STEAL THE DEAL

|  |  |
| --- | --- |
| **NAME** | **REGISTRATION-NO** |
| Nida Jawaid | 200968194 |
| Pravallika Srinivas Gajul | 200968182 |
| Sandhi Jain | 209309030 |
| Aditi Sangram Desai | 200968126 |
| Ishtha Singh | 200968200 |



Department of Data Science

Manipal Institute of Technology

2020-2024

CONTENTS

1. Certificate……………………………………………………………………………………1

2. Acknowledgement………………………………………………………………………..2

3. Abstract………………………………………………………………………………………3

4. Introduction…………………………………………………………………………………4

5. Motivation……………………………………………………………………………………5

7. Methodology…………………………………………………………………………………5

8. Role of Members…………………………………………………………………………..9

9. Modules……………………………………………………………………………………….11

10. Software Requirements………………………………………………………………14

11. MySQL Table……………………………………………………………………………..18

12. Schema Diagram………………………………………………………………………..19

13. E-R Diagram………………………………………………………………………………20

14. Database…………………………………………………………………………………..21

15. Frontend……………………………………………………………………………………22

17. Scope for Future………………………………………………………………………..24

18. Conclusion………………………………………………………………………………..25

19. References………………………………………………………………………………..25

CERTIFICATE

This is to certify that **Nida Jawaid**(200968194), **Pravallika Srinivas Gajul**(200968182), **Sandhi Jain**(209309030), **Aditi Sangram Desai**(200968126), **Ishtha Singh**(200968200) has completed the project **Steal The Deal** under the guidance of **Tojo Thomas** and **Akshay Bhat** during the academic year 2022 as per the guidelines given by Data Science Department of Manipal Institute of Technology.

­ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**Tojo Thomas Akshay Bhatt**

Assistant Professor-Senior Scale Assistant Professor-Senior Scale

DSCA,MIT DSCA,MIT

ACKNOWLEDGEMENT

We appreciate the trust laid in us by Prof. Tojo Thomas and Prof. Akshay Bhatt and providing us with an opportunity to undergo this Project as a part of our curriculum.

We are thankful for their support, cooperation and motivation provided during the training for constant inspiration.

We would also like to thank our college for providing us all the necessary resources for the project.

Last but not the least, we would like to thank everyone who is involved in this project for their cooperation and hard work.

ABSTRACT

E-commerce is a process of doing business through computer networks. Unlike traditional commerce that is carried out physically with effort of a person to go and get products, e-commerce has made it easier for humans to reduce physical work and save time.

Our project Steal The Deal is an e-commerce website where we tried to mimic other e-commerce websites like Amazon, Flipkart etc. This is to make the online shopping experience smooth, less time consuming and more productive.

INTRODUCTION

We live in a country where technology plays an important role for development. By e-commerce we mean buying and selling of products or services over electronic systems such as the Internet and other computer networks.

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming common place.

**AIM OF THE PROJECT**

The aim of this project is to develop a general purpose e-commerce store where products can be bought from the comfort of home through the Internet. Thus, it was required to create a site which handles online payments.

This report includes theoretical information about e-commerce, content management systems and it describes the development process of the site.

MOTIVATION

As college students, it becomes difficult to manage our lives in our hectic schedule. We have to be productive while still making time for ourselves. In such a scenario online shopping plays a vital role in making our lives a lot more easier.

METH0DOLOGY

We have made the project using the agile methodology.

What is the Agile methodology?

Agile methodology is a practice that helps continuous iteration of development and testing in the software development process. In this model, development and testing activities are concurrent, unlike the Waterfall model. This process allows more communication between customers, developers, managers, and testers.

**How PHP works on servers**

When a client (browser) sends a request for a PHP page to the server, the server reads the requested file from the disk (storage) and sends this file to the interpreter. The interpreter then runs the PHP code, fetches the DB data (if required) and puts it into HTML tags. Once the interpreter completes its tasks, it sends the result back to the server, which sends this data to the client (browser) that made a request for this page.

**MVC architecture with PHP**

The Model-View-Controller concept involved in software development evolved in the late 1980s. It’s a software architecture built on the idea that the logic of an application should be separated from its presentation. A system developed on the MVC architecture should allow a front-end developer and a back-end developer to work on the same system without interfering with each other.

*Model*

Model is the name given to the component that will communicate with the database to manipulate the data. It acts as a bridge between the View component and the Controller component in the overall architecture. It doesnt matter to the Model component what happens to the data when it is passed to the View or Controller components.

The code snippet for running first\_model.php is:

<?php

class Model

{

public $string;

public function \_\_construct()

{

$this->string = Lets start php with MVC;

}

}

?>

*View*

The View requests for data from the Model component and then its final output is determined. View interacts with the user, and then transfers the user’s reaction to the Controller component to respond accordingly. An example of this is a link generated by the View component, when a user clicks and an action gets triggered in the Controller.

To run first\_view.php, type:

<?php

class View

{

private $model;

private $controller;

public function \_\_construct($controller,$model)

{

$this->controller = $controller;

$this->model = $model;

}

public function output()

{

return<p>. $this->model->string . </p>;

}

}

?>

*Controller*

The Controller’s job is to handle data that the user inputs or submits through the forms, and then Model updates this accordingly in the database. The Controller is nothing without the user’s interactions, which happen through the View component.

The code snippet for running first\_controller.php is:

<?php

class Controller

{

private $model;

public function \_\_construct($model)

{

$this->model = $model;

}

}

?>

A simple way to understand how MVC works is as follows:

1) A user interacts with View.

2) The Controller handles the user input, and sends the information to the model.

3) Then the Model receives the information and manipulates it (either saving it or updating it by communicating with the database).

4) The View checks the state of the Model and responds accordingly (lists updated information).

In the following code snippet, by running first\_example.php we can see our MVC architecture work with PHP.

To run first\_example.php, you must type:

<?php

$model = new Model();

$controller = new Controller($model);

$view = new View($controller, $model);

echo $view->output();

?>

ROLE OF MEMBERS

**FRONTEND**

A front-end developer creates websites and application using web languages such as HTML, CSS, and JavaScript that allows users to access and interact with the site or app

**BACKEND**

A back-end developer works to develop the behind the scenes portions of a website or application, like data storage, security, site performance, or other server-side functions.

**UI/UX**

A UI UX developer is responsible for applying interactive and visual design principles on websites and web applications for a positive and cohesive user experience.

**DOCUMENTATION**

A documenter ensures that project requirements are fulfilled and to establish traceability concerning what has been done, who has done it, and when it has been done.

**DATA ANALYST**

A Data Analyst is responsible for analyzing data using statistical techniques, implementing and maintaining databases, gathering data from primary and secondary sources, identifying, analyzing and interpreting trends from the data.

MODULES

1. ***Homepage***

A homepage is the first touchpoint between you and a potential customer — you can think of it as a digital storefront for your online business. With roughly three seconds to make a good impression, you need to be strategic with your design choices on your homepage.

     High-performing [ecommerce homepages](https://www.bigcommerce.com/articles/ecommerce/) typically share the following elements:

* A clear set of product categories, either in the header or sidebar section.
* Hero images (or image sliders) featuring the main product or current offers.
* Curated presentation of recommended products, trending items, or product categories.
* Enter and exit pop-up or sticky offers for retention.

1. ***Shopping Page***

The shop page displays all the products by loading the products using PHP. On clicking the buy now button, an individual can access the page of an individual product. Products are split into various categories and divided into different page sections. We have made a page toggler with previous, 1, 2 and next buttons on it.

1. ***Registration***

If a new user wants to login to the page he/she can do so by signing up to the registration page. A person is prompted to provide a name, email and a password. Using PHP and JavaScript we prompt the user to give a password of length 6.

1. ***Login***

Returning users login through this page using the credentials they provided earlier during registration.

The Login username will be matched with the database.

1. ***Account Details***

Account details can be accessed by using the user icon, it consist of name and email of the user and also the provision of changing password.

1. ***Product Information Page***

The [product information management](https://resources.fabric.inc/glossary/pim) (PIM) module contains all the information related to the products in the e-commerce catalog. Some of the information managed by PIM includes:

* Title
* Descriptions
* Attributes
* Technical specifications
* Images
* PIM also handles the [taxonomies and relationships](https://resources.fabric.inc/answers/pim-data) between your products. This allows you to segment the display of products, making it easier for customers to search for the items they are looking for.

1. ***Order Management***

View, edit, create and fulfil orders from admin panel. Create one or multiple invoices, shipments and credit memos per order to allow for split fulfilment. Print invoices and packing slips. Email Notifications of Orders.

1. ***Cart***

The shopping cart allows customers to select and hold the products they wish to purchase. It tracks the user’s session, allowing them to leave the site and come back later with items still in their cart. It  has a [database](https://resources.fabric.inc/answers/shopping-cart-database-design) to store and retrieve the data necessary for the checkout process. There are three types of data including static, session, and processed data.

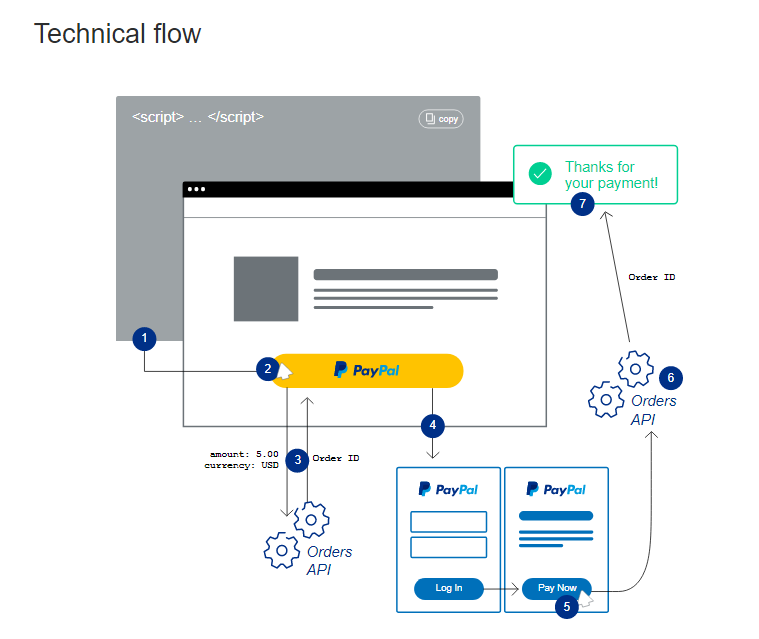
1. ***Checkout***

During the checkout process, the cart collects the customer’s payment information. This information is passed to the third-party payment processor. The details of the order are sent to other modules like the order management system (OMS), inventory management system, and customer relationship management (CRM) system.

10. ***Contact Us Page***

      Displays the contact details, address and mobile number of the site owner.

11. ***Payments Integration with PayPal***



1. We add the payment buttons to our web page.
2. The buyer clicks a button.
3. The button calls the PayPal [Orders API](https://developer.paypal.com/api/orders/v2/) to set up a transaction.
4. The button launches the PayPal Checkout experience.
5. The buyer approves the payment.
6. The button calls the PayPal [Orders API](https://developer.paypal.com/api/orders/v2/) to finalize the transaction.
7. We show a confirmation message to your buyer.

The **PayPal sandbox** is a self-contained, virtual testing environment that simulates the live PayPal production environment. The sandbox provides a shielded space where you can initiate and watch while your apps process PayPal API requests without touching any live PayPal accounts.

SOFTWARE REQUIREMENTS

Specific Requirements

* **Sell Configured to Ordered Products.**

1. The system shall display all the products that can be configured.
2. The system shall allow the user to select the product to configure.
3. The system shall display all the available components of the product to configure
4. The system shall enable users to add one or more components to the configuration.
5. The system shall notify the user about any conflict in the current configuration.
6. The system shall allow users to update the configuration to resolve conflict.
7. The system shall allow user to confirm the completion of current configuration

* **Detailed product Categorizations**

1. The system shall display detailed product categorization to the user.

* **Provide Comprehensive Product Details**

1. The system shall display detailed information of the selected products.
2. The system shall provide browsing options to see product details

* **Provide Filter facility based on price**

1. The system shall enable users to filter products based on price range.
2. The system shall enable users to navigate between results on the shop page.
3. The system shall notify the user when no matching product is found.

* **Maintain customer profile**

1. The system shall allow the user to create a profile and set his credential.
2. The system shall authenticate user credentials to view the profile.
3. The system shall allow user to update the profile information

* **Provide personalized profile**

1. The system shall display both the active and completed order history in the customer profile.
2. The system shall allow users to select the order from the order history.
3. The system shall display the detailed information about the selected order.
4. The system shall display the most frequently searched items by the user in the profile.
5. The system shall allow users to register for newsletters and surveys in the profile.

* **Provide Customer Support**

1. The system shall provide online help, FAQ’s customer support, and sitemap options for customer
2. support.
3. The system shall allow the user to select the support type he wants.
4. The system shall allow users to enter the customer and product information for the support.
5. The system shall display the customer support contact numbers on the screen

* **Detailed invoice for customer**

1. The system shall display a detailed invoice for the current order once it is confirmed.
2. The system shall optionally allow the user to print the invoice.

* **Provide shopping cart facility**

1. The system shall provide a shopping cart during online purchase.
2. The system shall allow users to add/remove products in the shopping cart.

* **Online purchase of products**

1. The system shall allow users to confirm the purchase.
2. The system shall enable users to enter the payment information.

 Usability

Graphical User Interface

1. The system shall provide a uniform look and feel between all the web pages.
2. The system shall provide a digital image for each product in the product catalog.
3. The system shall provide use of icons and toolbars.

Reliability & Availability

Performance

1. The product shall be web based and has to be run from a web server.
2. The product shall take initial load time depending on internet connection strength which also
3. depends on the media from which the product is run.
4. The performance shall depend upon hardware components of the client/customer.

Data Storage

1. The customer’s web browser shall never display a customer’s password.  It shall always be

echoed with special characters representing typed characters.

1. The customer’s web browser shall never display a customer’s credit card number after retrieving

from the database.  It shall always be shown with just the last 4 digits of the credit card number.

1. The system’s back-end servers shall never display a customer’s password.  The customer’s

password may be reset but never shown.

1. The system’s back-end servers shall only be accessible to authenticated administrators.
2. The system’s back-end databases shall be encrypted.

Supportability

Interfaces

1. There are many types of interfaces as such supported by the E-Store software system namely;
2. User Interface, Software Interface and Hardware Interface.
3. The protocol used shall be HTTP.
4. The port used is localhost:8000
5. There shall be a logical address of the system in IPv4 format.

User Interfaces

1. The user interface for the software shall be compatible with any browser such as Microsoft Edge, Mozilla or Google by which user can access the system.

Software Interfaces

1. The e-store system shall communicate with the Configurator to identify all the

available components to configure the product.

1. The e-store shall communicate with the content manager to get the product

specifications, offerings and promotions.

1. The e-store system shall communicate with billPay system to identify available

payment methods , validate the payments and process payment.

1. The e-store system shall communicate with the CRM system to provide support.
2. The e-store system shall communicate with the Sales system for order management.

Communications Interfaces

1. The e-store system shall use the HTTP protocol for communication over the internet and

for the intranet communication will be through TCP/IP protocol suite.

**Technologies Used:**

*Frontend:*

HTML, CSS, Javascript, Bootstrap

*Scripting Language:*

PHP

*Database:*

MySQL

We used MySQL because we have structured data and we needed a traditional relational database.

*Payment Integration*: Paypal API for payments.

MySQL TABLES

Text

Description automatically generated Text

Description automatically generated with low confidence

Text

Description automatically generated Text, table, letter

Description automatically generated

Table

Description automatically generated with low confidence

SCHEMA DIAGRAM

Diagram

Description automatically generated

ER DIAGRAM

Diagram

Description automatically generated

DATABASE



Graphical user interface, text, email, website

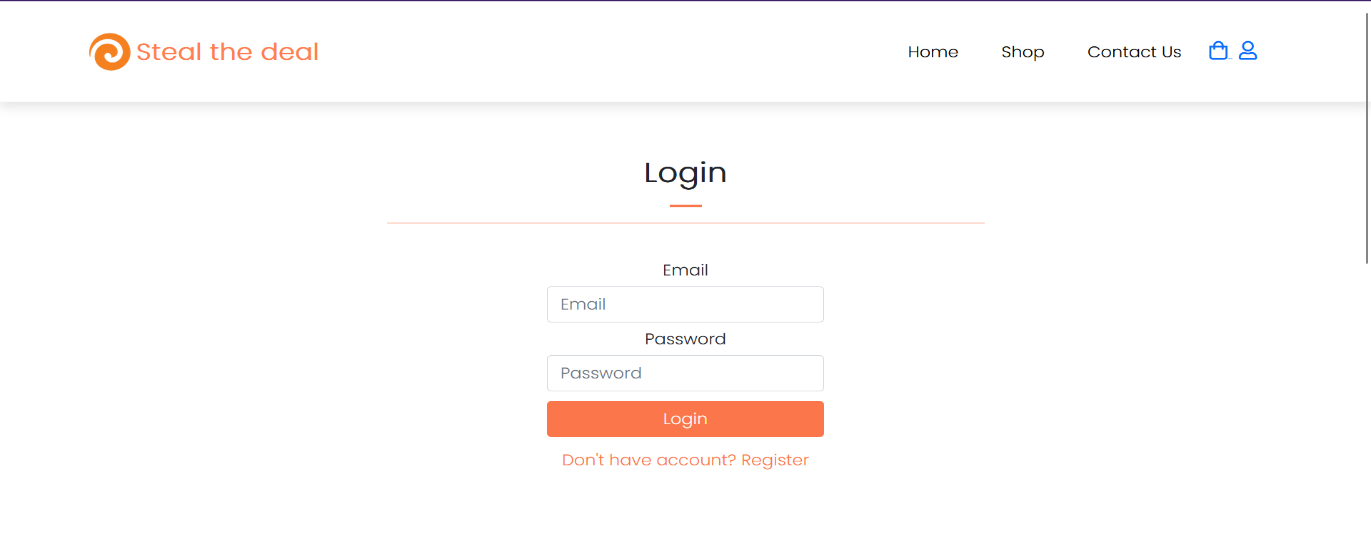
Description automatically generated

Graphical user interface, text, email, website

Description automatically generated

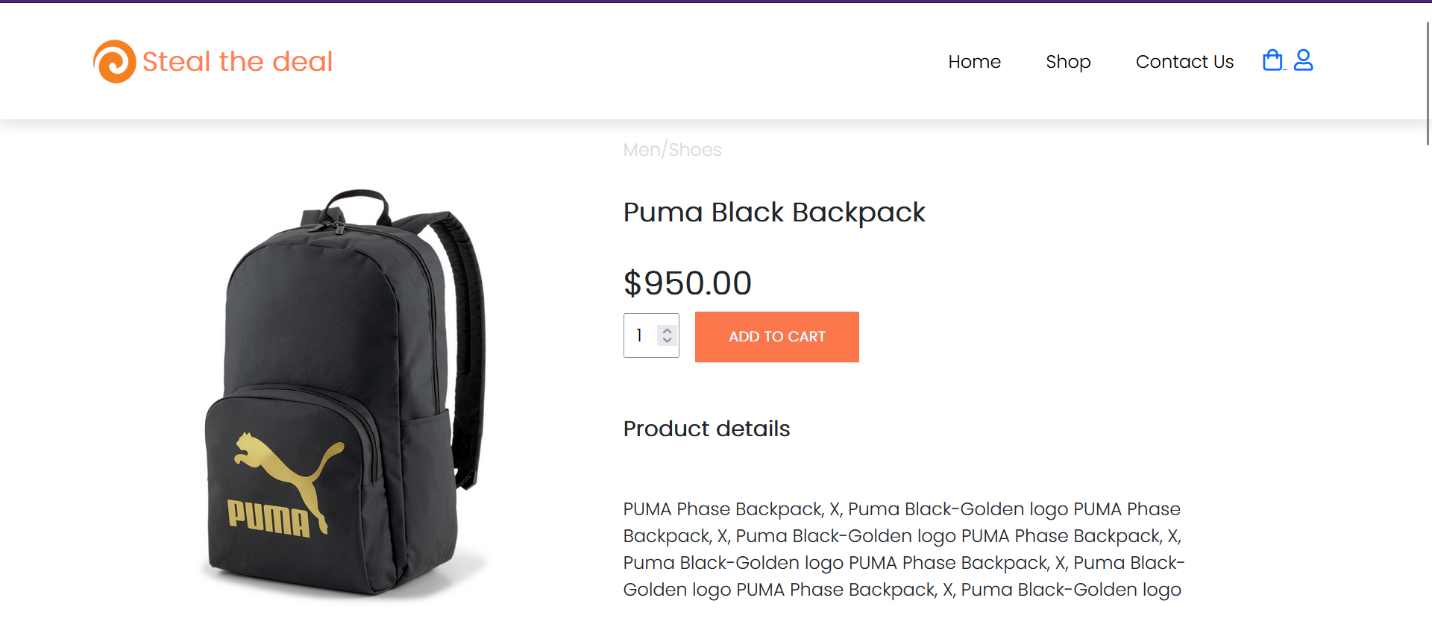
FRONTEND

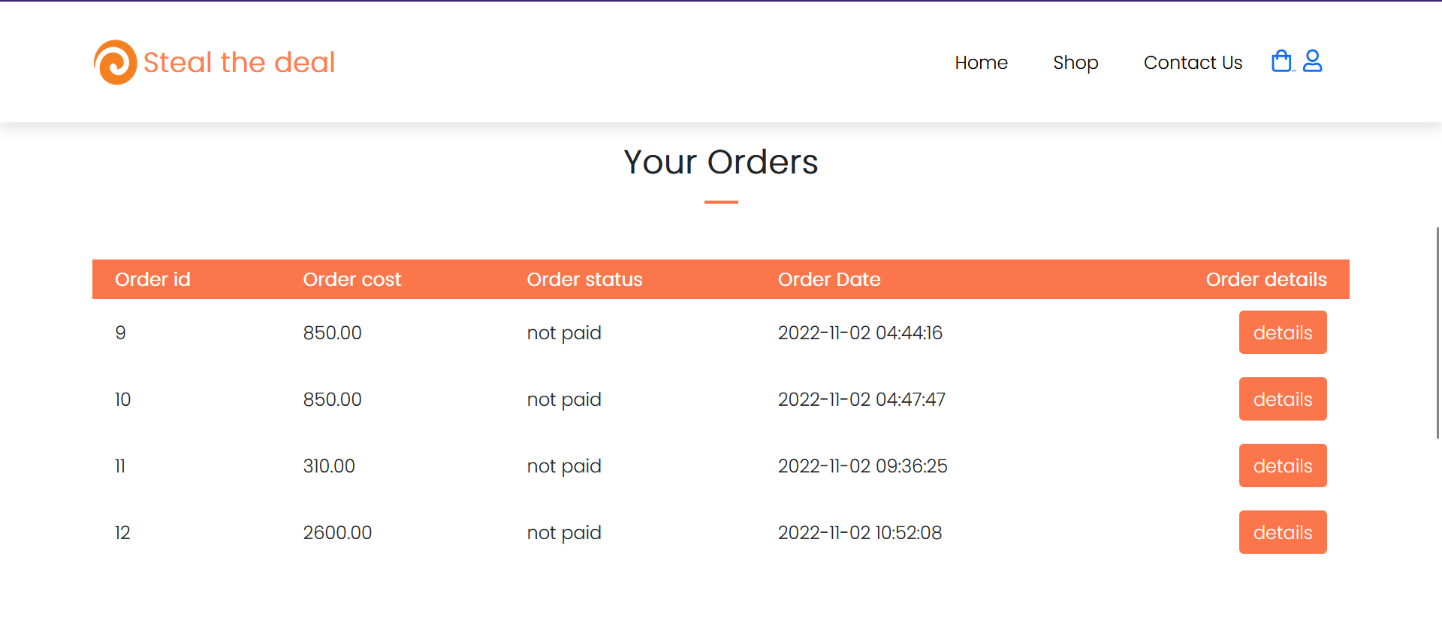
**Various Pages:**



Graphical user interface, website

Description automatically generated





SCOPE FOR FUTURE

1. A full fledged online shopping site accompanies the customer from selecting the products to providing a review post delivery. Hence an addition of a mode of delivery module with the following functionality:

* Providing a flexible delivery date time slot.
* Adding an order tracking functionality.
* Allowing an exchange and return of goods if permitted by the retailer.

1. This online service can be taken to a new level by the altering the login and registration page by allowing new retailers to register their e-shops on this site and login through the same for addition of new goods or updation of discounts or alternation and addition of products available.
2. Allowing alternative payment modes to suit a wide range of customers and their preferred mode of transaction.
3. Integrate a recommender system wherein similar products or frequently bought together products are suggested to the customer based on their recent searches.

CONCLUSION

* Successfully built a website with a database of products for consumers to purchase.

REFERENCES

[1] Bootstrap Documentation : [Bootstrap](https://getbootstrap.com/docs/5.2/getting-started/introduction/) is a feature-packed frontend tool.

[2] David W, (2001) “E-Commerce Strategy, Technologies and Applications”, McGraw Hill, pp. 3-143

[3] PHP Documentation: <https://www.php.net/manual/en/>

[4] Payments integration and checkout : <https://developer.paypal.com/docs/checkout/standard/integrate/>

[5] UI/UX Case Study: An Ecommerce App - <https://medium.com/@vedanshu/ui-ux-case-study-ecommerce-app-be3ce12d845f>

[6] Top 12 Shopping Apps UI/UX Design Case Studies - <https://medium.muz.li/top-12-shopping-app-ui-design-case-study-3dfaf5f18d35>

[7] e-Store Functions and Modules: <https://www.artio.net/magento-e-commerce/estore-functions-and-modules>