

| Sr. No. | Contents | Page Number |
|---------|---------------------------|-------------|
| 1. | Introduction | 3 |
| | 1.1 Organisation Profile | 3 |
| | 1.2 Project Overview | 3 |
| | 1.3 Existing System | 4 |
| | 1.4 Study System | 4 |
| | 1.5 Project Scope | 5 |
| 2. | Literature Survey | 6 |
| | 2.1 Background | 6 |
| | 2.2 History | 6 |
| | 2.3 Programming Languages | 7 |
| 3. | Backend Technology | 14 |
| | 3.1 SQLite | |
| | 3.1.1 Feature of SQLite | |
| | 3.1.2 Connectivity | |
| | 3.1.3 | |
| | 3.1.4 Clients And Tools | |
| 4. | Software Tools | 18 |
| | 4.1 VS Code | |

| | | |
|-----|-------------------------------|----|
| 5. | System Development | 18 |
| | 5.1 Class Diagram | |
| | 5.1. 1 ER Diagram | 19 |
| | 5.2 DFD Diagram | 20 |
| | 5.2.1 Main Page DFD | 22 |
| | 5.2.2 Sign-in And Sign-up DFD | 24 |
| 6. | System Requirements | 29 |
| 7. | Code | 30 |
| 8. | Screenshot | 52 |
| 9. | Conclusion | 68 |
| 10. | References | 69 |

1. INTRODUCTION

INTRODUCTION:-

Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the Internet. It is a form of electronic commerce. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet by using an android device. Thus the customer will get the service of online shopping and home delivery from his favorite shop.

PROJECT OBJECTIVE:-

The objective of the project is to make an application in android platform to purchase items in an existing shop. In order to build such an application complete web support need to be provided. A complete and efficient web application which can provide the online shopping experience is the basic objective of the project. The web application can be implemented in the form of an android application with web view.

PROJECT OVER VIEW:-

The central concept of the application is to allow the customer to shop virtually using the Internet and allow customers to buy the items and articles of their desire from the store. The information pertaining to the products are stored on an RDBMS at the server side (store). The Server processes the customers and the items are shipped to the address submitted by them. The application was designed into two modules first is for the customers who wish to buy the articles. Second is for the storekeepers who maintain and update the information pertaining to the articles and those of the customers' end user of this product is a departmental store where the application is hosted on the web and the administrator maintains the database. The application which is deployed at the customer database, the details of the items are brought forward from the database for the customer view based on the selection through the menu

and the database of all the products are updated at the end of each transaction. Data entry into the application can be done through various screens designed for various levels of users. Once the authorized personnel feed the relevant data into the system, several reports could be generated as per the security.

EXISTING SYSTEM

The current system for shopping is to visit the shop manually and from the available

product choose the item customer want and buying the item by payment of the price of

the item.

1. User must go to shop and select products.
2. It is difficult to identify the required product.
3. Description of the product limited.
4. It is a time-consuming process
5. Not in reach of distant users.

PROPOSED SYSTEM

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

orders. The system also recommends a home delivery system for the purchased products.

PROJECT SCOPE:-

This system can be implemented to any shop in the locality or to multinational branded shops having retail outlet chains. The system recommends a facility to accept the orders 4*7 and a home delivery system which can make customers happy. If shops are providing an online portal where their customers can enjoy easy shopping from anywhere, the shops won't be losing any more customers to the trending online shops such as flipkart or eBay. Since the application is available in the Smartphone it is easily accessible and always available.

Our designed online shopping system provides a 24×7 service, that is customers can surf the website, place orders anytime they wish to. Also, the delivery system works 24×7 hours a week. Some of the features that can be modified and added to this system in the future involve its implementation by local shopkeepers, where shops will be providing an online interface to customers for shopping and placing orders. Then some delivery persons can perform their work. This will be adding on benefit for the customers as it will save their time, plus it adds on for the shopkeepers also, as people will continue to shop from local shops rather than preferring to supermarkets every time. Also, since the deliveries from these local

vendors will not be as time-consuming as these days Flipkart, Amazon, etc. take but rather will be delivered the same day of an order placed. Else the shopkeeper can ask the customer that the product will be available by the next day, so if he/she still wants to place the order, it can be done. Again, return or exchange will be easy since the delivery boy can even do it as the store is nearby. Including a chat box for public benefit is also a great idea via which

people can directly have a conversation with some officials regarding any type of queries.

2. LITERATURE SURVEY

BACKGROUND:-

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

The History of Online Shopping:-

For many, it can be hard to remember the days when online shopping wasn't an option. Yet, despite its prevalence now, online shopping is a relatively new phenomenon. This graphic, presented by Logiq, outlines the brief history of online shopping and how it has evolved over the last few decades. We'll also touch on how companies can get ahead in this rapidly evolving space and what it takes to remain competitive in today's market. According to BigCommerce, the first inklings of online shopping began in England, back in the late 1970s.

1970s: The Early Days

In 1979, the English inventor Michael Aldrich invented a system that allowed consumers to connect with businesses electronically. He did this by connecting a consumer's TV to a retailer's computer via a telephone line. His invention was one of the first communication tools that

allowed for interactive, mass communication—but it was costly, and it didn't make sense financially for most businesses until the Internet became more widespread. By 1982, the world's first e-commerce company launched. The Boston Computer Exchange (BCE) was an online marketplace for people to

buy and sell used computers. The launch of BCE predates the advent of the World Wide Web, and because of this, the company operated on a dial up bulletin board system. By the mid-90s, the Internet had become an established hub for global connection. In 1995, the most popular web browser at the time, Netscape, had around 10 million users worldwide. That same year, Jeff Bezos launched Amazon, which at the time functioned as an online book marketplace. The company saw early signs of success—within 30 days of launching, it was shipping internationally to 45 different countries. A few years later, an online payment system called Confinity—now known as PayPal—was born. Amidst the global pandemic, businesses were forced to close their brick-and-mortar stores, and lockdown restrictions drove consumers online. By May 2020, e-commerce sales had reached \$82.5 billion, a 77% rise year-over-year.

2.1 Programming Languages :-

In this project, Python (Django) was chosen as a general purpose programming language, Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. and SQL lite was selected as backend database. Python(Django), JavaScript ,Ajax,Jquery were used for the client-side work.

2.1.1 Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

Django

★ What Is Django (Python)

Django is a popular python-based framework, and it is used for web development. It is a high-level web framework that allows building secure and maintainable websites quickly.

Django was created between 2003 and 2005 by Adrian Holovaty and Simon Willison at Lawrence Journal-World newspaper. It was started as an internal project at Lawrence Journal-World newspaper. Sometimes the web development team at Lawrence Journal-World newspaper had to develop new features or even complete applications in a short period of time. As a result, Django was created to meet the tight deadlines of news websites while still keeping the development process neat and maintainable. By 2005, Django had grown enough to manage lots of high sites, so the developers decided to make it an open-source project. So, Django was released under the BSD license in 2005. And the project was named after a famous jazz guitarist, Django Reinhardt.

To maintain Django, a foundation called the Django Software Foundation (DSF) was founded in 2008

2.1.2 Java Script

JavaScript is dynamic, high-level scripting language and considered to be one of the core three technologies of the world wide web. It is considered an important part of a web application. It is used for adding functionalities and making web pages interactive. In simple words, it informs

the browser about a certain activity or event that occurred and changes the web page as a response to that event, for example, a click on a button.

JavaScript is the scripting language of the Web. All modern HTML pages are using JavaScript. A scripting language is a lightweight programming language. JavaScript code can be inserted into any HTML page, and it can be executed by all types of web browsers. JavaScript is easy to learn.

★ WHY TO USE JAVASCRIPT

JavaScript is one of the 3 languages all web developers must learn:

- HTML to define the content of web pages
- CSS to specify the layout of web pages
- JavaScript to specify the behavior of web pages

Example

```
x = document.getElementById("demo"); //Find the HTML element with  
id="demo" x.innerHTML = "Hello JavaScript"; //Change the content of the  
HTML element
```

document.getElementById() is one of the most commonly used HTML DOM methods.

OTHER USES OF JAVASCRIPT

- Delete HTML elements
- Create new HTML elements
- Copy HTML elements
- In HTML, JavaScript is a sequence of statements that can be executed by the web browser.

JAVASCRIPT STATEMENTS

- JavaScript statements are "commands" to the browser.

- The purpose of the statements is to tell the browser what to do.
- This JavaScript statement tells the browser to write "Hello Dolly" inside an HTML element with id="demo":
- Semicolon separates JavaScript statements.
- Normally you add a semicolon at the end of each executable statement.
- Using semicolons also makes it possible to write many statement on one line

JAVASCRIPT CODE

- JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
- Each statement is executed by the browser in the sequence they are written.
- This example will manipulate two HTML elements:
- Example
- `document.getElementById("demo").innerHTML="Hello Dolly";`
`document.getElementById("myDIV").innerHTML="How are you?";`

JAVASCRIPT PROPERTIES:

- Properties are the values associated with a JavaScript object.
- A JavaScript object is a collection of unordered properties.
- Properties can usually be changed, added, and deleted, but some are read only.

2.1.3 HTML

HTML or Hypertext Markup Language is the standard [markup](#) language used to create [web](#) pages.

HTML is written in the form of [HTML](#) elements consisting of *tags* enclosed in [angle brackets](#) (like `<html>`). HTML tags most commonly

come in pairs like `<h1>` and `</h1>`, although some tags represent *empty elements* and so are unpaired, for example ``. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*). Though not always necessary, it is best practice to append a slash to tags which are not paired with a closing tag.

The purpose of a [webbrowser](#) is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website [semantically](#) along with cues for presentation, making it a [markup](#) language rather than a [programming](#) language.

HTML elements form the building blocks of all [websites](#). HTML allows (images and objects) to be embedded and can be used to create interactive forms. It provides a means to create structured documents by

denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

2.1.4 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

It is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style webpages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation..

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, <http://en.wikipedia.org/wiki/Color> colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one applied.

2.1.5 jQuery

jQuery is a fast and concise JavaScript Library created by John Resign in 2006

with a nice motto: **Write less, do more.** jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is

a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery –

DOM manipulation – The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross- browser open source selector engine called Sizzle.

Event handling – The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.

AJAX Support – The jQuery helps you a lot to develop a responsive and feature rich site using AJAX technology.

Animations – The jQuery comes with plenty of built-in animation Effects which you can use in your websites.

Lightweight – The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).

Cross Browser Support – The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+

Latest Technology – The jQuery supports CSS3 selectors and basic XPath syntax.

2.1.6 AJAX

AJAX stands for **A**synchronous **J**avaScript and **X**ML. AJAX is a new

technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.

- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger
- Data-driven as opposed to page-driven.

Rich Internet Application Technology.
- AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.
- AJAX is Based on Open Standards

- AJAX is based on the following open standards –

Browser-based presentation using HTML and Cascading Style Sheets (CSS).

3. Backend Technology:-

For this project work, MySQL was chosen as a database.

3.1 SQLite:-

★ What is SQLite?

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

★ Why SQLite?

SQLite does not require a separate server process or system to operate (serverless).

SQLite comes with zero-configuration, which means no setup or administration needed.

A complete SQLite database is stored in a single cross-platform disk file

SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.

SQLite is self-contained, which means no external dependencies.

SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.

SQLite supports most of the query language features found in SQL92 (SQL2) standard.

SQLite is written in ANSI-C and provides simple and easy-to-use API.

SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

3.1.1 Advantages of PDO:

o Database

The PDO extension can access any database which is written for PDO driver. There are several PDO drivers available which are used for FreeTDS, Microsoft SQL Server, Sybase, IBM DB2, Oracle Call Interface, Firebird/Interbase 6, and PostgreSQL databases, among many more.

The drivers are not available in every system automatically, so we have to find our available drivers and add ones when we need them.

o Databaseconnecting

There are different syntaxes available to establish the database connection. These syntaxes depend on specific databases. While using PDO, operations must be wrapped in try/catch blocks and utilize the exception technique.

Usually, only a single connection needs to create, and these connections are closed by programming the database to set as a null.

o Error handling

PDO permits to use exceptions for error handling. To produce an exception, PDO can be forced into a relevant error mode attribute.

There are three error modes, i.e., Silent (default), Warning, and Exception. Warning and Exception are more useful in DRY programming.

- a. Silent – It is a default error mode.
- b. Warning – It is useful for debugging.
- c. Exception – This mode allows graceful error handling while hiding data that a person might use to exploit your system.

Insert and Update

PDO reduces the commonly used insert and update database operation into a two-step process, i.e.

Prepare >> [Bind] >> Execute.

Through this method, we can take full advantage of PDO's prepared statements, which protect against malicious attacks through SQL injection.

Prepared statements are pre-compiled SQL statements that can be executed multiple times by sending this data to the server. This data, which is used within the placeholder, is automatically protected from the SQL injection attack.

4 Software Tools

For this project work following source tools were chosen to perform various tasks:

VS Code

- ★ What is VS Code used for?

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

5. SYSTEM DEVELOPMENT

5.1 Class Diagram

The class diagram is chosen to explain the design phase of the system. A class diagram describes classes of the system, attributes, and operations and relationships of the classes in a better way. We can also say that class diagrams are used to justify the structure or behavior of use cases of the system. Class diagrams best explain the conceptual model of the system in terms of entities and their relationships. The class diagram looks like a shape of a rectangle, comprising three compartments stacked vertically. The first top box comprises the class name, the second middle box contains the attributes of the class and third the last box contains the methods or functions performed by that class. The first compartment / box of the name is compulsory while rest of the two can be omitted to simplify the diagram. So, in any class diagram first compartment must be drawn while the second two compartments are optional.

The class "patient" contains multiple parameters (such as id, name, age, address), which depict the information of all the registered patients. The user class also contains the methods performed by these users such as get appointment, view/ create own medical record etc. In the same way, the class "doctor" has the parameters id, name, department, address possessing all the required information of the users registered as a doctor on to the system. Methods include accept/reject the appointment, check the patient, view a medical record of any patient etc. These methods are the functions performed by the users registered as a doctor on the system. The class "appointment"

has the parameters of date and time, explaining what time or day patient user has requested.

ER Diagram-

5.2 DATA FLOW DIAGRAM (DFD) :-

Generally, DFD's are used as a design notation to represent architectural

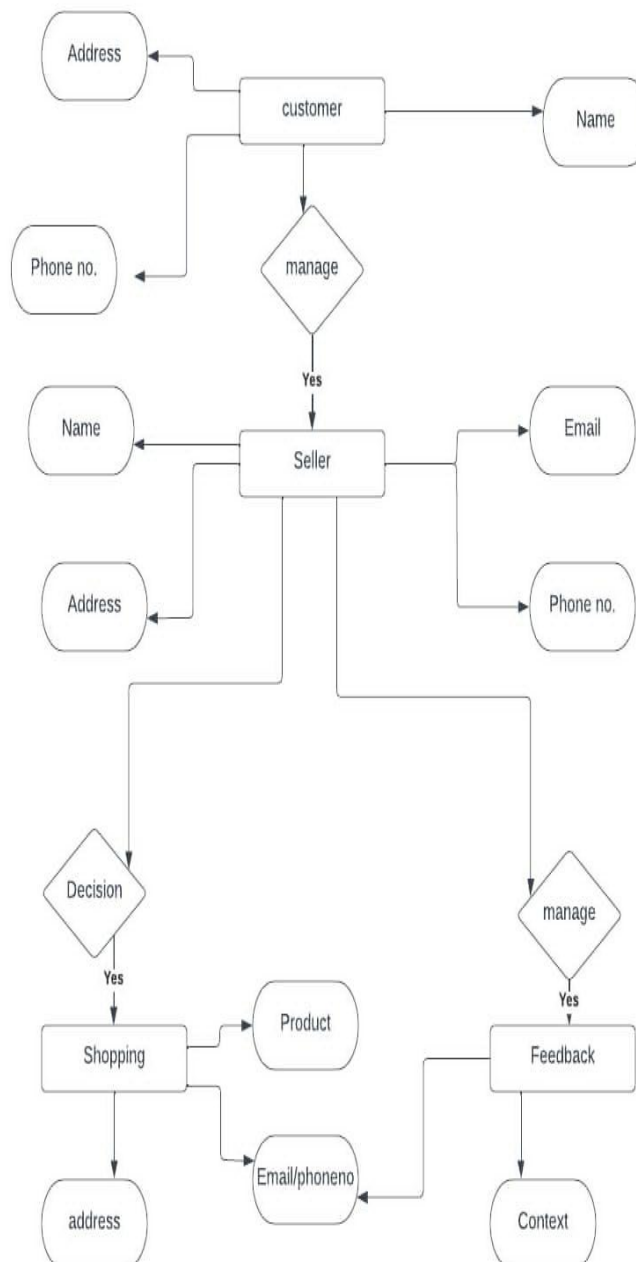
Design

(External
design) and

top level

design

(internal



design) specifications.

DFD's represent the system in hierarchical manner with one top level and many

Lower level diagrams with each representing separate parts of the system. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel. Since diagrammatic representations are easier to interpret as compared to the technical descriptions, the non-technical users can also understand the system details clearly. DFD consists of four basic notations which help to depict the information in the system. These notations are rectangle, circle, open-ended rectangle, and arrows.

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data

flows to or from an external agent, and which together provide all of the functionality of the system as a whole.

Represents an external entity that is the source

Rectangle

or destination of data within the system. Each external entity is represented by a meaningful and unique name.

Represent processes that show transformation or manipulation of data within the system.

Represent data stores that indicate the place for storing information within the system.

They are used to represent data flows that show the movement of data from its source to destination within the system.

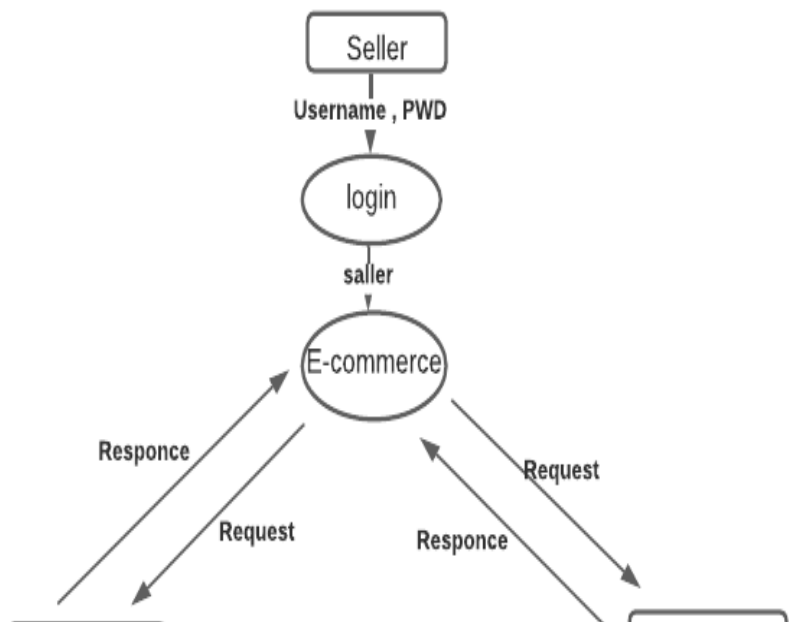
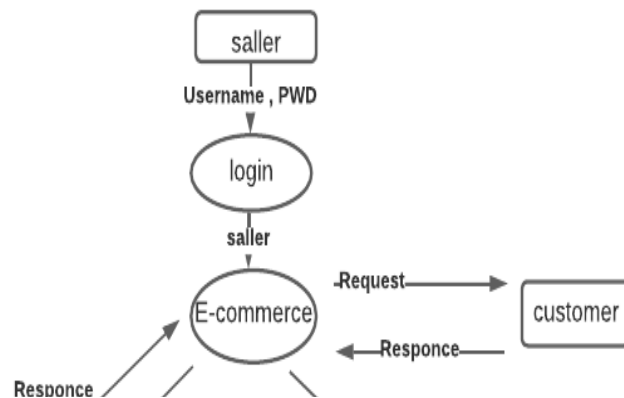
~~They are used to represent data flows that show the movement~~
of data from its source to destination within the system.

Circles

DFD of Context Level –

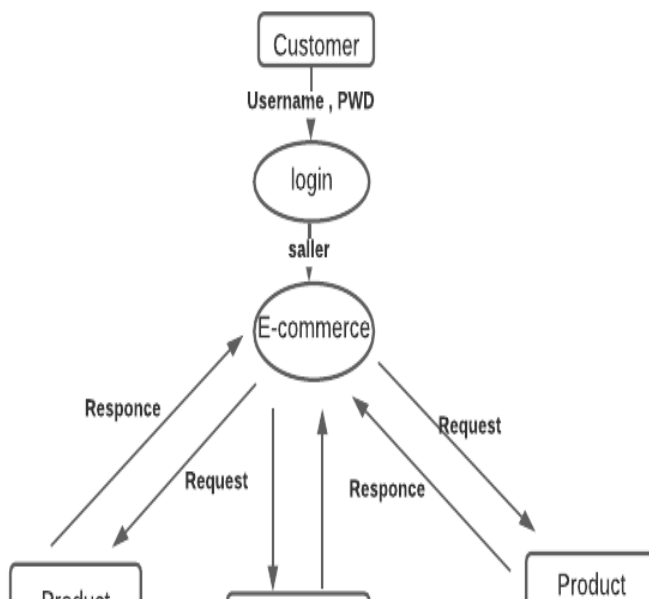
DFD of first level for SELLER

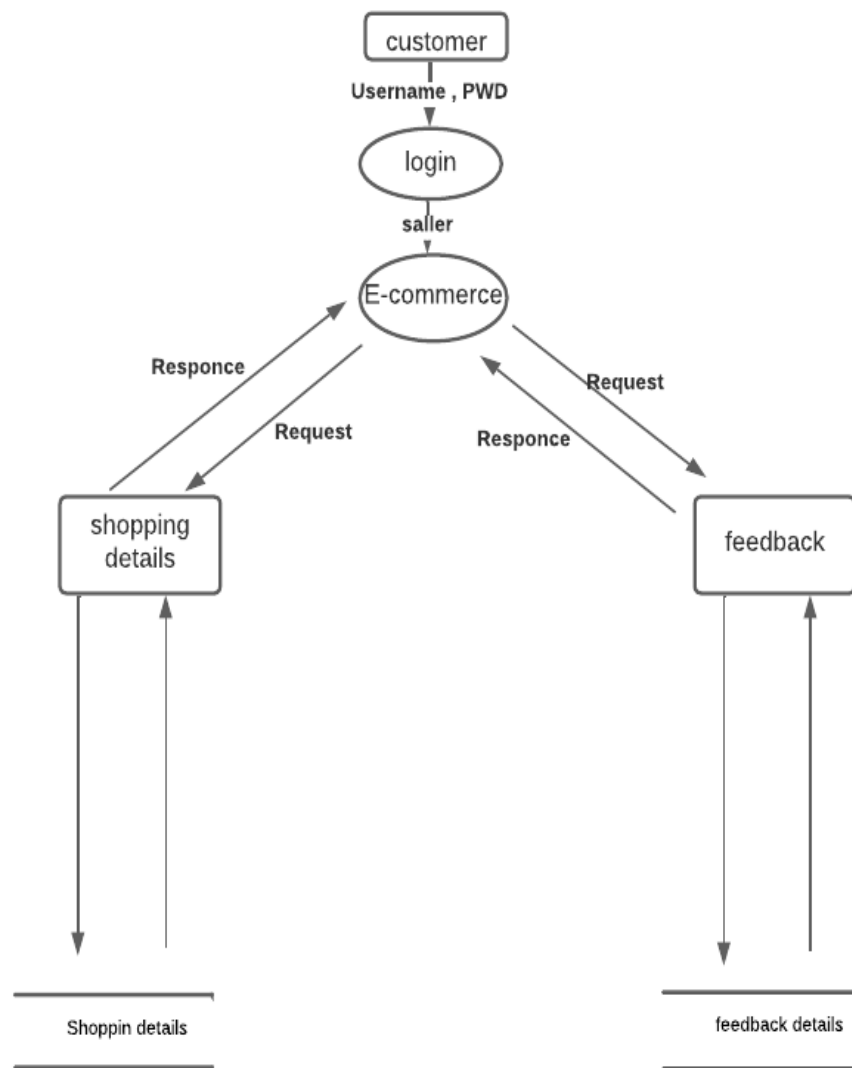
.DFD of second level for SELLER



.DFD of First level for Customer

DFD of Second level for Customer





6.SYSTEM REQUIREMENTS

SOFTWARE AND HARDWARE REQUIREMENT

The software requirement specification (SRS) and hardware specification forms the basis of software development. A main purpose of software requirement specification is the clear definition and specification of functionality and of the software product. It allows the developer to be carried out, performance level to be obtained and corresponding interface to be established.

HARDWARE REQUIREMENTS

Processor: 233 MHz processor

Monitor size: 14 inch

RAM: 128MB SD-RAM

Hard Disk: 2-4 GB Hard-Disk

SOFTWARE REQUIREMENTS

Technology : Web Application.

Front end: HTML, jQuery, Bootstrap & CSS, Ajax, Javascript

Back end : Python (Django)

server: Django Server

Design Tool : Vs Code

Web browser : Mozilla Firefox, Google Chrome, Apple Safari or any Gecko/

Web Kit based browser.

7.CODE

1) 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',
'Project.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()
```

2) checksum.py

```
# pip install pycryptodome
import base64
import string
import random
import hashlib
```

```
from Crypto.Cipher import AES
```

```
IV = "@@@@&&&&####$$$$"
BLOCK_SIZE = 16
```

```
def generate_checksum(param_dict, merchant_key, salt=None):
    params_string = __get_param_string__(param_dict)
    salt = salt if salt else __id_generator__(4)
    final_string = '%s|%s' % (params_string, salt)
```

```
    hasher = hashlib.sha256(final_string.encode())
    hash_string = hasher.hexdigest()
```

```
    hash_string += salt
```

```
    return __encode__(hash_string, IV, merchant_key)
```

```
def generate_refund_checksum(param_dict, merchant_key, salt=None):
    for i in param_dict:
        if("|" in param_dict[i]):
            param_dict = {}
            exit()
    params_string = __get_param_string__(param_dict)
```

```

salt = salt if salt else __id_generator__(4)
final_string = '%s|%s' % (params_string, salt)

hasher = hashlib.sha256(final_string.encode())
hash_string = hasher.hexdigest()

hash_string += salt

return __encode__(hash_string, IV, merchant_key)

```

```

def generate_checksum_by_str(param_str, merchant_key, salt=None):
    params_string = param_str
    salt = salt if salt else __id_generator__(4)
    final_string = '%s|%s' % (params_string, salt)

    hasher = hashlib.sha256(final_string.encode())
    hash_string = hasher.hexdigest()

    hash_string += salt

    return __encode__(hash_string, IV, merchant_key)

```

```

def verify_checksum(param_dict, merchant_key, checksum):
    # Remove checksum
    if 'CHECKSUMHASH' in param_dict:
        param_dict.pop('CHECKSUMHASH')

    # Get salt
    paytm_hash = __decode__(checksum, IV, merchant_key)
    salt = paytm_hash[-4:]
    calculated_checksum = generate_checksum(param_dict,
    merchant_key, salt=salt)
    return calculated_checksum == checksum

```

```

def verify_checksum_by_str(param_str, merchant_key, checksum):
    # Remove checksum
    #if 'CHECKSUMHASH' in param_dict:
        #param_dict.pop('CHECKSUMHASH')

    # Get salt
    paytm_hash = __decode__(checksum, IV, merchant_key)
    salt = paytm_hash[-4:]
    calculated_checksum = generate_checksum_by_str(param_str,
merchant_key, salt=salt)
    return calculated_checksum == checksum


def __id_generator__(size=6, chars=string.ascii_uppercase + string.digits
+ string.ascii_lowercase):
    return ''.join(random.choice(chars) for _ in range(size))


def __get_param_string__(params):
    params_string = []
    for key in sorted(params.keys()):
        if("REFUND" in params[key] or "|" in params[key]):
            respons_dict = {}
            exit()
        value = params[key]
        params_string.append(" if value == 'null' else str(value))
    return '|'.join(params_string)


__pad__ = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) *
chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
__unpad__ = lambda s: s[0:-ord(s[-1])]

```

```

def __encode__(to_encode, iv, key):
    # Pad
    to_encode = __pad__(to_encode)
    # Encrypt
    c = AES.new(key.encode('utf-8'), AES.MODE_CBC, iv.encode('utf-8'))
    to_encode = c.encrypt(to_encode.encode('utf-8'))
    # Encode
    to_encode = base64.b64encode(to_encode)
    return to_encode.decode("UTF-8")

```

```

def __decode__(to_decode, iv, key):
    # Decode
    to_decode = base64.b64decode(to_decode)
    # Decrypt
    c = AES.new(key.encode('utf-8'), AES.MODE_CBC, iv.encode('utf-8'))
    to_decode = c.decrypt(to_decode)
    if type(to_decode) == bytes:
        # convert bytes array to str.
        to_decode = to_decode.decode()
    # remove pad
    return __unpad__(to_decode)

```

```

if __name__ == "__main__":
    params = {
        "MID": "mid",
        "ORDER_ID": "order_id",
        "CUST_ID": "cust_id",
        "TXN_AMOUNT": "1",
        "CHANNEL_ID": "WEB",
        "INDUSTRY_TYPE_ID": "Retail",
        "WEBSITE": "xxxxxxxxxxx"
    }

```

```

print(verify_checksum(
    params, 'xxxxxxxxxxxxxxxx',

"CD5ndX8VVjlzjWbbYoAtKQllvtXPypQYOg0Fi2AUyKXZA5XSHiRF0FDj7
vQu66S8MHx9NaDZ/uYm3WBOWHf+sDQAmTyxqUipA7i1nllxrk="))

# print(generate_checksum(params, "xxxxxxxxxxxxxxxx"))

```

2.1) checksum.py

```

# pip install pycryptodome
import base64
import string
import random
import hashlib

from Crypto.Cipher import AES

IV = "@@@@&&&&####$$$$"
BLOCK_SIZE = 16

def generate_checksum(param_dict, merchant_key, salt=None):
    params_string = __get_param_string__(param_dict)

```

```
salt = salt if salt else __id_generator__(4)
final_string = '%s|%s' % (params_string, salt)
```

```
hasher = hashlib.sha256(final_string.encode())
hash_string = hasher.hexdigest()
```

```
hash_string += salt
```

```
return __encode__(hash_string, IV, merchant_key)
```

```
def generate_refund_checksum(param_dict, merchant_key, salt=None):
```

```
    for i in param_dict:
```

```
        if("|" in param_dict[i]):
```

```
            param_dict = {}
```

```
            exit()
```

```
    params_string = __get_param_string__(param_dict)
```

```
    salt = salt if salt else __id_generator__(4)
```

```
    final_string = '%s|%s' % (params_string, salt)
```

```
    hasher = hashlib.sha256(final_string.encode())
```

```
    hash_string = hasher.hexdigest()
```

```
    hash_string += salt
```

```
    return __encode__(hash_string, IV, merchant_key)
```

```
def generate_checksum_by_str(param_str, merchant_key, salt=None):
```

```
    params_string = param_str
```

```
    salt = salt if salt else __id_generator__(4)
```

```
    final_string = '%s|%s' % (params_string, salt)
```

```
    hasher = hashlib.sha256(final_string.encode())
```

```
    hash_string = hasher.hexdigest()
```



```

hash_string += salt

return __encode__(hash_string, IV, merchant_key)

def verify_checksum(param_dict, merchant_key, checksum):
    # Remove checksum
    if 'CHECKSUMHASH' in param_dict:
        param_dict.pop('CHECKSUMHASH')

    # Get salt
    paytm_hash = __decode__(checksum, IV, merchant_key)
    salt = paytm_hash[-4:]
    calculated_checksum = generate_checksum(param_dict,
merchant_key, salt=salt)
    return calculated_checksum == checksum

def verify_checksum_by_str(param_str, merchant_key, checksum):
    # Remove checksum
    #if 'CHECKSUMHASH' in param_dict:
        #param_dict.pop('CHECKSUMHASH')

    # Get salt
    paytm_hash = __decode__(checksum, IV, merchant_key)
    salt = paytm_hash[-4:]
    calculated_checksum = generate_checksum_by_str(param_str,
merchant_key, salt=salt)
    return calculated_checksum == checksum

def __id_generator__(size=6, chars=string.ascii_uppercase + string.digits
+ string.ascii_lowercase):
    return "".join(random.choice(chars) for _ in range(size))

```

```
def __get_param_string__(params):
    params_string = []
    for key in sorted(params.keys()):
        if("REFUND" in params[key] or "|" in params[key]):
            respons_dict = {}
            exit()
        value = params[key]
        params_string.append(" if value == 'null' else str(value))
    return '|'.join(params_string)
```

```
__pad__ = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) *
chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
__unpad__ = lambda s: s[0:-ord(s[-1])]
```

```
def __encode__(to_encode, iv, key):
    # Pad
    to_encode = __pad__(to_encode)
    # Encrypt
    c = AES.new(key.encode('utf-8'), AES.MODE_CBC, iv.encode('utf-8'))
    to_encode = c.encrypt(to_encode.encode('utf-8'))
    # Encode
    to_encode = base64.b64encode(to_encode)
    return to_encode.decode("UTF-8")
```

```
def __decode__(to_decode, iv, key):
    # Decode
    to_decode = base64.b64decode(to_decode)
    # Decrypt
    c = AES.new(key.encode('utf-8'), AES.MODE_CBC, iv.encode('utf-8'))
    to_decode = c.decrypt(to_decode)
    if type(to_decode) == bytes:
```

```

        # convert bytes array to str.
        to_decode = to_decode.decode()
    # remove pad
    return __unpad__(to_decode)

if __name__ == "__main__":
    params = {
        "MID": "mid",
        "ORDER_ID": "order_id",
        "CUST_ID": "cust_id",
        "TXN_AMOUNT": "1",
        "CHANNEL_ID": "WEB",
        "INDUSTRY_TYPE_ID": "Retail",
        "WEBSITE": "xxxxxxxxxxx"
    }

    print(verify_checksum(
        params, 'xxxxxxxxxxxxxxxxxx',

"CD5ndX8VVjlzjWbbYoAtKQllvtXPypQYOg0Fi2AUYKXZA5XSHiRF0FDj7
vQu66S8MHx9NaDZ/uYm3WBOWHf+sDQAmTyxqUipA7i1nLlXrk="))

    # print(generate_checksum(params, "xxxxxxxxxxxxxxxxxx"))

```

3) asgi.py

```
"""
```

ASGI config for Project project.

It exposes the ASGI callable as a module-level variable named
`application`.

For more information on this file, see
<https://docs.djangoproject.com/en/4.0/howto/deployment/asgi/>
"""

```
import os
```

```
from django.core.asgi import get_asgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Project.settings')
```

```
application = get_asgi_application()
```

4) settings.py

```
"""
```

Django settings for Project project.

Generated by 'django-admin startproject' using Django 2.1.5.

For more information on this file, see
<https://docs.djangoproject.com/en/2.1/topics/settings/>

For the full list of settings and their values, see
<https://docs.djangoproject.com/en/2.1/ref/settings/>

```
"""
```

```
import os
```

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/
```

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

```
SECRET_KEY =
```

```
'9h_#wy*6)%#ug3-uv@7xlryan5a36rqe^j5a$-i0@fo9szu=%n'
```

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

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
    'shop.apps.ShopConfig',  
    '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',
```

```
]
```

```
ROOT_URLCONF = 'Project.urls'
```

```
TEMPLATES = [
```

```
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': ['Project/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',
        ],
    },
},
]
```

```
WSGI_APPLICATION = 'Project.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Password validation
```

```
#
```

```
https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
```

```
{
    'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
    '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/2.1/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/2.1/howto/static-files/>


```
STATIC_URL = '/static/'
```

```
# Managing media
```

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

```
MEDIA_URL = '/media/'
```

5) urls.py

```
"""Project URL Configuration
```

The `urlpatterns` list routes URLs to views. For more information please see:

<https://docs.djangoproject.com/en/4.0/topics/http/urls/>

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `path("", views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `path("", Home.as_view(), name='home')`

Including another `URLconf`

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to `urlpatterns`: `path('blog/', include('blog.urls'))`

```
"""
```

```

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
# from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('shop/', include('shop.urls')),
    path("", include('shop.urls'))
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

6) wsgi.py

```

"""

```

WSGI config for Project project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
<https://docs.djangoproject.com/en/4.0/howto/deployment/wsgi/>

```

"""

```

```

import os

```

```

from django.core.wsgi import get_wsgi_application

```

```

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Project.settings')

```

```
application = get_wsgi_application()
```

7) admin.py

```
from django.contrib import admin  
from models import Product, Contact, Orders, OrderUpdate
```

```
# Register your models here.  
admin.site.register(Product)  
admin.site.register(Contact)  
admin.site.register(Orders)  
admin.site.register(OrderUpdate)
```

8) apps.py

```
from django.apps import AppConfig
```

```
class ShopConfig(AppConfig):
```

```
default_auto_field = 'django.db.models.BigAutoField'  
name = 'shop'
```

9) models.py

```
from django.db import models
```

```
# Create your models here.
```

```
class Product(models.Model):  
    product_id = models.AutoField  
    product_name = models.CharField(max_length=50)  
    category = models.CharField(max_length=50, default="")  
    subcategory = models.CharField(max_length=50, default="")  
    price = models.IntegerField(default=0)  
    desc = models.CharField(max_length=300)  
    pub_date = models.DateField()  
    image = models.ImageField(upload_to="shop/images", default="")  
  
    def __str__(self):  
        return self.product_name
```

```
class Contact(models.Model):  
    msg_id = models.AutoField(primary_key=True)  
    name = models.CharField(max_length=50)  
    email = models.CharField(max_length=70, default="")  
    phone = models.CharField(max_length=70, default="")  
    desc = models.CharField(max_length=500, default="")  
  
    def __str__(self):  
        return self.name
```

```
class Orders(models.Model):  
    order_id = models.AutoField(primary_key=True)  
    items_json = models.CharField(max_length=5000)
```

```
amount = models.IntegerField( default=0)
name = models.CharField(max_length=90)
email = models.CharField(max_length=111)
address = models.CharField(max_length=111)
city = models.CharField(max_length=111)
state = models.CharField(max_length=111)
zip_code = models.CharField(max_length=111)
phone = models.CharField(max_length=111, default="")
```

```
class OrderUpdate(models.Model):
    update_id = models.AutoField(primary_key=True)
    order_id = models.IntegerField(default="")
    update_desc = models.CharField(max_length=5000)
    timestamp = models.DateField(auto_now_add=True)

    def __str__(self):
        return self.update_desc[0:7] + "..."
```

10) urls.py(2)

```
from django.urls import path
from . import views
```

```
urlpatterns = [
    path("", views.index, name="ShopHome"),
    path("about/", views.about, name="AboutUs"),
    path("contact/", views.contact, name="ContactUs"),
    path("tracker/", views.tracker, name="TrackingStatus"),
    path("search/", views.search, name="Search"),
    path("products/<int:myid>", views.productView, name="ProductView"),
    path("checkout/", views.checkout, name="Checkout"),
    path("handlerequest/", views.handlerequest, name="HandleRequest"),
```

]

11) views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Product, Contact, Orders, OrderUpdate
from math import ceil
import json
from django.views.decorators.csrf import csrf_exempt
from PayTm import Checksum
# Create your views here.
from django.http import HttpResponse
# import checksum generation utility

# initialize an Hash/Array

MERCHANT_KEY = 'DQXJWg03974086132067'

# Create your views here.
def index(request):
    # products = Product.objects.all()
    # print(products)
    # params = {'no_of_slides':nSlides,'range':range(1,nSlides),
'product':products} #Dictionary
    # allProds=[[products,range(1,nSlides),nSlides],
    #           [products,range(1,nSlides),nSlides,]]
    allProds = []
    catprods = Product.objects.values('category', 'id')
    cats = {item['category'] for item in catprods}
    for cat in cats:
        prod = Product.objects.filter(category=cat)
        n = len(prod)
        nSlides = n // 4 + ceil((n / 4) - (n // 4))
        allProds.append([prod, range(1, nSlides), nSlides])
    params = {'allProds':allProds}
```

```

    return render(request, 'shop/index.html', params)

def about(request):
    return render(request, 'shop/about.html')

def checkout(request):
    if request.method=="POST":
        items_json = request.POST.get('itemsJson', "")
        name = request.POST.get('name', "")
        amount = request.POST.get('amount', "")
        email = request.POST.get('email', "")
        address = request.POST.get('address1', "") + " " +
request.POST.get('address2', "")
        city = request.POST.get('city', "")
        state = request.POST.get('state', "")
        zip_code = request.POST.get('zip_code', "")
        phone = request.POST.get('phone', "")
        order = Orders(items_json=items_json, name=name, email=email,
address=address, city=city,
                        state=state, zip_code=zip_code, phone=phone,
amount=amount)
        order.save()
        update = OrderUpdate(order_id=order.order_id, update_desc="The
order has been placed")
        update.save()
        thank = True
        id = order.order_id
        # return render(request, 'shop/checkout.html', {'thank':thank, 'id': id})
        # Request paytm to transfer the amount to your account after
payment by user
        "param_dict = {

            'MID': 'DQXJWg03974086132067',
            'ORDER_ID': str(order.order_id),
            'TXN_AMOUNT': str(amount),
            'CUST_ID': email,

```

```

        'INDUSTRY_TYPE_ID': 'Retail',
        'WEBSITE': 'WEBSTAGING',
        'CHANNEL_ID': 'WEB',
        'CALLBACK_URL': 'http://127.0.0.1:8000/shop/handlerequest',

    }
    #MERCHANT_KEY = 'DQXJWg03974086132067'
    return render(request, 'shop/paytm.html')
else:
    return render(request, 'shop/checkout.html')

def search(request):
    return render(request, 'shop/search.html')

def contact(request):
    thank = False
    if request.method=="POST":
        name = request.POST.get('name', "")
        email = request.POST.get('email', "")
        phone = request.POST.get('phone', "")
        desc = request.POST.get('desc', "")
        contact = Contact(name=name, email=email, phone=phone,
desc=desc)
        contact.save()
        thank = True
    return render(request, 'shop/contact.html', {'thank': thank})

def tracker(request):
    if request.method=="POST":
        orderId = request.POST.get('orderId', "")
        email = request.POST.get('email', "")
        try:
            order = Orders.objects.filter(order_id=orderId, email=email)
            if len(order)>0:

```



```

        update = OrderUpdate.objects.filter(order_id=orderId)
        updates = []
        for item in update:
            updates.append({'text': item.update_desc, 'time':
item.timestamp})
        response = json.dumps({"status": "success", "updates": updates,
"itemsJson": order[0].items_json}, default=str)
        return HttpResponse(response)
    else:
        return HttpResponse({'status': "noitem"})
except Exception as e:
    return HttpResponse({'status': "error"})

return render(request, 'shop/tracker.html')

```

```

def productView(request, myid):

```

```

    # Fetch the product using the id
    product = Product.objects.filter(id=myid)
    return render(request, 'shop/prodView.html', {'product': product[0]})

```

```

@csrf_exempt

```

```

def handlerequest(request):
    # paytm will send you post request here

```

```

    form = request.POST
    response_dict = {}
    for i in form.keys():
        response_dict[i] = form[i]
        if i == 'CHECKSUMHASH':
            checksum = form[i]

```

```

    verify = Checksum.verify_checksum(response_dict, MERCHANT_KEY,
checksum)

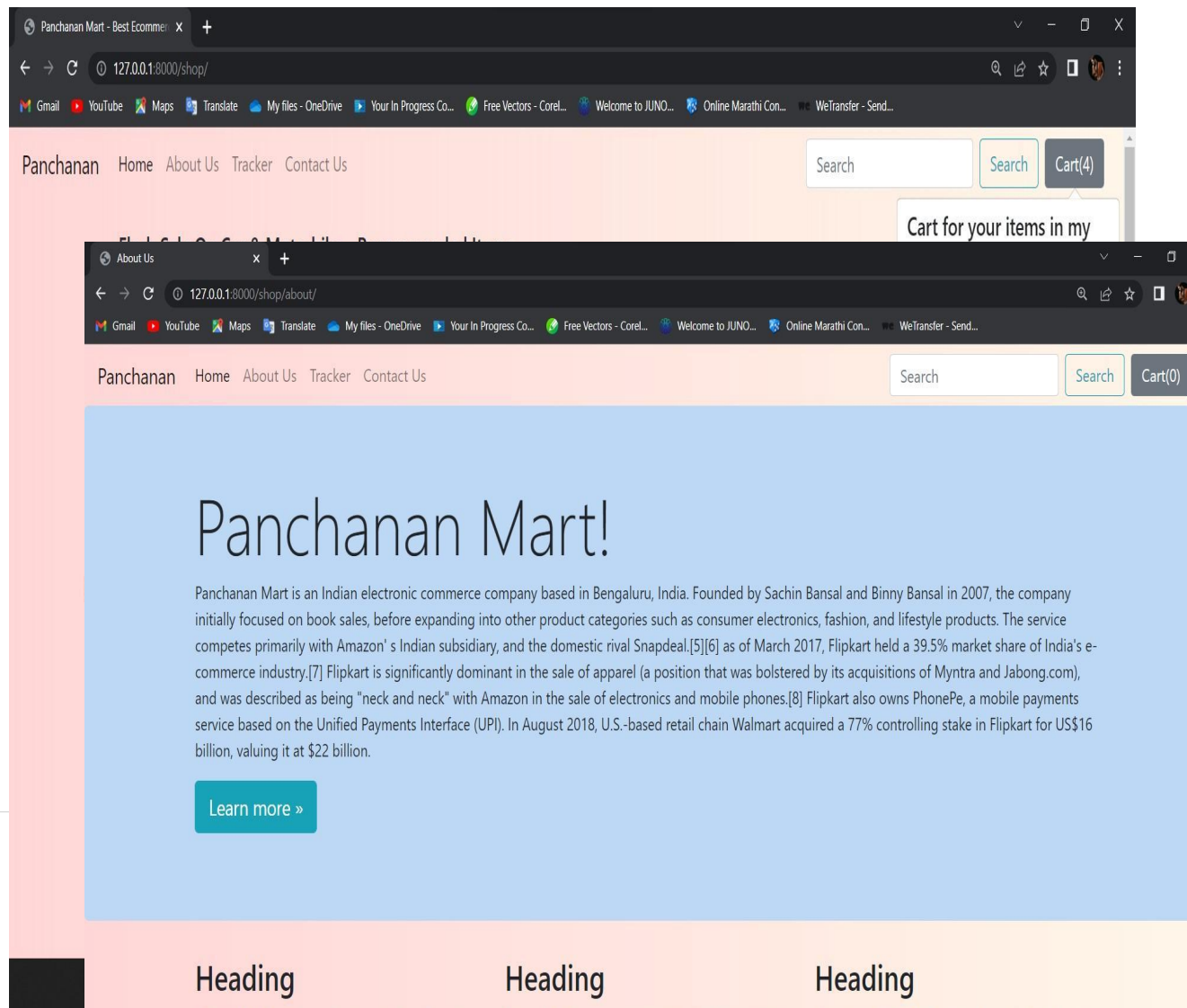
```

```
if verify:
    if response_dict['RESPCODE'] == '01':
        print('order successful')
    else:
        print('order was not successful because' +
response_dict['RESPMSG'])
    return render(request, 'shop/paymentstatus.html', {'response':
response_dict})
```

8.ScreenShot

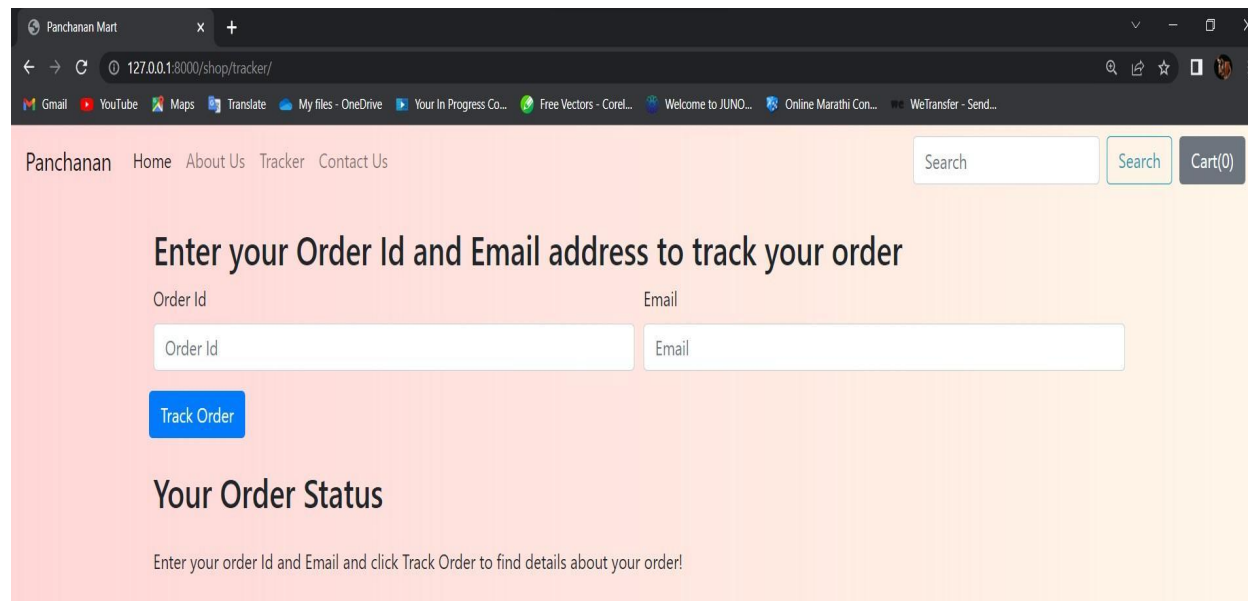
1) Home Index

2) About Us



3) Tracker

4) Contact Us



A screenshot of a web browser showing the 'Tracker' page of 'Panchanan Mart'. The browser's address bar shows the URL '127.0.0.1:8000/shop/tracker/'. The page has a navigation bar with links to 'Panchanan', 'Home', 'About Us', 'Tracker', and 'Contact Us'. There are search and cart buttons in the top right. The main heading is 'Enter your Order Id and Email address to track your order'. Below this are two input fields: 'Order Id' and 'Email'. A blue 'Track Order' button is positioned below the 'Order Id' field. The section is titled 'Your Order Status' with a subtext: 'Enter your order Id and Email and click Track Order to find details about your order!'.

Panchanan Mart

127.0.0.1:8000/shop/tracker/

Panchanan Home About Us Tracker Contact Us

Search Search Cart(0)

Enter your Order Id and Email address to track your order

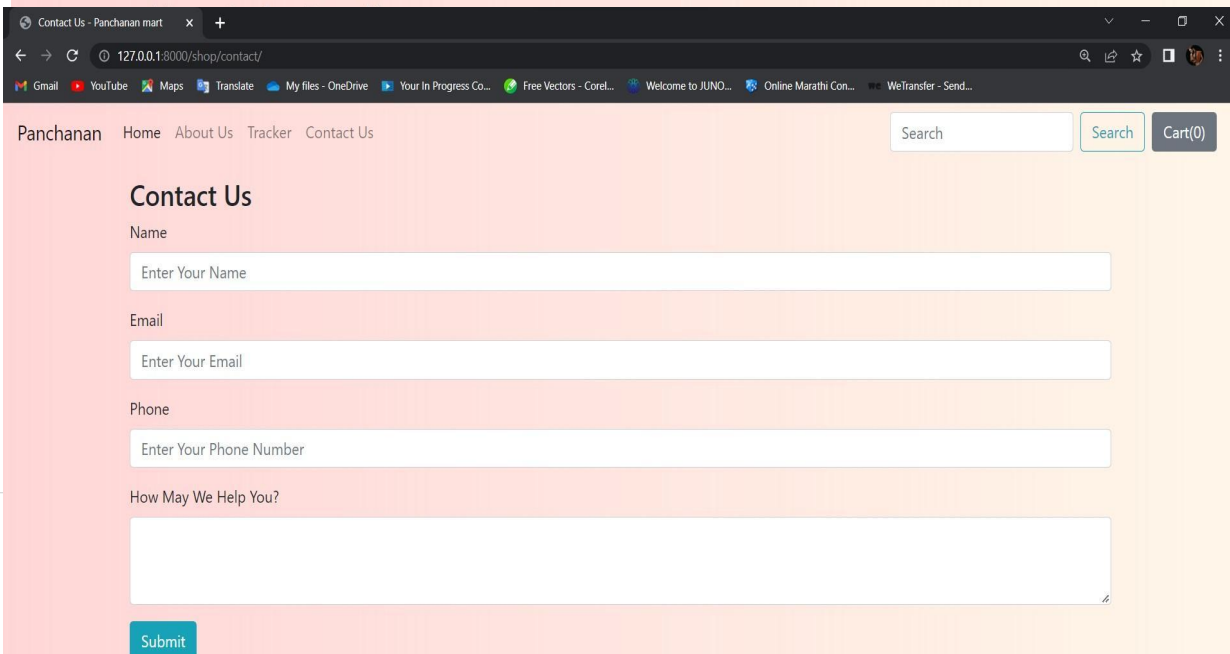
Order Id Email

Order Id Email

Track Order

Your Order Status

Enter your order Id and Email and click Track Order to find details about your order!



A screenshot of a web browser showing the 'Contact Us' page of 'Panchanan Mart'. The browser's address bar shows the URL '127.0.0.1:8000/shop/contact/'. The page has a navigation bar with links to 'Panchanan', 'Home', 'About Us', 'Tracker', and 'Contact Us'. There are search and cart buttons in the top right. The main heading is 'Contact Us'. Below this are four input fields: 'Name' (with placeholder 'Enter Your Name'), 'Email' (with placeholder 'Enter Your Email'), 'Phone' (with placeholder 'Enter Your Phone Number'), and a text area for 'How May We Help You?'. A blue 'Submit' button is at the bottom.

Contact Us - Panchanan mart

127.0.0.1:8000/shop/contact/

Panchanan Home About Us Tracker Contact Us

Search Search Cart(0)

Contact Us

Name

Enter Your Name

Email

Enter Your Email

Phone

Enter Your Phone Number

How May We Help You?

Submit

5) Check Out Page

6) Add Product

Checkout - Panchanan Mart x +

127.0.0.1:8000/shop/checkout/

Gmail YouTube Maps Translate My files - OneDrive Your In Progress Co... Free Vectors - Corel... Welcome to JUNO... Online Marathi Con... WeTransfer - Send...

Panchanan Home About Us Tracker Contact Us

Search Search Cart(4)

Step 1 - Panchanan Mart Checkout - Review Your Cart Items

Vega Crux Black Helm... 4

Your Cart Total Is **Rs. 4624**. Enter your details below & place your order. Thanks for using My Awesome Cart!

Step 2 - Enter Address & Other Details:

Name Email

Name Email

Address

1234 Main St

Address line 2

Apartment, studio, or floor

City State Zip

Enter State

Select product to change | Django x +

127.0.0.1:8000/admin/shop/product/

Gmail YouTube Maps Translate My files - OneDrive Your In Progress Co... Free Vectors - Corel... Welcome to JUNO... Online Marathi Con... WeTransfer - Send...

Django administration

WELCOME, PANCHANAN VIEW SITE / CHANGE

Home Shop Products

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

SHOP

Select product to change

Action: Go 0 of 5 selected

PRODUCT

Inkast Denim Co. Men's Slim Casual Shirt

Amazon Brand - Inkast Denim Co. Men's Slim Casual

7) Add other Users

8) Admin

Select user to change | Django admin

127.0.0.1:8000/admin/auth/user/

WELCOME, **PANCHANAN** [VIEW SITE](#) / [CHANGE PASSWORD](#)

Home > Authentication and Authorization > Users

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

SHOP

- Contacts + Add
- Order updates + Add
- Orderss + Add
- Products + Add

Select user to change

Search

Action: [dropdown] Go 0 of 1 selected

| USERNAME | EMAIL ADDRESS | FIRST NAME | LAST NAME | STAFF STATUS |
|-----------|------------------------|------------|-----------|--------------|
| panchanan | rushabhilode@gmail.com | | | Yes |

1 user

FILTER

- By staff status
 - All
 - Yes
 - No
- By superuser status
 - All
 - Yes
 - No
- By active
 - All
 - Yes
 - No

Dashboard

9) Add to Card

The image shows two overlapping screenshots. The top screenshot is of the Django administration interface for a site named 'Panchanan Mart'. The browser address bar shows '127.0.0.1:8000/admin/'. The interface has a dark theme. The main content area is titled 'Site administration' and is divided into two sections: 'AUTHENTICATION AND AUTHORIZATION' and 'SHOP'. The 'AUTHENTICATION AND AUTHORIZATION' section contains two rows: 'Groups' and 'Users', each with '+ Add' and 'Change' links. The 'SHOP' section contains four rows: 'Contacts', 'Order updates', 'Orderss', and 'Products', each with '+ Add' and 'Change' links. On the right side, there is a 'Recent actions' panel titled 'My actions' which lists several actions, each preceded by a red 'X' icon. The actions are: 'The ord...' (Order update), 'The ord...' (Order update), 'shipped...' (Order update), 'Orders object (56)' (Orders), 'Orders object (57)' (Orders), 'Orders object (58)' (Orders), 'Orders object (59)' (Orders), 'Orders object (60)' (Orders), 'Orders object (61)' (Orders), and 'Orders object (62)' (Orders). The bottom screenshot is of the 'Panchanan Mart' web storefront. The browser address bar shows '127.0.0.1:8000'. The page has a light pink background. At the top, there is a navigation bar with links: 'Panchanan', 'Home', 'About Us', 'Tracker', and 'Contact Us'. To the right of the navigation bar is a search bar with a 'Search' button and a 'Cart(0)' button. Below the navigation bar, there is a section titled 'Flash Sale On Car & Motorbike - Recommended Items'. This section contains two product cards. The first card shows a black motorcycle helmet. The second card shows a green spray bottle of 'Motto' car wash.

Site administration | Django site x +

127.0.0.1:8000/admin/

Gmail YouTube Maps Translate My files - OneDrive Your In Progress Co... Free Vectors - Corel... Welcome to JUNO... Online Marathi Con... WeTransfer - Send...

Django administration

WELCOME, PANCHANAN [VIEW SITE](#) / [CHANGE](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

| | | |
|--------|-------|--------|
| Groups | + Add | Change |
| Users | + Add | Change |

SHOP

| | | |
|---------------|-------|--------|
| Contacts | + Add | Change |
| Order updates | + Add | Change |
| Orderss | + Add | Change |
| Products | + Add | Change |

Recent actions

My actions

- ✗ The ord...
Order update
- ✗ The ord...
Order update
- ✗ shipped...
Order update
- ✗ Orders object (56)
Orders
- ✗ Orders object (57)
Orders
- ✗ Orders object (58)
Orders
- ✗ Orders object (59)
Orders
- ✗ Orders object (60)
Orders
- ✗ Orders object (61)
Orders
- ✗ Orders object (62)
Orders

Panchanan Mart - Best Ecommerce x +



127.0.0.1:8000

Gmail YouTube Maps Translate My files - OneDrive Your In Progress Co... Free Vectors - Corel... Welcome to JUNO... Online Marathi Con... WeTransfer - Send...

Panchanan Home About Us Tracker Contact Us

Search Search Cart(0)

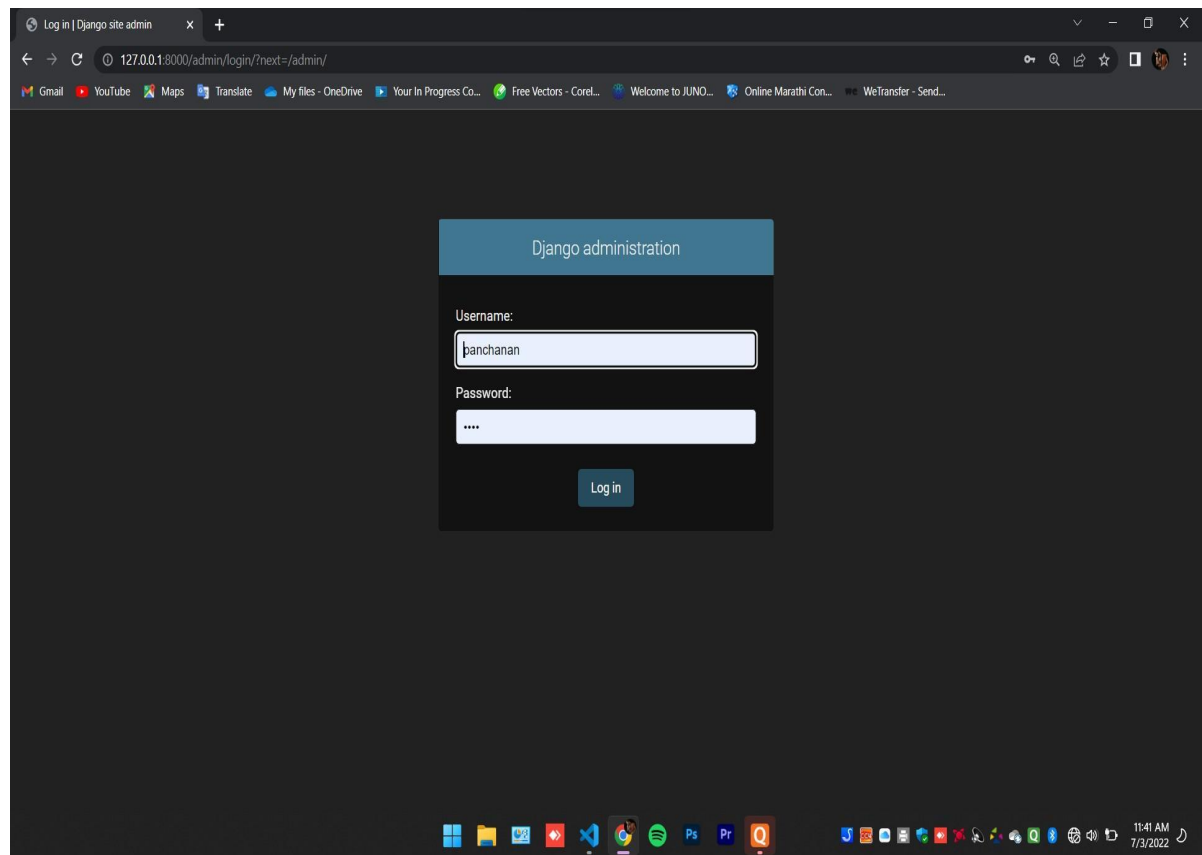
Flash Sale On Car & Motorbike - Recommended Items

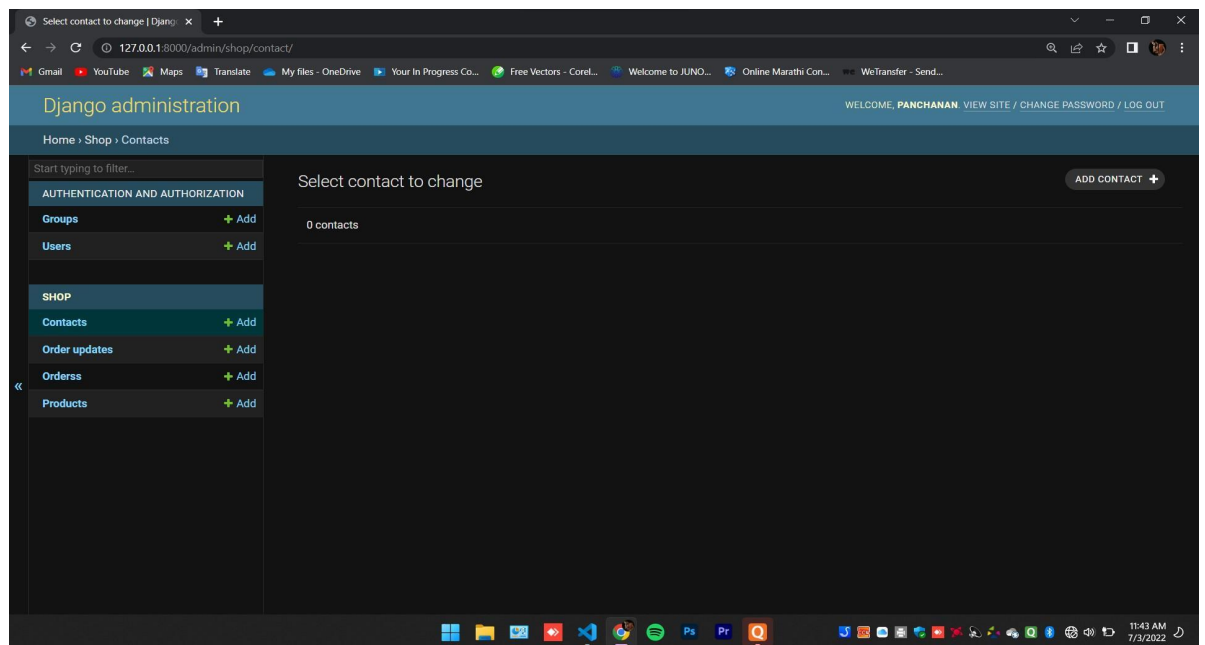


10) Admin Panel

11) Contact

Information





12) Terminal Command for Run

13) See Order

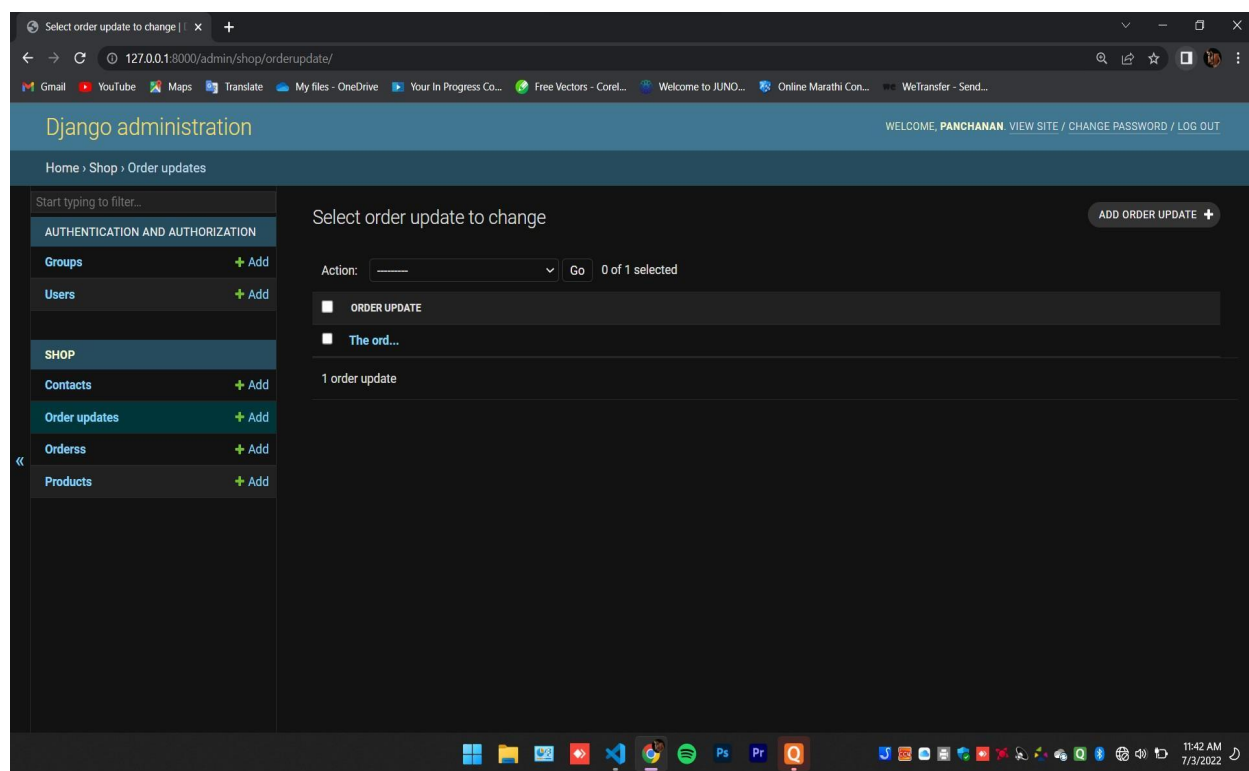
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS F:\MyShop> cd .\Project\
PS F:\MyShop\Project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 03, 2022 - 11:35:54
Django version 4.0.5, using settings 'Project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
█
```

14) Update Order Status



9.Conclusion

The project entitled Online shopping website was completed successfully.

The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a website for purchasing items from a shop.

This project helped us in gaining valuable information and practical knowledge on several topics like designing web pages using html & css, usage of responsive templates, designing of

php databases and management of database using mysql. The entire system is secured. Also the project helped us understanding about the development phases of a project and software development life cycle. We learned how to test different features of a project. This project has given us great satisfaction in having designed an application which can be implemented to any nearby shops or branded shops selling various kinds of products by simple modifications. There is a scope for further development in our project to a great extent. A number of features can be added to this system in future like providing moderator more control over products so that each moderator can maintain their own products.

Another feature we wished to implement was providing classes for customers so that different offers can be given to each class. System may keep track of history of purchases of each customer and provide suggestions based on their history. These features could have been implemented unless the time did not limit us.

10. Web References

- 1). www.javatpoint.com
- 2). www.w3school.com
- 3). www.tutorialspoint.com
- 4). www.youtube.com
- 5). www.google.com