**ABSTRACT**

The project “super market” deals with the automation of supermarket. This software will help sales person in managing the various types of records pertaining to his/her customer. The product will help the user to work in a highly effective and efficient environment. The sales person has been recording the customer information in the past and even in the present through their personal manual efforts and indeed it consumes their considerable time and energy that could be utilized in the better productive activities.

Apart from that, with increasing customer strength, the task of managing information of each individual customer is indeed a cumbersome task. There is a lot of reason for the introduction of this project. In manual system there are inefficiencies that are sales persons facing.

The information retrieval one of the foremost problems it is very difficult to gather the overall performance reports of the customer.

Predicting the sales depending the previous advertisement data from the supermarket and predicting the best advertisement way to grow sales as per the supermarkets.

Analysis the best way of advertisement and proving real time sales growth by using the polynomial regression.

Marketing Automation

**CONTENTS**

**CHAPTER PAGE NO**

**Chapter – 1 Introduction 3-8**

1.1 Introduction 3-4

1.2 Literature Survey 5-7

1.3 Problem Statement 8

1.4 Objective of Thesis 8

**Chapter - 2 System Setup 9-22**

2.1 System Specifications 9

2.2 System Analysis 9-10

2.3 Software Installation 11-13

2.4 System Design 14-20

2.5 Block Diagram 21

2.6 System Architecture 22

**Chapter - 3 Python 23-36**

3.1 Introduction to Python 23-31

3.2 Python Modules 32-34

3.3 Python Web Frameworks 35-36

**Chapter – 4 Project code and Screenshots 37-65**

4.1 Project Code 37-60

4.2 Screenshots 61-62

4.3 System Study 63

4.4 System Tesing 64-65

SKUCET, Dept of CSE Page 1

Marketing Automation

**Chapter – 5 66**

5.1 Conclusion 66

**References 67-68**

SKUCET, Dept of CSE Page 2

Marketing Automation

**1.INTRODUCTION**

**1.1 INTRODUCTION**

With the rapid development of modern science and technology, computer technology has penetrated into all fields and becomes the necessary tools for various industries, especially the Internet technology promotion and the establishment of the information highway. It makes the IT industry increasingly shows its unique advantages in the market competition.

Into the digital age, there is a huge data information waiting for processing and transmission, which makes the further development and use of the database is particularly urgent. As some small and medium-sized supermarkets in the domestic market, they are falling behind the large and medium-sized supermarkets during the informatization, but for these enterprises' resource management, information storage and processing also shows the urgent need.

To adapt to market competition, it requires efficient handling and management methods, so it is indispensable that accelerate the process of the computerization of supermarket. Small and medium-sized supermarkets has an important position in Chinese economic development. At present, our country has a lot of small and medium-sized supermarkets, regions are widely distributed, and the information level of small and medium-sized supermarkets is still very low.

With the development of the technology, the computer operation and management is increasingly simplified, computer knowledge is increasingly popularization. At the same time, the fast changing of market economy and intense competition, it is must become inevitable trend that the supermarket industry uses computer to manage inventory, sales, and many other link. As a small and medium-sized supermarkets, their main business is selling products.

But at present a lot of questions exist in the system operation, all sales orders are artificial fill, time-consuming and prone to errors; inventory is artificial bookkeeping and unable to keep track of the most accurate inventory situation; For sales data, it often spend a lot of time and energy to calculate the sales performance of each business membership and each good sales.

In order to solve the above problems and improve the economic benefit, we will prepare for computerized management for the supermarket. A supermarket is a large form of the traditional grocery store, it is a self-service shop offering a wide variety of food and household products, organized into aisles.

SKUCET, Dept of CSE Page 3

Marketing Automation

It is larger in size and has a wider selection than a traditional grocery store, but is smaller and more limited in the range of merchandise than a hypermarket or big-box market. This project is based on the sales transaction and billing of items in a supermarket. The first activity is based on adding the items to the system along with the rate which are present in the supermarket and the name of the items which the supermarket will agree to sell.

This authority is given only to admin (administrator). Any modifications to be done in the item name and the rate can be done only by admin. He also has the right to delete any item. As the customer buys the products and comes to the billing counter, the user is supposed to enter the item name he purchased and the quantity of the item he had purchased. This is not a huge a task.

This study is to produce software which manages the sales activity done in a supermarket, maintaining the stock details, maintaining the records of the sales done for a particular month/year. The users will consume less time in calculation and the sales activity will be completed within a fraction of seconds whereas manual system will make the user to write it down which is a long procedure and so paper work will be reduced and the user can spend more time on the monitoring the supermarket. The project will be user friendly and easy to use.

The system will display all the items whose name starts with the letter selected by the user. He can select out of those displayed. Finally a separate bill will be generated for each customer. This will be saved in the database. Any periodic records can be viewed at any time. If the stock is not available, the supermarket orders and buys from a prescribed vendor. The amount will be paid by deducting the total amount acquired in the sales activity. Admin provides a unique username and password for each employee through which he can login.

SKUCET, Dept of CSE Page 4

Marketing Automation

**1.2 LITERATURE SURVEY**

**[1] Yi Q, Li P. Design and Implementation on Supermarket Shopping Guide System Based on RFID and Internet of Things [J]. Journal of Computer Research & Development, 2010.**

Unstaffed retail shop has been emerging in the past years and significantly affected conventional shopping styles. In this area, unmanned retail container plays an important role, it can greatly influence the user shopping experience, the traditional way based on weighing sensors cannot sense what the customer is taking. This paper proposes a smart unstaffed retail shop scheme based on artificial intelligence (AI) and the internet of things (IoT), aiming at exploring the feasibility of implementing the unstaffed retail shopping style. Based on the data set of 11, 000 images in different scenarios that containing 10 different types of stock keeping unit (SKU), an end-to-end classification model trained by the MASK-RCNN method is developed for SKU counting and recognition, and the proposed solution in this study is able to achieve 97.7% counting accuracy and 98.7% recognition accuracy on the test dataset, which indicates that the system can make up for the deficiency of traditional unmanned container.

**Summary:**

A smart unstaffed retail shop scheme based on AI and the IoT, aiming at exploring the feasibility of implementing the unstaffed retail shopping style. Based on the data set of 11, 000 images in different scenarios that containing 10 different types of stock keeping unit (SKU), an end-to-end classification model trained by the MASK-RCNN method is developed for SKU counting and recognition. The method we proposed eliminates the process of character segmentation and therefore avoids the error brought by character segmentation.

**[2] Chen R, Peng L, Qin Y. Supermarket shopping guide system based on Internet of things[C]// Wireless Sensor Network, 2010. IET-WSN. IET International Conference on. IET, 2010:17-20**

When shopping in a big supermarket, customers often encounter two inconvenient places. The first problem is that it is difficult to find the target goods. The second problem is that the checkout process takes a long time. In this paper, Radio Frequency Identification and two-dimensional code technology are used to solve these two problems. The intelligent supermarket system mainly contains the following four parts: (1) Using RFID technology, the system realizes rapid warehousing and outbound operation to reduce the checkout time to seconds. Using collision detection technology, the system can prevent multi label repeated extraction. (2) The WIFI, two-dimensional code, mobile phone and server are combined together to realize the positioning and navigation function of the products. (3) SCM and RFID can realize antitheft function. (4) The two-dimensional code and mobile phone are combined together, to achieve the supply channel query function.

SKUCET, Dept of CSE Page 5

Marketing Automation

**Summary:**

The intelligent supermarket shopping system based on RFID secures fast shopping, rapid warehousing, theft prevention, fast path finding and search of commodity information source. The quick path finding function reduces the time that takes the customers to find their desired commodities. In addition, the 2D code scanning function provided by the system allows the customers to confirm the source channel of the commodity, and increase their loyalty. For the management of the supermarket, the system offers a number of automation services in terms of warehousing and management. This can greatly reduce the number of staff members and cut down the expenditure.

**[3]** **Zhang Y, Han L, Zhang Y. The Design of Supermarket Electronic Shopping Guide System Based on ZigBee Communication [M]// Advances in Computer Science, Environment, Ecoinformatics, and Education. Springer Berlin Heidelberg, 2011:219-224.**

With the development of economy and society, the emergence of large supermarkets provides people with a convenient place to buy necessities. To some extent, it facilitates the purchase and save time. However, the enormous size of the supermarket with the increase quantities and types of goods makes customer inconvenient to find out what they need and get the latest goods information. This paper presents a mobile electronic supermarket shopping guide system, which can be fitted in the shopping cart locating expected goods and offering the latest information on supermarket product. Based on the ZigBee networking technology and protocol analysis, The paper designs a ZigBee Network Model with the features of communication and location which mainly rests on CC2431 ZigBee network location property and further propose a mobile electronic supermarket shopping guide system which can locate expected goods , navigation and offer the latest information on supermarket product.

**Summary:**

The achievement of this wireless location system mainly depends on CC2431 ZigBee network environment and the built-in wireless location engine [3]. There are three types of nodes—Central node, blind nodes and reference nodes. Central node is constituted of the coordinator (FFD), initiating a network. Blind node made up of terminal device (RFD) is a key to locate, which can calculate the current coordinate. The coordinates of Reference node comprising the router are known in their respective networks, (FFD) which can help the blind node position.

SKUCET, Dept of CSE Page 6

Marketing Automation

**[4** **]Yang C, Liu D, Yao D, et al. Design of monitor-and-control system for Supermarket fresh area based on ZigBee[C]// International Conference on Electric Information and Control Engineering. IEEE, 2011:1302-1305**

Smart gadgets are being embedded almost in every aspect of our lives. From smart cities to smart watches, modern industries are increasingly supporting the Internet-ofThings (IoT). SysMART aims at making supermarkets smart, productive, and with a touch of modern lifestyle. While similar implementations to improve the shopping experience exists, they tend mainly to replace the shopping activity at the store with online shopping. Although online shopping reduces time and effort, it deprives customers from enjoying the experience. SysMART relies on cutting-edge devices and technology to simplify and reduce the time required during grocery shopping inside the supermarket. In addition, the system monitors and maintains perishable products in good condition suitable for human consumption. SysMART is built using state-of-the-art technologies that support rapid prototyping and precision data acquisition. The selected development environment is LabVIEW with its world-class interfacing libraries. The paper comprises a detailed system description, development strategy, interface design, software engineering, and a thorough analysis and evaluation

**Summary:**

SysMART is a modern IoT system that offers fast and safe shopping experiences. SysMART supports a bouquet of features that include indoor navigation, fast checkouts, and food tracking. SysMART interacts with the customers’ smartphone to provide real-time information. The cost associated with offering a premium service to customers is expected to have a high return on investment–with more customers visiting the supermarket for efficient grocery shopping and checkout. Future works include motorizing and tracking the cart to allow smoother shopping for elderly and kids, and facilitate transportation of heavy items. Moreover, future work includes accelerating security aspects and database queries using high performance computing.

SKUCET, Dept of CSE Page 7

Marketing Automation

**1.3 PROBLEM STAMENT:**

Our goal is to develop a web page which shop manager can aces the easily without seeing entire shop and consumer can shop his groceries from various remote locations. Consumer can find his groceries without searching entire shop. low maintenance cost and maintained with less people, only few skill people are needed for maintain. Entire market were managed by the admin and regional manager was admin for such super market. As the customer buys the products and comes to the billing counter, the user is supposed to enter the item name he purchased and the quantity of the item he had purchased. This is not a huge a task. This study is to produce software which manages the sales activity done in a supermarket, maintaining the stock details, maintaining the records of the sales done for a particular month/year. The users will consume less time in calculation and the sales activity will be completed within a fraction of seconds whereas manual system will make the user to write it down which is a long procedure and so paper work will be reduced and the user can spend more time on the monitoring the supermarket. Predicting the growth of sales has per the historical dataset which is collected from the supermarket as per the expenses spending on advertisement and predicting the sales as per amount spent on the advertisement.

**1.4 OBJECTIVE OF THE THESIES:**

Only few skill people are needed for maintain. Entire market were managed by the admin and regional manager was admin for such super market. As the customer buys the products and comes to the billing counter, the user is supposed to enter the item name he purchased and the quantity of the item he had purchased. This is not a huge a task. Predicting the growth of sales has per the historical dataset which is collected from the supermarket as per the expenses spending on advertisement this will avoid huge expenses on advertisement.

SKUCET, Dept of CSE Page 8

Marketing Automation

**CHAPTER 2**

**SYSTEM SETUP**

**2.1 SYSTEM SPECIFICATION**

**2.1.1 Software Requirements**

* Operating System: Windows 7+
* Serverside Script: HTML, CSS & JS
* IDE: PyCharm
* Libraries Used: Pandas, Numpy, OS, sklearn.
* Framework: DJango.

**2.1.2 Hardware Requirements**

# Processor - I3/Intel Processor

# RAM- 4GB (min)

* Hard Disk- 128 GB
* Key Board- Standard Windows Keyboard
* Mouse- Two or Three Button Mouse

**2.2 SYSTEM ANALYSIS**

**2.2.1 Modules:**

**Admin:**

**Login:** Admin has to log in by using the credential

**Add Regional Managers:** Admin can add regional managers according to the location.

**Delete Regional Managers:** Admin can remove regional managers

**Logout:** Admin can logout from his account.

**Regional Manager:**

**Login:** Regional Manager has to login to their account by using the credentials

**Add Product:** Regional Manager can add the product with the photo and description

**View Products:** Regional Manager can view all the products **Update Product:** Regional manager can update any product regarding stock, price, and category like many other

**Delete Product:** Regional Manager can delete the product

**View Orders:** Regional Manager can view the orders placed by the user

SKUCET, Dept of CSE Page 9

Marketing Automation

**Acceptance of Order:** Regional Manager has the ability to accept or reject the order

**Bill generation:** Regional Manager can generate the bill according to the user’s order

**Logout:** Finally logout

**User:**

**Register:** user has to register with his/her personal details

**Login:** User login with the help on registered username and password

**View Products:** user views all the products

**Place order:** user can place the order

**Bill payment:** The user has to pay the amount against the bill generated after the successful placement of order

**Logout:** finally, logout

**Predictions:**

Predicting the growth of sales has per the historical dataset which is collected from the supermarket as per the expenses spending on advertisement and predicting the sales as per amount spent on the advertisement.

**Collecting data:** collecting data from the Kaggle.

**Importing dataset:** importing the dataset by using pandas.

**Cleaning and organizing data:** data cleaning and organizingcan be done by using the pandas as per the requirements.

**Algorithm:** By using the logistic regression predicting the future sales based on types of advertisement.

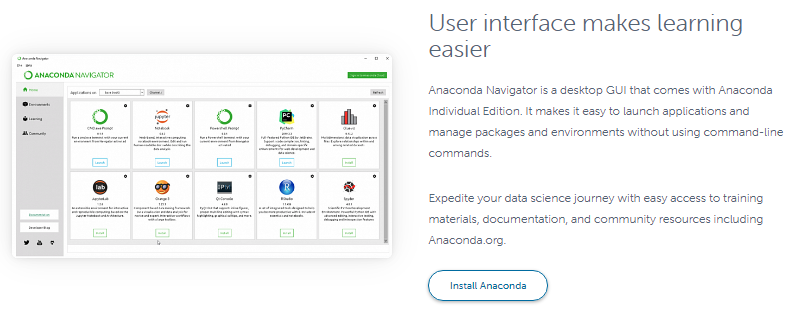
SKUCET, Dept of CSE Page 10

Marketing Automation

**2.3 SOFTWARE INSTALLATION**

#### 1. Visit the anaconda downloads page.

Go to the following link: [anaconda.com/downloads](https://www.anaconda.com/download/)



#### 2. Select windows.

Select windows where the three operating systems are listed.

#### 3. Download.

Choose python 3.6 version, 64 bit graphical installer.

4. Let it download in an .exe format.

#### 5. Open and run the installer.

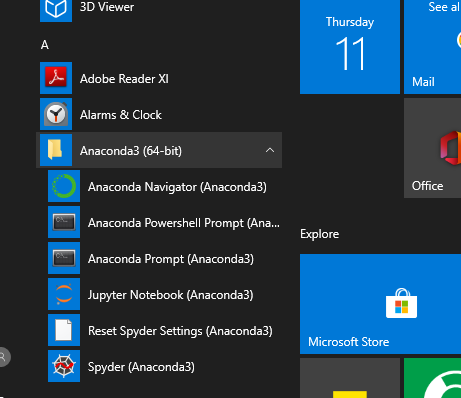
#### Once the download completes, open and run the .exe installer.

6. Click on next, i agree, install.

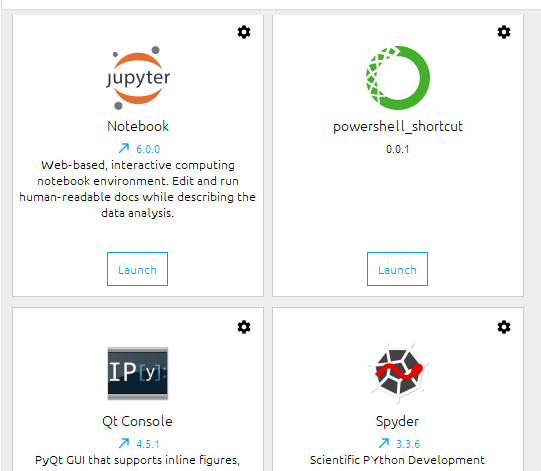
7. Completion of the installation, open your windows start menu and select the anaconda navigator.

SKUCET, Dept of CSE Page 11

Marketing Automation



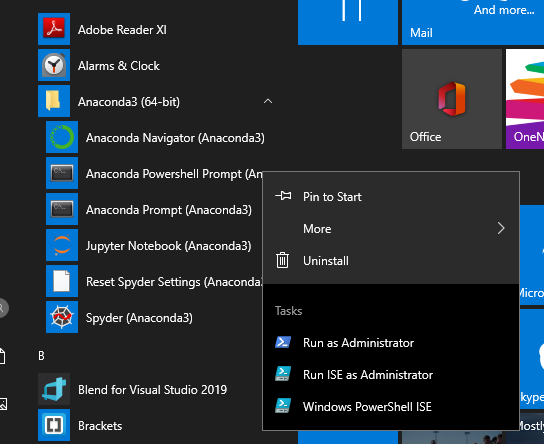
8. Click on jupiter launch button, your jupiter notebook will get start.



9. You need to install some packages to execute your project in a proper way.

10. Select windows start menu, right click on anaconda prompt, choose run as administrator.

SKUCET, Dept of CSE Page 12



11. Anaconda prompt will get open, with specified path, type “pip install package name” which you want to install (like jumpy, pandas, seaborne, sickie learn, matplotlib, pilot)

Ex: pip install numpy



12.You can also install required packages in Jupiter notebook directly by using the syntax as “! P

Ip install package name.

Ex: pip install numpy

SKUCET, Dept of CSE Page 13

Marketing Automation

**2.4 SYSTEM DESIGN**

**2.4.1 UML Diagrams:**

UML stands for unified modelling language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the object management group.

The goal is for uml to become a common language for creating models of object oriented computer software. In its current form uml is comprised of two major components: a meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, uml.

The unified modelling language is a standard language for specifying, visualization, constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems. The uml represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The uml is a very important part of developing objects-oriented software and the software development process. The uml uses mostly graphical notations to express the design of software projects.

**Goals:**

The primary goals in the design of the uml are as follows:

1. Provide users a ready-to-use, expressive visual modelling language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of oo tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

SKUCET, Dept of CSE Page 14

Marketing Automation

**2.4.2 Use case diagram:**

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.

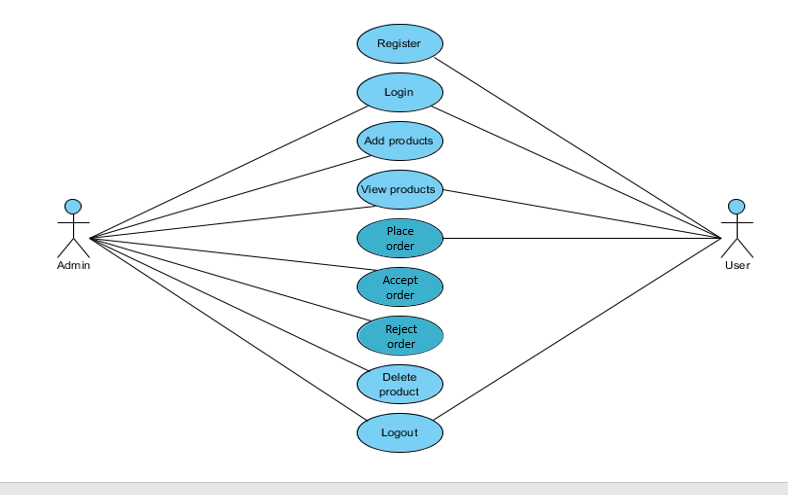


Fig 2.3.2 Use case diagram

SKUCET, Dept of CSE Page 15

Marketing Automation

**2.4.3 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.

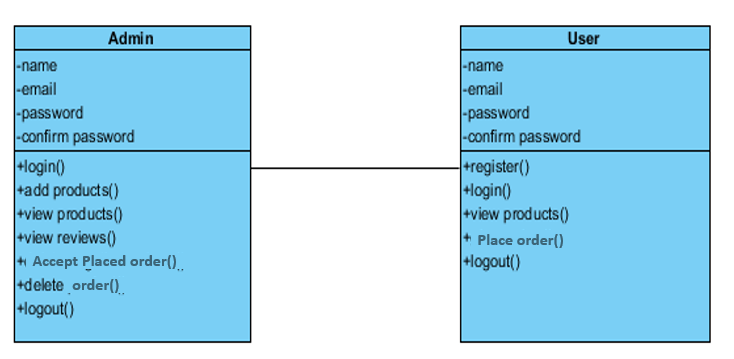


Fig 2.3.3 class diagram

**2.4.4 Sequence diagram:**

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.

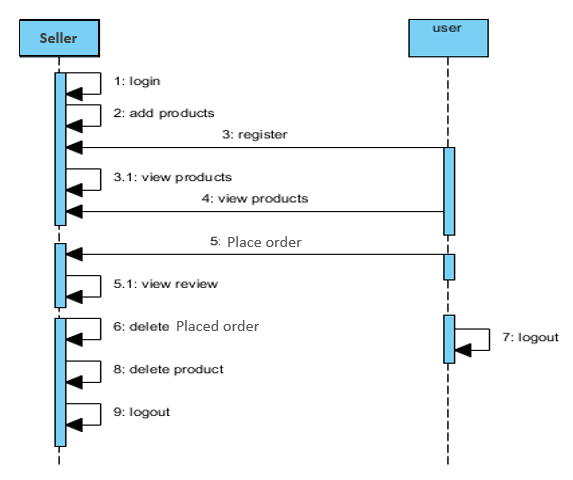
****

Fig 2.3.4 sequence diagram

SKUCET, Dept of CSE Page 16

Marketing Automation

**2.4.5 Collaboration diagram:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

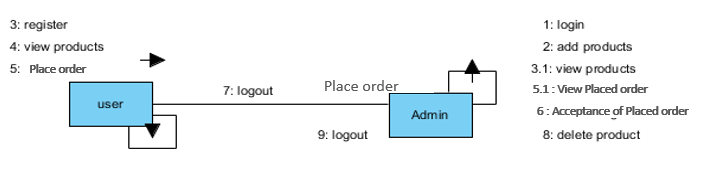
****

Fig 2.3.5 Collaboration diagram

**2.4.6 Deployment diagram:**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hard ware’s used to deploy the application.

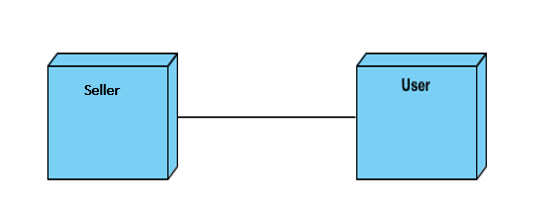
****

Fig 2.3.6 Deployment Diagram

SKUCET, Dept of CSE Page 17

Marketing Automation

**2.4.7 Component diagram:**

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executable, libraries etc. So the purpose of this diagram is different, component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details. Initially the system is designed using different uml diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.

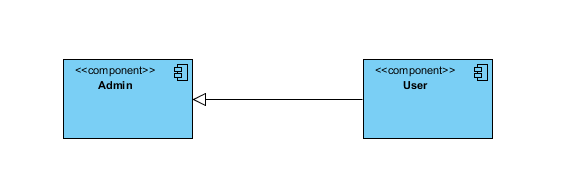


Fig 2.3.7 Component Diagram

**2.4.8 Activity diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the unified modelling 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.

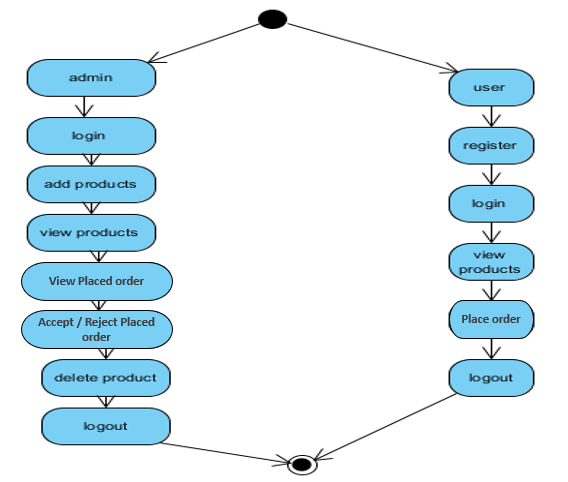
****

Fig.2.3.8 Activity Diagram

SKUCET, Dept of CSE Page 18

Marketing Automation

**2.4.9 ER Diagram:**

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.

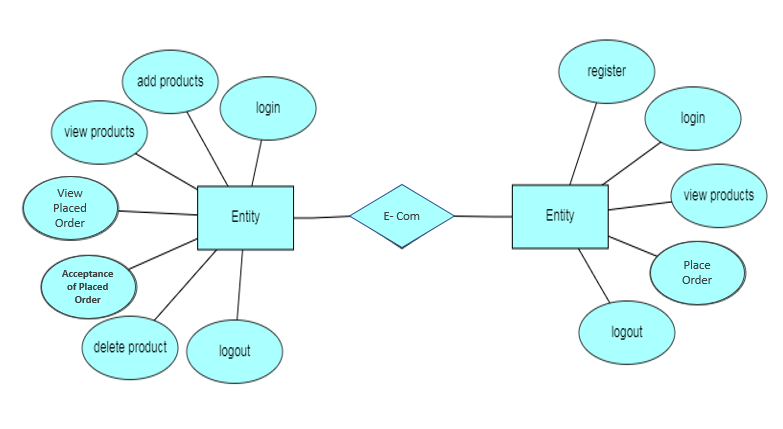


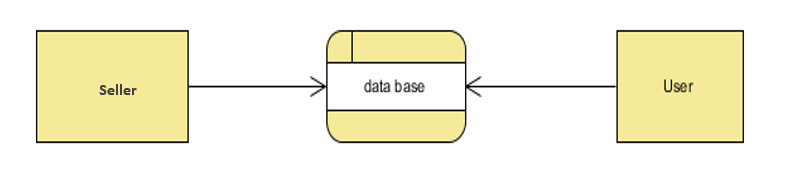
Fig 2.3.9 ER Diagram

**2.4.10 DFD Diagram:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

SKUCET, Dept of CSE Page 19

Marketing Automation



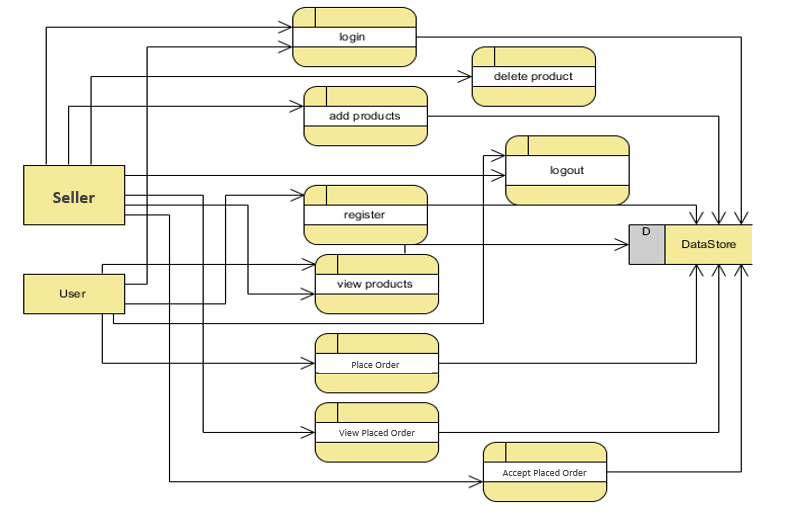


Fig 2.3.10 DFD Diagram

SKUCET, Dept of CSE Page 20

Marketing Automation

**2.5 BLOCK DIAGRAM:**

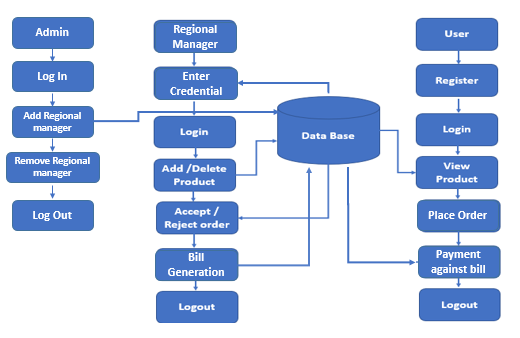
****

Fig.2.4.1 Block Diagram

SKUCET, Dept of CSE Page 21

Marketing Automation

**2.6 SYSTEM ARCHITECTURE**

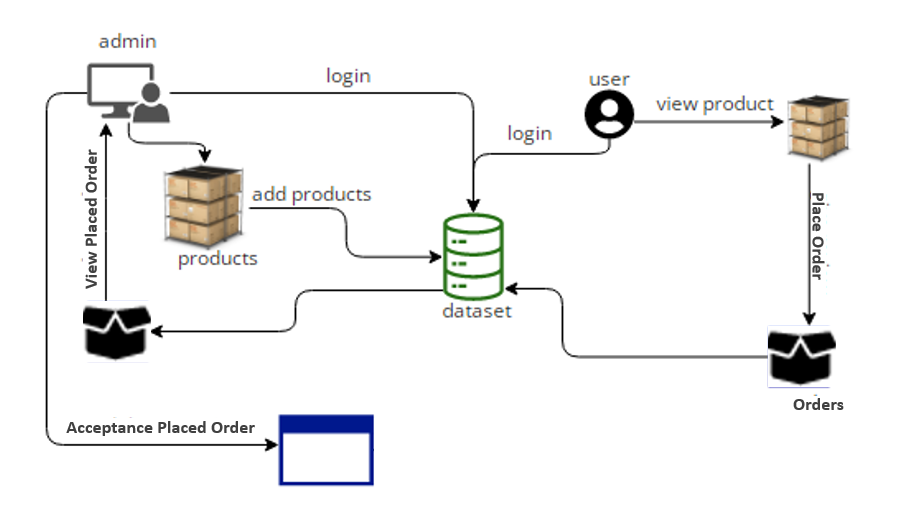
****

Fig 2.4.1 System Architecture

The Internet has become a major resource in modern business, thus electronic shopping has gained significance not only from the entrepreneur’s but also from the customer’s point of view. For the entrepreneur, electronic shopping generates new business opportunities and for the customer, it makes comparative shopping possible. As per a survey, most consumers of online stores are impulsive and usually make a decision to stay on a site within the first few seconds. “Website design is like a shop interior. If the shop looks poor or like hundreds of other shops the customer is most likely to skip to the other site. Hence we have designed the project to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible. In this project, the user is provided with an ecommerce web site that can be used to buy books online.

SKUCET, Dept of CSE Page 22

Marketing Automation

**CHAPTER 3**

# 3.1 INTRODUCTION TO PYTHON

**3.1.1 Python**

### What is a script?

Up to this point, i have concentrated on the interactive programming capability of python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable**

Basically, a script is a text file containing the statements that comprise a python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable**

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**You will need a text editor**

Just about any text editor will suffice for creating python script files.

You can use *Microsoft notepad, Microsoft WordPad, Microsoft word,*or just about any word processor if you want to.

**Difference between a script and a program**

**Script:**

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

**Program:**

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived .

**Python**

What is python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages.

SKUCET, Dept of CSE Page 23

Marketing Automation

**3.1.2 PYTHON CONCEPTS**

If you’re not interested in the the hows and whys of python, feel free to skip to the next chapter. In this chapter i will try to explain to the reader why i think python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object oriented, procedural, functional

• Easy to interface with c/objc/java/fortran.

• Easy-is to interface with c++ (via swig).

• Great interactive environment.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

**Python is interpreted** − python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to perl and php.

**Python is interactive** − you can actually sit at a python prompt and interact with the interpreter directly to write your programs.

**Python is object-oriented** − python supports object-oriented style or technique of programming that encapsulates code within objects.

**Python is a beginner's language** − python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to www browsers to games.

**History of python**

Python was developed by guido van possum in the late eighties and early nineties at the national research institute for mathematics and computer science in the netherlands. Python is derived from many other languages, including abc, modula-3, c, c++, algol-68, smalltalk, and unix shell and other scripting languages.

Python is copyrighted. Like perl, python source code is now available under the gnu general public license (gpl). Python is now maintained by a core development team at the institute, although guido van possum still holds a vital role in directing its progress.

SKUCET, Dept of CSE Page 24

Marketing Automation

**Python features**

Python's features include –

* **Easy-to-learn** − python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − python's source code is fairly easy-to-maintain.
* **A broad standard library** − python's bulk of the library is very portable and cross-platform compatible on unix, windows, and macintosh.
* **Interactive mode** − python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − you can add low-level modules to the python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − python provides interfaces to all major commercial databases.
* **Gui programming** − python supports gui applications that can be created and ported to many system calls, libraries and windows systems, such as windows mfc, macintosh, and the x window system of unix.
* **Scalable** − python provides a better structure and support for large programs than shell scripting.
* Apart from the above-mentioned features, python has a big list of good features, few are listed below
* It supports functional and structured programming methods as well as oop.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with c, c++, com, activex, corba, and java.

SKUCET, Dept of CSE Page 25

Marketing Automation

**Dynamic vs. Static**

Type’s python is a dynamic-typed language. Many other languages are static typed, such as c/c++ and java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in c if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type. This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. A number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program. Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. A floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.With python, it doesn’t matter. You simply give it whatever number you want and python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

**Variables**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

SKUCET, Dept of CSE Page 26

Marketing Automation

**Standard data types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## **Python numbers**

Number data types store numeric values. Number objects are created when you assign a value to them.

## **Python strings**

Strings in python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## **Python lists**

Lists are the most versatile of python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in c. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

## **Python tuples**

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses. The main differences between lists and tuples are: tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

SKUCET, Dept of CSE Page 27

Marketing Automation

**Python dictionary**

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in perl and consist of key-value pairs. A dictionary key can be almost any python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python:**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# 20 python libraries:

**1.** Requests. The most famous http library written by kenneth remits. It’s a must have for every python developer.

**2.** Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

**3.** Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

**4.** Pillow. A friendly fork of pil (python imaging library). It is more user friendly than pil and is a must have for anyone who works with images.

**5.** Sqlalchemy. A database library. Many love it and many hate it. The choice is yours.

**6.** Beautiful soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

**7.** Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

**8.** Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

**9.** Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

**11.** Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

SKUCET, Dept of CSE Page 28

Marketing Automation

**12.** Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

**13.** Pit. A gui toolkit for python. It is my second choice after python for developing gui’s for my python scripts.

**14.** Pit. Another python gui library. It is the same library in which the famous bit torrent client is created.

**15.** Scaly. A packet sniffer and analyzer for python made in python.

**16.** Pywin32. A python library which provides some useful methods and classes for interacting with windows.

**17.** Notch. Natural language toolkit – i realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

**18.** Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

**19.** Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure python distribution.

**20.** I python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Numpy**

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• offers matlab-ish capabilities within python

• fast array operations

• 2d arrays, multi-d arrays, linear algebra etc.

**Matplotlib**

• high quality plotting library.

SKUCET, Dept of CSE Page 29

Marketing Automation

**Python class and objects**

These are the building blocks of oop. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. A chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to oop later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something. Now will move into the big, scary world of object-oriented programming (oop). To be honest, it took me several months to get a handle on objects.

When i first learned c and c++, i did great; functions just made sense for me. Having messed around with basic in the early ’90s, i realized functions were just like subroutines so there wasn’t much new to learn.However, when my c++ course started talking about objects, classes, and all the new features of oop, my grades definitely suffered.

Once you learn oop, you’ll realize that it’s actually a pretty powerful tool. Plus many python libraries and apis use classes, so you should at least be able to understand what the code is doing. One thing to note about python and oop: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With python, you can get as technical as you want.

As you’ve already seen, python can do just fine with functions. Unlike languages such as java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code**.** Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of python oop ideas:

• the class statement creates a class object and gives it a name. This creates a new namespace.

• assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: classname.attribute.

• class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• calling a class (just like a function) creates a new instance of the class.

SKUCET, Dept of CSE Page 30

Marketing Automation

**Inheritance**

First off, classes allow you to modify a program without really making changes to it. To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class. However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses. The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all. Operator overloads Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to python’s object model and the object interfaces are more consistent to python’s built-in objects, hence overloading is easier to learn and use. User-made classes can override nearly all of python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now i will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. Trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in python. Virtually every module in the standard python library uses them, and python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• accessing a non−existent dictionary key will raise a key error exception.

• searching a list for a non−existent value will raise a value error exception

• calling a non−existent method will raise an attribute error exception.

• referencing a non−existent variable will raise a name error exception.

• mixing data types without coercion will raise a type error exception.

SKUCET, Dept of CSE Page 31

Marketing Automation

**3.2 PYTHON MODULES**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the python library.

To support this, python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

* As indicated above, code is usually developed in a file using an editor.
* To test the code, import it into a python session and try to run it.
* Usually there is an error, so you go back to the file, make a correction, and test again.
* This process is repeated until you are satisfied that the code works. T
* His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid. This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

**Functions in python**

It is possible, and very useful, to define our own functions in python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

SKUCET, Dept of CSE Page 32

Marketing Automation

## **Python namespace**

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname). An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (like the same file names in different directories) programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in python are implemented as python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a python program and when using namespaces.

Some namespaces in python:

* **Global names** of a module
* **Local names** in a function or method invocation
* **Built-in names**: this namespace contains built-in functions (e.g. Abs(), camp(), ...) And built-in exception names

**Garbage collection**

Garbage collector exposes the underlying memory management mechanism of python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

When attempting to add an object to a reference counter, a cyclical reference or reference cycle is produced. Because the object’s reference counter could never reach 0 (due to cycle), a reference counter cannot destroy the object. Therefore, in situations like this, we employ the universal waste collector. It operates and releases the memory used. A Generational Garbage Collector can be found in the standard library’s gc module.

SKUCET, Dept of CSE Page 33

Marketing Automation

**Python xml parser**

* + Xml is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.
  + What is xml? The extensible markup language xml is a markup language much like html or sgml.
  + This is recommended by the World Wide Web consortium and available as an open standard.
  + Xml is extremely useful for keeping track of small to medium amounts of data without requiring a sql-based backbone.
  + Xml parser architectures and apis the python standard library provides a minimal but useful set of interfaces to work with xml.
  + The two most basic and broadly used apis to xml data are the sax and dom interfaces.
  + Simple api for xml sax: here, you register callbacks for events of interest and then let the parser proceed through the document.
  + This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.
  + Document object model dom api : this is a world wide web consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an xml document.

Sax obviously cannot process information as fast as dom can when working with large files. On the other hand, using dom exclusively can really kill your resources, especially if used on a lot of small files.

Sax is read-only, while dom allows changes to the xml file. Since these two different apis literally complement each other, there is no reason why you cannot use them both for large projects.

As you can see, the hierarchical XML file data has been converted to a simple CSV file so that all news stories are stored in form of a table. This makes it easier to extend the database too.  
Also, one can use the JSON-like data directly in their applications! This is the best alternative for extracting data from websites which do not provide a public API but provide some RSS feeds.

SKUCET, Dept of CSE Page 34

Marketing Automation

**3.3 PYTHON WEB FRAMEWORKS**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common http operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. Url routing
2. Html, xml, json, and other output format tinplating
3. Database manipulation
4. Security against cross-site request forgery (csrf) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying python web frameworks, tinplating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part i agree although i've found sessions and database orms to be a helpful part of a framework when done well.

SKUCET, Dept of CSE Page 35

Marketing Automation

* What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
* Jingo vs. Flash vs. Pyramid: choosing a python web framework contains background information and code comparisons for similar web applications built in these three big python frameworks.
* This fascinating blog post takes a look at the code complexity of several python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to json and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? Is a language agnostic reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main python frameworks.
* This user-voted question & answer site asked "what are the best general purpose python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to python developers.

## Web frameworks learning checklist

1. Choose a major python web framework (jingo or flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

SKUCET, Dept of CSE Page 36

Marketing Automation

**CHAPTER 4**

**PROJECT CODE AND SCREENSHOTS**

**4.1 PROJECT CODE**

**4.1.1Admin.py**

from email import message

from django.shortcuts import render,redirect

from django.contrib import messages

from user.models import userorders

from . models import Regionalmanager

from django.conf import settings

from django.core.mail import send\_mail

def index(request):

return render(request,'index.html')

def adminlogin(request):

if request.method == "POST":

email = request.POST['email']

password = request.POST['password']

if email == "admin@gmail.com" and password == "admin":

messages.success(request,'Welcome to Online Supermarket Admin')

return render(request,'adminhome.html')

else:

messages.warning(request,'Admin Details are not valid')

return render(request,"admin.html")

return render(request,"admin.html")

SKUCET, Dept of CSE Page 37

Marketing Automation

def addregionalmanager(request):

if request.method == "POST":

name = request.POST['name']

email = request.POST['email']

password = request.POST['password']

contact = request.POST['contact']

address = request.POST['address']

print(name,password)

info = Regionalmanager.objects.filter(useremail=email,password=password).exists()

print(info)

if info == True:

messages.warning(request,"Details already Exists")

return render(request,'addregionalmanager.html')

else:

Regionalmanager.objects.create(username = name,useremail = email,password = password,contact = contact,Address = address)

message = f'Hi {name}'

subject =" Online Super Market "

# m1 = "This message is automatic generated so dont reply to this Mail"

# m2 = "Thanking you"

# m3 = "Regards"

# m4 = "Admin"

# email\_from = settings.EMAIL\_HOST\_USER

# recipient\_list = [email]

SKUCET, Dept of CSE Page 38

Marketing Automation

# text = message + '\n' + '\n' + m1 + '\n' + m2 + '\n' + m3 + '\n' + m4

# send\_mail(subject, text, email\_from, recipient\_list,fail\_silently=False,)

messages.warning(request,"Details added Successfuly")

return render(request,'addregionalmanager.html')

return render(request,'addregionalmanager.html')

def viewregionalmanager(request):

data = Regionalmanager.objects.all()

return render(request,"viewregionalmanager.html",{'data':data})

def deleteregionalmanager(id):

print(id)

Regionalmanager.objects.filter(id=id).delete()

return redirect("viewregionalmanager")

def soldproducts(request):

data = userorders.objects.all()

return render(request,'soldproducts.html',{'data':data})

**4.1.2 Regional Manager**

from itertools import count

from unicodedata import category

from django.shortcuts import render,redirect

from django.contrib import messages

from adminapp. models import Regionalmanager

from . models import Product,Category,Addemployee

SKUCET, Dept of CSE Page 39

Marketing Automation

from user. models import userdetails,userorders,feedback

import os

from django.core.files.storage import default\_storage

from django.conf import settings

from django.core.mail import send\_mail

def regionalmanager(request):

if request.method=="POST":

email = request.POST['email']

password = request.POST['password']

print(email)

print(password)

data = Regionalmanager.objects.filter(useremail = email,password = password).exists()

print(data)

if data == True:

return render(request,'regionalmanagerhome.html')

else:

messages.warning(request,"Details are not valid")

return render(request,"regionalmanager.html")

return render(request,"regionalmanager.html")

def addemployee(request):

if request.method == "POST":

employeename = request.POST['employeename']

employeeemail = request.POST['employeeemail']

SKUCET, Dept of CSE Page 40

Marketing Automation

employeeage = request.POST['employeeage']

employeepassword = request.POST['employeepassword']

employeecontact = request.POST['employeecontact']

employeeaddress = request.POST['employeeaddress']

data = Addemployee.objects.filter(employeeemail=employeeemail,employeepassword=employeepassword).exists()

if data ==True:

messages.warning(request,"Details already Exists")

return render(request,"addemployee.html")

else:

d = Addemployee(employeename=employeename,employeeemail=employeeemail,employeeage=employeeage,employeepassword=employeepassword,employeecontact=employeecontact,employeeaddress=employeeaddress)

d.save()

messages.success(request,"Details added Successfully")

# m1 = "This message is automatic generated so dont reply to this Mail"

# m2 = "Thanking you"

# m3 = "Regards"

# m4 = "Cloud Service Provider."

# con = "Your registration is completed you can login now"

# Email = employeeemail

# subject = " Online Super Market"

# email\_from = settings.EMAIL\_HOST\_USER

SKUCET, Dept of CSE Page 41

Marketing Automation

# recipient\_list = [Email]

# text = m1 + '\n' + con + '\n' + m2 + '\n' + m3 + '\n' + m4

# send\_mail(subject, text, email\_from, recipient\_list,fail\_silently=False,)

return render(request,"addemployee.html")

return render(request,"addemployee.html")

def addcategories(request):

if request.method=="POST":

print("-------------------")

category = request.POST['category']

productname = request.POST['productname']

productimage = request.FILES['productimage']

print(productimage)

default\_storage.save(productimage.name,productimage)

a = Category(category=category,productname = productname ,productimage =productimage,filename = productimage)

a.save()

print(category,productname,productimage,productimage)

return render(request,'addcategories.html')

def viewcategories(request):

data = Category.objects.order\_by('category').values\_list('category','productname','filename').distinct()

return render(request,'viewcategories.html',{'data':data})

def addproducts(request,category):

print(category)

SKUCET, Dept of CSE Page 42

Marketing Automation

if request.method == "POST":

print(category)

categoryname = category

productname = request.POST['productname']

productquantity = request.POST['productquantity']

productprice = request.POST['productprice']

productimage = request.FILES['productimage']

default\_storage.save(productimage.name,productimage) Product(categoryname=categoryname,productname=productname,productquantity=productquantity,productprice=productprice,productimage=productimage,filename=productimage)

b.save()

return render(request,'addproducts.html',{'category':category})

def customproducts(request):

# obj=emp.objects.all()[:10]

# data = Category.objects.all()[:1]

data = Category.objects.order\_by('category').values\_list('category','productname','filename').distinct()

return render(request,'customproducts.html',{"data":data})

def viewproducts(request,category):

print(category)

c = Product.objects.filter(categoryname = category)

return render(request,"viewproducts.html",{'c':c})

def categoryproducts(request,category):

print(category)

dat = Product.objects.filter(category=category)

SKUCET, Dept of CSE Page 43

Marketing Automation

print(dat)

return render(request,"categoryproducts.html",{dat:dat})

def vieworderplaced(request):

try:

d=userorders.objects.filter(status='accepted',useremail=request.session['useremail'])

return render(request,'orderplaced.html',{'data':d})

except:

return render(request,'orderplaced.html')

def accept(request,id):

print(id)

info = userorders.objects.filter(id = id)

newdata = []

for i in info:

newdata.append(i.id)

newdata.append(i.productprice)

newdata.append(i.productname)

newdata.append(i.totalprice)

newdata.append(i.productcount)

print(newdata)

m1 = "This message is automatic generated so dont reply to this Mail"

m2 = "Thanking you"

m3 = "Regards"

m4 = "Cloud Service Provider."

con = "Your request is Accepted for"

SKUCET, Dept of CSE Page 44

Marketing Automation

productname = newdata[2]

prodcuctprice = newdata[1]

productcount = newdata[4]

total = newdata[3]

Email = request.session['useremail']

# subject = " Online Super Market"

# email\_from = settings.EMAIL\_HOST\_USER

# recipient\_list = [Email]

# text = m1 + '\n' + con + '\n'+ productname +'\n'+ str(prodcuctprice) +'\n'+ str(productcount) +'\n'+ str(total)+ '\n'+ m2 + '\n' + m3 + '\n' + m4

# send\_mail(subject, text, email\_from, recipient\_list,fail\_silently=False,)

d=userorders.objects.get(status='accepted',useremail=Email)

d.status="orderaccepted"

d.save()

return redirect(vieworderplaced)

def usersfeedback(request,productname):

print(productname)

print("--------------")

dc = Product.objects.filter(productname=productname)

dc = [i.filename for i in dc]

dc = dc[0]

print(dc)

d = feedback.objects.filter(product=productname)

print(d)

SKUCET, Dept of CSE Page 45

Marketing Automation

return render(request,'usersfeedback.html',{'data':d,"nameproduct":productname,"dc":dc})

def reject(request,id):

print(id)

info = userorders.objects.filter(id = id)

newdata = []

for i in info:

newdata.append(i.id)

newdata.append(i.productprice)

newdata.append(i.productname)

newdata.append(i.totalprice)

newdata.append(i.productcount)

print(newdata)

m1 = "This message is automatic generated so dont reply to this Mail"

m2 = "Thanking you"

m3 = "Regards"

m4 = "Admin."

productname = newdata[2]

prodcuctprice = newdata[1]

productcount = newdata[4]

total = newdata[3]

con = "Your order is rejected because of Out of Stock"

Email = request.session['useremail']

subject = " Online Super Market"

email\_from = settings.EMAIL\_HOST\_USER

SKUCET, Dept of CSE Page 46

Marketing Automation

recipient\_list = [Email]

text = m1 + '\n' + con + '\n'+ productname +'\n'+ prodcuctprice +'\n'+ productcount +'\n'+ total+ '\n'+ m2 + '\n' + m3 + '\n' + m4

send\_mail(subject, text, email\_from, recipient\_list,fail\_silently=False,)

d=userorders.objects.get(status='accepted',useremail=Email)

d.status="orderrejected"

d.save()

return redirect(vieworderplaced)

def deletecategory(request,category,productname):

print(category)

print(productname)

Category.objects.filter(category=category,productname=productname).delete()

return redirect("viewcategories")

def deleteproduct(request,productname):

print(request.session['useremail'])

print(productname)

# userorders.objects.filter(productname=productname,useremail=request.session['useremail']).delete()

return redirect("viewcategories")

**4.1.3 Employee**

from django.shortcuts import render

from regionalmanagerapp . models import Addemployee

from django.contrib import messages

SKUCET, Dept of CSE Page 47

Marketing Automation

# Create your views here.

def employee(request):

if request.method =="POST":

email = request.POST['email']

password = request.POST['password']

d = Addemployee.objects.filter(employeeemail=email,employeepassword=password).exists()

print(d)

if d == True:

messages.success(request,"Employee Login Success")

return render(request,"employeehome.html")

else:

messages.success(request,"details are not valid")

return render(request,"employeelogin.html")

return render(request,"employeelogin.html")

**4.1.4 User**

from django.shortcuts import render,redirect

from django.contrib import messages

from adminapp. models import Regionalmanager

from regionalmanagerapp. models import Product,Category

from . models import userdetails,userorders,feedback

from django.contrib.sessions.models import Session

def userlogin(request):

if request.method == "POST":

SKUCET, Dept of CSE Page 48

Marketing Automation

useremail = request.POST['useremail']

password = request.POST['password']

info = userdetails.objects.filter(useremail = useremail,password=password).exists()

if info == True:

request.session['useremail'] = useremail

print(request.session['useremail'])

return render(request,'userhome.html')

else:

messages.warning(request,"Details are not valid")

return render(request,'userlogin.html')

return render(request,'userlogin.html')

def userregister(request):

if request.method == "POST":

username = request.POST['username']

useremail = request.POST['useremail']

address = request.POST['address']

contact = request.POST['contact']

password = request.POST['password']

confirmpassword = request.POST['confirmpassword']

if password == confirmpassword:

data = userdetails.objects.filter(useremail = useremail,password=password).exists()

if data == False:

a = userdetails(username=username,useremail=useremail,address=address,contact=contact,password=password)

SKUCET, Dept of CSE Page 49

Marketing Automation

a.save()

return render(request,'userlogin.html')

else:

messages.warning(request,"Details already Exist")

return render(request,'userregister.html')

else:

messages.warning(request,"Password and Confirm Password Not matched")

return render(request,'userregister.html')

return render(request,'userregister.html')

def userproducts(request):

# data = Category.objects.filter('category')[:1]

data = Category.objects.order\_by('category').values\_list('category','productname','filename').distinct()

return render(request,'userproducts.html',{'data':data})

def allcategory(request,category):

data = Product.objects.filter(categoryname=category)

return render(request,'allcategory.html',{'data':data})

def searchdata(request):

if request.method=="POST":

searchitem = request.POST['searchitem']

print(searchitem)

return render(request,"searchdata.html")

def additem(request):

if request.method=="POST":

categoryname = request.POST['categoryname']

SKUCET, Dept of CSE Page 50

Marketing Automation

productname = request.POST['productname']

productprice = request.POST['productprice']

productquantity = request.POST['productquantity']

print(categoryname,productname)

s = Product.objects.filter(categoryname = categoryname,productname = productname)

print(s)

datalist = []

for i in s:

datalist.extend([i.categoryname,i.productname,i.productquantity,i.productprice,i.productimage,i.filename])

print(datalist)

print()

orderdetails = datalist

category = orderdetails[0]

productname = orderdetails[1]

productquantity = orderdetails[2]

productprice = orderdetails[3]

totalprice = orderdetails[3]

productcount = orderdetails[2]

useremail = request.session['useremail']

d = userorders.objects.filter(productname = productname).exists()

if d == False:

d = userorders(productname=productname,productprice=productprice,totalprice=totalprice,productquantity=productquantity,productcount=productcount,useremail=useremail)

SKUCET, Dept of CSE Page 51

Marketing Automation

d.save()

else:

d = userorders.objects.get(productname=productname,useremail=useremail)

d.totalprice = int(d.productprice) + int(d.totalprice)

d.productcount = int(d.productquantity) + int(d.productcount)

d.save()

data1 = userorders.objects.filter(productname=productname)

data = Product.objects.filter(categoryname=category)

return render(request,'allcategory.html',{'data1':data1,'data':data,'categoryname':category,'productname':productname})

def removeitem(request):

if request.method == "POST":

categoryname = request.POST['categoryname']

productname = request.POST['productname']

productprice = request.POST['productprice']

productquantity = request.POST['productquantity']

print("--------")

print(categoryname,productname,productprice,productquantity)

d = userorders.objects.filter(productname = productname).exists()

print(d)

nes =[]

if d ==True:

s = userorders.objects.filter(productname = productname)

SKUCET, Dept of CSE Page 52

Marketing Automation

print(s)

for i in s:

nes.extend([i.productname,i.productprice,i.totalprice,i.productquantity,i.productcount])

print("===============")

print(nes)

productprice = nes[1]

totalprice = nes[2]

productcount = nes[4]

s = userorders.objects.get(productname = productname)

s.productcount = int(productcount)-1

print("============")

print(s.totalprice)

print("============")

if s.totalprice == 0:

s.totalprice = 0

else:

s.totalprice = int(totalprice) - int(productprice)

s.save()

else:

messages(request,"You did'nt selected any item")

data1 = userorders.objects.filter(productname=productname)

data = Product.objects.filter(categoryname=categoryname)

return render(request,'allcategory.html',{'data1':data1,'data':data,'categoryname':categoryname,"productname":productname})

SKUCET, Dept of CSE Page 53

Marketing Automation

def viewproductfeedback(request):

if request.method == "POST":

print("===============")

nameproduct = request.POST['nameproduct']

print(nameproduct)

dc = Product.objects.filter(productname=nameproduct)

dc = [i.filename for i in dc]

dc= dc[0]

d = feedback.objects.filter(product=nameproduct)

print(d)

return render(request,'feedback.html',{'data':d,"nameproduct":nameproduct,"dc":dc})

def addfeedback(request):

if request.method =="POST":

feedbackmsg = request.POST['feedbackmsg']

categoryname = request.POST['categoryname']

productname = request.POST['productname']

productprice = str(request.POST['productprice'])

productquantity = request.POST['productquantity']

a = feedback(product=productname,price=productprice,category=categoryname,feedback=feedbackmsg,useremail=request.session['useremail'])

a.save()

return redirect("userproducts")

def placeorder(request):

data1 = userorders.objects.filter(status='pending' , useremail=request.session['useremail'])

SKUCET, Dept of CSE Page 54

Marketing Automation

return render(request,'placeorder.html',{'data':data1})

def orderplaced(request):

dat = userorders.objects.get(status='pending',useremail=request.session['useremail'])

dat.status = "accepted"

dat.save()

return redirect("placeorder")

def payment(request):

data1 = userorders.objects.filter(status='orderaccepted', useremail=request.session['useremail'])

return render(request, 'payment.html', {'data': data1})

def pay(request,id):

print(id)

data1 = userorders.objects.filter(id=id,status='orderaccepted', useremail=request.session['useremail'])

print(data1)

return render(request,"pay.html", {'data': data})

**4.1.5 Setting.py**

from pathlib import Path

import os

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_file\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-70pf)(^xrv$q&i^8x2e1m$tuzkcf0pbrj\_$u=t)5\*pa\_yf#-=0'

SKUCET, Dept of CSE Page 55

Marketing Automation

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'adminapp.apps.AdminappConfig',

'regionalmanagerapp.apps.RegionalmanagerappConfig',

'user.apps.UserConfig',

'Employee.apps.EmployeeConfig',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',]

SKUCET, Dept of CSE Page 56

Marketing Automation

ROOT\_URLCONF = 'supermarket.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE\_DIR,'templates')],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

# mail code

EMAIL\_BACKEND="django.core.mail.backends.smtp.EmailBackend"

EMAIL\_HOST="smtp.elasticemail.com"

EMAIL\_USE\_TLS=True

EMAIL\_USE\_SSL=False

EMAIL\_HOST\_USER="ganapathideepak1234@gmail.com"

EMAIL\_HOST\_PASSWORD="tddtlqcnpvwsvwml"

EMAIL\_PORT='2525'

SKUCET, Dept of CSE Page 57

Marketing Automation

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

# static files

STATIC\_URL = '/static/'

STATICFILES\_DIRS=[os.path.join(BASE\_DIR,'static')]

STATIC\_ROOT=os.path.join('assets')

WSGI\_APPLICATION = 'supermarket.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'supermarket',

'USER':'root',

'PASSWORD':'',

'PORT':'3306'

}

}

# Password validation

# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

SKUCET, Dept of CSE Page 58

Marketing Automation

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.2/howto/static-files/

# Default primary key field type

# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

SKUCET, Dept of CSE Page 59

Marketing Automation

**4.1.6 Manage.py**

#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

"""Run administrative tasks."""

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'supermarket.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

if \_name\_ == '\_main\_':

main()

SKUCET, Dept of CSE Page 60

Marketing Automation

**4.2 SCREENSHOTS**

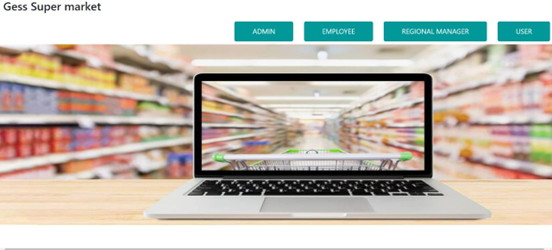


Fig. 4.2.1 Home page of Supermarket

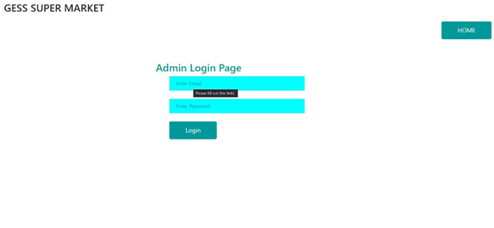


Fig 4.2.2 Admin Login Page

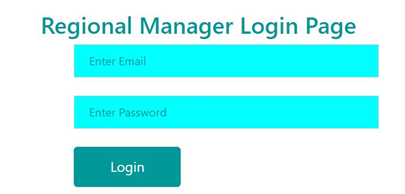


Fig. 4.2.3 Regional Manager Login Page

SKUCET, Dept of CSE Page 61

Marketing Automation

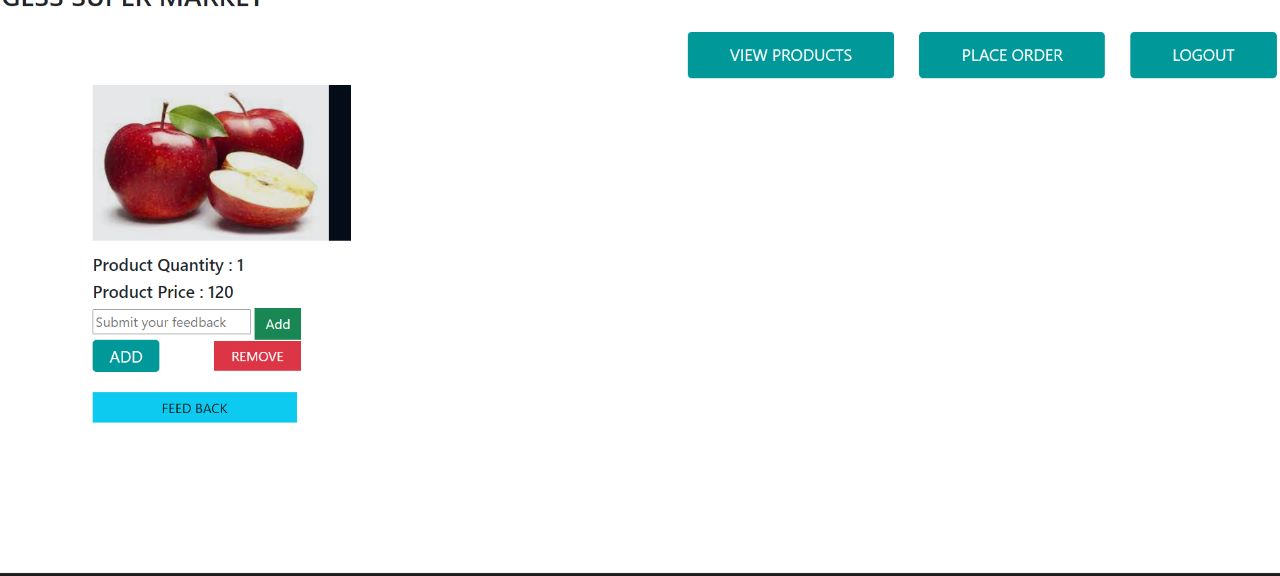


Fig.4.2.4 Product List page

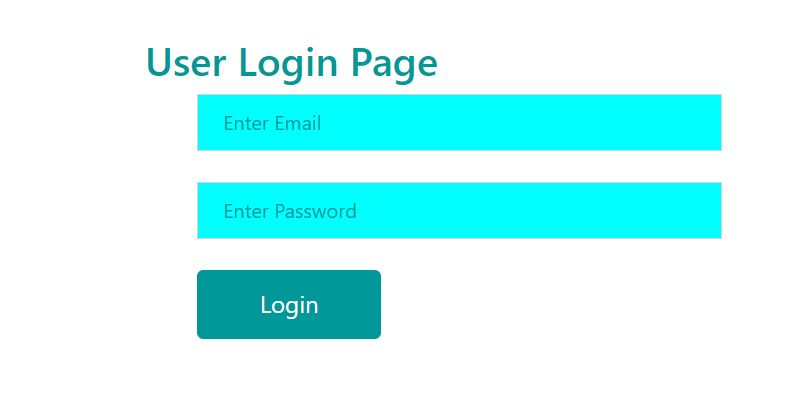


Fig 4.2.5 User Login Page



Fig. 4.2.6 Employee adding page

SKUCET, Dept of CSE Page 62

Marketing Automation

**4.3 SYSTEM STUDY**

**Feasibility study**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* Economical feasibility
* Technical feasibility
* Social feasibility

**Economical feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **Technical feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

SKUCET, Dept of CSE Page 63

Marketing Automation

### **4.4 SYSTEMTESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**4.4.1 Types of tests**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid input : identified classes of valid input must be accepted.

Invalid input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

SKUCET, Dept of CSE Page 64

Marketing Automation

**4.4.2 unit testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 4.4.3 integration testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. Components in a software system or – one step up – software applications at the company level – interact without error.

**Test results:** all the test cases mentioned above passed successfully. No defects encountered.

**4.4.4 acceptance testing**

User acceptance testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test results:** all the test cases mentioned above passed successfully. No defects encountered.

SKUCET, Dept of CSE Page 65

Marketing Automation

**CHAPTER 5**

**CONCLUSION**

* The sales growth is calculated by using the growth of the sales as per growth depends on the Advertisement.
* Accuracy is calculated by using different algorithms. The sales growth have been calculated with the better accuracy and precision.
* This would be of great help for the people. To achieve these results, various data mining techniques are utilized in python language.
* Thus, we studied and applied the concept of polynomial Regression in real time implementation as to ease the life of human..

SKUCET, Dept of CSE Page 66

Marketing Automation

**References**

**1.**S. Erevelles, N. Fukawa and L. Swayne, "Big data consumer analytics and the transformation of marketing", Journal of Business Research, vol. 69, no. 2, pp. 897-904, 2016.

**2.**P. Russom et al., "Big data analytics", TDWI best practices report fourth quarter, pp. 1-35, 2011.

**3.**D.N. Devi, C.K. Kumar and S. Prasad, "A feature based approach for sentiment analysis by using support vector machine", Advanced Computing (IACC) 2016 IEEE 6th International Conference on, pp. 3-8, 2016.

**4.**V. Narayanan, I. Arora and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model", International Conference on Intelligent Data Engineering and Automated Learning, pp. 194-201, 2013.

**5.**V. Hatzivassiloglou and K.R. McKeown, "Predicting the semantic orientation of adjectives", Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, pp. 174-181, 1997.

**6.**T. Wilson, J. Wiebe and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis", Proceedings of the conference on human language technology and empirical methods in natural language processing, pp. 347-354, 2005.

**7.**A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining", LREc, vol. 10, no. 2010, 2010.

**8.**M. Wahyudi and D.A. Kristiyanti, "Sentiment analysis of smartphone product review using support vector machine algorithm-based particle swarm optimization", Journal of Theoretical & Applied Information Technology, vol. 91, no. 1, 2016.

**9.**S. Tan and J. Zhang, "An empirical study of sentiment analysis for chinese documents", Expert Systems with applications, vol. 34, no. 4, pp. 2622-2629, 2008.

**10.**Q. Ye, Z. Zhang and R. Law, "Sentiment classification of online reviews to travel destinations by supervised machine learning approaches", Expert Systems with Applications, vol. 36, no. 3, pp. 6527-6535, 2009.

**11.**L.-C. Yu, J.-L. Wu, P.-C. Chang and H.-S. Chu, "Using a contextual entropy model to expand emotion words and their intensity for the sentiment classification of stock market news", Knowledge-Based Systems, vol. 41, pp. 89-97, 2013.

SKUCET, Dept of CSE Page 67

Marketing Automation

**12.**M. Hagenau, M. Liebmann and D. Neumann, "Automated news reading: Stock price prediction based on financial news using context-capturing features", Decision Support Systems, vol. 55, no. 3, pp. 685-697, 2013.

**13.**T. Xu, Q. Peng and Y. Cheng, "Identifying the semantic orientation of terms using s-hal for sentiment analysis", Knowledge-Based Systems, vol. 35, pp. 279-289, 2012.

**14.**G. Qiu, X. He, F. Zhang, Y. Shi, J. Bu and C. Chen, "Dasa: dissatisfaction-oriented advertising based on sentiment analysis", Expert Systems with Applications, vol. 37, no. 9, pp. 6182-6191, 2010.

**15.**T.-K. Fan and C.-H. Chang, "Blogger-centric contextual advertising", Expert systems with applications, vol. 38, no. 3, pp. 1777-1788, 2011.

**16.**K.M. Leung, Naive bayesian classifier, Polytechnic University Department of Computer Science/Finance and Risk Engineering, 2007.

**17.**S. Tong and D. Koller, "Support vector machine active learning with applications to text classification", Journal of machine learning research, vol. 2, pp. 45-66, Nov 2001.

**18.**I. Jenhani, N.B. Amor and Z. Elouedi, "Decision Trees as possibilistic classifiers", International Journal of Approximate Reasoning*,* vol. 48, no. 3, pp. 784-807, 2008.

SKUCET, Dept of CSE Page 68

Marketing Automation

**STUDENT BIO DATA**

|  |
| --- |
| **Name:** A Hari Eswar |
| **Father Name:** A Dronadri |
| **Roll. No:** 1910101 |
| **Date of Birth:** 09-05-2000 |
| **Nationality:** Indian |
| **Communication Address:**  Town/Village: Peruru Mandal: Ramagiri District: Anantapur  PIN Code: 515621  Ph. No: 7569593689  e-mail: harieswaradavala1@gmail.com |
| **Permanent Address:**  Town/Village: Peruru Mandal: : Ramagiri District: Anantapur  PIN Code: 515621  Ph. No: 7569593689  e-mail: harieswaradavala1@gmail.com |
| **Qualifications:** B.Tech |
| **Technical Skills:** Python, MYSQL, Machine learning, Data science |
| **Area of Interest:** Data science |
| **Declaration:** I here by declare that the above information is true to belief of my knowledge.  **Signature** |

SKUCET, Dept of CSE Page 69

Marketing Automation

|  |
| --- |
| **Name:** Ganapathi Deepak |
| **Father Name:** G Dharma Reddy |
| **Roll. No:** 1910120 |
| **Date of Birth:** 18-01-2002 |
| **Nationality:** Indian |
| **Communication Address:**  Town/Village: Katherapalle Mandal: Karvetinagaram District: Chittoor  PIN Code: 517582  Ph. No: 9493573100  e-mail: ganapathideepak1234@gmail.com |
| **Permanent Address:**  Town/Village: Katherapalle Mandal: Karvetinagaram District: Chittoor  PIN Code: 517582  Ph. No: 9493573100  e-mail: ganapathideepak1234@gmail.com |
| **Qualifications**: B.tech |
| **Technical Skills:** Python, DSA, OOPS, SQL |
| **Area of Interest:** Software Development |
| **Declaration:** I here by declare that the above information is true to belief of my knowledge.  **Signature** |

SKUCET, Dept of CSE Page 70

Marketing Automation

|  |
| --- |
| **Name:** Nadiganti Sathish |
| **Father Name:** N Harinath |
| **Roll. No:** 1910146 |
| **Date of Birth:** 27-08-2002 |
| **Nationality:** Indian |
| **Communication Address:**  Town/Village: Bukkarayasamudram Mandal: Bukkarayasamudram District: Anantapur  PIN Code: 515701  Ph. No: 98498660891  e-mail: nidigantisathish@gmail.com |
| **Permanent Address:**  Town/Village: Bukkarayasamudram Mandal: Bukkarayasamudram District: Anantapur  PIN Code: 515701  Ph. No: 9849866091  e-mail: nidigantisathish@gmail.com |
| **Qualifications:** B.Tech |
| **Technical Skills**: Python, SQL |
| **Area of Interest:** Software Development |
| **Declaration:** I here by declare that the above information is true to belief of my knowledge.  **Signature** |

SKUCET, Dept of CSE Page 71

Marketing Automation

|  |
| --- |
| **Name:** Shaik Mohammed Swaleheen |
| **Father Name:** Shaik Musa Kalimulla |
| **Roll. No:** 1910155 |
| **Date of Birth:** 16-08-2002 |
| **Nationality:** Indian |
| **Communication Address:**  Town/Village: Musalimadugu Mandal: Kothapalli District: Nandyal  PIN Code: 518422  Ph. No: 6301650752  e-mail: swaleheen08@gmail.com |
| **Permanent Address:**  Town/Village: Musalimadugu Mandal: Kothapalli District: Nandyal  PIN Code: 518422  Ph. No: 63016750752  e-mail: swaleheen08@gmail.com |
| **Qualifications:** B.tech |
| **Technical Skills:** Java, Python,SQL |
| **Area of Interest**: Full Stack |
| **Declaration:** I here by declare that the above information is true to belief of my knowledge.  **Signature** |

SKUCET, Dept of CSE Page 72