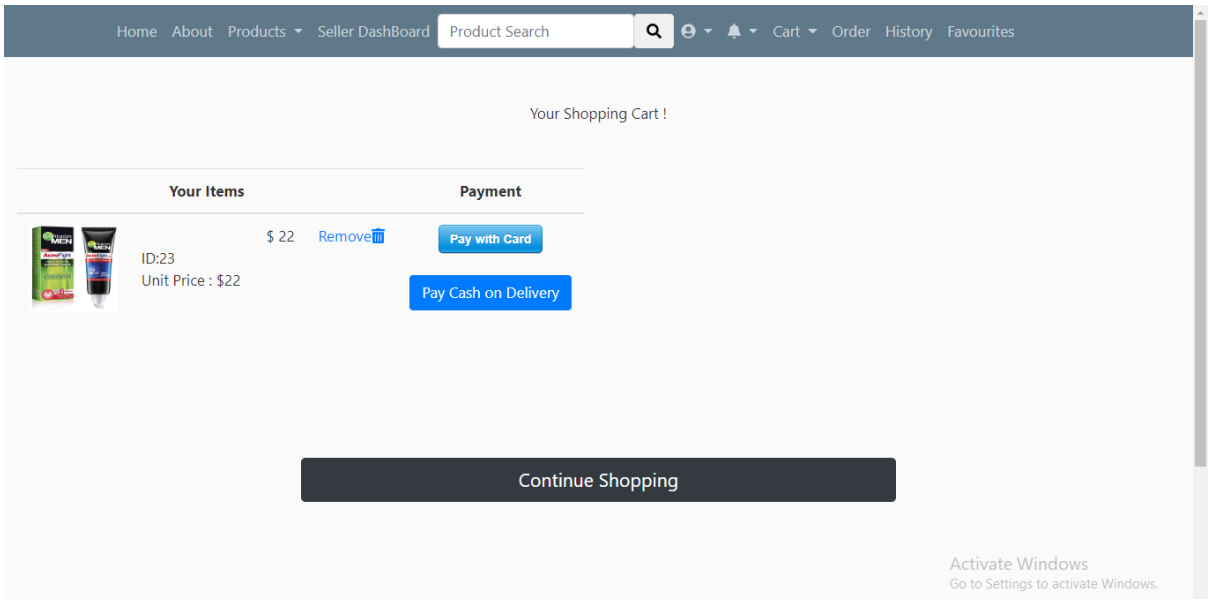


Figure 77: Added to cart

TEST 9	
Action	**Payment by cash**
Expected output	**Order created**
Actual output	**Order created**

Test Result	**Pass**

Figure 78: Test 9

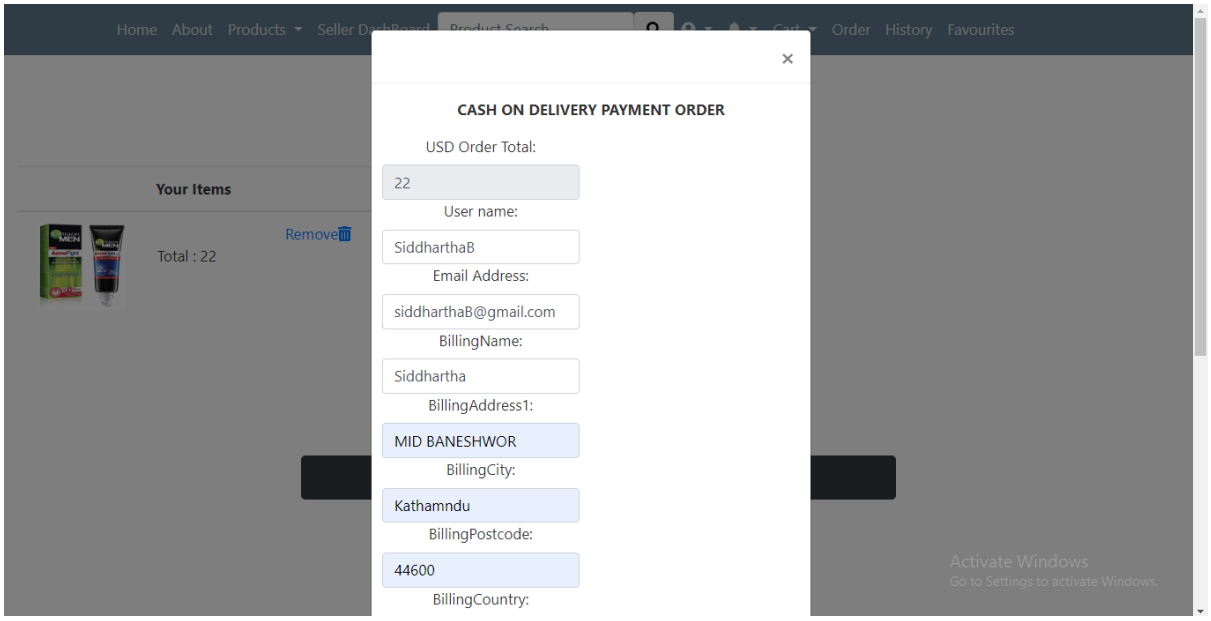


Figure 79: cart cash on delivery

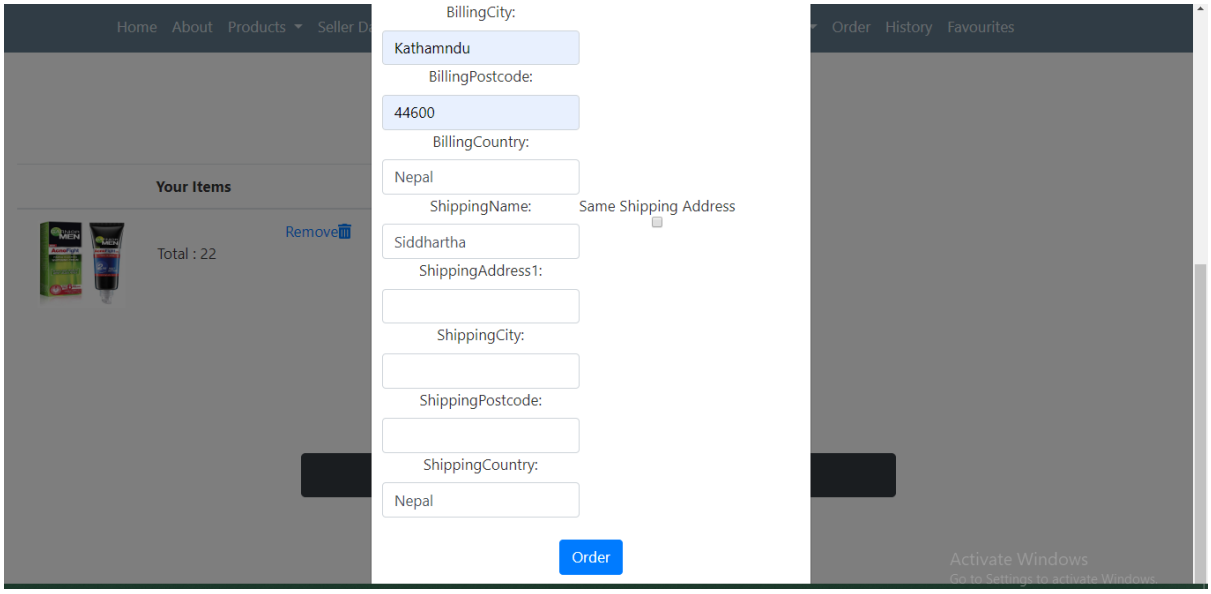


Figure 80: filling form

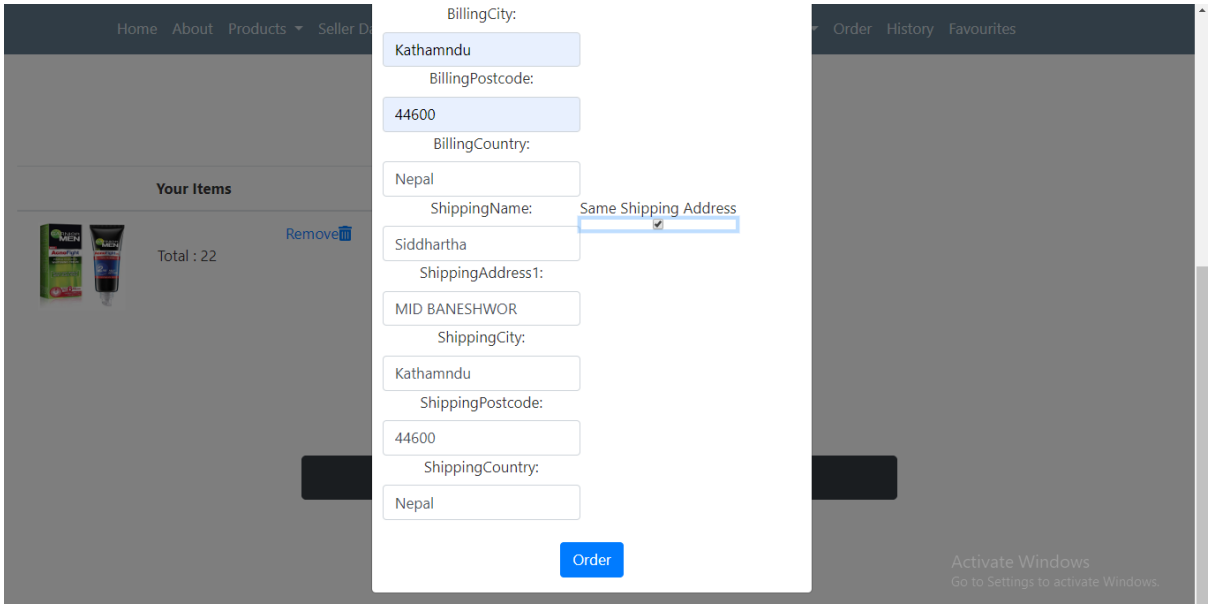


Figure 81:From filled

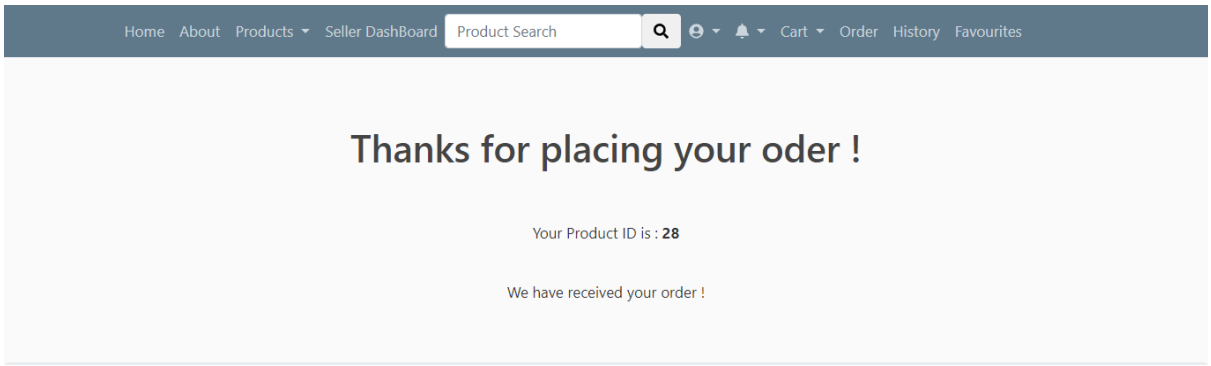


Figure 82:Order placed

TEST 10	
Action	**View order detail**
Expected output	**View order detail**
Actual output	**Viewed order details**
Test Result	**Pass**

Table 16:Test 10

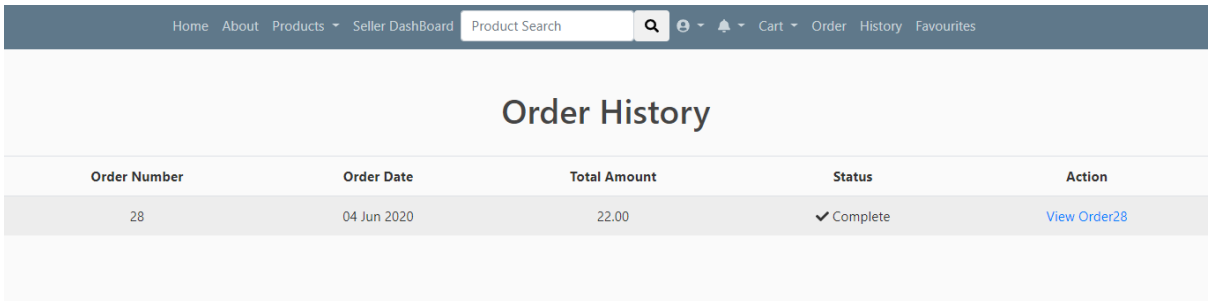


Figure 83:Order page

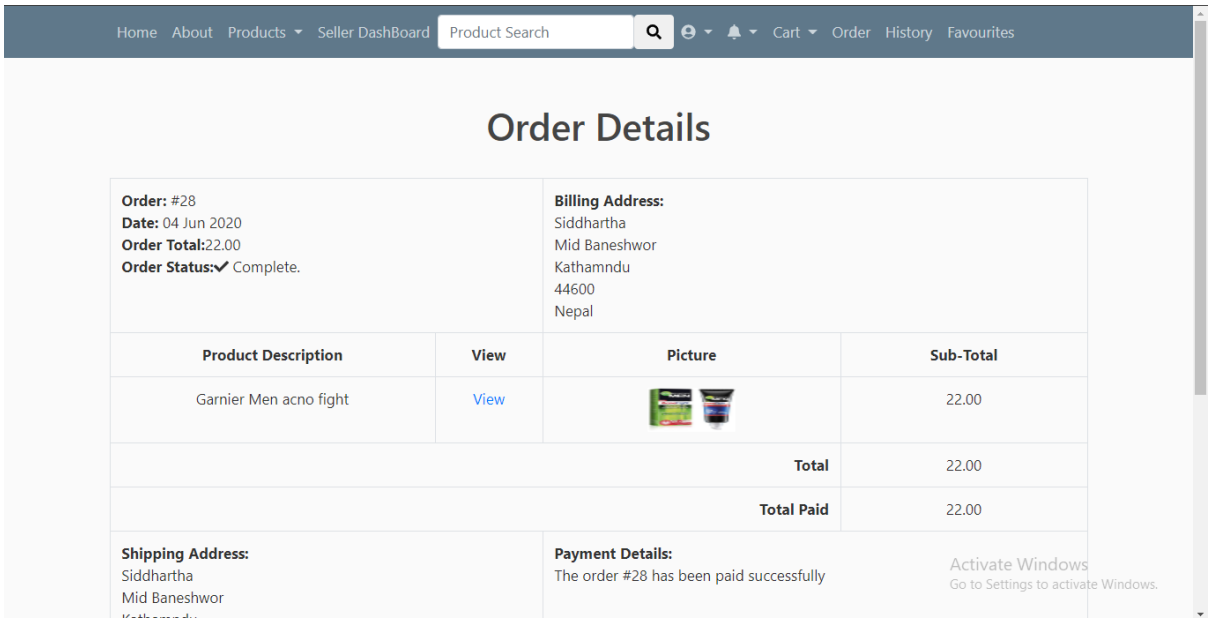


Figure 84: Order detail page

TEST 11	
Action	**Add shop product**
Expected output	**Added shop product**
Actual output	**Added shop product**
Test Result	**Pass**

Table 17: Test 11

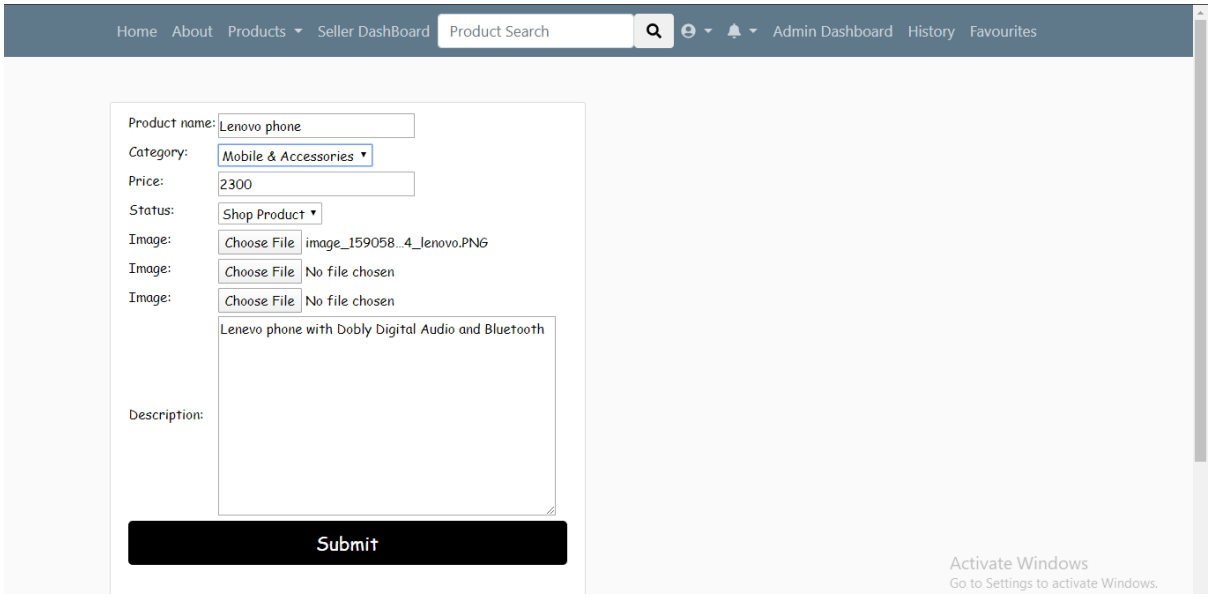


Figure 85:Adding shop products

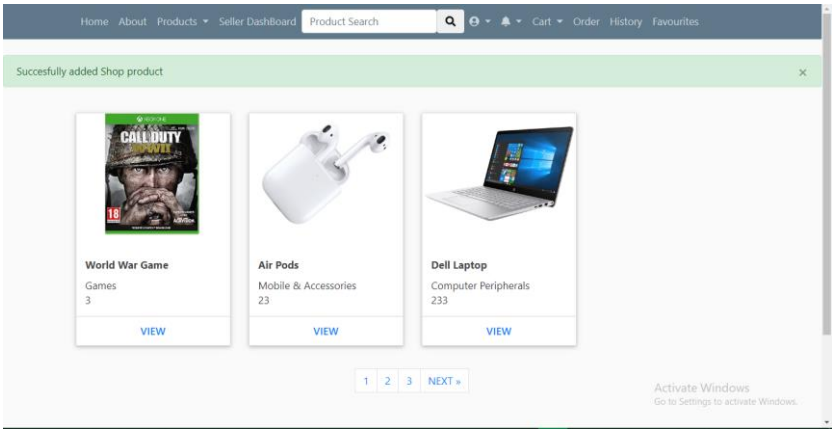


Figure 86: Successfully add shop product

TEST 12	
Action	**Searching product**
Expected output	**Searching Product displays**
Actual output	**Products displays**
Test Result	**Pass**

Table 18:Test 12

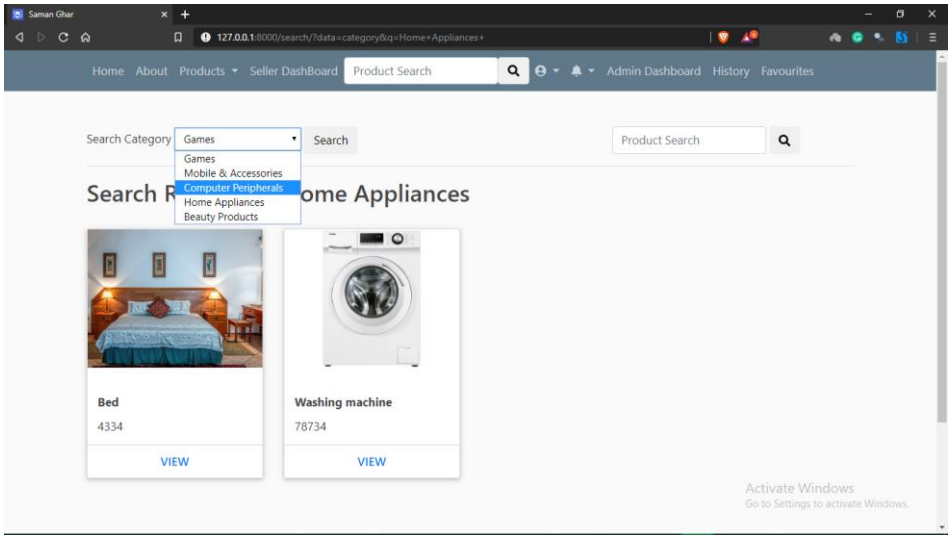


Figure 87:Search category products

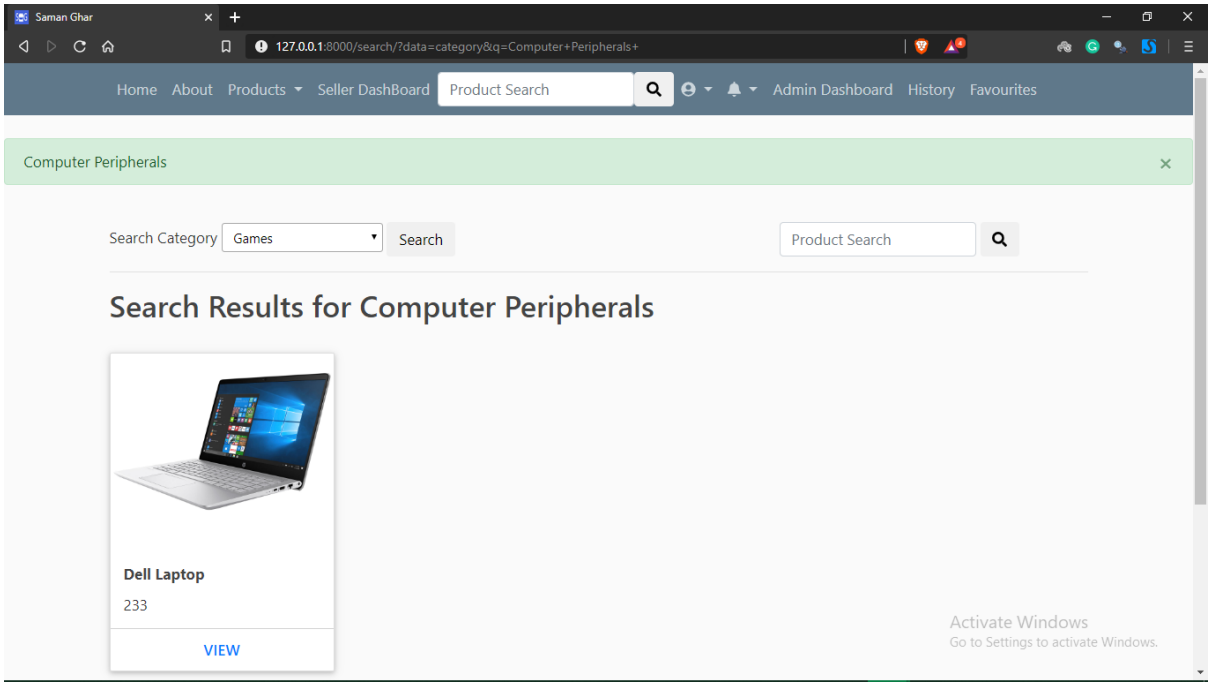


Figure 88:search

TEST 13	
Action	**Read Notification**
Expected output	**Read notification.**
Actual output	**Notification succsfully read.**
Test Result	**Pass**

Table 19: TEST 13

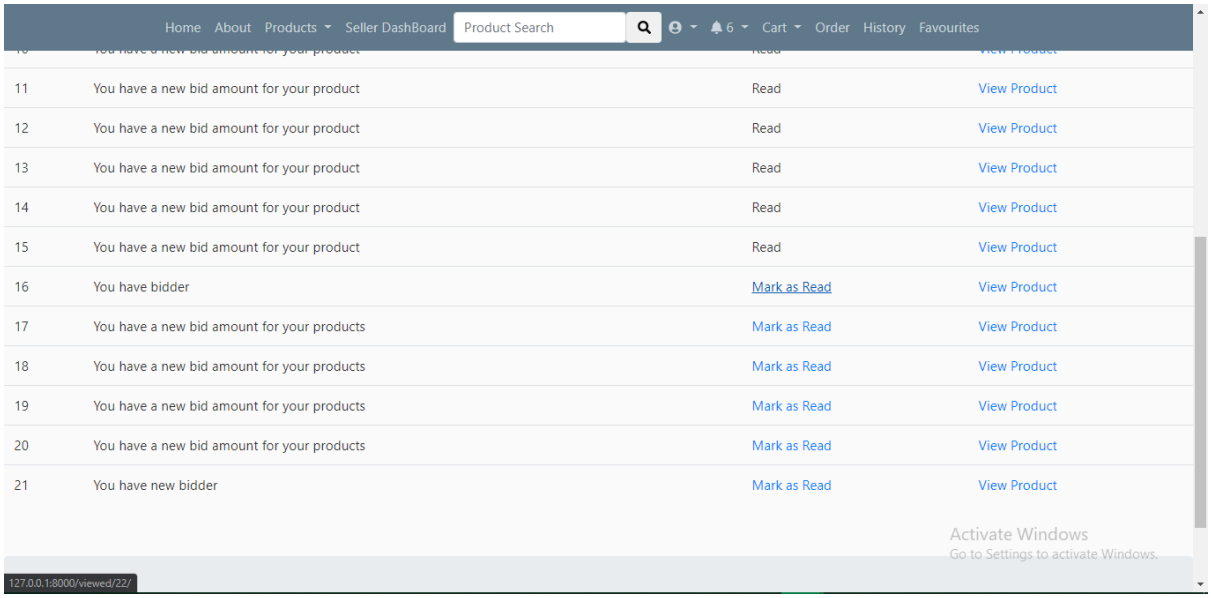


Figure 89: View notification

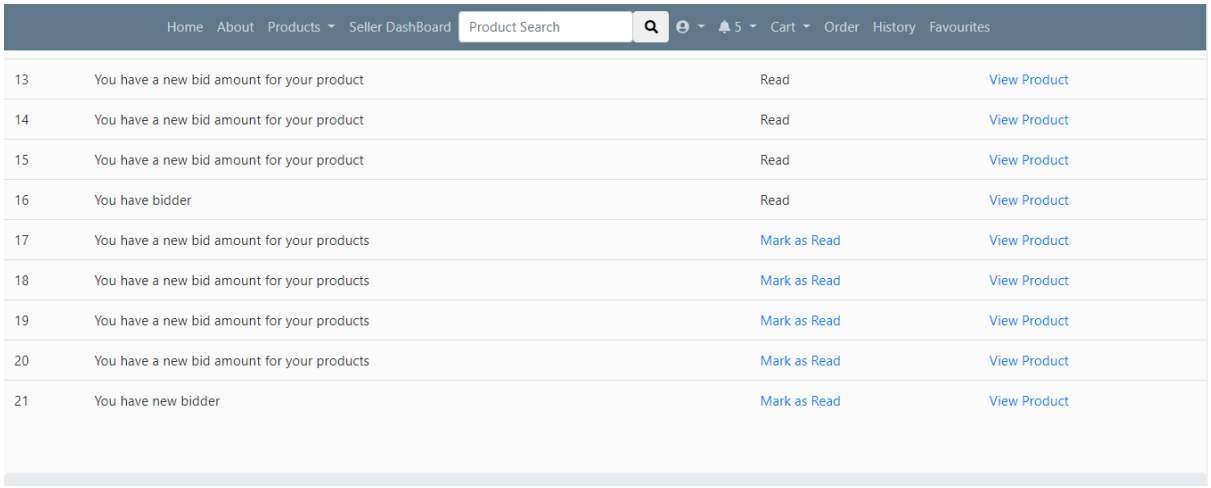


Figure 90: Viewed notification

TEST 14	
Action	**File a dispute**
Expected output	**File a dispute**
Actual output	**Dispute filed**
Test Result	**Pass**

Table 20: TEST 14

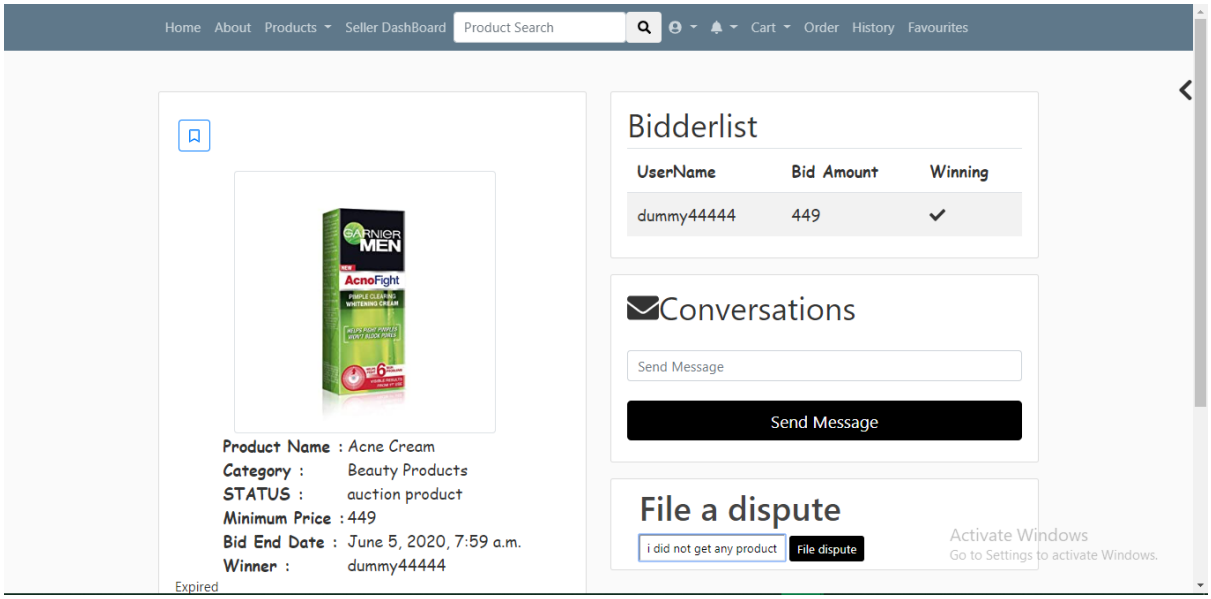
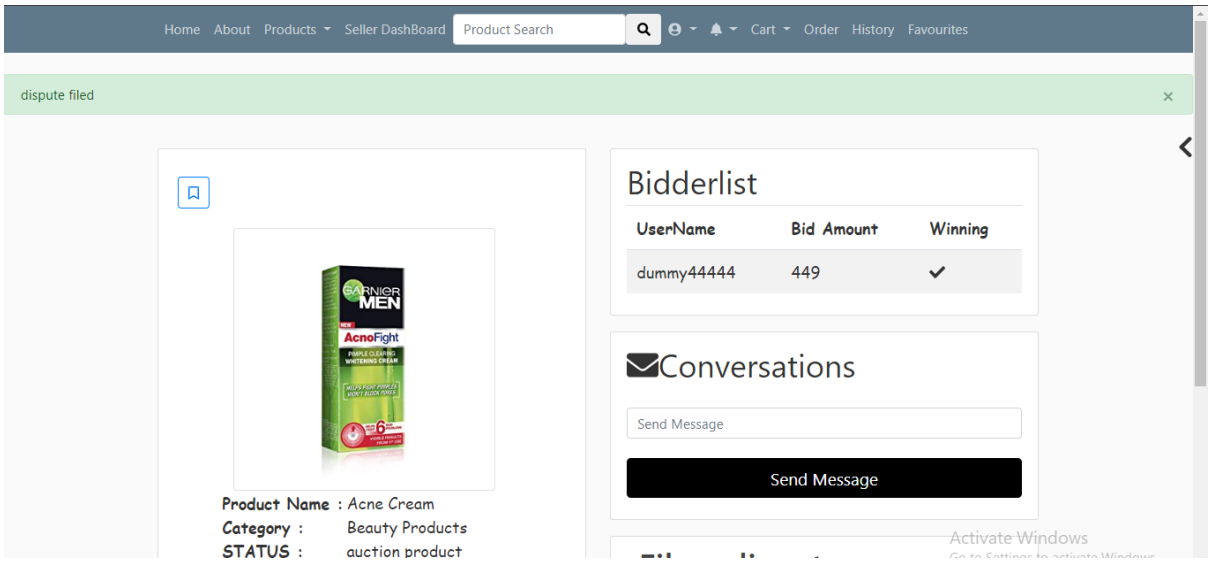


Figure 91: dispute



TEST 15	
Action	**Won Auction order**
Expected output	**Order auctions items**
Actual output	**Order succesfully**
Test Result	**Pass**

Table 21: Test 15

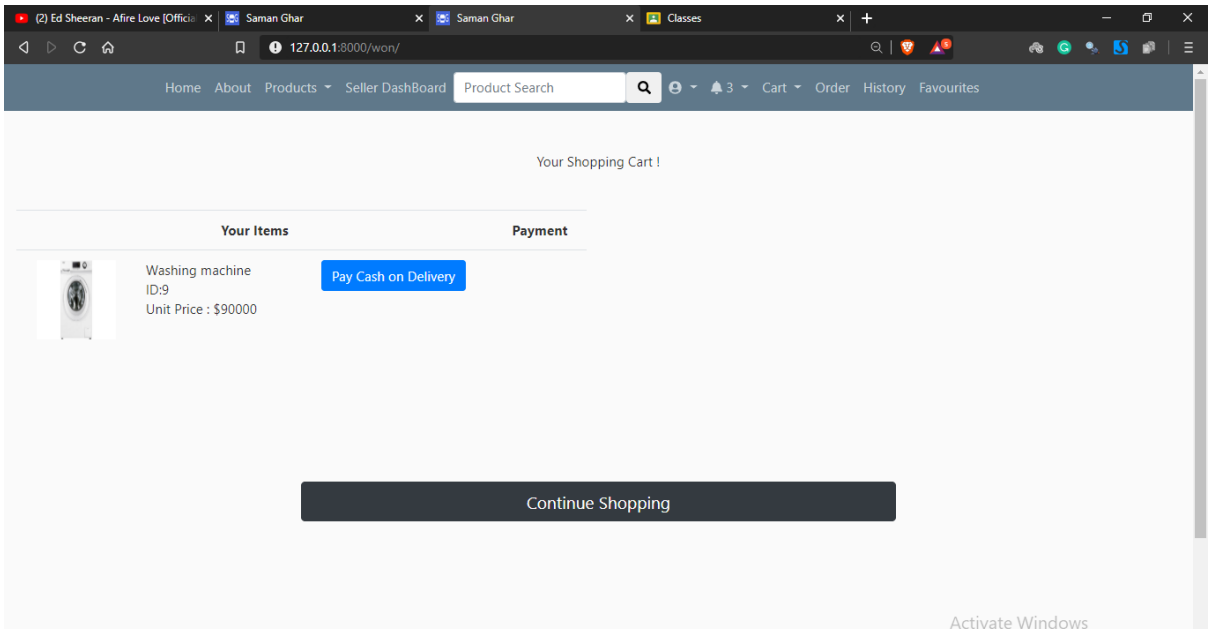


Figure 92: won auctions

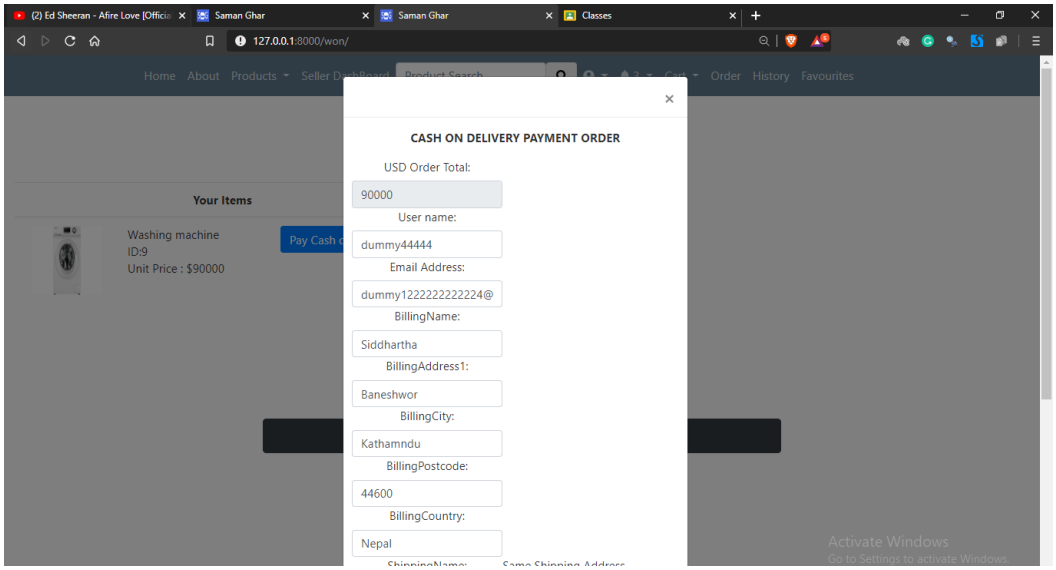


Figure 93:Filling form

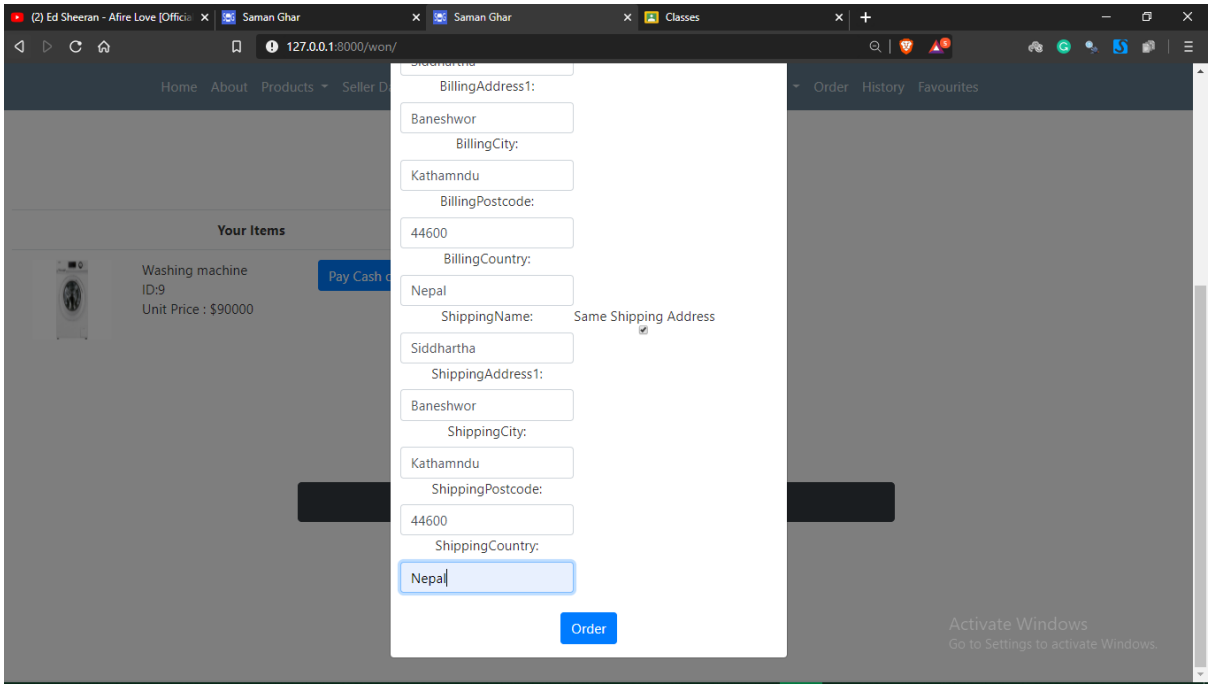


Figure 94: filling form

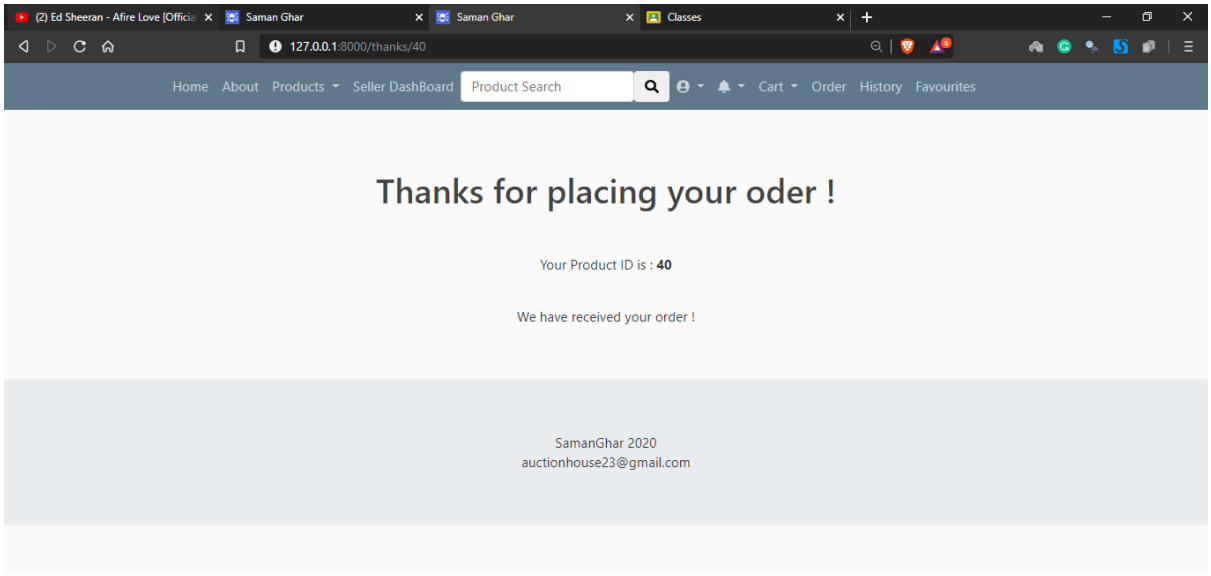


Figure 95:successfully order place

TEST 16	
Action	**Adding conversation**
Expected output	**Conversation displayed**
Actual output	**Conversation is displayed**
Test Result	**Pass**

Table 22: Test 16

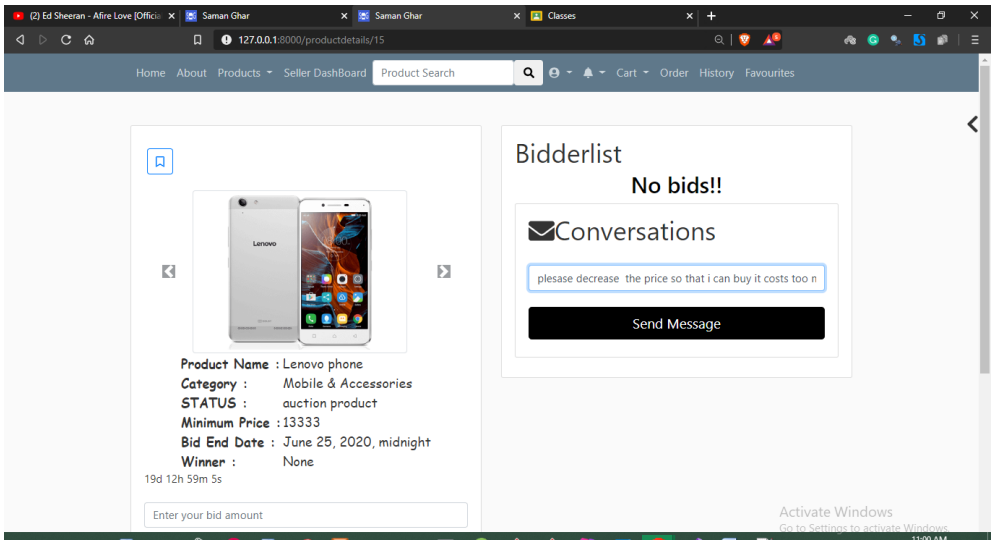


Figure 96: conversation

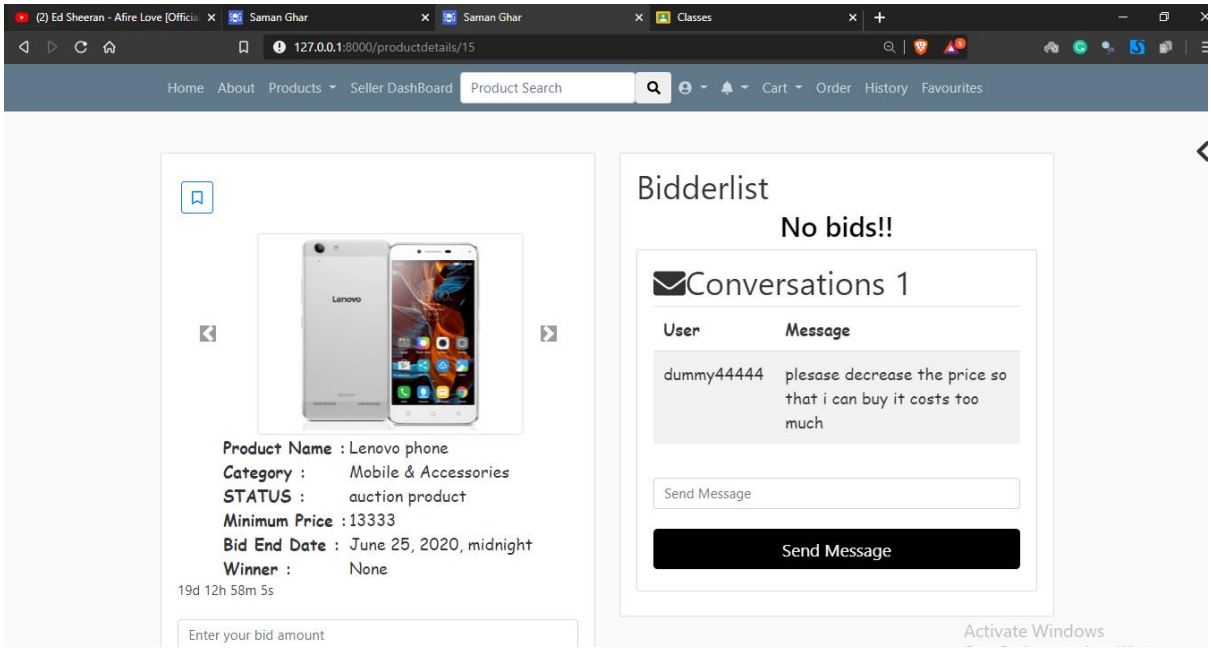


Figure 97: added converdation

TEST 17	
Action	**Logging out**
Expected output	**Successfully log out**
Actual output	**Logged out successfully**
Test Result	**Pass**

Table 23: Test 17

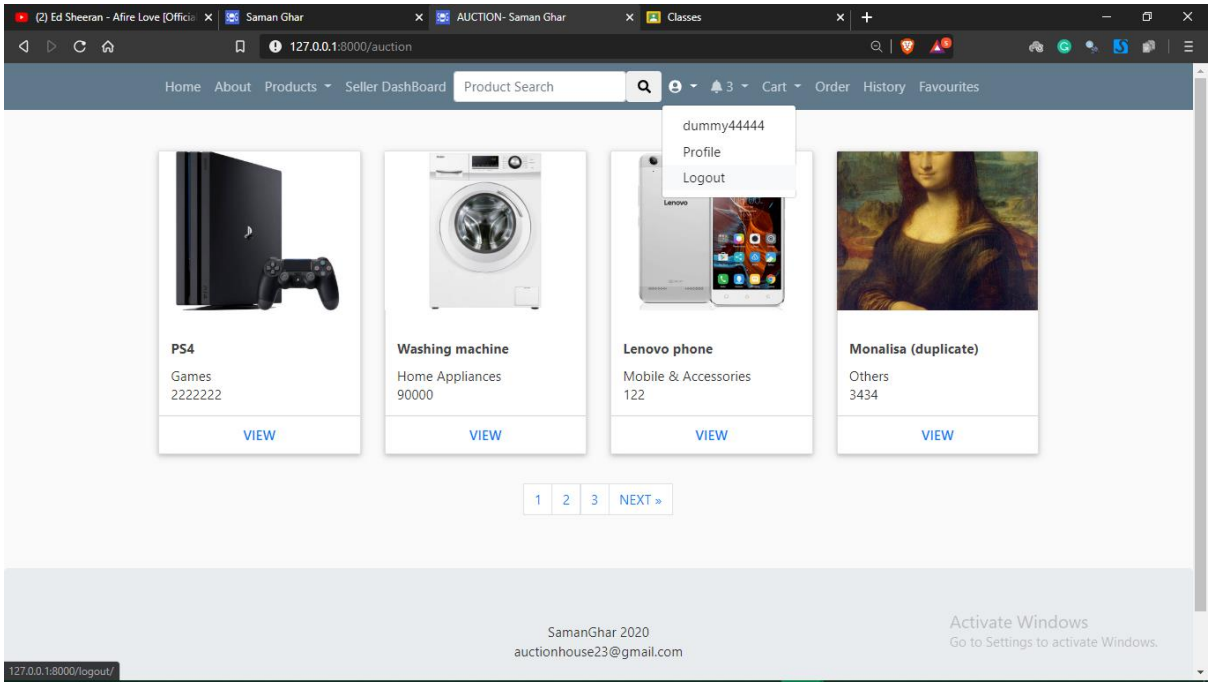


Figure 98: logout

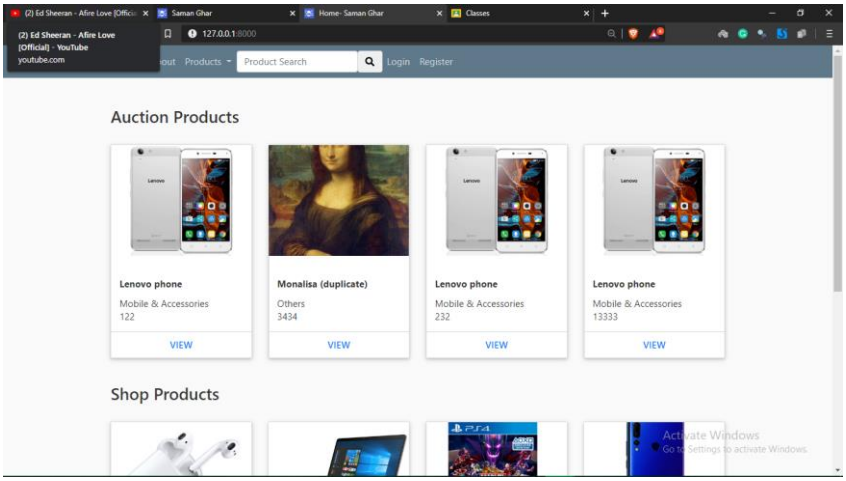


Figure 99: after logout

TEST 18	
Action	**Changing Password**
Expected output	**Password change**
Actual output	**Changed password**
Test Result	**Pass**

Table 24: Test18

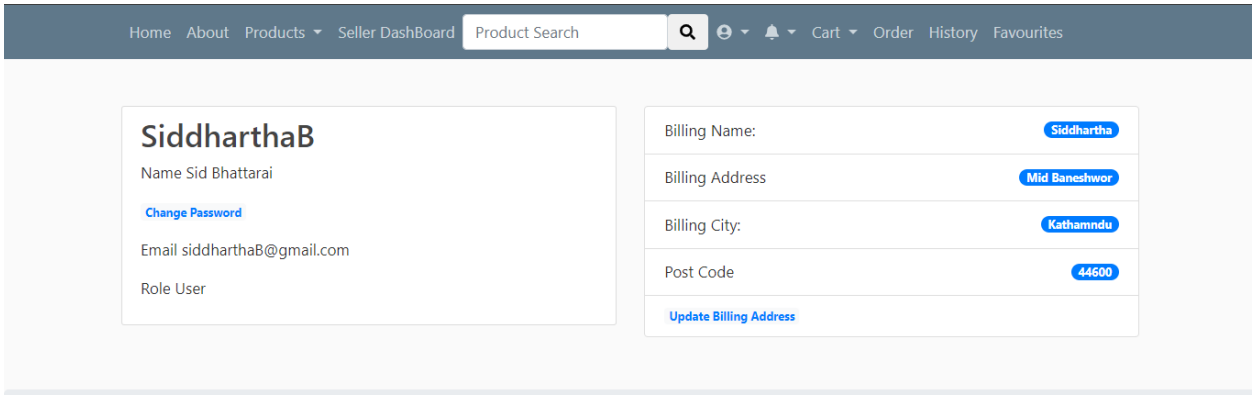


Figure 100: Change Password

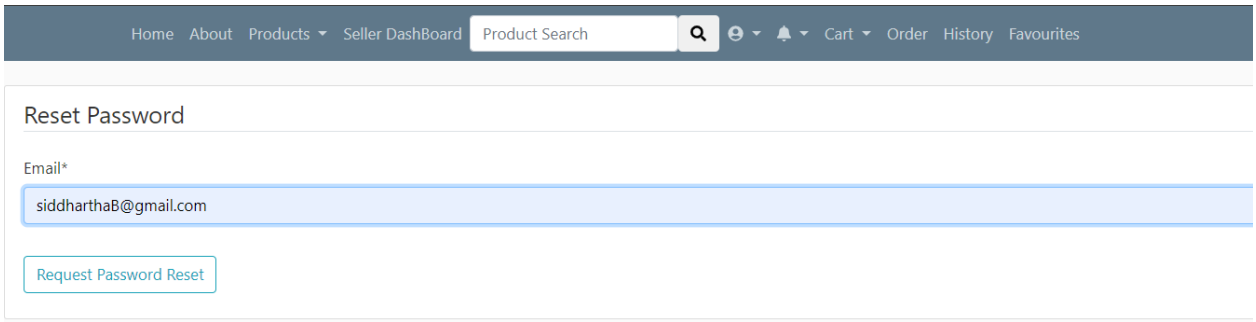


Figure 101:Reset Pasword

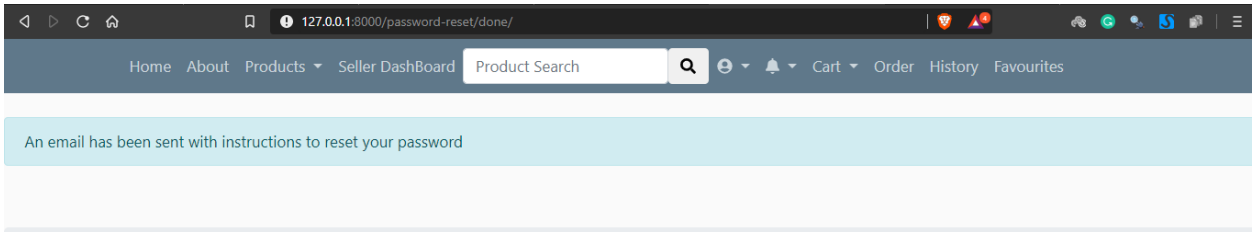


Figure 102: reset password sent

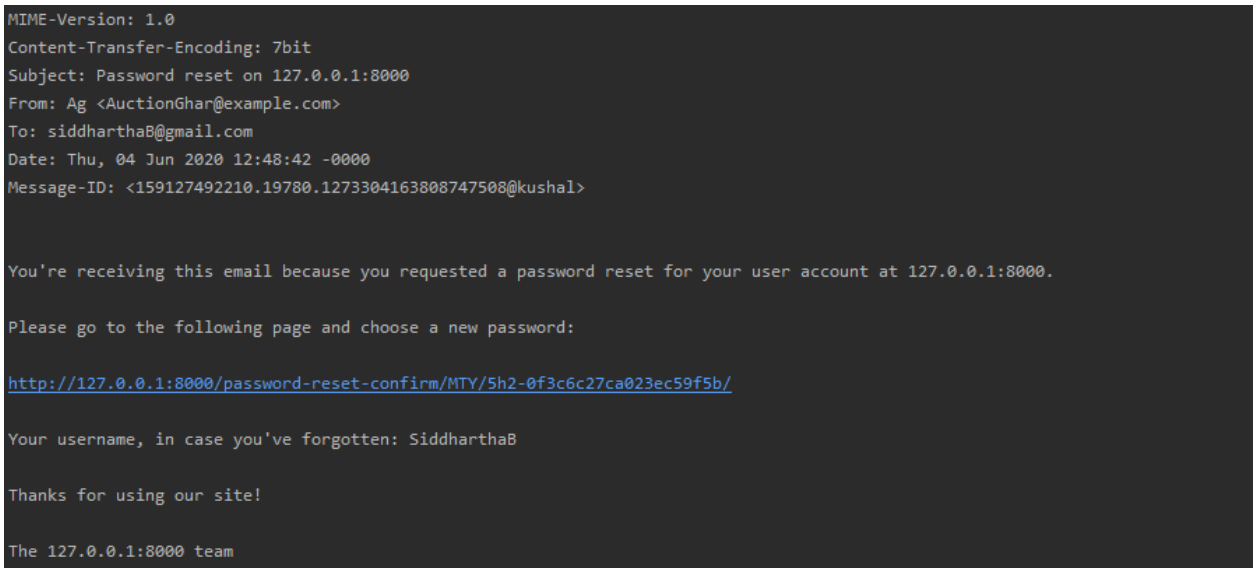


Figure 103: link to reset password

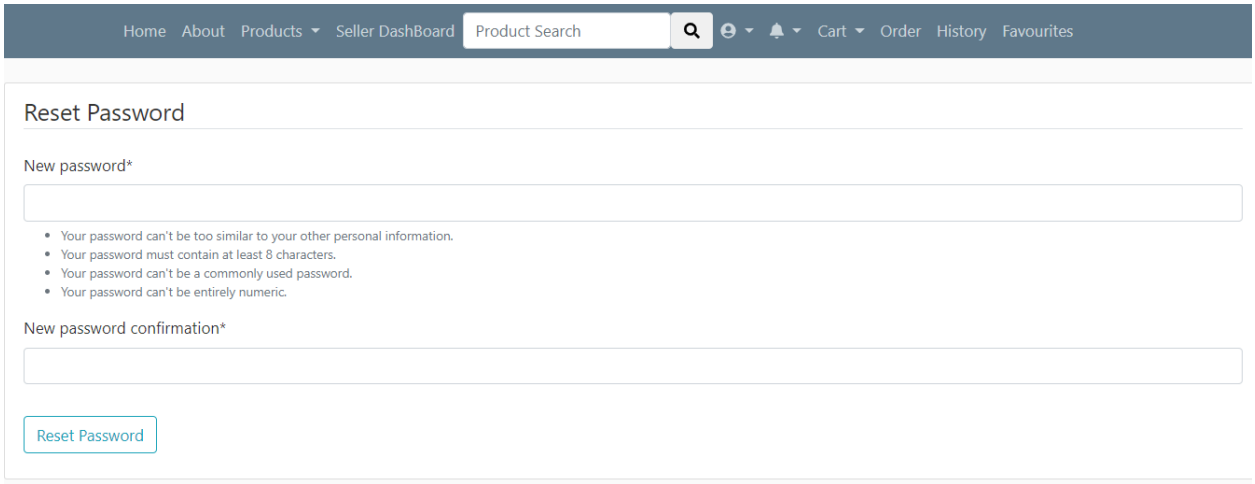


Figure 104: form for reset password

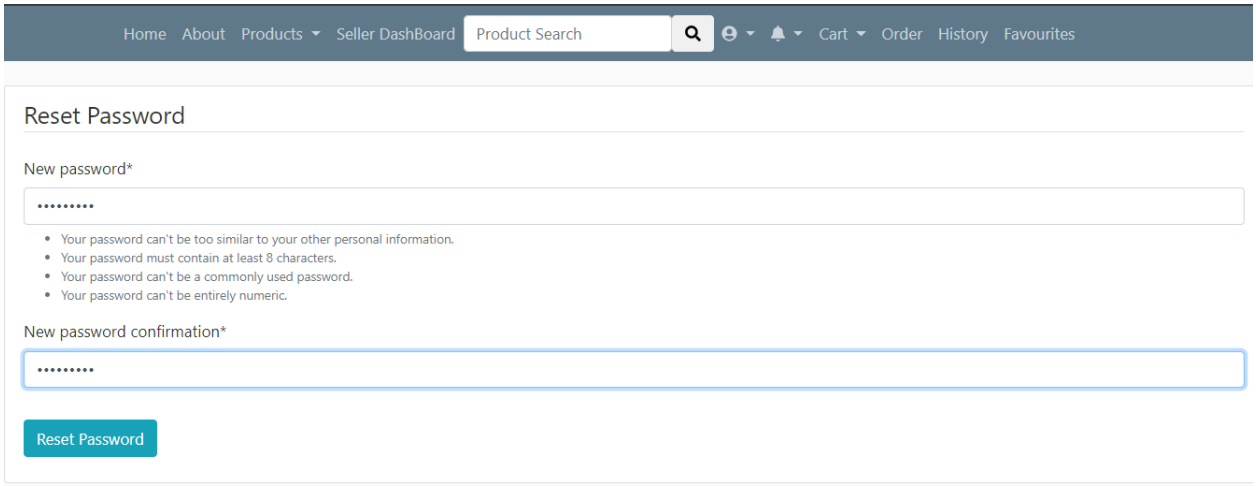


Figure 105:new password formed

TEST 19	
Action	**Updating profile**
Expected output	**Change billing address**
Actual output	**Address changed**
Test Result	**Pass**

Table 25: Test 19

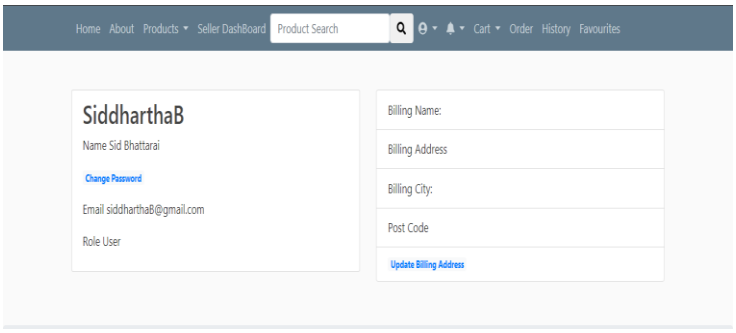


Figure 106:Profile update

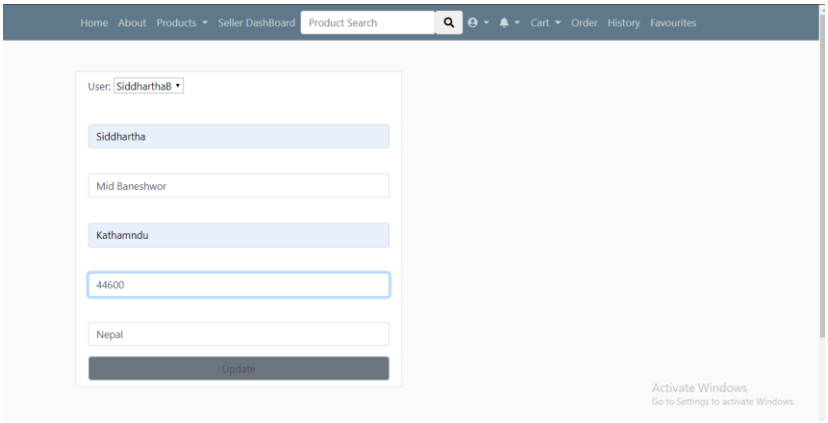


Figure 107:Form for profile update

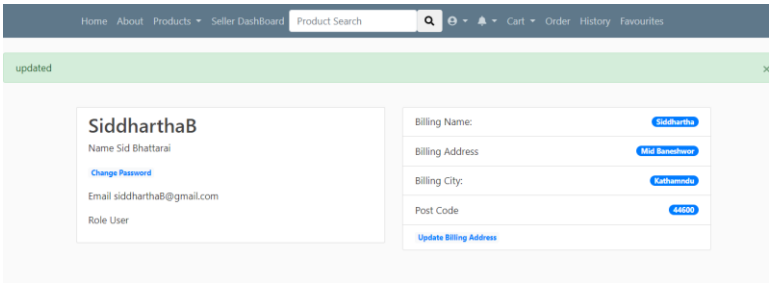


Figure 108:Updated profile

TEST 20	
Action	**Stripe payment**
Expected output	**Stripe payemnt**
Actual output	**Stripe payment successful.**
Test Result	**Pass**

Table 26: Test 20

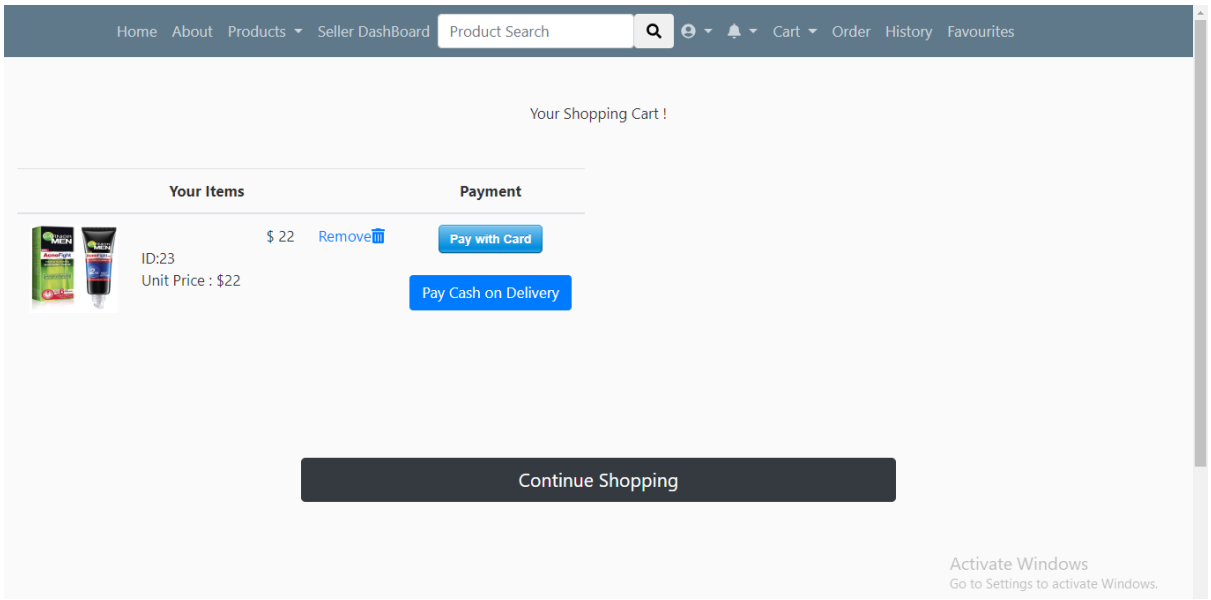


Figure 109: Cart

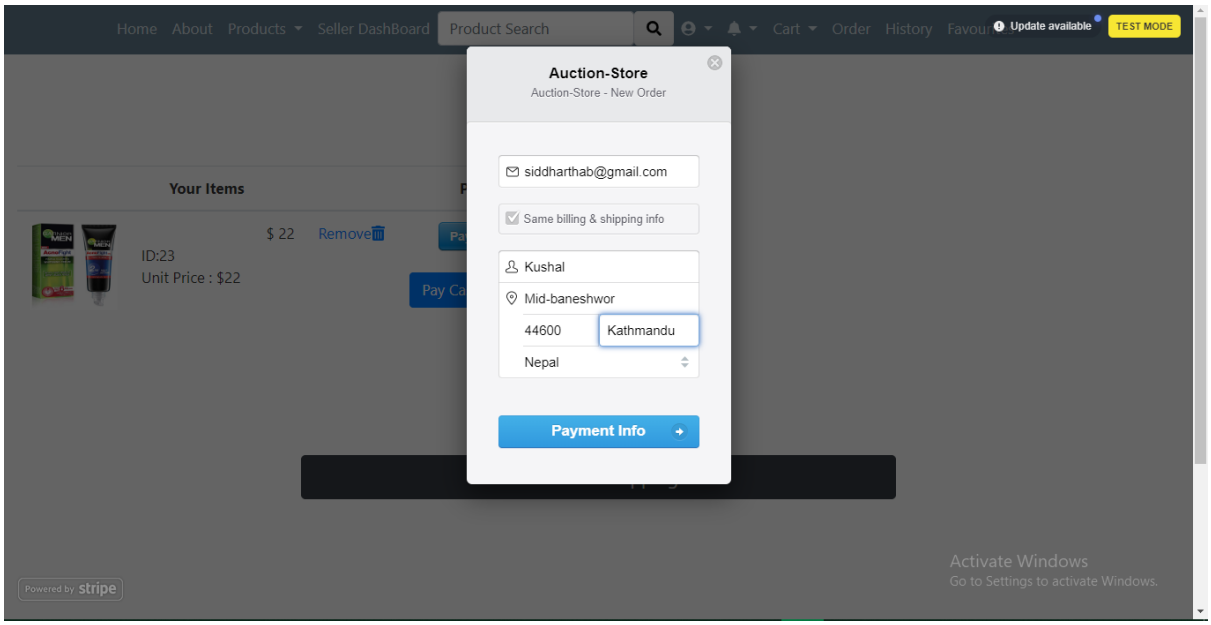


Figure 110: Stripe payment

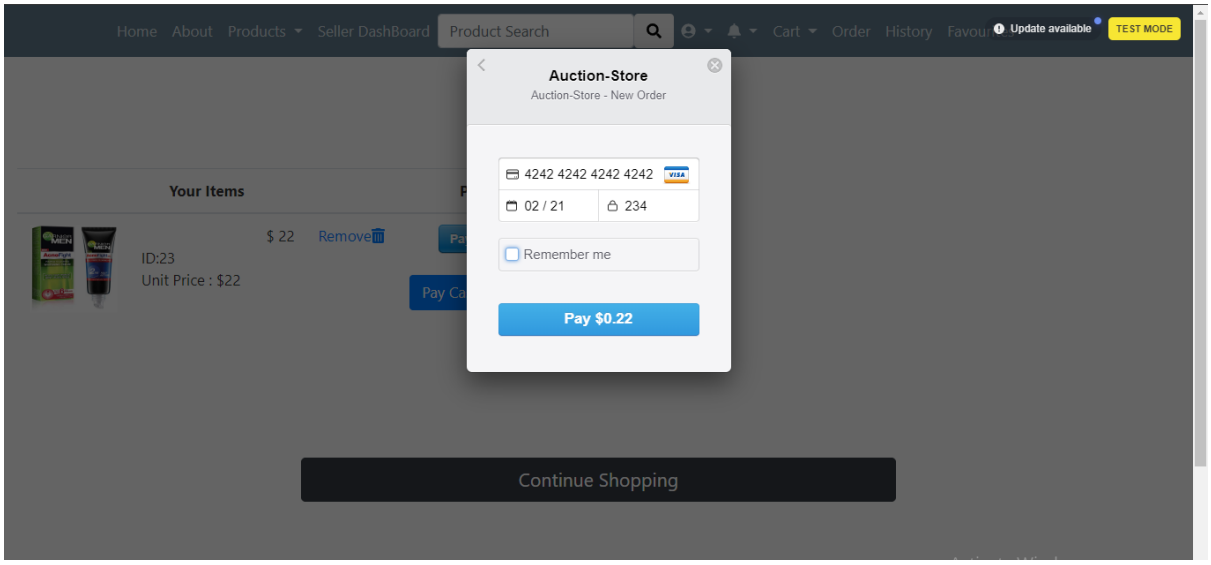


Figure 111:Stripe payment continued

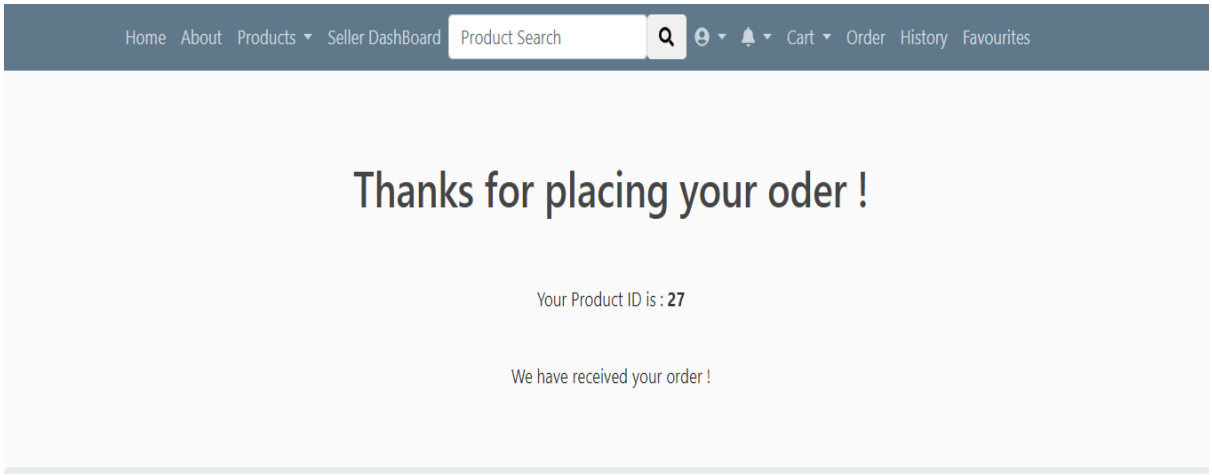


Figure 112:Order completed

4.4 CRITICAL ANALYSIS

The testing was completed with unit as well as system testing. The unit testing was done using coverage , and system testing was done with clear objectives that need to be pass or fail. While running the unit test in the system, every time a new test is created a new test database would be created which would be then deleted after the testing. The difficulty faced during the testing is the proper coding language the test is written in. while using only command prompt , the test were just completed and the data could not be displayed, but by the use of coverage the data test data can be viewed in the htmlcov folder of the system. The code that was used to display in test result in the file was coverage html.

With all the difficulties of the testing and participation of test in the project , testing is a hard job to do but can also be viewed as a clear way of identifying the errors in the program and help the programmer fix the bug. The testing part gave me confidence to do more coding and improve my coding and be a better coder so that my code may contain less portion of bugs.

Finding about different types of testing gave me an experience as there are many testing phases , methods which helps in making the code better . after doing testing I clearly had the knowledge that testing is also an equal and important portion in system development.

CHAPTER 5: CONCLUSION

5.1 LEGAL, SOCIAL AND ETHICAL ISSUES

5.1.1 LEGAL ISSUES

Incorporation Problem: If you are a company operated merely via a website, not being incorporated is a crucial problem. Any purchase and selling activity related to your products will be considered illegal and you can't claim your right in case of any fraud and corruption. Without incorporation, your business has no shelter. Transaction Issues: There can be transaction issues where the money transfer cannot be done. SO cash on delivery is available . Products issues :The users can post illegal items which can be a problem. (EuroLogo Marketing , 2018)

5.1.2 SOCIAL ISSUES

The user can post images which are socially awkward. Drugs, other harmful products can be sold in the system which can cause social issue. The major social issues related to e-commerce and privacy concern the development of "exception of privacy" or privacy norms, as well as public attitudes. In what areas of should we as a society encourage people to think they are in "private territory" as opposed to public view? The major political issues related to ecommerce and privacy concern the development of statutes that govern the relations between record keepers and individuals. Shipping issues causing complexity in process with very high cost.

5.1.3 ETHICAL ISSUES

Data theft can be one of the ethical issues. Online sellers though have a responsibility to their customers. They must ensure that their eCommerce transactions do not result in data theft or security breaches. Customers give you a lot of sensitive information to complete an eCommerce transaction: sharing wrong pictures for the display and selling another things. inaccurate product data, Selling Counterfeit Products. (Lunka, 2015) Web tracking can also be an issue with Cyber Squatting.

5.2 ADVANTAGES

The advantages of the project are as follows:

1. The products can be sold fast and easily.
2. 2nd hand items can be sold easily
3. Anyone can sell Products , licences not needed.
4. Simple and understandable UI.
5. Easy to Bid
6. Easy to communicate to user
7. Easy to file dispute.

5.4 LIMITATIONS

The limitation of the project are as follows:

1. The background task was not implemented .
2. The search portion only searched if the name matches or the category matched other search like price search is not available.
3. The recommendation of the system can is only simply based on category.
4. The algorithms was not used in the system.
5. Since the system is new recommendation cannot be used properly.

5.3 FUTURE WORK

AI enables an ecommerce website to recommend products uniquely suited to shoppers and enables people to search for products using conversational language or images, as though they were interacting with a person. (Meier, 2020) A forum could be created to add new requests where the users can communicate. A real time communication can be created to help the system progress. A tracking system can be implemented in the system.

CHAPTER 6: REFERENCES

brandeis.edu. (2016) *https://www.cs.brandeis.edu/~magnus/ief248a/eBay/history.html* [Online]. Available from: https://www.cs.brandeis.edu/~magnus/ief248a/eBay/history.html [Accessed 2 January 2020].

Daraz. (2020) *https://www.daraz.com.np/* [Online]. Available from: https://www.daraz.com.np/ [Accessed 22 March 2020].

Django Software Foundation. (2020) *https://docs.djangoproject.com/en/3.0/topics/db/models/* [Online]. Available from: https://docs.djangoproject.com/en/3.0/topics/db/models/ [Accessed 22 January 2020].

Django Software Foundation. (2020) *https://docs.djangoproject.com/en/3.0/topics/http/views/* [Online]. Available from: https://docs.djangoproject.com/en/3.0/topics/http/views/ [Accessed 2 January 2020].

Django Software Foundation. (2020) *https://docs.djangoproject.com/en/3.0/topics/templates/* [Online]. Available from: https://docs.djangoproject.com/en/3.0/topics/templates/ [Accessed 22 January 2020].

eBay Inc. (2020) *https://www.ebayinc.com/company/who-we-are/* [Online]. Available from: https://www.ebayinc.com/company/who-we-are/ [Accessed 1 January 2020].

EuroLogo Marketing. (2018) *common-legal-issues-faced-by-e-commerce-businesses/* [Online]. Available from: EuroLogo Marketing [Accessed 22 March 2020].

geeksforgeeks. (2019) *software-engineering-classical-waterfall-model* [Online]. Available from: https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/ [Accessed 2 January 2020].

hamrobazaar. (2020) *https://hamrobazaar.com/* [Online]. Available from: https://hamrobazaar.com/ [Accessed 3 March 2020].

Hsiao, A. (2018) *understanding-the-ebay-auction-automatic-bidding-system-1140186* [Online]. Available from: https://www.thebalancesmb.com/understanding-the-ebay-auction-automatic-bidding-system-1140186 [Accessed 1 January 2020].

IBM Developer. (2020) *https://www.ibm.com/developerworks/rational/library/4763.html* [Online]. Available from: https://www.ibm.com/developerworks/rational/library/4763.html [Accessed 22 March 2020].

Jacko, J.A. & Stephanidis, C. (2003) *Human-Computer Interaction: Theory and Practice Volume 1 of Human Factors and Ergonomics*. illustrated ed. CRC Press.

Learntek. (2019) *spiral-model* [Online]. Available from: https://www.learntek.org/blog/spiral-model/ [Accessed 2 January 2020].

Lunka, R. (2015) *ethical-issues-in-ecommerce/* [Online]. Available from: https://www.nchannel.com/blog/ethical-issues-in-ecommerce/ [Accessed 3 March 2020].

Meier, S. (2020) *the-future-of-ai-retail-and-roi* [Online]. Available from: https://www.bigcommerce.com/blog/ecommerce-ai/#the-future-of-ai-retail-and-roi [Accessed 3 March 2020].

My-Project-Management-Expert. (2009) *the-advantages-and-disadvantages-of-rup-software-development.html* [Online]. Available from: http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-rup-software-development.html [Accessed 4 January 2020].

ProductPlan. (2020) *moscow-prioritization/* [Online]. Available from: https://www.productplan.com/glossary/moscow-prioritization/ [Accessed 2 march 2020].

Rational. (2011) *Rational Unified Process: Best Practices for Software development Teams*. Rational Software White Paper. IBM.

Sharpened Productions. (2019) *rup Definiton* [Online]. Available from: https://techterms.com/definition/rup [Accessed 3 September 2019].

Study.com. (2019) *what-is-the-rational-unified-process-methodology-tools-examples.html* [Online]. Available from: https://study.com/academy/lesson/what-is-the-rational-unified-process-methodology-tools-examples.html [Accessed 3 September 2019].

CHAPTER 7: BIBLIOGRAPHY

Aldaej, R. et al. (2018) Analyzing, Designing and Implementing a Web-Based Auction online System. *International Journal of Applied Engineering Research*, 12, pp.8005-13.

Arnaudova, M. (2010) *ebay-analysis* [Online]. Available at: https://www.slideshare.net/MaryArnaudova/ebay-analysis [Accessed 1 January 2019].

Blackhawk Network. (2020) *4-ideas-real-time-e-commerce* [Online]. Available from: https://www.hawkcommerce.com/resource/4-ideas-real-time-e-commerce [Accessed 6 March 2020].

E Radar. (2020) *fair-warning-7-risks-of-the-online-auction/* [Online]. Available from: https://www.eradar.eu/fair-warning-7-risks-of-the-online-auction/ [Accessed 20 January 2020].

Marasini, M.K. (2019) *https://kathmandupost.com/columns/2019/07/15/digital-development-and-e-commerce-in-nepal* [Online]. Available from: https://kathmandupost.com/columns/2019/07/15/digital-development-and-e-commerce-in-nepal [Accessed 3 Feb 2020].

Medium. (2018) *https://medium.com/@boscacci/why-and-how-to-make-a-requirements-txt-f329c685181e* [Online]. Available from: https://medium.com/@boscacci/why-and-how-to-make-a-requirements-txt-f329c685181e [Accessed 22 April 2020].

Real Python. (2020) *https://realpython.com/testing-in-django-part-1-best-practices-and-examples/* [Online]. Available from: https://realpython.com/testing-in-django-part-1-best-practices-and-examples/ [Accessed 2 Jan 2020].

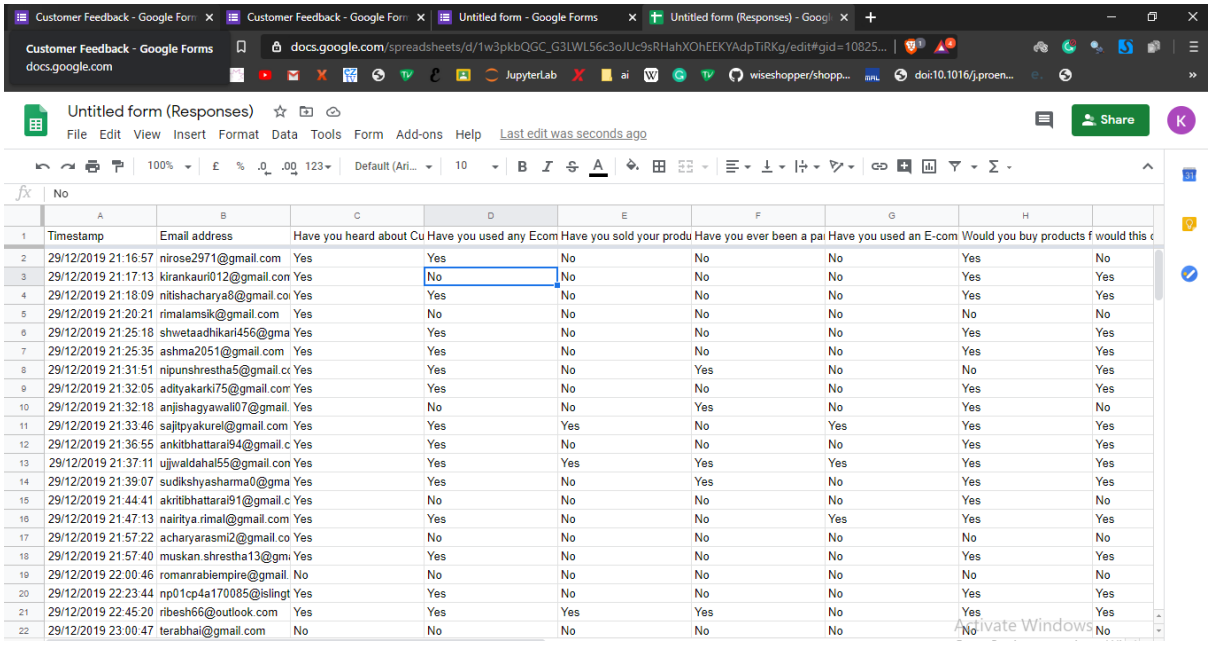
Schafer, J.B., Konstan, J.A. & Riedl, J. (2020) *schafer01ecommerce.pdf* [Online]. GroupLens Research Project Department of Computer Science and Engineering University of Minnesota University of Minnesota Available at: http://www.cs.umd.edu/~samir/498/schafer01ecommerce.pdf [Accessed 3 March 2020].

ShipBob, Inc. (2019) *https://www.shipbob.com/blog/ecommerce-order-tracking/* [Online]. Available from: https://www.shipbob.com/blog/ecommerce-order-tracking/ [Accessed 2 June 2020].

CHAPTER 8: APPENDIX

8.1 APPENDIX A: PRE-SURVEY

8.1.1 PRE-SURVEY FORM



	Timestamp	Email address	Have you heard about Cu	Have you used any Ecom	Have you sold your produ	Have you ever been a pa	Have you used an E-com	Would you buy products f	would this
2	29/12/2019 21:16:57	nirose2971@gmail.com	Yes	Yes	No	No	No	Yes	No
3	29/12/2019 21:17:13	kirankauri012@gmail.con	Yes	No	No	No	No	Yes	Yes
4	29/12/2019 21:18:09	nitishacharya8@gmail.co	Yes	Yes	No	No	No	Yes	Yes
5	29/12/2019 21:20:21	rimalamsik@gmail.com	Yes	No	No	No	No	No	No
6	29/12/2019 21:25:18	shwetaadhikari456@gma	Yes	Yes	No	No	No	Yes	Yes
7	29/12/2019 21:25:35	ashma2051@gmail.com	Yes	Yes	No	No	No	Yes	Yes
8	29/12/2019 21:31:51	nipunshrestha5@gmail.cc	Yes	Yes	No	Yes	No	No	Yes
9	29/12/2019 21:32:05	adityakarki75@gmail.con	Yes	Yes	No	No	No	Yes	Yes
10	29/12/2019 21:32:18	anjishagyawali07@gmail.	Yes	No	No	Yes	No	Yes	No
11	29/12/2019 21:33:46	sajitpyakurel@gmail.com	Yes	Yes	Yes	No	Yes	Yes	Yes
12	29/12/2019 21:36:55	ankitbhattarai94@gmail.c	Yes	Yes	No	No	No	Yes	Yes
13	29/12/2019 21:37:11	ujjwaldahal55@gmail.con	Yes	Yes	Yes	Yes	Yes	Yes	Yes
14	29/12/2019 21:39:07	sudikshyasharma0@gma	Yes	Yes	No	Yes	No	Yes	Yes
15	29/12/2019 21:44:41	akritibhattarai91@gmail.c	Yes	No	No	No	No	Yes	No
16	29/12/2019 21:47:13	nairitya.rimal@gmail.com	Yes	Yes	No	No	Yes	Yes	Yes
17	29/12/2019 21:57:22	acharyarasmi2@gmail.co	Yes	No	No	No	No	No	No
18	29/12/2019 21:57:40	muskan.shrestha13@gm	Yes	Yes	No	No	No	Yes	Yes
19	29/12/2019 22:00:46	romanrabiempire@gmail.	No	No	No	No	No	No	No
20	29/12/2019 22:23:44	np01cp4a170085@islingt	Yes	Yes	No	No	No	Yes	Yes
21	29/12/2019 22:45:20	ribesh66@outlook.com	Yes	Yes	Yes	Yes	No	Yes	Yes
22	29/12/2019 23:00:47	terabhai@gmail.com	No	No	No	No	No	No	No

Figure 113:pre survey form

8.1.2 SAMPLE OF FILLED PRE-SURVEY FORMS

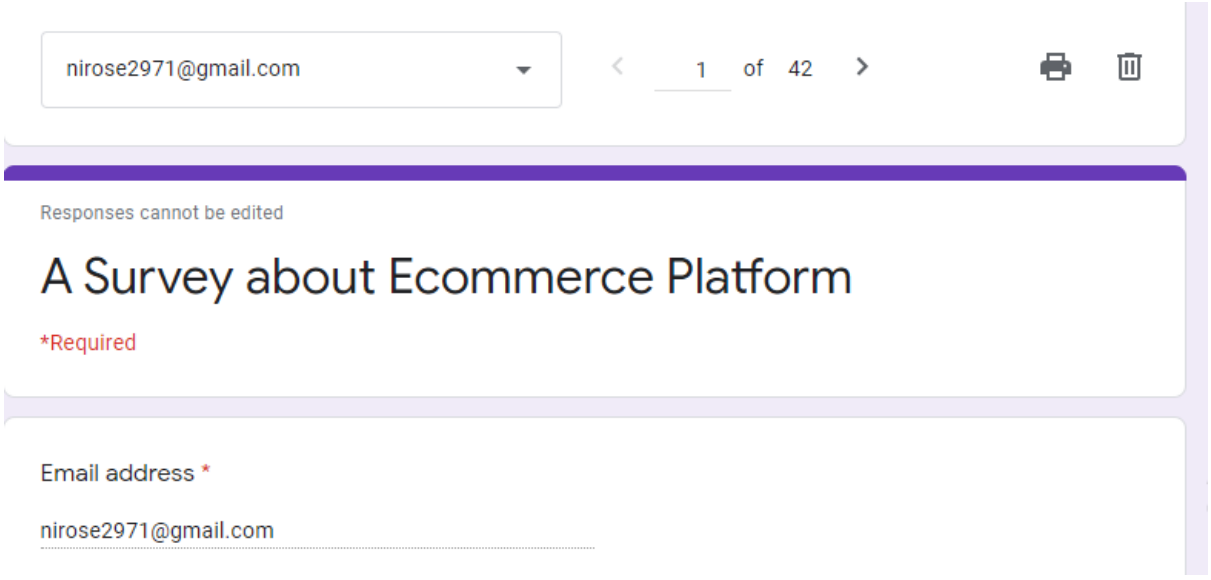


Figure 114:Survey -1

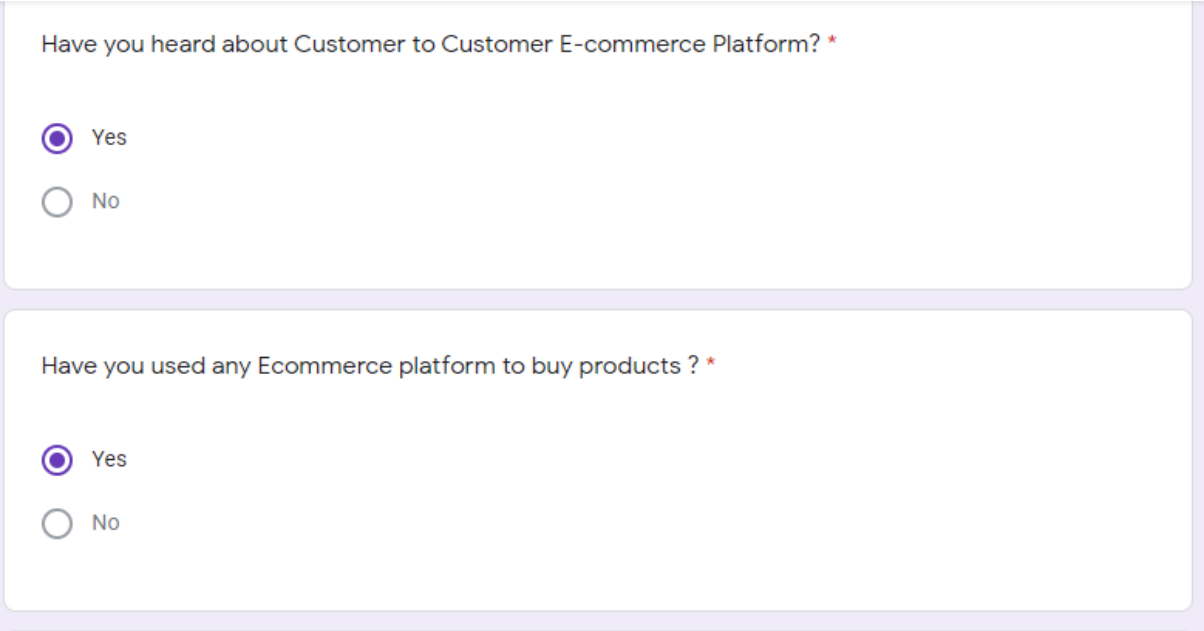


Figure 115: Survey -2

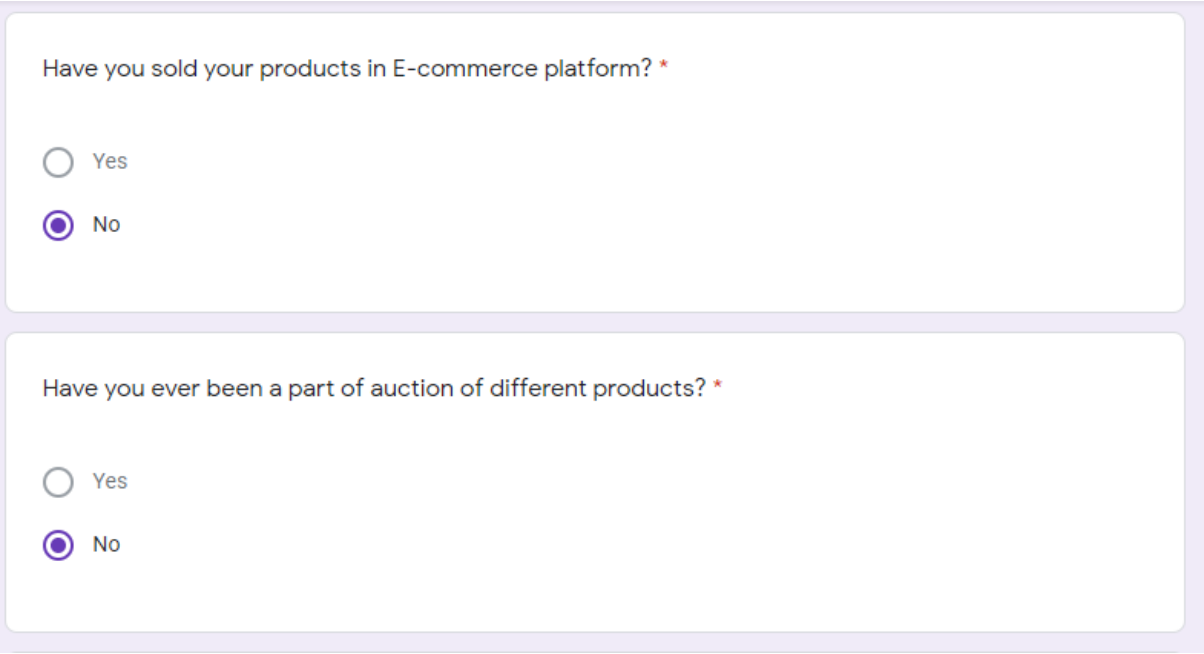


Figure 116: Survey -3

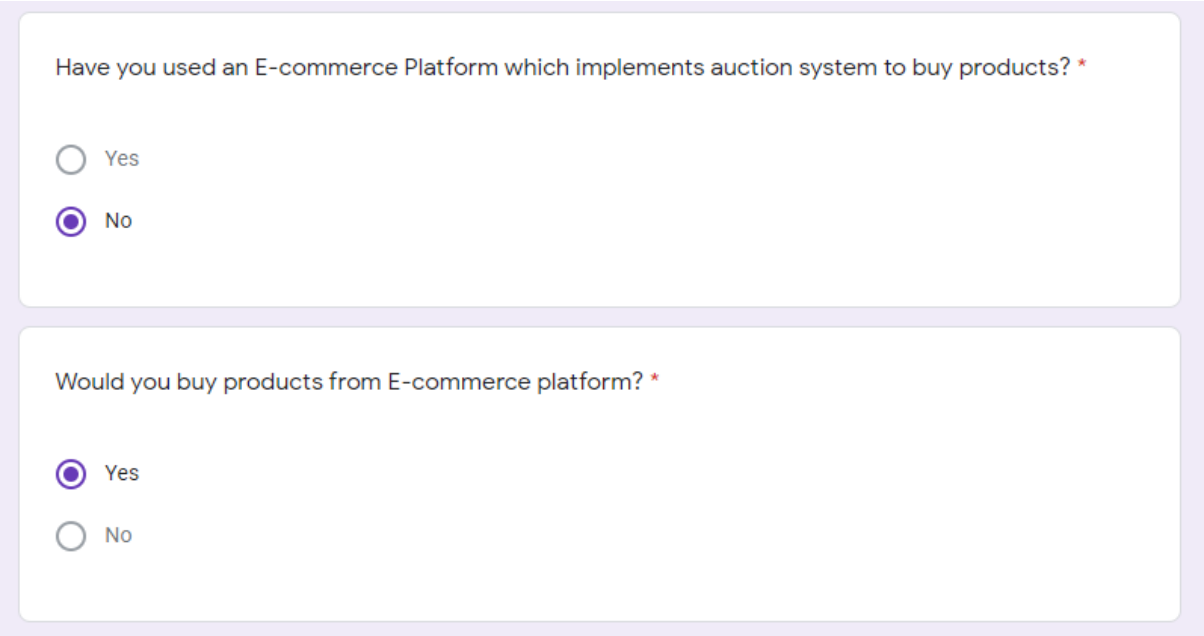


Figure 117: Survey -4

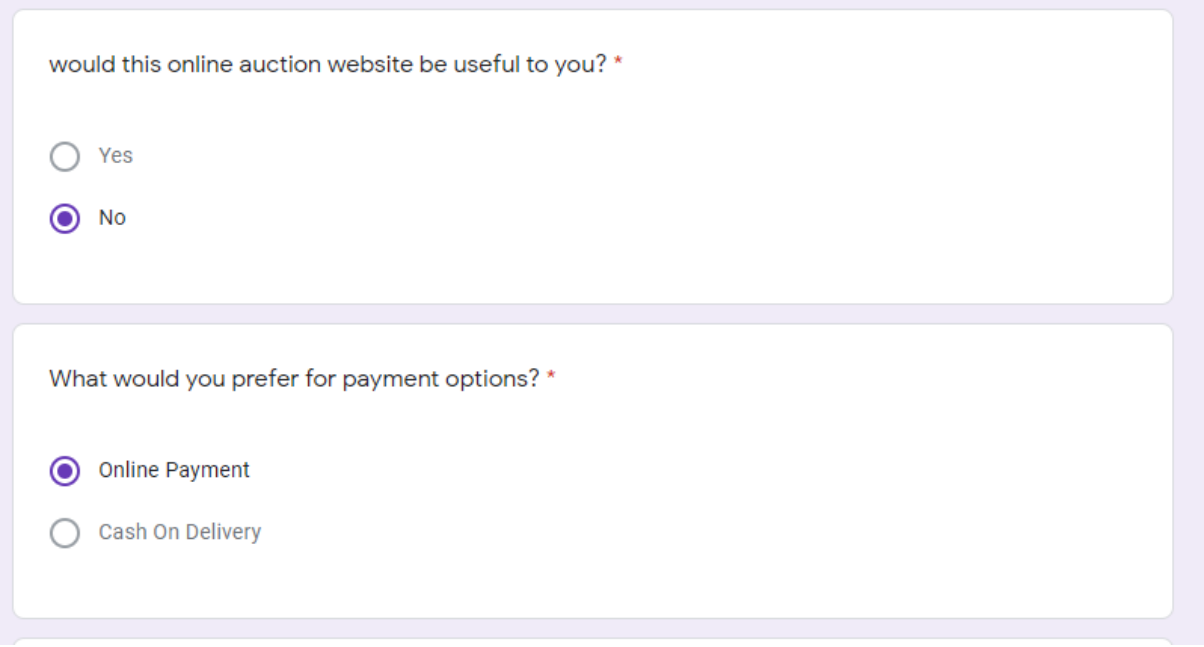


Figure 118: Survey- 5

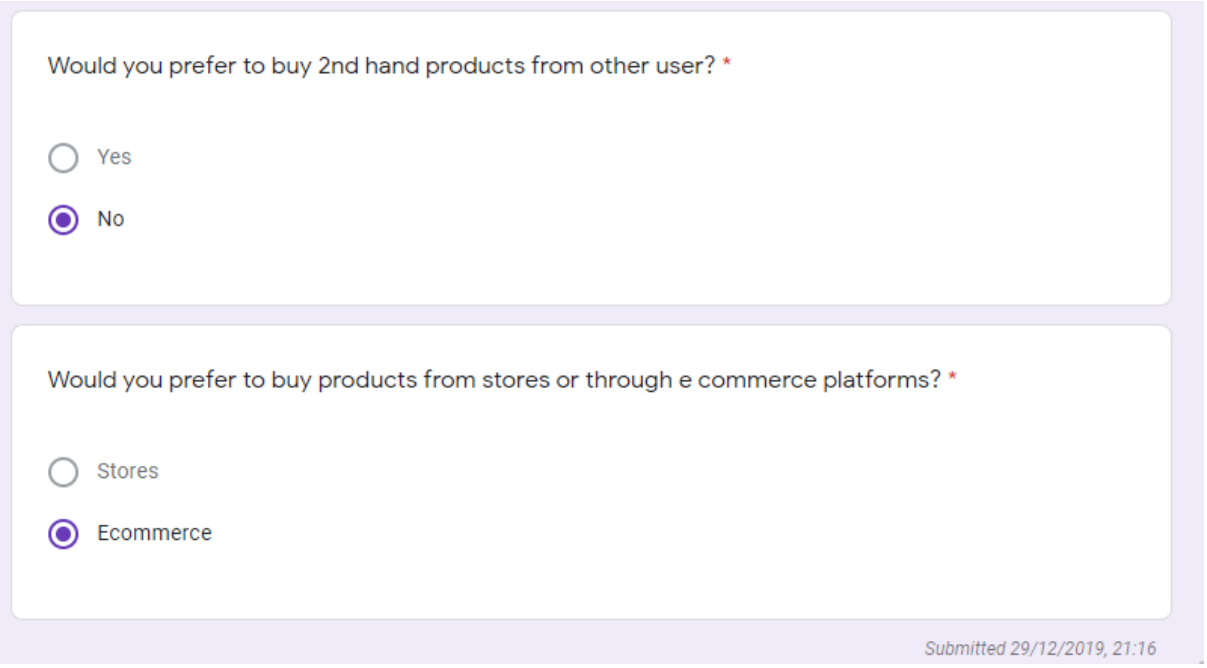


Figure 119: Survey -6

8.1.3 PRE-SURVEY RESULT

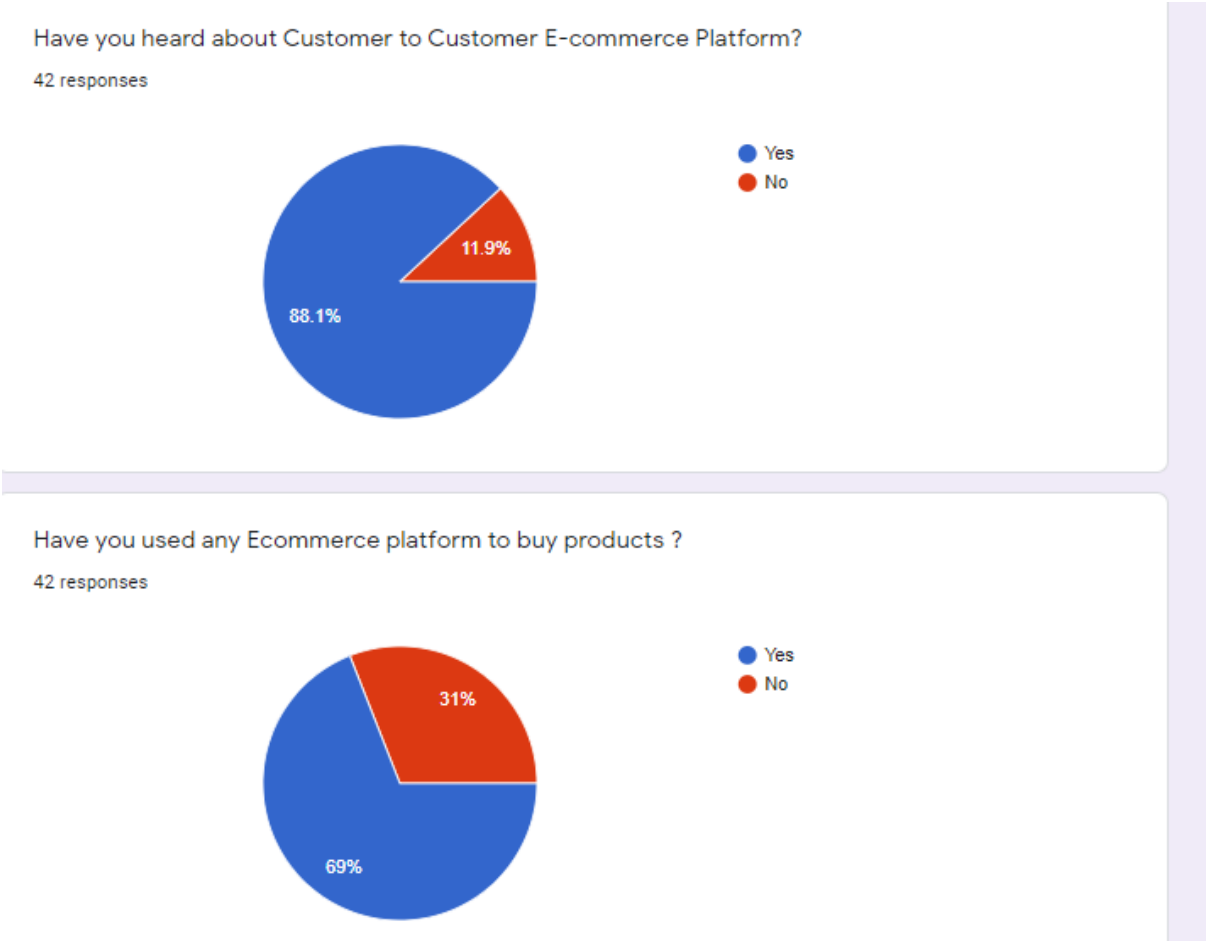


Figure 120:Result -1

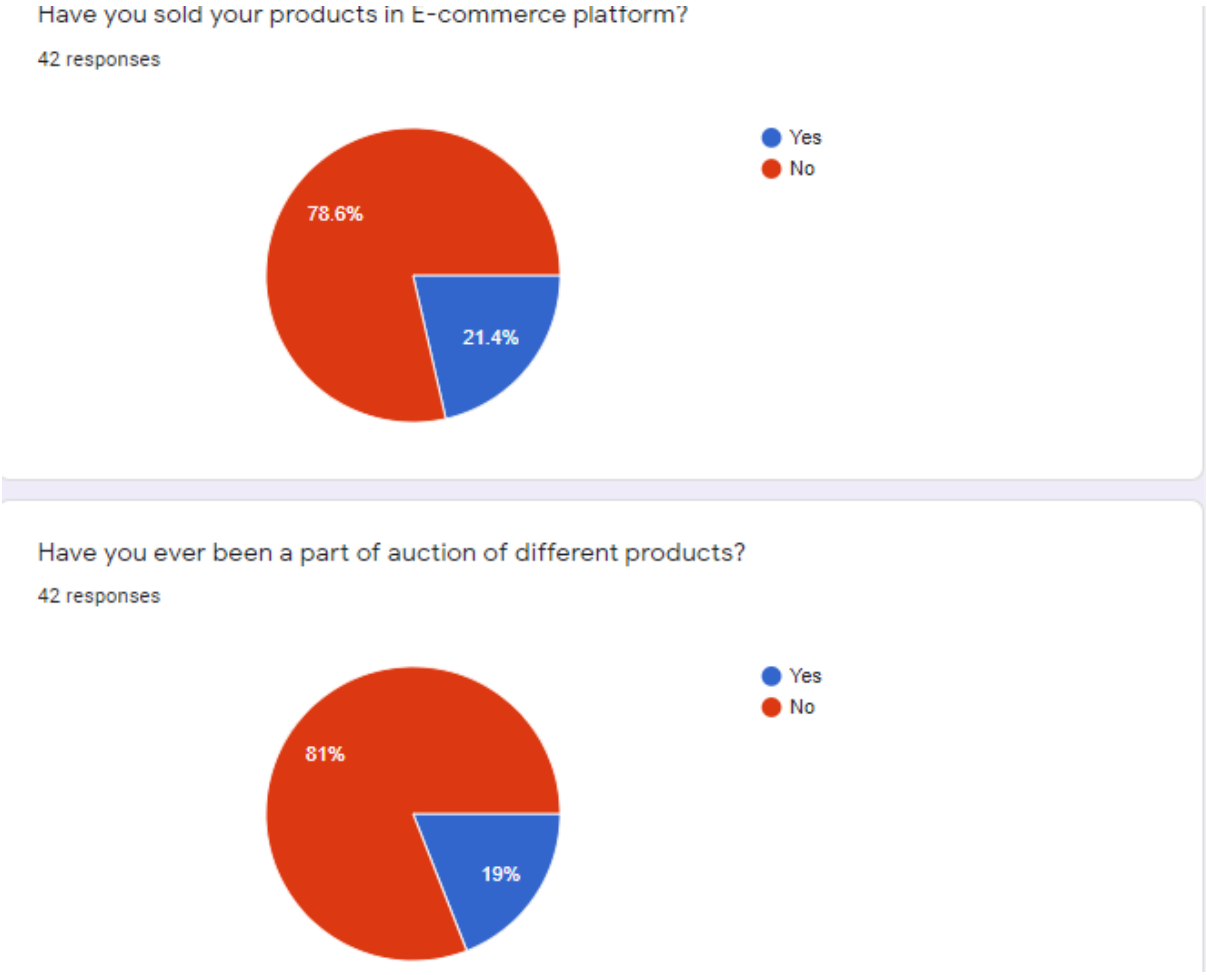


Figure 121: Result 2

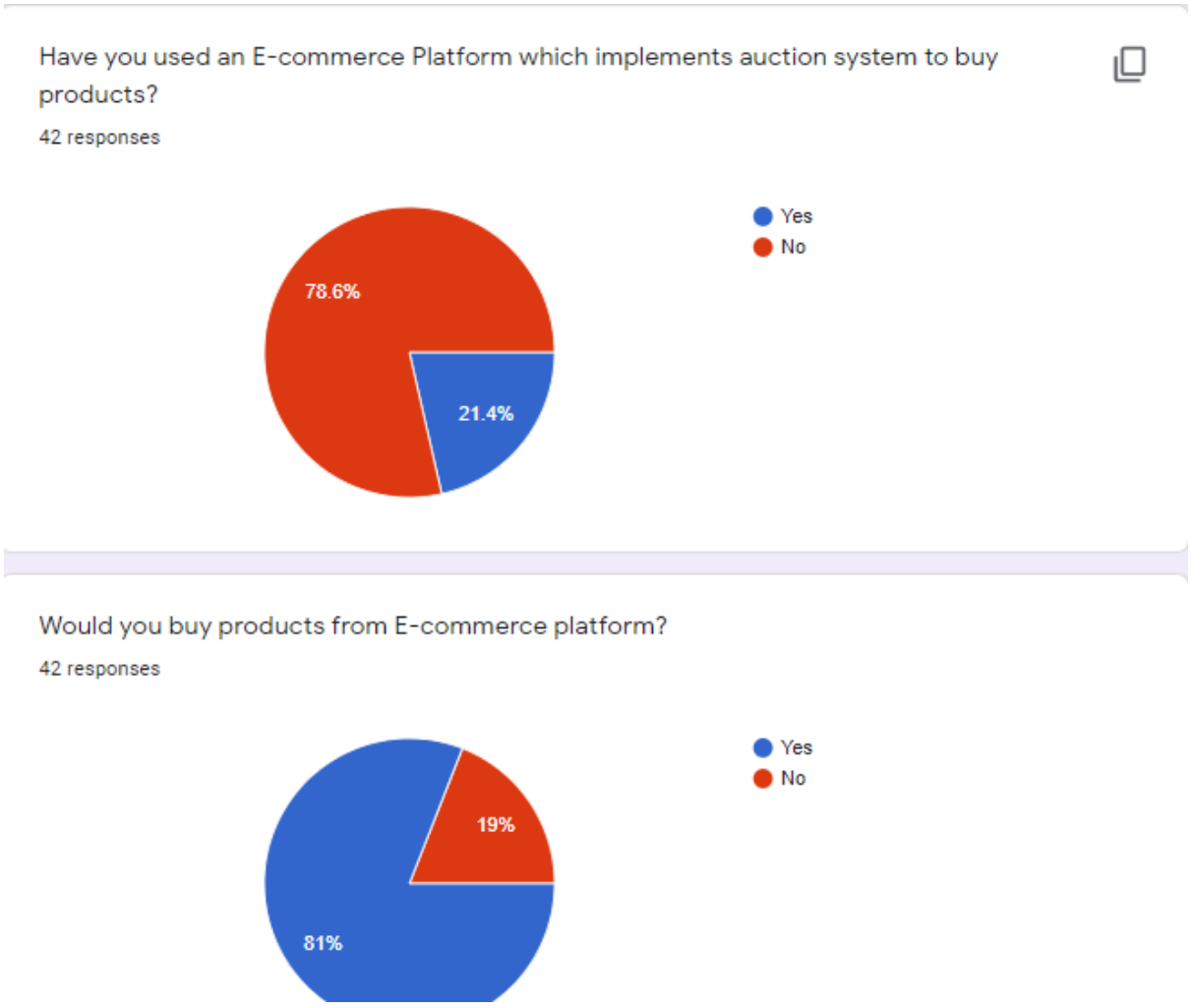


Figure 122: Result -3

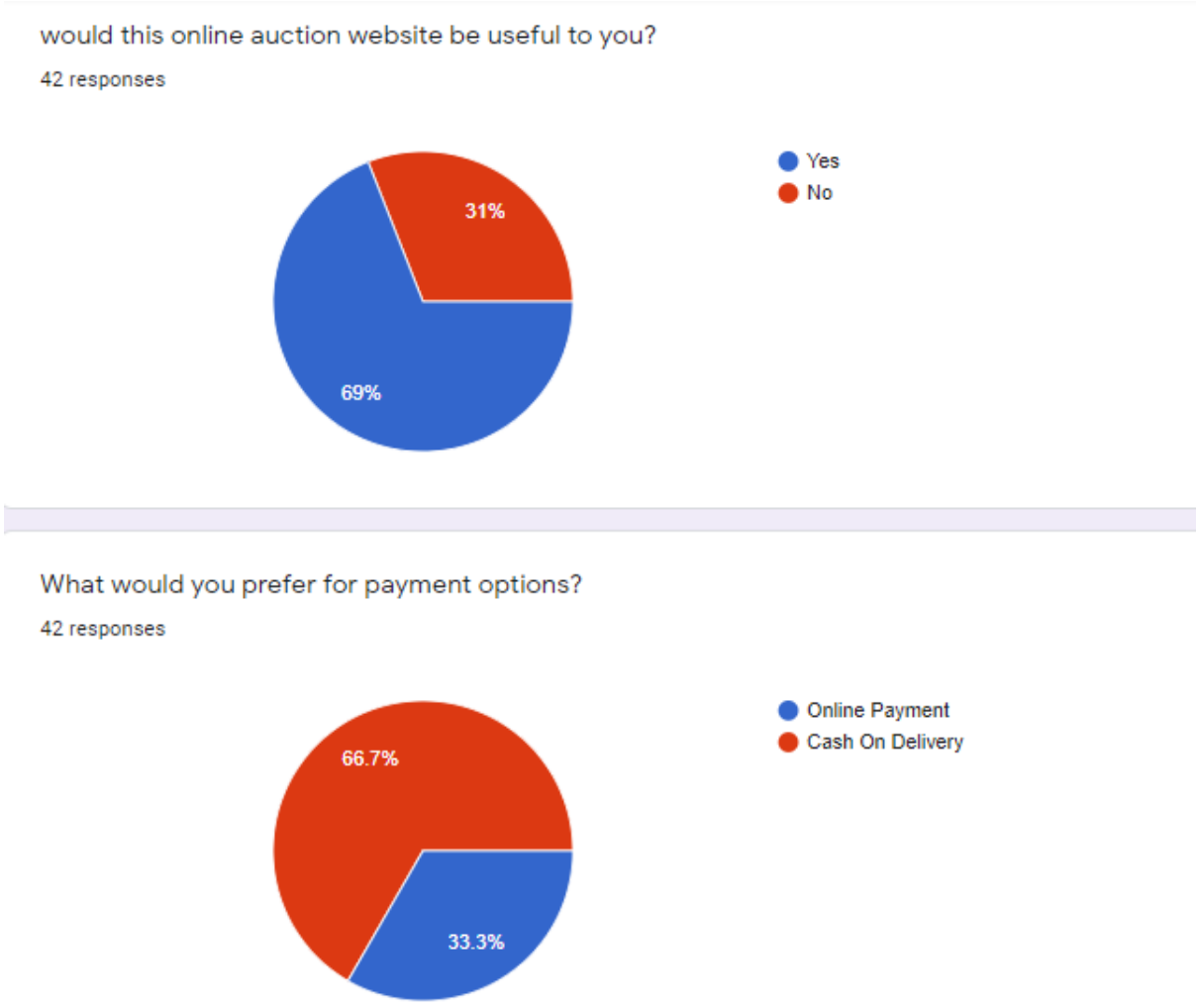


Figure 123: Result 4

Figure 124: Result 5

8.2 APPENDIX B: POST-SURVEY

8.2.1 POST-SURVEY FORM

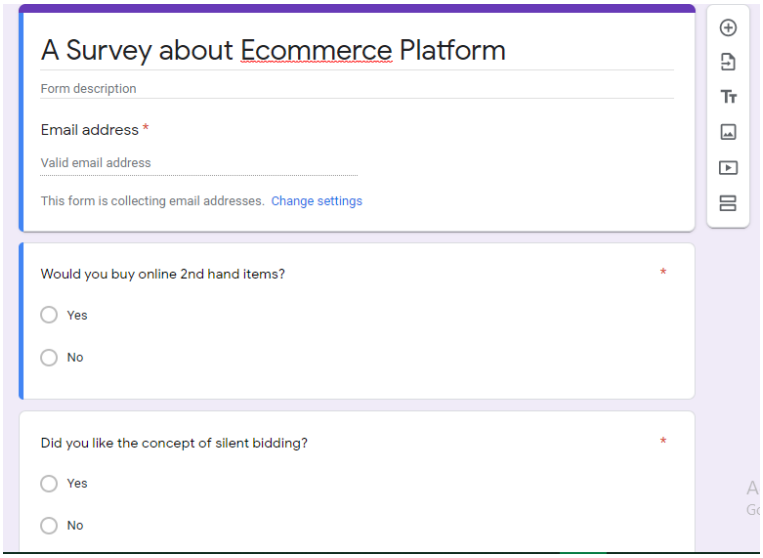


Figure 125: Form1

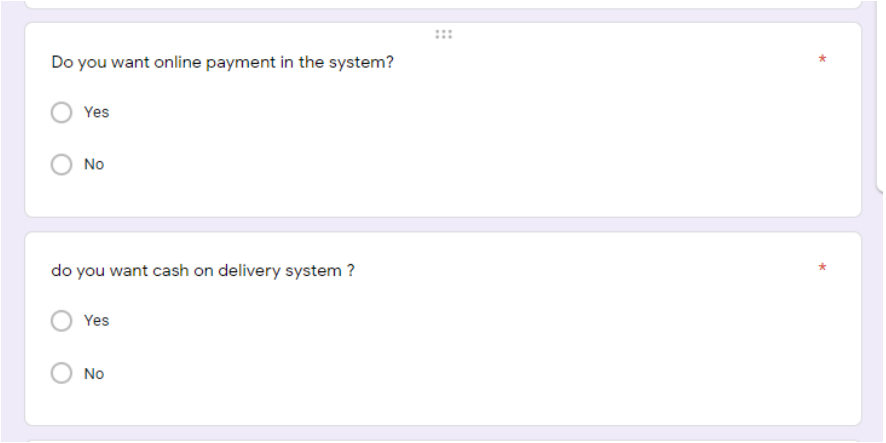


Figure 126:Form2

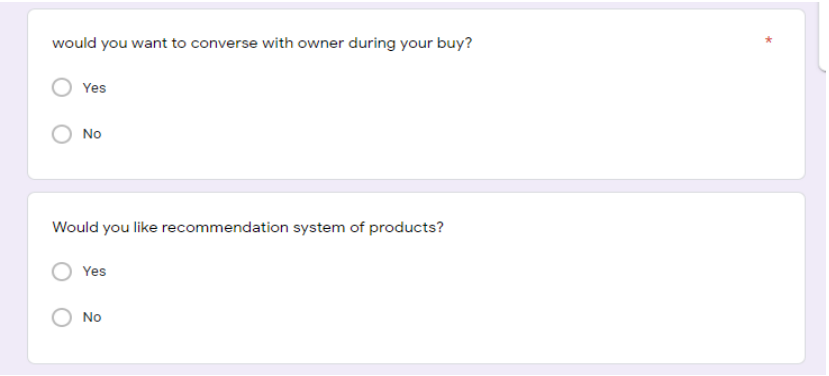


Figure 127:Form 3

8.2.2 SAMPLE OF FILLED POST-SURVEY FORMS

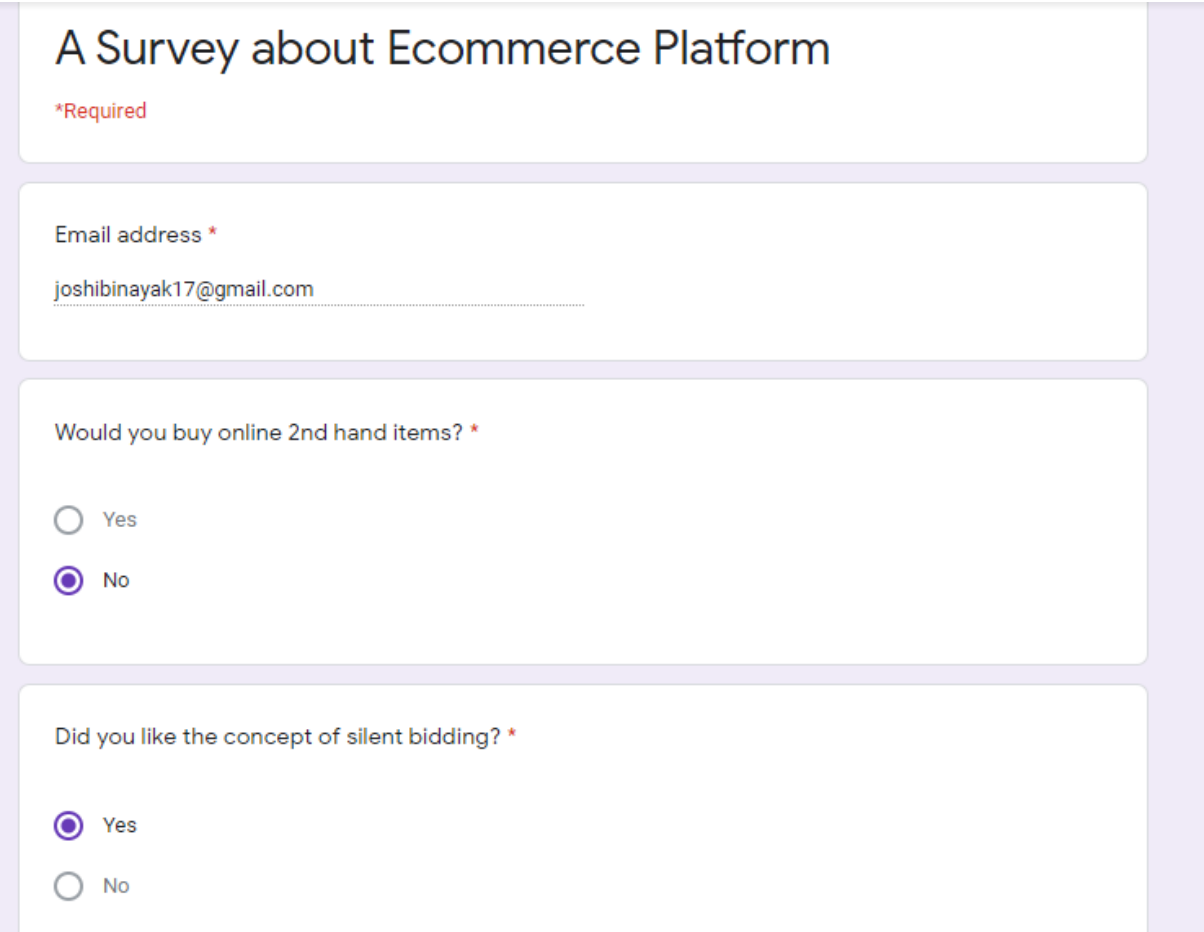


Figure 128: filled form1

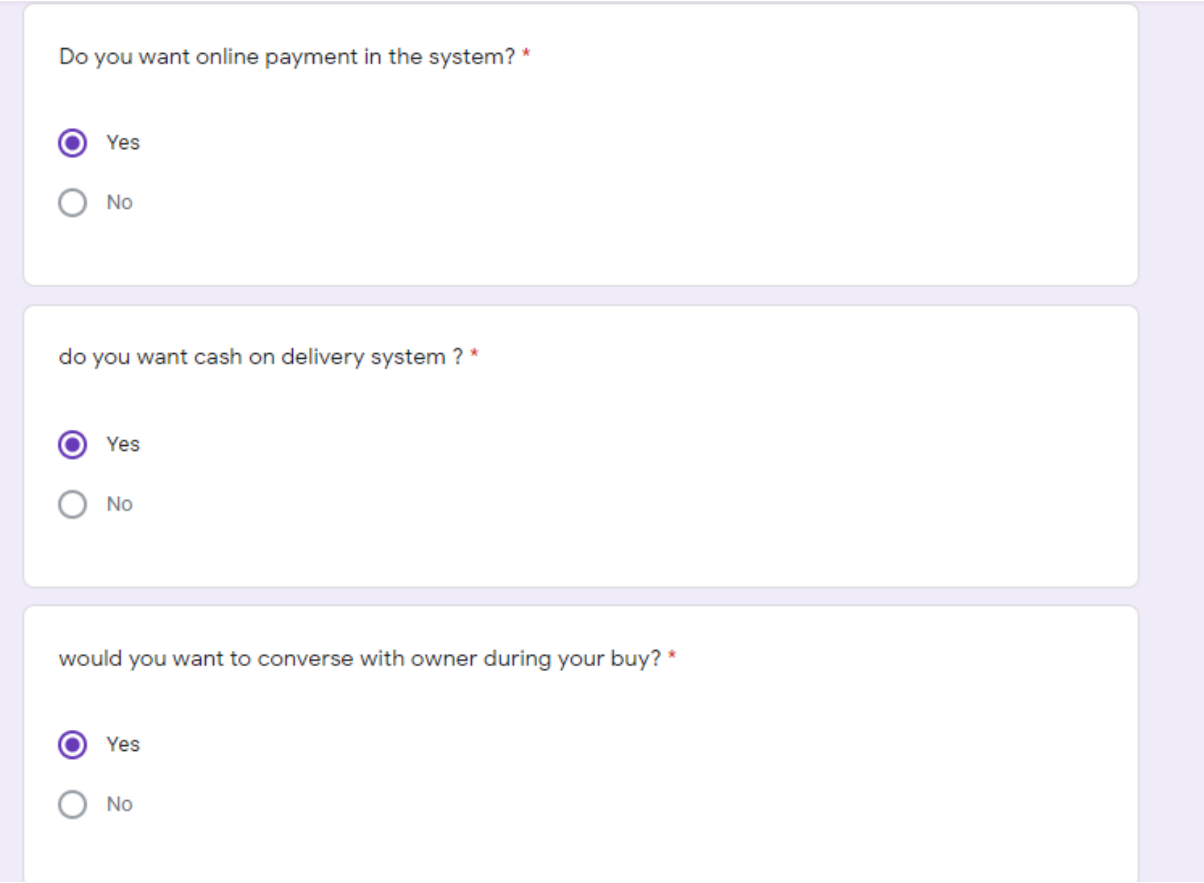


Figure 129: filled form-2

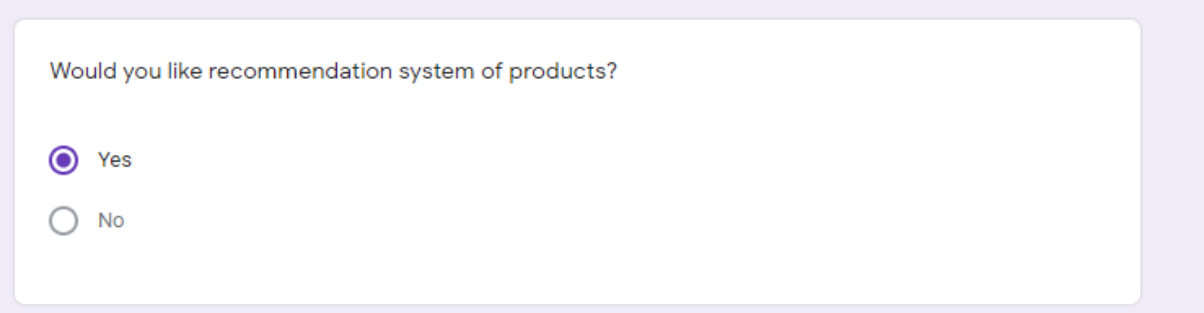


Figure 130: Filled form-3

8.2.3 POST-SURVEY RESULT

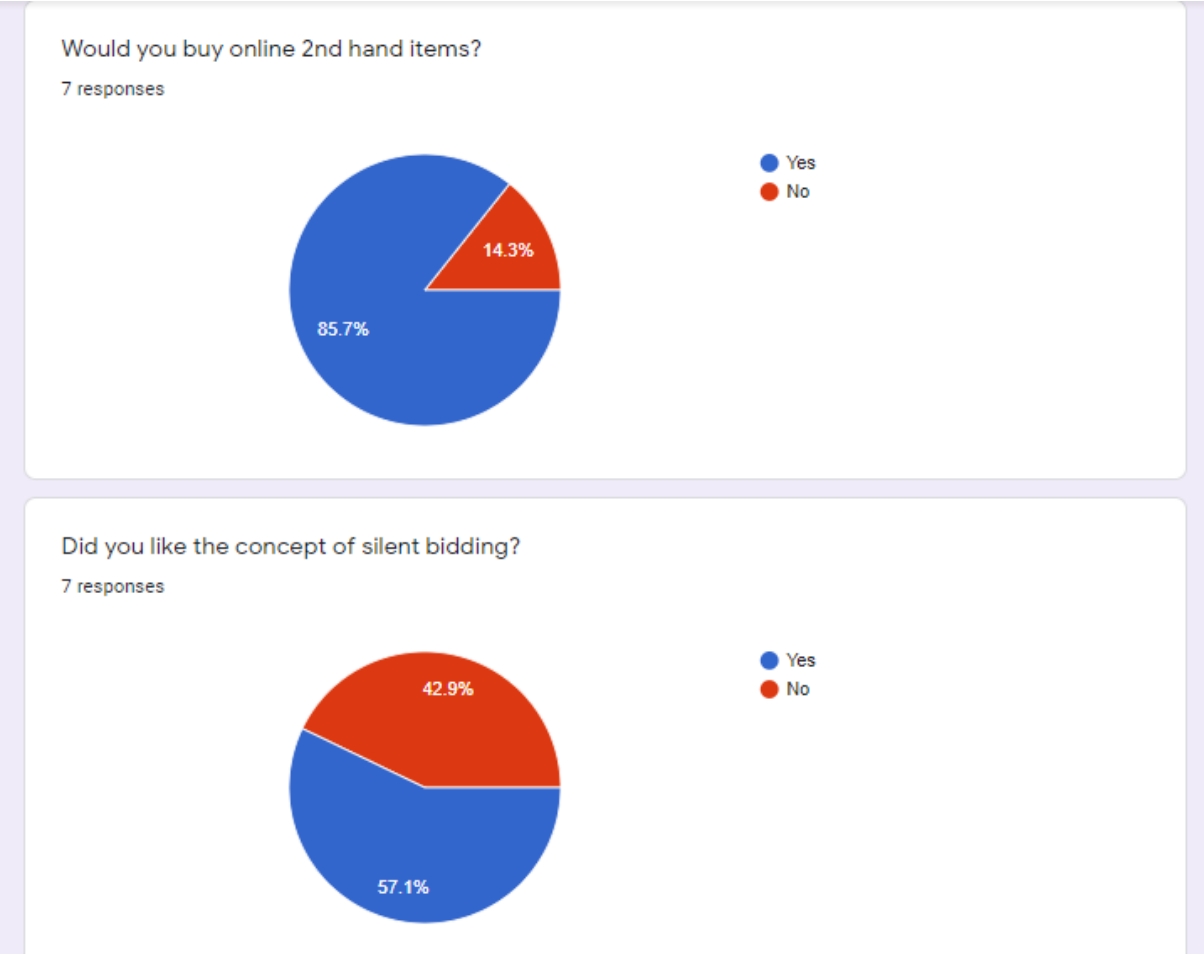


Figure 131:Result-1

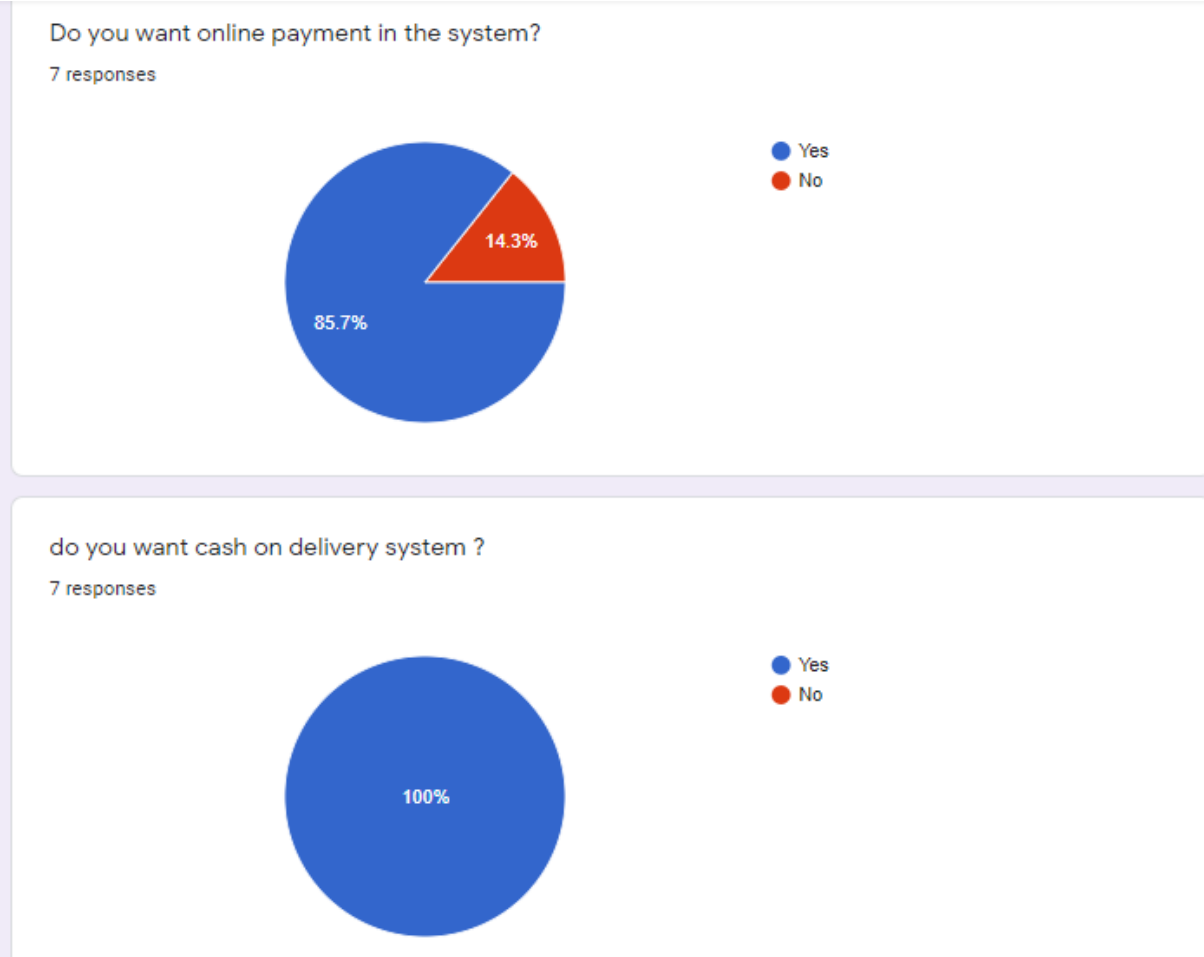


Figure 132:Result 2

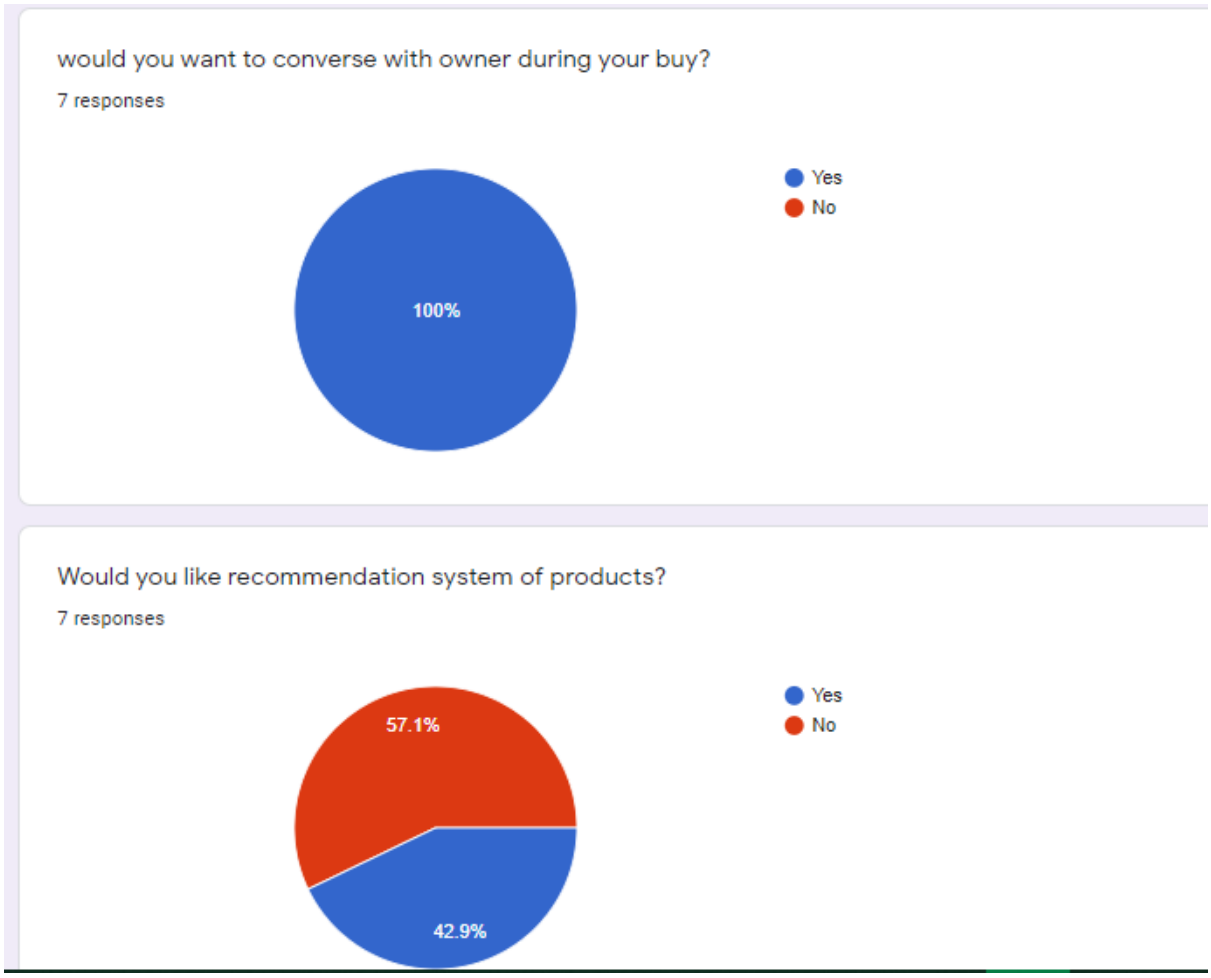


Figure 133:Result 3

8.3 APPENDIX C: SAMPLE CODES

8.3.1 SAMPLE CODE OF THE UI

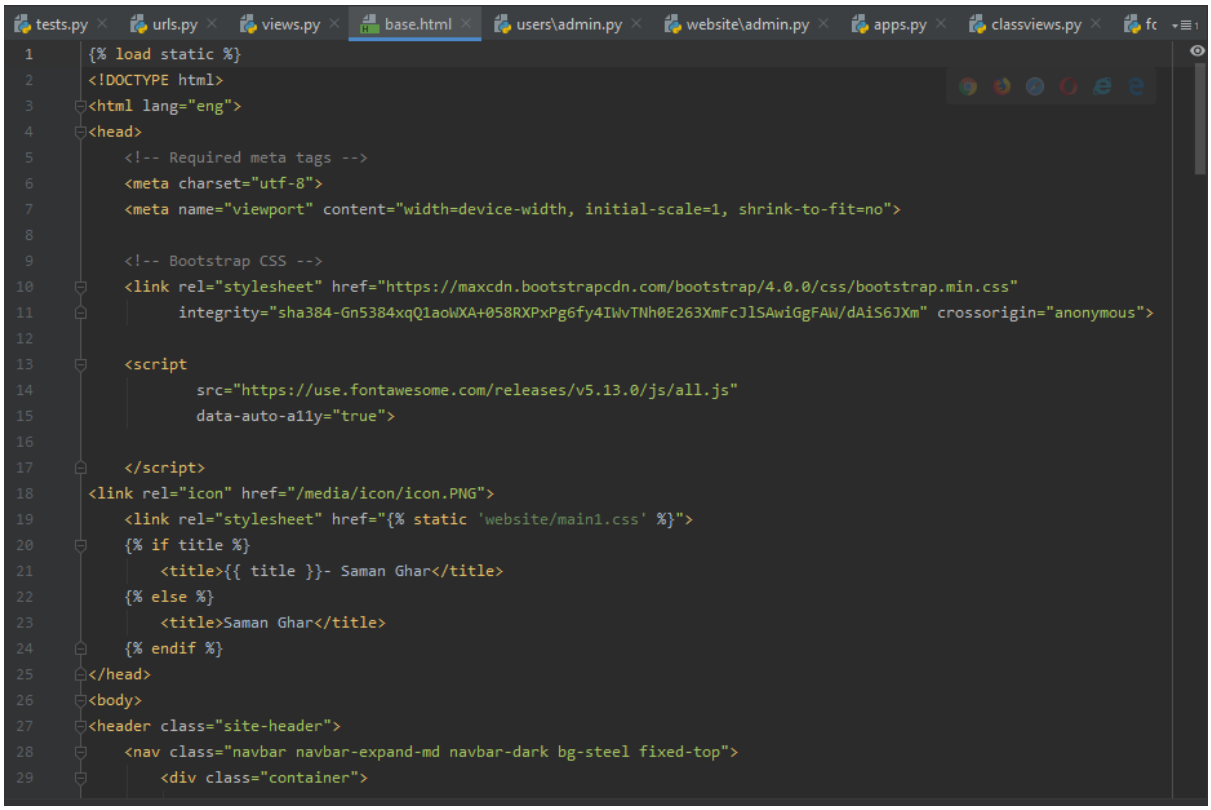


Figure 134:base code

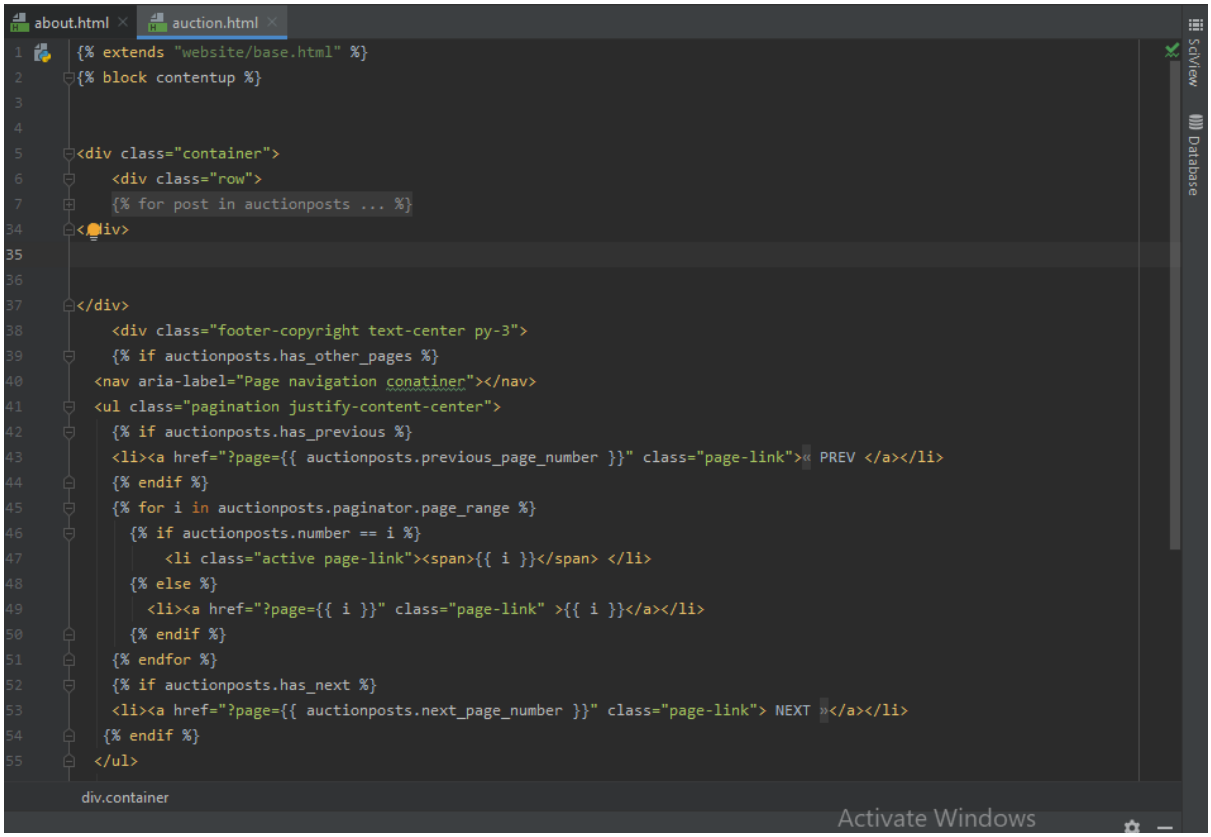


Figure 135:auction

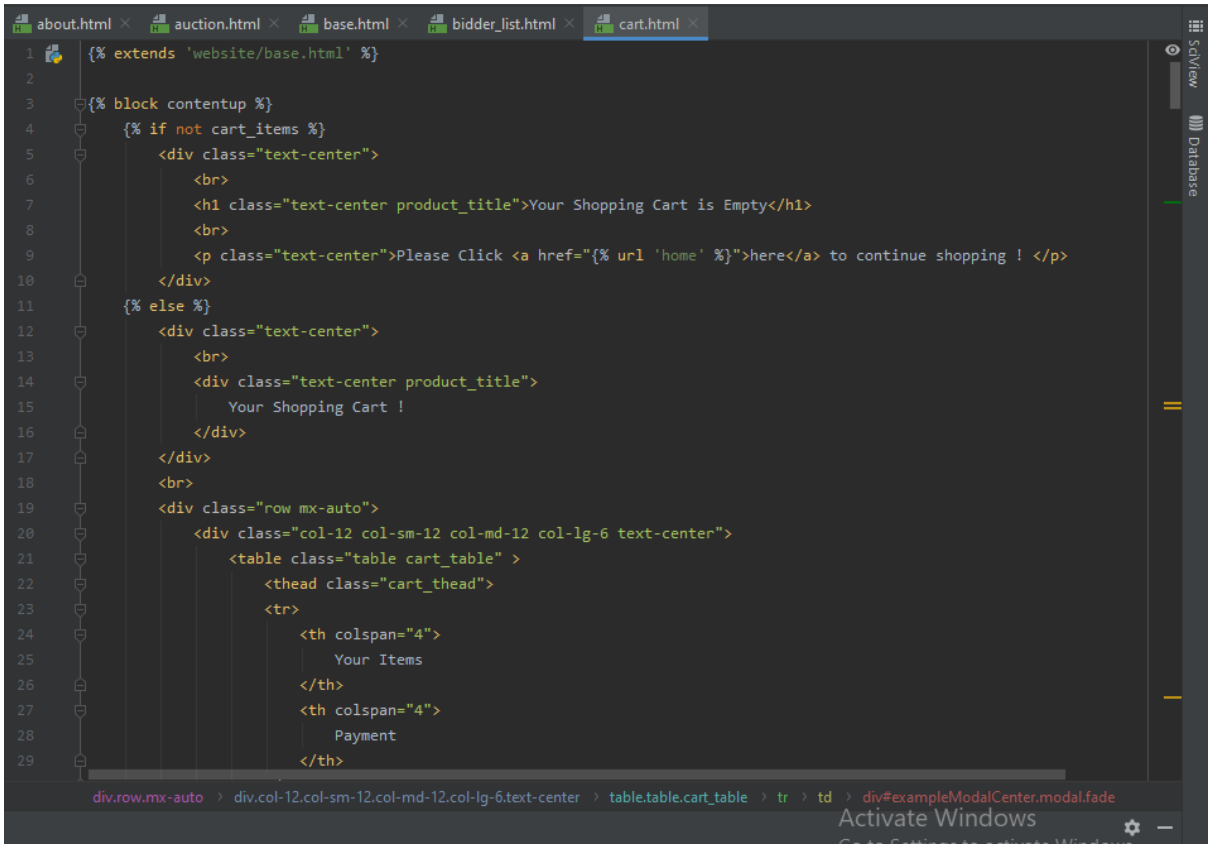


Figure 136:Cart

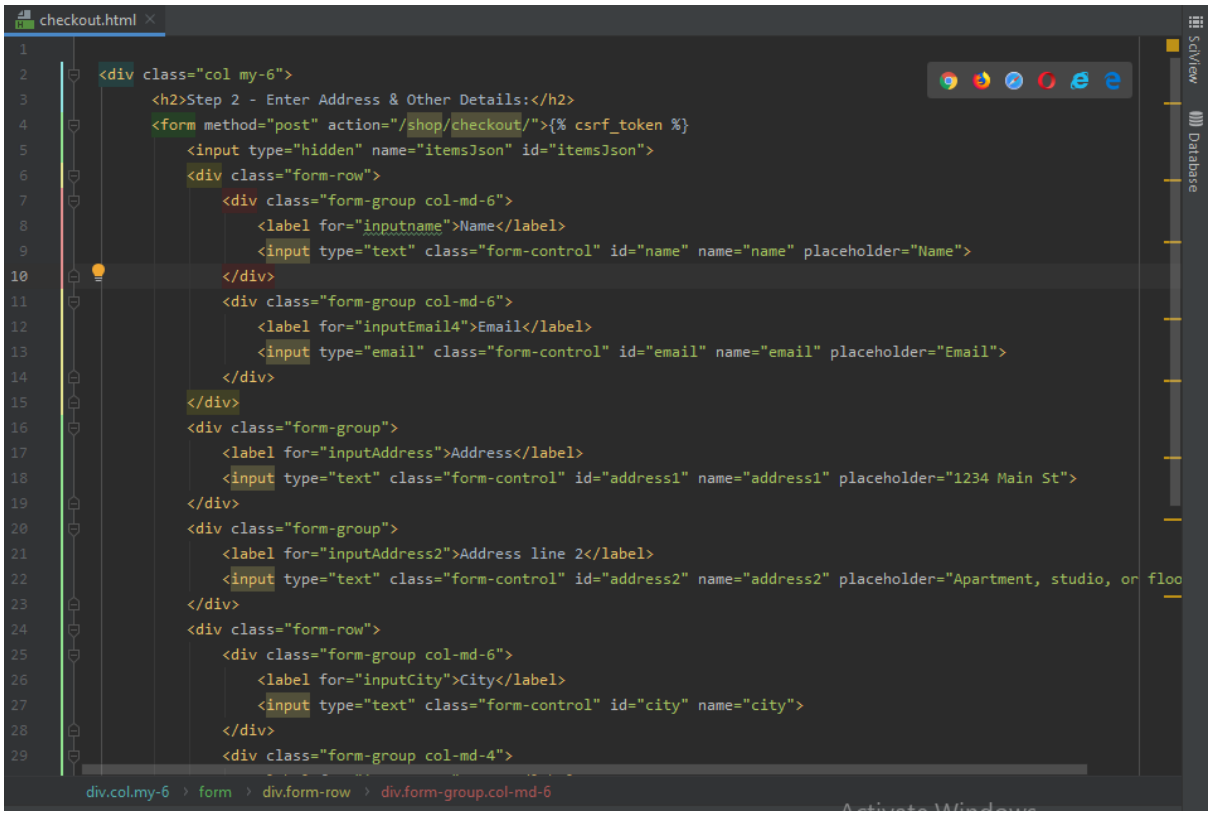


Figure 137:checkout

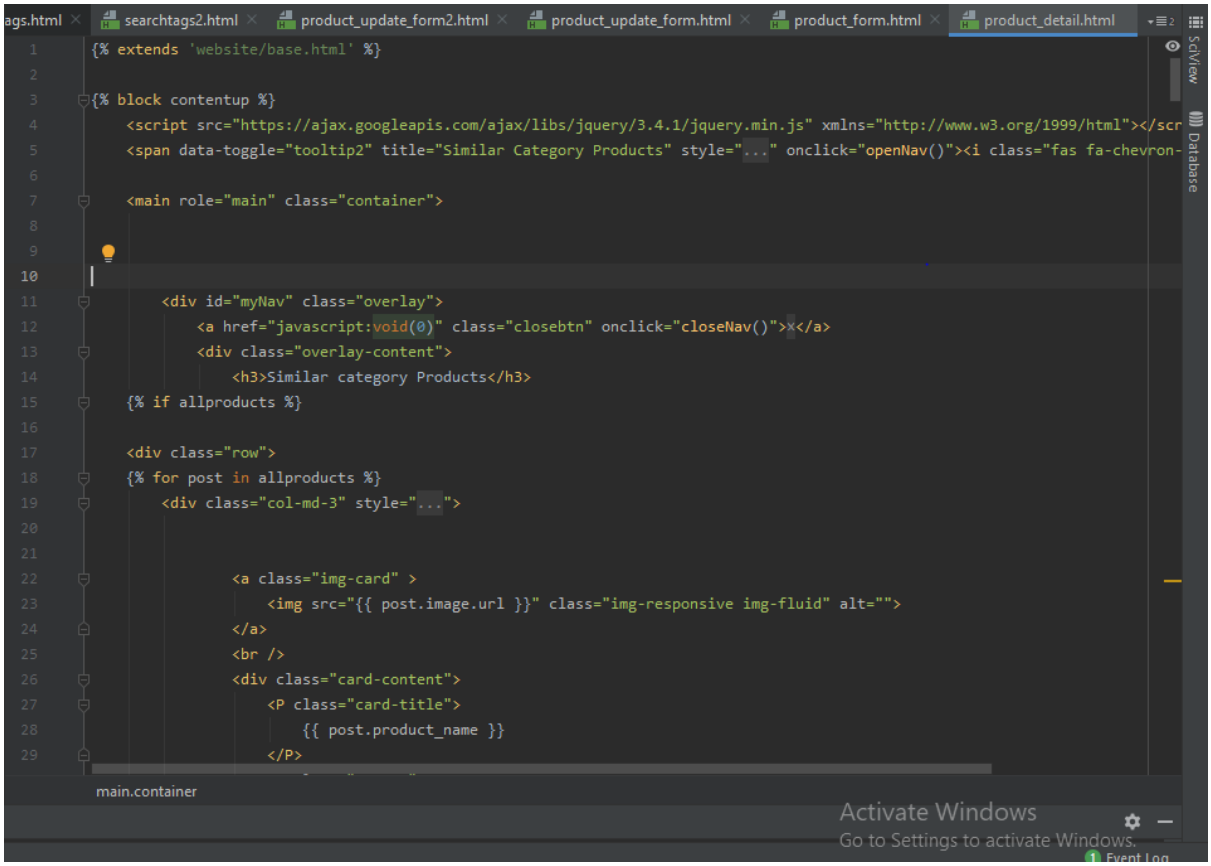


Figure 138:Product detail

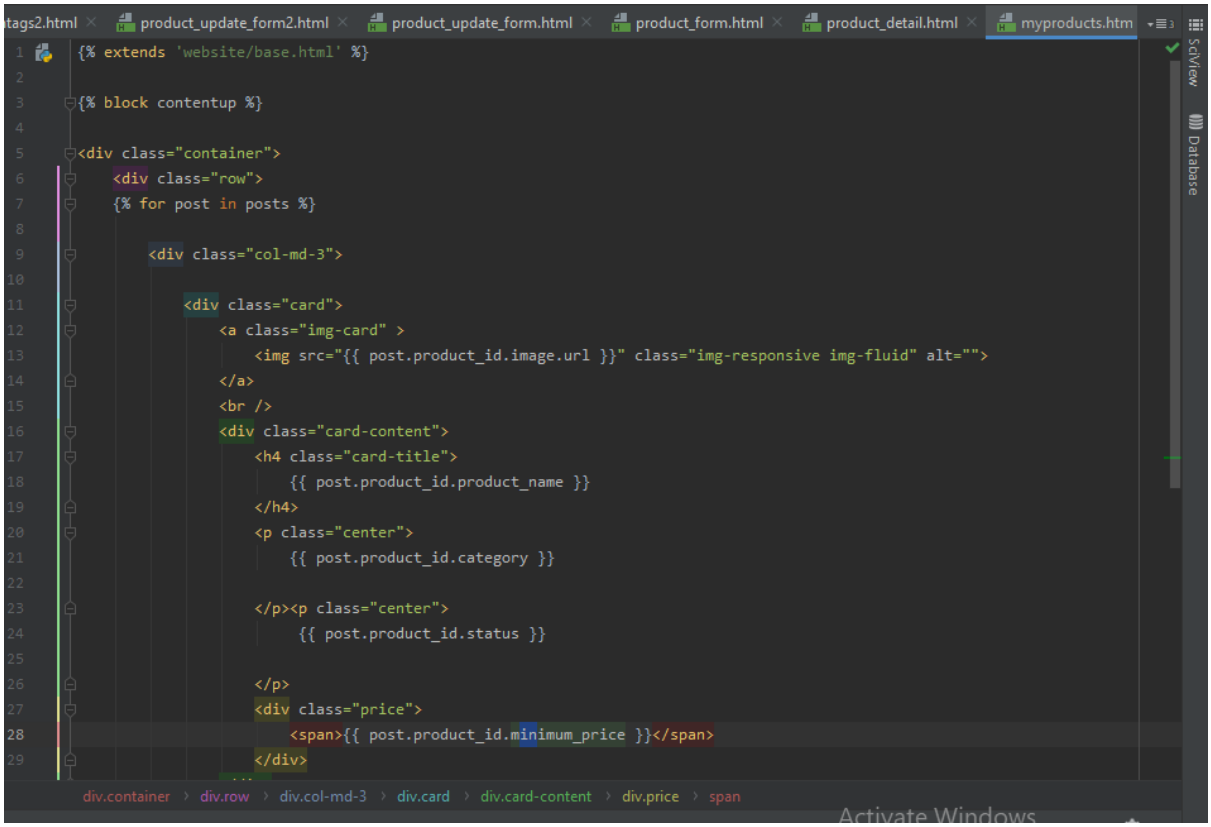


Figure 139:My Products

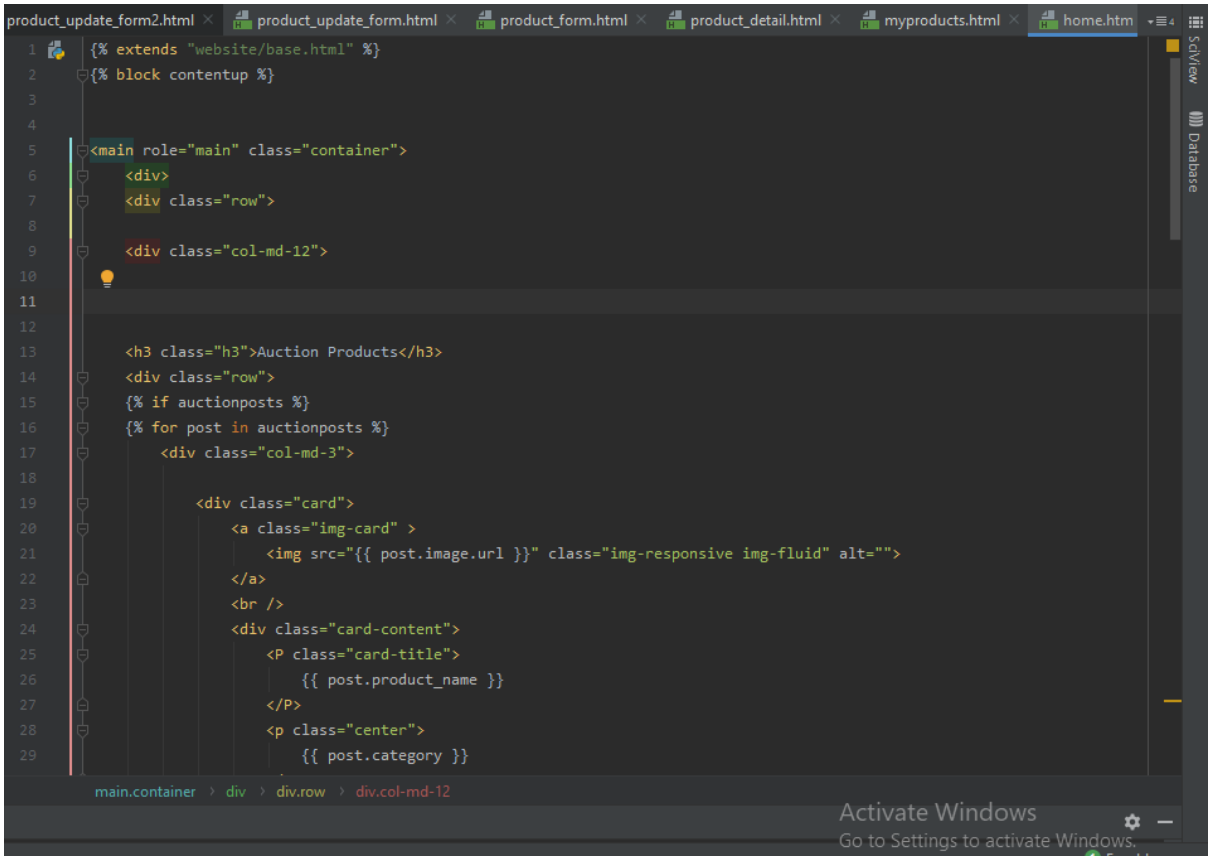


Figure 140:Home

8.3.2 SAMPLE CODE FOR THE SYSTEM

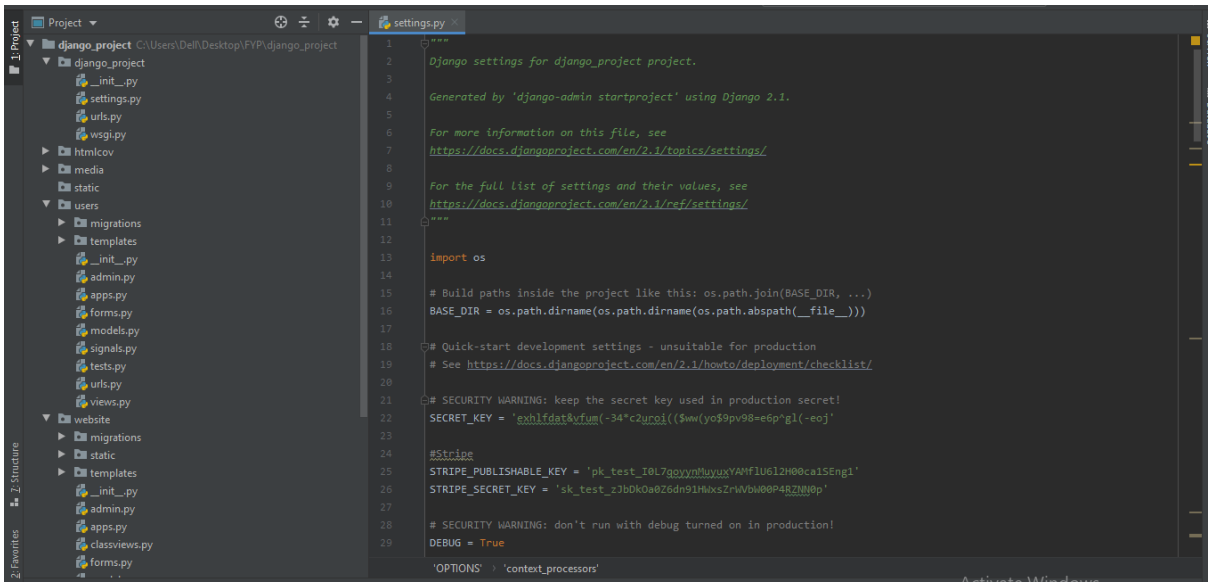


Figure 141: Django Settings

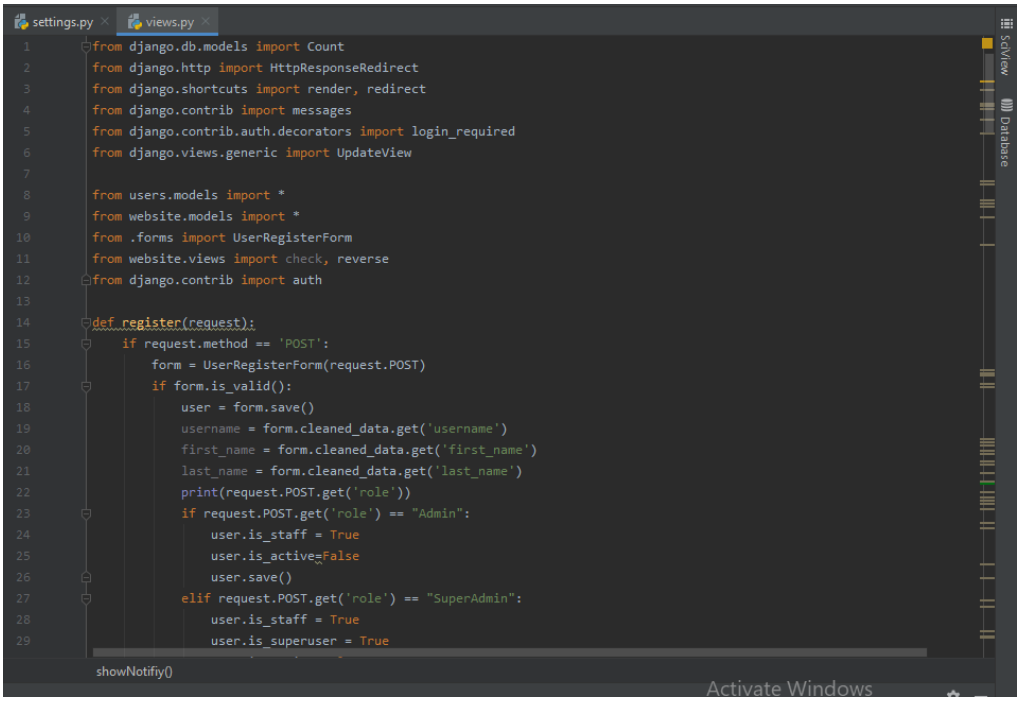


Figure 142:User Views

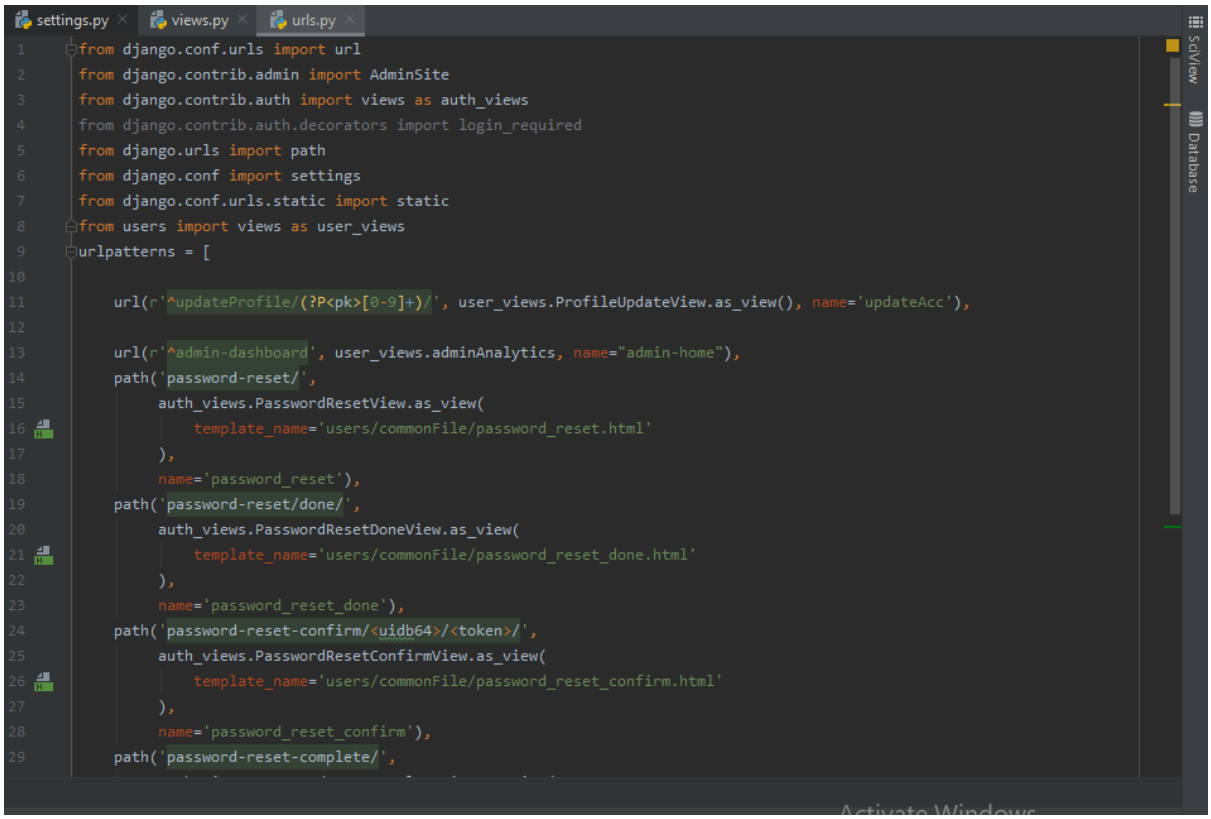


Figure 143:User urls

settings.py | views.py | tests.py | signals.py | urls.py

```python
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver
from .models import Profile


@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)


@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()

```

Figure 144:Signals

settings.py | views.py | tests.py | signals.py | models.py | urls.py

```python
from django.db import models
from django.contrib.auth.models import User

from django.utils import timezone
from website.models import Product


class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    billingName = models.CharField(max_length=250, blank=True)
    billingAddress1 = models.CharField(max_length=250, blank=True)
    billingCity = models.CharField(max_length=250, blank=True)
    billingPostcode = models.CharField(max_length=250, blank=True)
    billingCountry = models.CharField(max_length=250, blank=True, default='Nepal')

    def __str__(self):
        return f'{self.user.username} Profile'


class Conversation(models.Model):
    created = models.DateTimeField(default=timezone.now)
    user_name = models.ForeignKey(User, on_delete=models.CASCADE)
    product_id = models.ForeignKey(Product, on_delete=models.CASCADE)
    convo = models.CharField(max_length=300, default="")


class Notify(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    message = models.CharField(max_length=500, default="This notification is send to you by default.")
```

Figure 145:Models

Figure 146:Forms

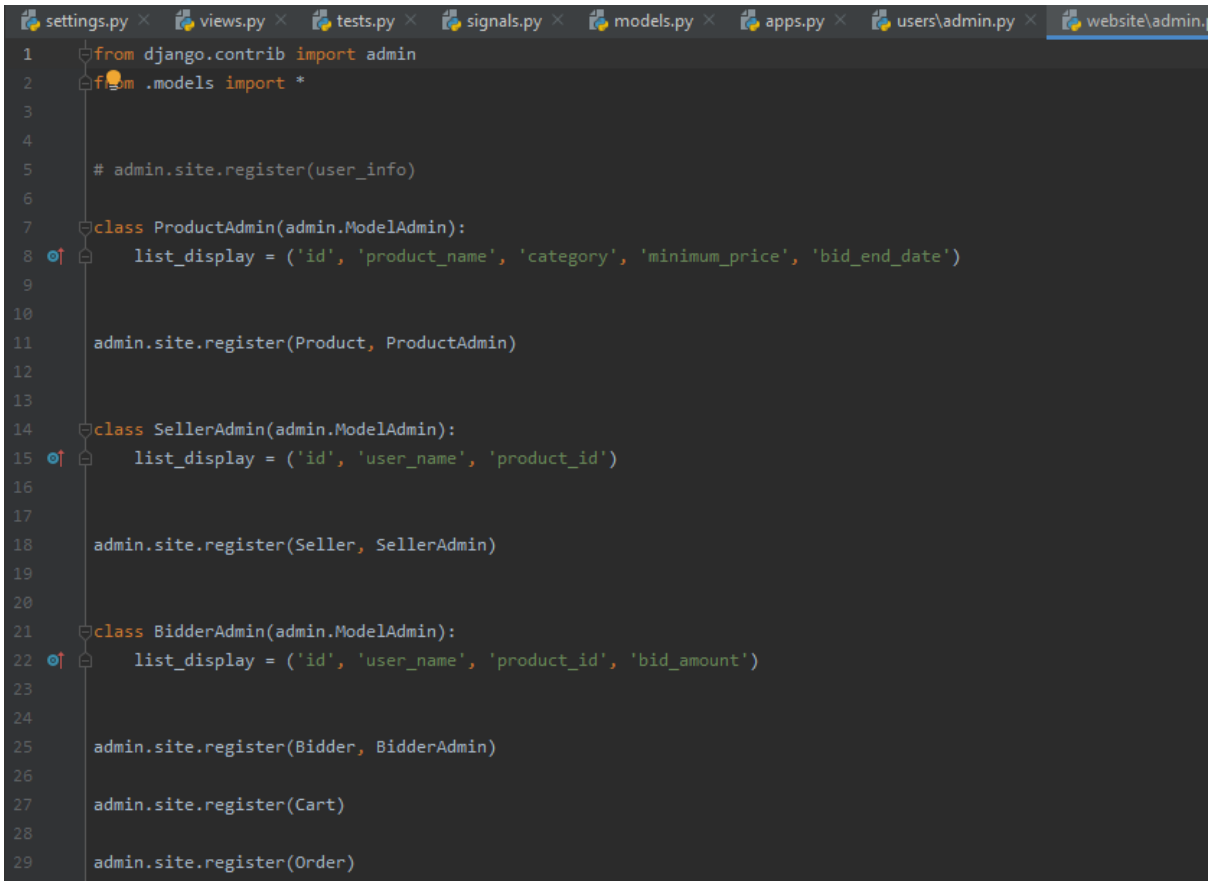


Figure 147:Admin

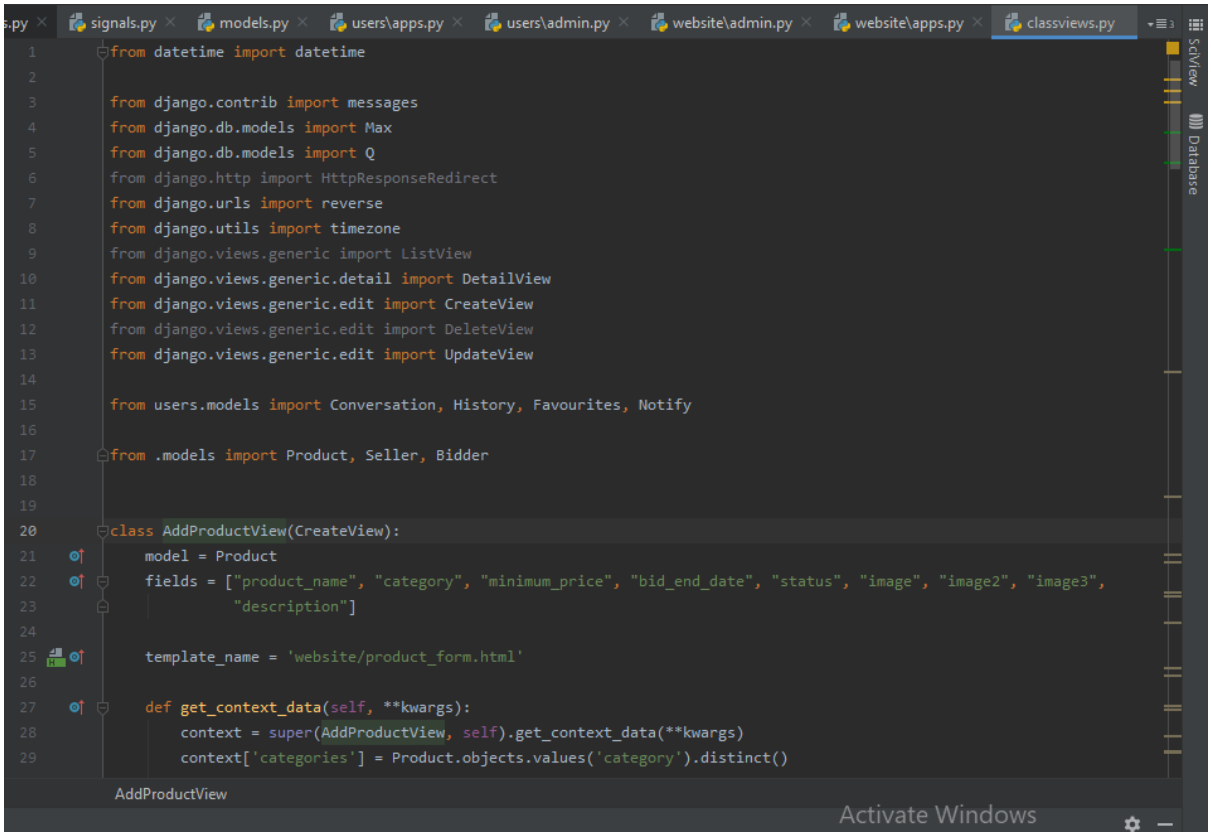


Figure 148: class Based Views

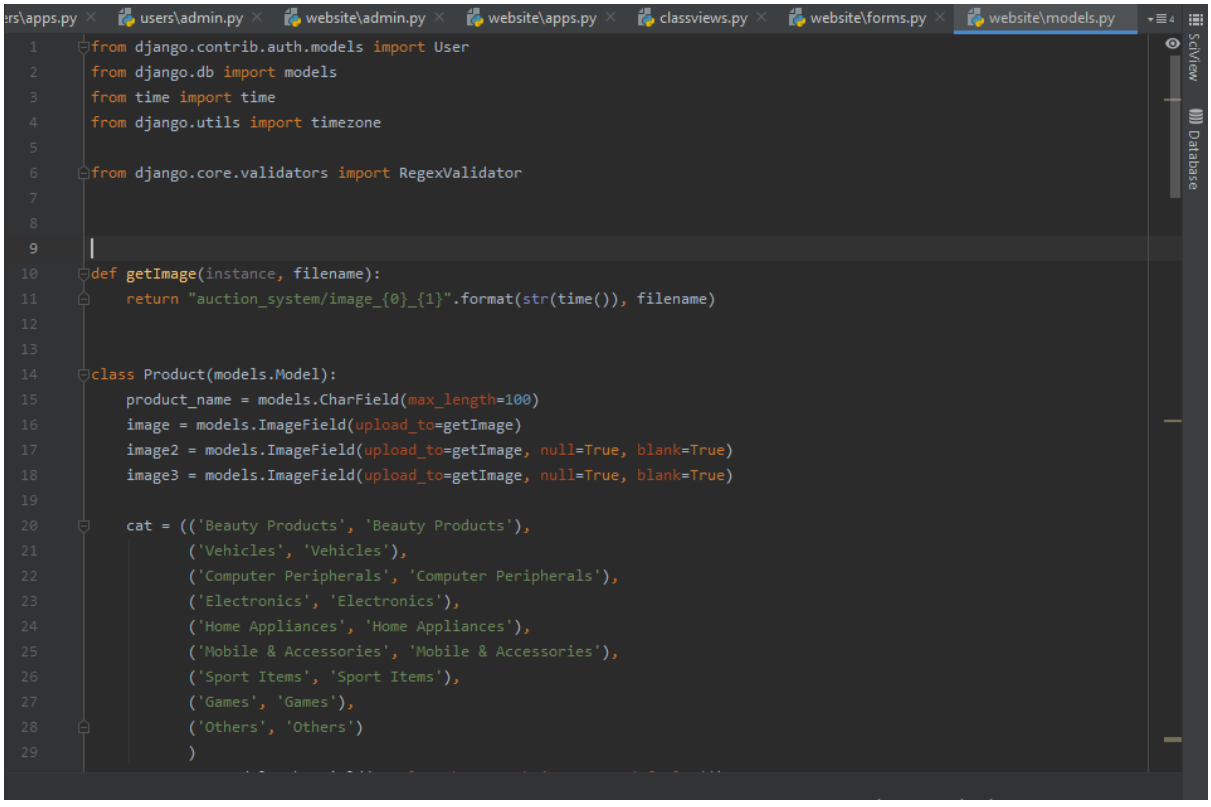


Figure 149:Models

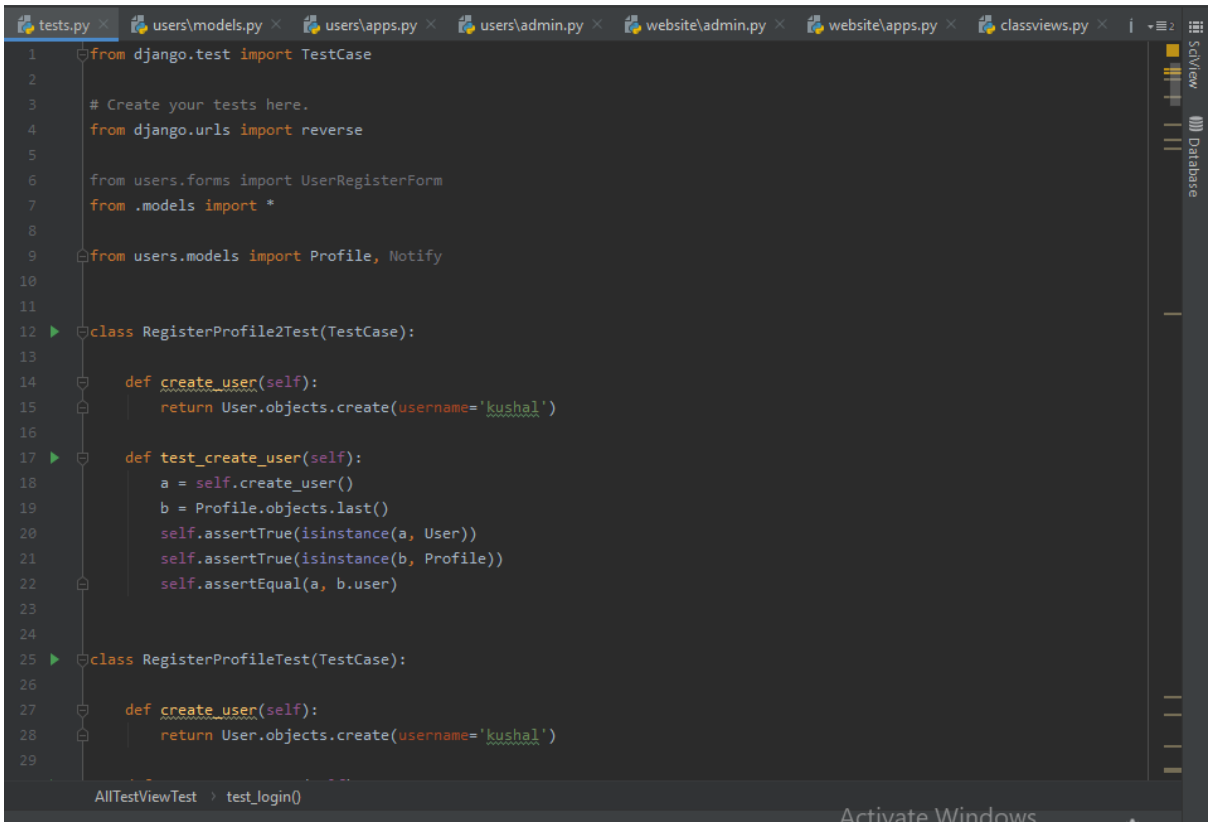


Figure 150:Tests

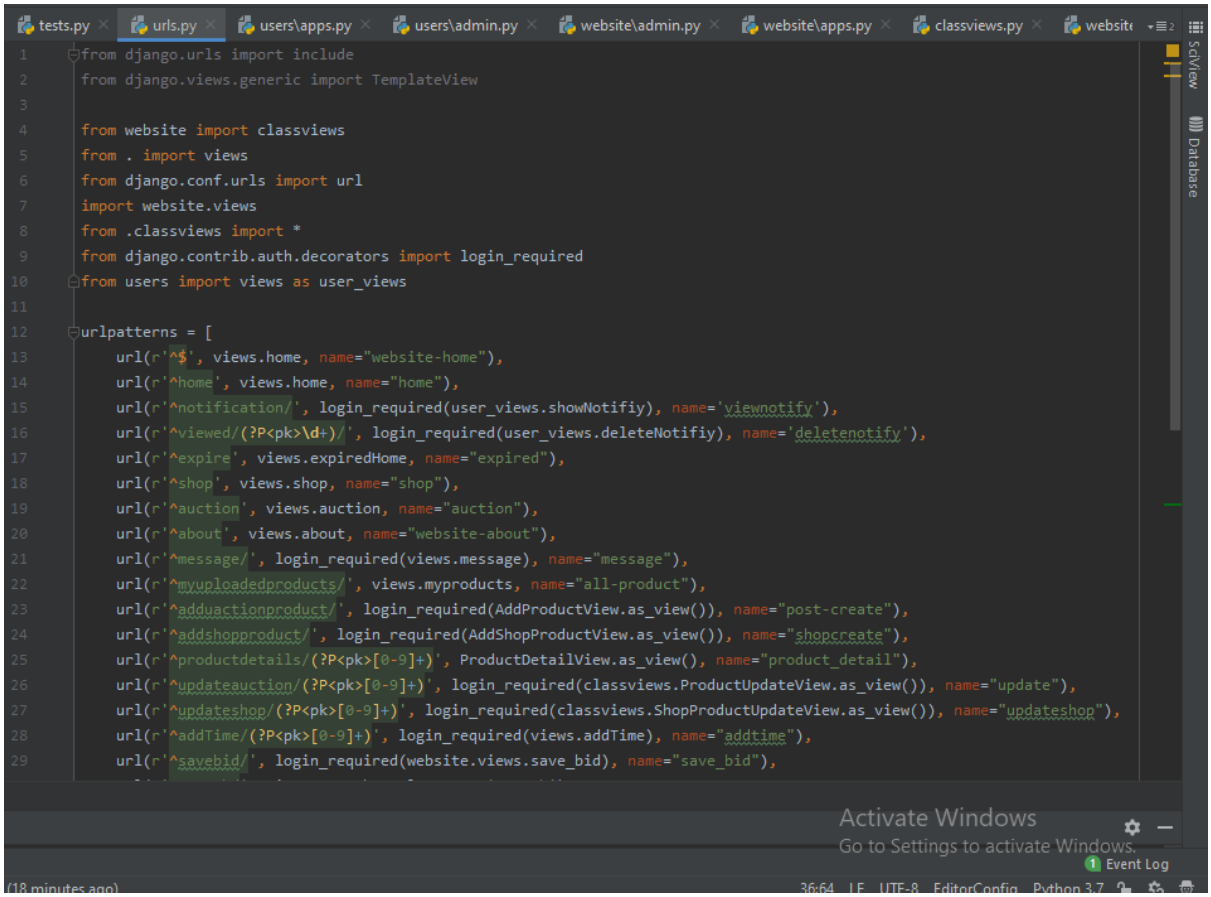


Figure 151:Website Urls

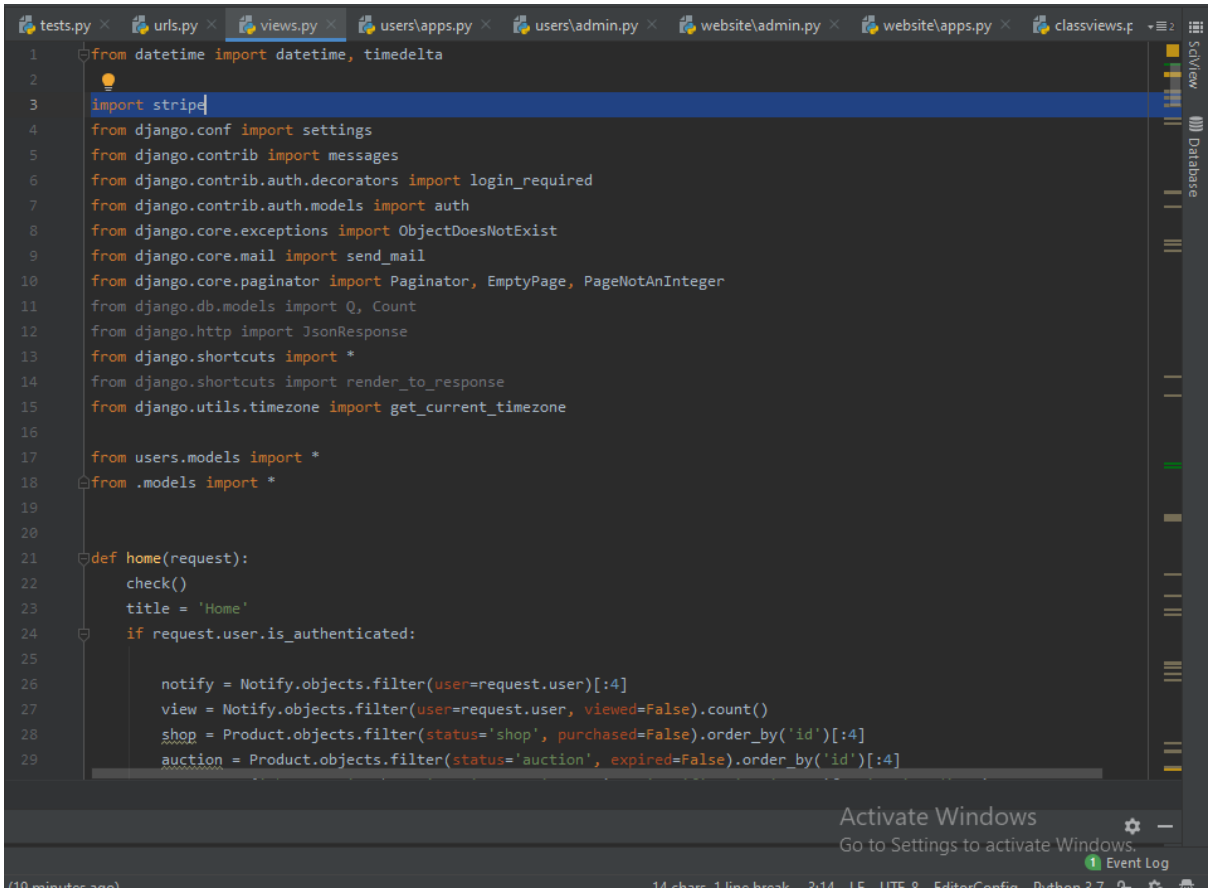
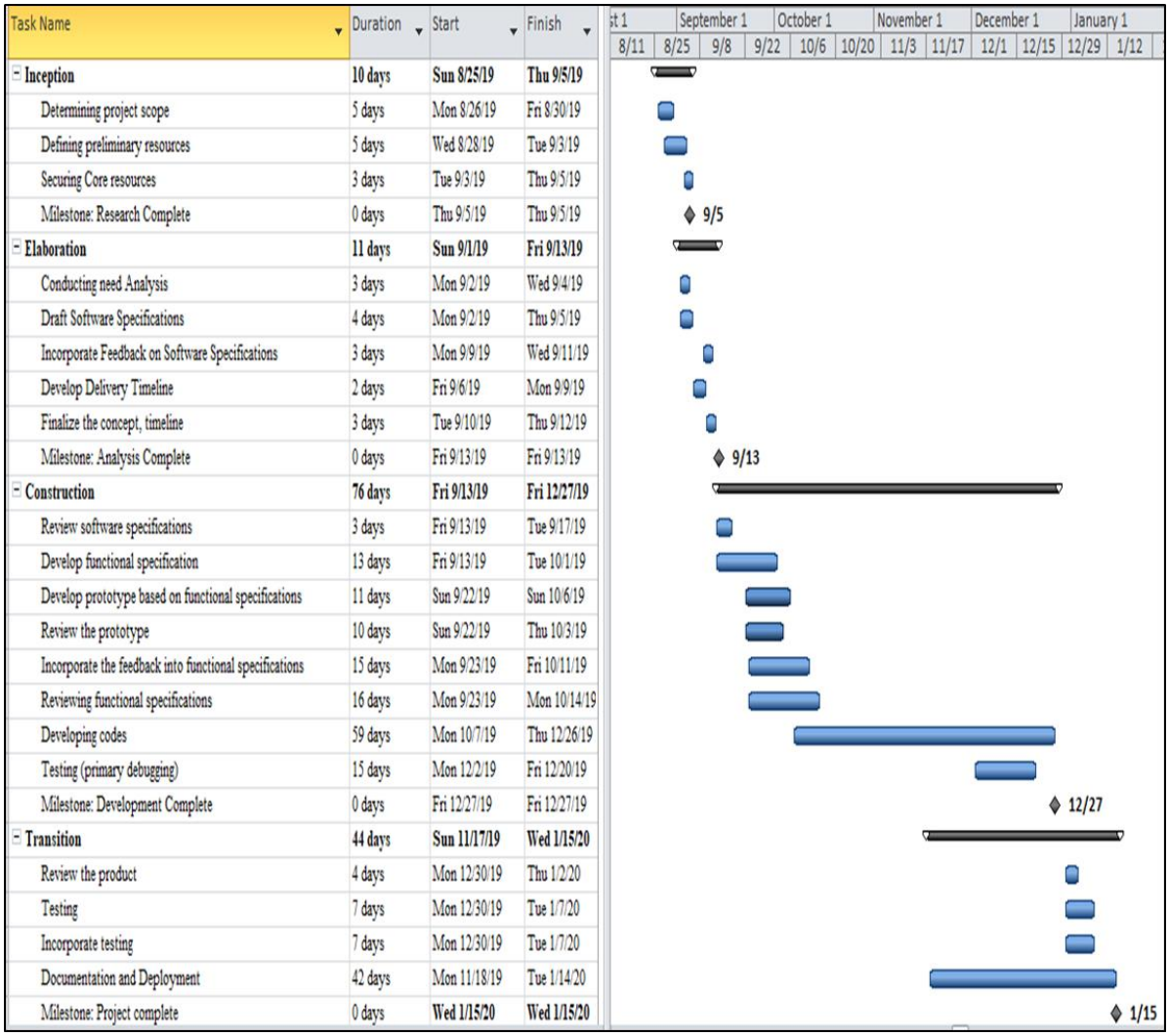


Figure 152: Function based website views

8.4 APPENDIX D: DESIGNS

8.4.1 GANTT CHART



Task Name	Duration	Start	Finish
Inception	**10 days**	**Sun 8/25/19**	**Thu 9/5/19**
Determining project scope	5 days	Mon 8/26/19	Fri 8/30/19
Defining preliminary resources	5 days	Wed 8/28/19	Tue 9/3/19
Securing Core resources	3 days	Tue 9/3/19	Thu 9/5/19
Milestone: Research Complete	0 days	Thu 9/5/19	Thu 9/5/19
Elaboration	**11 days**	**Sun 9/1/19**	**Fri 9/13/19**
Conducting need Analysis	3 days	Mon 9/2/19	Wed 9/4/19
Draft Software Specifications	4 days	Mon 9/2/19	Thu 9/5/19
Incorporate Feedback on Software Specifications	3 days	Mon 9/9/19	Wed 9/11/19
Develop Delivery Timeline	2 days	Fri 9/6/19	Mon 9/9/19
Finalize the concept, timeline	3 days	Tue 9/10/19	Thu 9/12/19
Milestone: Analysis Complete	0 days	Fri 9/13/19	Fri 9/13/19
Construction	**76 days**	**Fri 9/13/19**	**Fri 12/27/19**
Review software specifications	3 days	Fri 9/13/19	Tue 9/17/19
Develop functional specification	13 days	Fri 9/13/19	Tue 10/1/19
Develop prototype based on functional specifications	11 days	Sun 9/22/19	Sun 10/6/19
Review the prototype	10 days	Sun 9/22/19	Thu 10/3/19
Incorporate the feedback into functional specifications	15 days	Mon 9/23/19	Fri 10/11/19
Reviewing functional specifications	16 days	Mon 9/23/19	Mon 10/14/19
Developing codes	59 days	Mon 10/7/19	Thu 12/26/19
Testing (primary debugging)	15 days	Mon 12/2/19	Fri 12/20/19
Milestone: Development Complete	0 days	Fri 12/27/19	Fri 12/27/19
Transition	**44 days**	**Sun 11/17/19**	**Wed 1/15/20**
Review the product	4 days	Mon 12/30/19	Thu 1/2/20
Testing	7 days	Mon 12/30/19	Tue 1/7/20
Incorporate testing	7 days	Mon 12/30/19	Tue 1/7/20
Documentation and Deployment	42 days	Mon 11/18/19	Tue 1/14/20
Milestone: Project complete	0 days	**Wed 1/15/20**	**Wed 1/15/20**

Figure 153: Gantt chart 1

	Task Name	Duration	Start	Finish
1	**Inception**	**10 days**	**Sun 8/25/19**	**Thu 9/5/19**
2	Determining project scope	5 days	Mon 8/26/19	Fri 8/30/19
3	Defining preliminary resources	5 days	Wed 8/28/19	Tue 9/3/19
4	Securing Core resources	3 days	Tue 9/3/19	Thu 9/5/19
5	Milestone: Research Complete	0 days	Thu 9/5/19	Thu 9/5/19
6	**Elaboration**	**11 days**	**Sun 9/1/19**	**Fri 9/13/19**
7	Conducting need Analysis	3 days	Mon 9/2/19	Wed 9/4/19
8	Draft Software Specifications	4 days	Mon 9/2/19	Thu 9/5/19
9	Incorporate Feedback on Software Specifications	3 days	Mon 9/9/19	Wed 9/11/19
10	Develop Delivery Timeline	2 days	Fri 9/6/19	Mon 9/9/19
11	Finalize the concept, timeline	3 days	Tue 9/10/19	Thu 9/12/19
12	Milestone: Analysis Complete	0 days	Fri 9/13/19	Fri 9/13/19
13	**Construction**	**119 days**	**Fri 9/13/19**	**Tue 2/25/20**
14	Review software specifications	3 days	Fri 9/13/19	Tue 9/17/19
15	Develop functional specification	13 days	Fri 9/13/19	Tue 10/1/19
16	Develop prototype based on functional specifications	11 days	Sun 9/22/19	Sun 10/6/19
17	Review the prototype	10 days	Sun 9/22/19	Thu 10/3/19
18	Incorporate the feedback into functional specifications	15 days	Mon 9/23/19	Fri 10/11/19
19	Reviewing functional specifications	16 days	Mon 9/23/19	Mon 10/14/19
20	Developing codes	102 days	Tue 10/8/19	Tue 2/25/20
21	Testing (primary debugging)	9 days	Mon 2/10/20	Thu 2/20/20
22	Milestone: Development Complete	0 days	Tue 2/25/20	Tue 2/25/20
23	**Transition**	**33 days**	**Sat 2/1/20**	**Tue 3/17/20**
24	Review the product	8 days	Tue 2/25/20	Thu 3/5/20
25	Testing	8 days	Tue 2/25/20	Thu 3/5/20
26	Incorporate testing	8 days	Tue 2/25/20	Thu 3/5/20
27	Documentation and Deployment	33 days	Sat 2/1/20	Tue 3/17/20
28	Milestone: Project complete	0 days	**Tue 3/17/20**	**Tue 3/17/20**

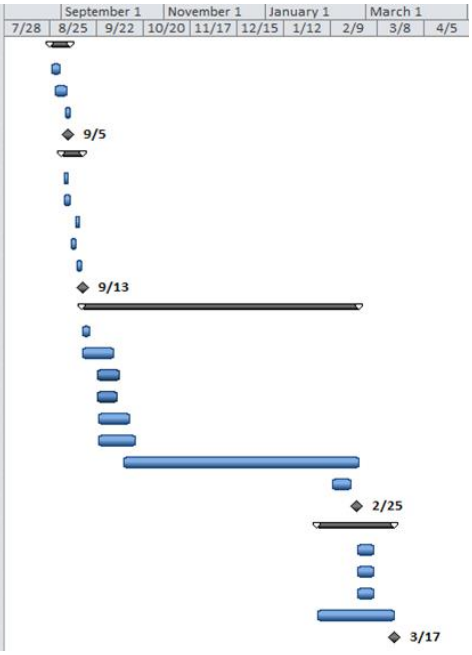


Figure 154: Gantt chart 2

8.4.2 WORK BREAKDOWN STRUCTURE

SN	Structure	Description	Time
1	**Inception**	Start of the project where research will be conducted and resources will be collected.	10- 15 days
	1.1	Determining project scope	
	1.2	Defining preliminary resources	
	1.3	Securing Core resources	
Research Complete			
2	**Elaboration**	Analysis of the project, Scope, and model creation, timeline creating use cases.	10- 15 days
	2.1	Conducting need Analysis	
	2.2	Draft Software Specifications	
	2.3	Incorporate Feedback on Software Specifications	
	2.4	Develop models, timeline, strategies	
	2.5	Finalise the concept, timeline, strategies	
Analysis Complete			
3	**Construction**	Designing and coding of the website, testing, completing website design, checking functionalities, testing is done in this phase.	3 – 4 months
	3.1	Review software specifications	
	3.2	Develop functional specification	
	3.3	Develop prototype based on functional Specifications	
	3.4	Review the prototype	
	3.5	Incorporate the feedback into functional Specifications	
	3.6	Reviewing functional specifications	
	3.7	Developing codes	
	3.8	Testing (primary debugging)	
Development Complete			
4	**Transition**	Final Stages of the product and deployment of the website with last of testing and writing documentation.	1and half month
	4.1	Review the product	
	4.2	Testing	
	4.3	Incorporate testing result	
	4.4	Documentation and Deployment	
Project complete			

Table 27: Work breakdown structure

8.4.3 SEQUENCE DIAGRAMS

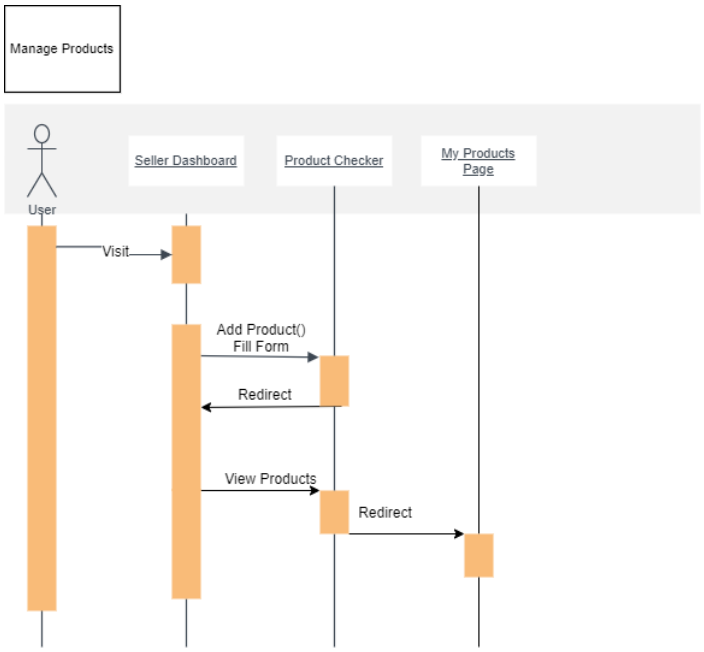


Figure 155:Manage Products

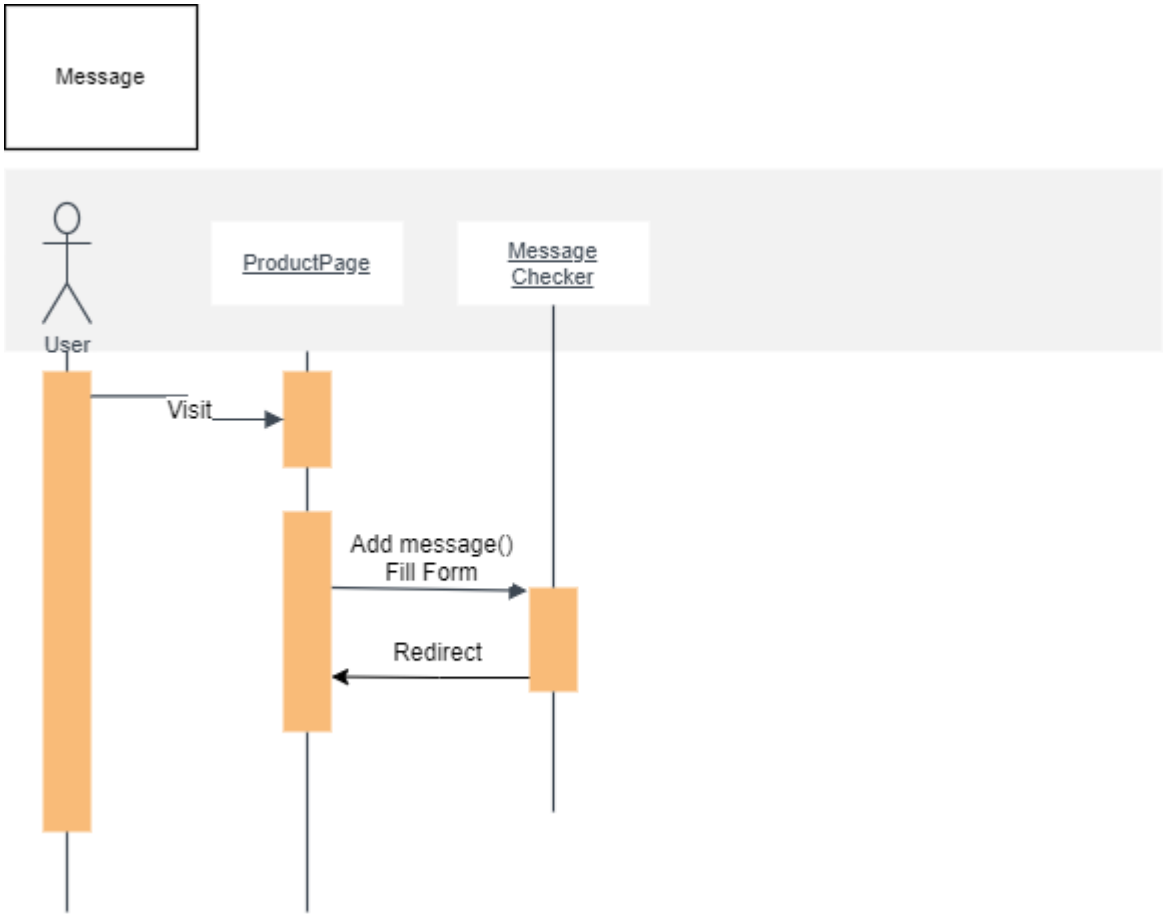


Figure 156:messages

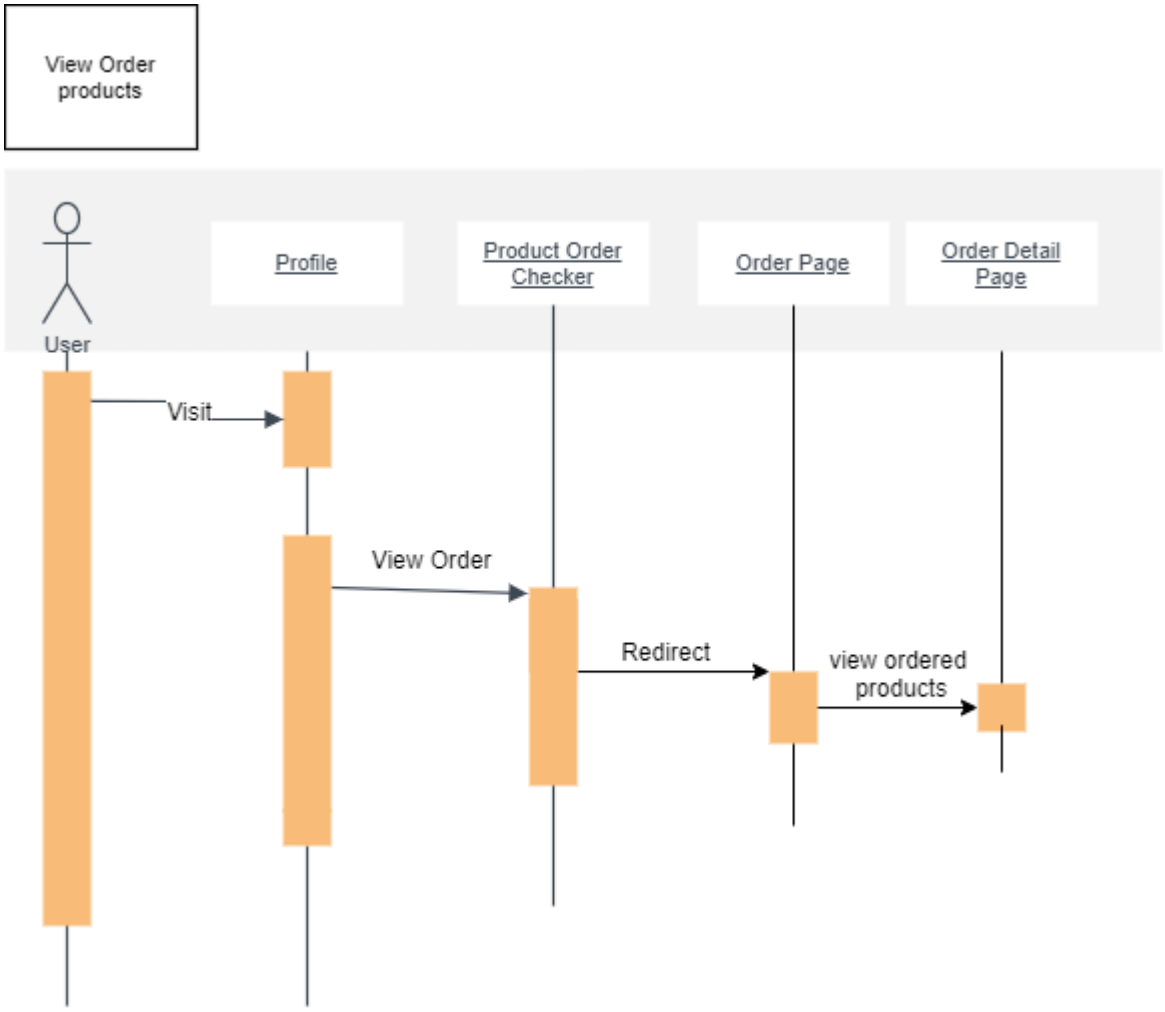


Figure 157:View Order Products

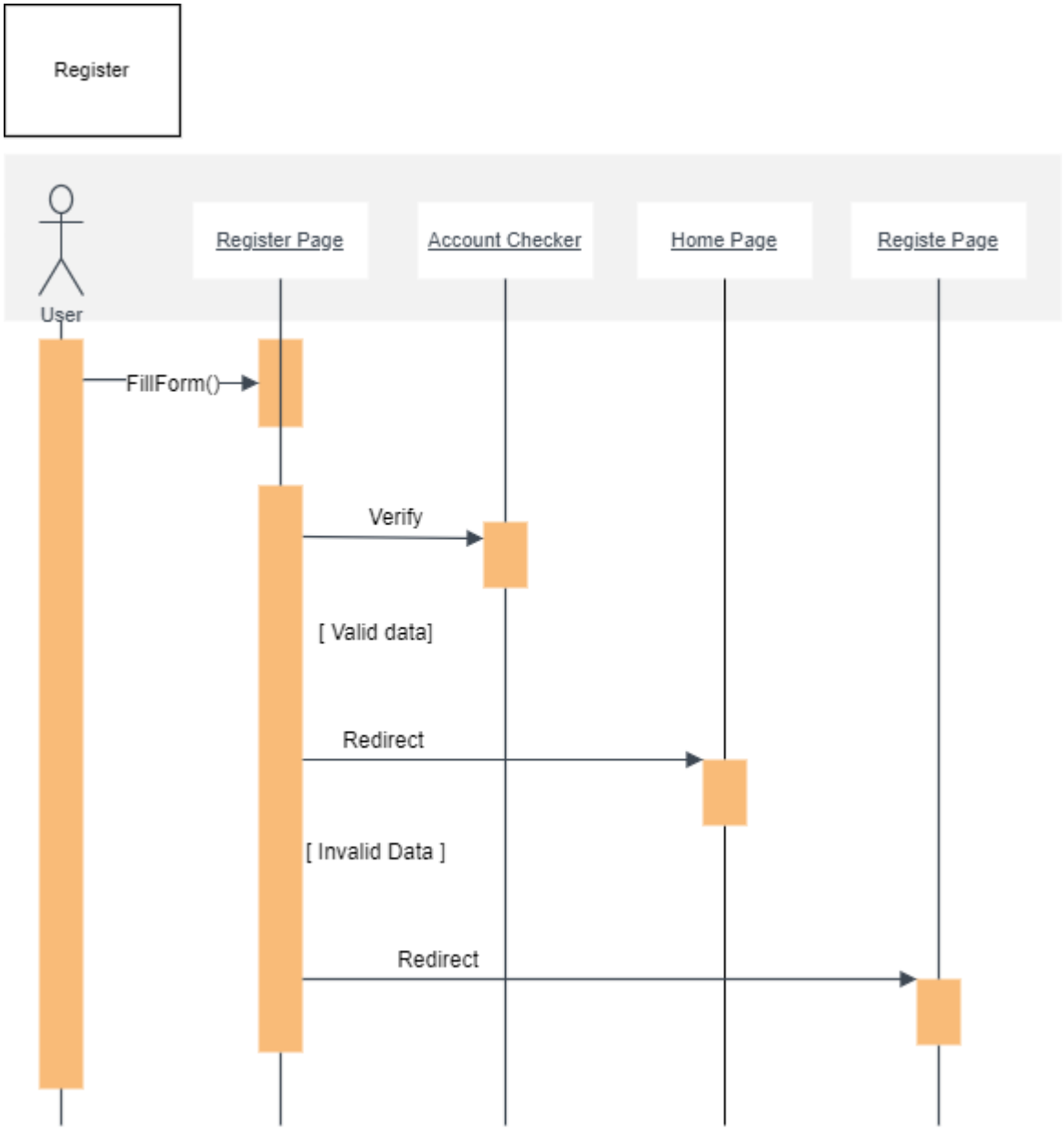


Figure 158:Register

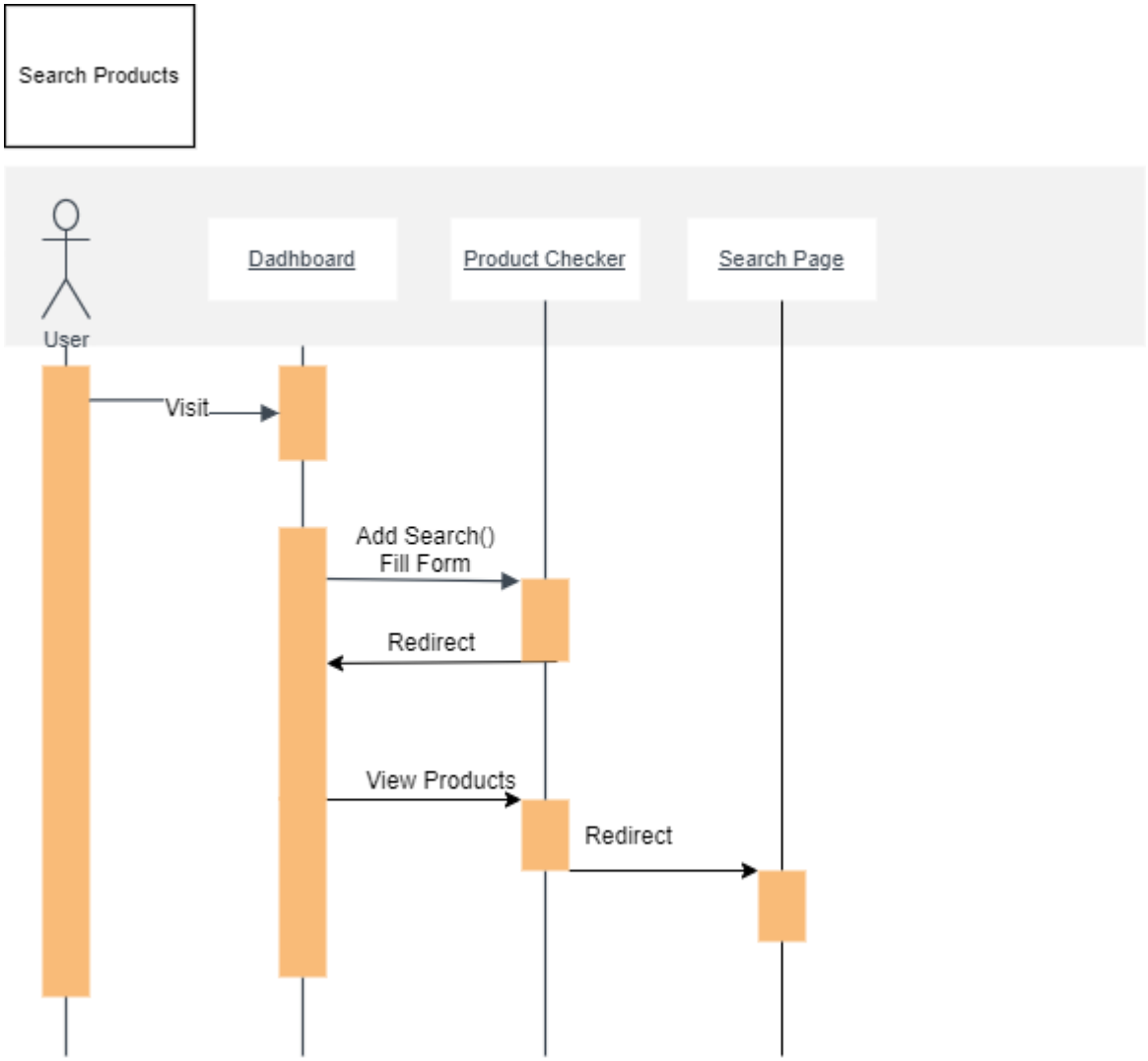


Figure 159: SearchProducts

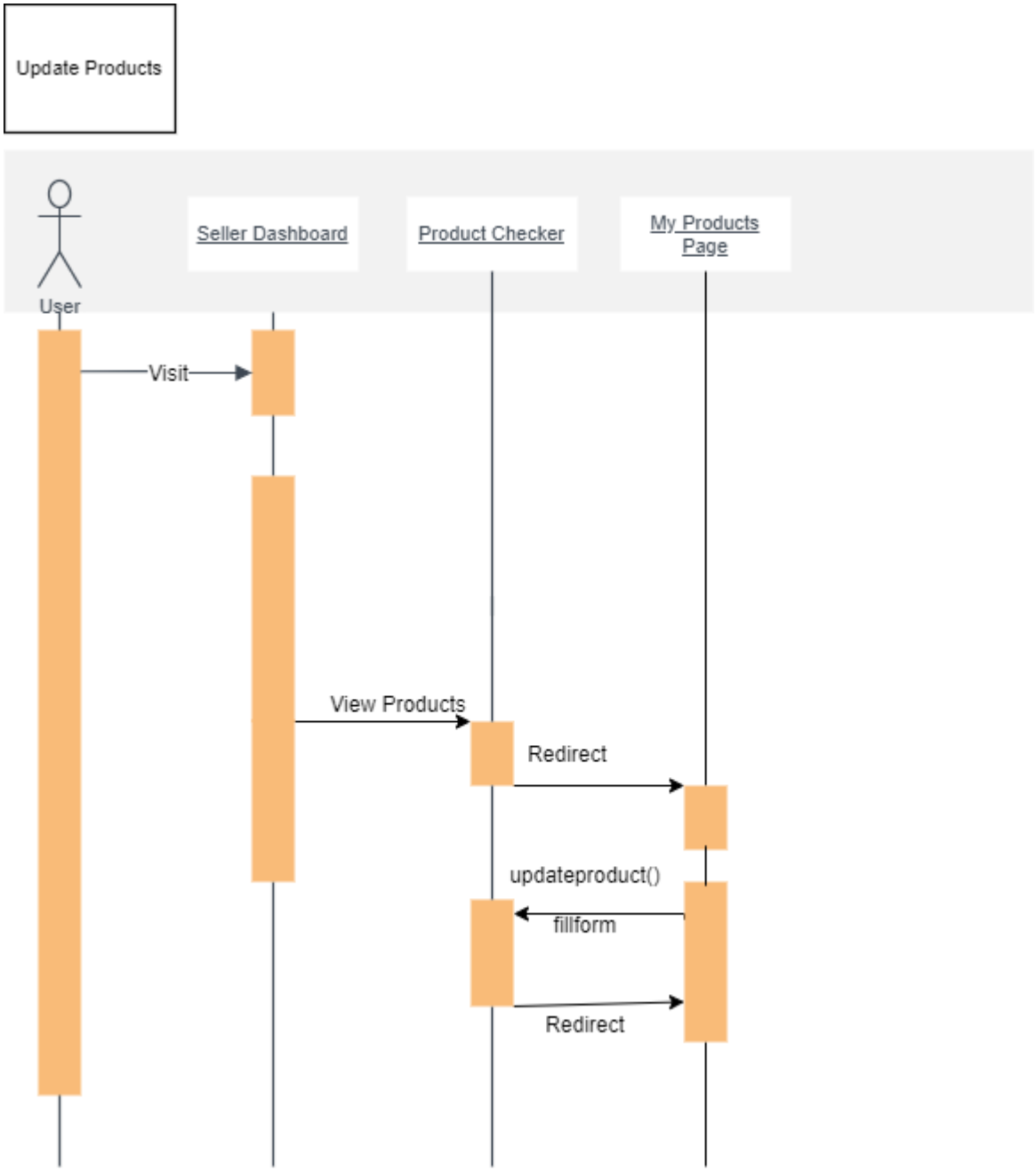


Figure 160: Update Products

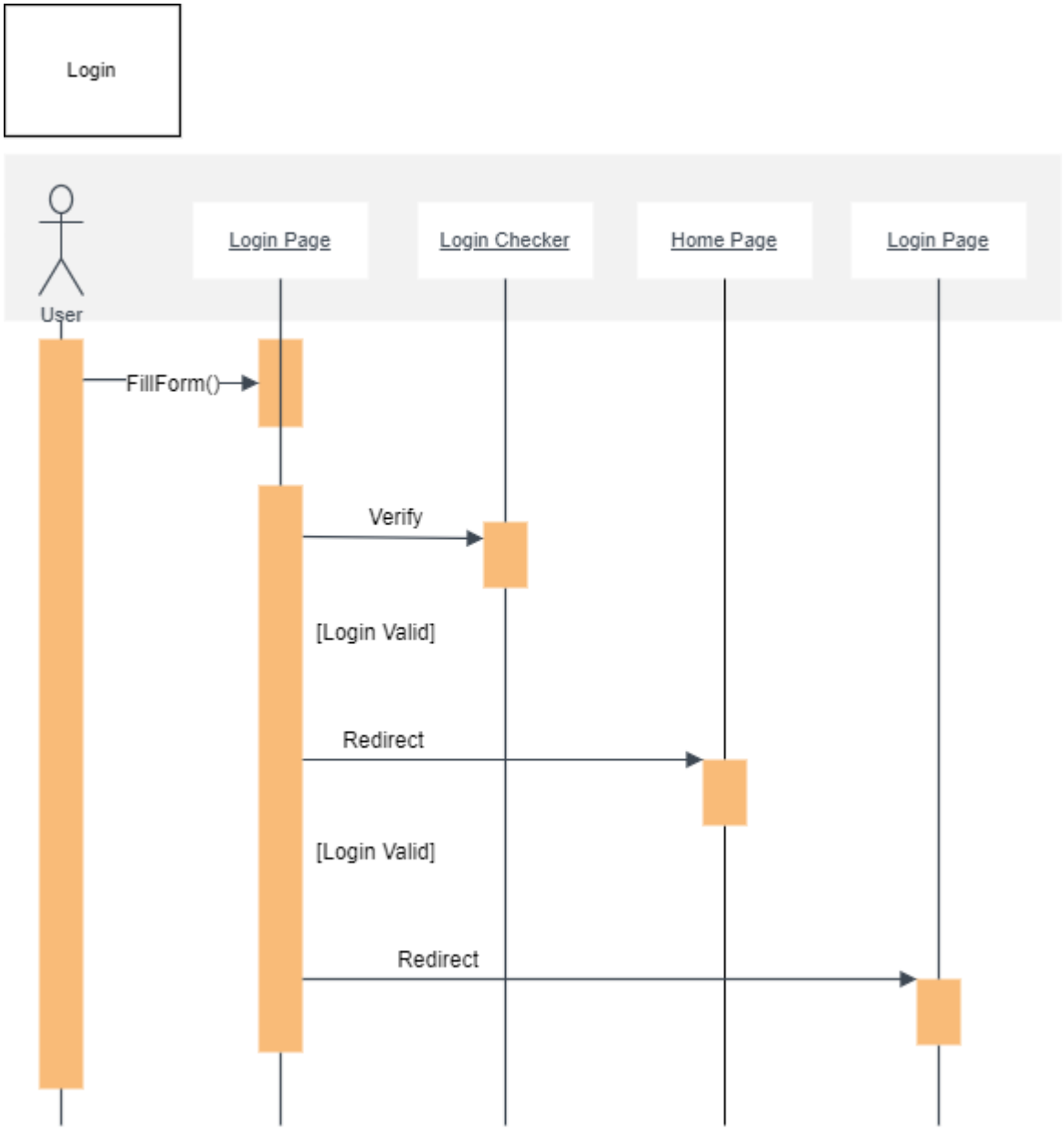


Figure 161: Login

8.4.4 ER-DIAGRAM

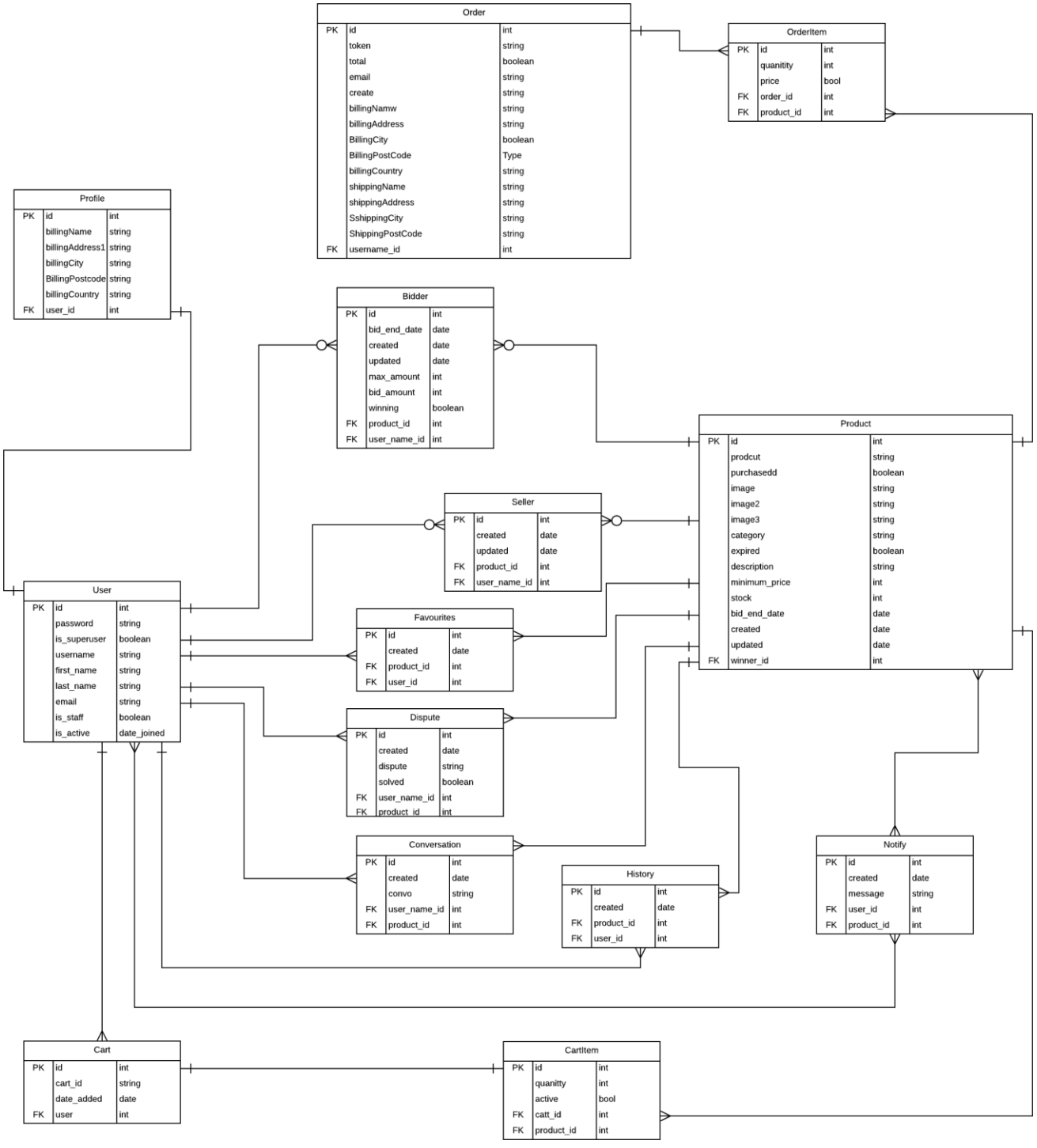


Figure 162:Final ER daigram

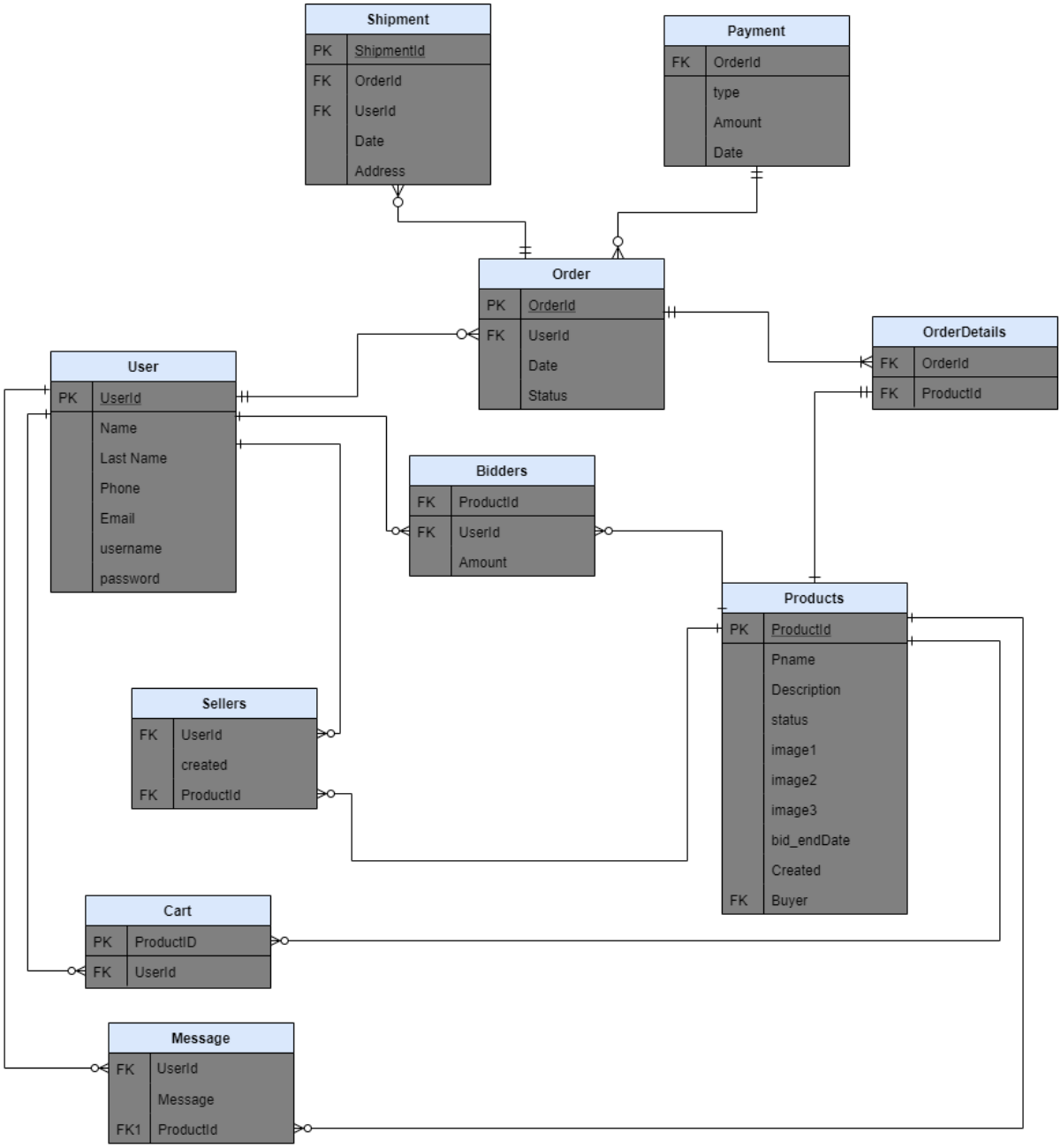


Figure 163: 1st Draft Diagram

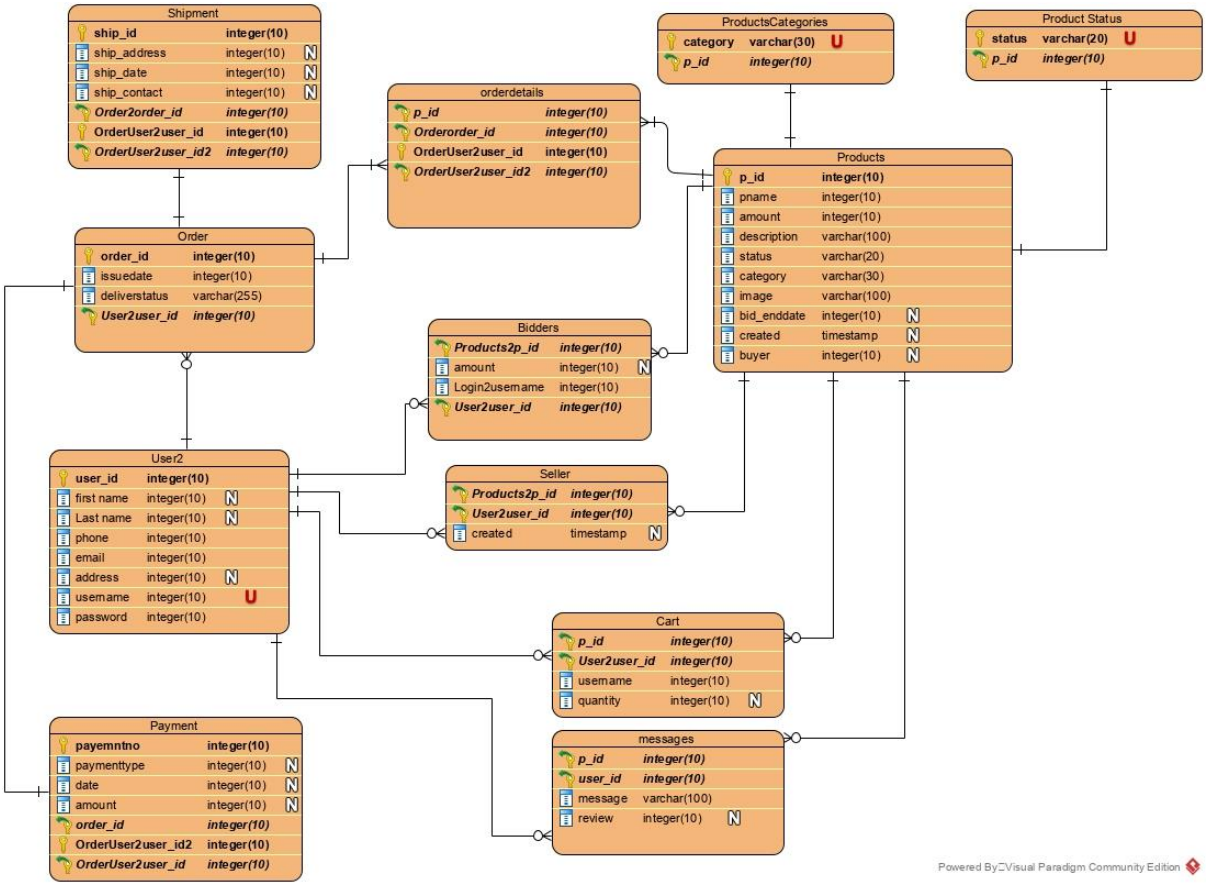


Figure 164: 2nd Draft

8.4.5 ACTIVITY DIAGRAM

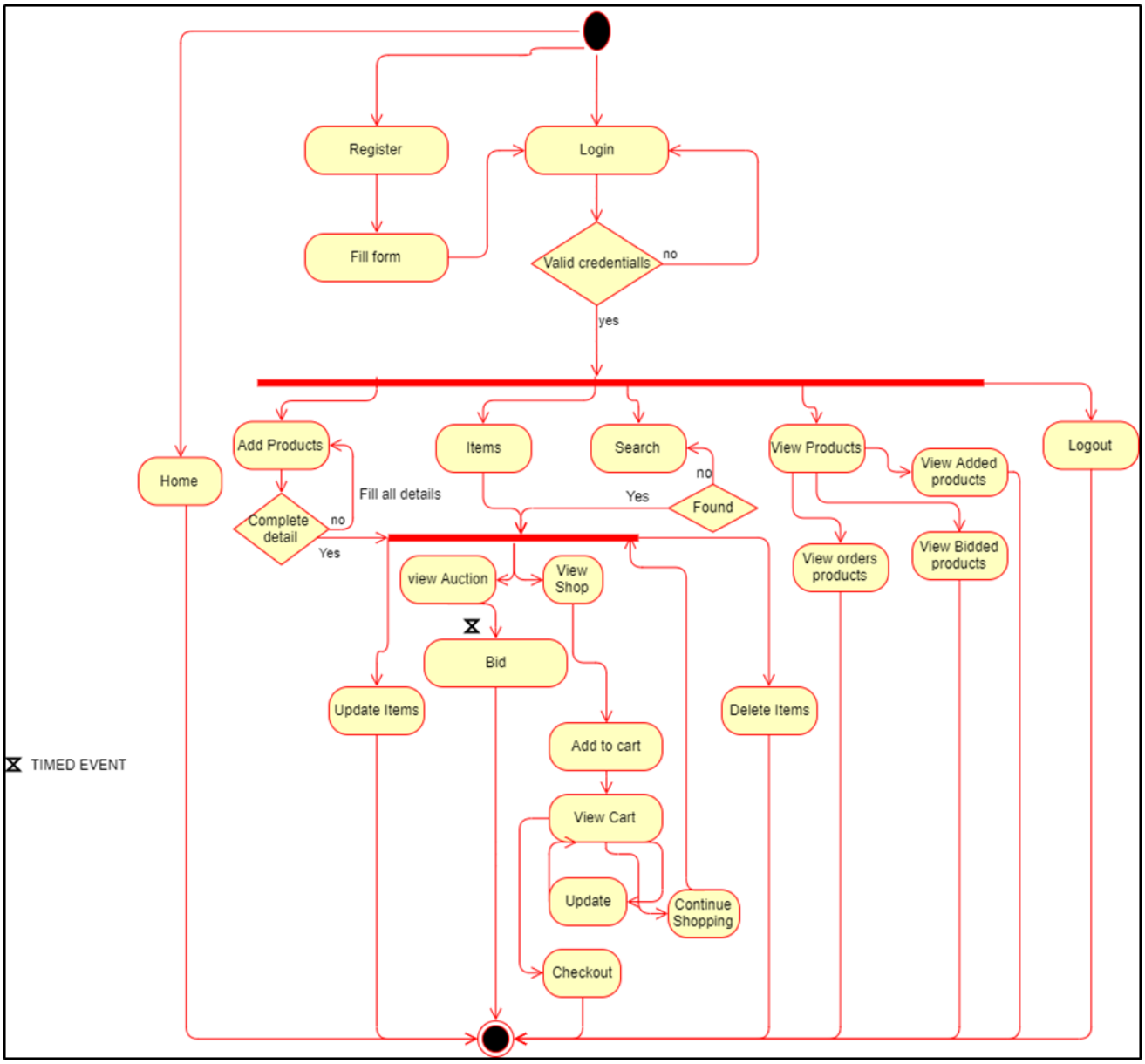


Figure 165:Activity diagram

8.4.6 DATA FLOW DIAGRAMS

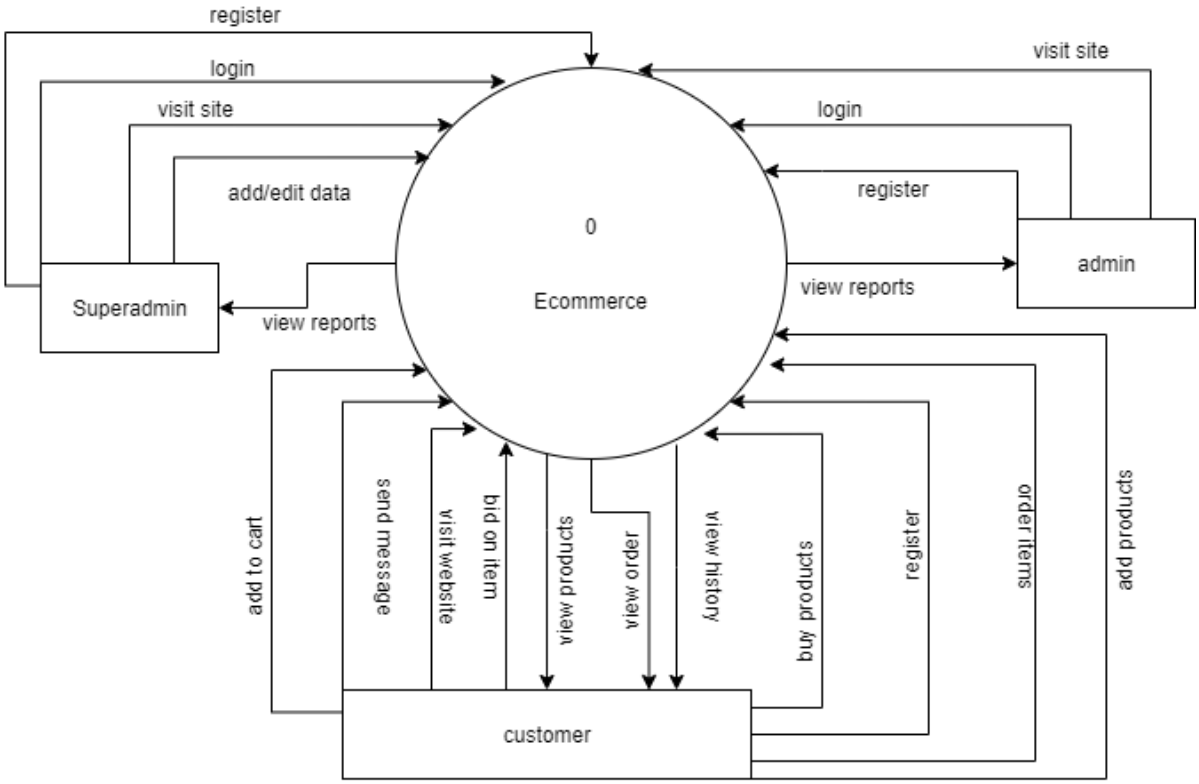


Figure 166:DFD level 0

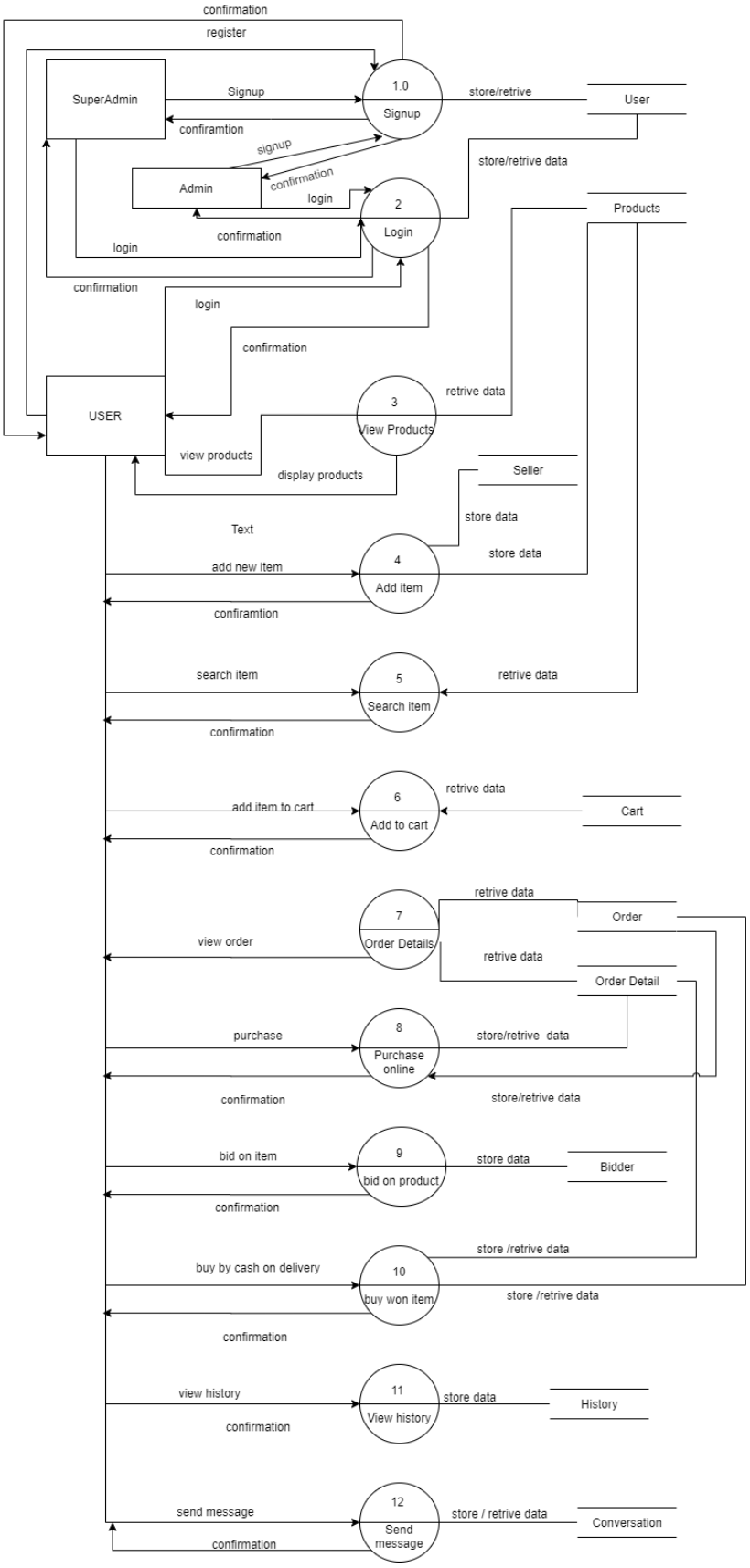


Figure 167:DFD level 1

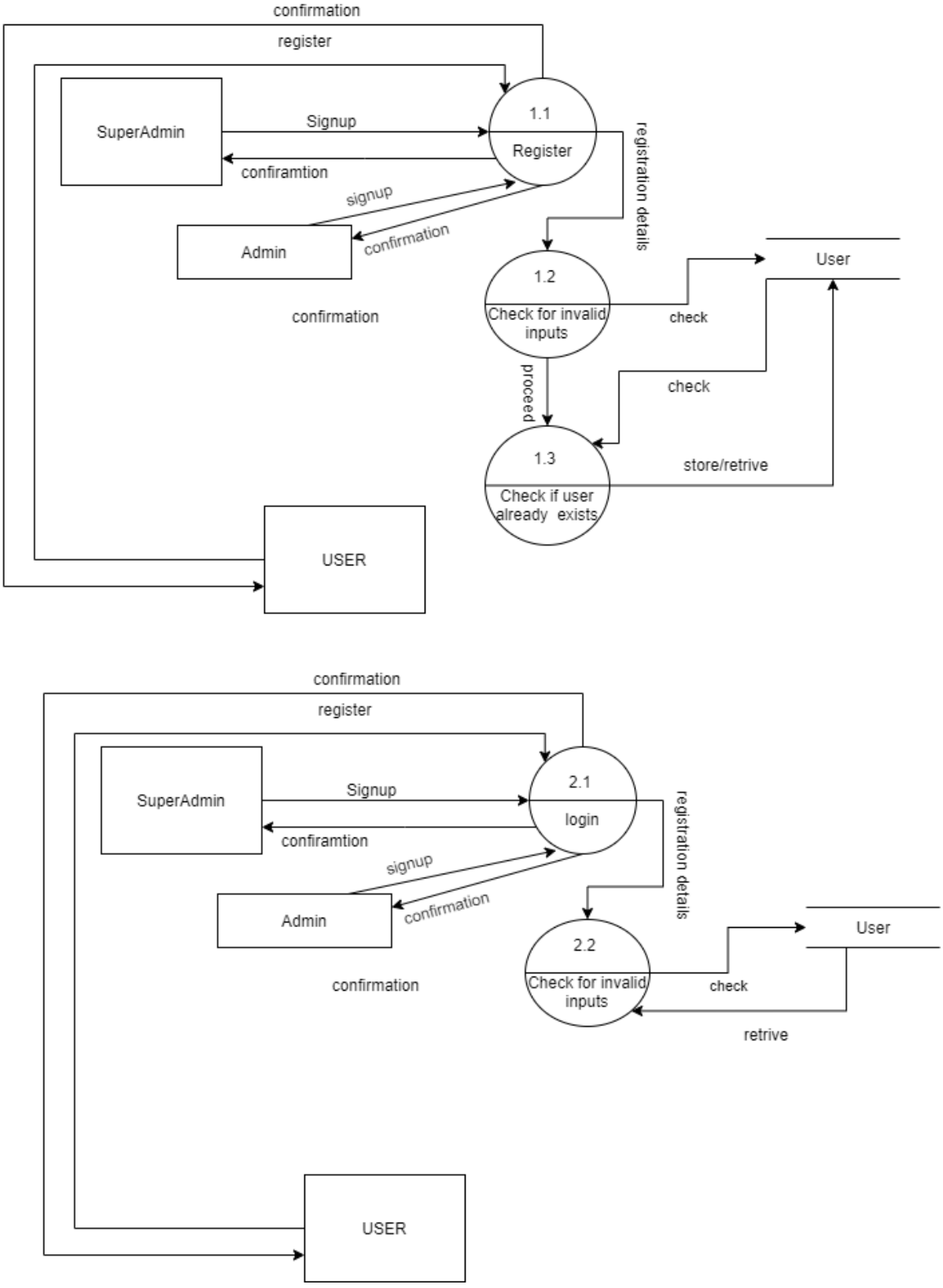


Figure 168: Dfd level1 of function 1 and 2

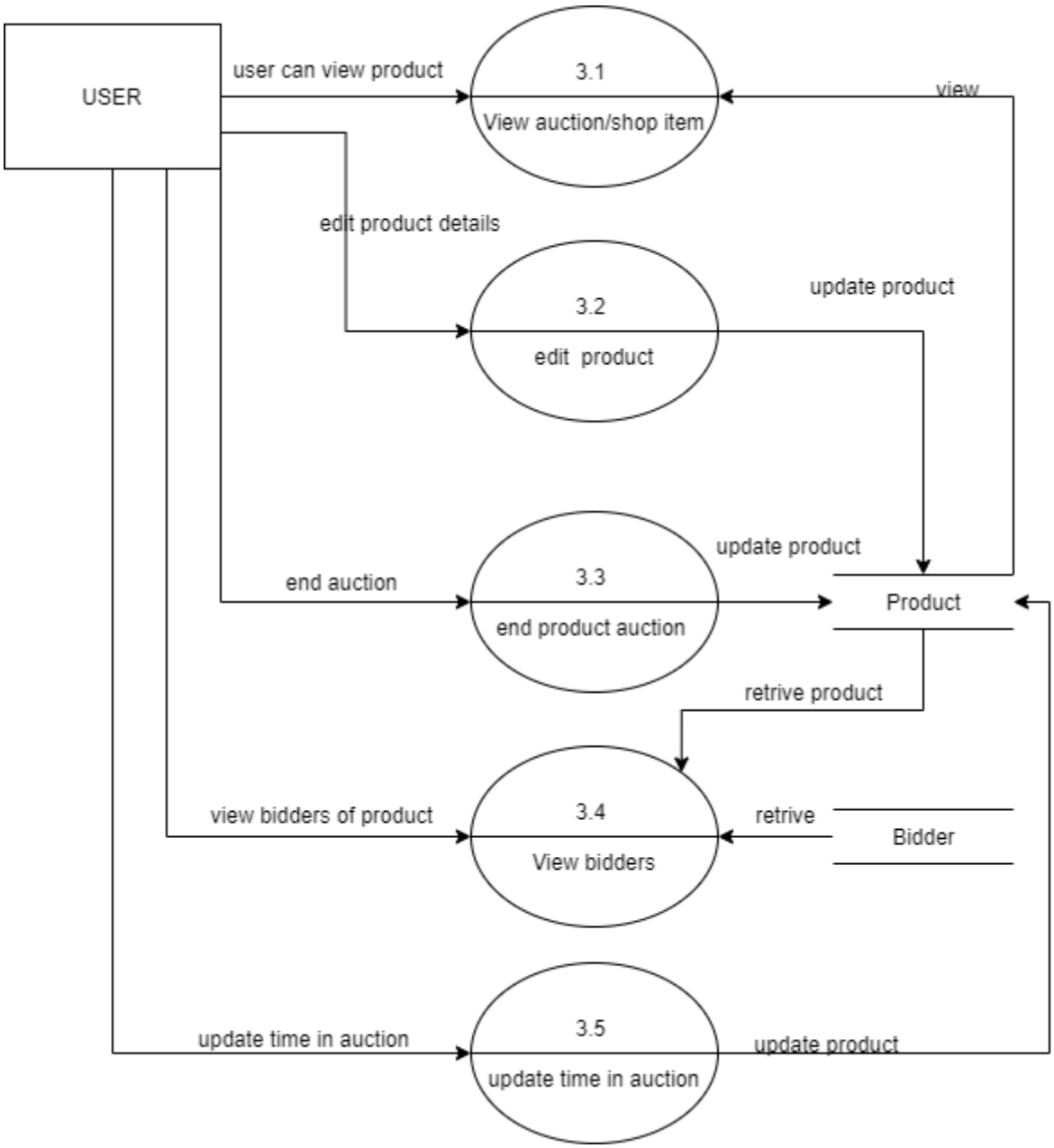


Figure 169:Dfd 3 level 2

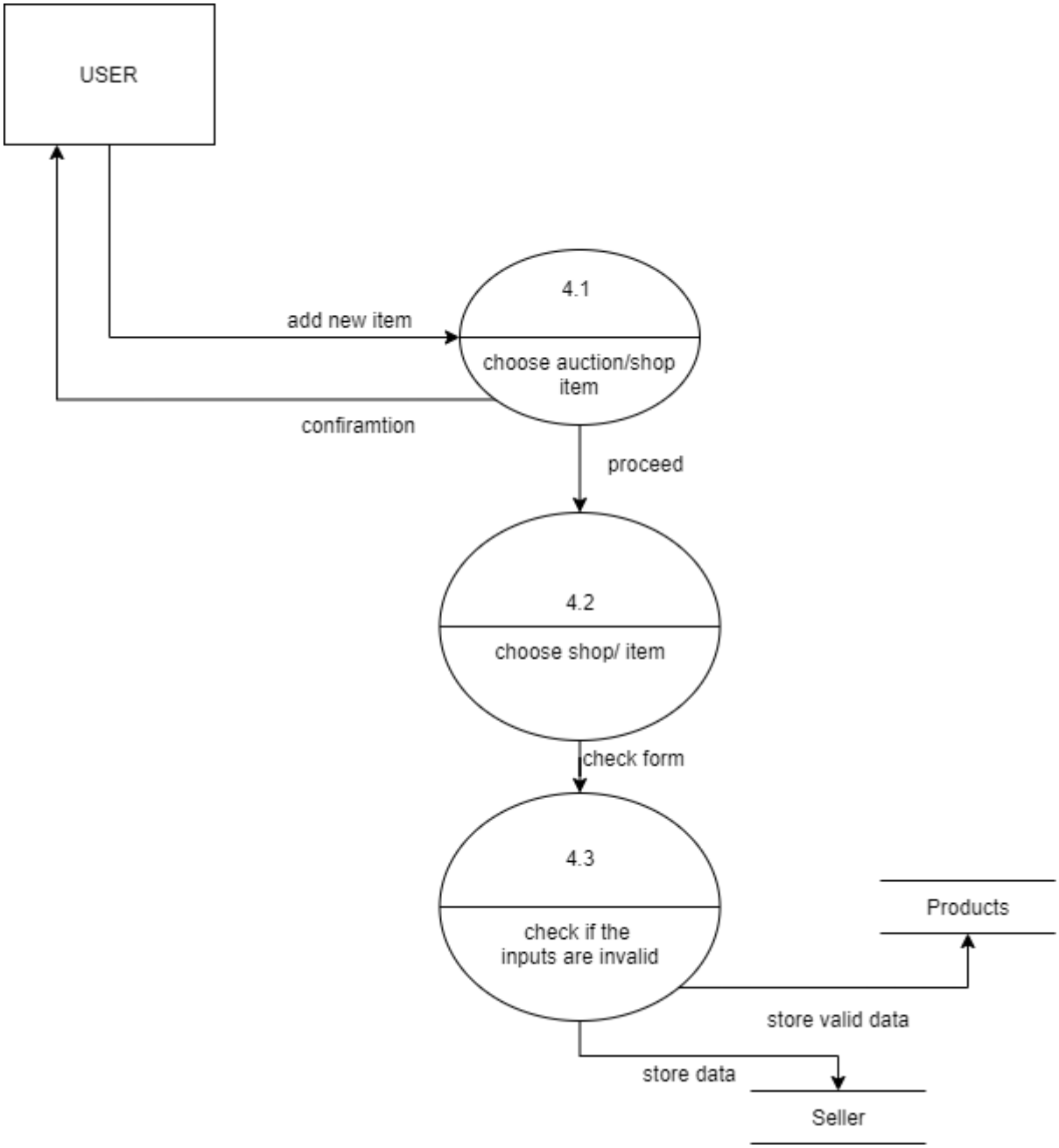


Figure 170: dfd diagram level2 of function 3

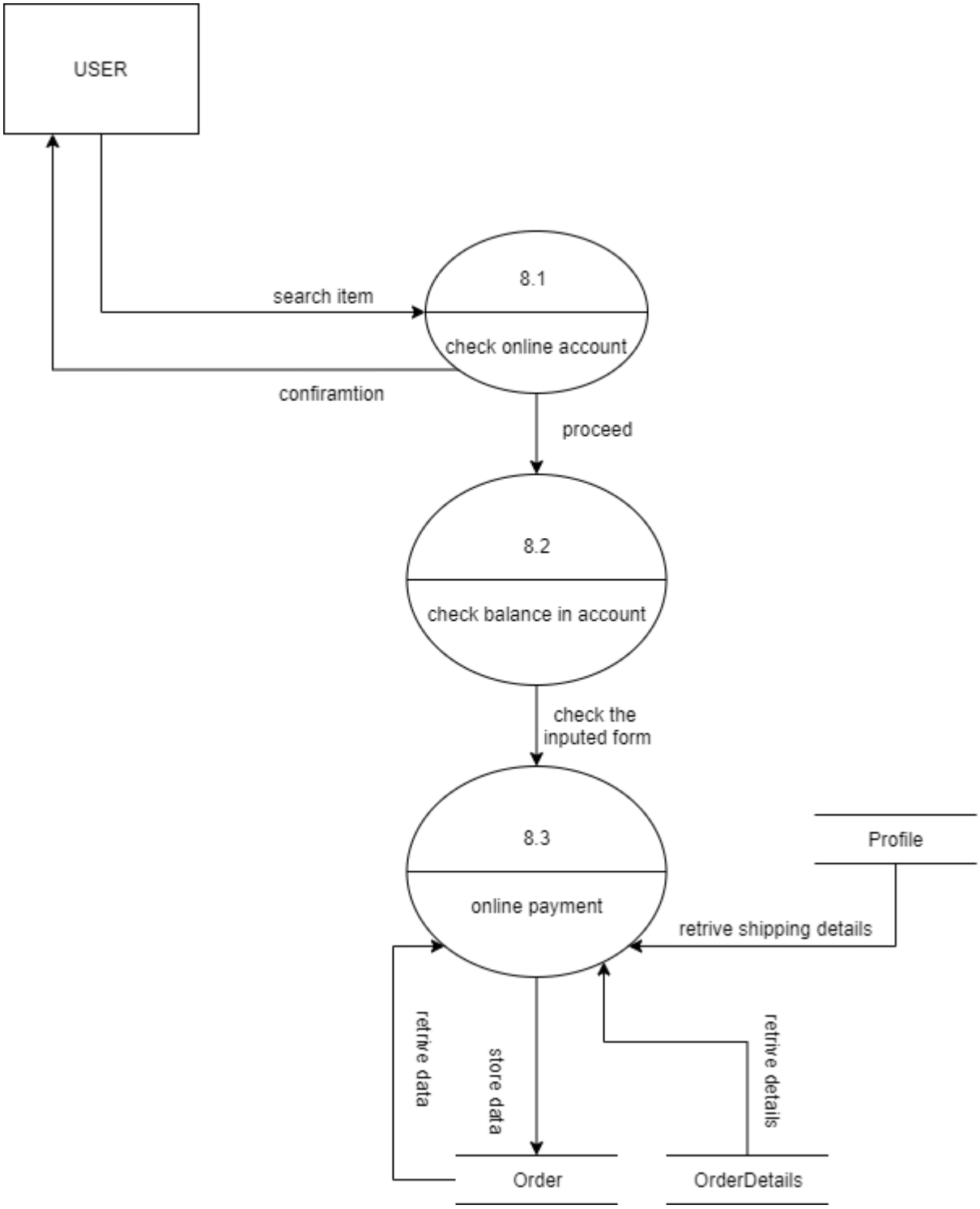


Figure 171: dfd diagram level2 of function8

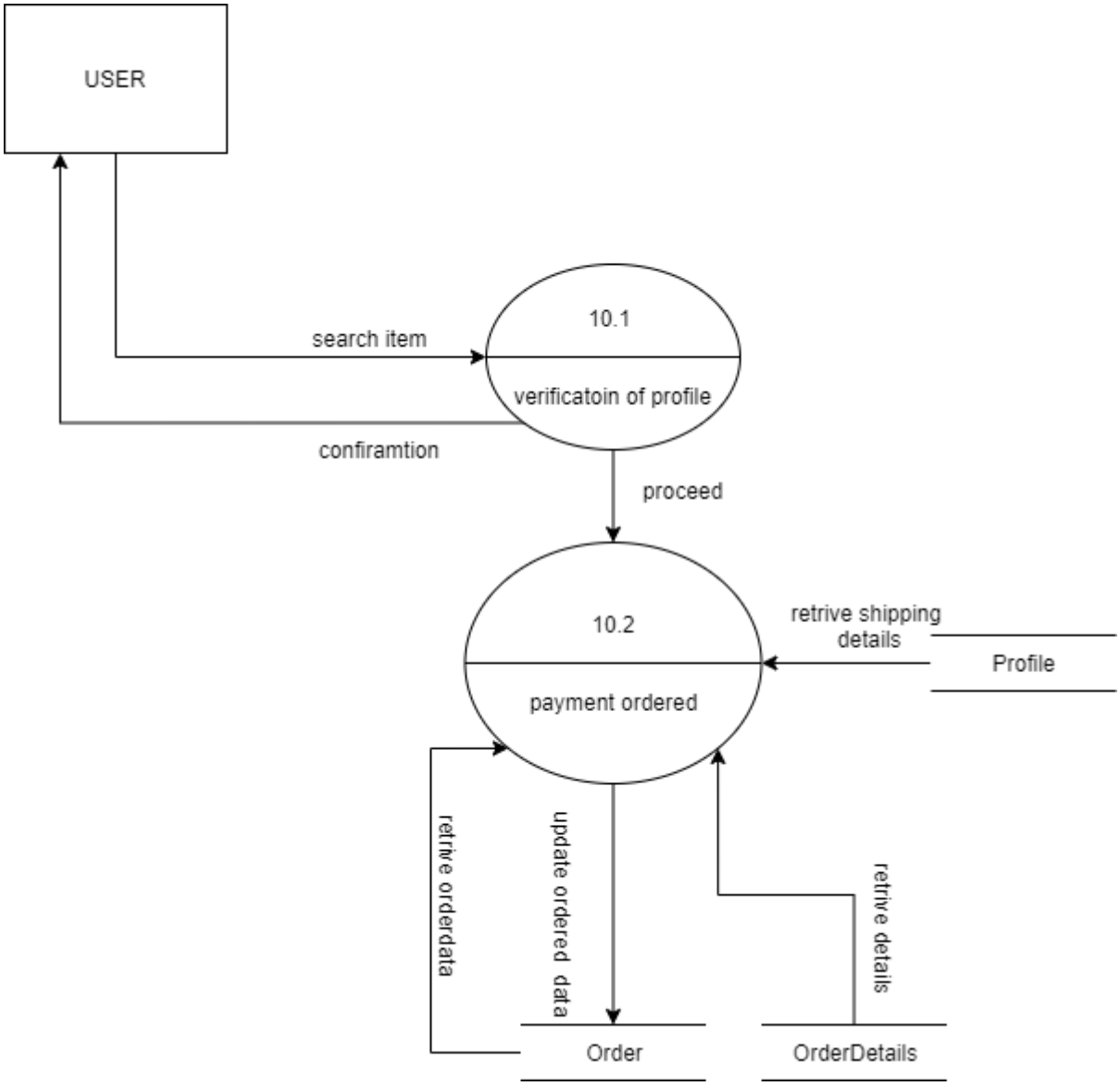


Figure 172: dfd diagram level2 of function8

8.4.7 USE CASE

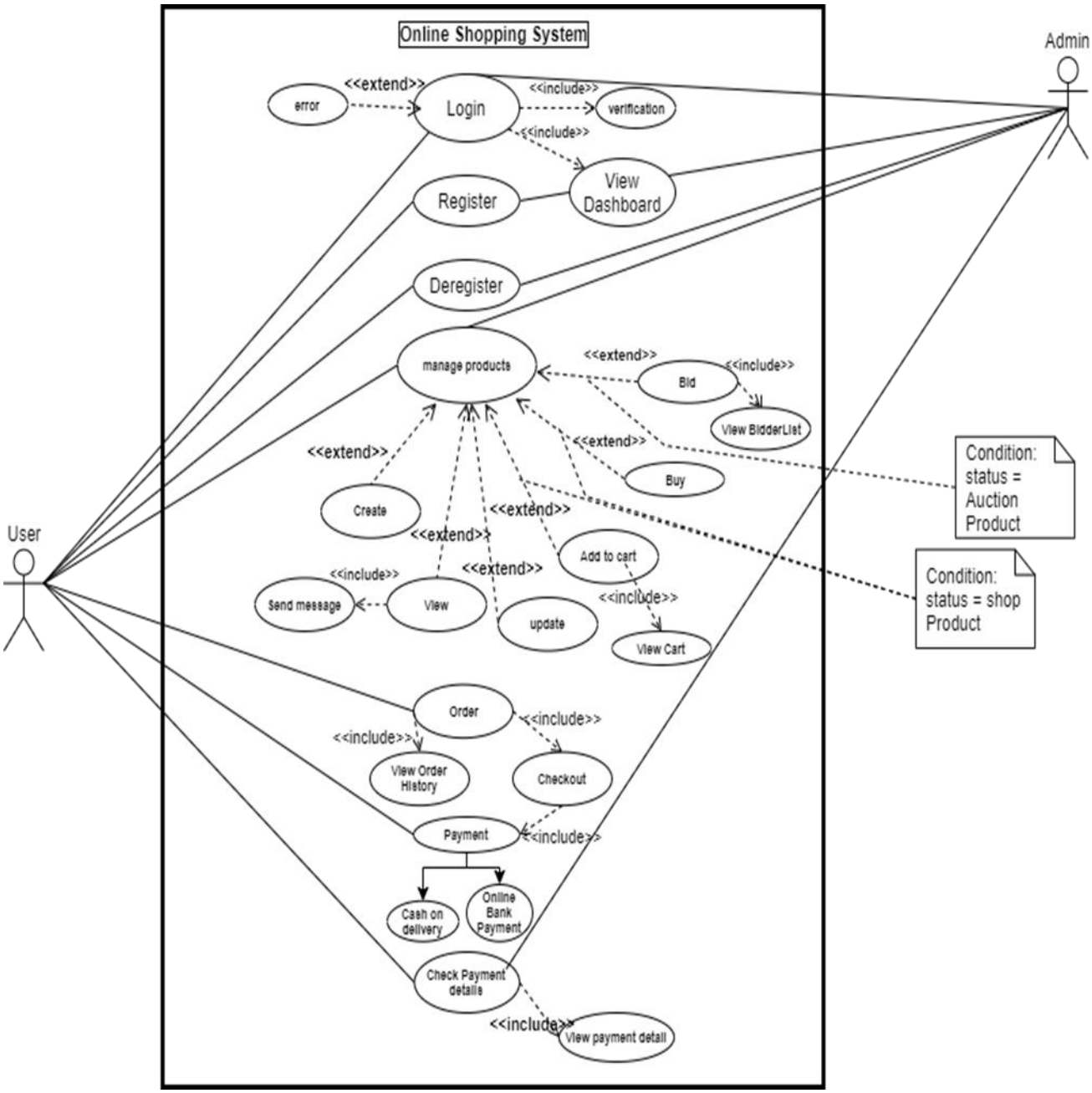


Figure 173:Use Case Diagram

8.4.8 WIREFRAME

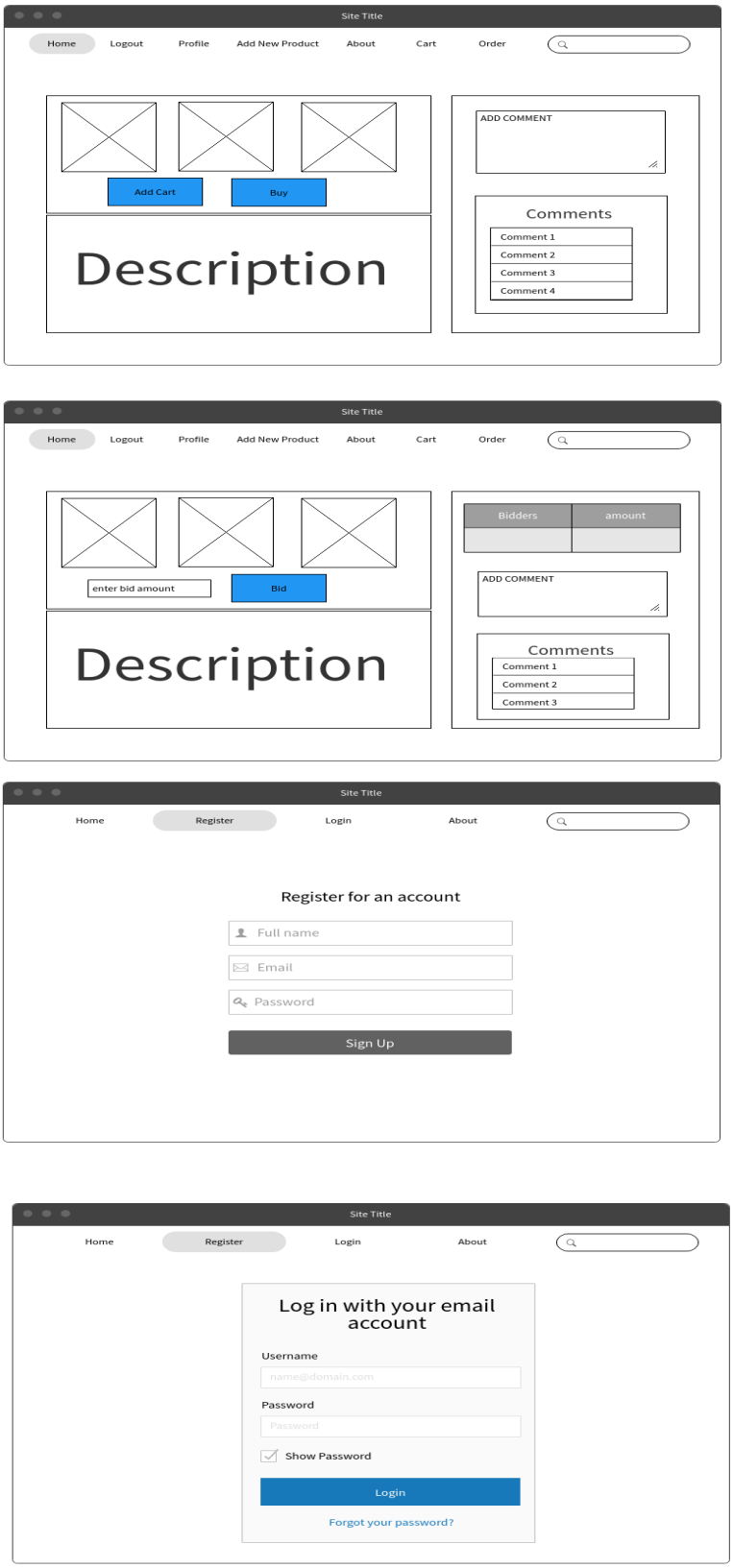


Figure 174:Wireframe part 1

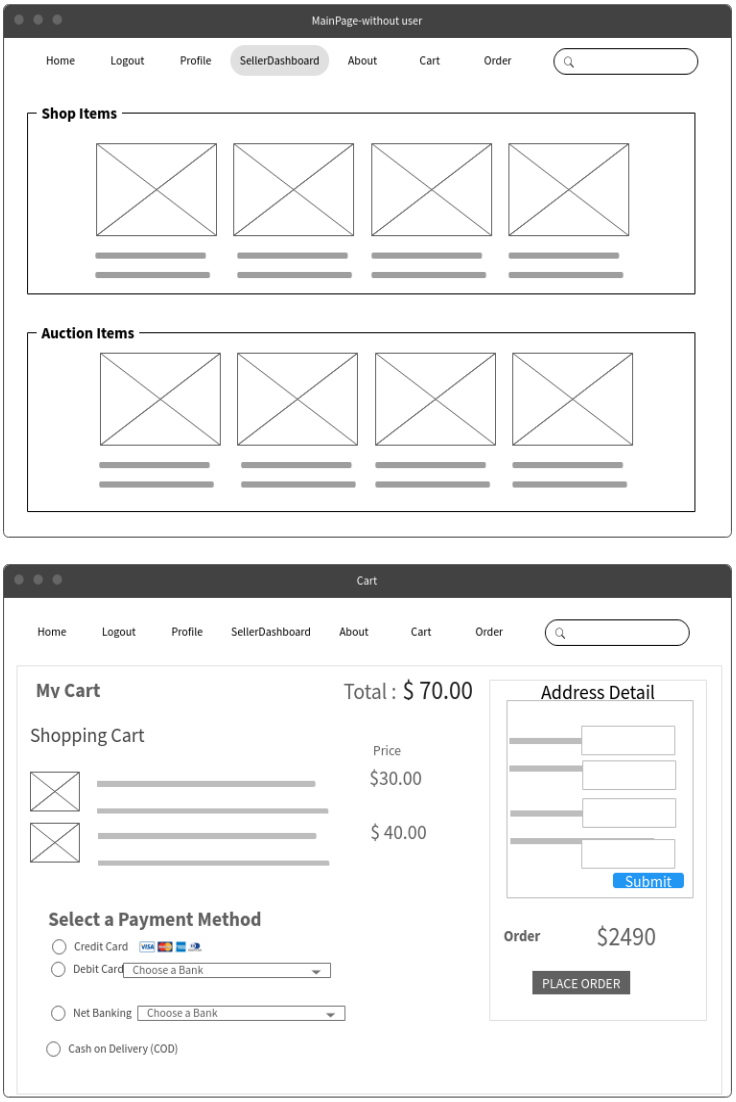


Figure 175: Wireframe part 2

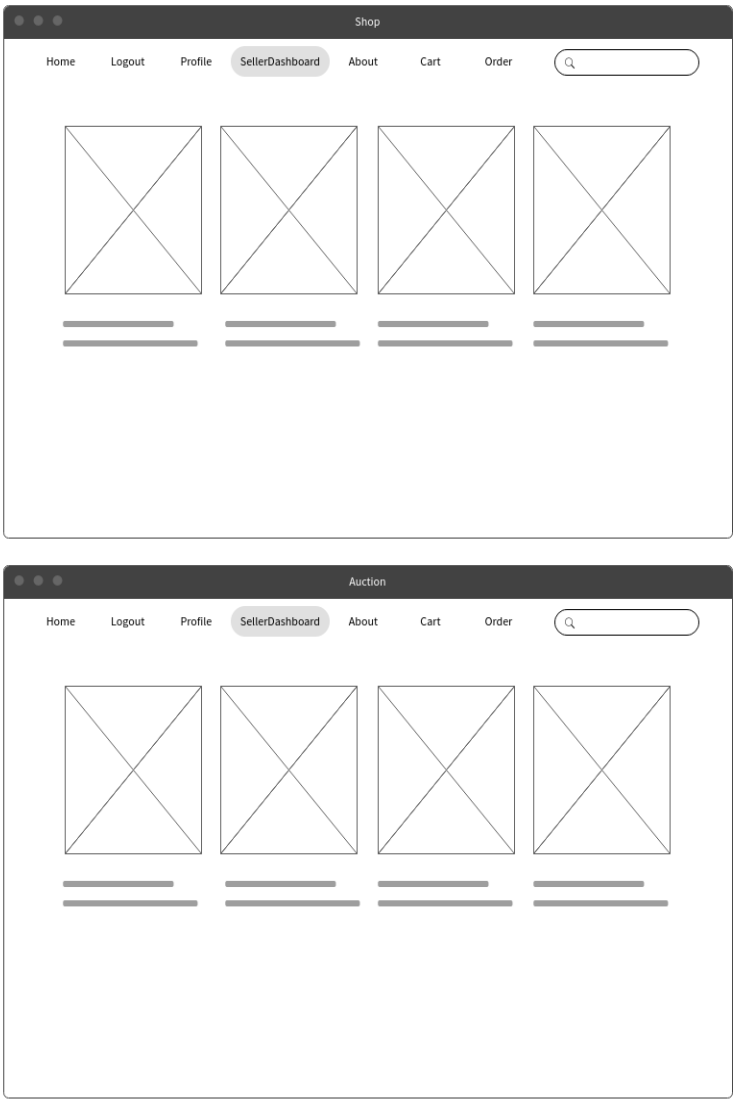


Figure 176: Wireframe part 3

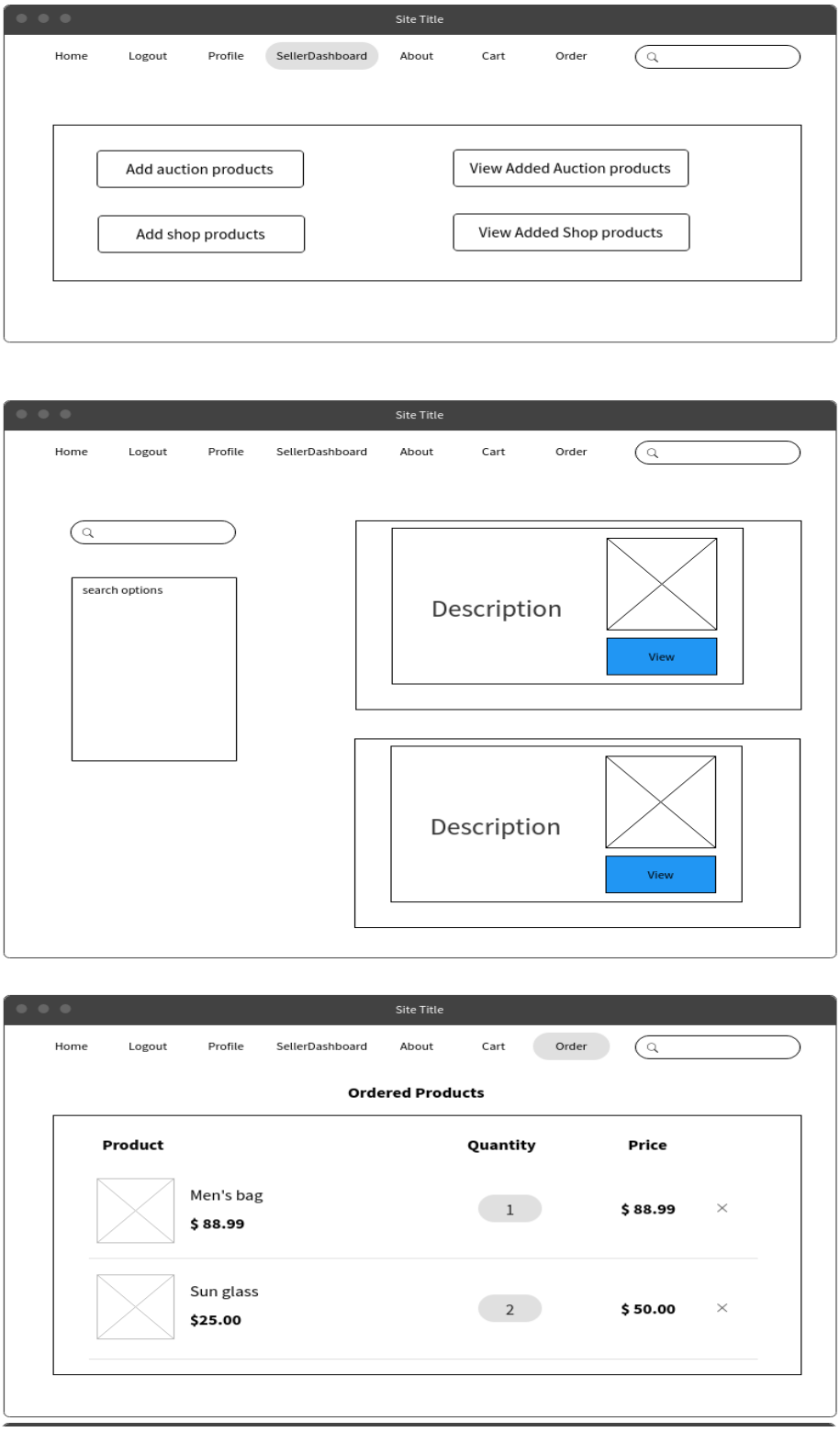


Figure 177: Wireframe part 4

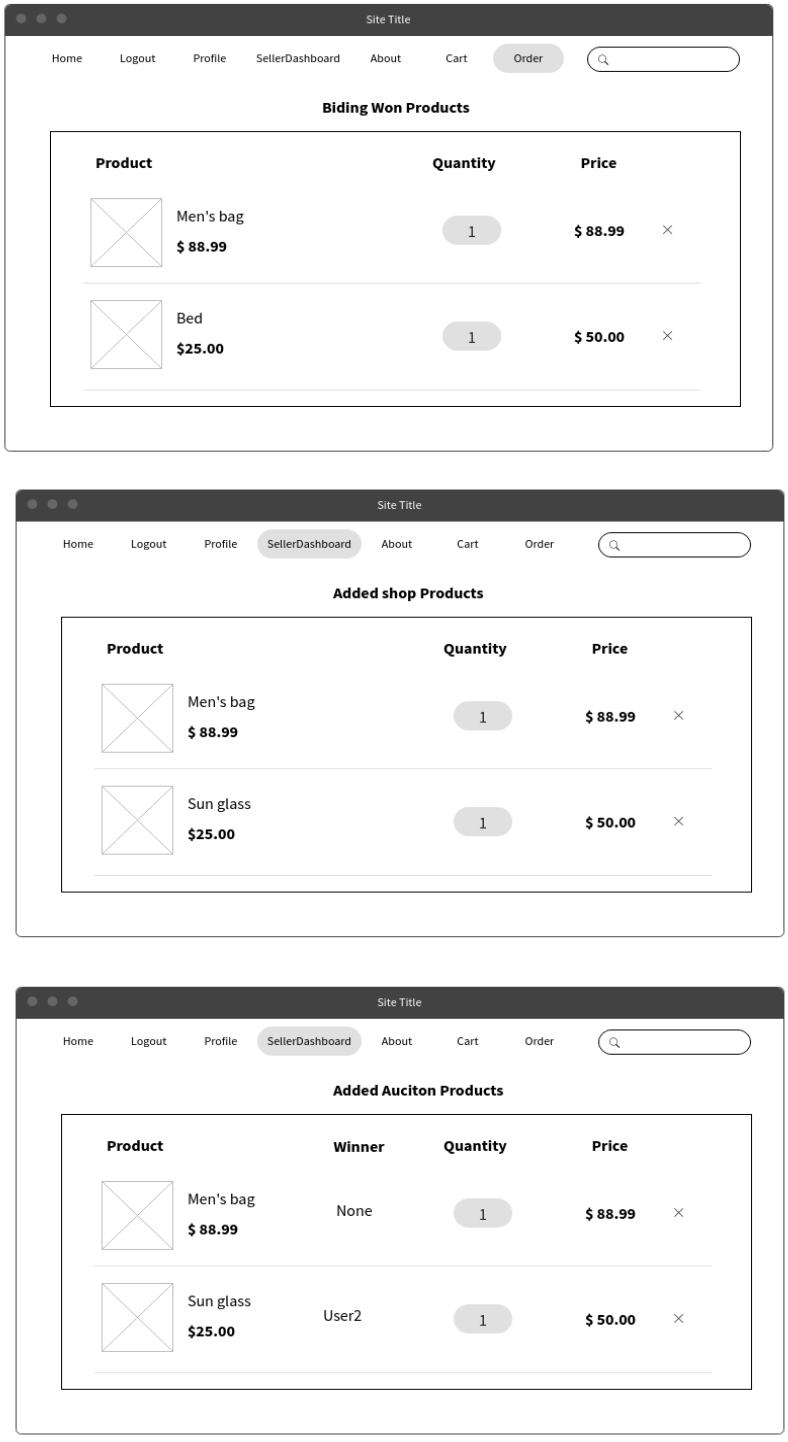


Figure 178: Wireframe part5

8.5 APPENDIX E: SCREENSHOTS OF THE SYSTEM

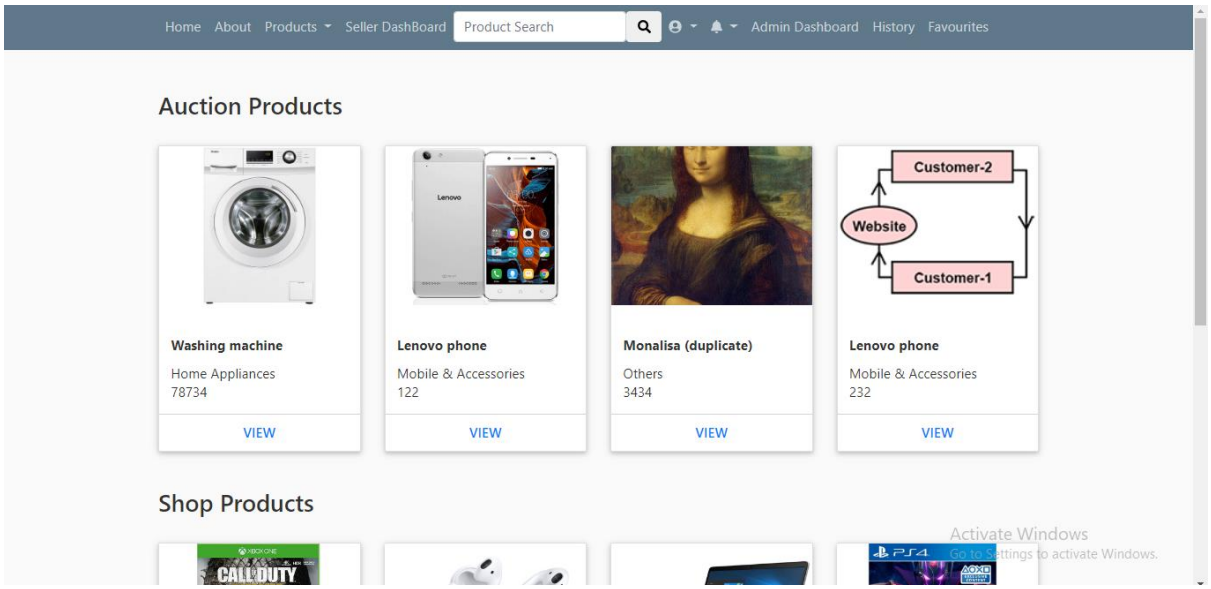


Figure 179: home page

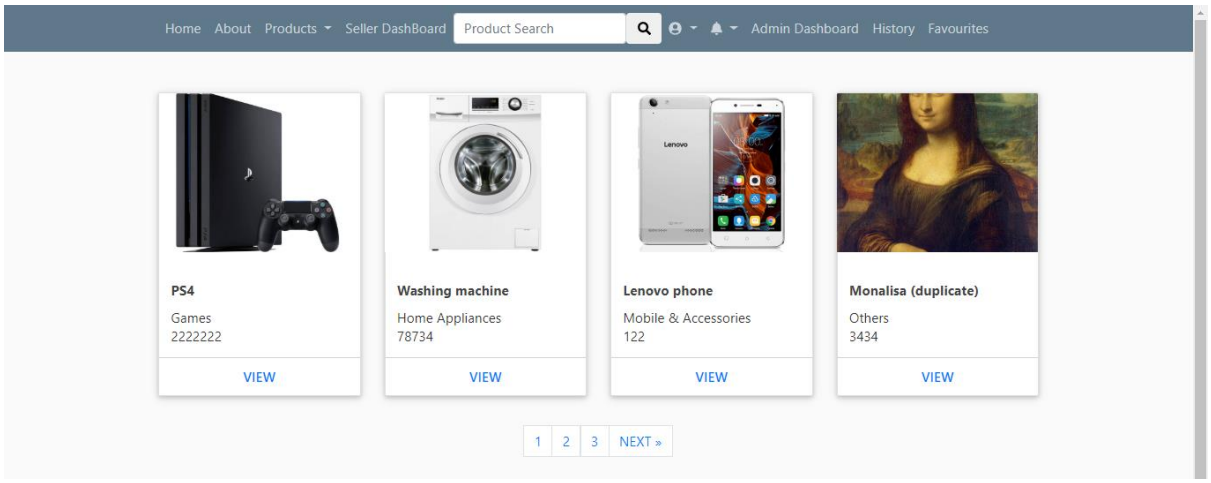


Figure 180:Auction page

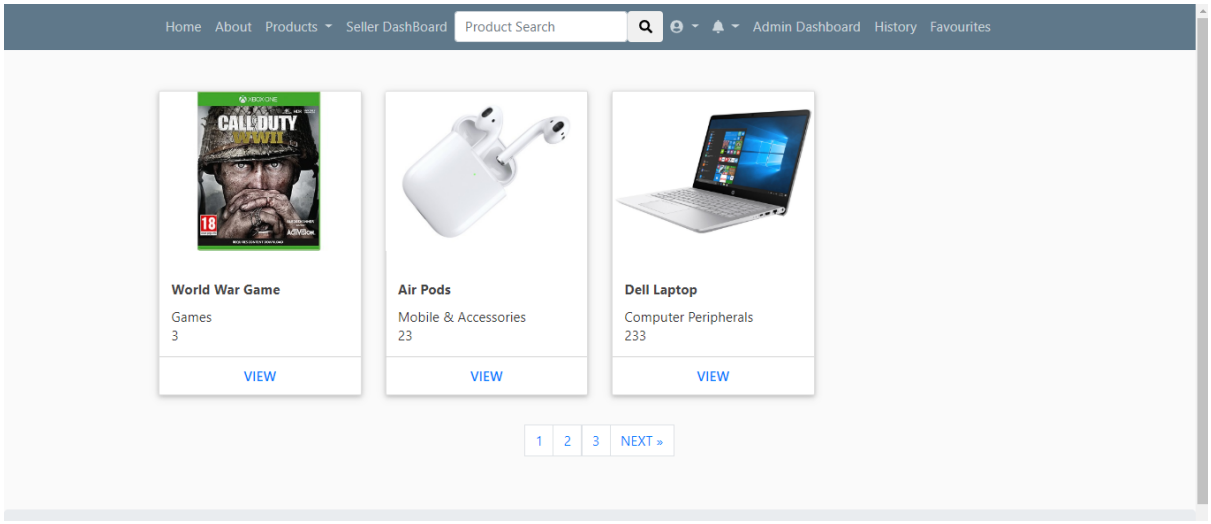


Figure 181: Shop page

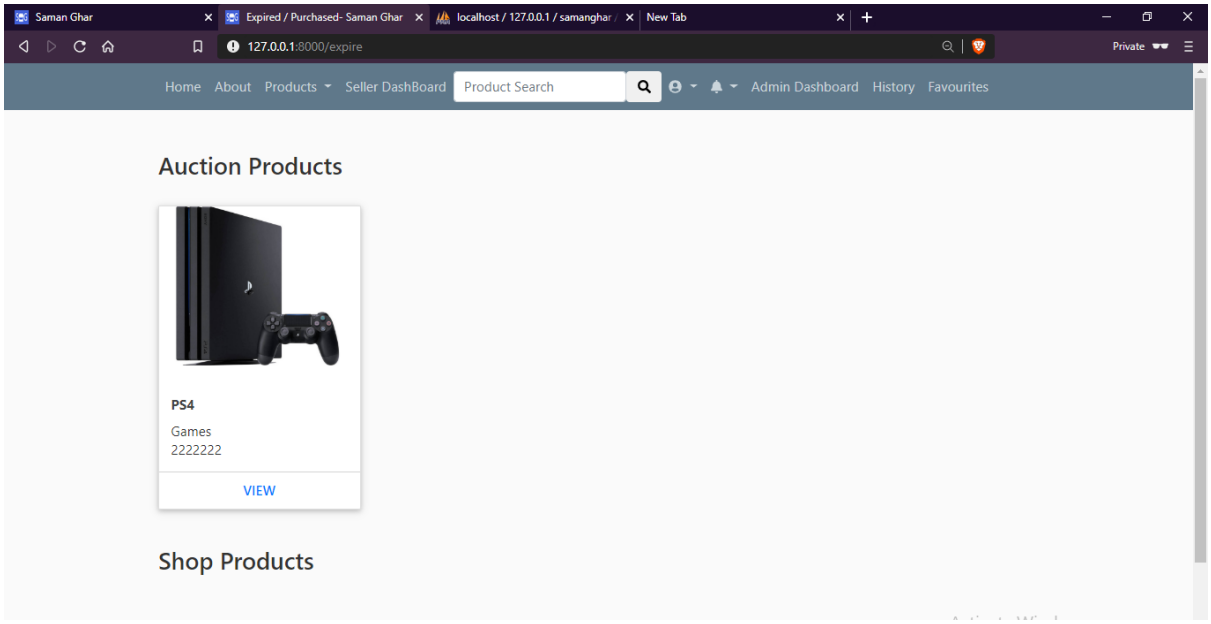


Figure 182: Expired/ purchased

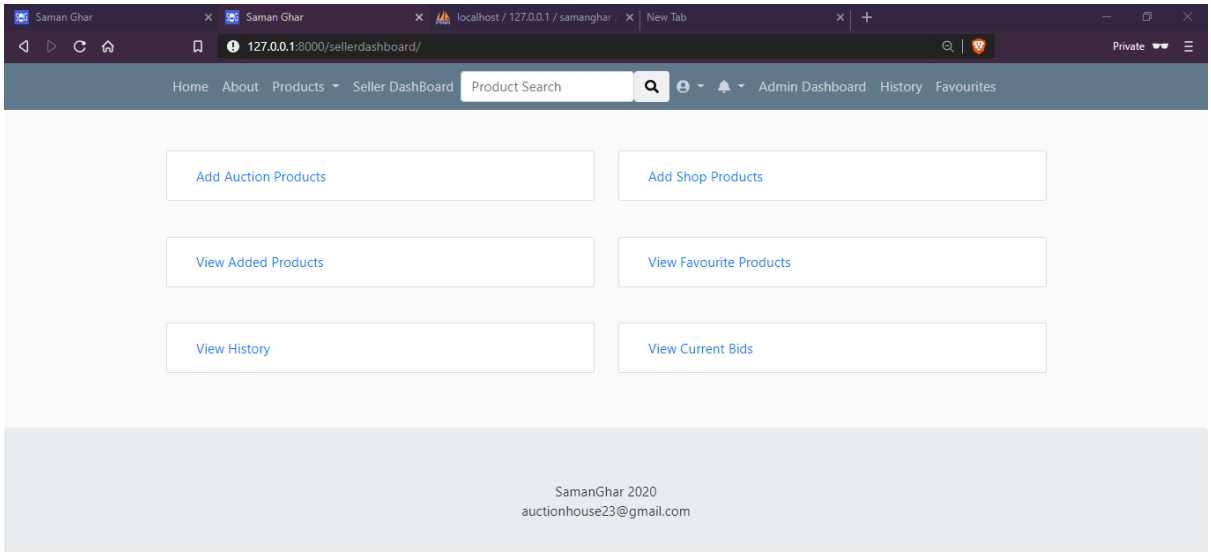


Figure 183: Seller dashboard

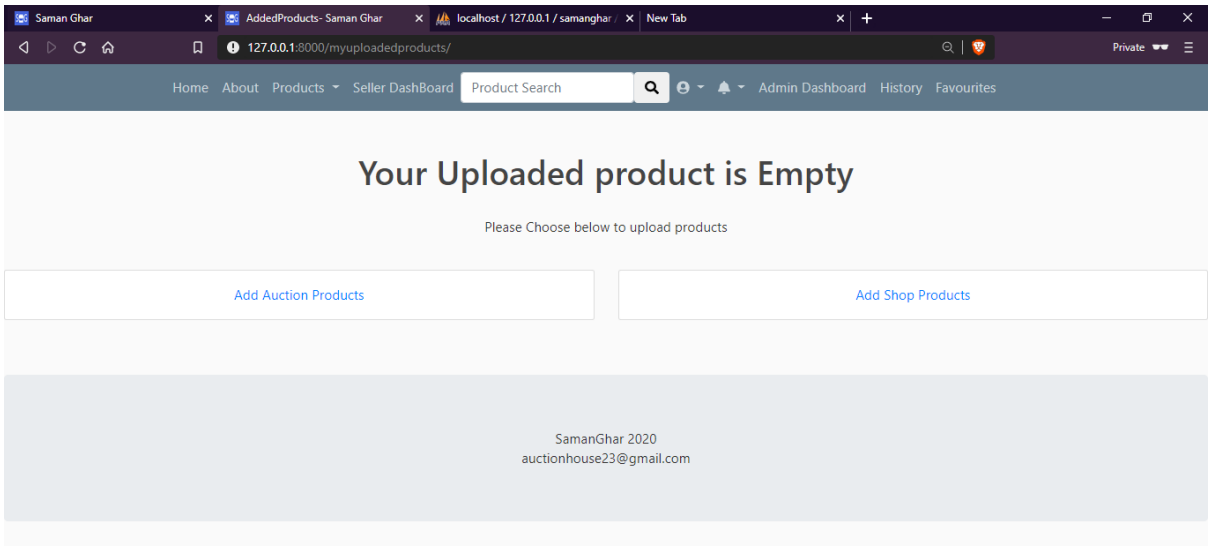


Figure 184:MY uploaded product

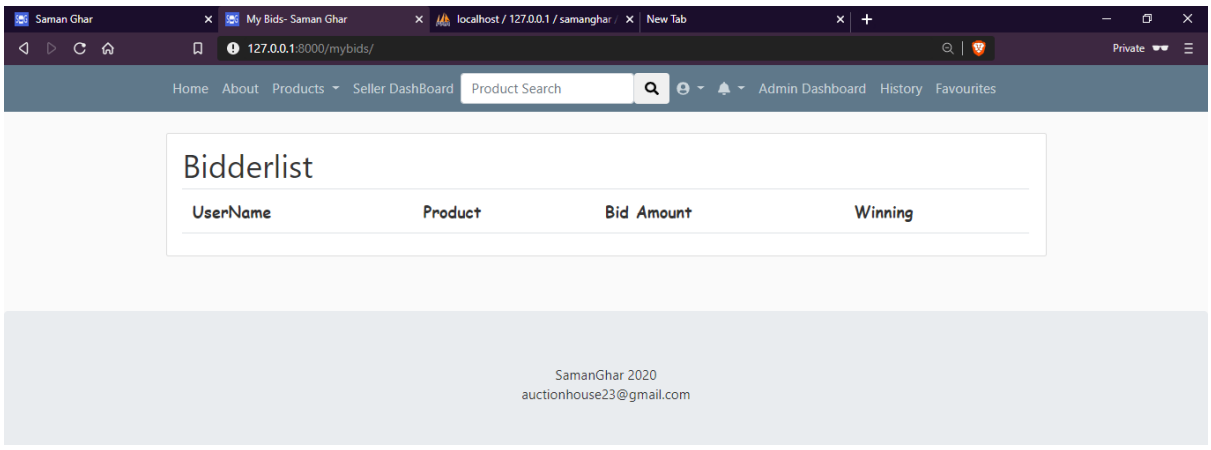


Figure 185: MY bids page

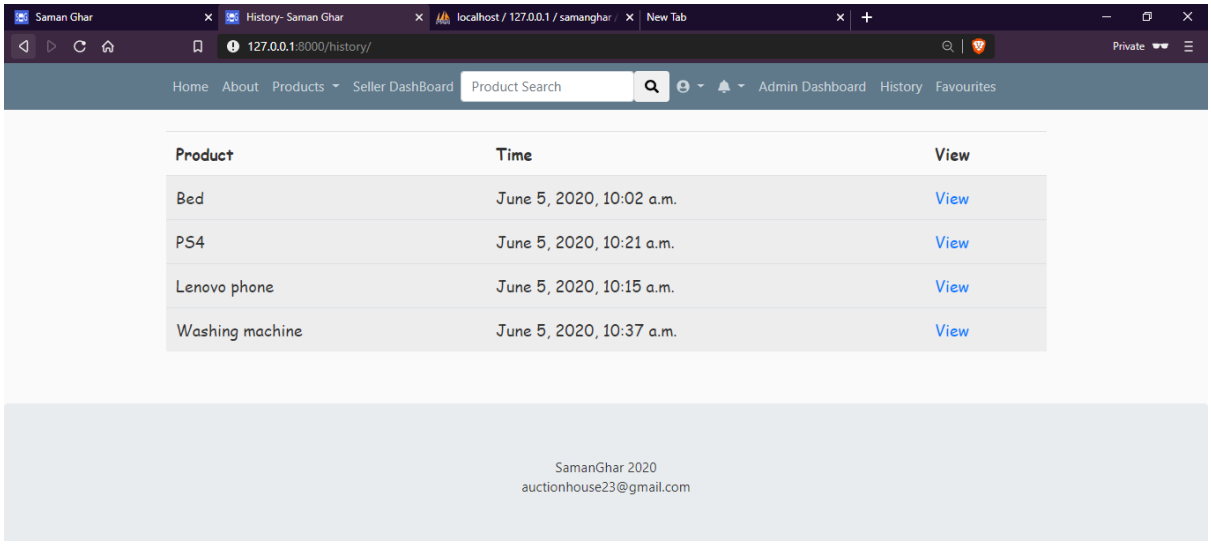


Figure 186:History page

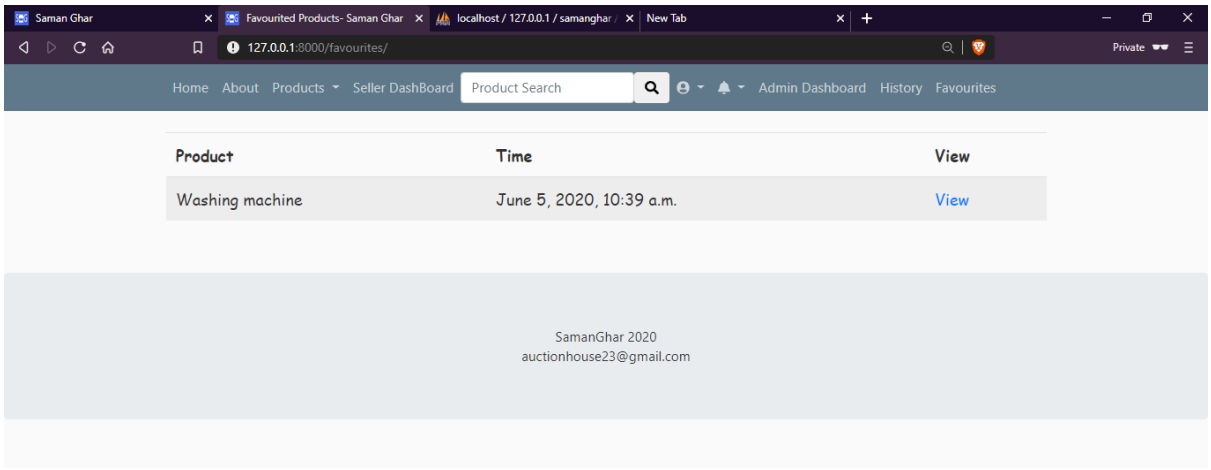


Figure 187: favourites

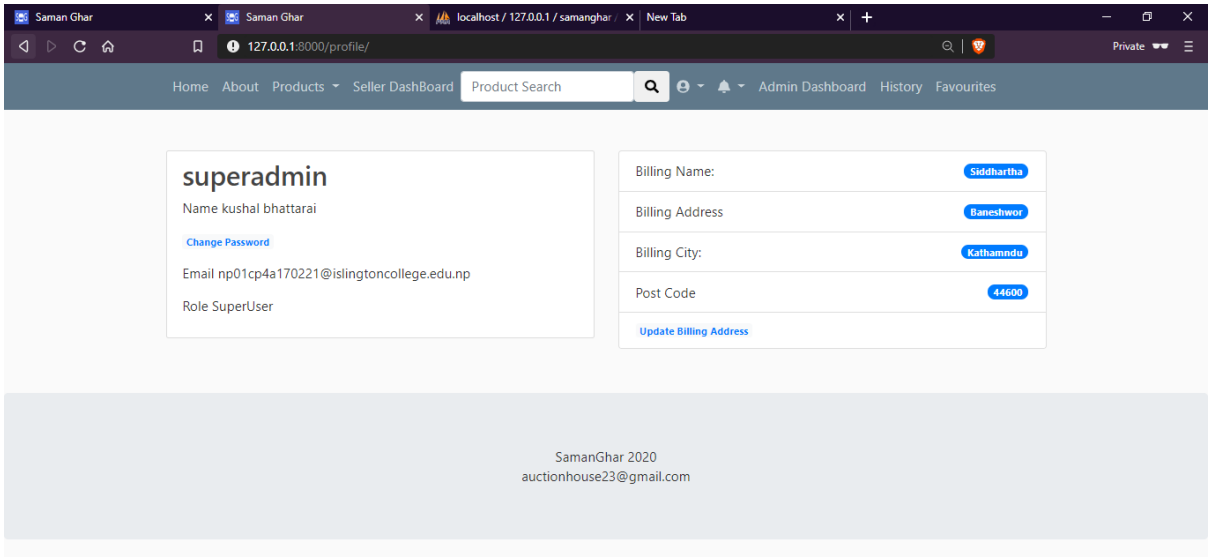


Figure 188: profile

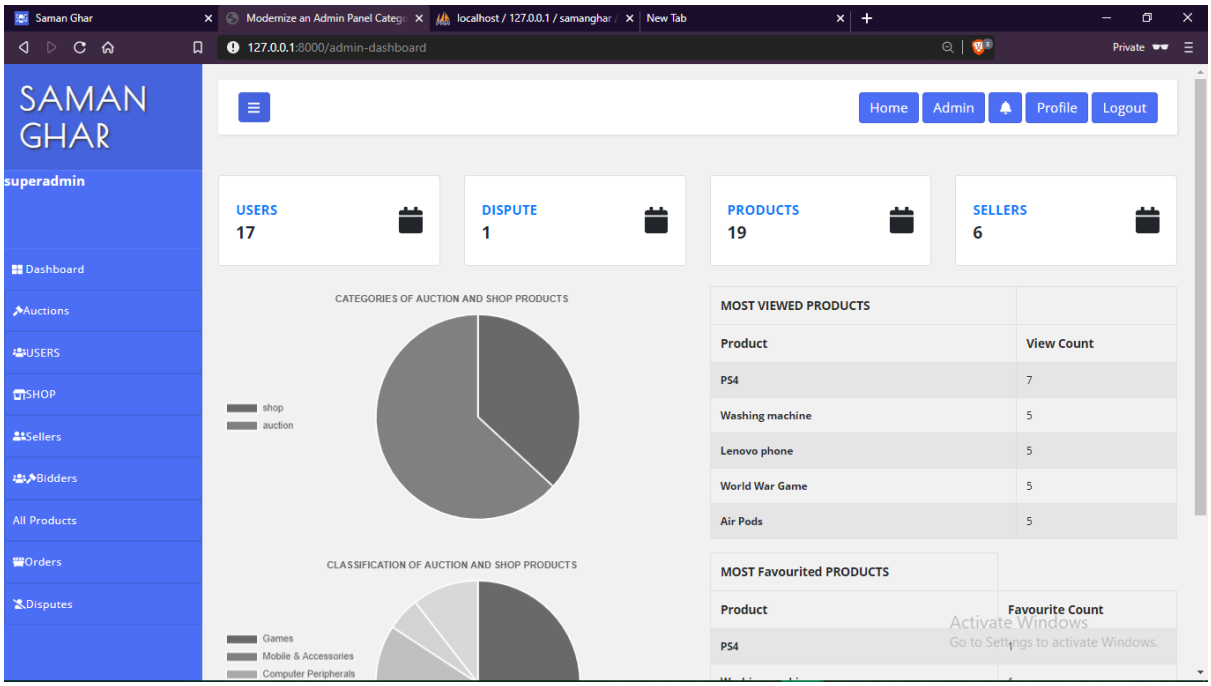


Figure 189:Admin dashoard

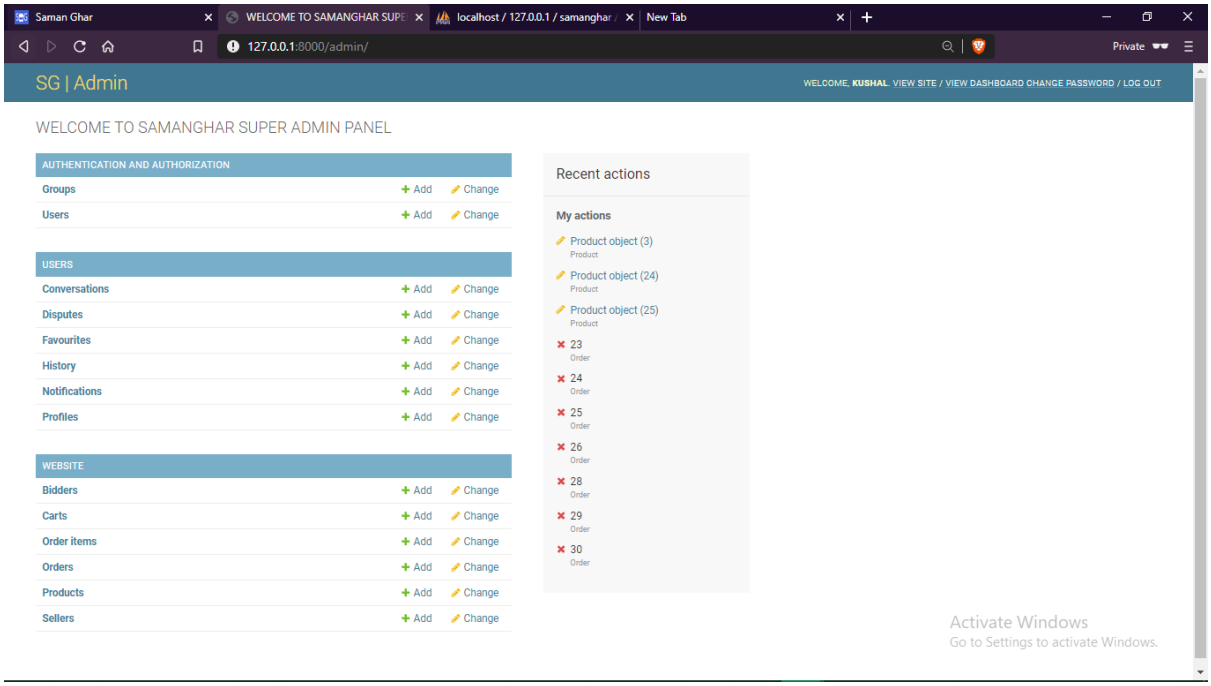


Figure 190: Super admin dash board

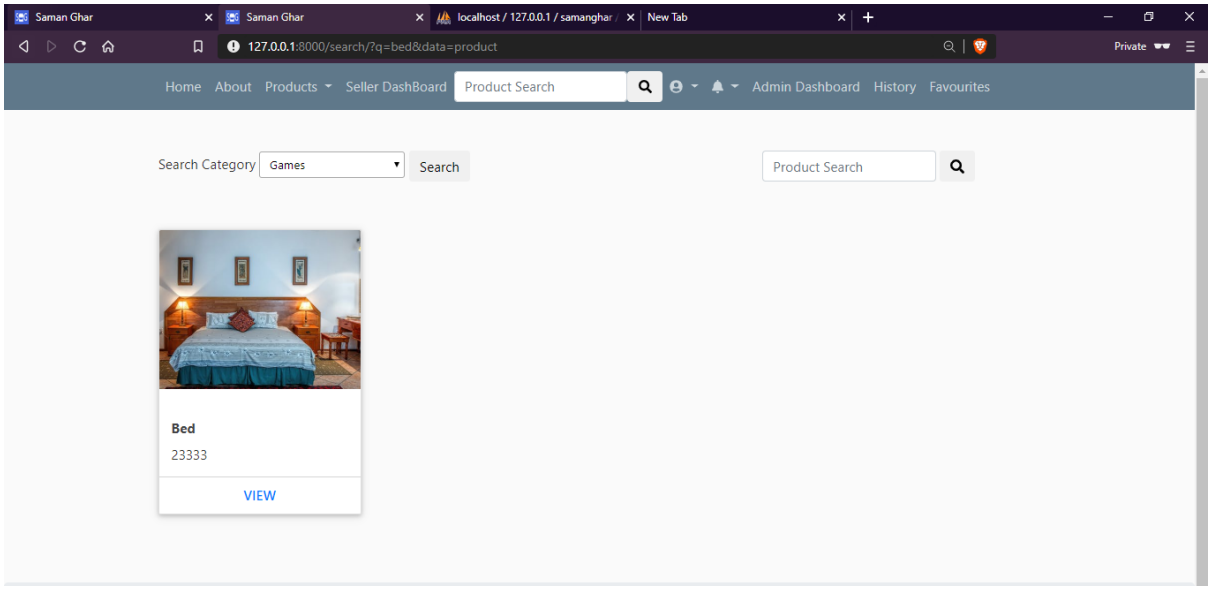


Figure 191:Search product

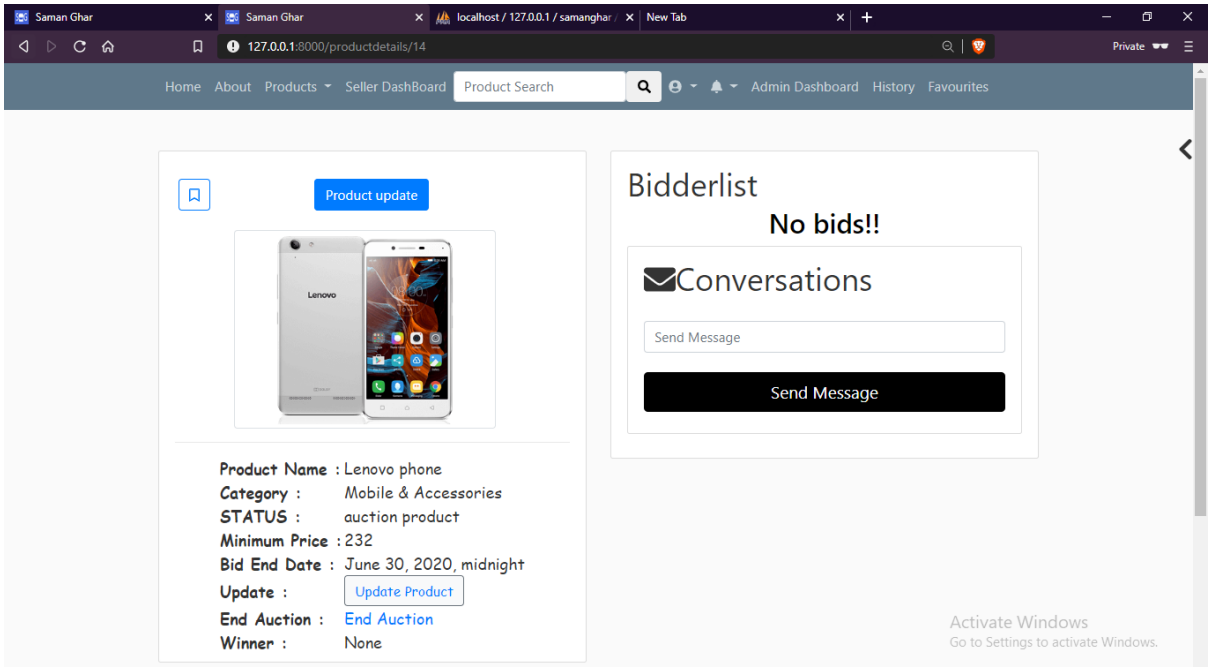


Figure 192:product deatils

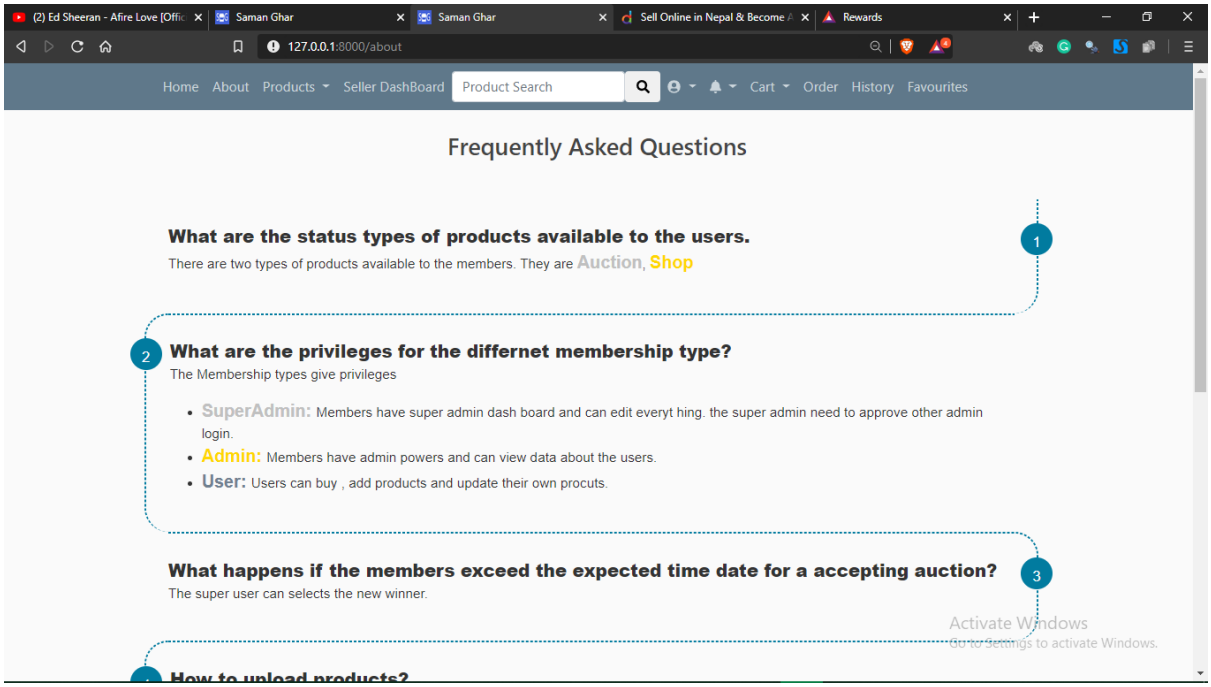


Figure 193: About page

8.6 APPENDIX F: USER FEEDBACK

8.6.1 USER FEEDBACK FORM

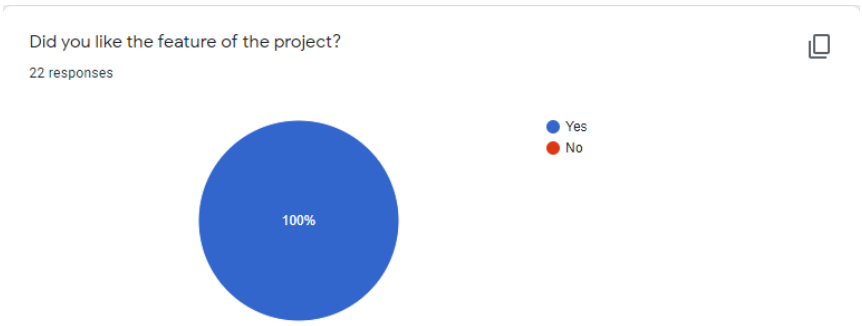


Figure 194: user feedback result -1

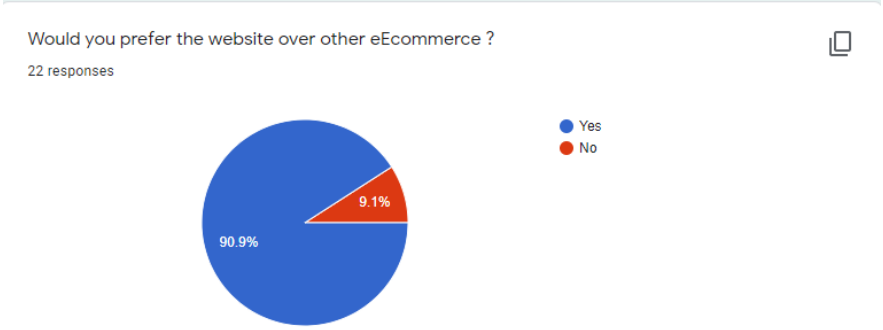


Figure 195: user feedback result -2

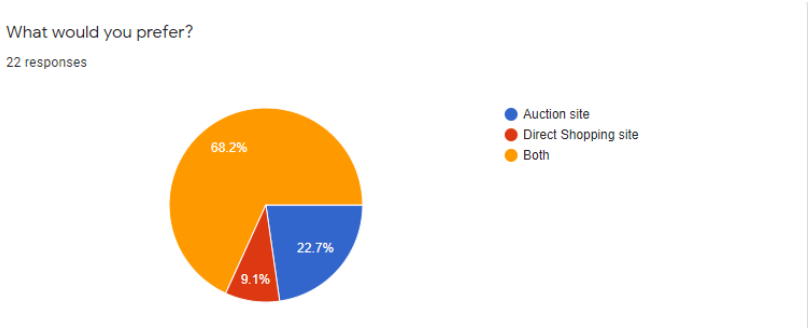


Figure 196: user feedback result -3

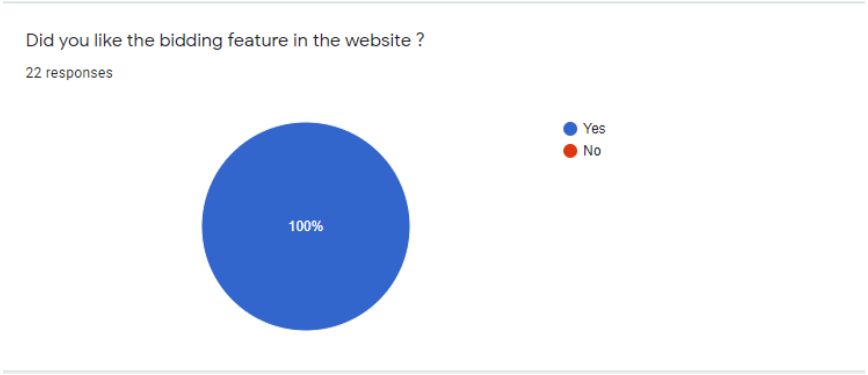


Figure 197: user feedback result - 4

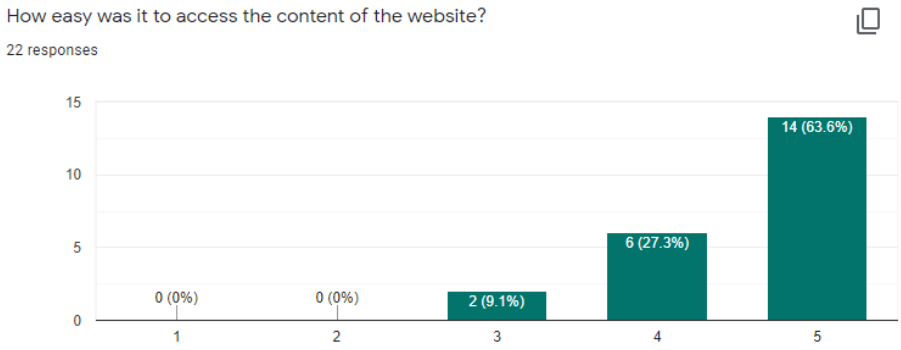


Figure 198: user feedback result -5

Figure 199: user feedback result 6

8.6.2 SAMPLE OF FILLED USER FEEDBACK FORMS

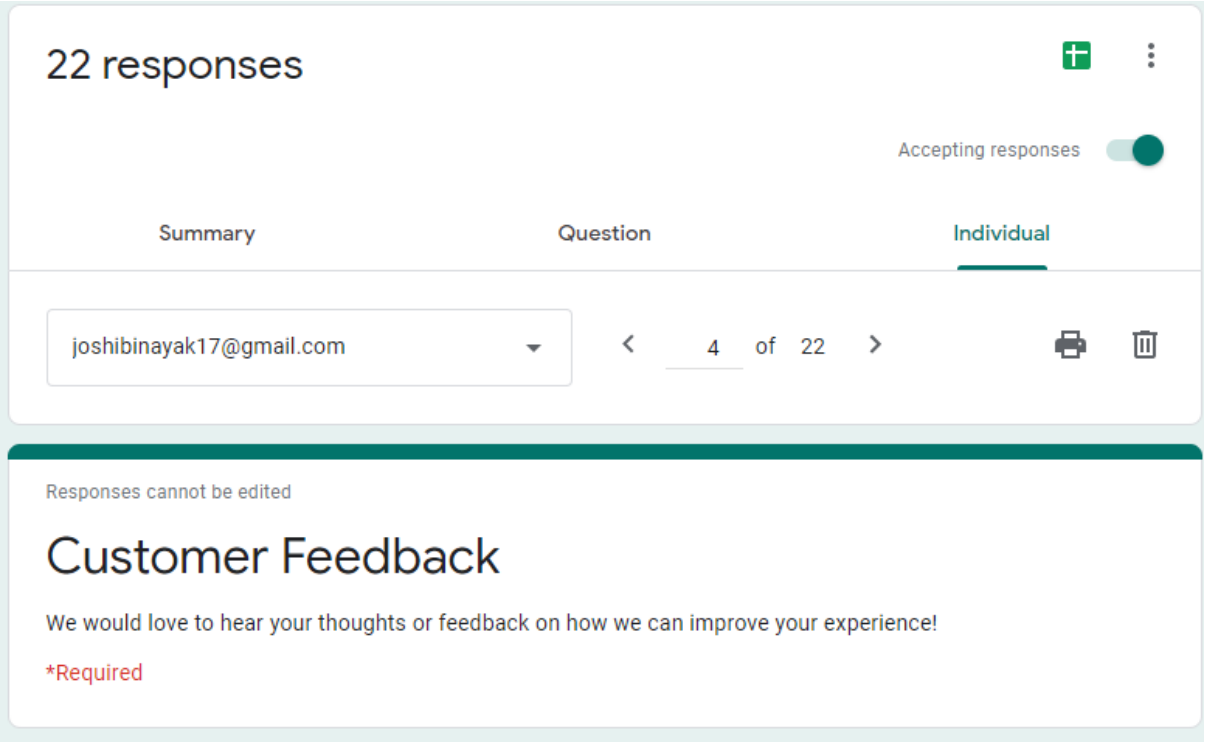


Figure 200: sample -1

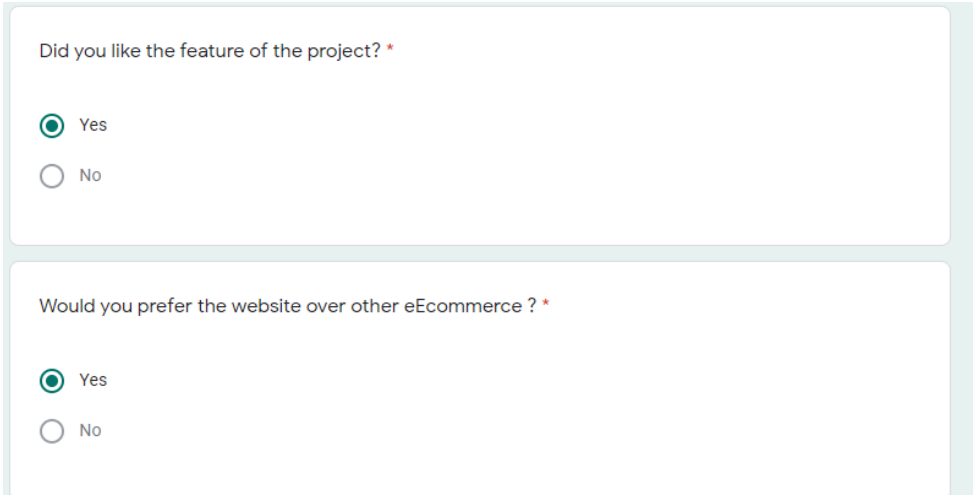


Figure 201: sample -2

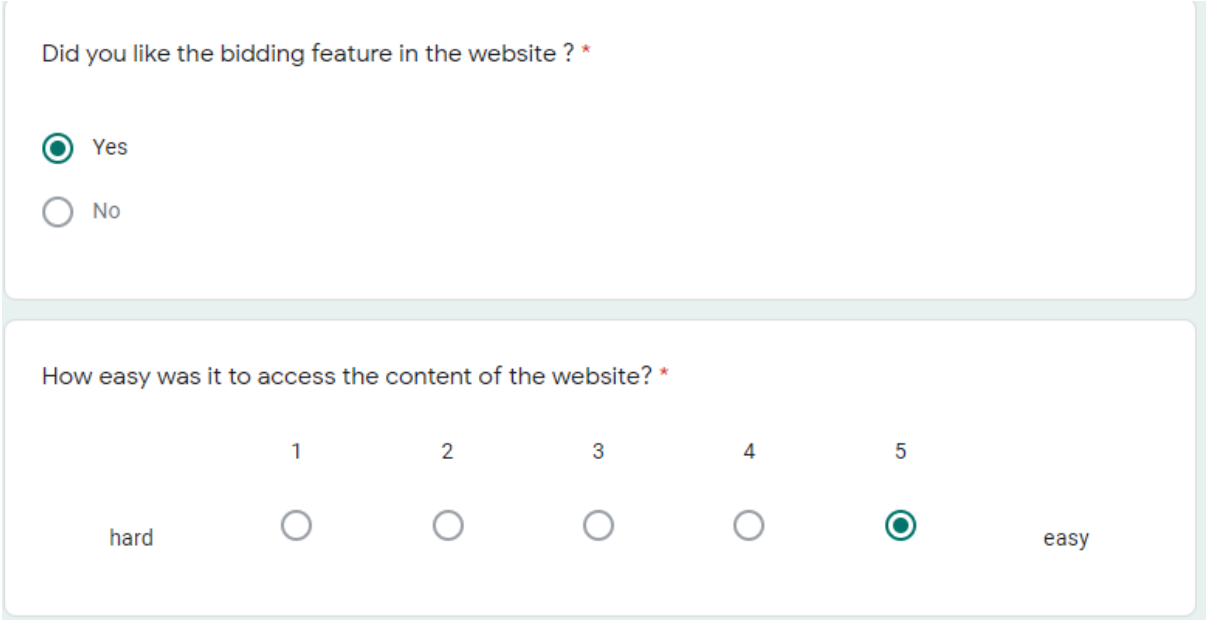


Figure 202: sample3

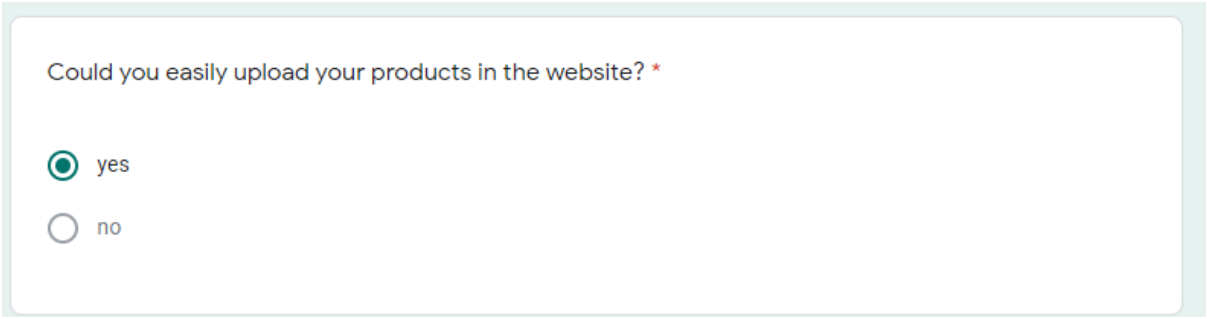


Figure 203: sample 4

8.7: APPENDIX G: FUTURE WORK

8.7.1 READINGS FOR FUTURE WORK

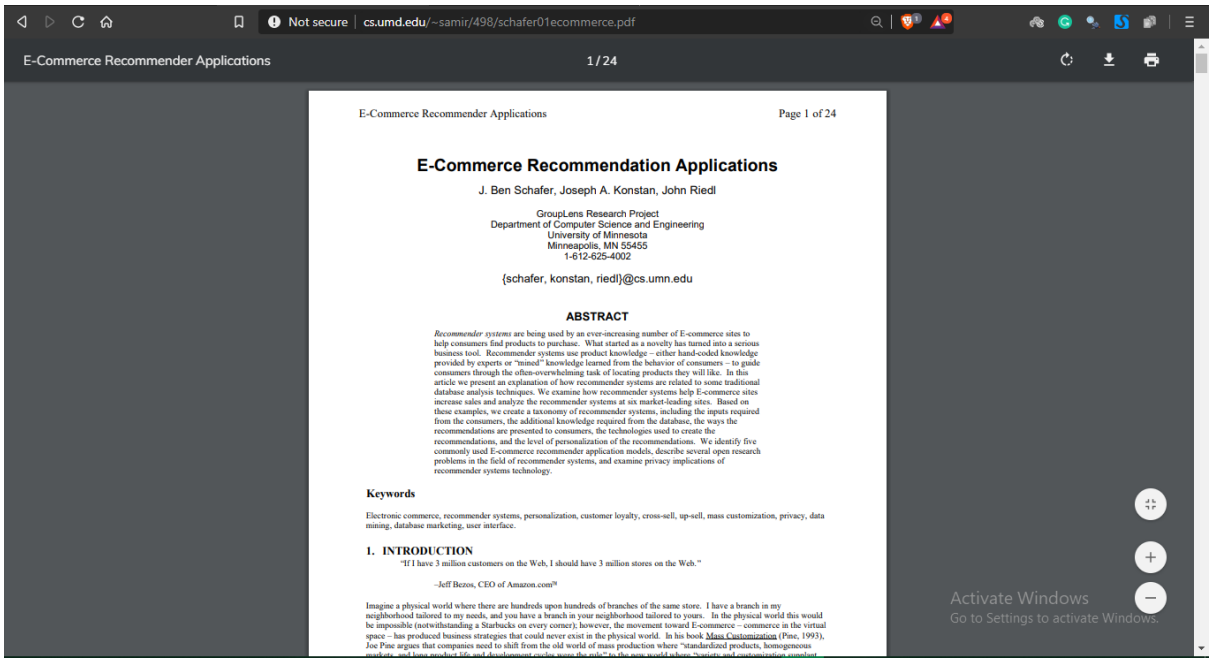


Figure 204: recommendation further work sample

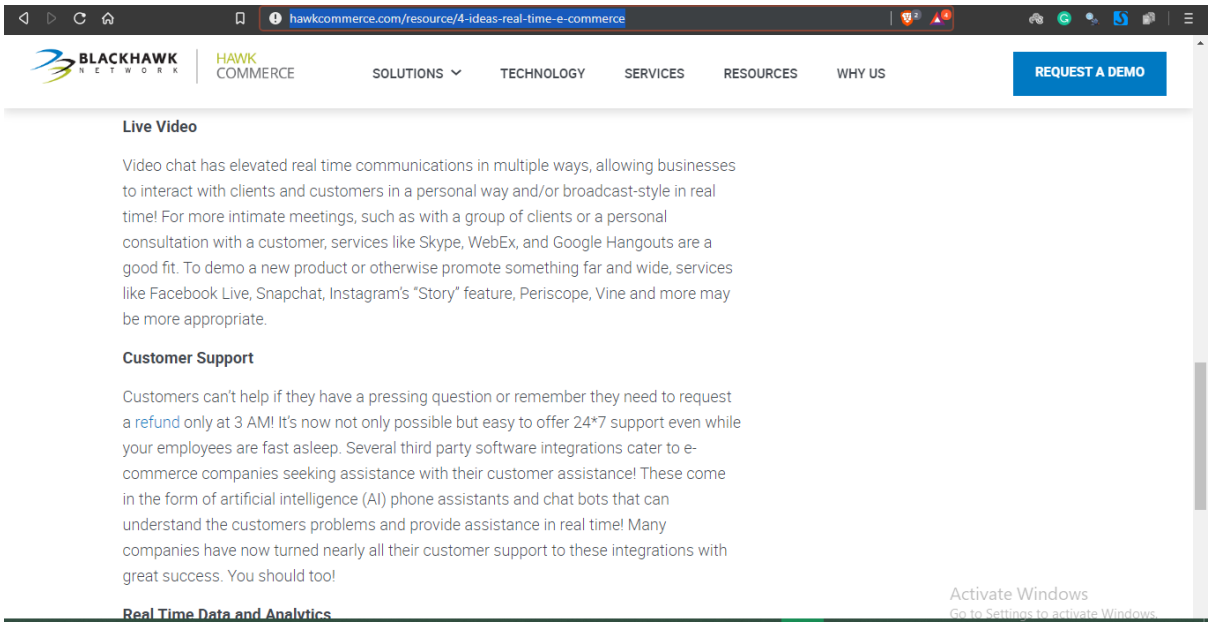


Figure 205:communication with buyer and supplier