**The Project Report on**

# ONLINE GIFT STORE USING DJANGO

**Submitted To**

**Acharya Nagarjuna University**

**A project report submitted in the partial fulfillment of the requirements for the**

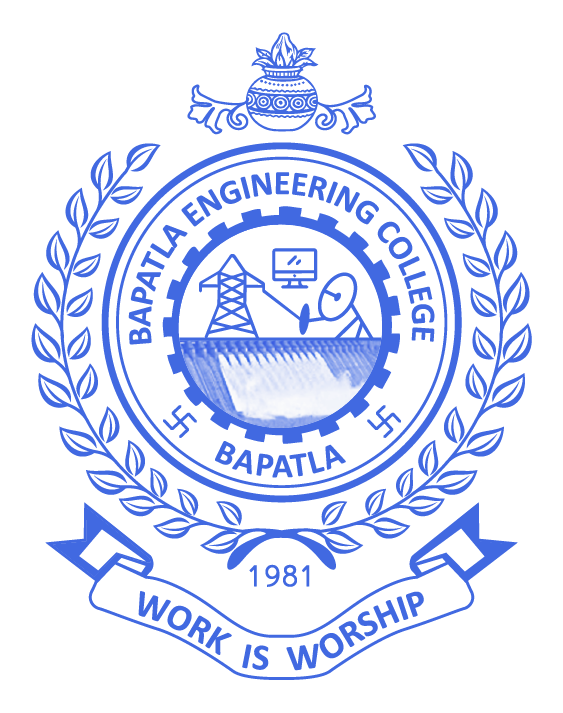
**Award of the Degree of**

**MASTER OF COMPUTER APPLICATIONS**

**Submitted By**

**ARUNURU NAVEEN**

**Reg.No: L20MC23014**

****

**Under The Esteemed Guidance Of**

**Sri N. KIRAN KUMAR M. Tech**

**Assistant Professor**

**DEPARTMENT OF MCA**

**BAPATLA ENGINEERING COLLEGE**

**(Approved by A.I.C.T.E)**

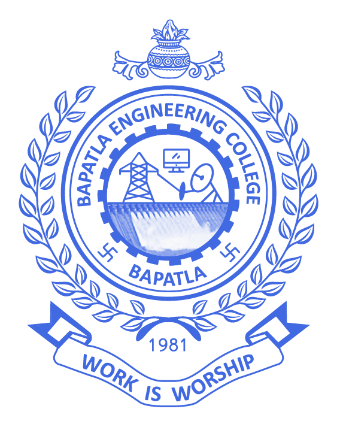
**(Affiliated to ACHARYA NAGARJUNA UNIVERSITY)**

**2019-2021**

**DEPARTMENT OF MCA**

**BAPATLA ENGINEERING COLLEGE**

**BAPATLA-522101**

****

**CERTIFICATE**

This is to certify that this project work entitled “**Secure Cloud Storage based on RLWE Problem”** is the bonafide work carried out by **ARUNURU NAVEEN**, **Reg.No: L20MC23014** submitted in Partial fulfillment of the requirement for the Award of Degree of “**Master of** **Computer Applications**”, during the academic year 2019-2021.

The results submitted in this project have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other university for the award of the any other degree/diploma.

**Project Guide: Head of the Department:**

**Sri N. Kiran Kumar M. Tech Sri K.N. Prasad M. Tech**

**Assistant Professor Associate Professor & Head**

**Dept. of MCA Dept. of MCA**

**Bapatla Engineering College Bapatla Engineering College**

**Bapatla. Bapatla.**

**Signature of External Examiner**

# ACKNOWLEDGEMENT

I sincerely thank the following distinguished personalities who have given their advice and support for successful completion of this work.

I would like to thank **Dr. V. Damodara Naidu**, Principal, Bapatla Engineering College, Bapatla for providing the online resources and allowing me to carry out this work.

I extended my sincere thanks to **Sri K.N. Prasad**, Head of the Department for extending their cooperation and providing the required resources.

I am deeply indebted to most respect and my project guide **Sri N. Kiran Kumar**, Assistant Professor Dept. of MCA for his valuable and inspiring guidance, comments, suggestions and encouragements.

I extended my sincere thanks to all other teaching and non-teaching staff of the department who helped directly or indirectly for their cooperation and encouragement.

**ARUNURU NAVEEN**

**(L20MC23014)**

**DECLARATION**

This is to declare that the project **“Secure Cloud Storage based on RLWE Problem”** at Bapatla Engineering College has been presented by me during the academic year **2019-2021** in partial fulfillment of the requirements for the **“Master of Computer Application”**.

I also declare that this project is the result of my own efforts and that it has not been submitted to any other universities for the award of degree or diploma.

**ARUNURU NAVEEN**

**(L20MC23014)**

Table of Contents

[ACKNOWLEDGEMENT iii](#_Toc80180563)

[ABSTRACT viii](#_Toc80180564)

[CHAPTER-1 1](#_Toc80180565)

[INTRODUCTION 1](#_Toc80180566)

[1.1 INTRODUCTION 1](#_Toc80180567)

[CHAPTER- 2 5](#_Toc80180568)

[LITERATURE SURVEY 5](#_Toc80180569)

[2.1 LITERATURE REVIEW 5](#_Toc80180570)

[CHAPTER-3 7](#_Toc80180571)

[THEORETICAL BACKGROUND 7](#_Toc80180572)

[3.1 INTRODUCTION: 7](#_Toc80180573)

[3.2 INTRODUCTION TO PYTHON 12](#_Toc80180574)

[3.3 BENFITS OF PYTHON 21](#_Toc80180575)

[CHAPTER-4 30](#_Toc80180576)

[SYSTEM ANALYSIS 30](#_Toc80180577)

[4.1 EXISTING SYSTEM: 30](#_Toc80180578)

[4.1.1 DISADVANTAGES OF EXISTING SYSTEM: 30](#_Toc80180579)

[4.2 PROPOSED SYSTEM: 30](#_Toc80180580)

[4.2.1 ADVANTAGES OF PROPOSED SYSTEM: 31](#_Toc80180581)

[CHAPTER- 5 32](#_Toc80180582)

[SYSTEM DESIGN 32](#_Toc80180583)

[5.1 INTRODUCTION 32](#_Toc80180584)

[5.2 MODULES 32](#_Toc80180585)

[5.2.1 DATASET: 32](#_Toc80180586)

[5.2.2 PREPROCESSING: 32](#_Toc80180587)

[5.2.3 GRAPHS: 32](#_Toc80180588)

[5.2.4 PREDICTION: 32](#_Toc80180589)

[5.3 SYSTEM ARCHITECTURE 33](#_Toc80180590)

[5.4 UML DAIGRAMS 34](#_Toc80180591)

[5.4.1 CONSTRUCTION OF USE CASE DIAGRAMS: 37](#_Toc80180592)

[5.4.2 SEQUENCE DIAGRAMS: 40](#_Toc80180593)

[5.4.3 CLASS DIAGRAM: 42](#_Toc80180594)

[5.4.4 ACTIVITY DIAGRAM: 42](#_Toc80180595)

[CHAPTER-6 44](#_Toc80180596)

[SYSTEM REQUIREMENTS 44](#_Toc80180597)

[6.1 SYSTEM REQUIREMENTS 44](#_Toc80180598)

[6.1.1 HARDWARE REQUIREMENTS: 44](#_Toc80180599)

[6.1.2 SOFTWARE REQUIREMENTS: 44](#_Toc80180600)

[CHAPTER-7 45](#_Toc80180601)

[SYSTEM IMPLEMENTATION 45](#_Toc80180602)

[7.1 INPUT AND OUTPUT DESIGNS 45](#_Toc80180603)

[7.1.1 LOGICAL DESIGN 45](#_Toc80180604)

[7.1.2 PHYSICAL DESIGN 45](#_Toc80180605)

[7.2 INPUT & OUTPUT REPRESENTATION 46](#_Toc80180606)

[7.2.1 INPUT DESIGN 46](#_Toc80180607)

[7.2.2 OBJECTIVES 47](#_Toc80180608)

[7.2.3 OUTPUT DESIGN 47](#_Toc80180609)

[CHAPTER-8 69](#_Toc80180610)

[SYSTEM TESTING 69](#_Toc80180611)

[8.1 INTRODUCTION: 69](#_Toc80180612)

[8.2 LEVELS OF TESTING 69](#_Toc80180613)

[8.2.1 BLACK BOX TESTING 70](#_Toc80180614)

[8.2.2 WHITE BOX TESTING 72](#_Toc80180615)

[CHAPTER-9 74](#_Toc80180616)

[OUTPUT SCREENS 74](#_Toc80180617)

[CONCLUSION 75](#_Toc80180618)

[REFERENCES 76](#_Toc80180619)

**List of Figures**

**Name of the figure Pg.no**

[Figure 3. 1 Structure of cloud computing 7](#_Toc80179715)

[Figure 3. 2 Characteristics of cloud computing 9](#_Toc80179716)

[Figure 3. 3 Structure of service models 10](#_Toc80179717)

[Figure 5. 1 System Architecture 33](#_Toc80179735)

[Figure 5. 2 Use Case Diagram 39](#_Toc80179736)

[Figure 5. 3 Sequence diagram 41](#_Toc80179737)

[Figure 5. 4 Class Diagram 42](#_Toc80179738)

[Figure 5. 5 Activity Diagram 43](#_Toc80179739)

# ABSTRACT

This “E-commerce Gift shop Website using Django” project was built mainly using Python and Django Framework. All customers can view wide range of products; however, if they decide to buy one, they must first signup and thereafter login in order to add that product to their shopping cart. Before purchasing a product, customers can look at the products uploaded by. Customer can add as many products to their shopping cart as they desire. The customers can then adjust the quantity of a product added to the cart. Finally, during checkout, Customers can provide their details, and place their order by paying the amount according to their order. The admin can then view the customer's information, including his or her order information and the address to which the order should be delivered and also whether the payment is paid or not.

.

**PROBLEM STATEMENT:**

* With the evolution of internet and so many other computing devices, market places have been brought to the fingertips of customers without leaving their homes, offices, and other places which can deny the customers access to the market at that particular moment. One significant outcome of the internet is the birth of online shopping which is also known as e-commerce (Electronic Commerce). Many organizations nowadays invest so much in e-commerce, some of the major giants are Amazon, Shopify, AliExpress, Jumia, Konga, etc. These major organizations major role is to serve as middlemen between the manufacturers and consumers also known as an online retailer between manufacturers and consumers. The importance of such service is so significant because consumers cannot always be at all the market places at the same time to purchase goods, for that reason, these major players in this business deal with goods from a variety of manufacturers from different industries ranging from edible goods (groceries, desserts, etc.) to non-edible goods (like computers and electronics, wears, utensils etc.) and so on. This process of goods supply chain still remains the same as the normal one and has relieved consumers the stress of going to different market places to purchase goods.

**OBJECTIVE:**

* Differing characteristics of local environments, both infrastructural and socio-economic, have created a significant level of variation in the acceptance and growth of e-commerce in different regions of the world. The sharp increase of internet usage, as well as the systematic progress of Information Technology has transformed the way goods and services are bought and sold, resulting to exponential growth in the number of online shoppers. Consequently, a lot of differences regarding online purchases have been revealed due to various customer characteristics and types of available products and services. Consumers shop gifts online for goal-oriented and experiential reasons, of which goal-oriented motives are more common among online shoppers than are experiential motive.

# CHAPTER-1

# INTRODUCTION

## Introduction

The objective of the project is to make a website that enables users to shop virtually through internet and to purchase the products of their choice from the retailer. Web based shopping is the course of clients purchases merchandise from a seller progressively over the Internet without the need of a mediator supplier. It's a kind of electronic exchange. This project instructs customers of a physical store with the benefits of online shopping. It allows you to buy things in a store from anywhere in the world using a website and the internet. As a result, the customer will be able to shop online and have their purchases delivered to their home from their preferred vendor. The Administrator and the Users are the 2 key groups in this project. As the administrator, the store manager is in charge. Administrators can adding, editing, updating, or deleting the products, allowing administrator to change the product names, price of products, and adding or removing products, among other things. The customer can utilize the shop to look for items, update their trucks, eliminate things from their trucks, and look at. The client can likewise alter his own data, like names, addresses, and other data.

Since the stress of going to different market places by the consumers have been cared for, some consumers still encounter some difficulties when using the services of this major organizations which may be due to language barrier. The paper focuses on removing the barrier thereby converting the official spoken language (English) to either a native language or any other language. The aim of this research is to develop an online shop with a popular python web framework called Django. The objectives are as follows: To explore the functionalities of the technologies driving some popular e-commerce sites; To develop an e-commerce website translated to some indigenous languages; To integrate a payment processor for managing payments for customers on the site; To build a recommendation engine to recommend goods related to the item being purchased at the moment. This paper shall explore the functionalities of a standard e-commerce website and other third-party software used for the development of an e-commerce website which are all based on the popular python web framework (Django).

.

# CHAPTER- 2

# literature survey

## 2.1 literature review

To uncover the concept of online shopping, a writing survey is an expressive review in light of a cautious assessment of past significant examinations connected with the numerous ideas of web based shopping. It discusses the current state of online shopping, its importance and concerns, the elements that influence online purchasing, and a critical examination of online shopping privacy and security issues. Clients can shop online 24 hours per day, as indicated by Ling. Shoppers can buy any labor and products whenever and from any area. When contrasted with in-store shopping, web based shopping is more easy to use since clients can satisfy their requirements with a couple of mouse clicks while never leaving their home. The customer can utilize the shop to look for items, update their trucks, eliminate things from their trucks, and look at. The client can likewise alter his own data, like names, addresses, and other data.

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library [1]. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. Django is a free and open-source web framework, which follows the modelview-template architectural pattern. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. Some well-known sites that use Django include the Public Broadcasting Service, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor. It was used on Pinterest, but later the site moved to a framework built over Flask [2]..

# Chapter-3

# Theoretical background

## 3.1 Introduction:

Having a great site architecture means making products and categories findable on your website, in a way that users and search engines can reach them as efficiently as possible. There are two concepts regarding site architecture: Efficient crawling and indexing. This refers to the Technical Architecture or TA and Classifying, labeling and organizing content. This refers to Information Architecture or IA. Together, information and technical architecture form the site architecture (SA). A good understanding of these two concepts will help to build search engine optimized websites that are search-engine and user-friendly [3]. Information Architecture: is the process of classifying and organizing content on a website while providing userfriendly access to that content, via navigation. This process is done (or should be done) by information architects. Technical architecture: is the process of designing the technical and functional aspects of a site. This is mostly done by web developers [4].

## 3.2 Introduction to PYTHON

**Python Technology**

Python technology is both a programming language and a platform.

**The python Programming Language**

THE PYTHON PROGRAMMING LANGUAGE IS A HIGH-LEVEL LANGUAGE THAT CAN BE CHARACTERIZED BY ALL OF THE FOLLOWING BUZZWORDS:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Python programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Python byte codes —the platform-independent codes interpreted by the interpreter on the Python platform. The interpreter parses and runs each Python byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

**BENEFITS OF PYTHON**

• Presence of Third-Party Modules

• Extensive Support Libraries

• Open Source and Community Development

• Learning Ease and Support Available

• User-friendly Data Structures

• Productivity and Speed

• Highly Extensible and Easily Readable Language.

**Python**

Python is high level language and it is also integrated version of the program. Python is an object-oriented approach and its main aim to help programmers to write the code clearly, logical code for small and large scale of project.

Pytrhon is dynamically typed and garbage collected it also support multiple programming and it is both procedure and object oriented and also functional programming. And structural programming also supported. It has many built in function it also supports filter, map and reduce function. All the machine learning algorithm and the libraries are being supported by the python programming language. Python also support list, dict, sets and other generators. Python code can be run in different platform such as anaconda, PyCharm etc.

The main goal of this programing language is as follows:

• Python is simple, object-oriented programming language.

• The language and implementation should provide support for software engineering principles such as strong type library preset for different machine learning algorithm, and all other algorithm in simple manner.

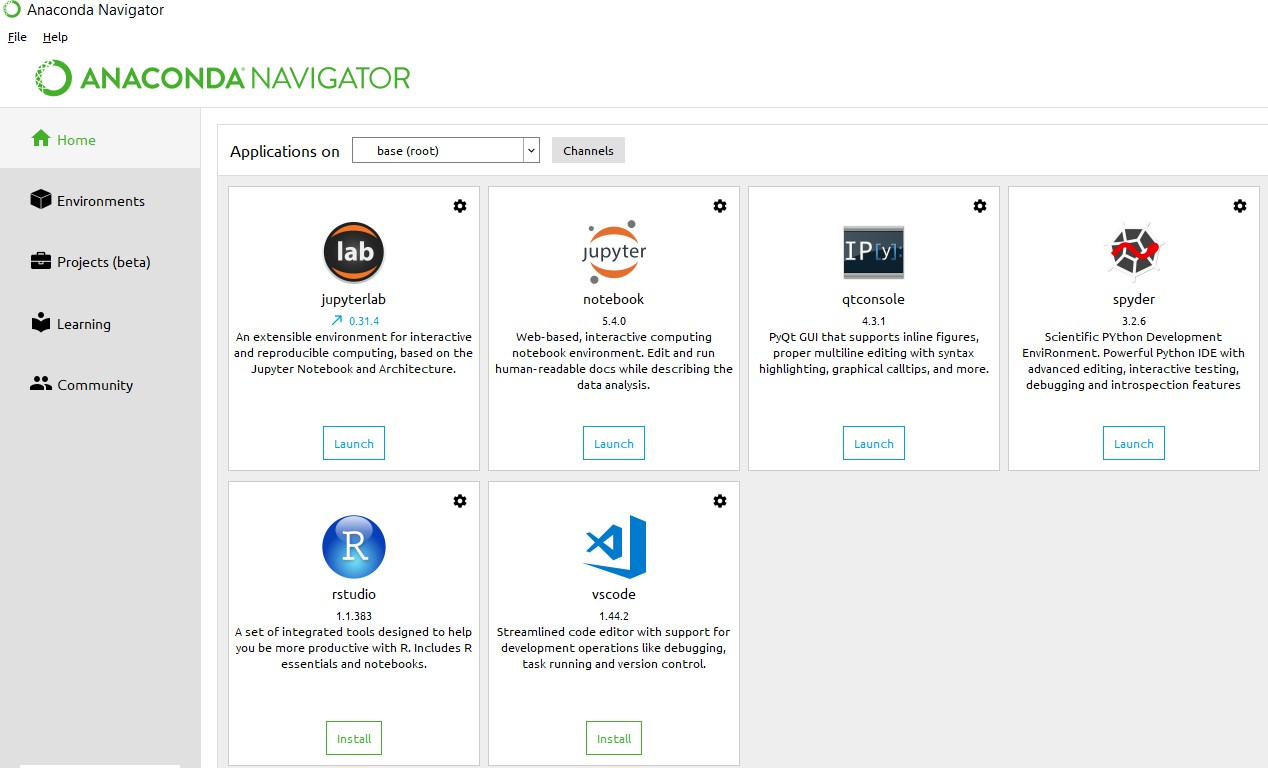
• Coding will be smooth in python and the data analysis can be easily done in python.

This is so much so to the point where we now have modules and APIs at our disposal, and you can engage in machine learning very easily without almost any knowledge at all of how it works. With the defaults from Scikit-learn, you can get 90-95% accuracy on many tasks right out of the gate. Machine learning is a lot like a car, you do not need to know much about how it works in order to get an incredible amount of utility from it.

Despite the apparent age and maturity of machine learning, I would say there's no better time than now to learn it, since you can actually use it. Machines are quite powerful, the one you are working on can probably do most of this series quickly. Data is also very plentiful lately.

**Anaconda**

Anaconda is free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine Learning applications, Large- scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. It is developed and maintained by Anaconda, Inc. The distribution incudes data-science packages suitable for Windows, Linux, and macOS. Packaged versions are required and are managed by the package management system anaconda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depends on, and a small number of other packages.



**Anaconda Console**

**import numpy as np**

* NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
* At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:
  + NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
  + The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
  + NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
  + A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

**import time**

This module provides various time-related functions. For related functionality, see also the datetime and calendar modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).

The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding leap seconds. Leap seconds are excluded from this total on all POSIX-compliant platforms.

The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.

Function strptime() can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.

UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.

DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.

The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g. on most Unix systems, the clock “ticks” only 50 or 100 times a second.

On the other hand, the precision of time() and sleep() is better than their Unix equivalents: times are expressed as floating point numbers, time() returns the most accurate time available (using Unix gettimeofday() where available), and sleep() will accept a time with a nonzero fraction (Unix select() is used to implement this, where available).

The time value as returned by gmtime(), localtime(), and strptime(), and accepted by asctime(), mktime() and strftime(), is a sequence of 9 integers. The return values of gmtime(), localtime(), and strptime() also offer attribute names for individual fields.

See struct\_time for a description of these objects.

Changed in version 3.3: The struct\_time type was extended to provide the tm\_gmtoff and tm\_zone attributes when platform supports corresponding struct tm members.

Changed in version 3.6: The struct\_time attributes tm\_gmtoff and tm\_zone are now available on all platforms.

**import os**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

Notes on the availability of these functions:

The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

On VxWorks, os.popen, os.fork, os.execv and os.spawn\*p\* are not supported.

**Django is a Model View Control (MVC) framework**

• Mange.py -This file is used basically as a command-line utility and for deploying, debugging, or running our web application. It contains code for run-server, or make migrations or migrations, etc. that we use in the shell. Anyway, we do not need to make any changes to the file.

• Db.sqlite3 is the database file.

• Static folder contains all the static files like CSS and images.

• Env folder is the project specific development environment. Its created through a command ‘virtualenev Env’

• Ecomm folder is the Django projects consisting of init, admin, models, views, apps and forms. o Models are basically the blueprints of the database we are using and hence contain the information regarding attributes and the fields etc of the database.

Views is a crucial one, it contains all the Views (usually as classes). Views.py can be considered as a file that interacts with the client. Views are a user interface for what we see when we render a Django Web application.   
o URLs Just like the project urls.py file, this file handles all the URLs of our web application.

Ecommerce folder is the Django application directory consisting of the following essential files

. o Init.py This file remains empty and is present them only to tell that this particular directory is a package.

o Settings.py This file is present for adding all the applications and the middleware application present. Also, it has information about templates and databases. Overall, this is the main file of our Django web application.

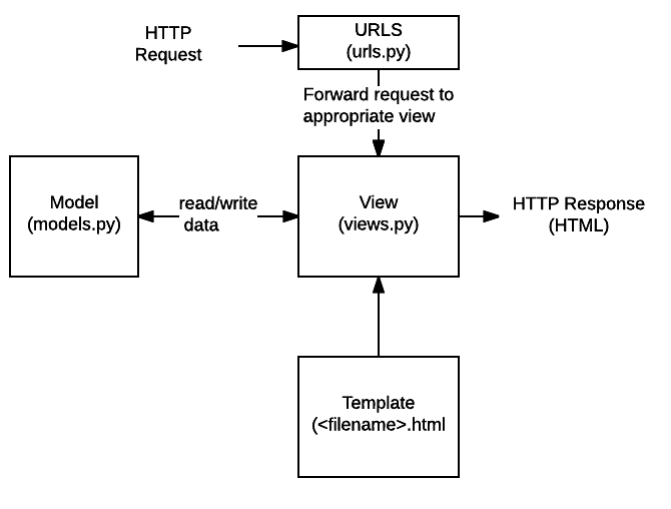
o Urls.py This file handles all the URLs of our web application. This file has the lists of all the endpoints that we will have for our website.

o Wsgi.py This file mainly concerns with the WSGI server and is used for deploying our applications on to servers like Apache etc.

o Asgi.py In the newer versions of Django, you will also find a file named as asgi.py apart from wsgi.py. ASGI can be considered as a succeeded interface to the WSGI. ASGI, short for Asynchronous Server Gateway interface.

# Creating the Form:

The last thing we want to do is permit our client to add a city straight in the structure. For that, we really want to create a structure. We could produce the structure straightforwardly, however since our structure will have identical field as our model, we can utilize our Model Form. Make another document called forms.py. To view this structure, we really want to make it in the view and pass it to our layout. That's what to do, let us update the file video that will make the structure. We will supplant the old city variable a similar time since we never again need it.



# Chapter-4

# System analysis

## 4.1 EXISTING SYSTEM:

The current system for gift shopping is to visit the shop manually and from the available product choose the item customer want and buying the item by payment of the price of the item .

### 4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. It is less user-friendly.

User must go to shop and select products.

3. It is difficult to identify the required product.

4. Description of the product limited.

5. It is a time-consuming process

6. Not in reach of distant users.

## 4.2 PROPOSED SYSTEM:

In the proposed system customer need not go to the shop for buying the products. He can order the product he wish to buy through the application in his webiste. The shop owner will be admin of the system. Shop owner can appoint moderators who will help owner in managing the customers and product orders. The system also recommends a home delivery system for the purchased products.

.

### 4.2.1 ADVANTAGES OF PROPOSED SYSTEM:

* Customers can select various type of gift products at one place
* Time saving for customers
* Track products details online

# CHAPTER- 5

# SYSTEM design

## 5.1 introduction

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

## 5.2 modules

### 5.2.1 . Admin:

### Using this module admin can create super users and login with valid details. Admin can check registered customers delete modify user and update details to database. Admin can add new products to website under different categories , admin can view product details list of orders from users and update status.

### 5.2.2 customer module:

Using this module customer can register with application login with valid username and passwords. User can update profile details select for products from list of available products and add to cart. Customer can check total amount and pay with details and check status of bookings.

**Process**

**A. Creating the Virtual Environment** After python has been installed on the pc, a virtual environment module needs to be installed on the pc through python’s package manager. The virtual environment makes it easy to run different versions of Django in isolation without interrupting the process of one another. The Django module is installed in the virtual environment along with some other modules that are needed to develop the application specifying their versions.

**B. Creating various Apps** When the platform is ready for development, various applications can be created in the project which are relevant to the main idea of the website. For this research, the apps created are as follows: coupons, shop, cart, orders, and payment. Each of these apps comes with some important files when generated by the same Django-admin command in the command line of the operating system which allow programmers build the website easily and some are created by the programmer because it is not created by default by the command.

**C. Creating a Superuser (admin)** The Superuser is simply the admin of the site. The admin account needs to be created from the command line through the Django-admin command. The admin account must be created in order to manage the site with a higher privilege than the users of the site. The admin has the privilege to create, retrieve, update, and delete data content and users from the site through the admin site. All models present in the models.py file must be registered in admin.py file in order to allow the models to be visible for the admin.

**D. Creating models** A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you’re storing. Each attribute of the model represents a database field.

**E. Creating views for the models** When the models have been successfully registered in the admin site, creating views for users is another task needed to be accomplished. This refers to the logical functionality between the request and response of the clients and servers. There are two types of views: 1. Function-based views: A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image or anything. The view itself contains whatever arbitrary logic is necessary to return that response. This code can live anywhere you want, as long as it’s on your Python path. For the sake of putting the code somewhere, the convention is to put views in a file called views.py, placed in your project or application directory. The view function returns an HTML page that includes the request of the user.

**Class-Based views:** Class-based views provide an alternative way to implement views as Python objects instead of functions. These allow you to structure your views and reuse code by harnessing inheritance and mixins. They do not replace function-based views, but have certain differences and advantages when compared to function-based views: i. Organization of code related to specific HTTP methods (GET, POST, etc.) can be addressed by separate methods instead of conditional branching. ii. Object oriented techniques such as mixins (multiple inheritance) can be used to factor code into reusable components.

F. Integrating a Payment Processor A payment gateway allows you to process payments online. Using a payment gateway, you can manage customer's orders and delegate payment processing to a reliable, secure third party. You won't have to worry about processing credit cards in your own system. There are several payment gateway providers to choose from.

## 5.3 system architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system. Organized in a way that supports reasoning about the structures and behaviors of the system.

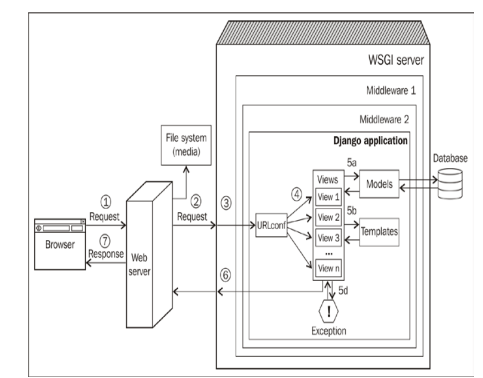
****

Figure 5. 1 System Architecture

Figure 1 shows the simplified journey of a web request from a visitor's browser to your Django application and back. The numbered paths are as follows: 1. The browser sends the request (essentially, a string of bytes) to the web server. 2. The web server (say, Nginx) hands over the request to a WSGI server (say, uWSGI) or directly serves a file (say, a CSS file) from the filesystem. 3. Unlike a web server, WSGI servers can run Python applications. The request populates a Python dictionary called environ and, optionally, passes through several layers of middleware, ultimately reaching your Django application. 4. URLconf contained in the urls.py of your application selects a view to handle the request based on the requested URL. The request has turned into HttpRequest (a Python object). 5. The selected view typically does one or more of the following things: a. Talks to a database via the models b. Renders HTML or any other formatted response using templates c. Returns a plain text response (not shown) d. Raises an exception 6. The Http Response object gets rendered into a string, as it leaves the Django application. 7. A beautifully rendered web page is seen in your user's browser.

**Advantages of Three-Tier:**

* Separates functionality from presentation.
* Clear separation – better understanding.
* Changes limited to well define components.
* Can be running on WWW.
* Effective network performance.

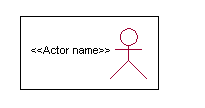
## 5.4 UML DAIGRAMS

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



<<Actor name>>

An actor is someone or something that:

Interacts with or uses the system.

* Provides input to and receives information from the system.
* Is external to the system and has no control over the use cases.

Actors are discovered by examining:

* Who directly uses the system?
* Who is responsible for maintaining the system?
* External hardware used by the system.
* Other systems that need to interact with the system.

Questions to identify actors:

* + Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
  + Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
  + Which external hardware or systems (if any) use the system to perform tasks?
  + What problems does this application solve (that is, for whom)?
  + And, finally, how do users use the system (use case)? What are they doing with the system?

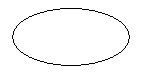
The actors identified in this system are:

1. System Administrator
2. Customer
3. Customer Care

Identification of use cases:

Use case: A use case can be described as a specific way of using the system from a user’s (actor’s) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

* Pattern of behavior the system exhibits
* A sequence of related transactions performed by an actor and the system
* Delivering something of value to the actor

Use cases provide a means to:

* capture system requirements
* communicate with the end users and domain experts
* test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

* For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
* Name the use cases.
* Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

* What are the tasks of each actor?
* Will any actor create, store, change, remove or read information in the system?
* What use case will store, change, remove or read this information?
* Will any actor need to inform the system about sudden external changes?
* Does any actor need to inform about certain occurrences in the system?
* What usecases will support and maintains the system?

**1.2 Flow of Events**

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

* When and how the use case starts and ends
* Use case/actor interactions
* Data needed by the use case
* Normal sequence of events for the use case
* Alternate or exceptional flows

### 5.4.1 Construction of Use case diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

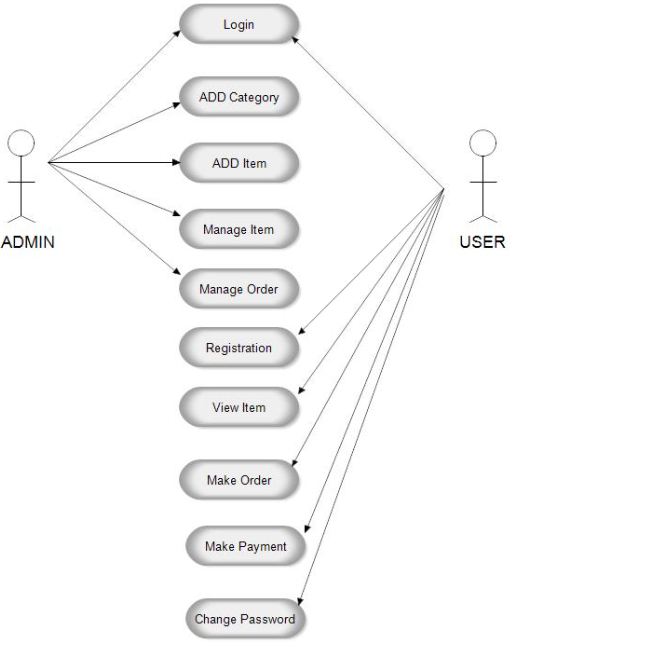


Figure 5. 2 Use Case Diagram

### 5.4.2 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

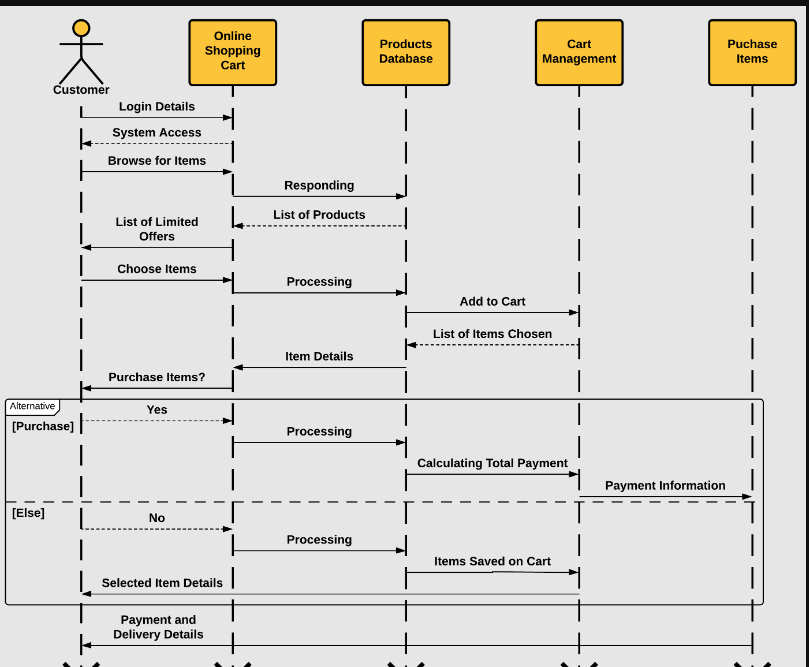


Figure 5. 3 Sequence diagram

### 5.4.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

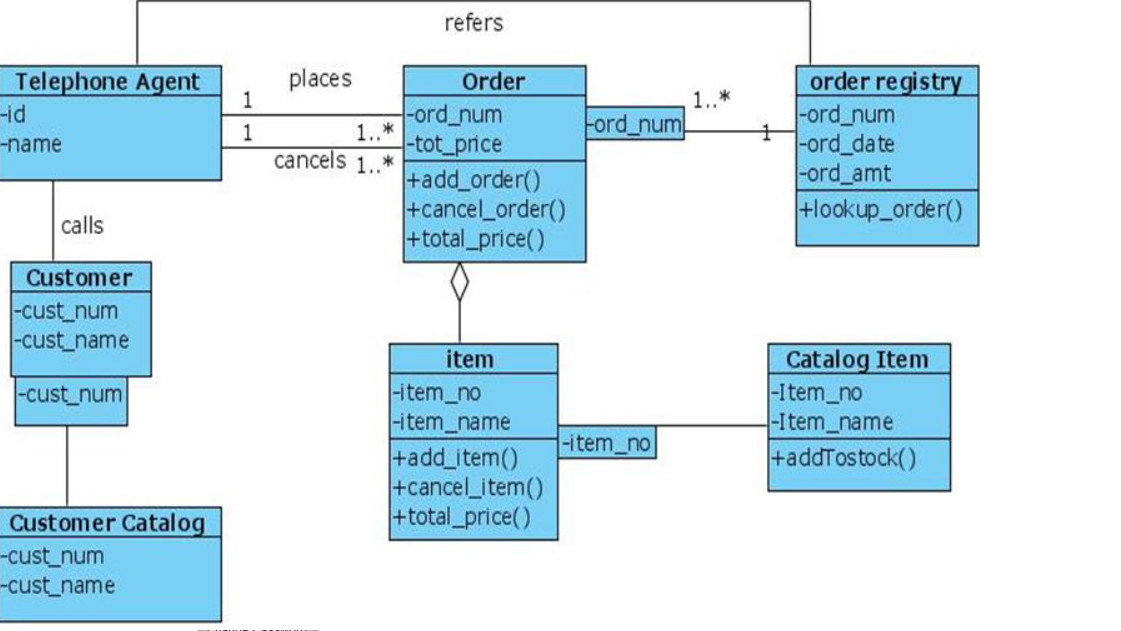


Figure 5. 4 Class Diagram

### 5.4.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

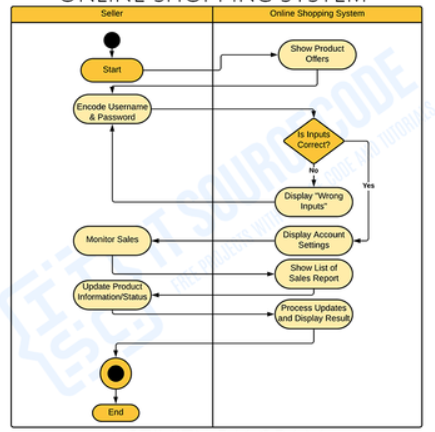
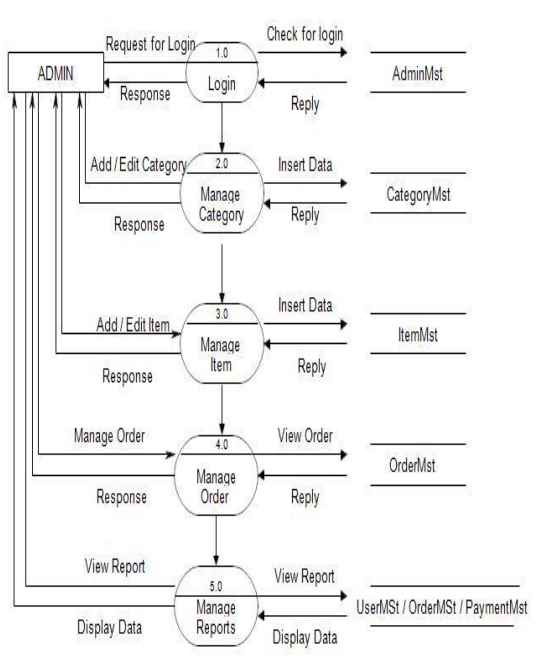
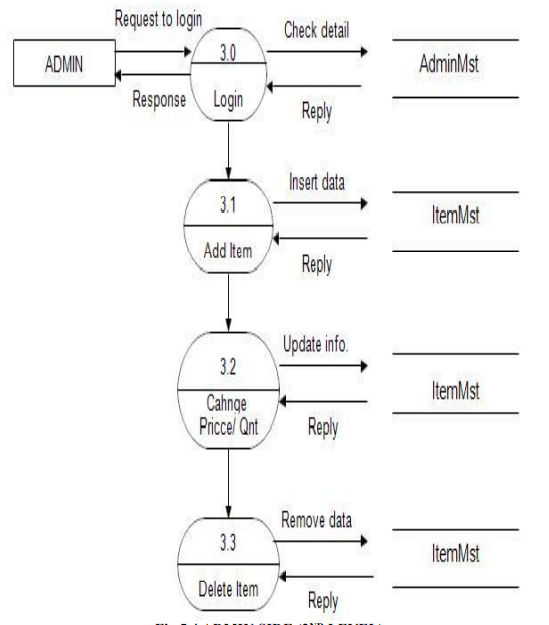
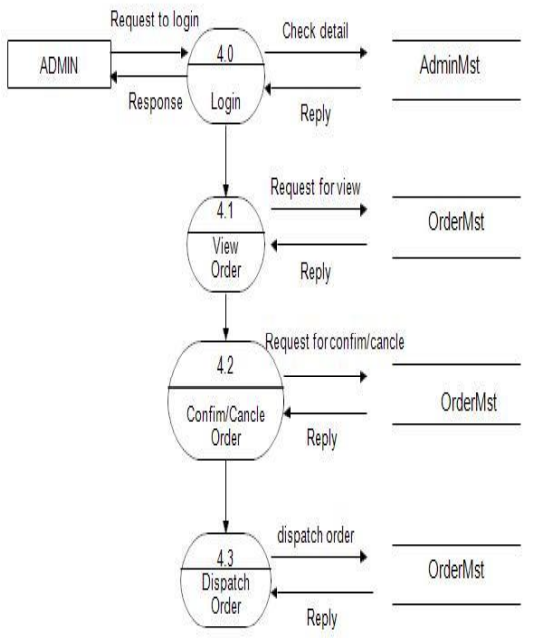


Figure 5. 5 Activity Diagram

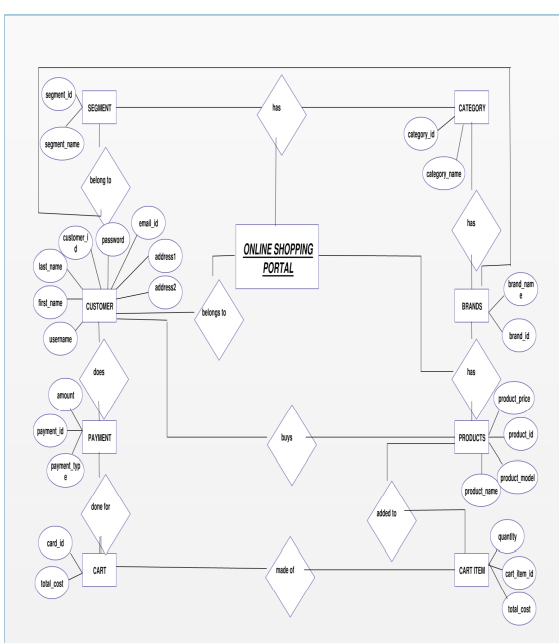
**DFD DIAGRAMS:**

****

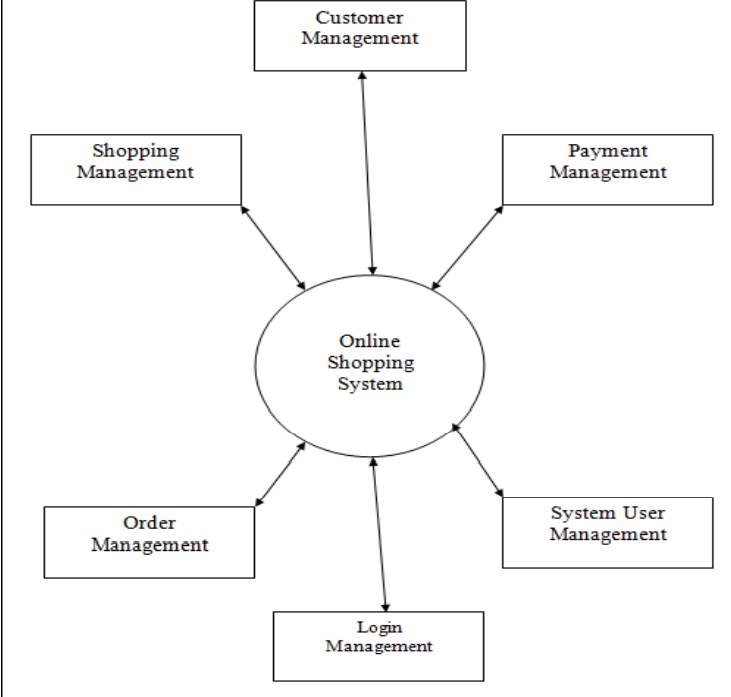
****

****

**ER DIAGRAM:**

****

CONTEXT DIAGRAM:



# 

# CHAPTER-6

# system requirements

## 6.1 SYSTEM REQUIREMENTS

### 6.1.1 HARDWARE REQUIREMENTS:

* System : Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
* Hard Disk : 1 TB.
* Input Devices : Keyboard, Mouse
* Ram : 4 GB.

### 6.1.2 SOFTWARE REQUIREMENTS:

* Operating system : Windows XP/7/10.
* Coding Language : Python
* Tool : Anaconda
* Interface : DJANGO
* Database : db SQL LITE

# Chapter-7

# System implementation

To conduct studies and analyses of an operational and technological nature, and To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

## 7.1 input and output designs

### 7.1.1 Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

### 7.1.2 Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

## 7.2 INPUT & OUTPUT REPRESENTATION

### 7.2.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

### 7.2.2 Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

### Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

* 1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
  2. Select methods for presenting information.
  3. Create document, report, or other formats that contain information produced by the system.

**Code**

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import make\_password

from store.models.customer import Customer

from django.views import View

class Signup (View):

def get(self, request):

return render (request, 'signup.html')

def post(self, request):

postData = request.POST

first\_name = postData.get ('firstname')

last\_name = postData.get ('lastname')

phone = postData.get ('phone')

email = postData.get ('email')

password = postData.get ('password')

# validation

value = {

'first\_name': first\_name,

'last\_name': last\_name,

'phone': phone,

'email': email

}

error\_message = None

customer = Customer (first\_name=first\_name,

last\_name=last\_name,

phone=phone,

email=email,

password=password)

error\_message = self.validateCustomer (customer)

if not error\_message:

print (first\_name, last\_name, phone, email, password)

customer.password = make\_password (customer.password)

customer.register ()

return redirect ('homepage')

else:

data = {

'error': error\_message,

'values': value

}

return render (request, 'signup.html', data)

def validateCustomer(self, customer):

error\_message = None

if (not customer.first\_name):

error\_message = "Please Enter your First Name !!"

elif len (customer.first\_name) < 3:

error\_message = 'First Name must be 3 char long or more'

elif not customer.last\_name:

error\_message = 'Please Enter your Last Name'

elif len (customer.last\_name) < 3:

error\_message = 'Last Name must be 3 char long or more'

elif not customer.phone:

error\_message = 'Enter your Phone Number'

elif len (customer.phone) < 10:

error\_message = 'Phone Number must be 10 char Long'

elif len (customer.password) < 5:

error\_message = 'Password must be 5 char long'

elif len (customer.email) < 5:

error\_message = 'Email must be 5 char long'

elif customer.isExists ():

error\_message = 'Email Address Already Registered..'

# saving

return error\_message

# Chapter-8

# System testing

## 8.1 INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

## 8.2 Levels of Testing

Code testing:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing:

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing:

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

### 8.2.1 BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

* Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
* Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements  such as performance, scalability, usability.
* Regression testing – Regression testing is done  after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

### 8.2.2 WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

WHAT DO YOU VERIFY IN WHITE BOX TESTING?

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

HOW DO YOU PERFORM WHITE BOX TESTING?

  To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

**STEP 1) UNDERSTAND THE SOURCE CODE**

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

**Step 2) CREATE TEST CASES AND EXECUTE**

The second basic step to white box testing involves testing the application’s source code for proper flow and structure. One way is by writing more code to test the application’s source code. The tester will develop little tests for each process or series of processes in the application. This  method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

Unit testing:

|  |  |
| --- | --- |
| Sl # Test Case : ­ | UTC­1 |
| Name of Test: ­ | Customer signup |
| Items being tested: ­ | Validation for signup ornot |
| Sample Input: ­ | Fill form |
| Expected output: ­ | Details stored in database if wrong details are given check validation |
| Actual output: ­ | Validation verified details stored in db |
| **Remarks: ­** | **Pass.** |

|  |  |
| --- | --- |
| Sl # Test Case : ­ | UTC­2 |
| Name of Test: ­ | Create super user |
| Items being tested: ­ | New super user created for Django |
| Sample Input: ­ | Enter data from anaconda prompt |
| Expected output: ­ | New super user created |
| Actual output: ­ | Admin can login with super user |
| Remarks: ­ | pass |

**Integration Testing:**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom­up integration testing and Top­down integration testing.

* + 1. **Bottom­up Integration**

This testing begins with unit testing, followed by tests of progressively higher­level combinations of units called modules or builds.

* + 1. **Top­down Integration**

In this testing, the highest­level modules are tested first and progressively, lower­level modules are tested thereafter.

In a comprehensive software development environment, bottom­up testing is usually done first, followed by top­down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Table 6.5 shows the test cases for integration testing and their results

|  |  |
| --- | --- |
| Sl # Test Case : ­ | ITC­1 |
| Name of Test: ­ | Add product to cart |
| Item being tested: ­ | Are products added to cart |
| Sample Input: ­ | Click on add option on product |
| Expected output: ­ | Details shown in cart |
| Actual output: ­ | Check out with cost |
| Remarks: ­ | Pass. |

|  |  |
| --- | --- |
| Sl # Test Case : ­ | ITC­2 |
| Name of Test: ­ | Payment process |
| Item being tested: ­ | Details of address displayed for address |
| Sample Input: ­ | Click on check out |
| Expected output: ­ | Address form opened and details stored in database |
| Actual output: ­ | Displayed order details to admin |
| Remarks: ­ | Pass. |

**System testing**:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black­box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:

System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.

The application is tested thoroughly to verify that it meets the functional and technical specifications.

The application is tested in an environment that is very close to the production environment where the application will be deployed.

System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

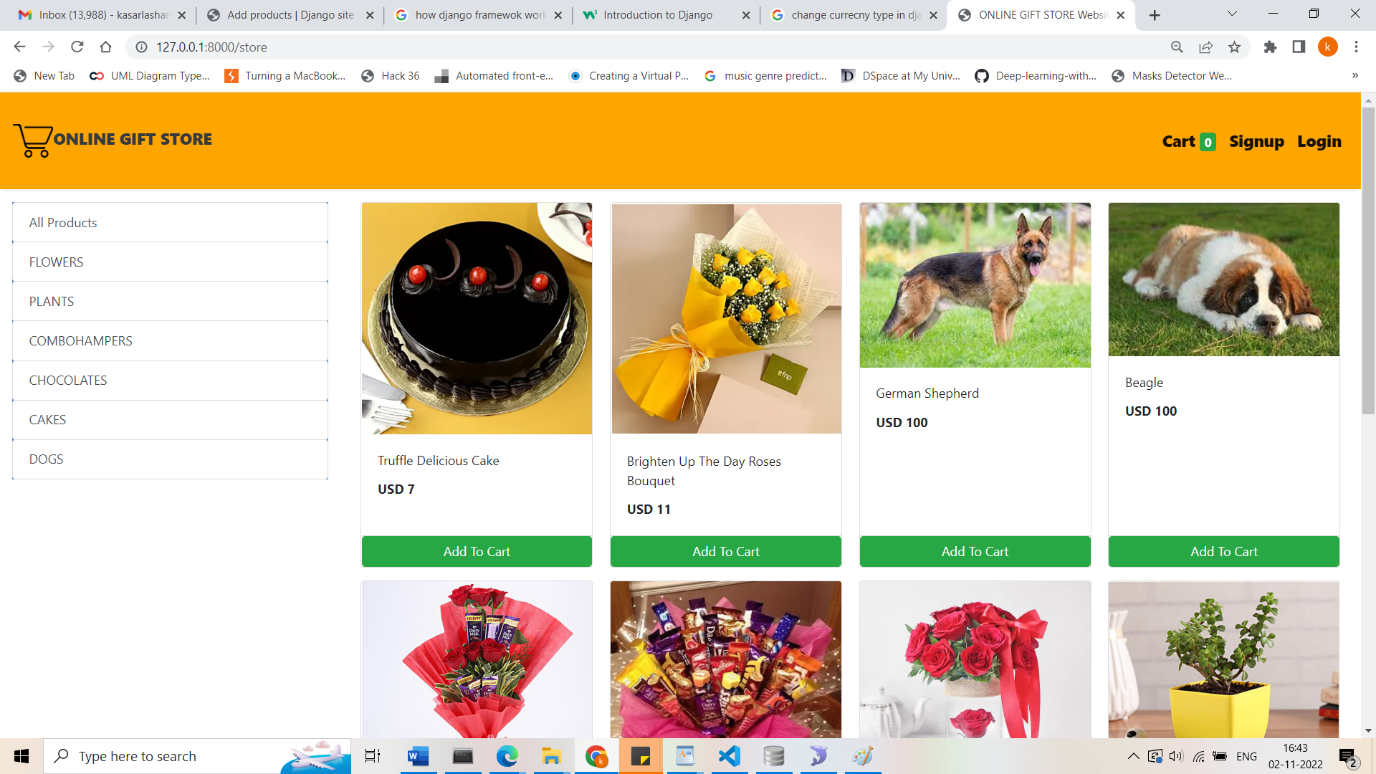
System Testing is shown in below tables

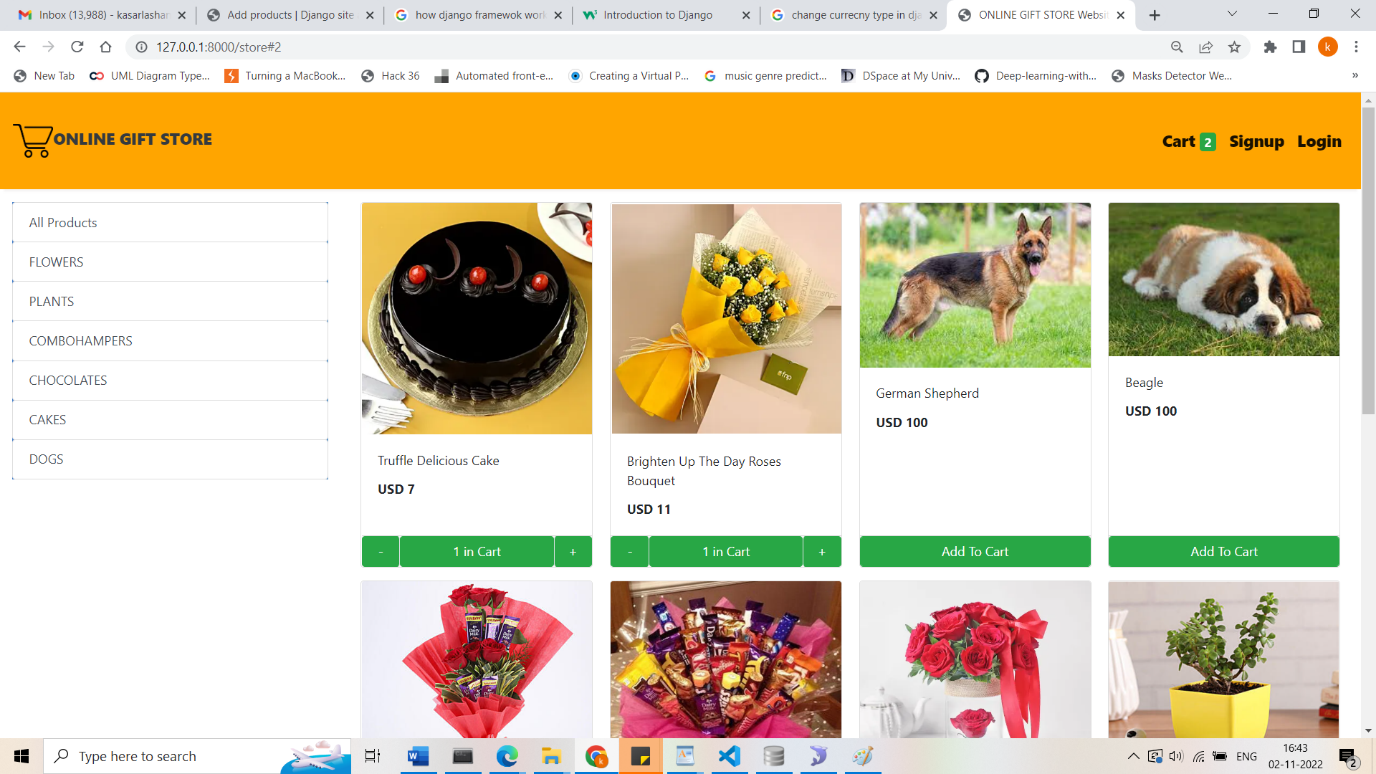
|  |  |
| --- | --- |
| Sl # Test Case : ­ | STC­1 |
| Name of Test: ­ | System testing in various versions of OS |
| Item being tested: ­ | OS compatibility. |
| Sample Input: ­ | Execute the program in windows XP/ Windows­7/8 |
| Expected output: ­ | Performance is better in windows­7 |
| Actual output: ­ | Same as expected output, performance is better in windows­7 |
| Remarks: ­ | Pass |

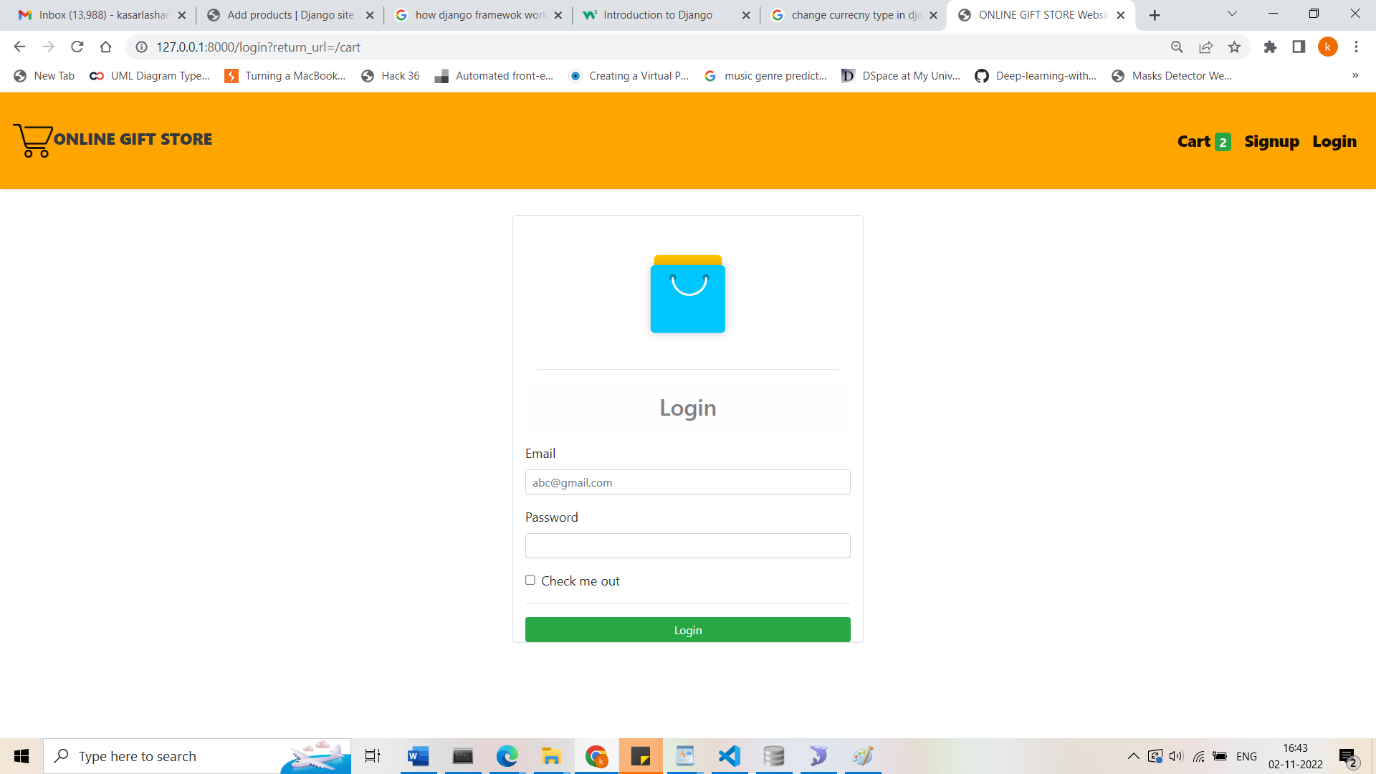
# CHAPTER-9

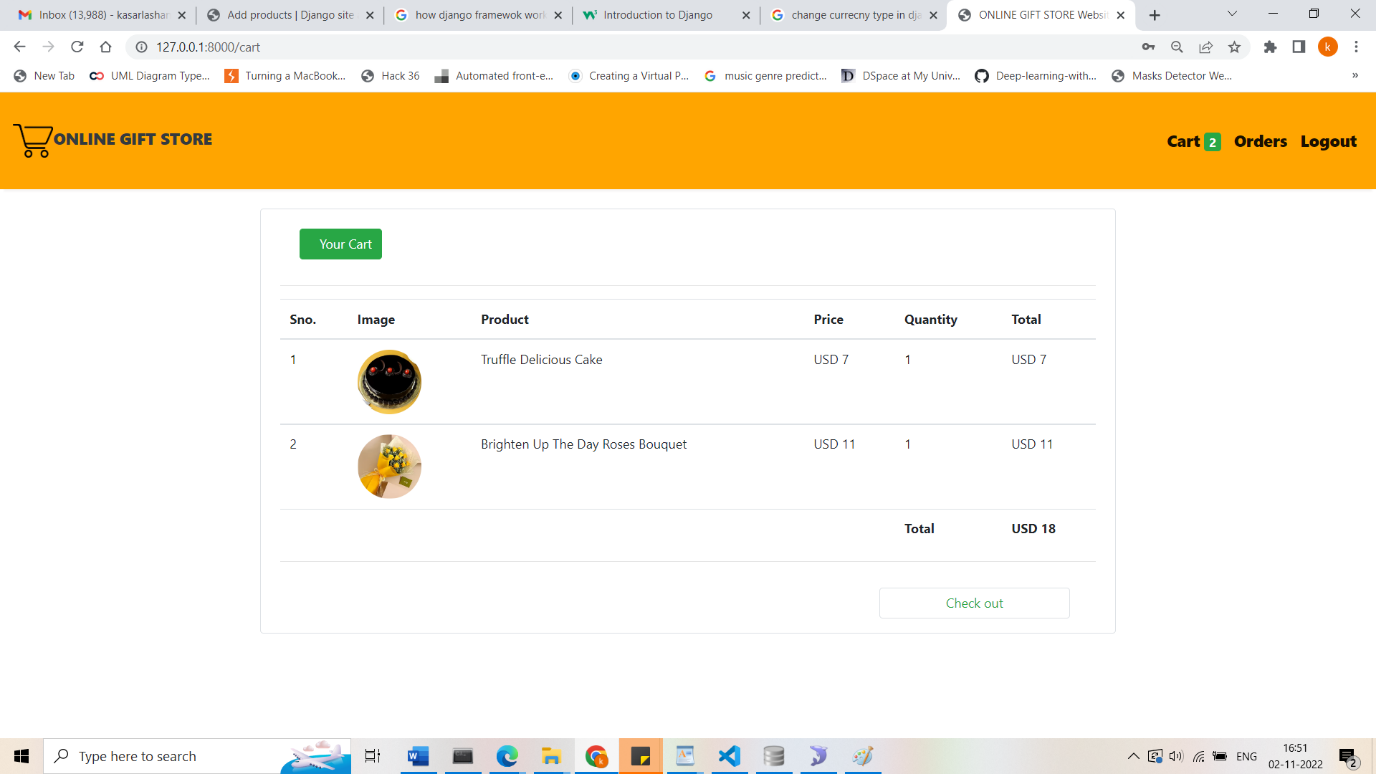
# Output Screens

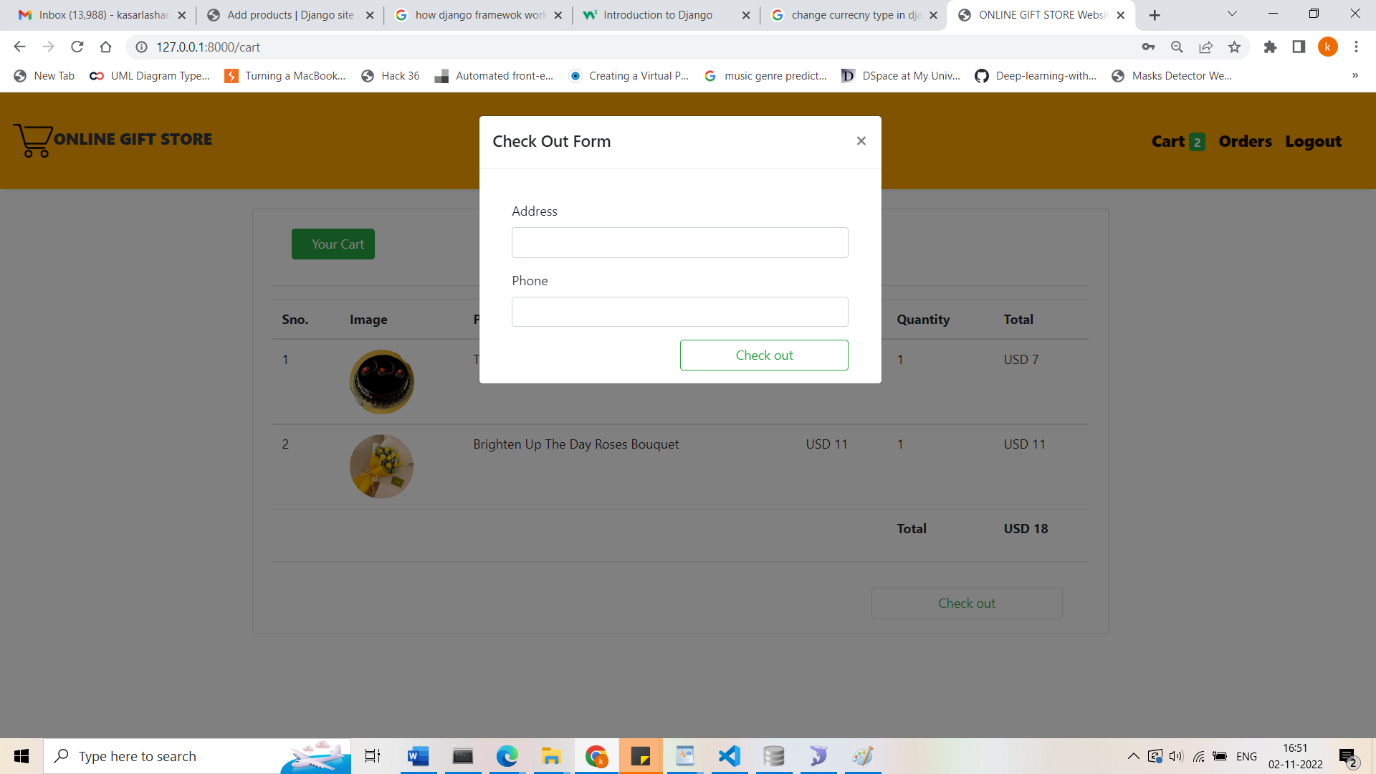
## 9.1 Dataset SCREEN

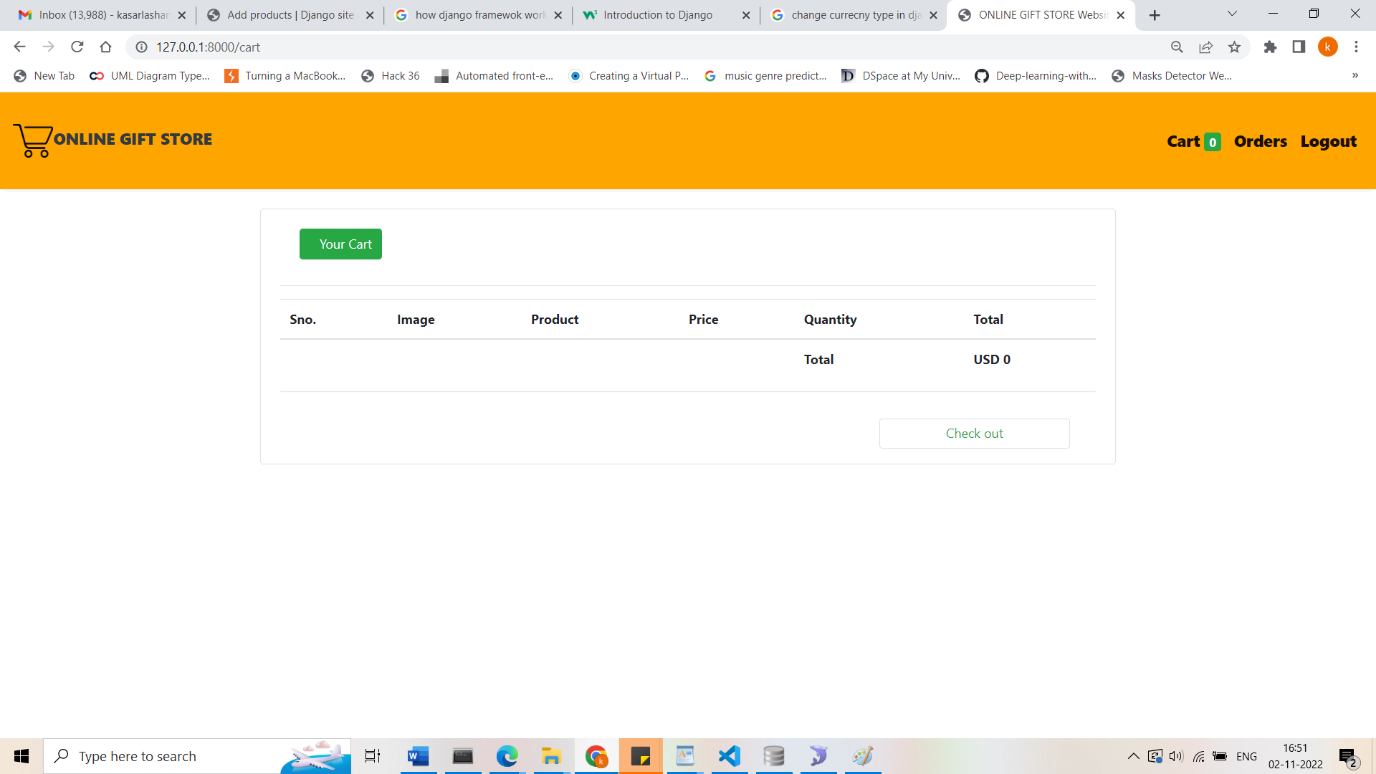


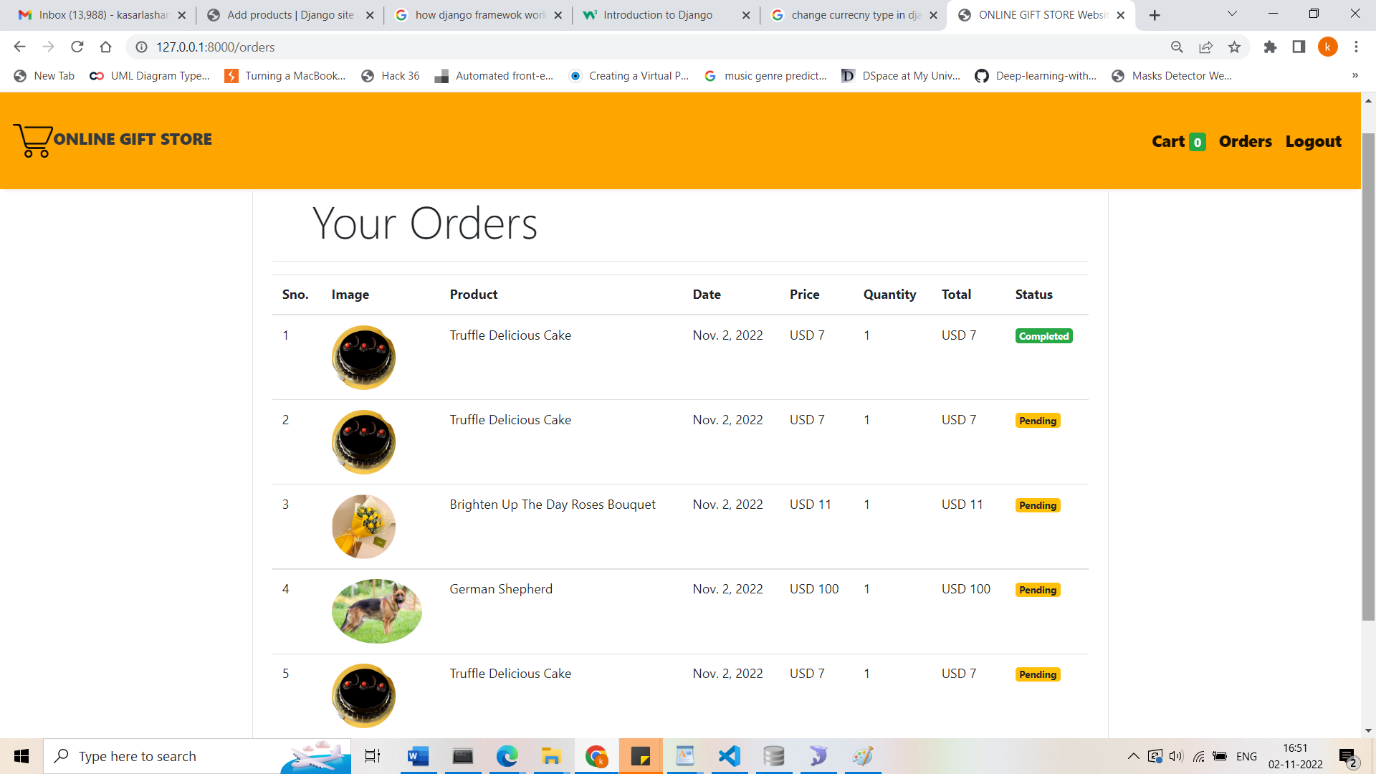


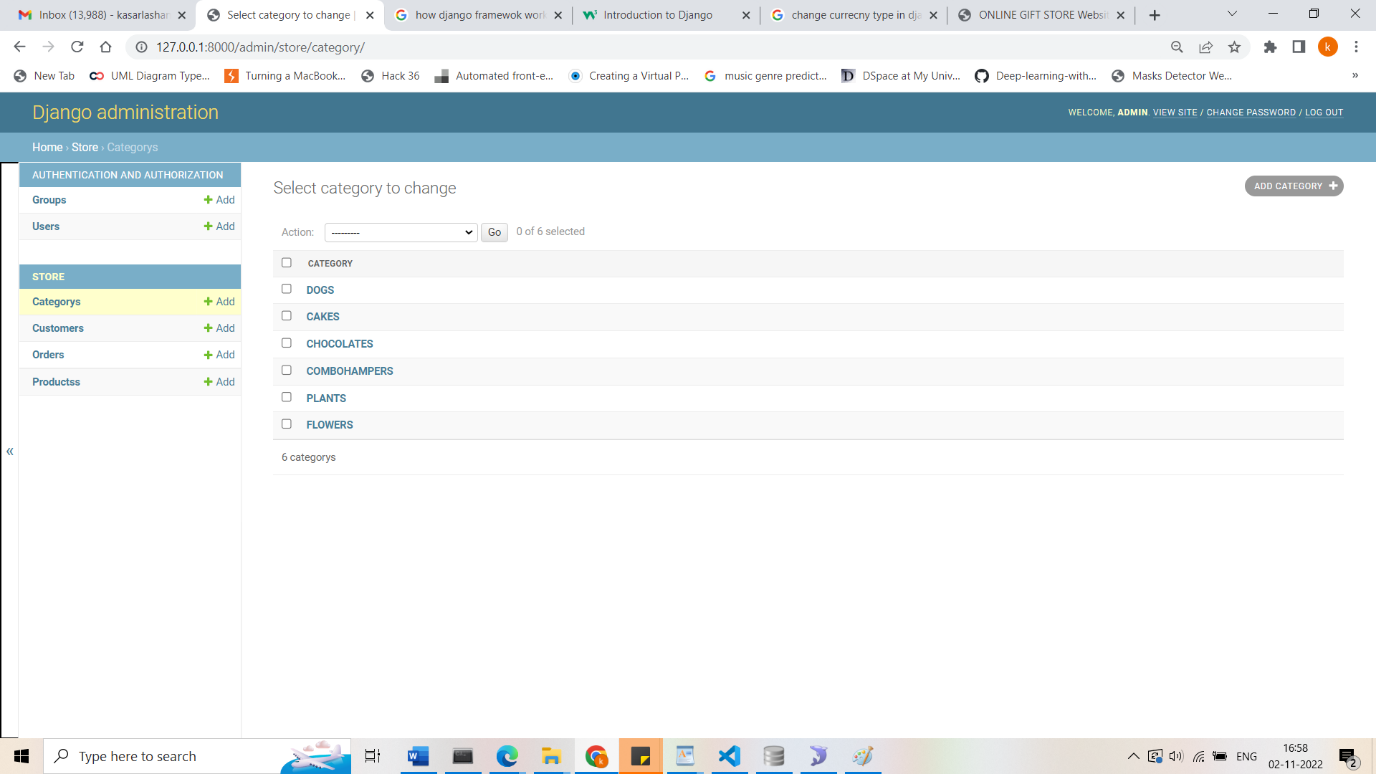


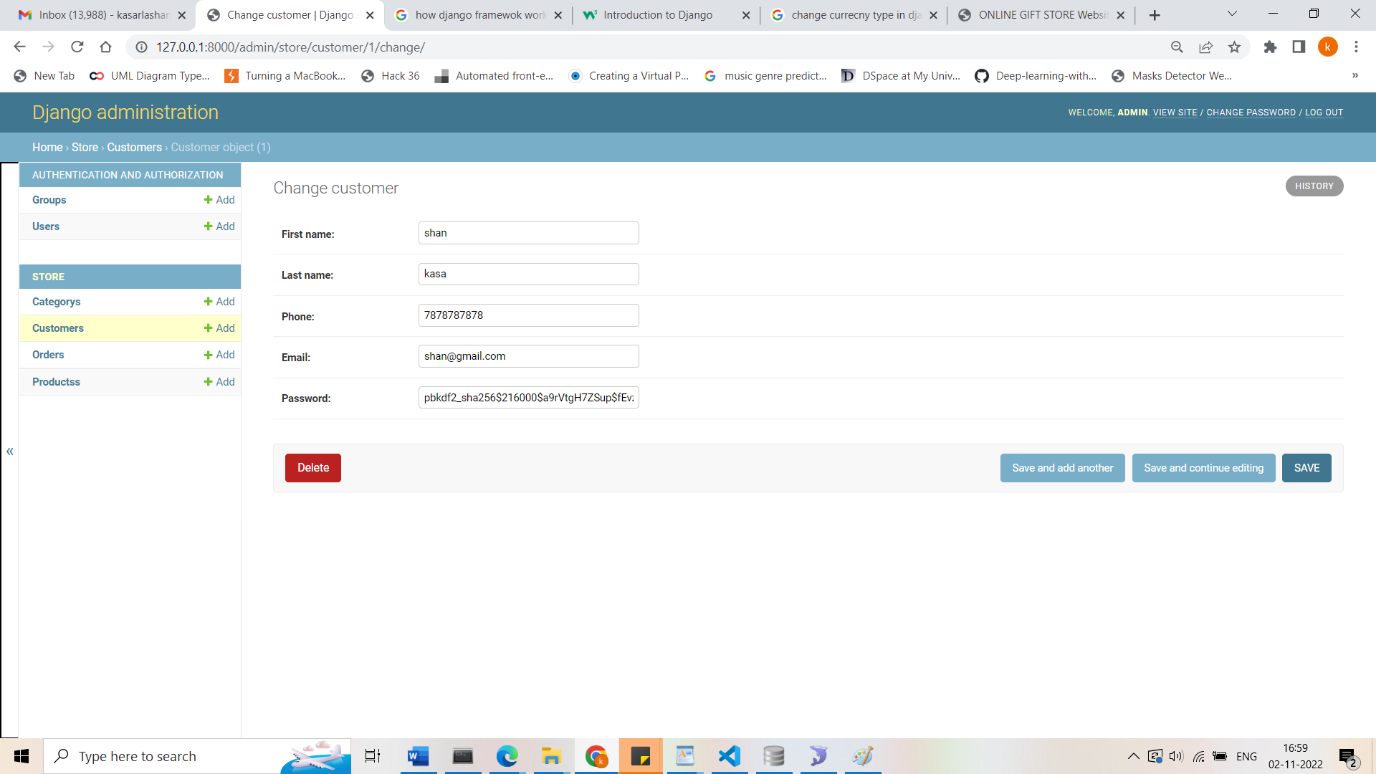


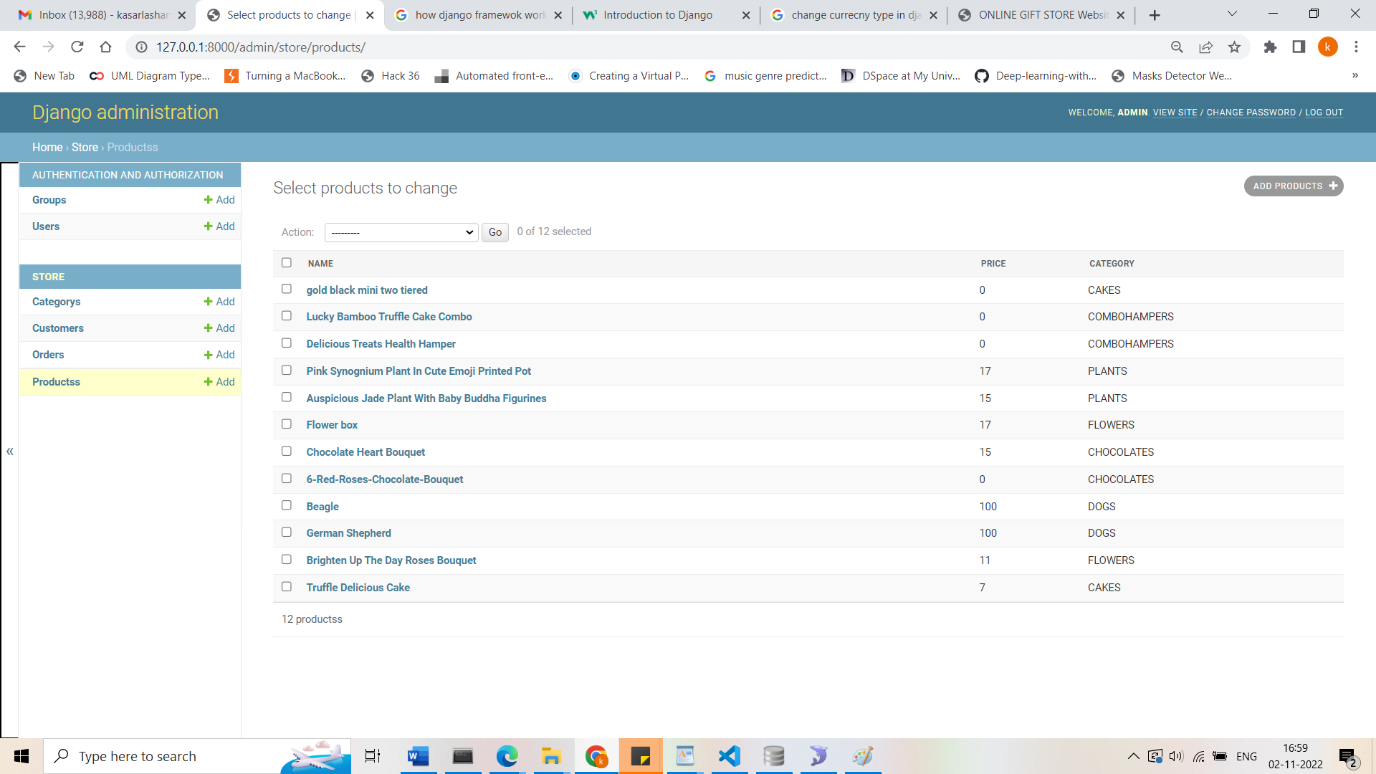
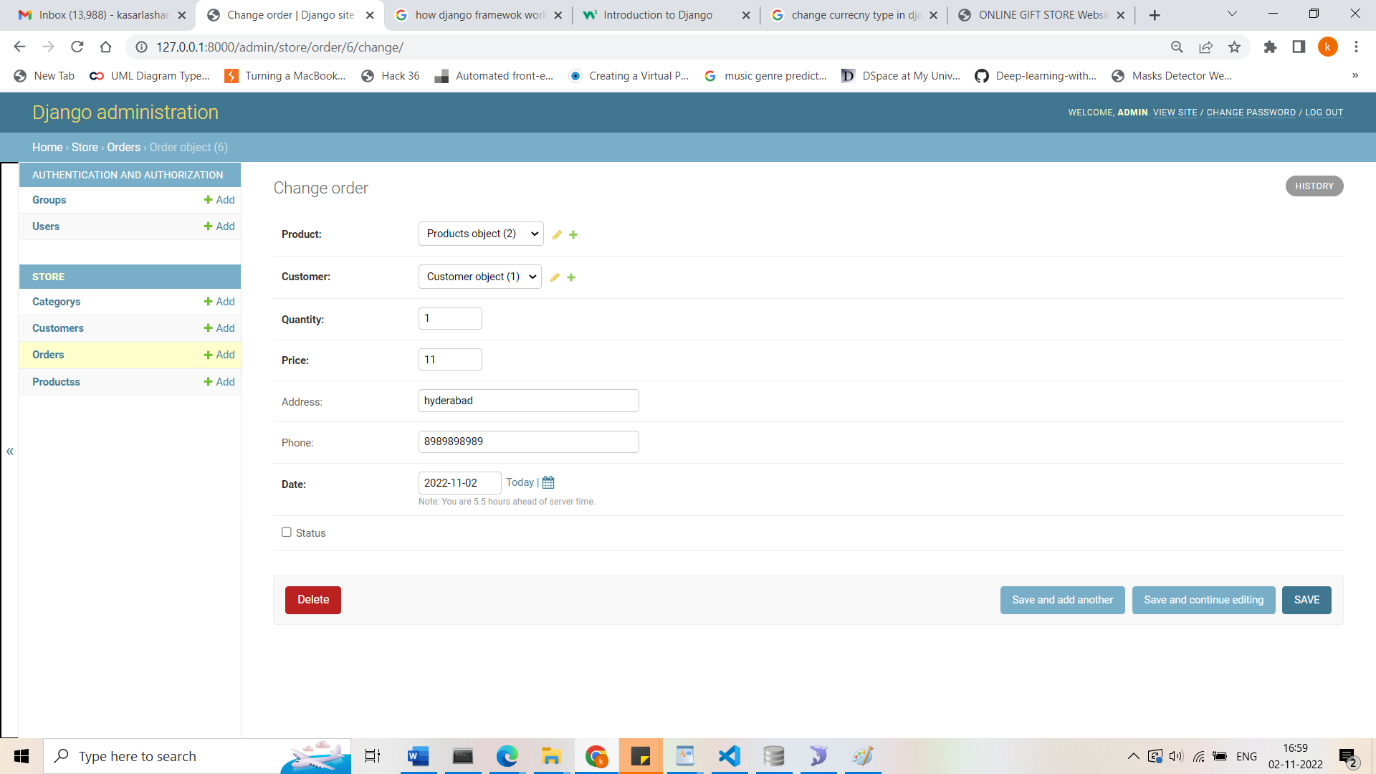


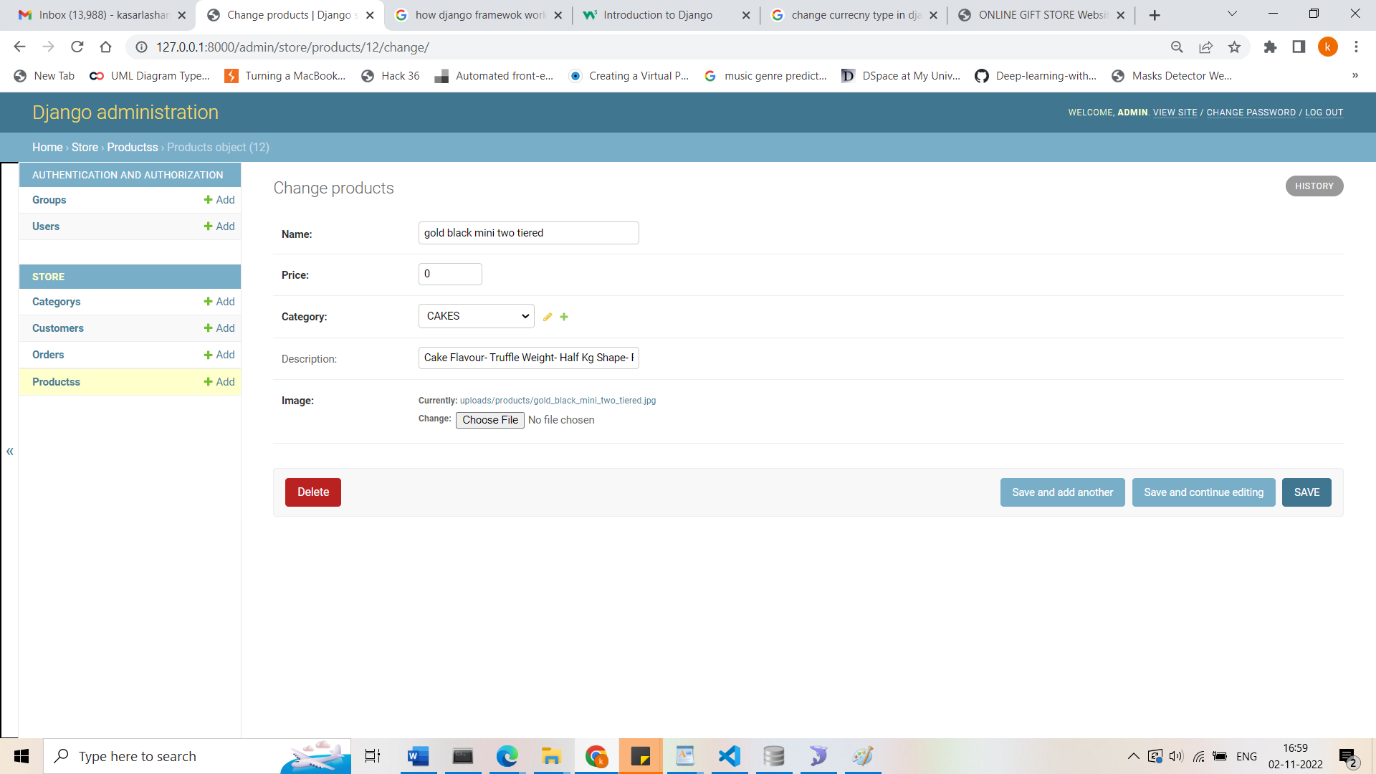












# CONCLUSION

The ‘Online gift store’ is designed to provide a web based application that would make searching, viewing and selection of a product easier. The search engine provides an easy and convenient way to search for products where a user can Search for a product interactively and the search engine would refine the products available based on the user’s input. The user can then view the complete specification of each product. They can also view the product details and also add own products.

# 

# REFERENCES

Python (Programming Language). (2019, May 10). In Wikipedia, The Free Encyclopedia. Retrieved 10:52, May 10, 2019, from <https://en.wikipedia.org/wiki/Python_(programming_language)>.

[2] Django (Python Web Framework). (2019, May 10). In Wikipedia, The Free Encyclopedia. Retrieved 10:52, May 10, 2019, from https://en.wikipedia.org/wiki/Django\_(web\_framework). [3] Website Architecture for Ecommerce. Retrieved 10:00, November 17, 2019, from <https://ecommercetuners.com/ecommerce-site-architecture>.

[4] Website Architecture for Ecommerce. Retrieved 10:00, November 17, 2019, from https://ecommercetuners.com/crawl-optimization-forecommerce.