



UNIVERSITY OF NAIROBI

FACULTY OF ENGINEERING DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

ONE-STOP DISCOUNT PLATFORM FOR STUDENT SUPPORT

PROJECT INDEX: PRJ 109

SUBMITTED BY: MUIRURI ANDREW MAINGI

REGISTRATION NUMBER: F17/101710/2017

SUPERVISOR(S): Prof. H. Ouma

Mr. Kinyua

EXAMINER: Eng. Kyalo

This project is submitted in partial fulfilment for the award of the Degree of Bachelor of Science in Electrical and Electronic Engineering at the University of Nairobi

SUBMITTED ON: 26th May 2022

DECLARATION OF ORIGINALITY

This form must be completed and signed for all works submitted to the University for Examination.

Name of student: _____

Registration Number: _____

Faculty/School/Institute: _____

Department: _____

Course Name: _____

Title of the work

DECLARATION

1. I understand what Plagiarism is and I am aware of the University's policy in this regard.
2. I declare that this _____ (Thesis, project, essay, assignment, paper, report e.t.c) is my original work and has not been submitted elsewhere for examination, award of a degree or publication. Where other people's work, or my own work has been used, this has properly been acknowledged and referenced in accordance with the University of Nairobi's requirements.
3. I have not sought or used the services of any professional agencies to produce this work
4. I have not allowed, and shall not allow anyone to copy my work with the intention of passing it off as his/her own work
5. I understand that any false claim in respect of this work shall result in disciplinary action in accordance with University Plagiarism Policy.

Signature: _____

Date: _____

DEDICATION

I dedicate this project to my family: my father, mother and brother.

ABSTRACT

Web-based applications are described as a particular type of software that allows users to interact with a remote server through a web browser interface. Nowadays, improvement in technology has enabled us to realize these web-based platforms for buying and selling of goods, where the platform acts as a third entity in the exchange to enable the other parties to access goods and services more easily and ensure a secure transfer of goods and services by monitoring for fraud. Such a platform can be created for students in schools and universities to provide a place where they can buy and sell goods safely and get discounts that will help them save on money and organize their lives better. In this paper, we introduce “Mwafunzi”, a one-stop online discount platform for students that can be accessed by any student on campus at their own convenience via phone or computer. The idea is to allow students to buy goods from traders or other students at reduced prices and to provide a place for unwanted goods to be advertised and sold instead of being thrown away. This would be particularly helpful as studies have shown that some students, in an effort to save money for other daily expenses like transport and rent, miss meals in universities or even go hungry for long periods of time. To gather more data on the effectiveness and use of an online discount platform in educational institutions, results from several surveys and questionnaires have been used and analyzed to a great extent.

Table of Contents

DECLARATION OF ORIGINALITY	ii
DEDICATION	iii
ABSTRACT.....	iv
LIST OF FIGURES AND TABLES.....	ix
LIST OF ABBREVIATIONS	xii
CHAPTER 1 - INTRODUCTION	1
1.1 - Background.....	1
1.2 - Problem Statement.....	1
1.3 - Justification.....	1
1.4 - Objectives	2
1.5 - Scope	2
CHAPTER 2 - LITERATURE REVIEW	3
2.1 - A Brief Overview of an Online Platform	3
2.2 - Life Before Online Shopping Platforms with Barter Trade.....	3
2.3 - Life with Online Shopping Platforms	3
2.4 - Need for Online Discount Platforms in Schools.....	5
2.5 - A Brief Overview of Web-Based Applications	6
2.6 - Front-End Features of an Online Discount Platform	7
2.7 - Back-End Features of an Online Discount Platform	10
CHAPTER 3 – DESIGN.....	12
3.1 - Introduction	12
3.2 - System Block Diagram	13
3.3 - Subsystem Block Diagrams	14
3.4 - Algorithm.....	16
3.5 - Master Flowchart.....	21
3.5.1 - Login Page	23
3.5.2 - Account Registration Page	25
3.5.3 - Account Recovery Page.....	26
3.5.4 - Home Page.....	27
3.5.5 - Search Form.....	29
3.5.6 - Category Page.....	30
3.5.7 - Commodity Page	32

3.5.8 - Add Commodity Page.....	34
3.5.9 - Contacts Page	36
3.5.10 - About Page	37
3.5.11 - Profile Page.....	38
3.5.12 - Shopping Cart.....	39
3.6 - Pseudocode	41
CHAPTER 4 – IMPLEMENTATION.....	50
4.1 – Database models	50
4.1.1 – The User Model	51
4.1.2 – The Commodity Model	52
4.1.3 – goodsCart Model.....	52
4.1.4 – The goodsPurchased Model	53
4.1.5 – The Pending Model	53
4.1.6 – The Img Model.....	54
4.1.7 – The Dashboard Model.....	54
4.1.8 – The goodsPending Model	55
4.2 - Landing page.....	55
4.3 - User login page.....	57
4.4 – Navigation bar and search bar.....	58
4.4.1 – Navigation Bar	58
4.4.2 – Search bar.....	59
4.5 – Profile page	61
4.6 – Catalog page.....	63
4.6.1 – Catalog Page commodity type routes.....	63
4.6.2 – Commodity Page HTML example.....	63
4.6.3 – Viewing the full picture of the commodity	65
4.6.4 – Reporting a commodity.....	65
4.7 – Shopping cart and checkout	66
4.8 – About and Contact Page.....	71
4.8.1 – About Page.....	71
4.8.2 – Contacts Page.....	72
CHAPTER 5 - RESULTS AND ANALYSIS	73
5.1 – Landing Page	73

5.1.1 – Landing Page	73
5.1.2 – Dashboard section	74
5.2 – User Login page	75
5.3 – Navigation bar and search bar	75
5.4 – Profile Page	76
5.4.1 – User information section	76
5.4.2 – User commodity section	76
5.4.3 – Goods sold and bought by user	77
5.5 – Catalog Page	77
5.5.1 – Electronics Page	78
5.5.2 – Foodstuffs Page	79
5.5.3 – Stationery Page	79
5.6 – Shopping Cart	80
5.6.1 – Shopping Cart page	80
5.6.2 – Payment Page	80
5.6.3 - Pay with MPESA Page	81
CHAPTER 6 - CONCLUSION AND RECOMMENDATIONS	82
6.1 - Conclusion	82
6.2 - Recommendations	82
BIBLIOGRAPHY	vii
References	vii
APPENDICES	x
Code Implemented for This Project	x
views.py	x
base.html	xxv
index.html	xxvi
about.html	xxviii
add_commodity.html	xxix
catalog.html	xxix
contacts.html	xxx
electronics.html	xxxi
cutlery.html	xxxii
foodstuffs.html	xxxiv

stationery.html	xxxvi
genreq.html	xxxvii
recovery.html	xxxviii
send_message.html	xxxviii
shopping_cart.html.....	xxxviii
payment_page.html.....	xxxix
upload_pics.html.....	xl
user.html.....	xl
models.py	xliii
oplat.py.....	xlvi

LIST OF FIGURES AND TABLES

Table 2.1: Results obtained from students' opinions about their nutrition habits	Page 5
Table 2.2: Results obtained from student's opinions about their nutrition habits	Page 6
Table 2.3: Vulnerability ranking for the years 2017 and 2021	Page 10
Figure 2.1: General components of an online shopping platform	Page 7
Figure 2.2: Search and filter tool in action	Page 8
Figure 4.1.1.1: The User Model	Page 51
Figure 4.1.2.1: The Commodity Model	Page 52
Figure 4.1.3.1: The goodsCart Model	Page 52
Figure 4.1.4.1: The goodsPurchased Model	Page 53
Figure 4.1.5.1: The Pending Model	Page 53
Figure 4.1.6.1: The Img Model	Page 54
Figure 4.1.7.1: The Dashboard Model	Page 54
Figure 4.1.8.1: The goodsPending Model	Page 55
Figure 4.2.1: Landing page route	Page 55
Figure 4.2.2: Landing page HTML code	Page 56
Figure 4.3.1: Login page HTML	Page 57
Figure 4.4.1.1: HTML code for the navigation bar	Page 58
Figure 4.4.2.1: HTML code for the search bar	Page 59
Figure 4.4.2.2: Route for processing searched letters	Page 59
Figure 4.4.2.3: A sample of the search page HTML template	Page 60

Figure 4.4.2.4: CSS styling applied to the commodity lists in the search page	Page 61
Figure 4.5.2: Profile page route	Page 61
Figure 4.5.3: HTML code that displays the user's general details	Page 62
Figure 4.6.1.1: Routes for the catalog page and respective commodity pages	Page 63
Figure 4.6.2.1: HTML code for the Electronics Page	Page 64
Figure 4.6.2.2: The HTML code for displaying the commodity details	Page 64
Figure 4.6.3.1: Route for viewing the full picture of a commodity	Page 65
Figure 4.6.4.1: View function for reporting a commodity	Page 65
Figure 4.7.1: Add to Cart Route	Page 66
Figure 4.7.2: Shopping cart default route	Page 66
Figure 4.7.3: The shopping cart HTML template	Page 67
Figure 4.7.4: The payment page route	Page 67
Figure 4.7.5: Routes for cash and MPESA payment options	Page 68
Figure 4.7.6: MPESA HTML template	Page 69
Figure 4.7.7: HTML template for viewing and confirming claim of goods	Page 70
Figure 4.7.8: HTML template showing button for confirming claim of goods	Page 70
Figure 4.8.1.2: About HTML template	Page 71
Figure 4.8.2.1: Contacts HTML template	Page 72
Figure 5.1.1: Landing page for a logged-in user	Page 73
Figure 5.1.2.1: Dashboard of a user with messages	Page 74
Figure 5.1.2.2: Displaying all messages of a user	Page 74
Figure 5.2.1: Login Page for user	Page 75
Figure 5.3: Navigation bar and search bar	Page 75

Figure 5.4.1: Profile page section containing user details and choice to edit	Page 76
Figure 5.4.2.1: Commodity section in the user profile page	Page 76
Figure 5.4.3.1: Section showing goods sold and bought by user	Page 77
Figure 5.5.1: The Catalog Page	Page 77
Figure 5.5.1.1: Electronics page with two commodities	Page 78
Figure 5.5.2.1: The foodstuffs page with one commodity	Page 79
Figure 5.5.3: Stationery Page	Page 79
Figure 5.6.1.2: The Shopping Cart Page	Page 80
Figure 5.6.2.1 – Payment Page	Page 80
Figure 5.6.3.1: Pay with MPESA Page	Page 81

LIST OF ABBREVIATIONS

CSRF	Cross-site Request Forgery
OWASP	Open Web Application Security Project
SQL	Structured Query Language
XSS	Cross-site Scripting

CHAPTER 1 - INTRODUCTION

1.1 - Background

In the past, buying and selling of goods and services necessarily involved the interaction of two parties(people), the buyer and the seller. The buyer had to physically go to the seller's shop or premises and give him/her currency in exchange for goods desired, and more often than not, getting a discount on the purchase was not guaranteed. Nowadays, improvement in technology has enabled us to realize software-based platforms for buying and selling of goods, where the platform acts as a third entity in the exchange to enable the other parties to access goods and services more easily and ensure a secure transfer of goods and services by monitoring for fraud. These platforms have also encouraged the use of online money transactions since the parties could be far away from each other and may need a means of exchanging goods for money and vice versa. The development of online platforms for exchange of goods and services between people has helped many to ensure faster and safer transactions.

1.2 - Problem Statement

The problem seeks to create a platform whereby students can buy and sell goods safely and get discounts. There is a need to store data on all the goods, students and transactions that is accessed via an application that can be run by those involved. The data should also be stored securely and transactions should only be done after the third party has received goods and services from both parties. This will ensure that the transactions are fair to avoid cases of fraud and theft.

1.3 - Justification

This online platform implementation will allow faster transactions between parties and will enable students to acquire goods at a cheaper price where the seller allows. The platform will also encourage fellow students to get rid of goods they don't need and sell them to those in need as the platform will have an option to display discounted unwanted goods from students willing to sell. Furthermore, through provision of discounts, some financial burden will be removed from the students since the saved amount can be used in other areas of the student's life, for example, transport. The platform will be safe and fair since a virtual third party is involved, which only dispatches the payment to the seller only if the attendant (real person) who manages it confirms

that the buyer received the goods and was satisfied. In this case, the virtual third party (the platform) will be used to store the details of the transaction for future reference.

1.4 - Objectives

a) Main Objective

1. To implement a one-stop discount platform for student support

b) Specific Objectives

1. To provide a safe platform for exchange of goods and services on-campus through the use of a virtual third party.
2. To provide a faster online option for exchange of goods and currency since there is faster access to buyers and sellers through the platform.
3. To provide goods to students at a reduced price to lessen financial burden through the provision of discounts.

1.5 - Scope

This is a platform for handling goods that are used by students and should only be securely accessible to them and not outsiders. Any user wanting to buy goods will be required to log in with a valid email address and password. Also, the platform should only display the name and contact information of the buyers and sellers from the database and should not show or leak any other information that may have been required for registration purposes.

CHAPTER 2 - LITERATURE REVIEW

2.1 - A Brief Overview of an Online Platform

According to [1], an online platform is a digital service that enables interaction between two or more different sets of people or parties through the Internet. This term can be used to refer to marketplaces, social media and other communication sites, payment systems and app stores. In the context of trading goods and services on these online platforms, the term “online shopping” is defined by [2] as the process of looking into goods and services and buying them via the Internet. As seen from the two terms defined above, the presence of Internet was one key component in developing online platforms for shopping, and [3] agrees by saying that, although there are various other ideas and reasons involved with the development of this shopping phenomenon, the Internet is very much involved.

2.2 - Life Before Online Shopping Platforms with Barter Trade

A convenient means of exchange is another idea that boosted the emergence of online shopping. According to [4], bartering is an old means of exchange which had been practiced long before the introduction of money. This system involved exchanging goods for goods of equivalent value, but this faced a problem of fairness, whereby the equivalent value between same goods to be exchanged was not arrived at easily. A second problem cited is that, say, a trader, who needed timber, with grains to exchange, could not guarantee that he/she would get such an agreement, as different people have different needs. This method, however, is still used today because of its advantages, and as [5] mentions, these include simplicity of process, no problem of underproduction and overproduction, and no concentration of economic power. In bartering, there is no wastage since only the required quantity is produced and there was no unnecessary accumulation of wealth. Although bartering has its advantages to this day, purchasing using monetary currencies is common in online shopping.

2.3 - Life with Online Shopping Platforms

In fact, [2] states that most online shoppers use modern currency transfer options such as credit cards, PayPal, electronic money transfer, electronic bank checks and gift cards. This means of exchange eliminates the problems associated with barter trade, as there is fairness in value of goods and services exchanged, and there is a guarantee of agreement with the right amount of money.

The amount to be paid is decided after placing the items to be bought in a shopping cart on the online platform, and as [2] describes, they are usually integrated into the company's own web server so that the ordering, payment and delivery can be optimized and automated. The emergence of online shopping made people's lives easier and more convenient, and as [3] states, there were problems experienced back then that seem fairly foreign nowadays. For instance, going to a physical shop required one to prepare to leave one's comfort and carry enough money for payment of goods and services. Also, if the goods to be obtained were bulky, transportation was a major concern, since one had to negotiate with other parties to facilitate safe transport. Furthermore, crowding of shopping places during peak times hindered comfortable shopping in malls and supermarkets, and one could not compare the prices of goods from different vendors so as to get a convenient price point. Nowadays, these problems are almost non-existent, since comparison of prices, purchase and transportation of goods can be done or arranged away from the store via an online platform.

Online shopping, however, has its disadvantages as well. According to [6], one major disadvantage associated with it is the possibility of fraud. Fake online shopping platforms can lure unsuspecting customers to send money for goods ordered, and in the end, customers may not receive ordered goods while they part ways with their hard-earned money. Also, the scammers may also dispatch counterfeit or damaged goods to the buyers which may cause harm or inconvenience. Another disadvantage is that the price of goods sold cannot be negotiated, thus the customer loses a means of lowering the price paid. Although there may be coupons and gift cards, they may be attached with conditions that may require the customer to spend even more money. Lack of adequate verbal and physical interaction, which is another problem associated with online platforms, may cause a lack of personal attention to customers, which may result in incomplete knowledge of the goods on sale. Also, the process of returning a product, if confirmed to be damaged, may be lengthy and tedious as most companies follow strict guidelines and protocols to ensure accountability on their end. This may also result in delay in delivery of the new goods since the company may be processing your request and others from different customers. However, as [3] states, the advantages greatly outweigh the disadvantages and the existence of online shopping has definitely made our lives easier to handle.

2.4 - Need for Online Discount Platforms in Schools

Implementing a means of facilitating online shopping with discount for students in schools and campuses is an excellent recommendation. According to [7], from the results of a questionnaire presented to 320 fourth-year students at Uludag University in Turkey, a high percentage of male and female students skip meals for various reasons, including insufficient amount of money to afford all three servings of food in a day.

Table 2.1: Results obtained from students' opinions about their nutrition habits

Nutrition Habits	Education Faculty				Engineering and Architecture Faculty			
	Female		Male		Female		Male	
	Yes	No	Yes	No	Yes	No	Yes	No
	f	%	f	%	f	%	f	%
Do you believe that you take your nourishment healthily?	49	58	30	23	19	25	55	61
	45,79	54,21	56,60	43,40	43,18	56,82	47,41	52,59
Have you lived any weight problems after starting university?	44	63	18	35	17	27	45	71
	41,12	58,88	33,96	66,03	38,64	61,36	38,79	61,21
Do you skip any meal within a day?	87	20	40	13	35	9	78	38
	81,30	18,70	75,47	24,53	79,55	20,45	67,24	32,76

From Table 2.1[7], it is seen that an average of 80.4% of female students and 71.36% of male students skip at least one meal within a day, and an average of 55.5% of female students and 48% of male students believe they do not take a healthy nourishment of food daily. These latter percentages are a cause for concern since, as [7] explains, university students are in their youthful stage where they need to develop their bodies by eating healthily and in adequate amounts. In a separate study, results obtained from 180 students from a different institution revealed that “time restriction” and “insufficient economic possibilities” are key reasons why students skip meals. Furthermore, “inadequacy of storage and cooking facilities” is another reason cited, where the students claim to lack fridges and other cooking amenities to safely prepare and store food, a problem likely caused by limited financial resources by students. The first recommendation given by [7] is to improve nutrition possibilities in places where students live, and this can be achieved by implementing an online discount platform which sells meals to students who are part of a learning institution. This recommendation can be backed up by the findings shown in Table 2.2[7], where a smaller percentage of students (an average of 8.75% of female and 10.23% of male

students) are seen to attach least significance on lunch as a meal and a higher percentage of students eat most meals at home compared to the school dining hall.

Table 2.2: Results obtained from student's opinions about their nutrition habits

Food and Accomodation Conditions	Education Faculty				Engineering and Architecture Faculty			
	Female		Male		Female		Male	
	f	%	f	%	f	%	f	%
Which of the following meals can you attach sufficient importance?								
Breakfast	31	28,97	12	22,64	14	31,81	23	19,82
Lunch	9	8,41	4	7,54	4	9,09	15	12,93
Dinner	40	37,38	22	41,50	20	45,45	54	46,55
All	27	25,23	15	28,30	6	13,63	24	20,68
At which of the following places do you eat meals most?								
Home	59	55,14	41	77,36	24	54,54	82	70,69
School / Dining Hall	28	26,17	8	15,10	9	20,45	7	6,03
Restaurant/Cafe/ Patisserie	20	18,69	4	7,54	11	25	27	23,27
Where do you reside ?								
With my family	35	32,71	13	24,53	21	47,73	45	38,79
House	31	28,97	30	56,60	10	22,72	47	40,52
Dormitory / Hostel	41	38,32	10	18,87	13	29,55	24	20,69

An online discount platform will not only help to save time in cooking and preparing food, but will also provide cheaper and lighter meal options compared to those offered by independent sellers.

2.5 - A Brief Overview of Web-Based Applications

In [12], web-based applications are described as a particular type of software that allows users to interact with a remote server through a web browser interface. They have a number of advantages over traditional desktop-based applications, for example, they have more portability in terms of accessibility from anywhere anytime as long as an internet connection exists, high scalability since it is much easier to increase the number of active users as compared to desktop programs, and are secure since in-built security mechanisms as well as web moderators can address any intrusions and errors quickly. [12] also adds that web-based applications can be a valuable tool for businesses since they are capable of monitoring financial transactions, tracking activities of users and controlling workflow of staff and workers involved with the application. [13] goes further and

states that most companies adopt web-based applications to improve communication within their businesses and improve their relationships with their clients.

2.6 - Front-End Features of an Online Discount Platform

To implement such a platform, the design requirements need to be carefully considered and implemented. As [2] suggests, a user-centered design is very important. A good online shopping platform should satisfy the consumers' expectations, and its complexity should be moderated in such a way as to keep customers interested and exploring. [2] adds that a good online shopping platform should provide detailed product information that is not available in physical retail stores so as to adequately inform the customers to enable them to make the right decisions based on their needs. Also, the owners of the platform must respond to customer queries and complaints on time and must be good stewards of their data to prevent private data leakage. [2] also says a well-designed online platform may lead to increased brand awareness, increased sales and customer loyalty, and may help to deal with issues associated with them, as discussed in [6] previously.

Some basic design requirements that are necessary for the implementation of an online discount platform are discussed in [8]. Four categories can exist, as seen in Figure 2.1[8].

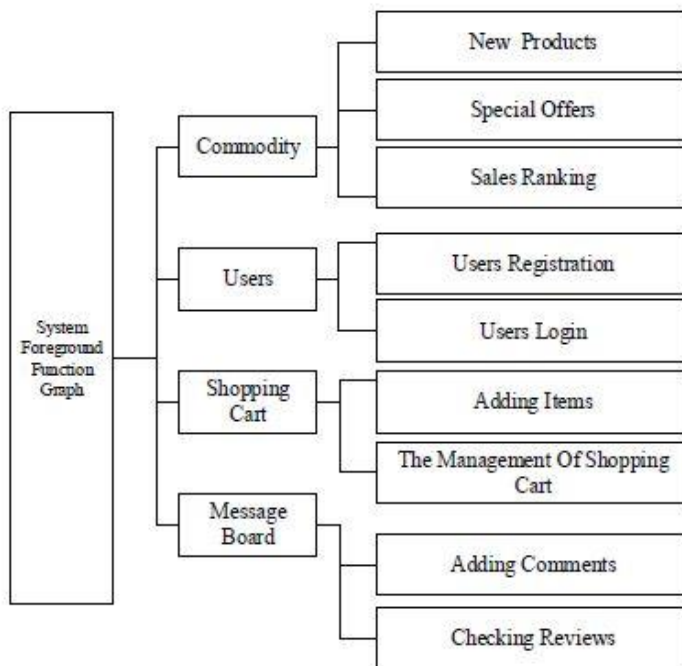


Figure 2.1: General components of an online shopping platform

According to [8], firstly, the platform should have access to a database for storing user and commodity information. User information such as usernames, passwords and addresses and commodity information includes their prices, images, weight and availability should be well organized for easy presentation to the user. [9] adds that the user could also be allowed to register his/her other social media handles such as Twitter and Facebook to enable easy login for a returning customer. Secondly, the platform should provide the user with adequate commodity information and accurate commodity classification so that the user can compare the products and make good decisions. [10] suggests that some websites and applications nowadays are showing 360-degree pictures of their commodities, which awes customers and increases the chances of a conversion. The next thing to implement is a shopping cart, where users can add and remove individual commodities as well as edit their quantities as needed. This is where an appropriate message board comes in, whereby, after every important action made on the platform, a message is displayed on a dialog box that appears for some few seconds and disappears. According to [9], adding a “save for later” function is a tool that could be used to turn visitors into loyal customers, whereby an unregistered user can add a commodity to a shopping cart and continue browsing the platform to discover more products.

To discover new products in the platform, an effective search and filter functionality could be implemented to enable customers to quickly find what they are looking for, as seen in Figure 2.2[9].

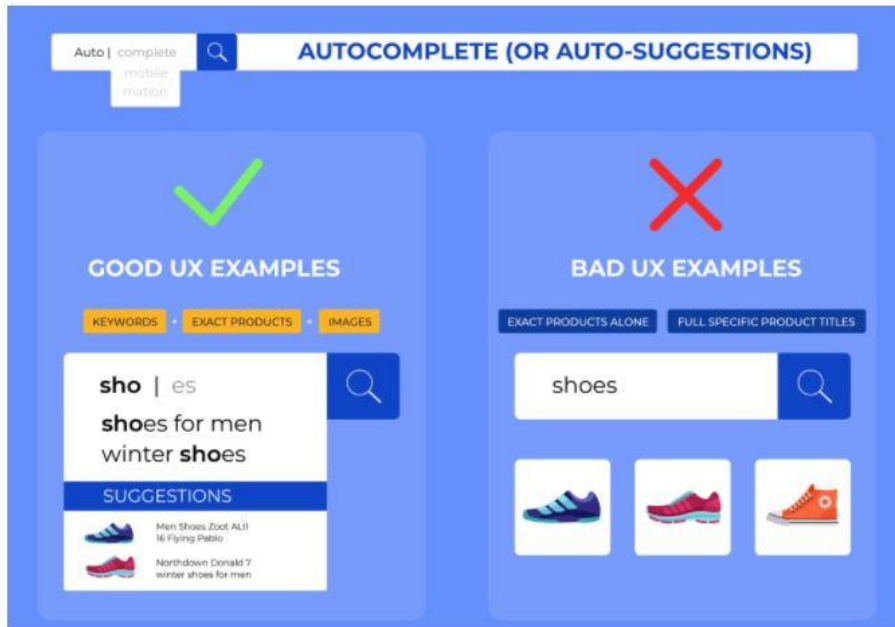


Figure 2.2: Search and filter tool in action

The search results should be neatly arranged alphabetically so that they can be easily parsed through and the filtering feature could categorize according to age, price and size of the goods sold. The platform should be able to apply more than one filter simultaneously so as to allow the user to find the exact product he/she is looking for. In the shopping cart implementation, [9] recommends adding popular payment options such as PayPal or credit card to facilitate easy checkout for the users. This feature should be a priority, since, according to statistics, easy checkout is the most popular feature for online shoppers, and therefore an online discount platform should not operate without this feature. [9] adds that implementing a customer support chat box is quite popular nowadays as it facilitates faster responses from human or computer operators to customers to enable a customer to make key buying decisions.

Special discounts can be offered to new and loyal customers to encourage them to continuously use the platform in the future. For specific goods offered by individual owners such as radios, furniture and computer equipment, their discount information can be provided by the same owners in the platform, while, for general goods offered by manufacturers such as flour, cooking oil and cocoa, discount rates can be determined using the guidelines given by [11]. One principle given is

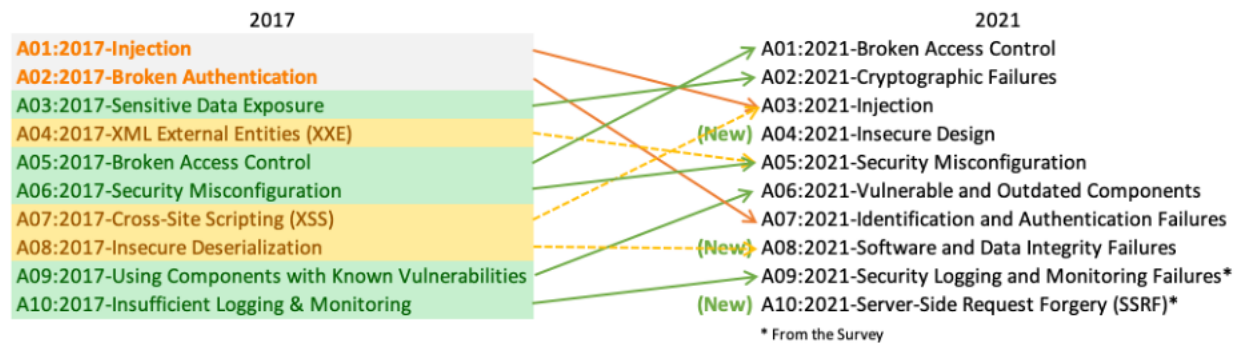
that the percentage of discount given depends on the demand for a given product. As prices for goods go down, demand goes up, and thus commodities of low price should not be heavily discounted. The opposite is true for high-priced goods, which could be adequately discounted to influence the customer to buy them. Another principle is that not every customer deserves a discount. According to [11], discounts should be earned rather than given freely to a customer. Shopping platforms should reward loyalty, and offering discount coupons to loyal customers is one way of ensuring they always come back. Online discount platforms need not follow this principle, as it defeats the purpose of existence of the platform. Giving discounts will not only increase sales, but it has the potential to get rid of unsold goods and acquire new customers.

2.7 - Back-End Features of an Online Discount Platform

[14] explores the concept of data storage using databases which could be of two types: SQL and NoSQL databases where, in SQL databases, the tables are relational and consists of columns and rows that are used to store data as records while in NoSQL databases, collections and documents are used instead of tables and records respectively. [14] adds that any of the two are perfectly applicable for small to medium applications, and that the difference between the two arises from the ACID paradigm, whereby SQL databases rely heavily on these principles to achieve a high level of reliability, while NoSQL databases relax some of these requirements in order to get a performance edge. [15] provides the requirements to be satisfied for successful web-DBMS integration as minimal administration overhead, the right and ability to use valuable corporate data in a fully secured manner, cost-reducing method which allows for scalability, among others. In this regard, [14] agrees by mentioning factors that need to be considered before choosing the database to be used, including performance, portability and ease of use.

[16] states that developing a secure online platform is a difficult task, and therefore attempts to study the effectiveness of the OWASP guideline to a developer. This guideline contains a list of attacks and vulnerabilities that may compromise the security of an application, and as [17] states, as per 2017, the most common and severe attack was SQL injection, which is injecting SQL code into a web application and being able to retrieve data from a database such as username and password credentials or changing sensitive data. This can be seen in Table 2.3[17], where the severity of SQL injection is seen to have reduced between the years 2017 and 2021.

Table 2.3: Vulnerability rankings for the years 2017 and 2021



Others include XSS, security misconfiguration, CSRF and broken authentication and session management. [14] explains that a CSRF attack occurs when a malicious website sends requests to the application server on which the user is currently logged in. [17] adds that this is how an attacker can cause a user to change their password, and that knowledge on this and other vulnerabilities described by OWASP can be a starting ladder on maintaining application security. [14] advises that the best way to store user passwords on a database is storing the hash of the password, and not the actual password.

[14] states that long waits for pages to load frustrates users since nobody likes slow applications, so it is important to detect and correct performance problems as soon as they appear. He states that cause of slow performance of online platforms can be due to slow database queries that get worse as the database grows bigger. He gives ways of solving this but first recommends identifying the slow queries by using end-to-end testing methods, implementing unit tests or using data obtained from coverage reports, and adds that manual testing is time-consuming and not as conclusive as using unit tests.

CHAPTER 3 – DESIGN

3.1 - Introduction

To implement the online discount web application, the following pages will be accessible to a registered user:

1. Login Page
2. Account Registration Page
3. Account Recovery Page
4. Home Page
5. Search Pages
6. Catalog page
7. Category page
8. Add Commodity Page
9. Commodity Page
10. Shopping Cart Page
11. Contacts Page
12. About Page
13. Account Settings Page

Upon opening the web application, the user will have to sign in, create or recover an account in order to use the platform. The main pages will be accessed by clicking the tabs displayed on the Home Page. Hence, the tabs on the Home Page are:

1. Home Tab
2. Catalog Tab
3. Contacts Tab
4. About Tab
5. Profile Tab
6. Log Out Tab

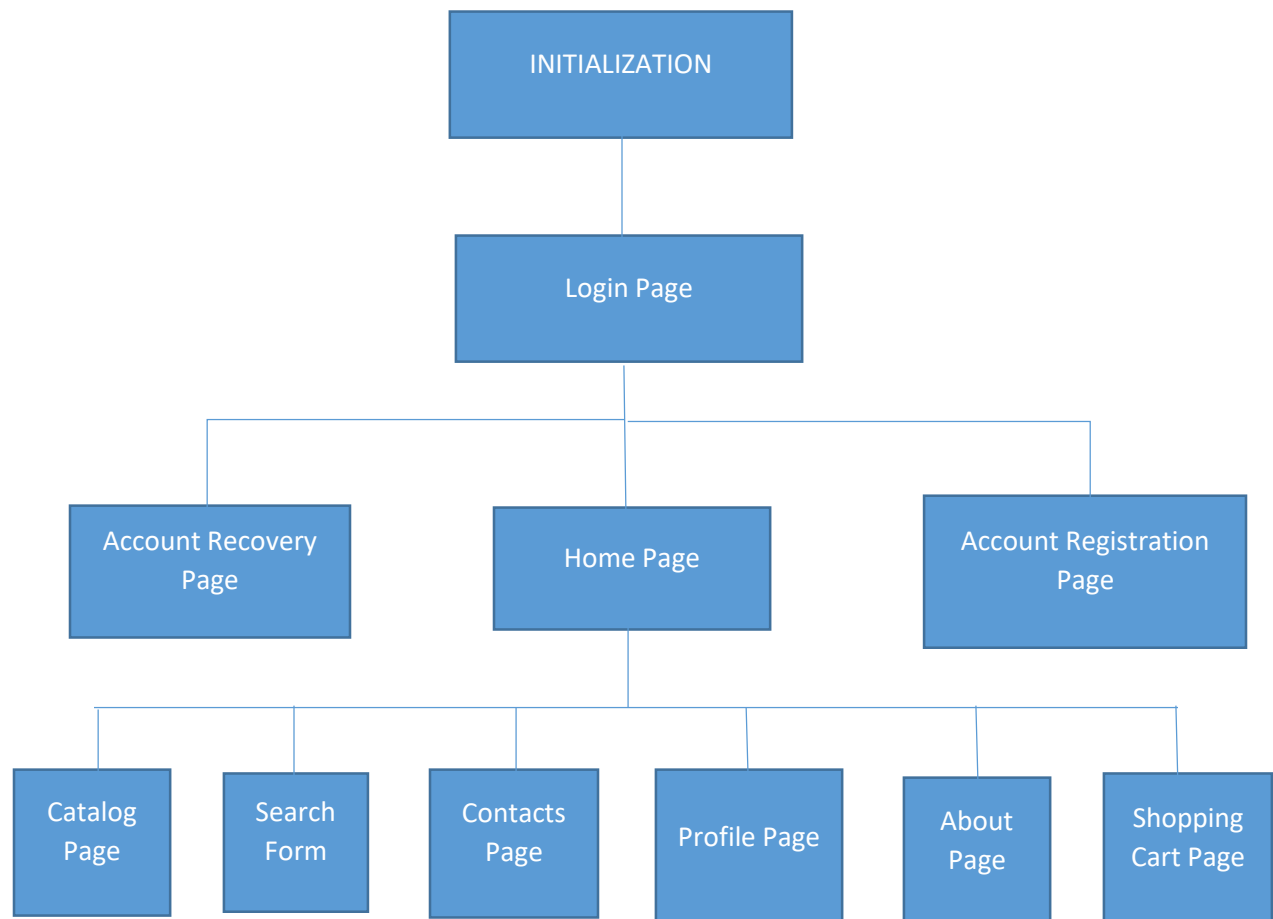
A search form will also be displayed on each main and category page to allow the user to search for a commodity easily. The shopping cart Page will be accessed by clicking the shopping cart icon which will be displayed at the top right side of every main page.

The Catalog Page contains the list of links of categories of goods on sale, that is:

1. Electronic
2. Cutlery
3. Stationery
4. Foodstuffs

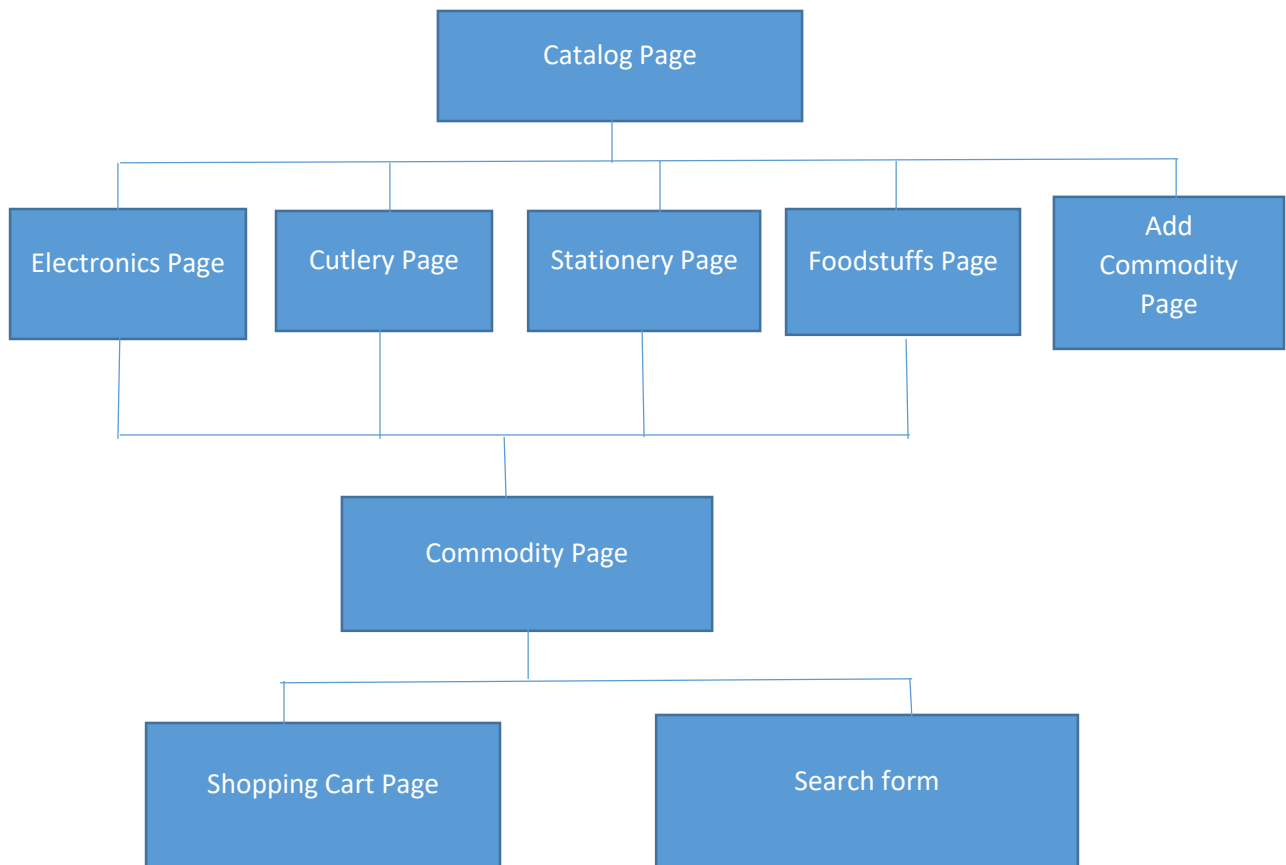
The individual links will lead to the individual category pages. The Add Commodity link will be displayed in the Catalog Page and will lead to the Add Commodity Page. The Account Settings Page will allow the user to change his/her credentials such as email, password and phone number. The Log Out Page will redirect the user to the Login Page, where a different user can then access the platform.

3.2 - System Block Diagram

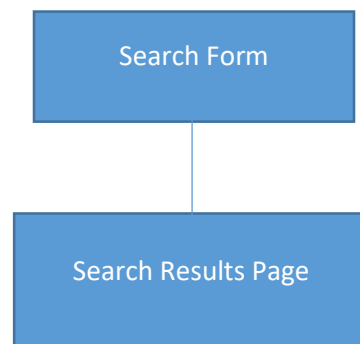


3.3 - Subsystem Block Diagrams

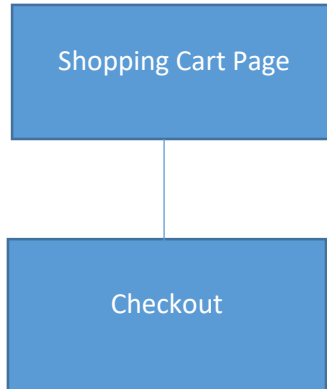
Catalog Page



Search Form



Shopping Cart Page



3.4 - Algorithm

Version 1

1. Login stage
2. Navigation stage
3. Checkout stage

Version 2

1. Login using user credentials
2. Navigate pages
3. Put item(s) in the online shopping cart
4. Checkout and payment in the shopping cart
5. Log out of the web application

Login Using User Credentials

- i. Load necessary variables
- ii. Select between “new user” and “existing user”
- iii. If the user is a new user, redirect to the account registration page
- iv. Else, display the email form, password form and submit button
- v. Input the user’s email and password in the login page
- vi. Select the type of user, that is Seller or Buyer
- vii. If the submit button has been clicked and user credentials are valid, redirect to home page
- viii. Else if the submit button has been clicked and the user credentials are incorrect, display a pop-up message that warns the user and deny entry
- ix. Else if the user has forgotten his/her password, redirect to account recovery page

Account Registration Page

- i. Load necessary variables
- ii. Display the user selector and forms used for inputting data, that is, name, student registration number, year of study, phone number, email and password
- iii. Select the type of user, that is, Seller or Buyer
- iv. Input the required data in all the fields
- v. If the submit button has been clicked and all fields have been filled and registration number is valid, store credentials in the database and redirect to the home page
- vi. Else if the submit button has been clicked and all fields have not been filled, display a pop-up that warns the user
- vii. Else if the submit button has been clicked and the registration number is not valid, display a warning pop-up that warns the user

Account Recovery Page

- i. Load necessary variables
- ii. Display the recovery email form and submit button
- iii. Input the recovery email address
- iv. If the submit button has been clicked and recovery email address is valid, an email is sent to that address with recovery instructions
- v. Else if the submit button has been clicked and the recovery email address is not valid, a pop-up warning is flashed on the screen

Home Page

- i. Load necessary variables
- ii. Display a welcome message, the Home, Catalog, Contacts, About and Log Out tabs, the Search Form, shopping cart icon and account settings tab
- iii. If the user selects the search form, expand the form to reveal more space for inputting text
- iv. Else if user selects the Home tab, redirect to the Home page
- v. Else if the user selects the Catalog tab, redirect to the Goods on Sale page
- vi. Else if the user selects the Contact tab, redirect to the Contacts page
- vii. Else if the user selects the About tab, redirect to the About page
- viii. Else if the user selects the Log Out tab, redirect to the Login page
- ix. Else if the user selects the shopping cart icon, redirect to the shopping cart page
- x. Else if the user selects the account settings tab, redirect to the account settings page

Search Form

- i. Load necessary variables
- ii. Display an expanded search bar and a search button
- iii. Type text to be searched
- iv. If the search button has been clicked and the searched text is valid, redirect to a page containing the appropriate search results
- v. Else if the search button has been clicked and the text is not valid, display a pop-up claiming there is no information available on the searched text

Catalog Page

- i. Load necessary variables
- ii. Display Home, Catalog, Contacts, About and Log Out tabs, search bar, search button, “add commodity” button and icons and links representing the categories of goods to be sold
- iii. If the user clicks on the “add commodity” button, redirect to the Add Commodity Page
- iv. Else if the user selects the search form, expand the form to reveal more space for inputting text
- v. Else if the user selects Electronics link, redirect to the Electronics Page
- vi. Else if the user selects Foodstuff link, redirect to the Foodstuffs Page
- vii. Else if the user selects Stationary link, redirect to the Stationary Page

- viii. Else if user selects Cutlery link, redirect to the Cutlery Page
- ix. Else if the user selects the Home tab, redirect to the home page
- x. Else if the user selects the Catalog tab, redirect to the catalog page
- xi. Else if the user selects the Contacts tab, redirect to the contacts page
- xii. Else if the user selects the About tab, redirect to the about page
- xiii. Else if the user selects the Log Out tab, redirect to the login page
- xiv. Else if the user selects the Account Settings tab, redirect to the account settings page
- xv. Else if the user clicks on the shopping cart icon, redirect to the shopping cart page

Electronics Commodity Page

- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, “add to cart” button, shopping counter, shopping cart icon, name, pictures and information about the particular electronics product
- iii. Else if the user selects the Home tab, redirect to the Home page
- iv. Else if the user clicks on the “Add to cart” button, the commodity is added to the shopping cart and the shopping counter increased

Foodstuffs Commodity Page

- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, “add to cart” button, shopping counter, shopping cart icon, name, pictures and information about the particular food product
- iii. Else if the user selects the Home tab, redirect to the Home page
- iv. Else if the user clicks on the “Add to cart” button, the commodity is added to the shopping cart and the shopping counter increased

Cutlery Commodity Page

- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, “add to cart” button, shopping counter, shopping cart icon, name, pictures and information about the particular cutlery product
- iii. Else if the user selects the Home tab, redirect to the Home page
- iv. Else if the user clicks on the “Add to cart” button, the commodity is added to the shopping cart and the shopping counter increased

Stationery Commodity Page

- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, “add to cart” button, shopping counter, shopping cart icon, name, pictures and information about the particular stationery

- iii. If the user selects the Home tab, redirect to the Home page
- iv. Else if the user clicks on the “Add to cart” button, the commodity is added to the shopping cart and the shopping counter increased

Add Commodity Page

- i. Load necessary variables
- ii. Display the fields required to input information that describes the commodity to be added, that is, information about its name, type, manufacture date, expiry date, quantity, any other additional info, and a submit button
- iii. Input the required data
- iv. If the submit button has been clicked and the commodity’s name, type and quantity entries are valid, then first, add the information to the database, display a confirmation pop-up and clear the filled fields
- v. Else if the user selects the Home tab, redirect to the Home page
- vi. Else if the user clicks on the shopping cart icon, redirect to the shopping cart page

Contacts Page

- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, shopping counter, shopping cart icon and contact information
- iii. If the user selects the Home tab, redirect to the Home page
- iv. Else if the user clicks on the shopping cart icon, redirect to the shopping cart page

About Page

- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, shopping counter, shopping cart icon and information about the app
- iii. If the user selects the Home tab, redirect to the Home page
- iv. Else if the user clicks on the shopping cart icon, redirect to the shopping cart page

Profile Page

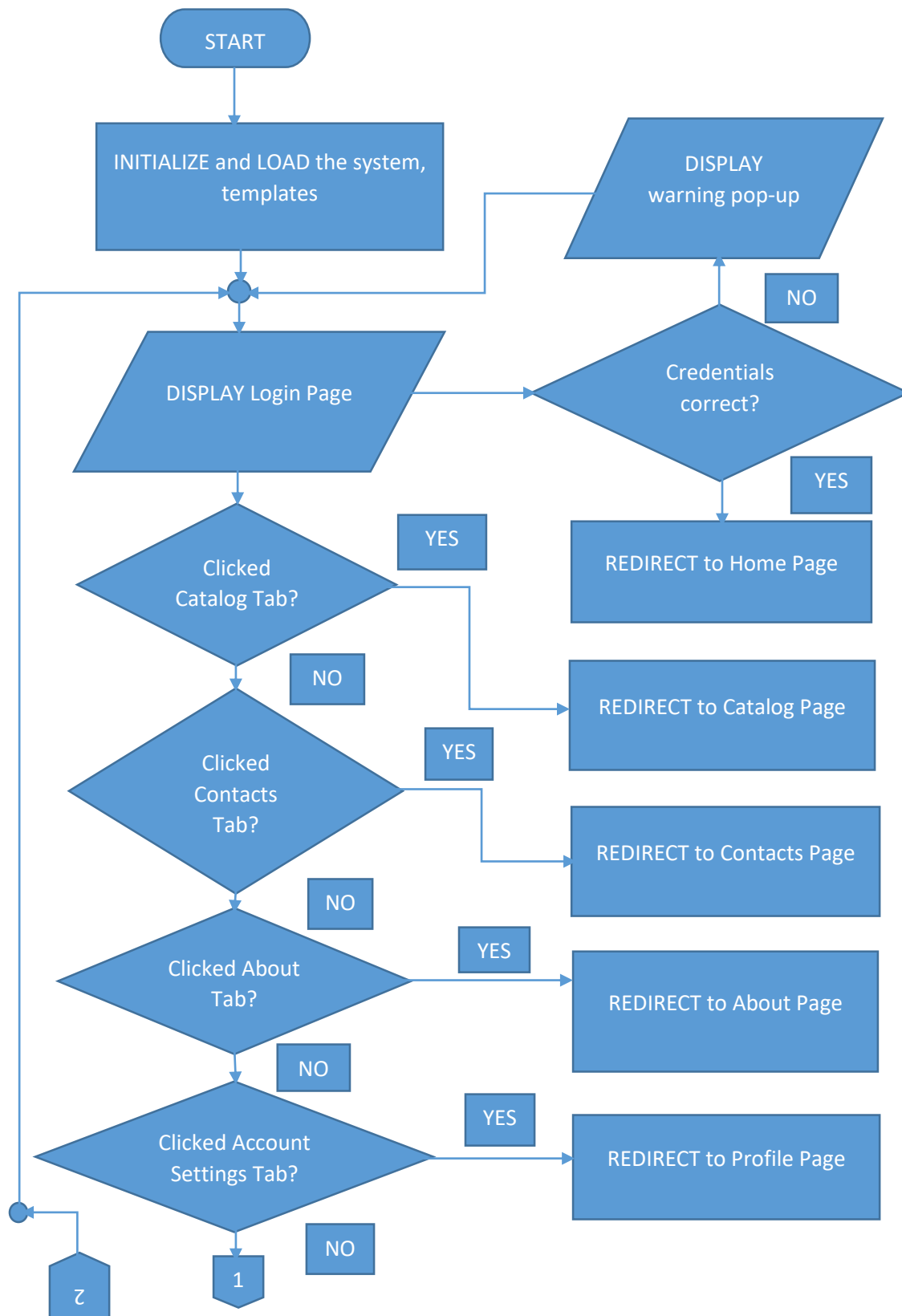
- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, shopping counter, shopping cart icon, and forms for inputting the relevant user information, that is, email, password, registration number, name, year of study and phone number

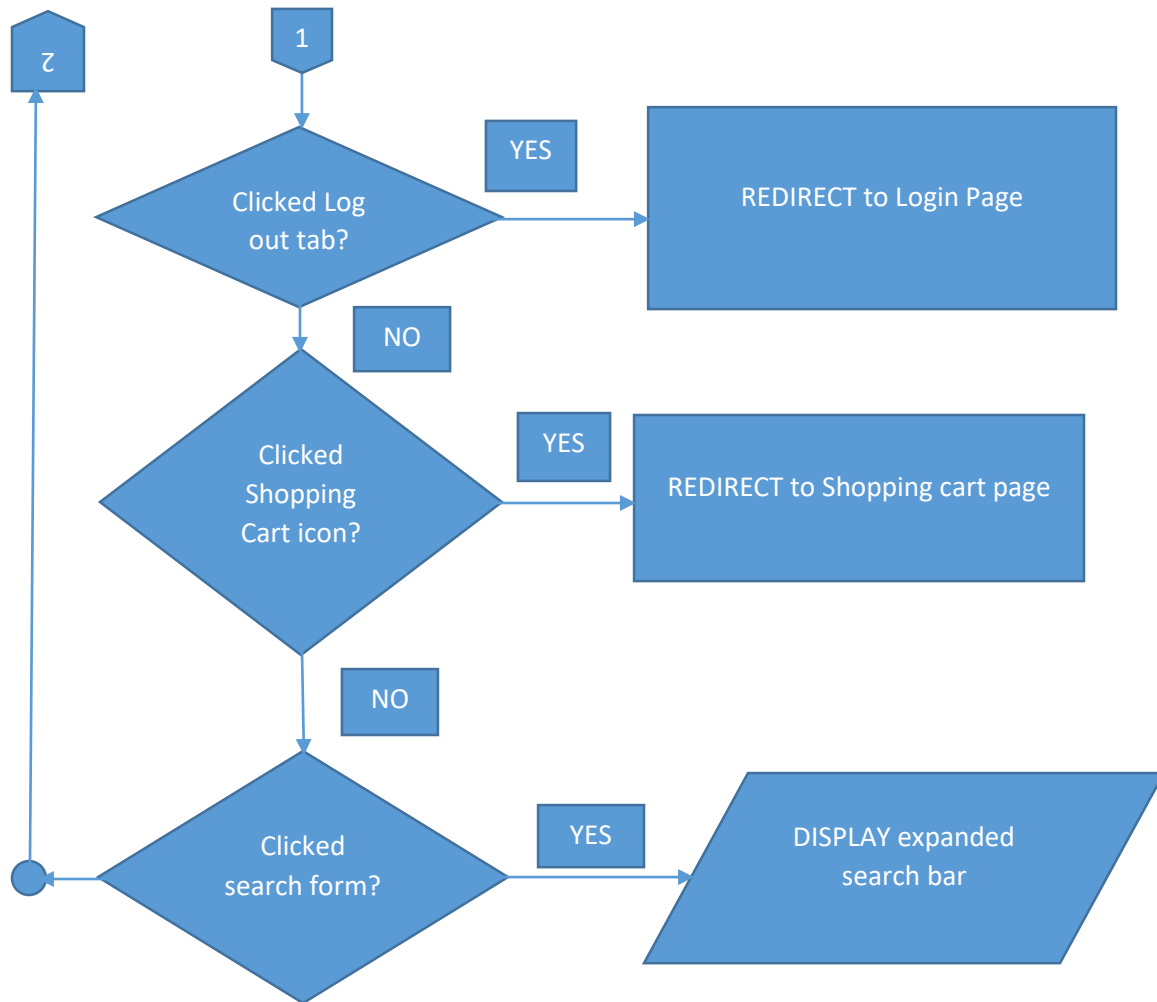
- iii. If the submit button is pressed and any new information different from that in the database has been added, then delete that particular credential and update to the new one
- iv. Else if the user selects the Home tab, redirect to the Home page

Shopping Cart Page

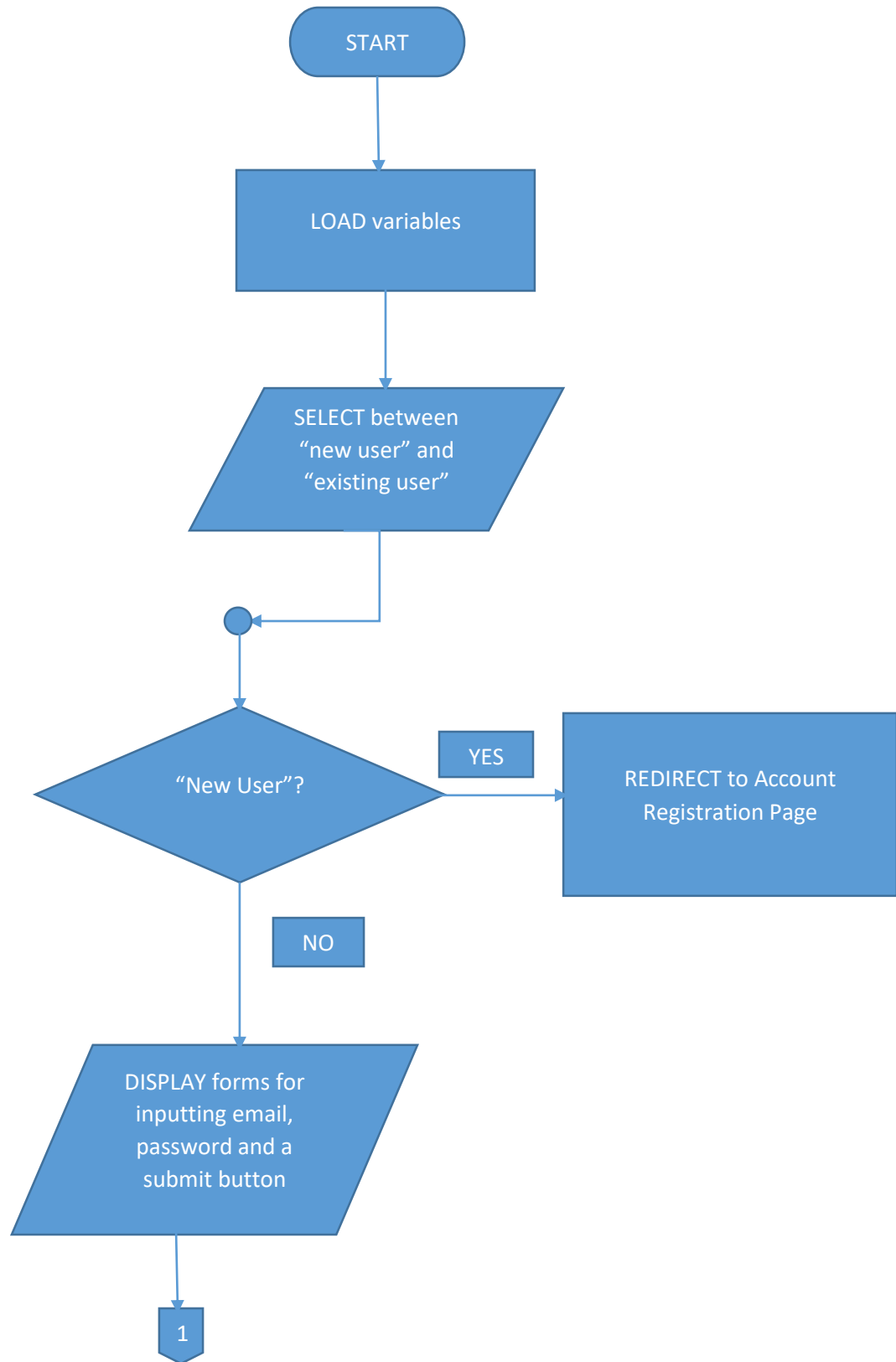
- i. Load necessary variables
- ii. Display Home, Catalog, Goods on Sale, Contacts, Account Settings, About and Log Out tabs, the Search form, search button, shopping counter, shopping cart icon, and the list of commodities that the user has ordered with their commodity type, name and count
- iii. If the user clicks the “Checkout” option, display a pop-up that shows the Paybill number to use for paying for the goods. If the payment is received, display a pop-up confirming the purchase and a new window containing the payment receipt
- iv. Else if the user increases the quantity of a certain type of commodity, display a confirmation message and increment the commodity count
- v. Else if the user decreases the quantity of a certain type of commodity, display a confirmation message and decrease the commodity count
- vi. Else if the user removes a commodity from the shopping cart, decrease the shopping counter and display a confirmation message

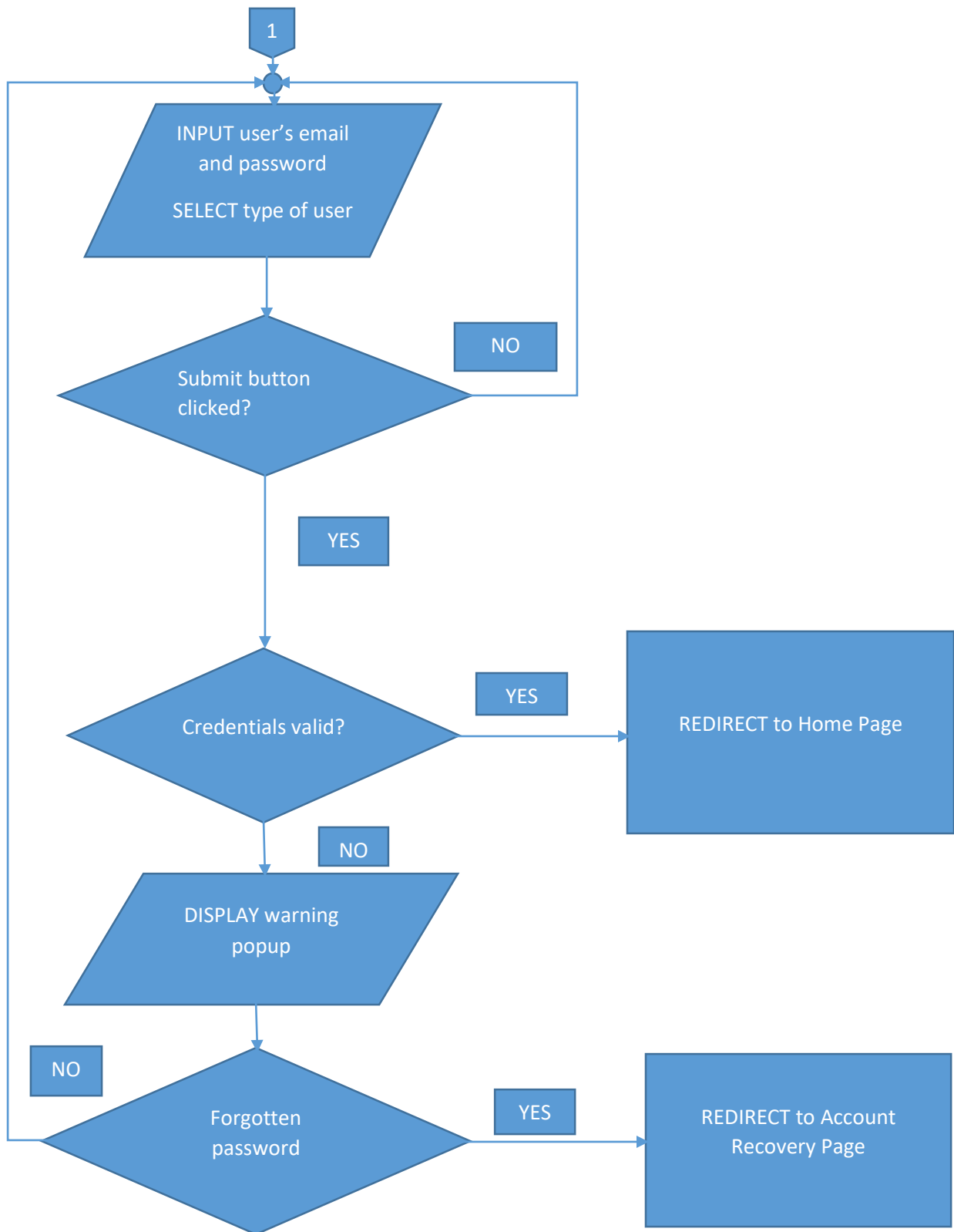
3.5 - Master Flowchart



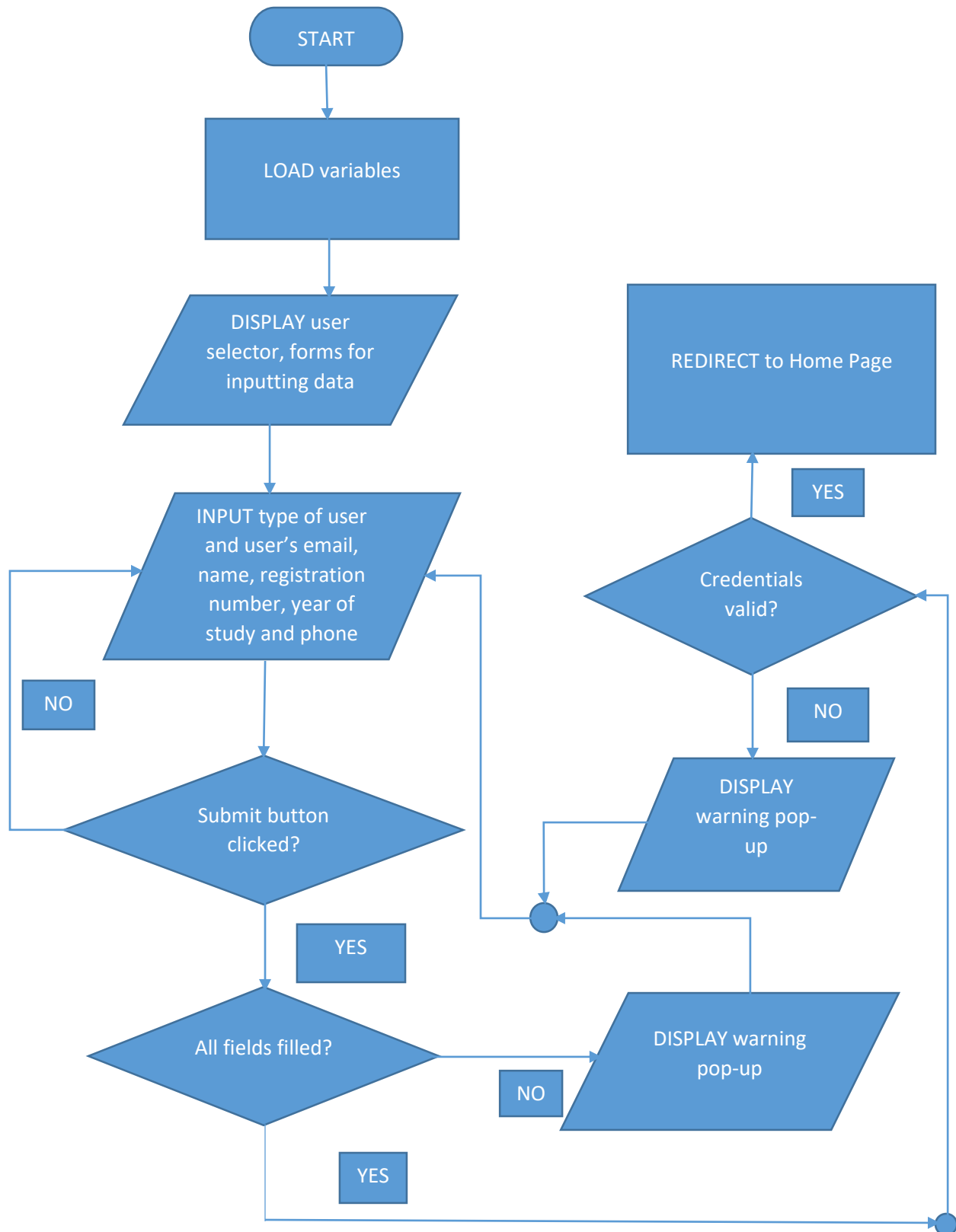


3.5.1 - Login Page

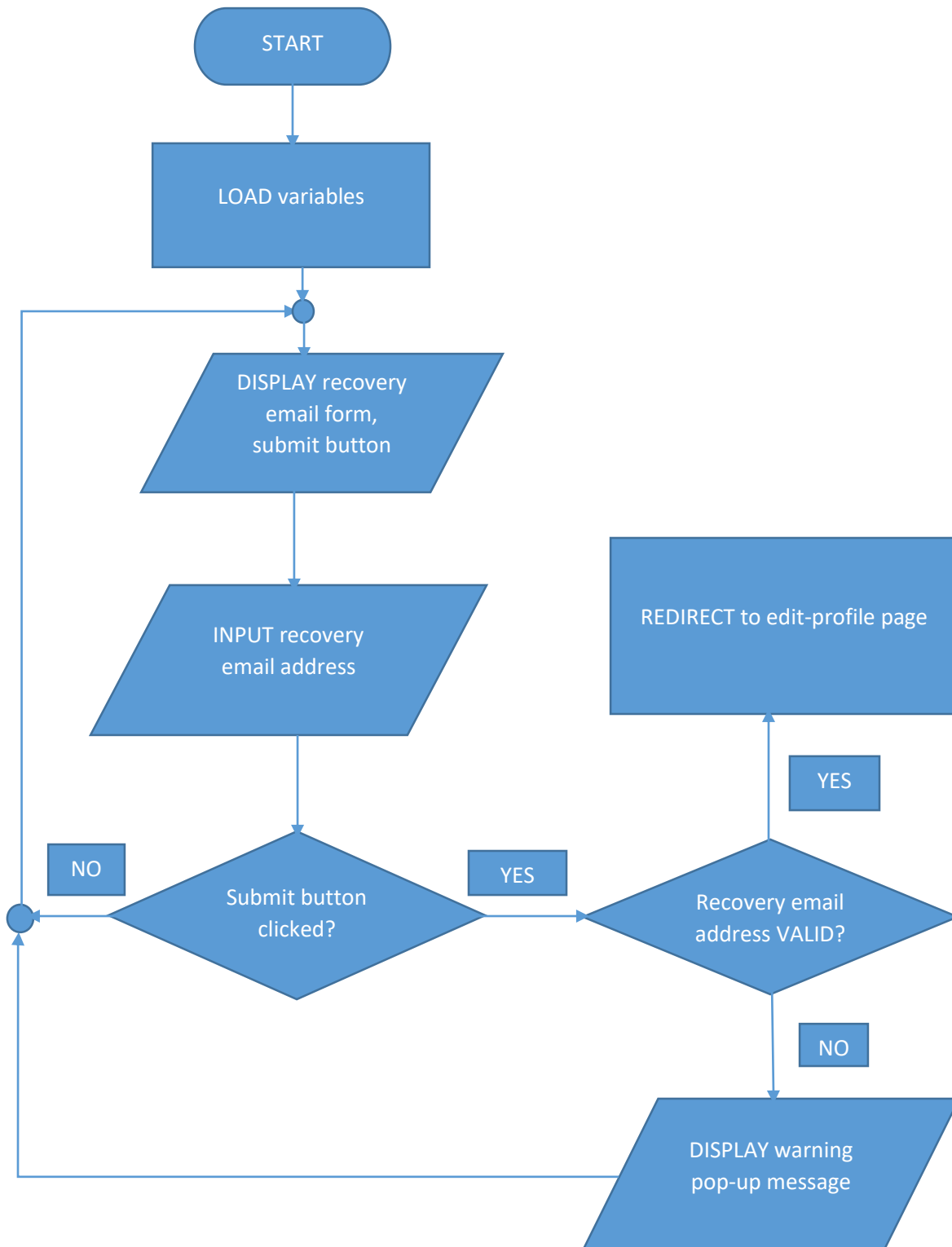




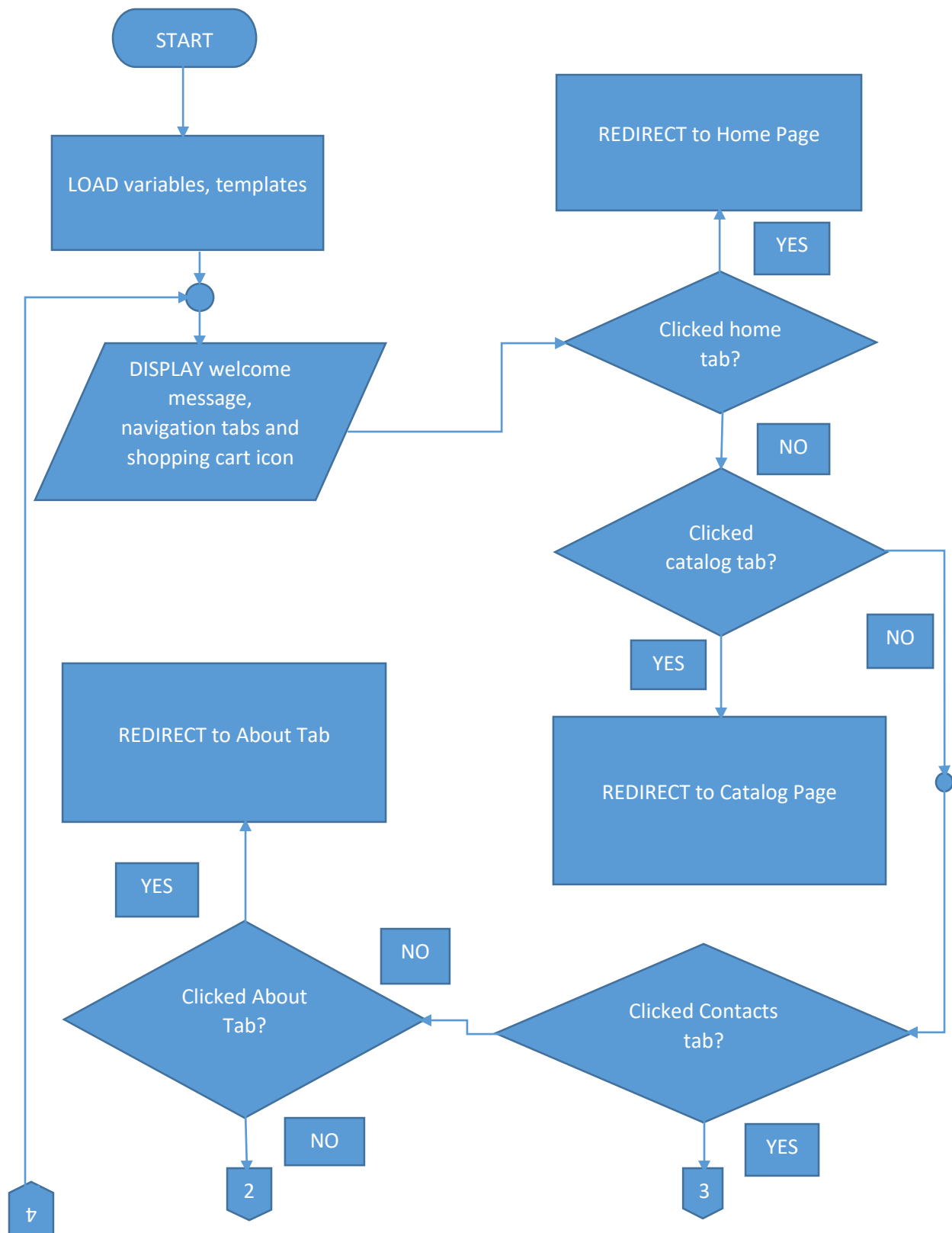
3.5.2 - Account Registration Page

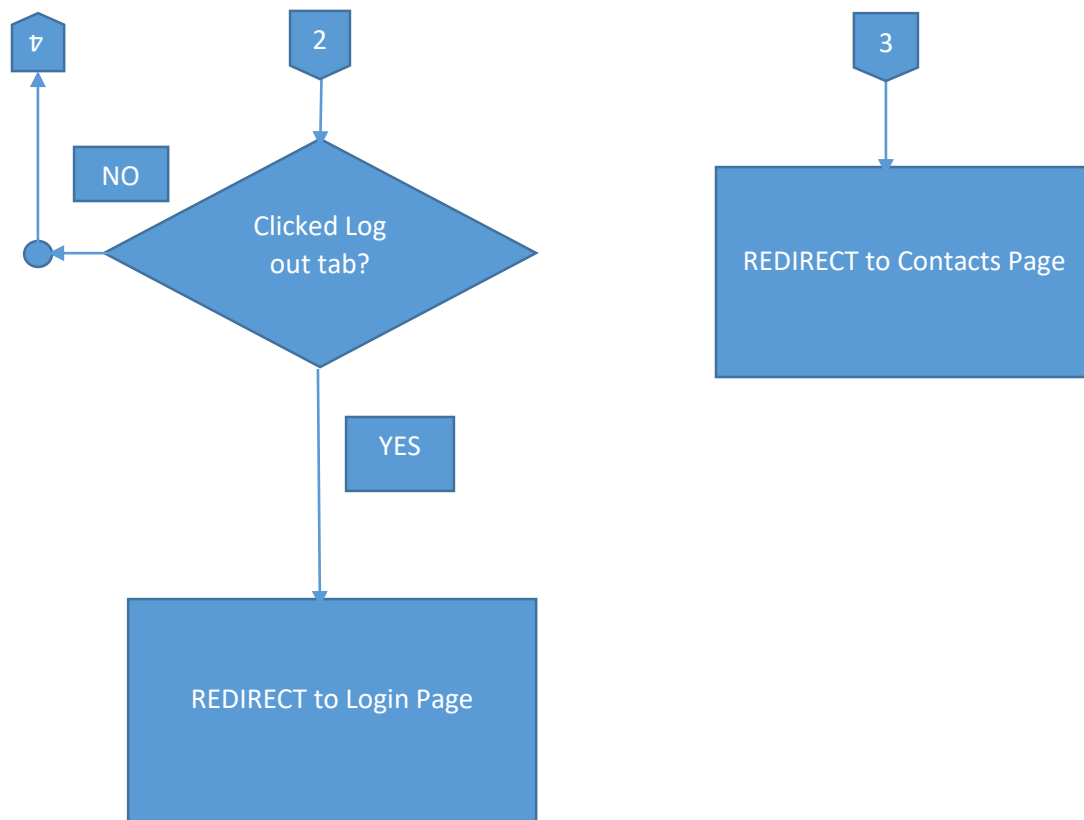


3.5.3 - Account Recovery Page

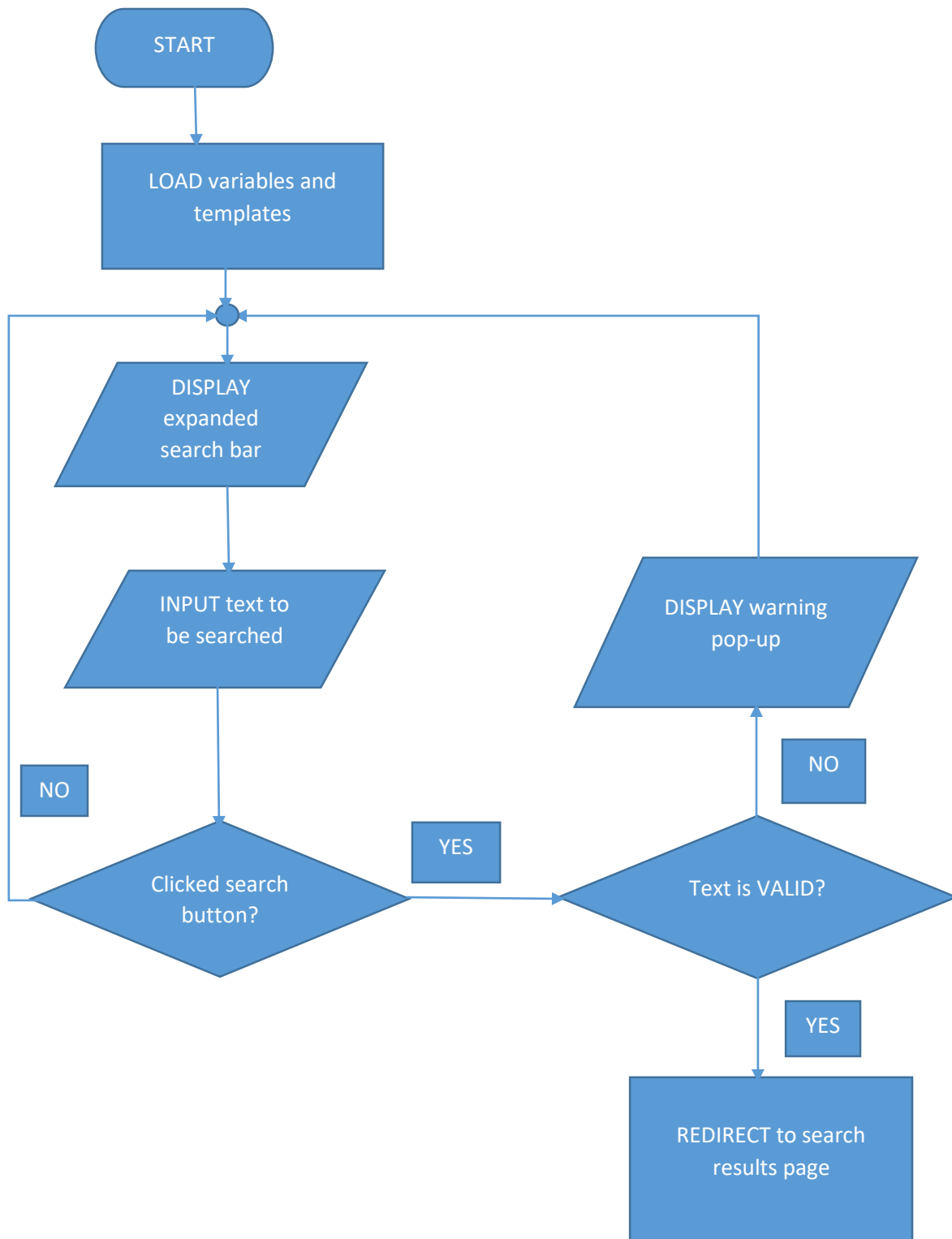


3.5.4 - Home Page

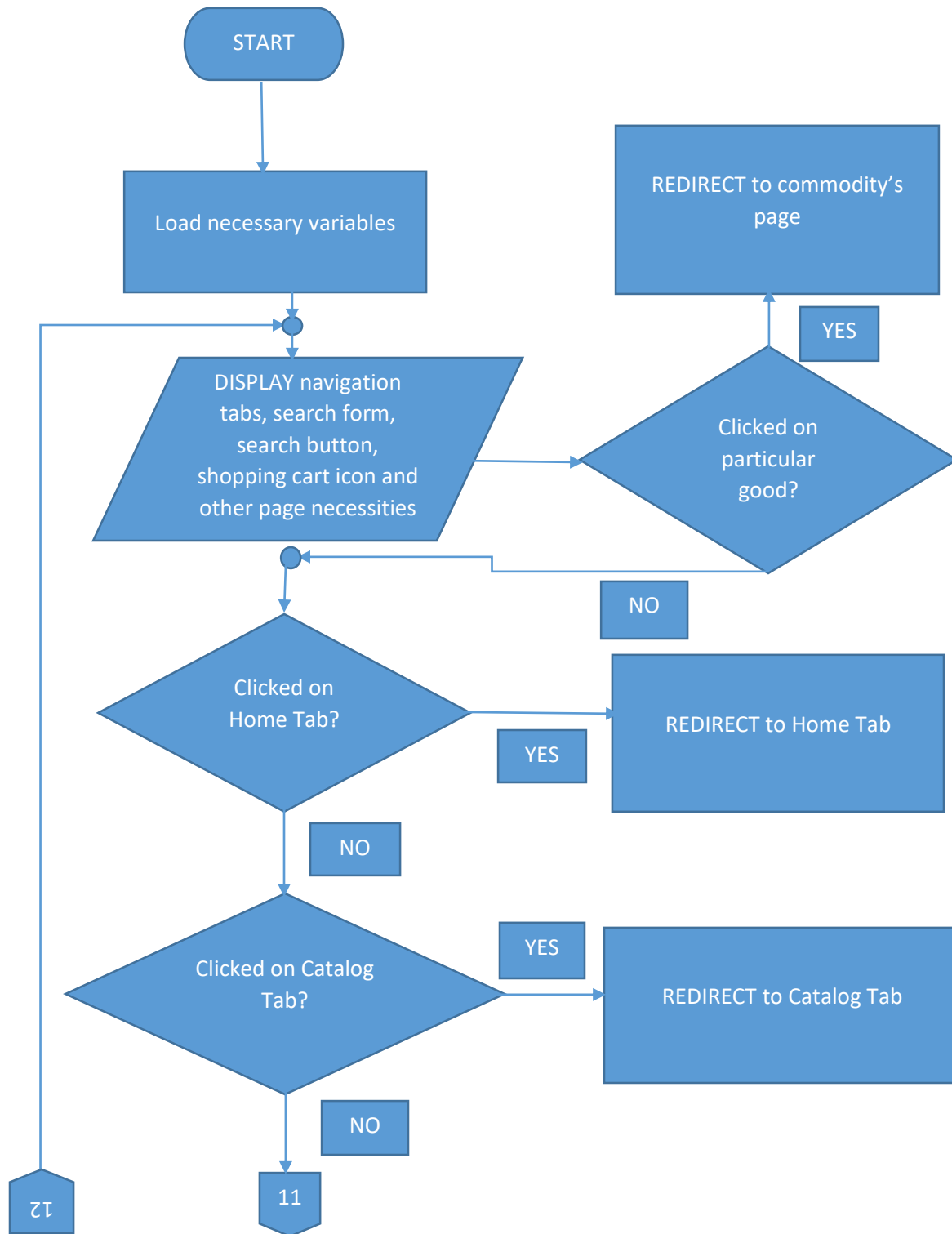


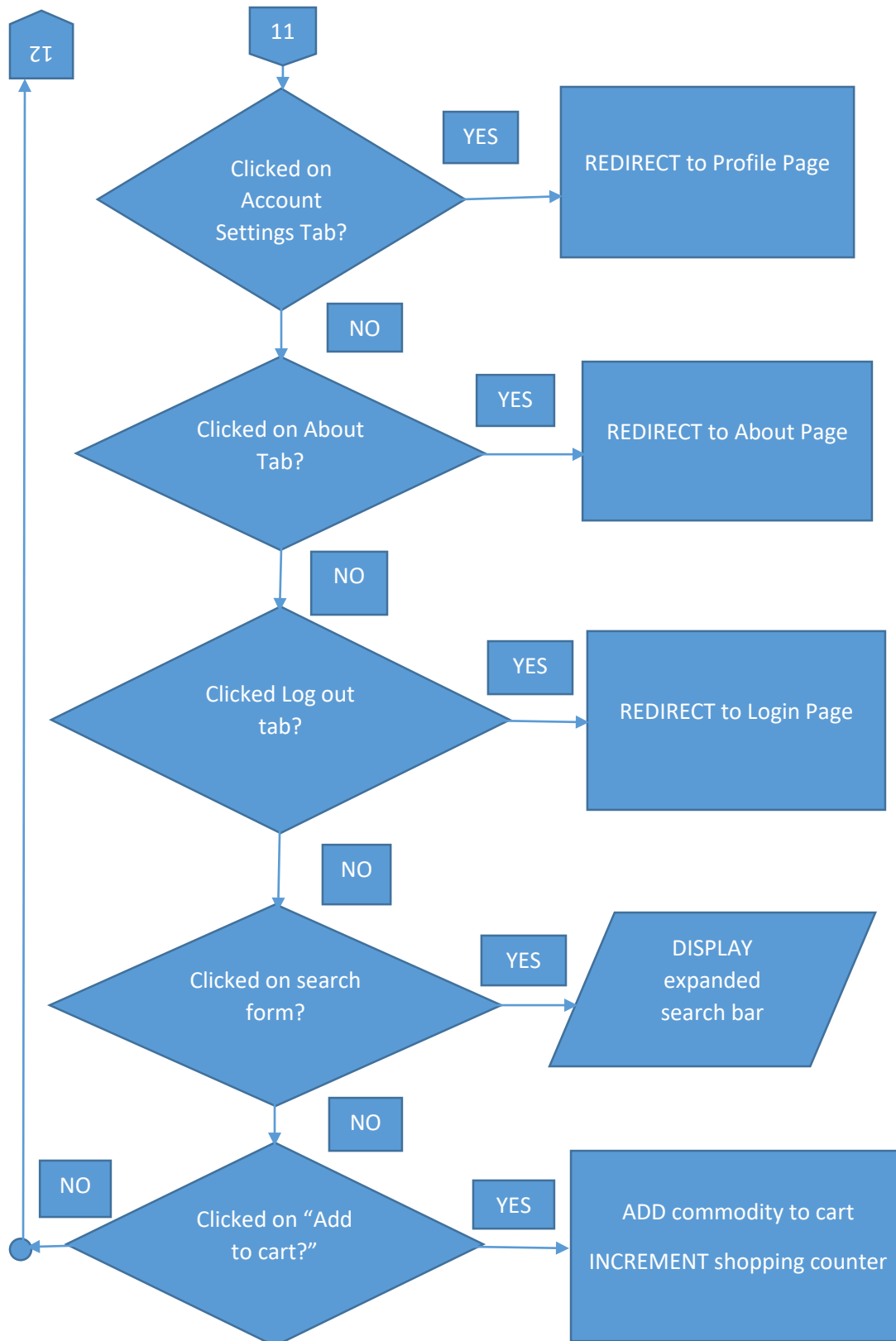


3.5.5 - Search Form

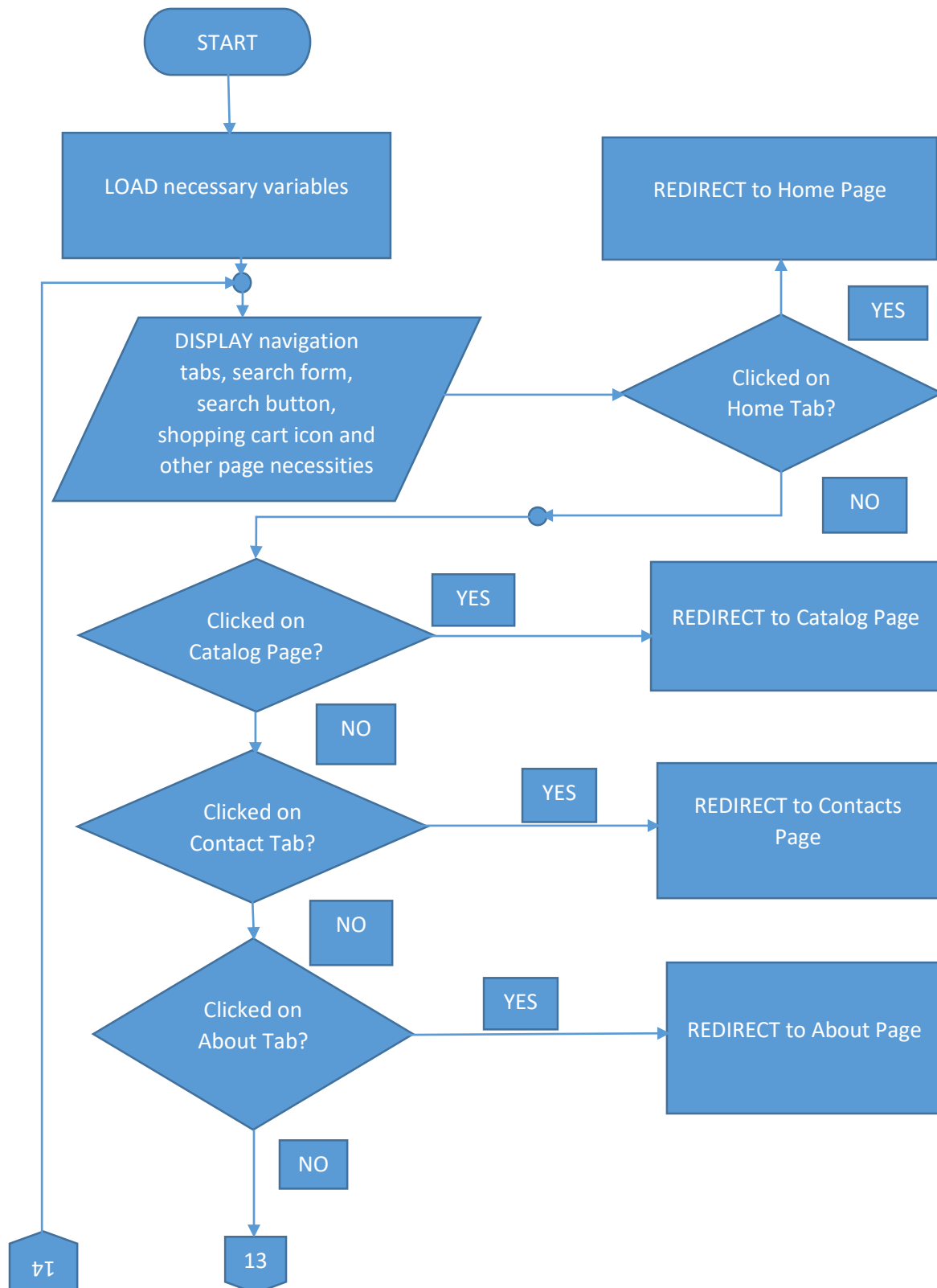


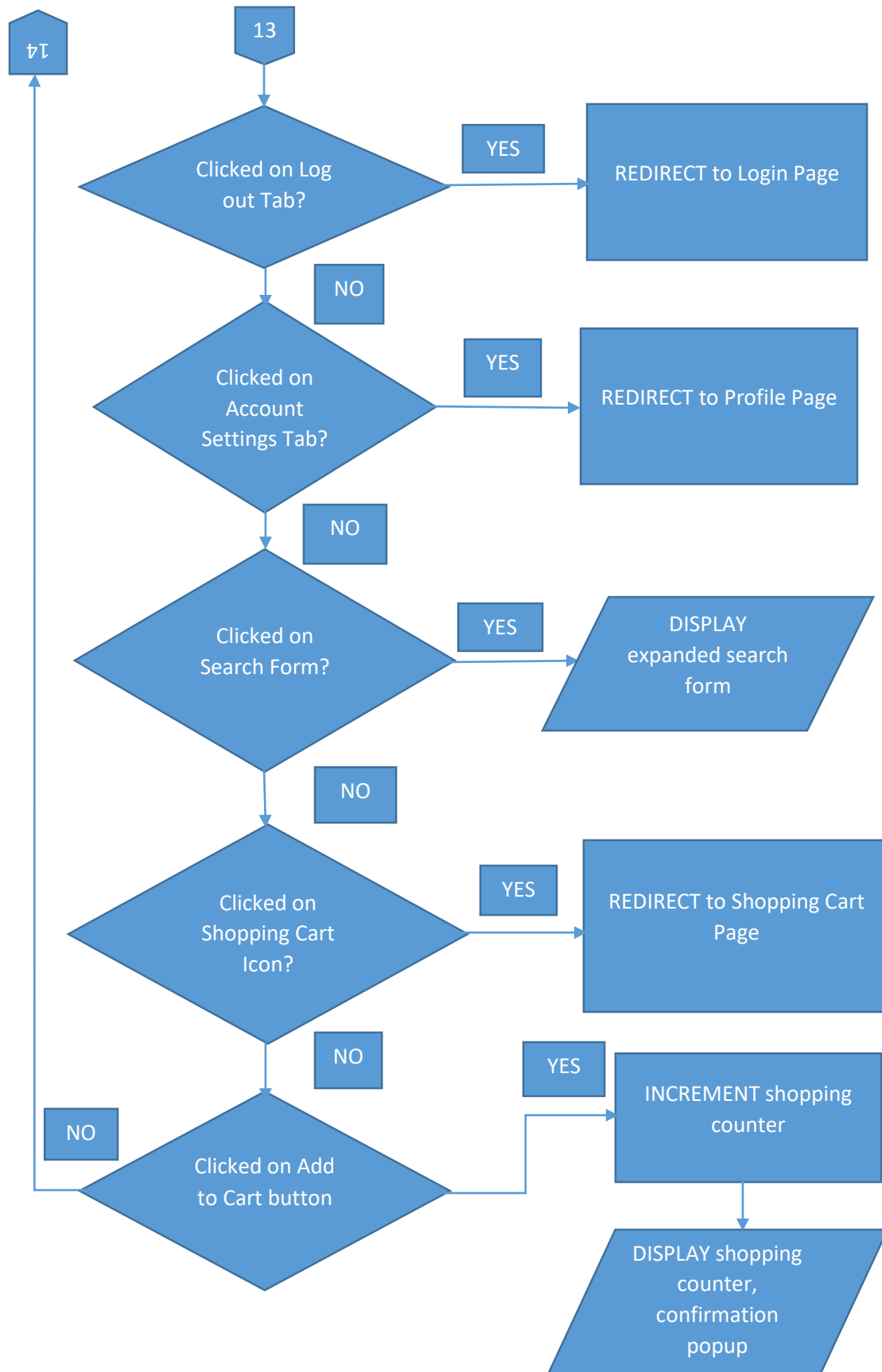
3.5.6 - Category Page



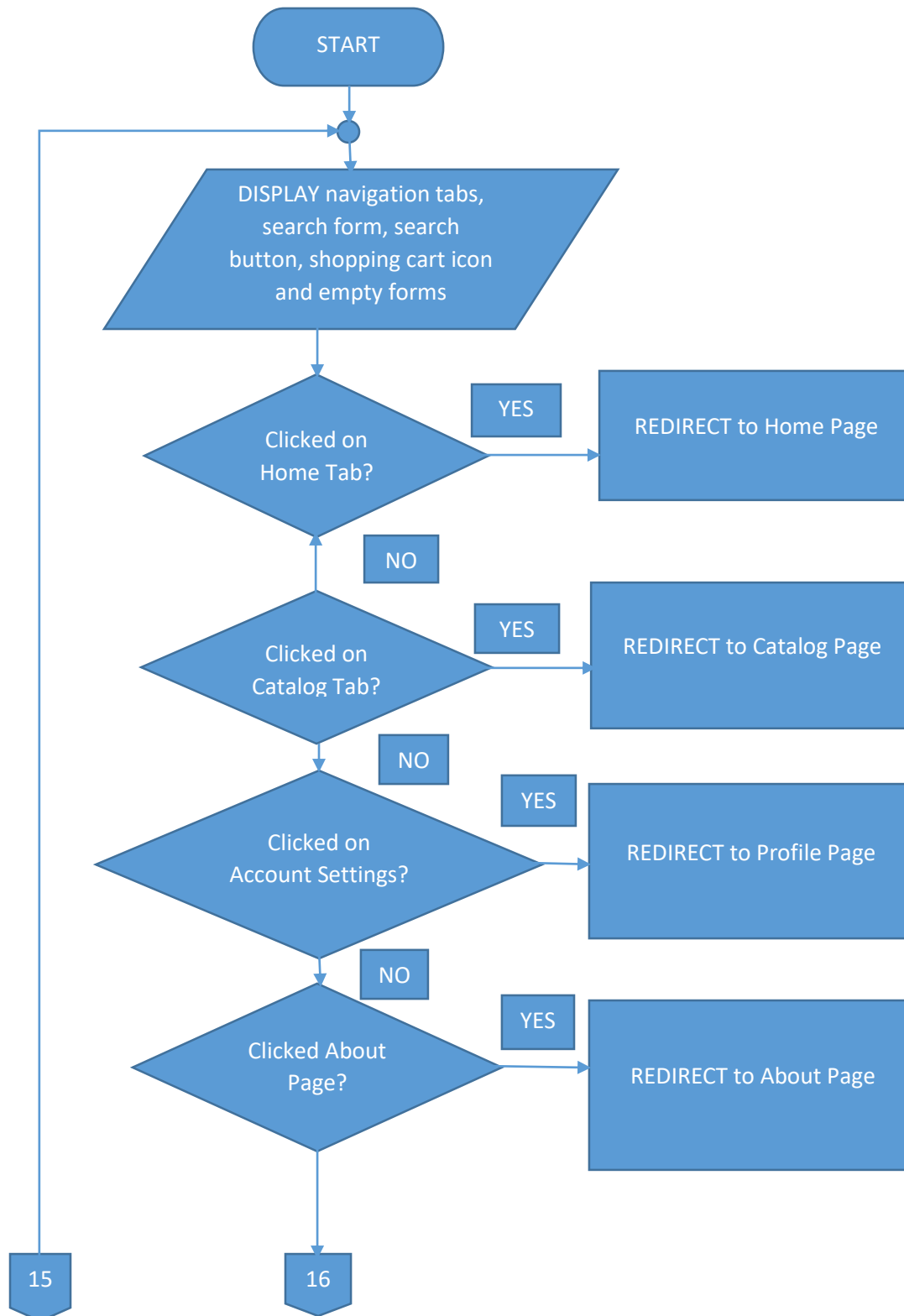


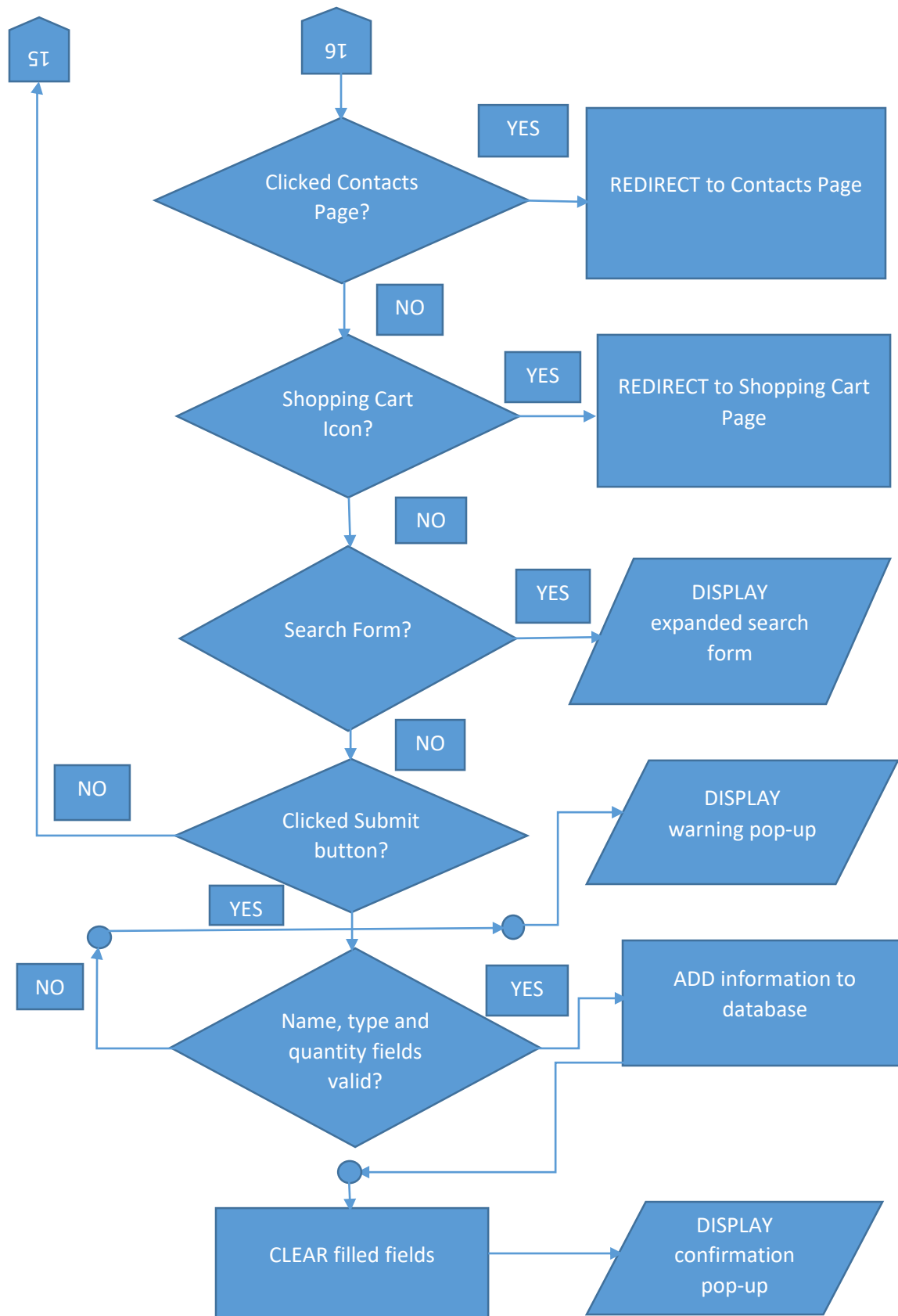
3.5.7 - Commodity Page



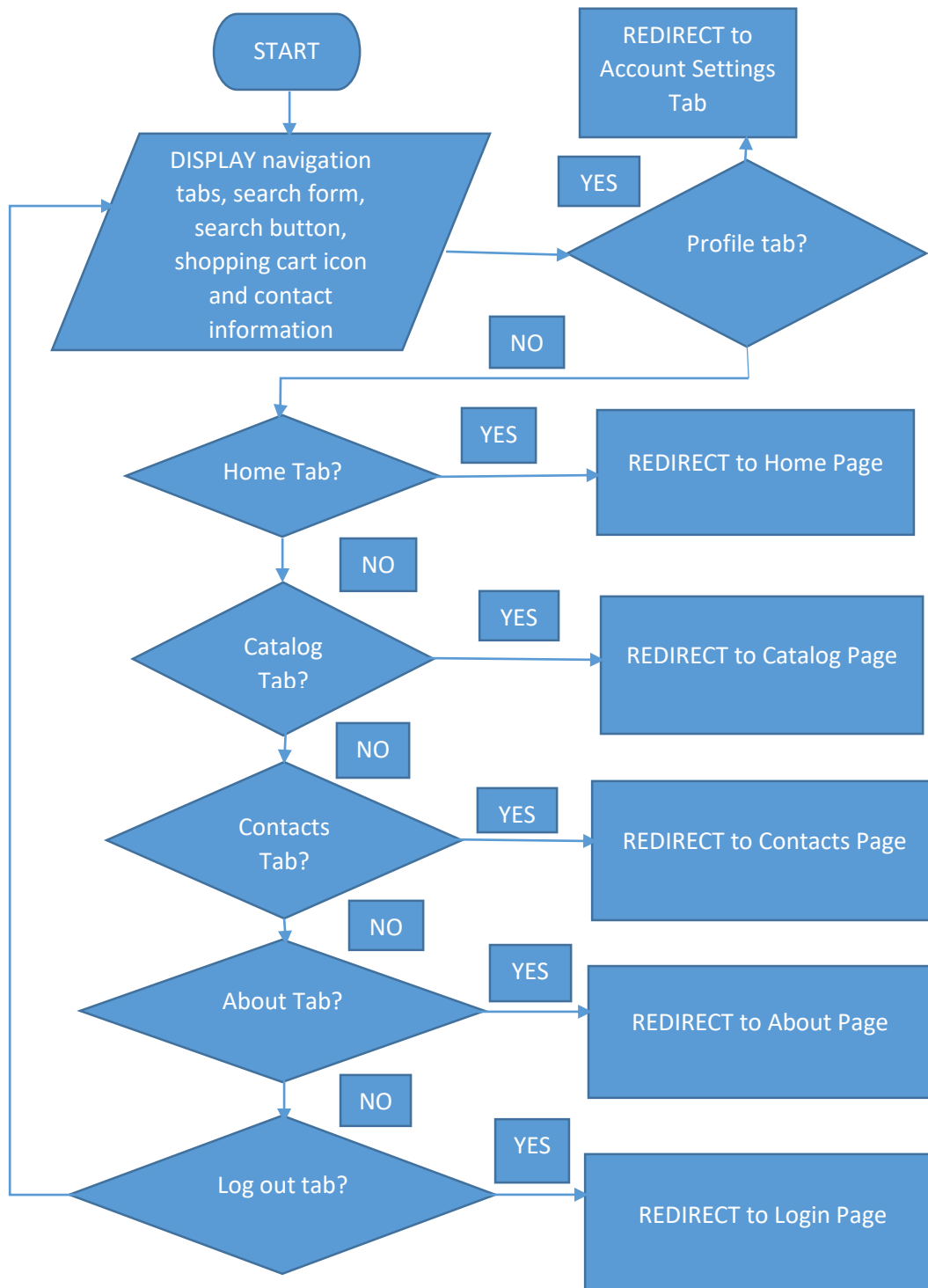


3.5.8 - Add Commodity Page

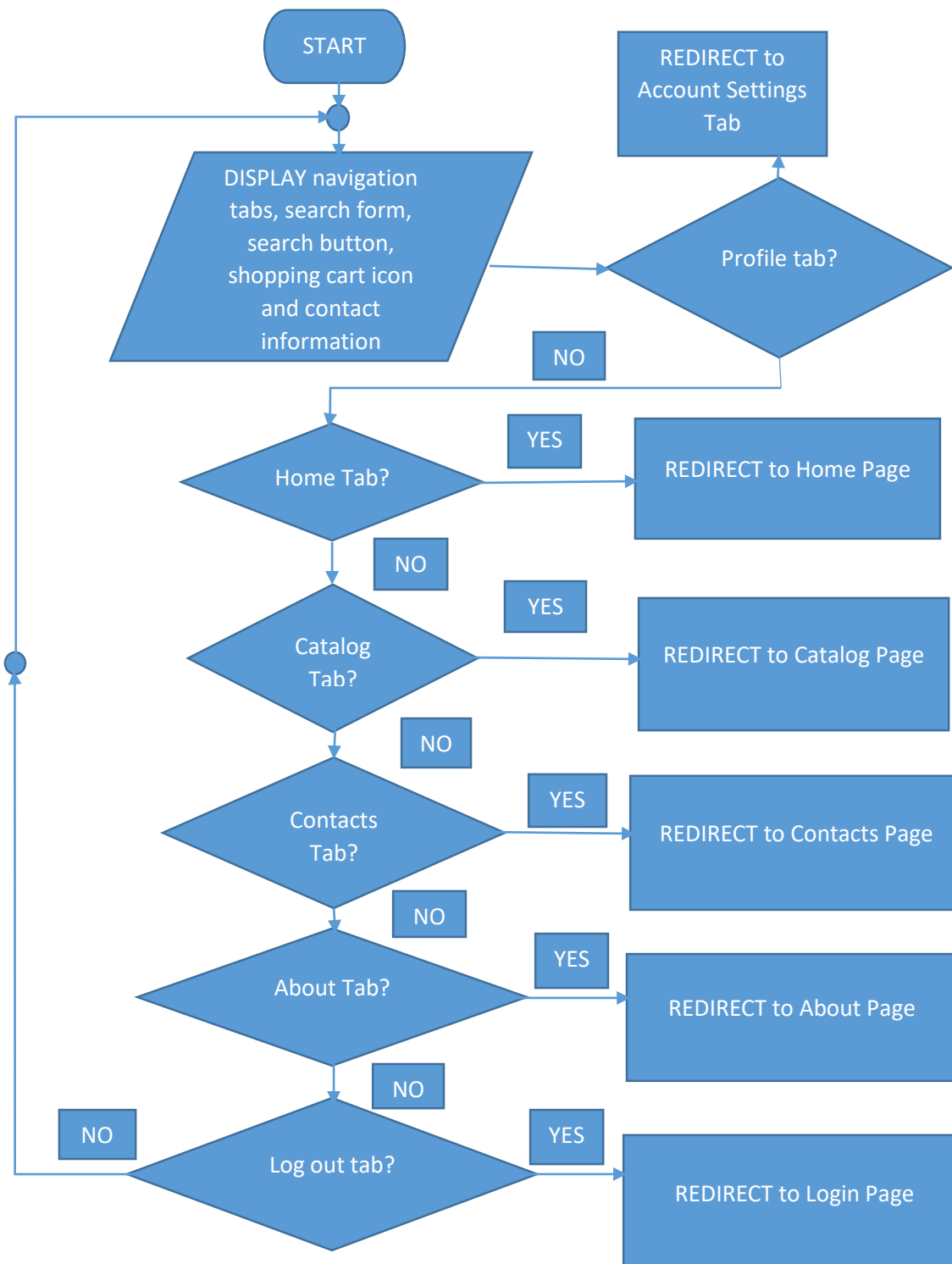




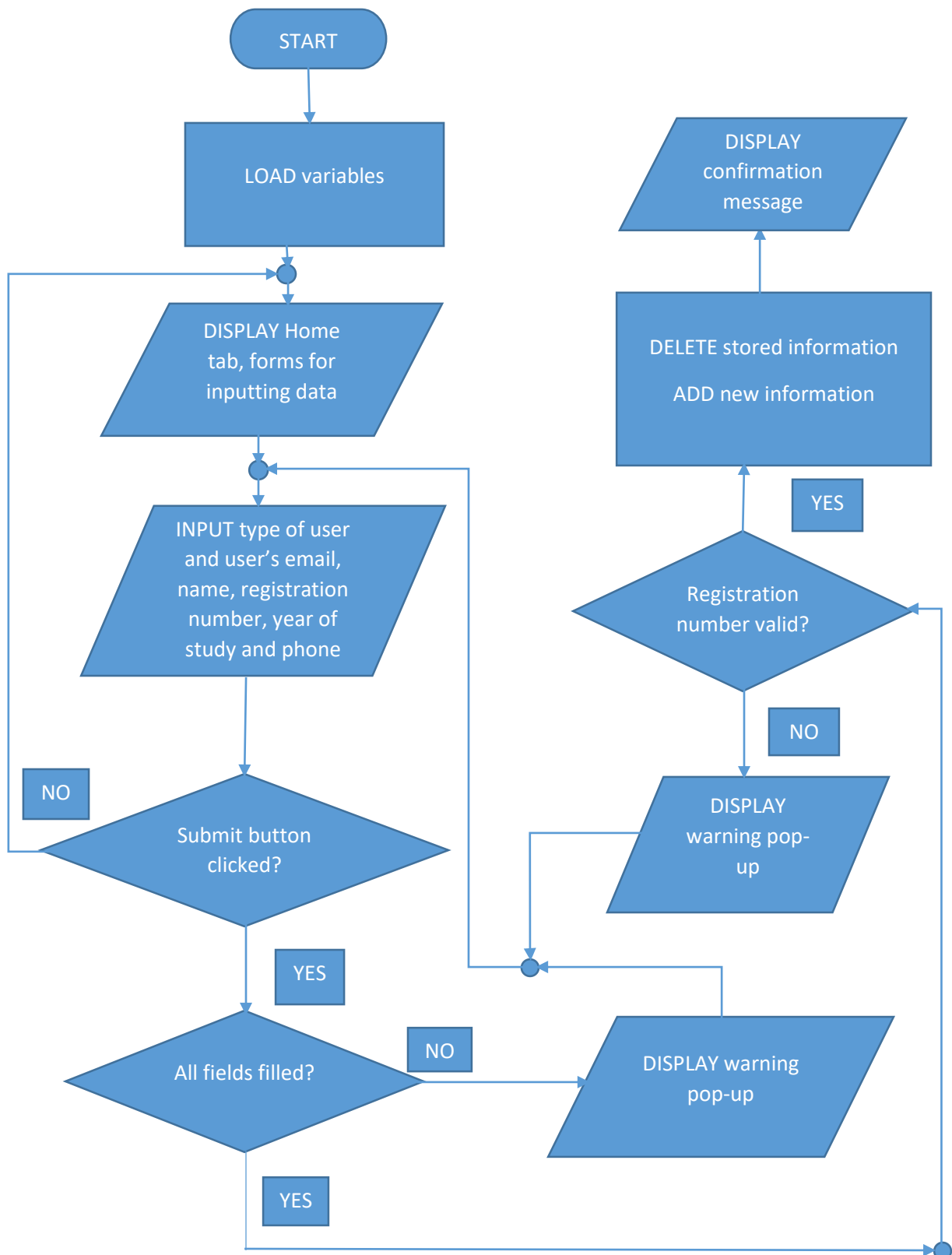
3.5.9 - Contacts Page



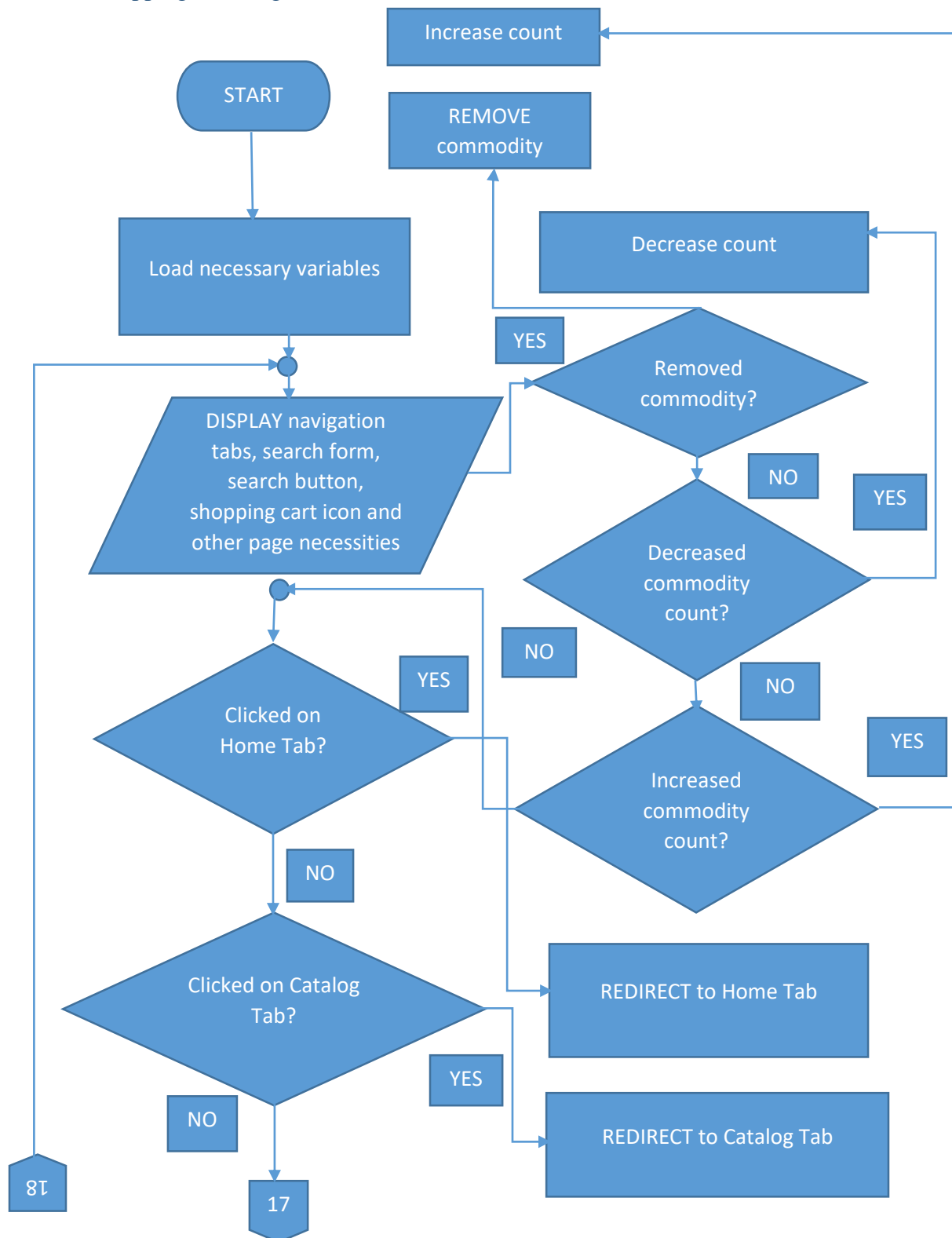
3.5.10 - About Page

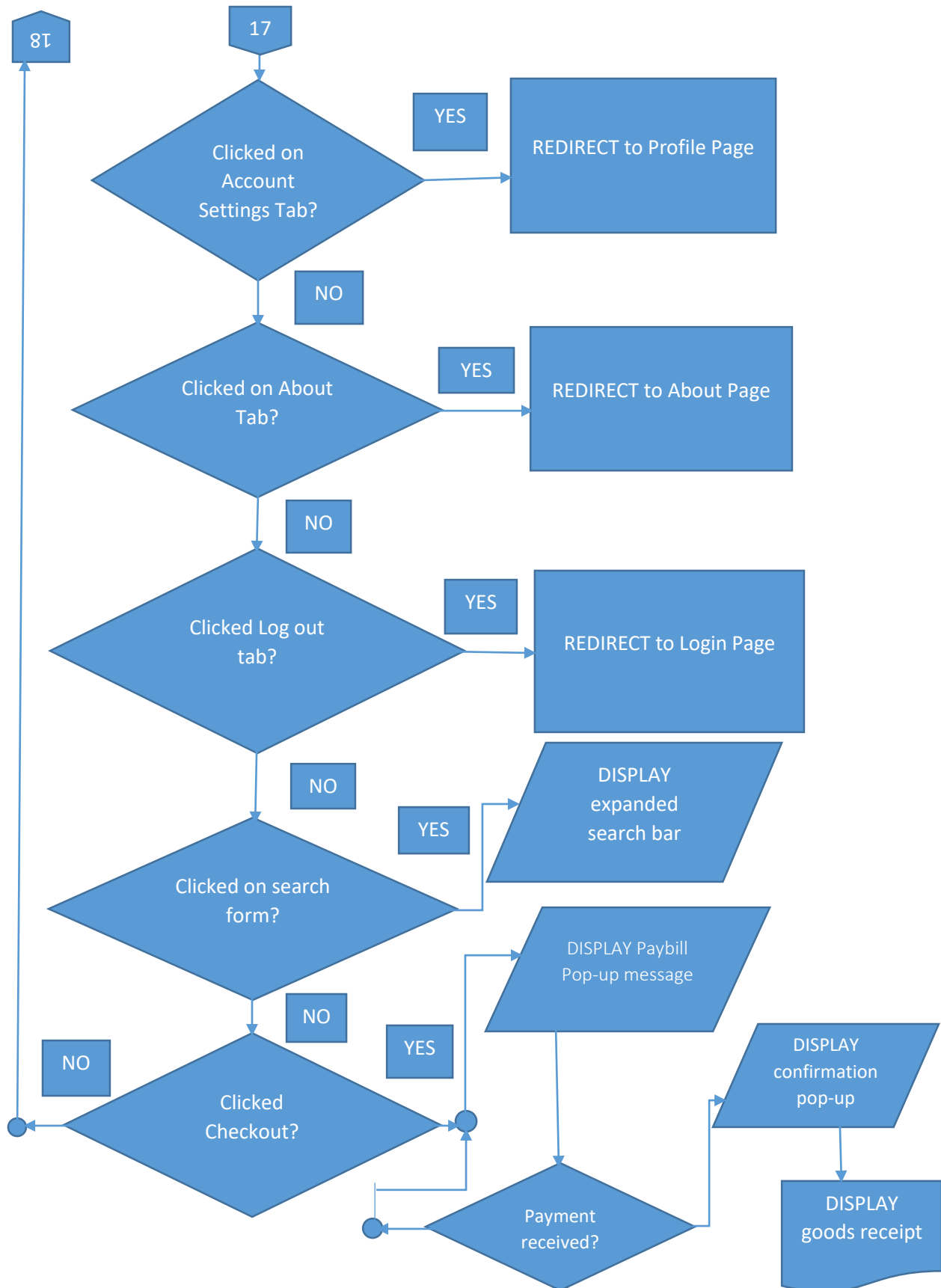


3.5.11 - Profile Page



3.5.12 - Shopping Cart Page





3.6 - Pseudocode

Login Page

BEGIN

 DECLARE emailForm, passwordForm, userType, newUser, warningPopupA, warningPopupB,
 newUserSelector, existingUserSelector, submitButton, forgotPasswordButton

 DISPLAY newUserSelector, existingUserSelector

 IF newUserIsLoggedOut

 DISPLAY link to account registration page

 ELSE

 DISPLAY emailForm, passwordForm

 INPUT email, password

 SELECT userType

 CLICK submitButton

 IF username AND password VALID

 REDIRECT to home page

 ELSE IF username OR password NOT VALID

 DISPLAY warningPopupMessageA

 IF click_forgotPasswordButton==TRUE

 REDIRECT to Account Recovery Page

 ELSE

 DISPLAY emailForm, passwordForm

END

Account Registration Page

BEGIN

 DECLARE userTypeSelector, email, password, regNumber, name, studyYear,
 phoneNumber, allFieldsFilled, emailForm, passwordForm, regNumberForm,
 ageForm, nameForm, studyYearForm, phoneNumberForm, submitButton,
 accountSettingsTab

 DISPLAY userTypeSelector, emailForm, passwordForm, regNumberForm, nameForm,
 studyYearForm, phoneNumberForm, submitForm

```

SELECT userType
INPUT email, password, regNumber, name, studyYear, phoneNumber
CLICK submitButton
IF allFieldsFilled==TRUE AND regNumber VALID
    STORE email, password, regNumber, name, studyYear, phoneNumber in database
    REDIRECT to home page
ELSE IF allFieldsFilled==FALSE
    DISPLAY warningPopupMessageB
ELSE IF regNumber not valid
    DISPLAY warningPopupMessageC
END

```

Account Recovery Page

```

BEGIN
DECLARE recoveryEmailField, submitButton, warningPopupMessage
DISPLAY recoveryEmailField, submitButton
INPUT recoveryEmail
IF click_submitButton==TRUE AND recoveryEmail VALID
    DISPLAY sentConfirmationPopupMessage
ELSE IF click_submitButton==TRUE AND recoveryEmail NOT VALID
    DISPLAY warningPopupMessage
END

```

Home Page

```

BEGIN
DECLARE welcomeMessage, homeTab, catalogTab, contactsTab, aboutTab, logOutTab,
    SearchForm, shoppingCartIcon, click_searchForm, click_homeTab,
    click_catalogTab, click_contactsTab, click_aboutTab, click_logOutTab,
    click_shoppingCartIcon, searchButton, accountSettingsTab
DISPLAY welcomeMessage, homeTab, catalogTab, contactsTab, aboutTab, logOutTab,
    searchForm, shoppingCartIcon, accountRecoveryTab
IF click_searchForm==TRUE

```

```

        DISPLAY expandedSearchForm
    ELSE IF click_homeTab==TRUE
        REDIRECT to home page
    ELSE IF click_catalogTab==TRUE
        REDIRECT to catalog page
    ELSE IF click_contactsTab==TRUE
        REDIRECT to contacts page
    ELSE IF click_aboutTab==TRUE
        REDIRECT to about page
    ELSE IF click_logOutTab==TRUE
        REDIRECT to login page
    ELSE IF click_shoppingCartIcon==TRUE
        REDIRECT to shopping cart page
    ELSE IF click_profileTab==TRUE
        REDIRECT to Account Settings Page
END

```

Search Form

```

BEGIN
    DECLARE expandedSearchForm, searchText, searchFormFilled, warningPopupE, searchButton
    DISPLAY expandedSearchForm, searchButton
    INPUT searchText
    IF click_searchButton==TRUE AND searchText is valid AND searchFormFilled=TRUE
        REDIRECT to search results page
    ELSE IF click_searchButton==TRUE AND searchText is NOT valid
        DISPLAY warningPopupMessageD
END

```

Catalog Page

```

BEGIN
    DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab,
        SearchForm, shoppingCartIcon, click_searchForm, click_homeTab,

```

```

        click_catalogTab, click_contactsTab, click_aboutTab, click_logOutTab,
        click_shoppingCartIcon, electronicsLink, foodstuffsIcon, foodstuffsLink,
        stationeryIcon, stationeryLink, cutleryIcon, cutleryLink, shoppingCartIcon,
        click_searchForm, click_electronicsLink, click_foodstuffsLink,
        click_stationeryLink, click_cutleryLink, addCommodityButton,
        accountSettingsTab

DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        electronicsIcon, electronicsLink, foodstuffsIcon, foodstuffsLink,
        stationeryIcon, stationeryLink, cutleryIcon, cutleryLink, shoppingCartIcon,
        accountSettingsTab

ELSE IF click_electronicsLink==TRUE
    REDIRECT to Electronics Page
ELSE IF click_foodstuffsLink==TRUE
    REDIRECT to Foodstuffs Page
ELSE IF click_stationeryLink==TRUE
    REDIRECT to Stationery Page
ELSE IF click_cutleryLink==TRUE
    REDIRECT to Cutlery Page

END

```

Electronics Commodity Page

```

BEGIN

DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        electronicsName, electronicsPicture, electronicsInfo, addShoppingButton,
        electronicsPrice, addedPopupMessage, shoppingCounter, click_addShoppingButton,
        accountRecoveryTab, shoppingCartIcon, shoppingCounter

DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        electronicsName, electronicsPicture, electronicsInfo, shoppingCartIcon.
        addShoppingButton, electronicsPrice, accountSettingsTab

IF click_homeTab==TRUE
    REDIRECT to home page

```



```

        IF click_addShoppingButton==TRUE,
            DISPLAY addedPopupMessage
            INCREMENT shoppingCounter
    END

```

Foodstuffs Commodity Page

```

BEGIN
    DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        foodstuffsName, foodstuffsPicture, foodstuffsInfo, checkoutTab, addToShoppingLink,
        foodstuffsPrice, addedPopupMessage, shoppingCounter, click_addToShoppingLink,
        accountSettingsTab
    DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        foodstuffsName, foodstuffsPicture, foodstuffsInfo, checkoutTab, addToShoppingLink,
        foodstuffsPrice, accountSettingsTab
    IF click_addToShoppingLink==TRUE
        DISPLAY addedPopUpMessage
        INCREMENT shoppingCounter
    ELSE IF click_homeTab==TRUE
    END

```

Cutlery Commodity Page

```

BEGIN
    DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        cutleryName, cutleryPicture, cutleryInfo, checkoutTab, addToShoppingLink,
        cutleryPrice, addedPopupMessage, shoppingCounter, click_addToShoppingLink,
        accountSettingsTab
    DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm, cutleryName,
        cutleryPicture, cutleryInfo, checkoutTab, addToShoppingLink, cutleryPrice,
        accountSettingsTab
    IF click_addToShoppingLink==TRUE
        DISPLAY addedPopupMessage
    END

```

```

        INCREMENT shoppingCounter
    ELSE IF click_homeTab==TRUE
        REDIRECT to home page
END

```

Stationery Commodity Page

```

BEGIN
    DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        stationeryName, stationeryPicture, stationeryInfo, checkoutTab, addToShoppingLink,
        stationeryPrice, addedPopupMessage, shoppingCounter, click_addToShoppingLink
    DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        stationeryName, stationeryPicture, stationeryPrice, checkoutTab, addToShoppingLink
    IF click_addToShoppingLink==TRUE
        DISPLAY addedPopUpMessage
        INCREMENT shoppingCounter
    ELSE IF click_homeTab==TRUE
        REDIRECT to home page
END

```

Add Commodity Page

```

BEGIN
    DECLARE commodityTypeSelector, commodityNameField, manufactureDateField,
        expiryDateField, quantityField, additionalInfoField, submitButton
    DISPLAY commodityTypeSelector, commodityNameField, manufactureDateField,
        expiryDateField, quantityField, additionalInfoField, submitButton
    INPUT commodityType, commodityName, manufactureDate, expiryDate,
        quantity, additionalInfo
    IF click_submitButton==TRUE AND commodityType AND commodityName AND quantity
        VALID
        ADD type, commodityName, manufactureDate, expiryDate, quantity AND
            additionalInfoField to database
        DISPLAY confirmationPopupMessageA
    END IF
END

```

```
CLEAR typeField, commodityNameField, manufactureDateField, expiryDateField,  
quantityField, additionalInfoField
```

Contacts Page

```
BEGIN
```

```
DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm, contactInfo,  
click_homeTab, click_catalogTab, click_contactsTab, click_aboutTab, click_logOutTab,  
click_searchForm, click_shoppingCartIcon, accountSettingsTab
```

```
DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm, contactInfo,  
accountSettingsTab
```

```
IF click_homeTab==TRUE
```

```
REDIRECT to home page
```

```
END
```

About Page

```
BEGIN
```

```
DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm, aboutInfo,  
click_homeTab, click_catalogTab, click_contactsTab, click_aboutTab, click_logOutTab,  
click_searchForm, click_shoppingCartIcon, accountSettingsTab
```

```
DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm, aboutInfo,  
accountSettingsTab
```

```
IF click_homeTab==TRUE
```

```
REDIRECT to home page
```

```
END
```

Profile Page

```
BEGIN
```

```
DECLARE homeTab, email, password,  
regNumber, name, studyYear, phoneNumber, emailForm, passwordForm,  
regNumberForm, nameForm, studyYearForm, phoneNumberForm,  
submitButton, accountSettingsTab
```

```
DISPLAY email, password, regNumber, name, studyYear, phoneNumber,  
emailForm, passwordForm, submitButton, accountSettingsTab,
```

```

        regNumberForm, nameForm, studyYearForm, phoneNumberForm
INPUT newEmail, newPassword, newRegNumber, newName, newStudyYear,
    newPhoneNumber

IF click_submitButton==TRUE AND newEmail != email
    DELETE email
    STORE newEmail in database
ELSE IF click_submitButton==TRUE AND newPassword != password
    DELETE password
    STORE newPassword
ELSE IF click_submitButton==TRUE AND newRegNumber != regNumber
    DELETE regNumber
    STORE newRegNumber
ELSE IF click_submitButton==TRUE AND newName!= name
    DELETE name
    STORE newName
ELSE IF click_submitButton==TRUE AND newStudyYear!= studyYear
    DELETE studyYear
    STORE newStudyYear
ELSE IF click_submitButton==TRUE AND newPhoneNumber != phoneNumber
    DELETE phoneNumber
    STORE newPhoneNumber

```

END

Shopping Cart Page

BEGIN

```

DECLARE homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
    checkoutLink, click_homeTab, click_catalogTab, click_contactsTab,
    click_aboutTab, click_logOutTab, click_searchForm, click_shoppingCartIcon,
    expandedSearchForm, commodityName, electronicsName, cutleryName,
    stationeryName, cutleryName, commodityCount, electronicsPrice, cutleryPrice,

```

```

        stationeryPrice, foodstuffsPrice, paybillPopUpMessage, confirmationPopUp,
        paymentReceived, goodsReceipt, accountSettingsTab
    DISPLAY homeTab, catalogTab, contactsTab, aboutTab, logOutTab, searchForm,
        checkoutLink, accountSettingTab, shoppingCartIcon, shoppingCounter,
    IF commodityName==electronicsName
        DISPLAY commodityCount, electronicsName, electronicsPrice
    ELSE IF commodityName==cutleryName
        DISPLAY commodityCount, cutleryName, cutleryPrice
    ELSE IF commodityName==stationeryName
        DISPLAY commodityCount, stationeryName, stationeryPrice
    ELSE IF commodityName==foodstuffsName
        DISPLAY commodityCount, foodstuffsName, foodstuffsPrice
    IF click_checkoutLink==TRUE,
        DISPLAY paybillPopUpMessage
        IF paymentReceived==TRUE
            DISPLAY confirmationPopUp, goodsReceipt
END

```

CHAPTER 4 – IMPLEMENTATION

The web application was developed using Python, Flask, HTML and CSS, where Python and Flask were used for the back-end and HTML and CSS were used for the front-end. Some key features of the online discount platform are:

1. Models for creating the database tables
2. Landing page
3. User login page
4. Navigation bar and search bar
5. Profile page
6. Catalog page with four types of commodities
7. Shopping Cart page and checkout
8. About and contacts page

4.1 – Database models

The database models are used to define the names and contents of the tables used to store data in the web application.

There are eight models:

1. User model
2. Commodity model
3. goodsCart model
4. goodsPurchased model
5. Pending model
6. Img model
7. Dashboard model
8. goodsPending Model

4.1.1 – The User Model

```
#-----CLASS User-----
class User(UserMixin, db.Model):
    __tablename__ = 'users'
    user_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), index=True)
    username=db.Column(db.String(64))
    email=db.Column(db.String(128), index=True)
    yearOfStudy= db.Column(db.String(11), index=True)
    regNumber = db.Column(db.String(64), index=True)
    phoneNumber = db.Column(db.Integer, unique=True, index=True)
    password=db.Column(db.String(128), index=True)
    #Password hashing in the User Model
    password_hash = db.Column(db.String(128))
    confirmed = db.Column(db.Boolean, default=False)
    avatar_hash = db.Column(db.String(32))
    role = db.Column(db.String(20), default = "User")
    secQuestion = db.Column(db.Text)
    secAnswer = db.Column(db.Text)
    commodity = db.relationship('Commodity', backref='author', lazy='dynamic')
    goodsAdded = db.relationship('goodsCart', backref='purchaser', lazy='dynamic')
    goodsPurchased = db.relationship('goodsPurchased', backref='purchaserer', lazy='dynamic')
    pendingIssues = db.relationship('Pending', backref='issues', lazy='dynamic')
    idss = db.relationship('Dashboard', backref='iss', lazy='dynamic')
    idss2 = db.relationship('goodsPending', backref='iss2', lazy='dynamic')
```

Figure 4.1.1.1: The User Model

This model stores information about the users and has the ‘user_id’ column as its primary key. The ‘__tablename__’ statement shows the name of the table while the ‘db.Column’ statement defines the columns of the table. Most of its columns have ‘index’ set to True to allow for faster and easy querying. We also have assignments that show relationships with other database models by utilizing the ‘db.relationship’ statement. This means that some table have foreign keys which are extracted from this table. The ‘backref’ attribute shows the backward relationship between the parent and child tables while the ‘lazy’ attribute shows how the items in the parent table will be loaded into this table. In our case, ‘lazy’ means a query statement is needed for them to be loaded. The ‘unique’ attribute indicates that no duplicate values can be stored in a table for that particular column.

4.1.2 – The Commodity Model

```
#-----Commodity Model-----  
class Commodity(db.Model):  
    __tablename__ = 'commodity'  
    id = db.Column(db.Integer, primary_key=True)  
    name = db.Column(db.Text)  
    type = db.Column(db.String(64))  
    manDate = db.Column(db.String(64))  
    expDate = db.Column(db.String(64))  
    quantity = db.Column(db.Integer)  
    oldPrice = db.Column(db.Integer)  
    discPrice = db.Column(db.Integer)  
    percDiscount = db.Column(db.Integer)  
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)  
    inStock = db.Column(db.String(5), default = "Yes")  
    author1 = db.Column(db.Text, db.ForeignKey('users.name'))  
    addDesc = db.Column(db.Text, nullable = True)  
    ids = db.relationship('Img', backref='isses', lazy='dynamic')  
#-----
```

Figure 4.1.2.1: The Commodity Model

This model creates the table that stores the details of the commodities on sale, and has 'id' as its primary key and 'author1' as its foreign key. As discussed before, the foreign key is derived from the Users model, and takes the value of the name of the user who posted the commodity.

4.1.3 – goodsCart Model

```
#-----goodsCart Model-----  
class goodsCart(db.Model):  
    __tablename__ = 'goodsCart'  
    id = db.Column(db.Integer, primary_key=True, unique = True)  
    name = db.Column(db.Text)  
    type = db.Column(db.String(64))  
    manDate = db.Column(db.String(64))  
    expDate = db.Column(db.String(64))  
    reqQuantity = db.Column(db.Integer)  
    price = db.Column(db.Integer)  
    percDiscount = db.Column(db.Integer)  
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)  
    added = db.Column(db.Boolean, default=True)  
    buyer = db.Column(db.Integer, db.ForeignKey('users.name'))  
    seller = db.Column(db.Text)  
#-----
```

Figure 4.1.3.1: The goodsCart Model

This model creates the table that stores the goods that are added into the shopping cart.

4.1.4 – The goodsPurchased Model

```
#-----goodsPurchased Model-----  
class goodsPurchased(db.Model):  
    __tablename__ = 'goodsPurchased'  
    id = db.Column(db.Integer, primary_key=True, unique = True)  
    name = db.Column(db.Text)  
    type = db.Column(db.String(64))  
    manDate = db.Column(db.String(64))  
    expDate = db.Column(db.String(64))  
    reqQuantity = db.Column(db.Integer)  
    price = db.Column(db.Integer)  
    percDiscount = db.Column(db.Integer)  
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)  
    bought = db.Column(db.Boolean, default=True)  
    claimed = db.Column(db.Boolean, default=False)  
    buyer = db.Column(db.Integer, db.ForeignKey('users.name'))  
    seller = db.Column(db.Text)  
#-----
```

Figure 4.1.4.1: The goodsPurchased Model

This model creates the table that stores the goods that are added into the shopping cart.

4.1.5 – The Pending Model

```
#-----Pending Model-----  
class Pending(db.Model):  
    __tablename__ = 'pending'  
    id = db.Column(db.Integer, primary_key=True, unique = True)  
    name = db.Column(db.Text)  
    username = db.Column(db.Text)  
    issue = db.Column(db.Text)  
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)  
    author_id = db.Column(db.Integer, db.ForeignKey('users.user_id'))  
#-----
```

Figure 4.1.5.1: The Pending Model

This model creates the table that stores requests from users, such as requests to become administrators.

4.1.6 – The Img Model

```
#-----Class for uploading pictures-----  
class Img(db.Model):  
    __tablename__ = 'images'  
    id = db.Column(db.Integer, primary_key=True, index=True)  
    img = db.Column(db.Text, nullable=True)  
    mimetype = db.Column(db.Text, nullable = False)  
    commID = db.Column(db.Integer, db.ForeignKey('commodity.id'))  
    #data=db.Column(db.LargeBinary)  
#-----
```

Figure 4.1.6.1: The Img Model

This model creates the table that stores pictures of the goods on sale.

4.1.7 – The Dashboard Model

```
#-----Class for dashboard messages-----  
class Dashboard(db.Model):  
    __tablename__ = 'dashboard'  
    id = db.Column(db.Integer, primary_key=True, index=True)  
    name = db.Column(db.String(50))  
    sender = db.Column(db.String(50), default="Admin")  
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)  
    message = db.Column(db.Text)  
    use_id = db.Column(db.Integer, db.ForeignKey('users.user_id'))  
#-----
```

Figure 4.1.7.1: The Dashboard Model

This model creates the table that stores pictures of the goods on sale

4.1.8 – The goodsPending Model

```
#-----CommodityPending Model-----  
class goodsPending(db.Model):  
    __tablename__ = 'goodspending'  
    id = db.Column(db.Integer, primary_key=True)  
    name = db.Column(db.Text)  
    type = db.Column(db.String(64))  
    manDate = db.Column(db.String(64))  
    expDate = db.Column(db.String(64))  
    quantity = db.Column(db.Integer)  
    oldPrice = db.Column(db.Integer)  
    discPrice = db.Column(db.Integer)  
    percDiscount = db.Column(db.Integer)  
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)  
    inStock = db.Column(db.String(5), default = "Yes")  
    author2 = db.Column(db.Integer, db.ForeignKey('users.name'))  
    addDesc = db.Column(db.Text, nullable = True)  
#-----
```

Figure 4.1.8.1: The goodsPending Model

This model creates the table that stores goods that are to be reviewed by administrators before being placed on sale.

4.2 - Landing page

```
#-----Landing page route-----  
@main.route('/')  
def index():  
    return render_template('index.html')  
#-----
```

Figure 4.2.1: Landing page route

The first line is a comment that is used to show that the code section is for the landing page. The second line is a decorator function that shows the block of code is a route to access a certain page. Its arguments show the physical HTTPS address added at the end of the main 'http://127.0.0.1:5000' address.

The next line then declares the view function that holds the operations that the route will undertake in order to return a certain page or result. In our case, the view function contains a return call that displays the landing page template, called 'index.html'.

```

<p>Welcome to Mwafunzi!</p>
<link rel = "stylesheet" href = '/static/main.css/'/>
{% if current_user.confirmed == True %}
{% if messages == [] %}
  <div class="page-header">
    <h2>Dashboard</h2>
  </div>
  <p><b>No messages have been sent to you yet....</b></p>
  <div class="page-header">
    <h2>Other Options</h2>
  </div>
  <p>
    Would you like to <b>explore our catalog</b> of goods on sale?
    <a href="/catalog/"> Click here </a>
    to go to Catalog Page
  </p>

  <p>
    Would you like to <b>sell</b> to other students?
    <a href="/add-commodity/"> Click here </a>
    to go to add a commodity for sale
  </p>

  <p>
    Would you like to <b>become an administrator</b> or <b>relinquish the role</b>?
    <a href="/become-admin/"> Click here </a>
    to submit a request
  </p>
  <hr>
  <link rel = "stylesheet" href = '/static/main.css/'/>
  
{% else %}
  <div class="page-header">
    <h2>Dashboard</h2>
  </div>

```

Figure 4.2.2: Landing page HTML code

When accessing the web application for the first time, the user is taken to the landing page which contains a greeting followed by the user's username if he/she is registered. For unregistered users or registers users whose session has expired, the landing page displays "Hello, Stranger!" with options to login or register.

Additionally, there is a dashboard that shows the user the messages that an admin has sent, maybe in response to a query or reported claim. The 'Other Options' section includes three choices:

- a) A link to explore the catalog of goods on sale
- b) A link to add a commodity for sale
- c) A link to submit a request to become an admin

4.3 - User login page

```
{% block page_content %}
<div class="page-header">
|   <h1>Login</h1>
</div>
<div class="col-md-4">
|   {{ wtf.quick_form(form) }}
</div>
<!-------link to the registration page----->
<p>
|   New user?
|   <a href="{{ url_for('auth.register') }}">
|   Click here to register
|   </a>
</p>
<p>
|   Forgot Password?
|   <a href="{{ url_for('auth.forgot') }}">
|   Click here
|   </a>
```

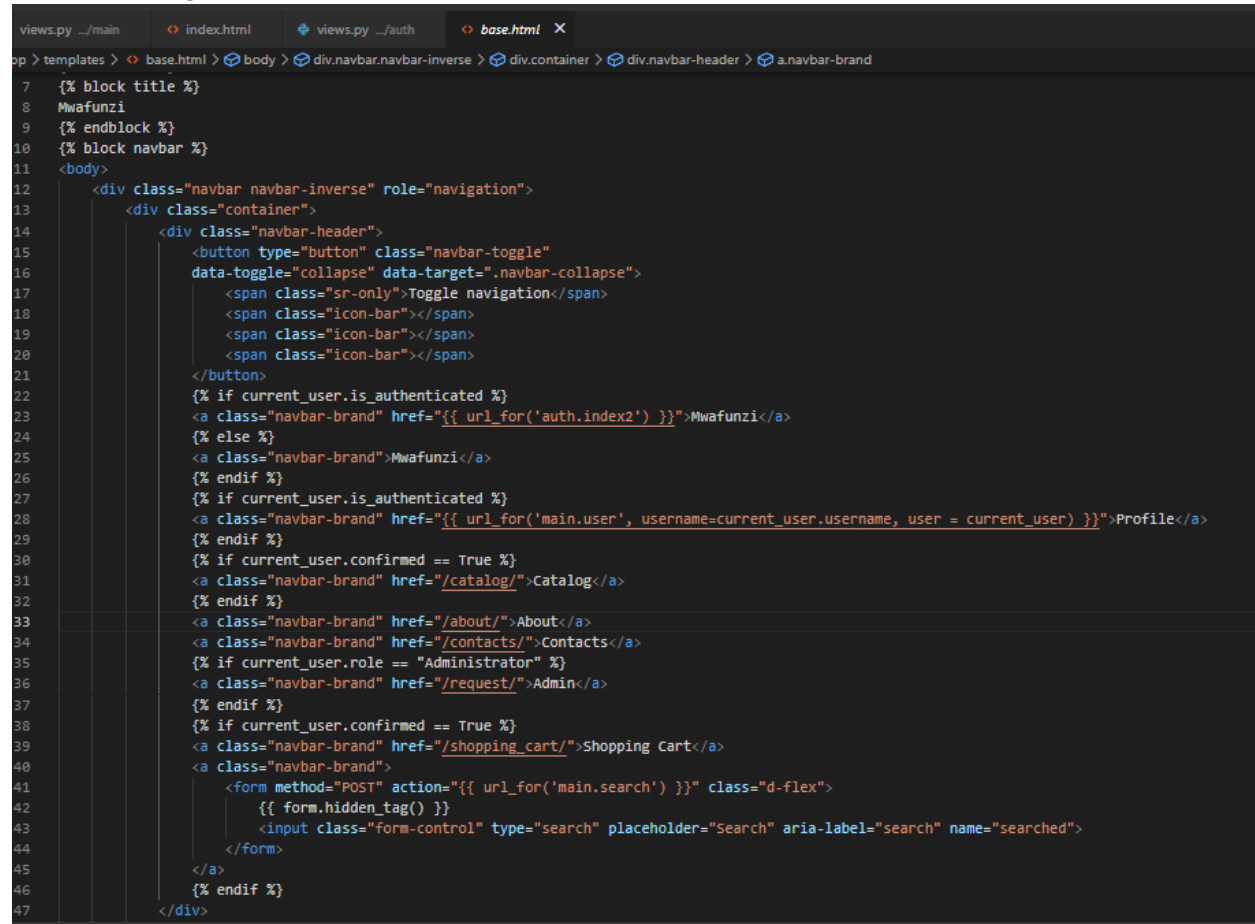
Figure 4.3.1: Login page HTML

The headers are under the `<h1>` tag and a form is quickly generated using the `'wtf.quick_form'` function from the Flask-WTF package. The page therefore displays email and password fields and options to register or reset password if the user needs to. A checkbox is also included if the user wants to be remembered beyond the session.

Additionally, there are options on the side for registration, if the user is new, and account recovery in case the user forgets his/her password. The link under each of these options have an `'url_for'` function that passes operation to the respective view functions specified within its arguments.

4.4 – Navigation bar and search bar

4.4.1 – Navigation Bar



```
7  {% block title %}
8  Mwafunzi
9  {% endblock %}
10 {% block navbar %}
11 <body>
12   <div class="navbar navbar-inverse" role="navigation">
13     <div class="container">
14       <div class="navbar-header">
15         <button type="button" class="navbar-toggle"
16           data-toggle="collapse" data-target=".navbar-collapse">
17           <span class="sr-only">Toggle navigation</span>
18           <span class="icon-bar"></span>
19           <span class="icon-bar"></span>
20           <span class="icon-bar"></span>
21         </button>
22         {% if current_user.is_authenticated %}
23         <a class="navbar-brand" href="{{ url_for('auth.index2') }}">Mwafunzi</a>
24         {% else %}
25         <a class="navbar-brand">Mwafunzi</a>
26         {% endif %}
27         {% if current_user.is_authenticated %}
28         <a class="navbar-brand" href="{{ url_for('main.user', username=current_user.username, user = current_user) }}">Profile</a>
29         {% endif %}
30         {% if current_user.confirmed == True %}
31         <a class="navbar-brand" href="/catalog/">Catalog</a>
32         {% endif %}
33         <a class="navbar-brand" href="/about/">About</a>
34         <a class="navbar-brand" href="/contacts/">Contacts</a>
35         {% if current_user.role == "Administrator" %}
36         <a class="navbar-brand" href="/request/">Admin</a>
37         {% endif %}
38         {% if current_user.confirmed == True %}
39         <a class="navbar-brand" href="/shopping_cart/">Shopping Cart</a>
40         <a class="navbar-brand">
41           <form method="POST" action="{{ url_for('main.search') }}" class="d-flex">
42             {{ form.hidden_tag() }}
43             <input class="form-control" type="search" placeholder="Search" aria-label="search" name="searched">
44           </form>
45         </a>
46         {% endif %}
47     </div>
```

Figure 4.4.1.1: HTML code for the navigation bar

All pages have a navigation bar and a search bar for accessing pages and commodities quickly. This is achieved by using a base HTML template, conveniently named ‘base.html’ from which other templates inherit blocks of code to prevent repetition. This technique is called templating.

If the user is an administrator, an ‘Admin’ tab will be visible on the navigation bar which, when clicked, redirects to a page that has requests and tasks from the users that need to be carried out. If the user is not confirmed, only the Home, About and Contacts pages are visible, while the Home page is not clickable.

4.4.2 – Search bar

```
<a class="navbar-brand">
  <form method="POST" action="{{ url_for('main.search') }}" class="d-flex">
    {{ form.hidden_tag() }}
    <input class="form-control" type="search" placeholder="Search"
      aria-label="search" name="searched">
  </form>
</a>
```

Figure 4.4.2.1: HTML code for the search bar

The <a> tag is used to enclose the form since the search bar itself is a link between pages and a search route. The <form> tag defines the form field for the search bar, and has the method attribute 'POST' which sends data to a specific location or function. The action attribute defines where the search contents are sent and processed, and in our case, they are sent to the 'search' function in the main folder.

```
#-----Search box-----
@main.route('/search/', methods = ['GET','POST'])
def search():
    form = SearchForm()
    if form.validate_on_submit():
        bruh = form.searched.data
        posts = Commodity.query.filter(Commodity.name.like('%'+ bruh +%')).order_by(
            Commodity.timestamp).all()
        usersz = User.query.filter(
            User.name.like('%'+ bruh +%')).all()
        user = User.query.filter_by(name=current_user.name).first()
        return render_template('search.html', searched = bruh,
                               form=form, posts=posts, user=user, users=usersz)
    flash('No such commodity on sale!')
    return render_template('index.html')
#-----
```

Figure 4.4.2.2: Route for processing searched letters

Here, in addition to the ‘POST’ method, we utilize the ‘GET’ method for fetching data, maybe from a database. The search form which was created in the file “forms.py” is imported and stored in a variable ‘form’. When the form contents are validated to be present (ensuring the form has at least one letter in it), database-querying statements are used to check if the letter or combination of letters is present in the name of an item in the database. This is done by using the regular expression ‘%’.

Since the search form is also to be used by administrators to search for users, a database-querying statement is written in the same way as before using the expression ‘%’. The results of the queries are stored in variables then passed into the ‘search.html’ template to render the search results.

```

{% if posts==[] %}
<p>
    No such item is on sale.
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>
</p>
{% elif user.role == 'Administrator' %}
{% for post in posts %}
<div class="search-item">
    
    <div class="final">
        <div class="profile-thumbnail">
            
        </div>
        <a class="commodity-seller" href="{{ url_for('.user', username=post.author.username) }}">
            <b>Seller :</b> {{ post.author1 }}</a>
        <div class="commodity-name"><b>Commodity name :</b> {{ post.name }}</div>
        <div class="commodity-type"><b>Commodity Type :</b> {{ post.type }}</div>
        <div class="commodity-quantity"><b>Quantity on sale :</b> {{ post.quantity }}</div>
        <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b>
            {{ post.oldPrice }}</div>
        <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings :</b>
            {{ post.discPrice }}</div>
        <div class="commodity-discount"><b>Percentage Discount :</b>
            {{ post.percDiscount }}%</div>
        <div class="commodity-stock"><b>In Stock :</b> {{ post.inStock }}</div>

        <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=post.id) }}">
            Add to Cart
        </a>
        <a class="btn btn-default" href="{{ url_for('.view all', id=post.id) }}">

```

Figure 4.4.2.3: A sample of the search page HTML template

If the user is an administrator, the search page displays the matching commodities as well as the users. The pictures of the commodities are loaded using the src attribute in the tag, whereby the ID of the commodity post is passed using a 'url_for' function to display the image with the corresponding ID. The width and height attributes resize the image displayed while the 'id' and 'class' attributes are used to enable CSS editing.


```

#left-box{
    float:left;
}
#right-box{
    float:right;
}
.final{
    display: block;
    margin-top: 50px;
    margin-left: 250px;
}
.final2{
    display: block;
    margin-top: 50px;
    margin-left: 50px;
}
.centre-text{
    margin-left : 100px;
    margin-right : auto;
}

```

Figure 4.4.2.4: CSS styling applied to the commodity lists in the search page.

CSS styling is used to alter the physical appearance of code in the templates when they are rendered, such as size, distance from margins and color. The ‘left-box’ styling makes the block of code to be pushed towards the left margin while the ‘right-box’ styling does the opposite. The ‘final2’ styling displays the code as an organized block which is distanced from the top and left margins by 50px.

4.5 – Profile page

```

#-----Profile page route with commodity posts-----
@main.route('/user/<username>')
def user(username):
    user = User.query.filter_by(username=username).first()
    if user is None:
        abort(404)
    commodities = user.commodity.order_by(Commodity.timestamp.desc()).all()
    user1_role = current_user.role
    bout = goodsPurchased.query.filter_by(buyer = current_user.name).all()
    bout1 = goodsPending.query.filter_by(author2 = current_user.name).all()
    return render_template('user.html', user=user, commodities=commodities,
        role = user1_role, bout = bout, bout1 = bout1)
#-----

```

Figure 4.5.2: Profile page route

The view function for the profile page uses a database-querying statements to retrieve records for commodities sold by the user, commodities purchased by the user and the commodities pending review by admins respectively. If the user does not exist, an error page is displayed. Afterwards, the profile page HTML template, 'user.html', is then rendered and the records passed as arguments to the 'render_template' function.

```
{% block page_content %}
<div>
  
  <div class="profile-header">
    <h1>{{ user.username }}</h1>
  </div>
  {% if user.username or user.location %}
  <p>
    <ol><b>Name :</b> {% if user.name %}{{ user.name }}{% endif %}</ol>
    <ol><b>Email :</b> {% if user.email %}{{ user.email }}{% endif %}</ol>
    <ol><b>Year of Study :</b> {% if user.yearOfStudy %}{{ user.yearOfStudy }}{% endif %}</ol>
    <ol><b>Registration Number :</b> {% if user.regNumber %}{{ user.regNumber }}{% endif %}</ol>
    <ol><b>Phone Number :</b> {% if user.phoneNumber %}{{ user.phoneNumber }}{% endif %}</ol>
  </p>
</div>
</div>
```

Figure 4.5.3: HTML code that displays the user's general details

The <p> tag has been used to display the commodity details as a paragraph while the tag is used to display each line of information as an unordered list. The header enclosed by the <h1> tag displays the username of the user and the if-statement checks if the user details exist by checking if either the user's username or location is present in the database. The tag is used to bolden the line headers. To personalize the profile page, a gravatar icon is loaded and displayed for the user using the tag and 'src' attribute.

[4.6 – Catalog page](#)

[4.6.1 – Catalog Page commodity type routes](#)

```
@main.route('/catalog/')
def catalog():
    return render_template('catalog.html')

@main.route('/catalog/electronics/')
def commoditypg1():
    commodities = Commodity.query.filter_by(type='Electronics').order_by(
        Commodity.timestamp.desc()).all()
    if commodities is None:
        return render_template('foodstuffs.html', commodities = None)
    return render_template('electronics.html', commodities = commodities)

@main.route('/catalog/cutlery/')
def commoditypg2():
    commodities = Commodity.query.filter_by(type='Cutlery').order_by(
        Commodity.timestamp.desc()).all()
    if commodities is None:
        return render_template('foodstuffs.html', commodities = None)
    return render_template('cutlery.html', commodities = commodities)
```

[Figure 4.6.1.1: Routes for the catalog page and respective commodity pages](#)

The routes contain database-querying statements that return a list of commodities of the type requested, and if statements that check if the type of commodities queried exist. The routes then render the respective templates after the if statements are run. The route for each commodity is similar except each queries for a different commodity in the database.

[4.6.2 – Commodity Page HTML example](#)

```
<div class="page-header">
    <h3>Electronic commodities on sale</h3>
</div>
<link rel="stylesheet" href="/static/main.css"/>
<ul class="commodities">
    {% if commodities == [] %}
    <p> Sorry! Currently, there are no electronics on sale</p>
    <hr>
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>

    {% elif current_user.role == 'Administrator' %}
    <div class="banner">
        
    </div>
```

[Figure 4.6.2.1: HTML code for the Electronics Page](#)

For example, the HTML template for Electronics page has a header, and commodities displayed if they exist in the database. If none exist, a message is displayed with a default picture and a message communicating the same. If the user is an administrator, the seller name will appear as a link to that user's profile page to allow the administrator to carry out administrator tasks if needed.

```
<div class="commodity-name"><b>Commodity name :</b> {{ commodity.name }}</div>
<div class="commodity-type"><b>Commodity Type :</b> {{ commodity.type }}</div>
<div class="commodity-quantity"><b>Quantity on sale :</b>
|   {{ commodity.quantity }}</div>
<div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b>
|   {{ commodity.oldPrice }}</div>
<div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings :</b>
|   {{ commodity.discPrice }}</div>
<div class="commodity-discount"><b>Percentage Discount :</b>
|   {{ commodity.percDiscount }}%</div>
<div class="commodity-stock"><b>In Stock :</b>
|   {{ commodity.inStock }}</div>

<a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity.id) }}">
  Add to Cart</a>
</a>
<a class="btn btn-default" href="{{ url_for('.view_all', id=commodity.id) }}"
  target="_blank" rel="noopener noreferrer">
  See full picture</a>
```

Figure 4.6.2.2: The HTML code for displaying the commodity details

Each individual commodity detail is enclosed in a <div> tag to separate them and are displayed by using a dot operator that accesses the desired attribute on the specific commodity type. Each commodity post has three buttons:

- a) One button for adding the commodity to the shopping cart
- b) Another viewing the full picture in a separate tab
- c) Another for reporting the commodity if necessary

The 'target' and 'rel' attributes are included to make the page for viewing the full picture to open in a new tab.

4.6.3 – Viewing the full picture of the commodity

```
#-----view full picture-----  
@main.route('/view-all/<id>')  
def view_all(id):  
    img = Img.query.filter_by(commID=id).first()  
    comm = Commodity.query.filter_by(id=id).first()  
    if not img:  
        return 'No image with that ID', 404  
    return Response(img.img, mimetype=img.mimetype)  
#-----
```

Figure 4.6.3.1: Route for viewing the full picture of a commodity

The first database-querying statement is used to retrieve the image from the database. The if-statement deals with the possibility that no image with that ID exists, and returns an error message. Otherwise, the function 'Response' function loads the image and displays it in the browser.

4.6.4 – Reporting a commodity

```
#-----Reporting a posted commodity-----  
@main.route('/report/<id>', methods = ['GET','POST'])  
def report(id):  
    form = GeneralForm()  
    if form.validate_on_submit():  
        issues = Pending(name = current_user.name,  
                        username = current_user.username,  
                        issue = form.issue.data)  
        db.session.add(issues)  
        db.session.commit()  
        flash('Your report has been submitted!')  
        return redirect(url_for('auth.index2'))  
    flash('Please be descriptive! State the name of the seller, the commodity and the is.  
    return render_template('genreq.html', form = form)  
#-----
```

Figure 4.6.4.1: View function for reporting a commodity

The 'Report' button takes the user to a form page to fill in the complaint. The report form is loaded and a message is flashed alerting the user to be descriptive. The 'form.validate_on_submit()' is used to check if all form fields have been filled. Once done, the information entered in the forms is stored in the respective database table, and a message is flashed to the user confirming the same.

4.7 – Shopping cart and checkout

```
#-----Route for adding to shopping cart-----
@main.route('/add-cart/<id>', methods=['GET', 'POST'])
def add_to_cart(id):
    commodities=Commodity.query.filter_by(id=id).first()
    form = QuantityForm()
    if form.validate_on_submit():
        req = goodsCart(id = commodities.id,
            name = commodities.name,
            type = commodities.type,
            manDate = commodities.manDate,
            expDate = commodities.expDate,
            reqQuantity = form.goodQuantity.data,
            price = (form.goodQuantity.data)*commodities.discPrice,
            percDiscount = commodities.percDiscount,
            timestamp = commodities.timestamp,
            buyer = current_user.name,
            seller = commodities.author1)
        if form.goodQuantity.data <= commodities.quantity and form.goodQuantity.data>0:
            db.session.add(req)
            db.session.commit()
            commodities = goodsCart.query.filter_by(added=True).all()
            flash("The commodity has been added to the shopping cart!")
            return render_template('shopping_cart.html', commodities = commodities)
        else:
            flash("Cannot add required quantity to shopping cart! Check the quantity on se
    return render_template('select-quantity.html', form = form)
```

Figure 4.7.1: Add to Cart Route

When the form is validated, the commodity's information is added to the 'goodsCart' table and an if-statement checks if the user has input the right quantity with respect to what is available. A message is flashed after adding the commodity to the database to notify the user and the shopping cart page is rendered that shows the user the commodities he/she has added.

```
#-----Shopping cart Page-----
@main.route('/shopping_cart/')
def shopping():
    commodities = goodsCart.query.filter_by(added=True).all()
    return render_template('shopping_cart.html', commodities = commodities)
#-----
```

Figure 4.7.2: Shopping cart default route

This is the default route for viewing commodities in the shopping cart. A database-querying statement is used to fetch a list of goods in the 'goodsCart' table, and the result stored in a variable that is passed with the shopping cart HTML template in a return function to display the goods.

```

<p>
    Below is the list of commodities you want to buy:
</p>
<table class="table">
    <tr>
        <th>Commodity ID</th>
        <th>Seller</th>
        <th>Commodity name</th>
        <th>Commodity Type</th>
        <th>Quantity to be bought</th>
        <th>Price in Kenyan Shillings</th>
        <th>Option 1</th>
        <th>Option 2</th>
    </tr>
    {% for commodity3 in commodities %}
    <tr>
        <td><div class="commodity-id">{{ commodity3.id }}</div></td>
        <td><div class="commodity-author">{{ commodity3.seller }}</div></td>
        <td><div class="commodity-name">{{ commodity3.name }}</div></td>
        <td><div class="commodity-type">{{ commodity3.type }}</div></td>
        <td><div class="commodity-quantity">{{ commodity3.reqQuantity }}</div></td>
        <td><div class="commodity-price">{{ commodity3.price }}</div></td>
        <td>
            <a class="btn btn-default" href="{{ url_for('.delete_cart', id = commodity3.id) }}">
                Remove from shopping Cart
            </a>
        </td>
        <td>
            <a class="btn btn-default" href="{{ url_for('.change_quantity', id = commodity3.id) }}">
                Change quantity
            </a>
        </td>
    </tr>
    {% endfor %}

```

Figure 4.7.3: The shopping cart HTML template

The template uses a for loop to display all the commodities that were added to the shopping cart. Each item is displayed as a block containing the commodity's properties and options to delete from the cart and to change the quantity to be bought.

```

#-----Payment Page-----
@main.route('/payment_page/')
def payment():
    commodities = goodsCart.query.filter_by(added=True).all()
    return render_template('payment_page.html', commodities = commodities)
#-----

```

Figure 4.7.4: The payment page route

A database-querying statement is used to fetch a list of commodities that have been added to the shopping cart under the 'goodsCart' model table. The result list is stored in a variable and passed along with the payment page HTML template to render a page with the commodities.

```
#-----Route for paying in cash-----
@main.route('/cash_payment/')
def cash_payment():
    commodities = goodsCart.query.filter_by(buyer = current_user.name).all()
    coomm1 = goodsCart.query.filter_by(buyer = current_user.name).first()
    sum = 0
    for comm4 in commodities:
        sum = sum + comm4.price
    return render_template('cash.html', sum = sum, commodities = commodities,
        coomm1=coomm1)

#-----Route for paying via MPESA-----
@main.route('/mpesa_payment/')
def mpesa_payment():
    commodities = goodsCart.query.filter_by(buyer = current_user.name).all()
    coomm1 = goodsCart.query.filter_by(buyer = current_user.name).first()
    sum = 0
    for comm5 in commodities:
        sum = sum + comm5.price
    return render_template('mpesa.html', sum = sum, commodities = commodities,
        coomm1=coomm1)

#-----
```

Figure 4.7.5: Routes for cash and MPESA payment options

The two routes are similar and contain the respective database-querying statements to fetch commodities. Also, they both use a for-loop to add and get the total price of goods in the shopping cart to show the user.


```

<p>
  <b>Buy Goods and Services</b>
</p>
<p>
  <b>Till Number: <i>1089667</i></b>
</p>
<p>
  <b>Name: <i>MMAFUNZI PLATFORM SERVICES</i></b>
</p>
Afterwards, report to the Collection Centre at Mahatma Gandhi Wing with your
<b>school ID card</b>, <b>the respective MPESA message</b> and the following details to collect your purchased goods:
<p>
  <b>Receipt Number: <i>0000{{ coomm1.id }}</i></b>
</p>
<table class="table">
  <tr>
    <th>Quantity </th>
    <th>Item </th>
  </tr>
  {% for coomm in commodities%}
  <tr>
    <td>
      <div class="commodity-short1">{{ coomm.reqQuantity }}</div>
    </td>
    <td>
      <div class="commodity-short1">{{ coomm.name }}</div>
    </td>
  </tr>
  {% endfor %}
</table>
<p>
  <b>Total cash to be paid: <i>Ksh. {{sum}}</i></b>
</p>
<a class="btn btn-default" href="{{ url_for('.confirm') }}">
  Confirm
</a>

```

Figure 4.7.6: MPESA HTML template

The template uses the <p> tag to instruct the user on what to do after confirming the purchase of goods. The commodities' quantity and name are displayed to remind the user of what he/she is buying. A 'Confirm' button confirms the purchase when clicked and sends the information to the administrators for future reference.

```

{% block page_content %}
<div class="page-header">
  <h1>Goods bought by users</h1>
</div>
<ul class="bought">
  {% if bought == [] %}
  <p> Sorry! No goods have been bought yet...</p>
  <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>
  {% else %}
  <table class="table">
    <tr>
      <th>Receipt number </th>
      <th>Buyer </th>
      <th>Commodity name </th>
      <th>Commodity Type </th>
      <th>Expiry Date </th>
      <th>Manufacture Date </th>
      <th>Quantity bought </th>
      <th>Price in Kenyan Shillings </th>
      <th>Time of Purchase </th>
      <th>Bought </th>
      <th>Claimed </th>
      <th>Sold by </th>
      <th>Option </th>
    </tr>
    {% for boughts in bought %}
    <tr>
      <td>
        <div class="commodity-receipt">0000{{ boughts.id }}</div>
      </td>
    </tr>
    </table>
  {% endif %}
</ul>
{% endblock %}

```

Figure 4.7.7: HTML template for viewing and confirming claim of goods

When the user presents himself physically to the office to claim his/her goods, the administrator accesses the ‘Goods Bought’ page that has information on goods that have been bought and may be confirmed and unconfirmed.

```

    <div class="commodity-bought">{{ boughts.bought }}</div>
  </td>
  <td>
    <div class="commodity-claimed">{{ boughts.claimed }}</div>
  </td>
  <td>
    <div class="commodity-seller">{{ boughts.seller }}</div>
  </td>
  <td>
    {% if boughts.claimed == False %}
    <a class="btn btn-default" href="{{ url_for('.confirm_claim', id = boughts.id) }}">
      Confirm claim
    </a>
    {% endif %}
  </td>
</tr>
{% endfor %}
</table>
{% endif %}
</ul>
{% endblock %}

```

Figure 4.7.8: HTML template showing button for confirming claim of goods

When the ‘Confirm’ button is pressed, the page is updated to show that the good has been claimed.

4.8 – About and Contact Page

4.8.1 – About Page

```
{% block page_content %}
<link rel = "stylesheet" href = '/static/main.css/'/>
<div class="page-header">
|   <h1>Mwafunzi Discount Platform</h1>
</div>
<div>
|   <p>
|       Established in 2022, Mwafunzi is an online discount platform
|       that serves the students of the University of Nairobi by
|       providing goods at discounts.
|   </p>
</div>
<div class="page-header">
|   <h1>Opening Hours</h1>
</div>
<div>
|   <p>
|       Our office in Mahatma Gandhi Wing is <b>open on weekdays from 8.00AM to
|       5.30PM</b> and on <b>weekends from 10.00AM to 4.00PM.</b>
|   </p>
</div>
```

Figure 4.8.1.2: About HTML template

The about page displays general information about the platform as well as the opening hours of the main office where students go to claim their goods.

4.8.2 – Contacts Page

```
{% block page_content %}
<div class="page-header">
  <h1>Mwafunzi Discount Platform</h1>
</div>
<div>
  <div class="page-header">
    <h2>Address</h2>
  </div>
  <div>
    <ul>
      <li>P.O BOX 1240-00100 GPO</li>
      <li>Nairobi</li>
      <li>Tel: +254701018899</li>
      <li>Twitter: Mwafunzi Discount Platform</li>
    </ul>
  </div>
  <div class="page-header">
    <h2>Directions</h2>
  </div>
  <ul>
    <li>Mwafunzi Building, 1st Floor</li>
    <li>Next to Mahatma Gandhi Building</li>
  </ul>
  <hr>
  
  </ul>
</div>
</div>
{% endblock %}
```

Figure 4.8.2.1: Contacts HTML template

The About HTML template renders a page that contains the address of and directions to the main office at Mahatma Gandhi wing. A picture with map of the area is displayed to give further information of where the office is located.

CHAPTER 5 - RESULTS AND ANALYSIS

When the code is running, the pages are displayed and are linked to one another using buttons, tabs and links. The following pages and items are of particular importance:

1. Landing page
2. User login page
3. Navigation bar and search bar
4. Profile page
5. Catalog page with four types of commodities
6. Shopping Cart page and checkout

5.1 – Landing Page

5.1.1 – Landing Page

Hello, Drew49!

Welcome to Mwafunzil

Dashboard

No messages have been sent to you yet....

Other Options

Would you like to **explore our catalog** of goods on sale? [Click here](#) to go to Catalog Page

Would you like to **sell** to other students? [Click here](#) to go to add a commodity for sale

Would you like to **become an administrator** or **relinquish the role**? [Click here](#) to submit a request

Figure 5.1.1: Landing page for a logged-in user

The landing page contains three sections:

- a) A greeting section with the user's username followed by a welcome message.
- b) A dashboard section where messages from administrators are seen by the user
- c) An 'Other Options' section where a user has options to explore the catalog, add commodities for sale or request to become an administrator.

5.1.2 – Dashboard section

Dashboard

Most Recent

Sender	Time Sent	Message
Admin	2022-05-11 08:19:36.129267	Hello, again =)

See all

Figure 5.1.2.1: Dashboard of a user with messages

The dashboard shows the most recent message sent, but all messages can be viewed by clicking the ‘See all’ button.

All messages

Sender	Time Sent	Message
Admin	2022-05-11 08:19:36.129267	Hello, again =)
Admin	2022-05-11 07:48:30.411728	Hello there! Welcome to Mwafunzi.

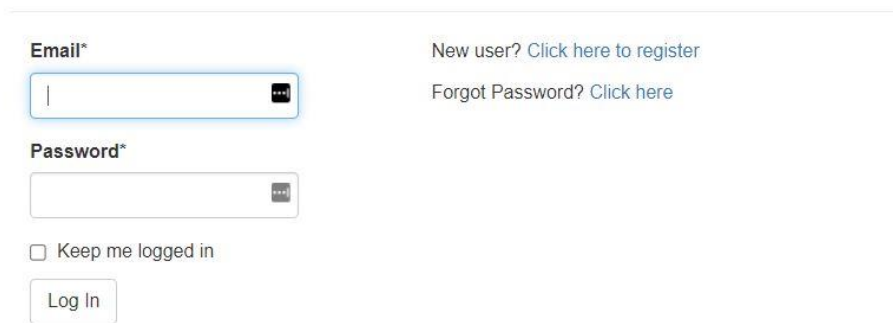
© Mwafunzi Platform 2022

Figure 5.1.2.2: Displaying all messages of a user

The messages are displayed in descending order (from the newest to the oldest). This can be verified from the ‘Time Sent’ section.

5.2 – User Login page

Login



The login page features a form with two input fields: 'Email*' and 'Password*'. The 'Email*' field is highlighted with a blue border and contains a cursor. To the right of the email field are two links: 'New user? Click here to register' and 'Forgot Password? Click here'. Below the password field is a checkbox labeled 'Keep me logged in' and a 'Log In' button.

Email*

New user? [Click here to register](#)

Forgot Password? [Click here](#)

Password*

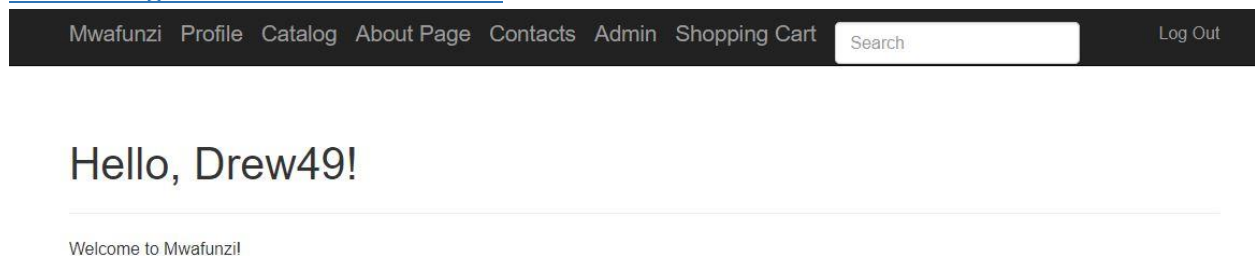
☐ Keep me logged in

Log In

Figure 5.2.1: Login Page for user

The login page displays the fields for inputting the user email and password, a 'Keep me logged in' checkbox and a 'Log In' button.

5.3 – Navigation bar and search bar



The navigation bar is a dark horizontal strip containing links: 'Mwafunzi', 'Profile', 'Catalog', 'About Page', 'Contacts', 'Admin', and 'Shopping Cart'. To the right of these links is a search bar with the placeholder text 'Search' and a 'Log Out' button.

Hello, Drew49!

Welcome to Mwafunzi!

Figure 5.3 – Navigation bar and search bar

All the elements of the navigation bar are displayed if the user is authenticated and confirmed.

5.4 – Profile Page

5.4.1 – User information section



Drew49

Name : Andrew Muirui

Email : drewmuiruri49@gmail.com

Year of Study : Fifth Year

Registration Number : F17/101710/2017

Phone Number : 799148618

Edit Profile [Admin]


Send Message

Figure 5.4.1: Profile page section containing user details and choice to edit

A user gravatar along with the user's details are displayed in this section.

5.4.2 – User commodity section

Commodities on sale by Drew49

Photo	Seller	Commodity name	Commodity Type	Quantity on sale	Price per unit	Percentage Discount	In Stock	Option
	Drew49	Watch	Electronics	7	Ksh 850	6%	Yes	Delete commodity from sale

Commodities pending by Drew49


Photo	Seller	Commodity name	Commodity Type	Quantity on sale	Price per unit	Percentage Discount	In Stock	Option
	Andrew Muiruri	Soda	Foodstuffs	10	Ksh 65	7%	Yes	Delete

Figure 5.4.2.1: Commodity section in the user profile page

This section contains a table showing the approved commodities that are on sale and another that shows details of those yet to be approved.

5.4.3 – Goods sold and bought by user

Commodities bought by Drew49

[See records](#)

Commodities sold by Drew49

[See records](#)

Figure 5.4.3.1: Section showing goods sold and bought by user

This section acts as an online reference for future use if needed.

5.5 – Catalog Page

Catalog Page

Here at Mwafunzi, we are dedicated to providing students with quality goods that pass through a thorough inspection phase before reaching the consumer.

Here are the types of goods we offer:



Electronics



Foodstuffs



Stationery



Cutlery

Or would you like to add a commodity for sale instead?



[Add Commodity](#)

Figure 5.5.1: The Catalog Page

This page contains links to all four types of commodities plus a link to add a commodity for sale.

5.5.1 – Electronics Page



Seller : [Andrew Muiruri](#)

Commodity name : Calculator

Commodity Type : Electronics

Quantity on sale : 7

Old Price per unit in Kenyan Shillings : 1600

Discounted Price per unit in Kenyan Shillings : 1400

Percentage Discount : 12%

In Stock : Yes

Additional Description : Black, Casio

[Add to Cart](#)

[See full picture](#)



Seller : [Andrew Muiruri](#)

Commodity name : SmartWatch

Commodity Type : Electronics

Quantity on sale : 30

Old Price per unit in Kenyan Shillings : 900

Discounted Price per unit in Kenyan Shillings : 780

Percentage Discount : 13%

In Stock : Yes

Additional Description : Black, plastic

[Add to Cart](#)

[See full picture](#)

Figure 5.5.1.1: Electronics page with two commodities

The Electronics page consists of a title, a picture at the top and a list of electronic commodities on sale. Each item on the list consists of the commodity's details, including its name and discounted price. If there are no electronics goods on sale, a default picture of a whale will be displayed with a message that communicates the same. Also, there are options to add to cart or see the full commodity picture.

If the user is an administrator, the seller's name will appear as a link that direct to that seller's profile page. This feature is also available for all the other commodity pages.

5.5.2 – Foodstuffs Page



Seller : Andrew Muiruri
Commodity name : Soda
Commodity Type : Foodstuffs
Manufacture Date : 28/08/2022
Expiry Date : 09/09/2022
Quantity on sale : 10
Old Price per unit in Kenyan Shillings : 1000
Discounted Price per unit in Kenyan Shillings : 970
Percentage Discount : 3%
In Stock : Yes
Additional Description : A pack of 10 soda cans

[Add to Cart](#) [See full picture](#)

Figure 5.5.2.1: The foodstuffs page with one commodity

For the foodstuffs page, the manufacture date and expiry date have been included because foodstuffs are perishable.

5.5.3 – Stationery Page

Stationery commodities on sale



Figure 5.5.3: Stationery Page

The stationery page, as well as the cutlery page, resembles that of the other commodities.

[5.6 – Shopping Cart](#)

[5.6.1 – Shopping Cart page](#)

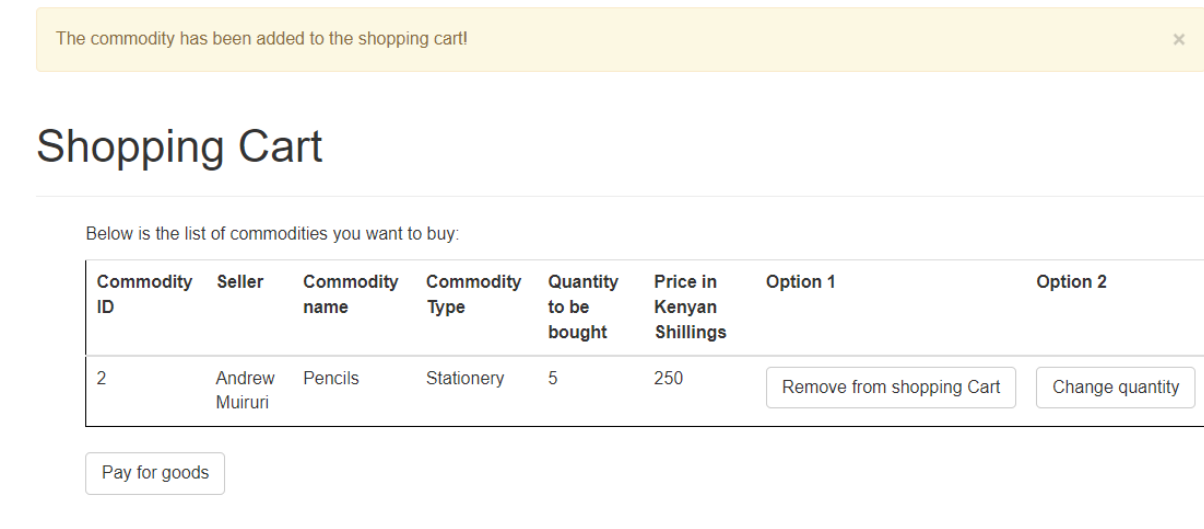


Figure 5.6.1.2: The Shopping Cart Page

This page shows the commodities in the user’s shopping cart in a table as well as their important details. There are options to remove a commodity from the shopping cart or change a commodity’s quantity.

[5.6.2 – Payment Page](#)

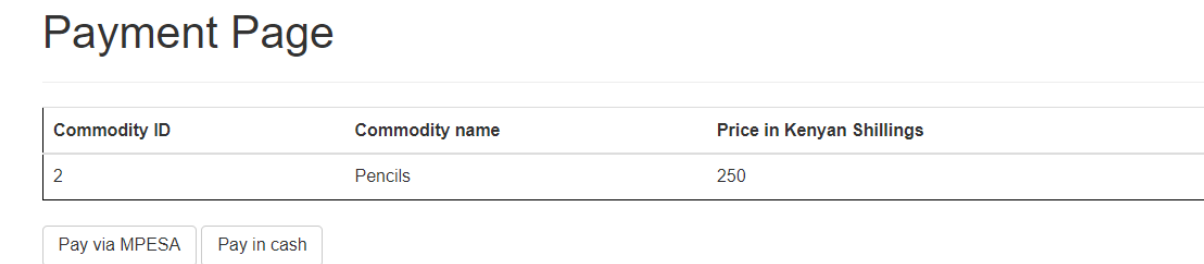


Figure 5.6.2.1 – Payment Page

Here, a user is presented with two options: To pay with MPESA or pay in cash. Both options are to be finalized when the user goes to claim goods at the main office.

5.6.3 - Pay with MPESA Page

Pay with MPESA

Please pay the amount shown below to the following Till Number:

Buy Goods and Services

Till Number: 1089667

Name: MWAFUNZI PLATFORM SERVICES

Afterwards, report to the Collection Centre at Mahatma Gandhi Wing with your **school ID card**, the **respective MPESA message** and the following details to collect your purchased goods:

Receipt Number: 00002

Quantity	Item
5	Pencils

Total cash to be paid: Ksh. 250

Confirm

Please contact us at **+254701018899** for any assistance and/or enquiries.

Figure 5.6.3.1: Pay with MPESA Page

The details of the transaction are displayed to the user as well as instructions to follow when claiming goods from the main office. The ‘Confirm’ button confirms the purchase and the commodity is automatically deleted from the shopping cart. Also, the quantity of the commodity is adjusted on the commodity page to show the stock that remains. The page for paying in cash is similar to this page.

CHAPTER 6 - CONCLUSION AND RECOMMENDATIONS

6.1 - Conclusion

Mwafunzi, the one-stop online discount platform, was implemented using Python, Flask, CSS and HTML. This is a platform that enables students to buy and sell new and used commodities to and from each other to minimize wastage and encourage reusability. To be able to buy or sell, a student needs to register himself in the platform and await user detail confirmation from the admins in order to be authenticated. Once authenticated, a user is free to explore the catalog of goods, buy and add commodities, as well as become an administrator to monitor the activities of other students.

To add a commodity, a user needs to decide what margin of discount to give and input the original value of the commodity, the new value and the rest of the details such as type, quantity and date of manufacture. To buy a commodity, the user needs to add the commodity to the shopping cart and choose a preferred method of payment: either via MPESA or cash. Once the purchase is confirmed, the user needs to claim his/her goods from the main office, where an administrator will confirm the details of the user in the platform and the dispatch of goods.

6.2 - Recommendations

This project would be better implemented using Django macro-framework, since:

1. An online discount platform needs well-defined separate administrator and user sides, instead of defining roles in a database
2. Such a platform needs a lot of considerations which would be handled neatly using Django, such as:
 - i) Monitoring perishable foodstuffs and removing them from sale before the expire
 - ii) Monitoring commodity posts that are out of stock
 - iii) Monitoring user activities to prevent fraud and sale of illegal commodities

BIBLIOGRAPHY

References

WEBSITES

- [1] OECD, “An Introduction to Online Platforms and Their Role in the Digital Transformation”, OECD iLibrary, May 2019, [What is an “online platform”? | An Introduction to Online Platforms and Their Role in the Digital Transformation | OECD iLibrary \(oecd-ilibrary.org\)](#) (Accessed January 19, 2022)
- [2] New World Encyclopedia, “Online shopping”, April 2020, [Online shopping - New World Encyclopedia](#) (Accessed January 19, 2022)
- [3] Our Shopee, “How Online Shopping Has Made Our Lives Easier”, Medium, [How Online Shopping has Made our Lives Easier | by Our Shopee | Medium](#) (Accessed January 19, 2022)
- [4] Vishnava, “Barter System”, Cleartax, January 2022, [barter system - Definition, Latest News, and Why barter system is Important? \(cleartax.in\)](#) (Accessed January 19, 2022)
- [5] Next Gurukul, “Barter System: Advantages and Disadvantages”, [Notes On Barter System - Advantages and disadvantages - ICSE Class 10 Economics \(nextgurukul.in\)](#) (Accessed January 19, 2022)
- [6] Iamcheated.com Research Team, “Disadvantages of Online Shopping”, Iamcheated, October 2018, [Disadvantages of Online Shopping \(indianmoney.com\)](#) (Accessed January 19, 2022)

PERIODICAL

- [7] Ozlem Doygun, Selma Gulec, “The problems faced by university students and proposals for solution, Procedia – Social and Behavioural Sciences, pp. 1115-1123.

CONFERENCE

- [8] Fan Wei and Qian Zang, “Design and Implementation of an Online Shopping System Based on B/S Model”, presented at MATEC Web of Conferences in School of Computer Science, Xi’an Shiyu University, 710065, Shaanxi, P. R. China, 2018.

WEBSITES

- [9] Anastasiia Lastovetska, "How to Build an E-Commerce Website from Scratch & Start Selling Products Worldwide", MLSDev, November 2021, [How to Build an eCommerce Website from Scratch | MLSDev](#)
- [10] SME Joinup, "5 Essential Components of an E-Commerce Platform", August 2017, [5 Essential Components Of An E-Commerce Platform - SMEJoinup](#)
- [11] Simba Dube, "11 E-Commerce Discount Pricing Strategies That Will Increase Conversion Rate of Your E-Commerce Website", Invesp, April 2021, [11 E-Commerce Discount Pricing Strategies That Will Increase Conversion Rate of Your e-Commerce Website \(invespcro.com\)](#)
- [12] Lvivity Team, "Web-Based Application: What It is, and Why You Should Use It", Lvivity. [Web-Based Application: What It Is, and Why You Should Use It \(lvivity.com\)](#)
- [13] Kathleen Kotwica, "The Benefits and Security Risks of Web-Based Applications for Business", ScienceDirect, 2013. [Web-Based Application - an overview | ScienceDirect Topics](#)

E-BOOK

- [14] Miguel Grinberg, *Flask Web Development: Developing Web Applications with Python*, Sebastopol, CA, US: O'Reilly Media Inc. 2018. [Online] Available: Z-Library

WEBSITE

- [15] W3 Schools, "Web-Based Database Management System", [Web-based Database Management System \(w3schools.in\)](#)

JOURNAL

- [16] Khairul Anwar Sedek, Norlis Osman, Mohd Nizam Osman and Hj. Jusoff, "Developing a Secure Web Application Using OWASP Guidelines", Research Management Institute(RMI) of UiTM, Vol. 2, No. 4, pp 137-143, November 2009

WEBSITE

- [17] Shivesh Singh, “OWASP Top 10 Security Guidelines”, Bajra Technologies, September 2017. [OWASP Top 10 Security Guidelines. We have heard of numerous cases of... | by Shivesh Singh | Bajra Technologies Blog](#)

APPENDICES

Code Implemented for This Project

views.py

```
#Application Configuration-----
import os
basedir = os.path.abspath(os.path.dirname(__file__))
class Config:
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'BUsiN@ss56nyM'
    MAIL_SERVER = os.environ.get('MAIL_SERVER', 'smtp.googlemail.com')
    MAIL_PORT = int(os.environ.get('MAIL_PORT', '587'))
    MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS', 'true').lower() in \
        ['true', 'on', '1']
    MAIL_USERNAME = os.environ.get('MAIL_USERNAME')
    MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD')
    FLASKY_MAIL_SUBJECT_PREFIX = '[Flasky]'
    FLASKY_MAIL_SENDER = 'Flasky Admin <flasky@example.com>'
    FLASKY_ADMIN = os.environ.get('FLASKY_ADMIN')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    @staticmethod
    def init_app(app):
        pass
class DevelopmentConfig(Config):
    DEBUG = True
    SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') or \
        'sqlite:/// ' + os.path.join(basedir, 'data-dev.sqlite')
class TestingConfig(Config):
    TESTING = True
    SQLALCHEMY_DATABASE_URI = os.environ.get('TEST_DATABASE_URL') or \
        'sqlite://'
class ProductionConfig(Config):
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
        'sqlite:/// ' + os.path.join(basedir, 'data.sqlite')
config = {
    'development': DevelopmentConfig,
    'testing': TestingConfig,
    'production': ProductionConfig,
    'default': DevelopmentConfig}
#-----
#-----Login form-----
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField, SubmitField, TextAreaField
from wtforms import SelectField
from wtforms.validators import DataRequired, Length, Email

class LoginForm(FlaskForm):
    email = StringField('Email*', validators=[DataRequired(), Length(1, 64), Email()])
    password = PasswordField('Password*', validators=[DataRequired()])
    remember_me = BooleanField('Keep me logged in')
    submit = SubmitField('Log In')
#-----
#-----user registration form-----
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Length, Email, Regexp, EqualTo
from wtforms import ValidationError, IntegerField
from ..models import User

class RegistrationForm(FlaskForm):
```

```

name = StringField('Full Name*', validators=[DataRequired()])
username = StringField('Username*', validators=[
    Length(1, 64), Regexp('^[A-Za-z][A-Za-z0-9_]*$', 0,
        'Usernames must have only letters, numbers, dots or '
        'underscores'), DataRequired()])
email = StringField('Email*', validators=[DataRequired(), Length(1, 64), Email()])
yearOfStudy = SelectField('Year of Study*', choices=[
    ('First Year',"First Year"), ('Second Year',"Second Year"), ('Third Year',"Third
Year"),
    ('Fourth Year',"Fourth Year"), ('Fifth Year',"Fifth Year")],
validators=[DataRequired()])
regNumber=StringField('Registration Number*', validators=[DataRequired(),
    Length(1, 15), Regexp('^[A-Za-z][A-Za-z0-9/]*$')])
phoneNumber = StringField('Phone Number*', validators=[DataRequired(),
    Length(1, 10) ])
secQuestion = TextAreaField('Your security question*(In case of forgotten password)',
    validators=[DataRequired(), Length(1, 100)])
secAnswer = TextAreaField('Your security answer*(In case of forgotten password)',
    validators=[DataRequired(), Length(1, 100)])
password = PasswordField('Password*', validators=[
    DataRequired(), EqualTo('password2', message='Passwords must match.')])
password2 = PasswordField('Confirm password*', validators=[DataRequired()])
submit = SubmitField('Register')
class EditProfileForm(FlaskForm):
    name = StringField('Full Name*', validators=[DataRequired()])
    username = StringField('Username*', validators=[
        Length(1, 64), Regexp('^[A-Za-z][A-Za-z0-9_]*$', 0,
            'Usernames must have only letters, numbers, dots or '
            'underscores'), DataRequired()])
    email = StringField('Email*', validators=[DataRequired(), Length(1, 64), Email()])
    yearOfStudy = SelectField('Year of Study*', choices=[
        ('First Year',"First Year"), ('Second Year',"Second Year"), ('Third Year',"Third
Year"),
        ('Fourth Year',"Fourth Year"), ('Fifth Year',"Fifth Year")],
validators=[DataRequired()])
    regNumber=StringField('Registration Number*', validators=[DataRequired(),
        Length(1, 15), Regexp('^[A-Za-z][A-Za-z0-9/]*$')])
    phoneNumber = StringField('Phone Number*', validators=[DataRequired(),
        Length(1, 10) ])
    secQuestion = TextAreaField('Your security question*(In case of forgotten password)',
        validators=[DataRequired(), Length(1, 100)])
    secAnswer = TextAreaField('Your security answer*(In case of forgotten password)',
        validators=[DataRequired(), Length(1, 100)])
    password = PasswordField('Password*', validators=[
        DataRequired(), EqualTo('password2', message='Passwords must match.')])
    password2 = PasswordField('Confirm password*', validators=[DataRequired()])
    submit = SubmitField('Make changes')
#-----
#-----Profile editing form for Administrators-----
class EditProfileAdminForm(FlaskForm):
    #name = StringField('Real name*', validators=[Length(0, 64)])
    email = StringField('Email*', validators=[DataRequired(), Length(1, 64),
        Email()])
    username = StringField('Username*', validators=[
        DataRequired(), Length(1, 64),
        Regexp('^[A-Za-z][A-Za-z0-9_]*$', 0,
            'Usernames must have only letters, numbers, dots or '
            'underscores')])
    yearOfStudy = SelectField('Year of Study*', choices=[
        ('First Year',"First Year"), ('Second Year',"Second Year"), ('Third Year',"Third
Year"),

```

```

        ('Fourth Year',"Fourth Year"), ('Fifth Year',"Fifth Year"]],
validators=[DataRequired()])
    regNumber=StringField('Registration Number', validators=[DataRequired(),
        Length(1, 15), Regexp('^[A-Za-z][A-Za-z0-9/*$')])
    phoneNumber = StringField('Phone Number', validators=[DataRequired(),
        Length(1, 10) ])
    confirmed = BooleanField('Confirmed')
    role = SelectField('Role*', choices=[('User',"User"),('Administrator',"Administrator")],
validators=[DataRequired()])
    submit = SubmitField('Submit')
class Email(FlaskForm):
    userEmail = StringField('Input your email*', validators = [DataRequired(),
        Length(1, 50)])
    submit = SubmitField('Submit')
#-----
#-----
class AccountRecovery(FlaskForm):
    secAnswer = StringField('Answer*', validators = [DataRequired(),
        Length(1, 100)])
    submit = SubmitField('Submit')
#-----
def validate_email(self, field):
    if User.query.filter_by(email=field.data).first():
        raise ValidationError('Email already registered.')
def validate_username(self, field):
    if User.query.filter_by(username=field.data).first():
        raise ValidationError('Username already in use.')
def validate_regnumber(self, field):
    if User.query.filter_by(regNumber=field.data).first():
        raise ValidationError('Registration number already in use.')
def validate_password(self, field):
    if User.query.filter_by(password=field.data).first():
        raise ValidationError('Password already in use.')
#-----
#-----Commodity form-----
class CommodityForm(FlaskForm):
    name = StringField('Commodity Name*', validators=[Length(0, 64), DataRequired()])
    type = SelectField("Type of Commodity*", choices=[
        ('Electronics',"Electronics"), ('Cutlery',"Cutlery"), ('Stationery',"Stationery"),
        ('Foodstuffs',"Foodstuffs")], validators=[DataRequired()])
    manDate = StringField('Date of manufacture* (could be the year or date in preferred
format)', validators=[Length(0,16)])
    expDate = StringField('Date of expiry (could be None for Electronics, Cutlery and
Stationery)*', validators=[Length(0,16), DataRequired()])
    quantity = IntegerField('Quantity on sale', validators=[DataRequired()])
    oldPrice = IntegerField('Old Price per unit in Kenyan Shillings*',
validators=[DataRequired()])
    discPrice = IntegerField('Discounted Price per unit in Kenyan Shillings*',
validators=[DataRequired()])
    addDesc = TextAreaField('Add any additional descriptions here: ')
    submit = SubmitField('Submit')
#-----
#-----Shopping cart quantity form-----
class QuantityForm(FlaskForm):
    goodQuantity = IntegerField('How many would you like to
buy?*',validators=[DataRequired()])
    submit = SubmitField('Confirm')
#-----
#-----Form for granting admin priveledges-----
class PendingForm(FlaskForm):
    issue = SelectField('Which issue would you like to be addressed? *', choices =

```

```

        [('Grant admin priveledges',"Grant admin priveledges"),
        ('Relinquish admin priveledges', "Relinquish admin priveledges")],
validators=[DataRequired()])
    submit = SubmitField('Confirm')
#-----
#-----Search Form-----
class SearchForm(FlaskForm):
    searched = StringField("Searched", validators=[DataRequired()])
    submit = SubmitField("Submit")
#-----
#-----Form for granting admin priveledges-----
class GeneralForm(FlaskForm):
    issue = TextAreaField('Which general issue would you like to be addressed? *',
validators=[DataRequired()])
    submit = SubmitField('Confirm')
#-----
#-----Form for sending message to user-----
class SendMessageForm(FlaskForm):
    message = TextAreaField('Type your message here*', validators=[DataRequired()])
    submit = SubmitField('Send Message')
#-----
from flask import render_template, redirect, request, url_for, flash
from flask_login import login_user, current_user
from . import auth
from ..models import Pending, User, Dashboard #check and confirm db
from .forms import LoginForm, RegistrationForm, Email, AccountRecovery#check and confirm
registration form
from app.email import send_email
from ..__init__ import db
from ..email import send_email

@auth.route('/index/')
def index2():
    messages = Dashboard.query.filter_by(name =
current_user.name).order_by(Dashboard.timestamp.desc()).first()
    return render_template('index.html', messages = messages)
#-----
#-----Welcoming message-----
@auth.route('/')
def index():
    return render_template('index.html')
#-----
@auth.route('/login/', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user is None:
            flash('Invalid username or password.')
        if user is not None and user.verify_password(form.password.data):
            login_user(user, form.remember_me.data)
            next = request.args.get('next')
            if next is None or not next.startswith('/'):
                next = url_for('auth.index2')
            flash('Login successful.')
            if current_user.confirmed == False:
                flash('Your account will be confirmed shortly!')
            return redirect(next)
    return render_template('auth/login.html', form=form)
#-----
#-----Log out route-----

```

```

from flask_login import logout_user, login_required
@auth.route('/logout')
@login_required
def logout():
    logout_user()
    flash('You have been logged out.')
    return redirect(url_for('auth.index'))
#-----
#-----user registration route-----
@auth.route('/register/', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        user = User(name=form.name.data,
                    email=form.email.data,
                    username=form.username.data,
                    password=form.password.data,
                    yearOfStudy=form.yearOfStudy.data,
                    regNumber=form.regNumber.data,
                    phoneNumber=form.phoneNumber.data,
                    secQuestion=form.secQuestion.data,
                    secAnswer=form.secAnswer.data)
        op1 = User.query.filter_by(username=form.username.data).first()
        op2 = User.query.filter_by(regNumber=form.regNumber.data).first()
        op3 = User.query.filter_by(phoneNumber=form.phoneNumber.data).first()
        op4 = User.query.filter_by(email=form.email.data).first()
        if op1:
            flash('Username already taken!')
            return render_template('auth/register.html', form=form)
        elif op2:
            flash('Registration number already taken!')
            return render_template('auth/register.html', form=form)
        elif op3:
            flash('Phone number already taken!')
            return render_template('auth/register.html', form=form)
        elif op4:
            flash('Email already taken!')
            return render_template('auth/register.html', form=form)
        else:
            issues = Pending(name = form.name.data,
                            username = form.username.data,
                            issue = "Confirm user registration." )
            db.session.add(user)
            db.session.add(issues)
            db.session.commit()
            flash('You have been registered. Please log in and await confirmation.')
            return redirect(url_for('auth.login'))
    return render_template('auth/register.html', form=form)
#-----
#-----Route for recovering account-----
@auth.route('/forgot', methods=['GET', 'POST'])
def forgot():
    form = Email()
    if form.validate_on_submit():
        userf = User.query.filter_by(email=form.userEmail.data).first()
        if userf:
            email = userf.email
            return redirect(url_for('auth.question', email = email))
        else:
            flash('No such email exists!')
    return render_template('email.html', form = form)

```

```

#-----Route for asking security question-----
@auth.route('/question/<email>', methods=['GET', 'POST'])
def question(email):
    userf = User.query.filter_by(email=email).first()
    question = userf.secQuestion
    form = AccountRecovery()
    if form.validate_on_submit():
        if form.secAnswer.data == userf.secAnswer:
            flash('Correct! Please change your account details accordingly.')
            return redirect(url_for('main.edit_profile2', id = userf.user_id))
        else:
            flash('Wrong answer. Try again')
    return render_template('recovery.html', form = form, question = question)
#-----Landing page route-----
@main.route('/about/')
def about():
    return render_template('about.html')
#-----Landing page route-----
@main.route('/')
def index():
    if current_user.is_anonymous:
        return render_template('index.html')
    else:
        messages = Dashboard.query.filter_by(name = current_user.name).order_by(
            Dashboard.timestamp.desc()).first()
        if messages is None:
            return render_template('index.html', messages=None)
        return render_template('index.html', messages=messages)
#-----
@main.route('/contacts/')
def contacts():
    return render_template('contacts.html')
@main.route('/catalog/')
def catalog():
    return render_template('catalog.html')
@main.route('/catalog/electronics/')
def commoditypg1():
    commodities = Commodity.query.filter_by(type='Electronics').order_by(
        Commodity.timestamp.desc()).all()
    if commodities is None:
        return render_template('foodstuffs.html', commodities = None)
    return render_template('electronics.html', commodities = commodities)
@main.route('/catalog/cutlery/')
def commoditypg2():
    commodities = Commodity.query.filter_by(type='Cutlery').order_by(
        Commodity.timestamp.desc()).all()
    if commodities is None:
        return render_template('foodstuffs.html', commodities = None)
    return render_template('cutlery.html', commodities = commodities)
@main.route('/catalog/foodstuffs/')
def commoditypg3():
    commodities = Commodity.query.filter_by(type='Foodstuffs').order_by(
        Commodity.timestamp.desc()).all()
    if commodities is None:
        return render_template('foodstuffs.html', commodities = None)
    return render_template('foodstuffs.html', commodities = commodities)
@main.route('/catalog/stationery/')
def commoditypg4():
    commodities = Commodity.query.filter_by(type='Stationery').order_by(

```



```

        Commodity.timestamp.desc()).all()
    if commodities is None:
        return render_template('foodstuffs.html', commodities = None)
    return render_template('stationery.html', commodities = commodities)
#-----edit profile route-----
@main.route('/edit-profile', methods=['GET', 'POST'])
@login_required
def edit_profile():
    form = EditProfileForm()
    if form.validate_on_submit():
        current_user.name = form.name.data
        current_user.username = form.username.data
        current_user.email = form.email.data
        current_user.yearOfStudy = form.yearOfStudy.data
        current_user.regNumber = form.regNumber.data
        current_user.phoneNumber = form.phoneNumber.data
        current_user.password = form.password.data
        current_user.secQuestion = form.secQuestion.data
        current_user.secAnswer = form.secAnswer.data
        db.session.add(current_user._get_current_object())
        db.session.commit()
        flash('Your profile has been updated.')
        return redirect(url_for('.user', username=current_user.username))
    form.name.data = current_user.name
    form.email.data = current_user.email
    form.username.data = current_user.username
    form.yearOfStudy.data = current_user.yearOfStudy
    form.regNumber.data = current_user.regNumber
    form.phoneNumber.data = current_user.phoneNumber
    form.secQuestion.data = current_user.secQuestion
    form.secAnswer.data = current_user.secAnswer
    return render_template('edit_profile.html', form=form)
#-----
#-----edit profile route for administrators-----
@main.route('/edit-profile/<int:id>', methods=['GET', 'POST'])
@login_required
def edit_profile_admin(id):
    user = User.query.get_or_404(id)
    form = EditProfileAdminForm(user=user)
    if form.validate_on_submit():
        user.email = form.email.data
        user.username = form.username.data
        user.confirmed = form.confirmed.data
        user.role = form.role.data
        user.regNumber = form.regNumber.data
        user.phoneNumber = form.phoneNumber.data
        db.session.add(user)
        db.session.commit()
        flash('The profile has been updated.')
        return redirect(url_for('.user', username=user.username))
    form.email.data = user.email
    form.username.data = user.username
    form.confirmed.data = user.confirmed
    form.role.data = user.role
    form.regNumber.data = user.regNumber
    form.phoneNumber.data = user.phoneNumber
    return render_template('edit_profile.html', form=form, user=user)
#-----
#-----Route for adding commodities-----
@main.route('/add-commodity/', methods=['GET', 'POST'])
def addCommodity():

```

```

form = CommodityForm()
if current_user.is_authenticated and form.validate_on_submit():#was --is.authenticated()
    commodity = goodsPending(name=form.name.data,
        type = form.type.data,
        manDate = form.manDate.data,
        expDate = form.expDate.data,
        quantity = form.quantity.data,
        oldPrice = form.oldPrice.data,
        discPrice = form.discPrice.data,
        percDiscount = round((((form.oldPrice.data)-
(form.discPrice.data))/(form.oldPrice.data))*100),
        author2=current_user.name,
        addDesc = form.addDesc.data) #was previously current_user._get_current_object()
    issuess = Pending(name = current_user.name,
        username = current_user.username,
        issue = "Commodity uploaded for review.")
    db.session.add(issuess)
    db.session.add(commodity)
    db.session.commit()
    flash('Please also include a picture. Your post will be deleted otherwise!')
    return render_template('upload_pics.html')
    commodities = Commodity.query.order_by(Commodity.timestamp.desc()).all()
    return render_template('add_commodity.html', form=form, commodities=commodities)
#-----
#-----Profile page route with commodity posts-----
@main.route('/user/<username>')
def user(username):
    user = User.query.filter_by(username=username).first()
    if user is None:
        abort(404)
    commodities = user.commodity.order_by(Commodity.timestamp.desc()).all()
    user1_role = current_user.role
    bout = goodsPurchased.query.filter_by(buyer = current_user.name).all()
    bout1 = goodsPending.query.filter_by(author2 = current_user.name).all()
    return render_template('user.html', user=user, commodities=commodities,
        role = user1_role, bout = bout, bout1 = bout1)
#-----
#-----Shopping cart Page-----
@main.route('/shopping_cart/')
def shopping():
    commodities = goodsCart.query.filter_by(added=True).all()
    return render_template('shopping_cart.html', commodities = commodities)
#-----
#-----Payment Page-----
@main.route('/payment_page/')
def payment():
    commodities = goodsCart.query.filter_by(added=True).all()
    return render_template('payment_page.html', commodities = commodities)
#-----
#-----Route for deleting a commodity-----
@main.route('/delete_commodity/<id>', methods=['GET', 'POST'])
def delete(id):
    comm=Commodity.query.filter_by(id=id).first()
    comm10=goodsCart.query.filter_by(id=id).first()
    comm30=goodsPending.query.filter_by(id=id).first()
    if current_user.is_authenticated:
        if comm != None:
            name = comm.author1
            user = User.query.filter_by(name=name).first()
            del14 = Img.query.filter_by(commID=comm.id).first()
            username = user.username

```

```

        db.session.delete(comm)
        if del4:
            db.session.delete(del4)
        db.session.commit()
        if comm10 != None:
            db.session.delete(comm10)
            db.session.commit()
        flash("The commodity has been deleted and is no longer on sale!")
        return redirect(url_for('.user', username = username))
    elif comm30 != None:
        name = comm30.author2
        user = User.query.filter_by(name=name).first()
        username = user.username
        db.session.delete(comm30)
        db.session.commit()
        flash("The commodity has been deleted!")
        return redirect(url_for('.user', username = username))
    commodities = user.commodity.order_by(Commodity.timestamp.desc()).all()
    user1_role = current_user.role
    bout = goodsPurchased.query.filter_by(buyer = current_user.name).all()
    return render_template('user.html', user=user, commodities=commodities, role = user1_role,
bout = bout)
#-----
#-----Route for adding to shopping cart-----
@main.route('/add-cart/<id>', methods=['GET', 'POST'])
def add_to_cart(id):
    commodities=Commodity.query.filter_by(id=id).first()
    form = QuantityForm()
    if form.validate_on_submit():
        req = goodsCart(id = commodities.id,
            name = commodities.name,
            type = commodities.type,
            manDate = commodities.manDate,
            expDate = commodities.expDate,
            reqQuantity = form.goodQuantity.data,
            price = (form.goodQuantity.data)*commodities.discPrice,
            percDiscount = commodities.percDiscount,
            timestamp = commodities.timestamp,
            buyer = current_user.name,
            seller = commodities.author1)
        if form.goodQuantity.data <= commodities.quantity and form.goodQuantity.data>0:
            db.session.add(req)
            db.session.commit()
            commodities = goodsCart.query.filter_by(added=True).all()
            flash("The commodity has been added to the shopping cart!")
            return render_template('shopping_cart.html', commodities = commodities)
        else:
            flash("Cannot add required quantity to shopping cart! Check the quantity on sale
and try again")
            return render_template('select-quantity.html', form = form)
#-----
#-----Route for changing quantity req in cart-----
@main.route('/change-quantity/<id>', methods=['GET', 'POST'])
def change_quantity(id):
    oldComm = goodsCart.query.filter_by(id=id).first()
    commodities=Commodity.query.filter_by(id=id).first()
    form = QuantityForm()
    if form.validate_on_submit():
        change = goodsCart(id = oldComm.id,
            name = oldComm.name,
            type = oldComm.type,

```

```

        manDate = oldComm.manDate,
        expDate = oldComm.expDate,
        reqQuantity = form.goodQuantity.data,
        price = (form.goodQuantity.data)*commodities.discPrice,
        percDiscount = oldComm.percDiscount,
        timestamp = oldComm.timestamp,
        buyer = current_user.name,
        seller = commodities.author1)
    if form.goodQuantity.data <= commodities.quantity and form.goodQuantity.data>0:
        db.session.delete(oldComm)
        db.session.add(change)
        db.session.commit()
        commodities = goodsCart.query.filter_by(added=True).all()
        flash("Quantity of goods to be bought has been changed!")
        return render_template('shopping_cart.html', commodities = commodities)
    else:
        flash("Cannot change required quantity in shopping cart! Check quantity on sale
and try again")
        return render_template('select-quantity.html', form = form)
#-----
#-----Route for deleting a commodity from shopping cart-----
@main.route('/delete_cart/<id>')
def delete_cart(id):
    comm=goodsCart.query.filter_by(id=id).first()
    if current_user.is_authenticated:
        db.session.delete(comm)
        db.session.commit()
        flash("The commodity has been deleted from the shopping cart!")
        commitiess = goodsCart.query.filter_by(added=True).all()
        return render_template('shopping_cart.html', commodities = commitiess)
    return render_template('shopping_cart.html', commodities = commitiess)
#-----
#-----Route for paying in cash-----
@main.route('/cash_payment/')
def cash_payment():
    commodities = goodsCart.query.filter_by(buyer = current_user.name).all()
    coomm1 = goodsCart.query.filter_by(buyer = current_user.name).first()
    sum = 0
    for comm4 in commodities:
        sum = sum + comm4.price
    return render_template('cash.html', sum = sum, commodities = commodities,
        coomm1=coomm1)
#-----
#-----Route for paying via MPESA-----
@main.route('/mpesa_payment/')
def mpesa_payment():
    commodities = goodsCart.query.filter_by(buyer = current_user.name).all()
    coomm1 = goodsCart.query.filter_by(buyer = current_user.name).first()
    sum = 0
    for comm5 in commodities:
        sum = sum + comm5.price
    return render_template('mpesa.html', sum = sum, commodities = commodities,
        coomm1=coomm1)
#-----
#-----Route for confirming purchase of goods-----
@main.route('/confirm/')
def confirm():
    commodities = goodsCart.query.filter_by(buyer = current_user.name).all()
    for comm6 in commodities:
        commodities2 = Commodity.query.filter_by(id=comm6.id).first()
        quan = commodities2.quantity - comm6.reqQuantity

```

```

if quan ==0:
    change2 = Commodity(name = commodities2.name,
        type = commodities2.type,
        manDate = commodities2.manDate,
        expDate = commodities2.expDate,
        quantity = commodities2.quantity - comm6.reqQuantity,
        oldPrice = commodities2.oldPrice,
        discPrice = commodities2.discPrice,
        percDiscount = commodities2.percDiscount,
        timestamp = commodities2.timestamp,
        inStock = 'No',
        author1 = commodities2.author1)
    change3 = goodsPurchased(name = commodities2.name,
        type = commodities2.type,
        manDate = commodities2.manDate,
        expDate = commodities2.expDate,
        reqQuantity = comm6.reqQuantity,
        price = (commodities2.discPrice)*(comm6.reqQuantity),
        percDiscount = commodities2.percDiscount,
        timestamp = commodities2.timestamp,
        bought = True,
        claimed = False,
        buyer = current_user.name,
        seller = commodities2.author1)
else:
    change2 = Commodity(name = commodities2.name,
        type = commodities2.type,
        manDate = commodities2.manDate,
        expDate = commodities2.expDate,
        quantity = commodities2.quantity - comm6.reqQuantity,
        oldPrice = commodities2.oldPrice,
        discPrice = commodities2.discPrice,
        percDiscount = commodities2.percDiscount,
        timestamp = commodities2.timestamp,
        author1 = commodities2.author1)
    change3 = goodsPurchased(name = commodities2.name,
        type = commodities2.type,
        manDate = commodities2.manDate,
        expDate = commodities2.expDate,
        reqQuantity = comm6.reqQuantity,
        price = (commodities2.discPrice)*(comm6.reqQuantity),
        percDiscount = commodities2.percDiscount,
        timestamp = commodities2.timestamp,
        bought = True,
        claimed = False,
        buyer = current_user.name,
        seller = commodities2.author1)
db.session.add(change2)
db.session.add(change3)
db.session.delete(commodities2)
db.session.delete(comm6)
db.session.commit()
flash("Your order has been confirmed! Please visit our office to claim your goods.")
return render_template('index.html')
#-----
#-----Route for requesting to be an admin-----
@main.route('/become-admin/', methods=['GET', 'POST'])
def become_admin():
    form = PendingForm()
    if form.validate_on_submit():
        issuess = Pending(name = current_user.name,

```

```

        username = current_user.username,
        issue = form.issue.data)
    db.session.add(issue)
    db.session.commit()
    flash('Your request has been submitted! You will be contacted soon.')
    return render_template('index.html')
    return render_template('become_admin.html', form=form)
#-----
#-----Route for viewing requests-----
@main.route('/request/')
def request():
    requests = Pending.query.order_by(Pending.timestamp.desc()).all()
    return render_template('requests.html', requests = requests)
#-----
#-----Route for deleting a request-----
@main.route('/delete-request/<id>', methods=['GET', 'POST'])
def delete_request(id):
    del1=Pending.query.filter_by(id=id).first()
    db.session.delete(del1)
    db.session.commit()
    flash('The request has been deleted!')
    return redirect(url_for('.request'))
#-----
#-----Route for seeing goods bought on admin side-----
@main.route('/bought/', methods=['GET', 'POST'])
def bought():
    bought = goodsPurchased.query.order_by(goodsPurchased.timestamp.desc()).all()
    return render_template('bought.html',bought = bought)
#-----
#-----Route to confirm claim of goods-----
@main.route('/confirm_claim/<id>', methods=['GET', 'POST'])
def confirm_claim(id):
    good1 = goodsPurchased.query.filter_by(id=id).first()
    change4 = goodsPurchased(id = good1.id,
        name = good1.name,
        type = good1.type,
        manDate = good1.manDate,
        expDate = good1.expDate,
        reqQuantity = good1.reqQuantity,
        price = good1.price,
        percDiscount = good1.percDiscount,
        timestamp = good1.timestamp,
        bought = True,
        claimed = True,
        buyer = good1.buyer,
        seller = good1.seller)
    db.session.add(change4)
    db.session.delete(good1)
    db.session.commit()
    flash('The claim has been confirmed!')
    bought = goodsPurchased.query.order_by(goodsPurchased.timestamp.desc()).all()
    return render_template('bought.html', bought = bought)
#-----
#-----Route for showing purchased goods-----
@main.route('/show_bought/<user>')
def show_bought(user):
    comm12 = goodsPurchased.query.filter_by(buyer = user).all()
    sum1 = 0
    for comm in comm12:
        sum1 = sum1+comm.price
    return render_template('showbought.html', commodities = comm12, sum = sum1, user = user)

```

```

#-----Route for showing sold goods-----
@main.route('/show_sold/<user>')
def show_sold(user):
    comm13 = goodsPurchased.query.filter_by(seller = user).all()
    sum2 = 0
    for comm in comm13:
        sum2 = sum2+comm.price
    return render_template('showsold.html', commodities = comm13, sum = sum2, user = user)

#-----Route for seeing out-of-stock goods-----
@main.route('/outstock')
def outstock():
    stock = Commodity.query.filter_by(inStock="No").all()
    return render_template('stock.html', stocks = stock)

#-----Route for deleting goods out of stock-----
@main.route('/delete-stock/<id>')
def delete_stock(id):
    del2 = Commodity.query.filter_by(id=id).first()
    del3 = Img.query.filter_by(commID=id).first()
    if del2:
        db.session.delete(del2)
    if del3:
        db.session.delete(del3)
    db.session.commit()
    stock = Commodity.query.filter_by(inStock="No").all()
    flash('Out of stock commodity has been deleted!')
    return render_template('stock.html', stocks = stock)
    flash('Commodity is no longer available!')
    return render_template('stock.html', stocks = stock)

#-----Search box-----
@main.route('/search/', methods = ['GET', 'POST'])
def search():
    form = SearchForm()
    if form.validate_on_submit():
        bruh = form.searched.data
        posts = Commodity.query.filter(Commodity.name.like('%'+ bruh +'%')).order_by(
            Commodity.timestamp).all()
        usersz = User.query.filter(
            User.name.like('%'+ bruh +'%')).all()
        user = User.query.filter_by(name=current_user.name).first()
        return render_template('search.html', searched = bruh,
            form=form, posts=posts, user=user, users=usersz)
    flash('No such commodity on sale!')
    return render_template('index.html')

#-----Route for sending message to user-----
@main.route('/send_message/<name>', methods = ['GET', 'POST'])
def send_message(name):
    form=SendMessageForm()
    if form.validate_on_submit():
        mess = Dashboard(name = name,
            message = form.message.data)
        db.session.add(mess)
        db.session.commit()
        flash("Your message has been sent successfully!")
        requests = Pending.query.order_by(Pending.timestamp.desc()).all()
        return render_template('requests.html', requests=requests)
    return render_template('send_message.html', form=form)

```

```

#-----Route for uploading images-----
@main.route('/upload', methods=['GET', 'POST'])
def upload():
    from flask import request
    pic = request.files['pic']
    if not pic:
        flash('No picture uploaded')
        return render_template('upload_pics.html')
    #filename = secure_filename(pic.filename)
    mimetype = pic.mimetype
    comm11 = goodsPending.query.order_by(goodsPending.timestamp.desc()).first()
    img = Img(img=pic.read(), mimetype=mimetype, commID=comm11.id)
    db.session.add(img)
    db.session.commit()
    flash('Your image has been uploaded successfully! Upon commodity verification you will be
    messaged by our admins on the next step')
    return render_template('catalog.html')

#-----view full picture-----
@main.route('/view/<id>')
def view(id):
    img = Img.query.filter_by(id=id).order_by(Img.timestamp.desc()).first()
    if not img:
        return 'No image with that ID', 404
    return Response(img.img, mimetype=img.mimetype)

#-----view all pictures-----
@main.route('/view-all/<id>')
def view_all(id):
    img = Img.query.filter_by(commID=id).first()
    comm = Commodity.query.filter_by(id=id).first()
    if not img:
        return 'No image with that ID', 404
    return Response(img.img, mimetype=img.mimetype)

#-----Route for general issues/requests-----
@main.route('/general', methods=['GET', 'POST'])
def general():
    form = GeneralForm()
    if form.validate_on_submit():
        issues = Pending(name = current_user.name,
            username = current_user.username,
            issue = form.issue.data)
        db.session.add(issues)
        db.session.commit()
        flash('Your general concern has been submitted!')
        return redirect(url_for('auth.index2'))
    return render_template('genreq.html', form = form)

#-----Reporting a posted commodity-----
@main.route('/report/<id>', methods = ['GET', 'POST'])
def report(id):
    form = GeneralForm()
    if form.validate_on_submit():
        issues = Pending(name = current_user.name,
            username = current_user.username,
            issue = form.issue.data)
        db.session.add(issues)
        db.session.commit()
        flash('Your report has been submitted!')
        return redirect(url_for('auth.index2'))

```



```

        flash('Please be descriptive! State the name of the seller, the commodity and the issue
with the commodity.')
        return render_template('genreq.html', form = form)
#-----
#-----edit profile route-----
@main.route('/edit-profile2/<id>', methods=['GET', 'POST'])
def edit_profile2(id):
    form = EditProfileForm()
    if form.validate_on_submit():
        old = User.query.filter_by(user_id=id).first()
        new = User(name = form.name.data,
                    username = form.username.data,
                    email = form.email.data,
                    yearOfStudy = form.yearOfStudy.data,
                    regNumber = form.regNumber.data,
                    phoneNumber = form.phoneNumber.data,
                    password = form.password.data,
                    secQuestion = form.secQuestion.data,
                    secAnswer = form.secAnswer.data)
        db.session.delete(old)
        db.session.commit()
        db.session.add(new)
        db.session.commit()
        flash('Your profile has been updated! Please login.')
        return redirect(url_for('auth.login'))
    return render_template('edit_profile.html', form=form)
#-----
#-----Route for rendering rules and regulations-----
@main.route('/rules/')
def rules():
    return render_template('rules.html')
#-----
#-----Route for viewing a specific commodity-----
@main.route('/comview/<id>', methods=['GET', 'POST'])
def comview(id):
    comm78 = goodsPending.query.filter_by(id=id).first()
    comm87 = Commodity.query.filter_by(name=comm78.name).first()
    return render_template('view_comm.html', commodity=comm78, commodity2=comm87)
#-----
#-----Add to Commodities list after approval-----
@main.route('/add/<time>', methods=['GET', 'POST'])
def add(time):
    comm112 = goodsPending.query.filter_by(timestamp=time).first()
    commtype = comm112.type
    if comm112:
        comm122 = Commodity(name = comm112.name,
                            type = comm112.type,
                            manDate = comm112.manDate,
                            expDate = comm112.expDate,
                            quantity = comm112.quantity,
                            oldPrice = comm112.oldPrice,
                            discPrice = comm112.discPrice,
                            percDiscount = comm112.percDiscount,
                            author1 = comm112.author2,
                            inStock = comm112.inStock,
                            addDesc = comm112.addDesc)
        if commtype == 'Foodstuffs':
            issues = Pending(name = comm112.author2,
                             issue = "Foodstuff added! Monitor accordingly.")
            db.session.add(comm122)
            db.session.delete(comm112)

```

```

        db.session.commit()
        flash('The commodity post has been verified!')
        return redirect(url_for('.request'))
    return render_template('index.html')
#-----
#-----See all messages in dashboard-----
@main.route('/see-all/<id>', methods = ['GET', 'POST'])
def see_all(id):
    user = User.query.filter_by(user_id = id).first()
    mess = Dashboard.query.filter_by(name =
user.name).order_by(Dashboard.timestamp.desc()).all()
    return render_template('alldash.html', messages = mess)
#-----
#-----Show commodity picture in pending list-----
@main.route('/viewsss/<id>')
def viewsss(id):
    img = Img.query.filter_by(id=id).order_by(Img.timestamp.desc()).first()
    if not img:
        return 'No image with that ID', 404
    return Response(img.img, mimetype=img.mimetype)
#-----
#-----Deleting a commodity from GoodsPending-----
@main.route('/delpend/<id>', methods=['GET', 'POST'])
def delpend(id):
    comm = goodsPending.query.filter_by(id=id).first()
    img1 = Img.query.filter_by(id=id).order_by(Img.id.desc()).first()
    db.session.delete(comm)
    if img1:
        db.session.delete(img1)
    db.session.commit()
    requests = Pending.query.order_by(Pending.timestamp.desc()).all()
    flash('Pending good has been deleted!')
    return render_template('requests.html', requests = requests)
#-----

```

base.html

```

{% extends "bootstrap/base.html" %}
{% block head %}
{{ super() }}
<link rel="icon" type="image/x-icon" href="{{ url_for('static', filename='favicon1.ico') }}">
<link rel = "stylesheet" href = '/static/main.css/'/>
{% endblock %}
{% block title %}
Mwafunzi
{% endblock %}
{% block navbar %}
<body>
    <div class="navbar navbar-inverse" role="navigation">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle"
data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                {% if current_user.is_authenticated %}
                <a class="navbar-brand" href="{{ url_for('auth.index2') }}">Mwafunzi</a>
                {% else %}
                <a class="navbar-brand">Mwafunzi</a>
            
```

```

        {% endif %}
        {% if current_user.is_authenticated %}
        <a class="navbar-brand" href="{ { url_for('main.user',
username=current_user.username, user = current_user) } }">Profile</a>
        {% endif %}
        {% if current_user.confirmed == True %}
        <a class="navbar-brand" href="/catalog/">Catalog</a>
        {% endif %}
        <a class="navbar-brand" href="/about/">About</a>
        <a class="navbar-brand" href="/contacts/">Contacts</a>
        {% if current_user.role == "Administrator" %}
        <a class="navbar-brand" href="/request/">Admin</a>
        {% endif %}
        {% if current_user.confirmed == True %}
        <a class="navbar-brand" href="/shopping_cart/">Shopping Cart</a>
        <a class="navbar-brand">
            <form method="POST" action="{ { url_for('main.search') } }" class="d-flex">
                { { form.hidden_tag() } }
                <input class="form-control" type="search" placeholder="Search"
                    aria-label="search" name="searched">
            </form>
        </a>
        {% endif %}
    </div>
    <!--Login and Log out navigation bar links----->
    <ul class="nav navbar-nav navbar-right">
        {% if current_user.is_authenticated %}
        <li><a href="{ { url_for('auth.logout') } }">Log Out</a></li>
        {% else %}
        <li><a href="{ { url_for('auth.login') } }">Log In</a></li>
        {% endif %}
    </ul>
    <!------->
</div>
</div>
</body>
{% endblock %}
{% block content %}
<div class="container">
    {% for message in get_flashed_messages() %}
    <div class="alert alert-warning">
        <button type="button" class="close" data-dismiss="alert">&times;</button>
        { { message } }
    </div>
    {% endfor %}
    {% block page_content %}<body></body>{% endblock %}
    {% block asidecontent %}<body></body>{% endblock %}
</div>
<hr>
<div>
    {% block copyright %}
    <p class="centre-text"><b>    © Mwafunzi Platform 2022</b></p>
    {% endblock %}
</div>
{% endblock %}
#-----
index.html
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}Mwafunzi{% endblock %}

```

```

{% block page_content %}
<div>
    {% if current_user.is_authenticated %}
        <div class="page-header">
            <h1>Hello, {{ current_user.username }}! </h1>
        </div>
        <p>Welcome to Mwafunzi!</p>
        <link rel = "stylesheet" href = '/static/main.css/'/>
        {% if current_user.confirmed == True %}
        {% if messages == [] %}
            <div class="page-header">
                <h2>Dashboard</h2>
            </div>
            <p><b>No messages have been sent to you yet....</b></p>
            <div class="page-header">
                <h2>Other Options</h2>
            </div>
            <p>
                Would you like to <b>explore our catalog</b> of goods on sale?
                <a href="/catalog/"> Click here </a>
                to go to Catalog Page
            </p>

            <p>
                Would you like to <b>sell</b> to other students?
                <a href="/add-commodity/"> Click here </a>
                to go to add a commodity for sale
            </p>
            <p>
                Would you like to <b>become an administrator</b> or <b>relinquish the
role</b>?
                <a href="/become-admin/"> Click here </a>
                to submit a request
            </p>
            <p>
                <b>TIP:</b> Use the search bar to find the commodities you seek easily!
            </p>
            <hr>
            <link rel = "stylesheet" href = '/static/main.css/'/>
            
        {% else %}
            <div class="page-header">
                <h2>Dashboard</h2>
            </div>
            <div class="page-header">
                <h3>Most Recent</h3>
            </div>
            <table class="table">
                <tr>
                    <th>Sender</th>
                    <th>Time Sent</th>
                    <th>Message</th>
                </tr>
                <tr>
                    <td>{{ messages.sender }}</td>
                    <td>{{ messages.timestamp }}</td>
                    <td>{{ messages.message }}</td>
                </tr>
            </table>
        {% endblock %}

```

```

        <a class="btn btn-default" href="{ { url_for('main.see_all', id =
current_user.user_id) }}">
            See all
        </a>
<hr>
<div class="page-header">
    <h2>Other Options</h2>
</div>
<p>
    Would you like to <b>explore our catalog</b> of goods on sale?
    <a href="/catalog/"> Click here </a>
    to go to Catalog Page
</p>

<p>
    Would you like to <b>sell</b> to other students?
    <a href="/add-commodity/"> Click here </a>
    to go to add a commodity for sale
</p>
<p>
    Would you like to <b>become an administrator</b> or <b>relinquish the
role</b>?
    <a href="/become-admin/"> Click here </a>
    to submit a request
</p>
<p>
    <b>TIP:</b> Use the search bar to find the commodities you seek easily!
</p>
<hr>
<link rel = "stylesheet" href = '/static/main.css/'/>

    {% endif %}
    {% endif %}
{% else %}
    <div class="page-header">
        <h1>Hello, Stranger! </h1>
    </div>
    <p> Please log in to obtain full functionality</p>
    <a href="{ { url_for('auth.login') }}"> Click here to log in </a>
    <p> New User?</p>
    <a href="{ { url_for('auth.register') }}"> Click here to register</a>
    {% endif %}
</div>
{% endblock %}
#-----
about.html
{% extends "base.html" %}
{% block title %}About Page{% endblock %}
{% block page_content %}
<link rel = "stylesheet" href = '/static/main.css/'/>
<div class="page-header">
    <h1>Mwafunzi Discount Platform</h1>
</div>
<div>
    <p>
        Established in 2022, Mwafunzi is an online discount platform
        that serves the students of the University of Nairobi by
        providing goods at discounts.
    </p>

```

```

    </p>
</div>
<div class="page-header">
    <h1>Opening Hours</h1>
</div>
<div>
    <p>
        Our office in Mahatma Gandhi Wing is <b>open on weekdays from 8.00AM to
        5.30PM</b> and on <b>weekends from 10.00AM to 4.00PM.</b>
    </p>
</div>
<hr>
<div>
    <img src = "{{ url_for('static', filename = 'image17.jpg') }}" class="center"/>
</div>
{% endblock %}
#-----

```

add_commodity.html

```

{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}Mwafunzi - Add Commodity{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1>Add Commodity</h1>
</div>
<div>
    {{ wtf.quick_form(form) }}
</div>
{% endblock %}
#-----

```

alldash.html

```

{% extends "base.html" %}
{% block title %} Mwafunzi - All Messages{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1> All messages </h1>
</div>
<table class="table">
    <tr>
        <th>Sender</th>
        <th>Time Sent</th>
        <th>Message</th>
    </tr>
    {% for message in messages %}
    <tr>
        <td>{{message.sender}}</td>
        <td>{{message.timestamp}}</td>
        <td>{{message.message}}</td>
    </tr>
    {% endfor %}
</table>
{% endblock %}
#-----

```

catalog.html

```

{% extends "base.html" %}
{% block title %}Catalog Page{% endblock %}
{% block content %}
<div class="container">
    {% for message in get_flashed_messages() %}
    <div class="alert alert-warning">

```

```

        <button type="button" class="close" data-dismiss="alert">&times;</button>
        {{ message }}
    </div>
    {% endfor %}
    <link rel="stylesheet" href="/static/main.css"/>
    <div class="page-header">
        <h1>Catalog Page</h1>
    </div>
    {% block asidecontent %}
    <div class="goods-link">
        <p>Here at Mwafunzi, we are dedicated to providing students with quality goods
            that pass through a thorough inspection phase before reaching the consumer.
        </p>
        <p>Here are the types of goods we offer:</p>
        <img src = "{{ url_for('static', filename='image5.png') }}" width="50"
height="50"/> <a class="link good-type" href="/catalog/electronics/"> Electronics</a>
        <img src = "{{ url_for('static', filename='image6.png') }}" width="50" height="50"
class="link good-type2"/> <a href="/catalog/foodstuffs/"> Foodstuffs</a>
        <img class="link good-type2" src = "{{ url_for('static', filename='image8.png') }}"
width="50" height="50"/> <a class="link good-type" href="/catalog/stationery/">
Stationery</a>
        <img src = "{{ url_for('static', filename='image7.jpg') }}" width="50" height="50"
class="link good-type2"/> <a href="/catalog/cutlery/"> Cutlery</a>
        <p>
            Or would you like to add a commodity for sale instead?
        </p>
        <img src = "{{ url_for('static', filename='image10.jpg') }}" width="50"
height="50"/><a class="link add-commodity" href="/add-commodity/"> Add Commodity</a>
    </div>
    {% endblock %}
</div>
{% endblock %}
#-----

```

contacts.html

```

{% extends "base.html" %}
{% block title %}Contacts Page{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1>Mwafunzi Discount Platform</h1>
</div>
<div>
    <div class="page-header">
        <h2>Address</h2>
    </div>
    <div>
        <ul>
            <li>P.O BOX 1240-00100 GPO</li>
            <li>Nairobi</li>
            <li>Tel: +254701018899</li>
            <li>Twitter: Mwafunzi Discount Platform</li>
        </ul>
    </div>
    <div class="page-header">
        <h2>Directions</h2>
    </div>
    <ul>
        <li>Mwafunzi Building, 1st Floor</li>
        <li>Next to Mahatma Gandhi Building</li>
        <hr>
        
    </ul>
</div>
</div>
{% endblock %}
#-----
electronics.html
{% extends "base.html" %}
{% block title %}Mwafunzi - Electronics Page{% endblock %}
{% block asidecontent %}
<div class="page-header">
    <h3>Electronic commodities on sale</h3>
</div>
<link rel="stylesheet" href="/static/main.css"/>
<ul class="commodities">
    {% if commodities == [] %}
    <p> Sorry! Currently, there are no electronics on sale</p>
    <hr>
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>
    {% elif current_user.role == 'Administrator' %}
    <div class="banner">
        
    </div>
    <hr>
    {% for commodity in commodities %}
    
    <div class="final">
        <div class="profile-thumbnail">
            
        </div>
        <div>
            <a class="commodity-owner" href="{{ url_for('.user',
username=commodity.author.username) }}">Seller : {{ commodity.author.name }}</a>
        </div>
        <div class="commodity-name"><b>Commodity name :</b> {{ commodity.name }}</div>
        <div class="commodity-type"><b>Commodity Type :</b> {{ commodity.type }}</div>
        <div class="commodity-quantity"><b>Quantity on sale :</b>
            {{ commodity.quantity }}</div>
        <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b>
            {{ commodity.oldPrice }}</div>
        <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings :</b>
            {{ commodity.discPrice }}</div>
        <div class="commodity-discount"><b>Percentage Discount :</b>
            {{ commodity.percDiscount }}%</div>
        <div class="commodity-stock"><b>In Stock :</b>
            {{ commodity.inStock }}</div>
        <div class="commodity-description"><b>Additional Description :</b>
            {{ commodity.addDesc }}</div>
        <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity.id) }}">
            Add to Cart</a>
        <a>
        <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity.id) }}"
            target="_blank" rel="noopener noreferrer">
            See full picture</a>
        </a>
    </div>
    <hr>

```



```

</div>
{% endfor %}
{% else %}

{% for commodity in commodities %}
    <div>
        
        <div class="final">
            <div class="profile-thumbnail">
                
            </div>
            <div class="commodity-author"><b>Seller :</b> {{ commodity.author.name
}}</div>
            <div class="commodity-name"><b>Commodity name :</b> {{ commodity.name
}}</div>
            <div class="commodity-type"><b>Commodity Type :</b> {{ commodity.type
}}</div>
            <div class="commodity-quantity"><b>Quantity on sale :</b> {{
commodity.quantity }}</div>
            <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :
:</b> {{ commodity.oldPrice }}</div>
            <div class="commodity-price"><b>Discounted Price per unit in Kenyan
Shillings :</b> {{ commodity.discPrice }}</div>
            <div class="commodity-discount"><b>Percentage Discount :</b> {{
commodity.percDiscount }}</div>
            <div class="commodity-stock"><b>In Stock :</b> {{ commodity.inStock
}}</div>
            <div class="commodity-description"><b>Additional Description :</b> {{
commodity.addDesc }}</div>
            <a class="btn btn-default" href="{{ url_for('.add_to_cart',
id=commodity.id) }}">
                Add to Cart</a>
            </a>
            <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity.id)
}}" target="_blank" rel="noopener noreferrer">
                See full picture</a>
            </a>
            <a class="btn btn-default" href="{{ url_for('.report', id=commodity.id)
}}" target="_blank" rel="noopener noreferrer">
                Report
            </a>
            <hr>
        </div>
    </div>
{% endfor %}
{% endif %}
</ul>
{% endblock %}
#-----

```

cutlery.html

```

{% extends "base.html" %}
{% block title %}Mwafunzi - Cutlery Page{% endblock %}
{% block asidecontent %}
<div class="page-header">
    <h3>Cutlery commodities on sale</h3>
</div>
<link rel="stylesheet" href="/static/main.css"/>

```

```

<ul class="commodities">
  {% if commodities == [] %}
  <p> Sorry! Currently, there are no cutlery on sale</p>
  <hr>
  <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>

  {% elif current_user.role == 'Administrator' %}
  
  <hr>
  {% for commodity in commodities %}
    
    <div class="final">
      <div class="profile-thumbnail">
        
      </div>
      <a class="commodity-owner" href="{{ url_for('.user',
username=commodity.author.username) }}">Seller : {{ commodity.author.name }}</a>
      <div class="commodity-name"><b>Commodity name :</b> {{ commodity.name }}</div>
      <div class="commodity-type"><b>Commodity Type :</b> {{ commodity.type }}</div>
      <div class="commodity-quantity"><b>Quantity on sale :</b> {{ commodity.quantity
    }}</div>
      <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b> {{
commodity.oldPrice }}</div>
      <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings
:</b> {{ commodity.discPrice }}</div>
      <div class="commodity-discount"><b>Percentage Discount :</b> {{
commodity.percDiscount }}%</div>
      <div class="commodity-stock"><b>In Stock :</b> {{ commodity.inStock }}</div>
      <div class="commodity-description"><b>Additional Description :</b> {{
commodity.addDesc }}</div>
      <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity.id) }}">
        Add to Cart
      </a>
      <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity.id) }}"
target="_blank" rel="noopener noreferrer">
        See full picture
      </a>
    </div>
  </div>
  {% endfor %}
  {% else %}
  
  <hr>
  {% for commodity in commodities %}
    
    <div class="final">
      <div class="profile-thumbnail">
        
      </div>
      <div class="commodity-author"><b>Seller :</b> {{ commodity.author.name }}</div>
      <div class="commodity-name"><b>Commodity name :</b> {{ commodity.name }}</div>
      <div class="commodity-type"><b>Commodity Type :</b> {{ commodity.type }}</div>

```

```

        <div class="commodity-quantity"><b>Quantity on sale :</b> {{ commodity.quantity
}}</div>
        <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b> {{
commodity.oldPrice }}</div>
        <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings
:</b> {{ commodity.discPrice }}</div>
        <div class="commodity-discount"><b>Percentage Discount :</b> {{
commodity.percDiscount }}%</div>
        <div class="commodity-stock"><b>In Stock :</b> {{ commodity.inStock }}</div>
        <div class="commodity-description"><b>Additional Description :</b> {{
commodity.addDesc }}</div>

        <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity.id) }}">
            Add to Cart</a>
        </a>
        <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity.id) }}"
target="_blank" rel="noopener noreferrer">
            See full picture
        </a>
        <a class="btn btn-default" href="{{ url_for('.report', id=commodity.id) }}"
target="_blank" rel="noopener noreferrer">
            Report
        </a>
    </hr>
</div>
    {% endfor %}
    {% endif %}
</ul>
{% endblock %}
<u>foodstuffs.html</u>
{% extends "base.html" %}
{% block title %}Mwafunzi - Foodstuffs Page{% endblock %}
{% block asidecontent %}
<link rel = "stylesheet" href = '/static/main.css/'/>
<div class="page-header">
    <h3>Foodstuffs on sale</h3>
</div>
<ul class="commodities">
    {% if commodities == [] %}
    <p> Sorry! Currently, there are no foodstuffs on sale</p>
    <hr>
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>

    {% elif current_user.role == 'Administrator' %}
    
    <hr>
    {% for commodity1 in commodities %}
    
    <div class="final">
        <div class="profile-thumbnail">
            
        </div>
        <a class="commodity-owner" href="{{ url_for('.user',
username=commodity1.author.username) }}">Seller : {{ commodity1.author.name }}</a>
        <div class="commodity-name"><b>Commodity name :</b> {{ commodity1.name }}</div>
        <div class="commodity-type"><b>Commodity Type :</b> {{ commodity1.type }}</div>

```

```

        <div class="commodity-manDate"><b>Manufacture Date :</b> {{ commodity1.manDate
}}</div>
        <div class="commodity-expDate"><b>Expiry Date :</b> {{ commodity1.expDate }}</div>
        <div class="commodity-quantity"><b>Quantity on sale :</b> {{ commodity1.quantity
}}</div>
        <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b> {{
commodity1.oldPrice }}</div>
        <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings
:</b> {{ commodity1.discPrice }}</div>
        <div class="commodity-discount"><b>Percentage Discount :</b> {{
commodity1.percDiscount }}%</div>
        <div class="commodity-stock"><b>In Stock :</b> {{ commodity1.inStock }}</div>
        <div class="commodity-description"><b>Additional Description :</b> {{
commodity1.addDesc }}</div>

        <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity1.id) }}">
            Add to Cart</a>
        <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity1.id) }}"
target="_blank" rel="noopener noreferrer">
            See full picture
        </a>
        <hr>
    </div>
    {% endfor %}
    {% else %}
    
    <hr>
    {% for commodity1 in commodities %}
        
        <div class="final">
            <div class="profile-thumbnail">
                
            </div>
            <div class="commodity-author"><b>Seller :</b> {{ commodity1.author.name }}</div>
            <div class="commodity-name"><b>Commodity name :</b> {{ commodity1.name }}</div>
            <div class="commodity-type"><b>Commodity Type :</b> {{ commodity1.type }}</div>
            <div class="commodity-manDate"><b>Manufacture Date :</b> {{ commodity1.manDate
}}</div>
            <div class="commodity-expDate"><b>Expiry Date :</b> {{ commodity1.expDate }}</div>
            <div class="commodity-quantity"><b>Quantity on sale :</b> {{ commodity1.quantity
}}</div>
            <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b> {{
commodity1.oldPrice }}</div>
            <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings
:</b> {{ commodity1.discPrice }}</div>
            <div class="commodity-discount"><b>Percentage Discount :</b> {{
commodity1.percDiscount }}%</div>
            <div class="commodity-stock"><b>In Stock :</b> {{ commodity1.inStock }}</div>
            <div class="commodity-description"><b>Additional Description :</b> {{
commodity1.addDesc }}</div>

            <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity1.id) }}">
                Add to Cart
            </a>
            <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity1.id) }}"
target="_blank" rel="noopener noreferrer">
                See full picture
    
```

```

        </a>
        <a class="btn btn-default" href="{ { url_for('.report', id=commodity1.id) }}"
target="_blank" rel="noopener noreferrer">
            Report
        </a>
    </div>
    <hr>
</div>
{% endfor %}
{% endif %}
</ul>
{% endblock %}
stationery.html
{% extends "base.html" %}
{% block title %}Mwafunzi - Stationery Page{% endblock %}
{% block asidecontent %}
<div class="page-header">
    <h3>Stationery commodities on sale</h3>
</div>
<link rel="stylesheet" href="/static/main.css"/>
<ul class="commodities">
    {% if commodities == [] %}
    <p> Sorry! Currently, there are no stationery on sale</p>
    <hr>
    <img src = "{ { url_for('static', filename = 'image3.jpg') }}" class="center"/>

    {% elif current_user.role == 'Administrator' %}
    
    <hr>
    {% for commodity in commodities %}
    
    <div class="final">
        <div class="profile-thumbnail">
            
        </div>
        <a class="commodity-owner" href="{ { url_for('.user',
username=commodity.author.username) }}">Seller : { { commodity.author.name }}</a>
        <div class="commodity-name"><b>Commodity name :</b> { { commodity.name }}</div>
        <div class="commodity-type"><b>Commodity Type :</b> { { commodity.type }}</div>
        <div class="commodity-quantity"><b>Quantity on sale :</b> { { commodity.quantity
}}</div>
        <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b> { {
commodity.oldPrice }}</div>
        <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings :</b> { {
commodity.discPrice }}</div>
        <div class="commodity-discount"><b>Percentage Discount :</b> { { commodity.percDiscount
}}%</div>
        <div class="commodity-stock"><b>In Stock :</b> { { commodity.inStock }}</div>
        <div class="commodity-description"><b>Additional Description :</b> { {
commodity.addDesc }}</div>

        <a class="btn btn-default" href="{ { url_for('.add_to_cart', id=commodity.id) }}">
            Add to Cart</a>
        </a>
        <a class="btn btn-default" href="{ { url_for('.view_all', id=commodity.id) }}"
target="_blank" rel="noopener noreferrer">
            See full picture

```

```

        </a>
        <hr>
    </div>
    {% endfor %}
    {% else %}
    
    {% for commodity in commodities %}
    
    <div class="final">
        <div class="profile-thumbnail">
            
        </div>
        <div class="commodity-author"><b>Seller :</b> {{ commodity.author.name }}</div>
        <div class="commodity-name"><b>Commodity name :</b> {{ commodity.name }}</div>
        <div class="commodity-type"><b>Commodity Type :</b> {{ commodity.type }}</div>
        <div class="commodity-quantity"><b>Quantity on sale :</b> {{ commodity.quantity
    }}</div>
        <div class="commodity-oprice"><b>Old Price per unit in Kenyan Shillings :</b> {{
commodity.oldPrice }}</div>
        <div class="commodity-price"><b>Discounted Price per unit in Kenyan Shillings :</b> {{
commodity.discPrice }}</div>
        <div class="commodity-discount"><b>Percentage Discount :</b> {{ commodity.percDiscount
    }}%</div>
        <div class="commodity-stock"><b>In Stock :</b> {{ commodity.inStock }}</div>
        <div class="commodity-description"><b>Additional Description :</b> {{
commodity.addDesc }}</div>

        <a class="btn btn-default" href="{{ url_for('.add_to_cart', id=commodity.id) }}">
            Add to Cart</a>
        </a>
        <a class="btn btn-default" href="{{ url_for('.view_all', id=commodity.id) }}"
target="_blank" rel="noopener noreferrer">
            See full picture
        </a>
        <a class="btn btn-default" href="{{ url_for('.report', id=commodity.id) }}"
target="_blank" rel="noopener noreferrer">
            Report
        </a>
        <hr>
    </div>
    {% endfor %}
    {% endif %}
</ul>
{% endblock %}
<u>genreq.html</u>
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}Mwafunzi - General Requests {% endblock %}
{% block page_content %}
<div class="page-header">
    <h1>Submit Requests</h1>
</div>
<div class="col-md-4">
    {{ wtf.quick_form(form) }}
</div>
{% endblock %}

```

#-----

recovery.html

```
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}Mwafunzi - Answer question{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1>Answer</h1>
</div>
<div>
    <p>{{ question }}</p>
</div>
<div>
    {{ wtf.quick_form(form) }}
</div>
{% endblock %}
```

#-----

send_message.html

```
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}Mwafunzi - Send Message{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1>Send Message</h1>
</div>
<div class="col-md-4">
    {{ wtf.quick_form(form) }}
</div>
{% endblock %}
```

#-----

shopping_cart.html

```
{% extends "base.html" %}
{% block title %}Shopping Cart Page{% endblock %}
{% block page_content %}
<link rel = "stylesheet" href = '/static/main.css/'/>
<div class="page-header">
    <h1>Shopping Cart</h1>
</div>
<ul class="commodities">
    {% if commodities == [] %}
    <p> Sorry! Currently, there are no items in cart...</p>
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>
    {% else %}
    <p>
        Below is the list of commodities you want to buy:
    </p>
    <table class="table">
        <tr>
            <th>Commodity ID</th>
            <th>Seller</th>
            <th>Commodity name</th>
            <th>Commodity Type</th>
            <th>Quantity to be bought</th>
            <th>Price in Kenyan Shillings</th>
            <th>Option 1</th>
            <th>Option 2</th>
        </tr>
        {% for commodity3 in commodities %}
        <tr>
```

```

        <td><div class="commodity-id">{{ commodity3.id }}</div></td>
        <td><div class="commodity-author">{{ commodity3.seller }}</div></td>
        <td><div class="commodity-name">{{ commodity3.name }}</div></td>
        <td><div class="commodity-type">{{ commodity3.type }}</div></td>
        <td><div class="commodity-quantity">{{ commodity3.reqQuantity }}</div></td>
        <td><div class="commodity-price">{{ commodity3.price }}</div></td>
        <td>
            <a class="btn btn-default" href="{{ url_for('.delete_cart', id =
commodity3.id) }}">
                Remove from shopping Cart
            </a>
        </td>
        <td>
            <a class="btn btn-default" href="{{ url_for('.change_quantity', id =
commodity3.id) }}">
                Change quantity
            </a>
        </td>
    </tr>
{% endfor %}
</table>
<a class="btn btn-default" href="{{ url_for('.payment', commodities = commodities) }}">
    Pay for goods
</a>
{% endif %}
</ul>
{% endblock %}

```

#-----

payment_page.html

```

{% extends "base.html" %}
{% block title %}Payment Page{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1>Payment Page</h1>
</div>
<div>
    <table class="table">
        <tr>
            <th>Commodity ID</th>
            <th>Commodity name</th>
            <th>Price in Kenyan Shillings</th>
        </tr>
        {% for commodity in commodities %}
        <tr>
            <td>
                <div class="commodity-id">{{ commodity.id }}</div>
            </td>
            <td>
                <div class="commodity-name">{{ commodity.name }}</div>
            </td>
            <td>
                <div class="commodity-price">{{ commodity.price }}</div>
            </td>
        </tr>
        {% endfor %}
    </table>
    <a class="btn btn-default" href="{{ url_for('.mpesa_payment') }}">
        Pay via MPESA
    </a>
    <a class="btn btn-default" href="{{ url_for('.cash_payment') }}">

```



```

        Pay in cash
    </a>
</div>
{% endblock %}

upload_pics.html
{% extends "base.html" %}
{% block title %}Mwafunzi - Upload pictures{% endblock %}
{% block page_content %}
<div class="page-header">
    <h1> Upload pictures of the commodity </h1>
</div>
<form action="/upload" enctype="multipart/form-data" method="POST">
    <input type="file" name="pic"/>
    <input type="submit" value="Upload a file"/>
</form>
{% endblock %}
#-----

user.html
{% extends "base.html" %}
{% block title %}Mwafunzi - {{ user.username }}{% endblock %}
{% block page_content %}
<div>
    
    <div class="profile-header">
        <h1>{{ user.username }}</h1>
    </div>
    {% if user.username or user.location %}
    <p>
        <ol><b>Name :</b> {% if user.name %}{{ user.name }}{% endif %}</ol>
        <ol><b>Email :</b> {% if user.email %}{{ user.email }}{% endif %}</ol>
        <ol><b>Year of Study :</b> {% if user.yearOfStudy %}{{ user.yearOfStudy }}{% endif
%</ol>
        <ol><b>Registration Number :</b> {% if user.regNumber %}{{ user.regNumber }}{% endif
%</ol>
        <ol><b>Phone Number :</b> {% if user.phoneNumber %}{{ user.phoneNumber }}{% endif
%</ol>
    </p>
</div>
{% if role == 'Administrator' %}
<a class="btn btn-default" href="{{ url_for('.edit_profile_admin', id=user.user_id) }}">
Edit Profile [Admin]
</a>
<a class="btn btn-default" href="{{ url_for('.send_message', name = user.name) }}">Send
Message</a>
<a class="btn btn-default" href="{{ url_for('main.see_all', id = user.user_id) }}">
    See user's dashboard messages
</a>
{% else %}
<a class="btn btn-default" href="{{ url_for('.edit_profile') }}">
Edit Profile
</a>
<a class="btn btn-default" href="{{ url_for('.general') }}">
    Submit general request to admins
</a>
{% endif %}
{% endif %}
{% endblock %}
{% block asidecontent %}
<link rel = "stylesheet" href = '/static/main.css/'/>
<div class="page-header">

```

```

    <h3>Commodities on sale by {{ user.username }}</h3>
</div>
<ul class="commodities">
    {% if commodities == [] %}
    <p> Sorry! Currently, you have no items on sale...</p>
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>
    {% else %}
    <table class="table">
        <tr>
            <th>Photo</th>
            <th> Seller </th>
            <th> Commodity name </th>
            <th> Commodity Type </th>
            <th> Quantity on sale </th>
            <th> Price per unit </th>
            <th> Percentage Discount </th>
            <th> In Stock </th>
            <th> Option </th>
        </tr>
        {% for commodity in commodities %}
        <tr>
            <td>
                
            </td>
            <td>
                <div class="commodity-author">{{ commodity.author.username }} </div>
            </td>
            <td>
                <div class="commodity-name">{{ commodity.name }} </div>
            </td>
            <td>
                <div class="commodity-type">{{ commodity.type }} </div>
            </td>
            <td>
                <div class="commodity-quantity">{{ commodity.quantity }} </div>
            </td>
            <td>
                <div class="commodity-price">Ksh {{ commodity.discPrice }} </div>
            </td>
            <td>
                <div class="commodity-discount">{{ commodity.percDiscount }}% </div>
            </td>
            <td>
                <div class="commodity-stock">{{ commodity.inStock }} </div>
            </td>
            <td>
                <div class="commodity-delete">
                    <a class="btn btn-default" href="{{ url_for('.delete', id = commodity.id) }}">
                        Delete commodity from sale
                    </a>
                </div>
            </td>
        </tr>
        {% endfor %}
    </table>
    {% endif %}
</ul>
{% if current_user.role == 'Administrator' %}
<div class="page-header">

```

```

    <h3>Commodities pending by {{ user.username }}</h3>
</div>
<ul class="commodities">
    {% if bout1 == [] %}
    <p> Sorry! Currently, no pending items...</p>
    <img src = "{{ url_for('static', filename = 'image3.jpg') }}" class="center"/>
    {% else %}
    <table class="table">
        <tr>
            <th>Photo</th>
            <th> Seller </th>
            <th> Commodity name </th>
            <th> Commodity Type </th>
            <th> Quantity on sale </th>
            <th> Price per unit </th>
            <th> Percentage Discount </th>
            <th> In Stock </th>
            <th> Option </th>
        </tr>
        {% for boutss in bout1 %}
        <tr>
            <td>
                
            </td>
            <td>
                <div class="commodity-author">{{ boutss.author2 }} </div>
            </td>
            <td>
                <a href="{{ url_for('.comview', id = boutss.id) }}" target="_blank"
rel="noopener noreferrer">
                    <div class="commodity-name">{{ boutss.name }} </div>
                </a>
            </td>
            <td>
                <div class="commodity-type">{{ boutss.type }} </div>
            </td>
            <td>
                <div class="commodity-quantity">{{ boutss.quantity }} </div>
            </td>
            <td>
                <div class="commodity-price">Ksh {{ boutss.discPrice }} </div>
            </td>
            <td>
                <div class="commodity-discount">{{ boutss.percDiscount }}% </div>
            </td>
            <td>
                <div class="commodity-stock">{{ boutss.inStock }} </div>
            </td>
            <td>
                <div class="commodity-delete">
                    <a class="btn btn-default" href="{{ url_for('.delpend', id = boutss.id) }}">
                        Delete
                    </a>
                </div>
            </td>
        </tr>
        {% endfor %}
    </table>
    {% endif %}

```

```

</ul>
{% endif %}
{% if user.confirmed == True %}
    <div class="page-header">
        <h3>Commodities bought by {{ user.username }}</h3>
    </div>
    <div>
        <a class="btn btn-default" href="{{ url_for('.show_bought', user = user.name) }}"> See
records</a>
    </div>
    <div class="page-header">
        <h3>Commodities sold by {{ user.username }}</h3>
    </div>
    <div>
        <a class="btn btn-default" href="{{ url_for('.show_sold', user = user.name) }}"> See
records</a>
    </div>
{% endif %}
{% endblock %}
#-----

```

models.py

```

#importing the necessary modules
import os
from flask_sqlalchemy import SQLAlchemy
from flask import Flask
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import UserMixin
from itsdangerous import TimedSerializer as Serializer
from flask import current_app
from datetime import datetime
import hashlib
from flask import request
#database configuration
basedir = os.path.abspath(os.path.dirname(__file__))
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = \
    'sqlite:/// ' + os.path.join(basedir, 'data.sqlite')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
#-----CLASS User-----
class User(UserMixin, db.Model):
    __tablename__ = 'users'
    user_id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), index=True)
    username=db.Column(db.String(64))
    email=db.Column(db.String(128), index=True)
    yearOfStudy= db.Column(db.String(11), index=True)
    regNumber = db.Column(db.String(64), index=True)
    phoneNumber = db.Column(db.Integer, unique=True, index=True)
    password=db.Column(db.String(128), index=True)
    #Password hashing in the User Model
    password_hash = db.Column(db.String(128))
    confirmed = db.Column(db.Boolean, default=False)
    avatar_hash = db.Column(db.String(32))
    role = db.Column(db.String(20), default = "User")
    secQuestion = db.Column(db.Text)
    secAnswer = db.Column(db.Text)
    commodity = db.relationship('Commodity', backref='author', lazy='dynamic')
    goodsAdded = db.relationship('goodsCart', backref='purchaser', lazy='dynamic')
    goodsPurchased = db.relationship('goodsPurchased', backref='purchaserer', lazy='dynamic')

```

```

pendingIssues = db.relationship('Pending', backref='issues', lazy='dynamic')
idss = db.relationship('Dashboard', backref='iss', lazy='dynamic')
idss2 = db.relationship('goodsPending', backref='iss2', lazy='dynamic')
def __repr__(self):
    return '<User %r>' % self.username
@property
def password(self):
    raise AttributeError('password is not a readable attribute')
@password.setter
def password(self, password):
    self.password_hash = generate_password_hash(password)
def verify_password(self, password):
    return check_password_hash(self.password_hash, password)
#-----Gravatar URL generation-----
    if self.email is not None and self.avatar_hash is None:
        self.avatar_hash = self.gravatar_hash()
def gravatar_hash(self):
    return hashlib.md5(self.email.lower().encode('utf-8')).hexdigest()
def gravatar(self, size=100, default='identicon', rating='g'):
    if request.is_secure:
        url = 'https://secure.gravatar.com/avatar'
    else:
        url = 'http://www.gravatar.com/avatar'
    hash = self.avatar_hash or self.gravatar_hash()
    return '{url}/{hash}?s={size}&d={default}&r={rating}'.format(
        url=url, hash=hash, size=size, default=default, rating=rating)
#-----
#--evaluating whether a user has a given permission(copied to __init__.py)-----
def can(self, perm):
    return self.role is not None and self.role.has_permission(perm)
def is_administrator(self):
    return self.can(Permission.ADMIN)
def gravatar(self, size=100, default='identicon', rating='g'):
    url = 'https://secure.gravatar.com/avatar'
    hash = hashlib.md5(self.email.lower().encode('utf-8')).hexdigest()
    return '{url}/{hash}?s={size}&d={default}&r={rating}'.format(
        url=url, hash=hash, size=size, default=default, rating=rating)
#-----
#-----Commodity Model-----
class Commodity(db.Model):
    __tablename__ = 'commodity'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
    type = db.Column(db.String(64))
    manDate = db.Column(db.String(64))
    expDate = db.Column(db.String(64))
    quantity = db.Column(db.Integer)
    oldPrice = db.Column(db.Integer)
    discPrice = db.Column(db.Integer)
    percDiscount = db.Column(db.Integer)
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    inStock = db.Column(db.String(5), default = "Yes")
    author1 = db.Column(db.Text, db.ForeignKey('users.name'))
    addDesc = db.Column(db.Text, nullable = True)
    ids = db.relationship('Img', backref='isses', lazy='dynamic')
#-----goodsCart Model-----
class goodsCart(db.Model):
    __tablename__ = 'goodsCart'
    id = db.Column(db.Integer, primary_key=True, unique = True)
    name = db.Column(db.Text)
    type = db.Column(db.String(64))

```

```

manDate = db.Column(db.String(64))
expDate = db.Column(db.String(64))
reqQuantity = db.Column(db.Integer)
price = db.Column(db.Integer)
percDiscount = db.Column(db.Integer)
timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
added = db.Column(db.Boolean, default=True)
buyer = db.Column(db.Integer, db.ForeignKey('users.name'))
seller = db.Column(db.Text)
#-----
#-----goodsPurchased Model-----
class goodsPurchased(db.Model):
    __tablename__ = 'goodsPurchased'
    id = db.Column(db.Integer, primary_key=True, unique = True)
    name = db.Column(db.Text)
    type = db.Column(db.String(64))
    manDate = db.Column(db.String(64))
    expDate = db.Column(db.String(64))
    reqQuantity = db.Column(db.Integer)
    price = db.Column(db.Integer)
    percDiscount = db.Column(db.Integer)
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    bought = db.Column(db.Boolean, default=True)
    claimed = db.Column(db.Boolean, default=False)
    buyer = db.Column(db.Integer, db.ForeignKey('users.name'))
    seller = db.Column(db.Text)
#-----
#-----Pending Model-----
class Pending(db.Model):
    __tablename__ = 'pending'
    id = db.Column(db.Integer, primary_key=True, unique = True)
    name = db.Column(db.Text)
    username = db.Column(db.Text)
    issue = db.Column(db.Text)
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    author_id = db.Column(db.Integer, db.ForeignKey('users.user_id'))
#-----
#-----Class for uploading pictures-----
class Img(db.Model):
    __tablename__ = 'images'
    id = db.Column(db.Integer, primary_key=True, index=True)
    img = db.Column(db.Text, nullable=True)
    mimetype = db.Column(db.Text, nullable = False)
    timestamp = db.Column(db.DateTime, default = datetime.utcnow)
    commID = db.Column(db.Integer, db.ForeignKey('commodity.id'))
#-----
#-----Class for dashboard messages-----
class Dashboard(db.Model):
    __tablename__ = 'dashboard'
    id = db.Column(db.Integer, primary_key=True, index=True)
    name = db.Column(db.String(50))
    sender = db.Column(db.String(50), default="Admin")
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    message = db.Column(db.Text)
    use_id = db.Column(db.Integer, db.ForeignKey('users.user_id'))
#-----
#-----CommodityPending Model-----
class goodsPending(db.Model):
    __tablename__ = 'goodspending'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)

```

```

type = db.Column(db.String(64))
manDate = db.Column(db.String(64))
expDate = db.Column(db.String(64))
quantity = db.Column(db.Integer)
oldPrice = db.Column(db.Integer)
discPrice = db.Column(db.Integer)
percDiscount = db.Column(db.Integer)
timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
inStock = db.Column(db.String(5), default = "Yes")
author2 = db.Column(db.Integer, db.ForeignKey('users.name'))
addDesc = db.Column(db.Text, nullable = True)
#-----
oplat.py
#main script
import os
from app import create_app, db
from app.models import User#, Role
from flask_migrate import Migrate
from flask import Flask
from app.main.forms import SearchForm
app = create_app(os.getenv('FLASK_CONFIG') or 'default')
migrate = Migrate(app, db)
@app.shell_context_processor
def make_shell_context():
    return dict(db=db, User=User#, Role=Role)
#-----Route for passing stuff into the HTML file containing search form-----
@app.context_processor
def base():
    form = SearchForm()
    return dict(form=form)
#-----

```