

Mini Project Report

of

Database Systems Lab (CSE 2262)

E -Commerce Market Place

SUBMITTED

BY

ANURAG KASAT

Reg No.:-210962180

Roll No.:58

Joy Podder

Reg No. 210962184

Roll No. 59

Section:A

Department of Computer Science and Engineering

**Manipal Institute of Technology, Manipal.**

**May 2023**

****

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal

13/05/2023

CERTIFICATE

This is to certify that the project titled E -Commerce Market Place is a record of the bonafide work done by Anurag Kasat, Joy Podder (Reg. No. 210962180,Reg No. 210962184) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology B. Tech Computer Science and Engineering (Artificial Intelligence & Machine Learning) of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and signature of examiners:

1. Dr.Dinesh Acharya U,Professor, CSE Dept
2. Asst. Prof Rajesh Gopakumar, CSE Dept

**TABLE OF CONTENTS**

**1. ABSTRACT**

**2. INTRODUCTION**

**3. PROBLEM STATEMENT & OBJECTIVES**

**4. TABLE CREATION**

**5. ER DIAGRAM**

**6. SQL QUERIES AND PROCEDURES**

**7. RESULTS AND SNAPSHOTS**

**8. REFERENCES**

**Abstract:**

The Ecommerce Database Management System (DBS) project is aimed at developing a robust and efficient database system to support an ecommerce platform. The database system is designed to manage data for an online store including products, orders, customers, payments, and shipping. The Ecommerce DBS project aims to provide a user-friendly interface that enables authorized users to manage the ecommerce platform's data with ease. The system is developed using database management tools and programming languages such as SQL Php and node js to implement the data model, manage data access, and process data transactions.

The project scope includes data modeling, database design, implementation, and testing of the system. The database is designed to be scalable and capable of handling the growth of data in the ecommerce platform. The project also focuses on ensuring the system's availability, and security.

Project is expected to provide features such Top Sale for when an item is listed for sale, Recommendation according to user Interests. Hierarchical Listing of products.

**INTRODUCTION**

An ecommerce website is an online platform that enables businesses to sell their products or services to customers over the internet. It allows customers to browse products, place orders, make payments, and receive their purchases without leaving their homes. Ecommerce websites have become an increasingly popular way for businesses to reach customers and increase sales, especially in the digital age where people are more inclined to shop online.

An ecommerce website typically includes a range of features that facilitate online transactions. These features may include product listings with descriptions and images, shopping carts, secure payment gateways, shipping options, customer reviews and ratings, and customer service support. The website may also include tools for managing inventory, tracking orders, and analyzing sales data.

Building an ecommerce website requires a combination of technical skills, such as web development and design, as well as business acumen to understand the needs of customers and optimize the online shopping experience. Ecommerce platforms like Shopify, WooCommerce, and Magento provide pre-built templates and tools that can simplify the process of creating an ecommerce website.

Successful ecommerce websites require a combination of effective marketing strategies, excellent customer service, and a seamless online shopping experience. By creating an online storefront that is easy to use, secure, and reliable, businesses can reach new customers and drive sales growth.

**PROBLEM STATEMENT & OBJECTIVES:-**

E-commerce provides an easy way to sell products to a large customer base.The purpose of any e-commerce website is to help customers narrow down their broad ideas and enable them to finalize the products

The problem statement for an E-commerce marketplace database system involves developing a centralized platform that connects multiple sellers with many potential buyers. The database system should be designed to manage and store large amounts of data associated with product listings, seller profiles, customer transactions, and payment details.

The E-commerce marketplace database system should provide a user-friendly interface for both sellers and buyers. Sellers should be able to create profiles, list their products, manage their inventory, and view their sales. Buyers should be able to search and browse through products, add them to their cart.

Overall, the problem statement for an E-commerce marketplace database system involves creating a scalable, reliable, and secure platform that connects multiple sellers with many potential buyers. The system should provide a user-friendly interface, high scalability, and high security while providing features for automatic categorization, recommendations, and filtering to help customers find relevant products easily.

Develop a scalable and secure platform.

Provide a user-friendly interface.

Enable efficient product management.

**Functional Requirements:**

* Choosing of Brands(Category)
* Filter based on company
* Add to cart / Wishlist
* Popularity(Top Selling)
* Reviews
* Description of product

**Data:**

* Types of Brands
* Specification
* Product description
* Customer data
* Reviews
* Top Sales
* WishList

**TABLES:-**

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

CREATE TABLE cart (

  cart\_id int(11) NOT NULL AUTO\_INCREMENT,

  user\_id int(11) NOT NULL,

  item\_id int(11) NOT NULL,

  wishlisted BOOLEAN NOT NULL DEFAULT FALSE,

  PRIMARY KEY (cart\_id),

  FOREIGN KEY (user\_id) REFERENCES user(user\_id),

  FOREIGN KEY (item\_id) REFERENCES product(item\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE product (

  item\_id int(11) NOT NULL AUTO\_INCREMENT,

  item\_brand varchar(200) NOT NULL,

  item\_name varchar(255) NOT NULL,

  item\_price double(10,2) NOT NULL,

  item\_image varchar(255) NOT NULL,

  PRIMARY KEY (item\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE user (

  user\_id int(11) NOT NULL AUTO\_INCREMENT,

  first\_name varchar(100) NOT NULL,

  last\_name varchar(100) NOT NULL,

  PRIMARY KEY (user\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE wishlist (

  cart\_id int(11) NOT NULL,

  user\_id int(11) NOT NULL,

  item\_id int(11) NOT NULL,

  PRIMARY KEY (cart\_id),

  FOREIGN KEY (user\_id) REFERENCES user(user\_id),

  FOREIGN KEY (item\_id) REFERENCES product(item\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE top\_sales (

  item\_id int(11) NOT NULL,

  sales\_count int(11) NOT NULL DEFAULT 0,

  PRIMARY KEY (item\_id),

  FOREIGN KEY (item\_id) REFERENCES product(item\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE brand (

  brand\_id int(11) NOT NULL AUTO\_INCREMENT,

  brand\_name varchar(200) NOT NULL,

  PRIMARY KEY (brand\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE cart ADD CONSTRAINT fk\_cart\_user FOREIGN KEY (user\_id) REFERENCES user(user\_id);

ALTER TABLE cart ADD CONSTRAINT fk\_cart\_product FOREIGN KEY (item\_id) REFERENCES product(item\_id);

ALTER TABLE wishlist ADD CONSTRAINT fk\_wishlist\_user FOREIGN KEY (user\_id) REFERENCES user(user\_id);

ALTER TABLE wishlist ADD CONSTRAINT fk\_wishlist\_product FOREIGN KEY (item\_id) REFERENCES product(item\_id);

ALTER TABLE wishlist ADD CONSTRAINT fk\_wishlist\_cart FOREIGN KEY (cart\_id) REFERENCES cart(cart\_id);

ALTER TABLE top\_sales ADD CONSTRAINT fk\_top\_sales\_product FOREIGN KEY (item\_id) REFERENCES product(item\_id);

ALTER TABLE product MODIFY item\_id int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=14;

ALTER TABLE user MODIFY user\_id int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

ALTER TABLE cart MODIFY cart\_id int(11) NOT NULL AUTO\_INCREMENT;

INSERT INTO product (item\_id, item\_brand, item\_name, item\_price, item\_image) VALUES

(1, 'Samsung', 'Samsung Galaxy 10', 80000.00, './assets/products/1.png'),

(2, 'Redmi', 'Redmi Note 7', 15000.00, './assets/products/2.png'),

(3, 'Redmi', 'Redmi Note 6', 14000.00, './assets/products/3.png'),

(4, 'Redmi', 'Redmi Note 5', 13000.00, './assets/products/4.png'),

(5, 'Redmi', 'Redmi Note 4', 12000.00, './assets/products/5.png'),

(6, 'Redmi', 'Redmi Note 8', 20000.00, './assets/products/6.png'),

(7, 'Redmi', 'Redmi Note 9', 30000.00, './assets/products/8.png'),

(8, 'Redmi', 'Redmi Note', 25000.00, './assets/products/10.png'),

(9, 'Samsung', 'Samsung Galaxy S6', 70000.00, './assets/products/11.png'),

(10, 'Samsung', 'Samsung Galaxy S7', 75000.00, './assets/products/12.png'),

(11, 'Apple', 'Apple iPhone 5', 20000.00, './assets/products/13.png'),

(12, 'Apple', 'Apple iPhone 6', 30000.00, './assets/products/14.png'),

(13, 'Apple', 'Apple iPhone 7', 40000.00, './assets/products/15.png');

INSERT INTO top\_sales (item\_id, sales\_count) VALUES

(1, 100),

(2, 50),

(3, 75),

(6, 200);

INSERT INTO brand (brand\_name) VALUES

('Samsung'),

('Redmi'),

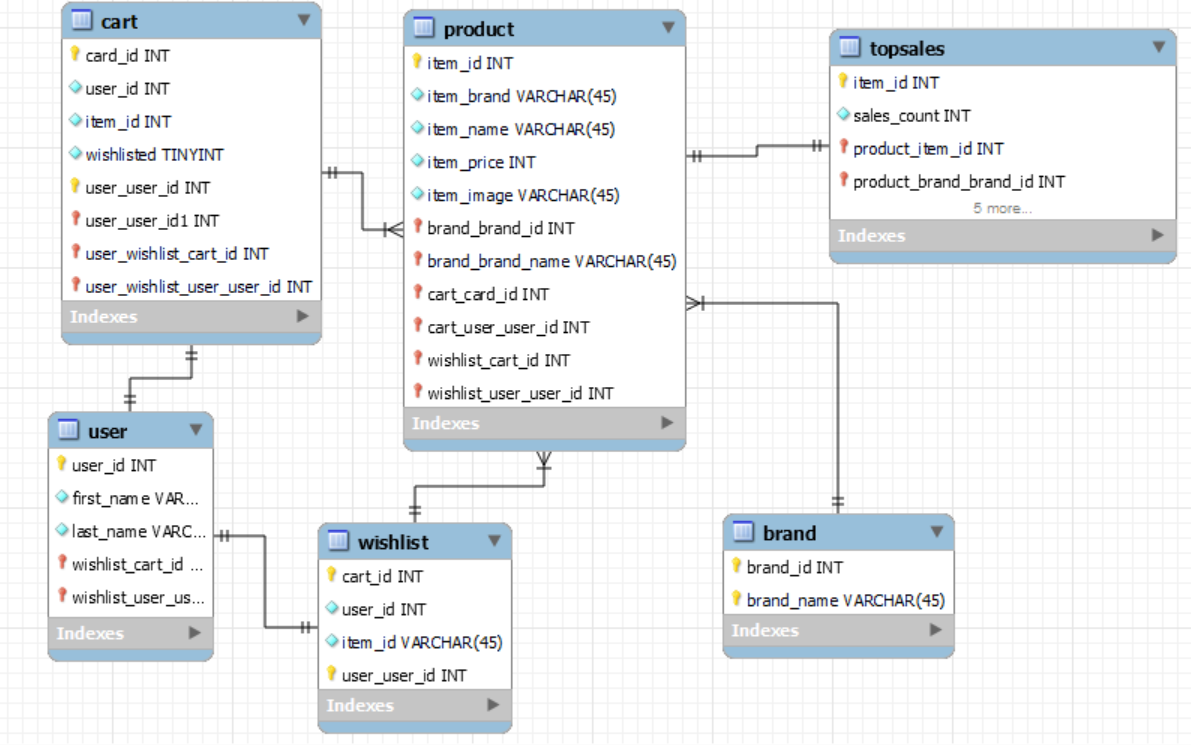
('Apple');

INSERT INTO user (user\_id, first\_name, last\_name) VALUES

(1, 'Anurag', 'Kasat'),

(2, 'Joy', 'Podder');

**ER DIAGRAM:**

****

**Queries,Connectivity & Data Access:-**

<?php

// php cart class

class Cart

{

    public $db = null;

    public function \_\_construct(DBController $db)

    {

        if (!isset($db->con)) return null;

        $this->db = $db;

    }

    // insert into cart table

    public  function insertIntoCart($params = null, $table = "cart"){

        if ($this->db->con != null){

            if ($params != null){

                // "Insert into cart(user\_id) values (0)"

                // get table columns

                $columns = implode(',', array\_keys($params));

                $values = implode(',' , array\_values($params));

                // create sql query

                $query\_string = sprintf("INSERT INTO %s(%s) VALUES(%s)", $table, $columns, $values);

                // execute query

                $result = $this->db->con->query($query\_string);

                return $result;

            }

        }

    }

    // to get user\_id and item\_id and insert into cart table

    public  function addToCart($userid, $itemid){

        if (isset($userid) && isset($itemid)){

            $params = array(

                "user\_id" => $userid,

                "item\_id" => $itemid

            );

            // insert data into cart

            $result = $this->insertIntoCart($params);

            if ($result){

                // Reload Page

                header("Location: " . $\_SERVER['PHP\_SELF']);

            }

        }

    }

    // delete cart item using cart item id

    public function deleteCart($item\_id = null, $table = 'cart'){

        if($item\_id != null){

            $result = $this->db->con->query("DELETE FROM {$table} WHERE item\_id={$item\_id}");

            if($result){

                header("Location:" . $\_SERVER['PHP\_SELF']);

            }

            return $result;

        }

    }

    // calculate sub total

    public function getSum($arr){

        if(isset($arr)){

            $sum = 0;

            foreach ($arr as $item){

                $sum += floatval($item[0]);

            }

            return sprintf('%.2f' , $sum);

        }

    }

    // get item\_it of shopping cart list

    public function getCartId($cartArray = null, $key = "item\_id"){

        if ($cartArray != null){

            $cart\_id = array\_map(function ($value) use($key){

                return $value[$key];

            }, $cartArray);

            return $cart\_id;

        }

    }

    // Save for later

    public function saveForLater($item\_id = null, $saveTable = "wishlist", $fromTable = "cart"){

        if ($item\_id != null){

            $query = "INSERT INTO {$saveTable} SELECT \* FROM {$fromTable} WHERE item\_id={$item\_id};";

            $query .= "DELETE FROM {$fromTable} WHERE item\_id={$item\_id};";

            // execute multiple query

            $result = $this->db->con->multi\_query($query);

            if($result){

                header("Location :" . $\_SERVER['PHP\_SELF']);

            }

            return $result;

        }

    }

}

<?php

class DBController

{

    // Database Connection Properties

    protected $host = 'localhost';

    protected $user = 'root';

    protected $password = '';

    protected $database = "shopee";

    // connection property

    public $con = null;

    // call constructor

    public function \_\_construct()

    {

        $this->con = mysqli\_connect($this->host, $this->user, $this->password, $this->database);

        if ($this->con->connect\_error){

            echo "Fail " . $this->con->connect\_error;

        }

    }

    public function \_\_destruct()

    {

        $this->closeConnection();

    }

    // for mysqli closing connection

    protected function closeConnection(){

        if ($this->con != null ){

            $this->con->close();

            $this->con = null;

        }

    }

}

<?php

// Use to fetch product data

class Product

{

    public $db = null;

    public function \_\_construct(DBController $db)

    {

        if (!isset($db->con)) return null;

        $this->db = $db;

    }

    // fetch product data using getData Method

    public function getData($table = 'product'){

        $result = $this->db->con->query("SELECT \* FROM {$table}");

        $resultArray = array();

        // fetch product data one by one

        while ($item = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

            $resultArray[] = $item;

        }

        return $resultArray;

    }

    // get product using item id

    public function getProduct($item\_id = null, $table= 'product'){

        if (isset($item\_id)){

            $result = $this->db->con->query("SELECT \* FROM {$table} WHERE item\_id={$item\_id}");

            $resultArray = array();

            // fetch product data one by one

            while ($item = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

                $resultArray[] = $item;

            }

            return $resultArray;

        }

    }

}

**CONVERTED SQL QURIES:-**

**CART:-**

CREATE OR REPLACE TYPE item\_type AS OBJECT (

item\_id NUMBER,

price NUMBER

);

/

-- This type will be used to represent items in the shopping cart and the wishlist.

CREATE OR REPLACE TYPE item\_list\_type AS TABLE OF item\_type;

/

-- This type will be used to represent a list of items in the shopping cart..

CREATE OR REPLACE PROCEDURE insert\_into\_cart (

i\_user\_id NUMBER,

i\_item\_id NUMBER

) IS

BEGIN

INSERT INTO cart (user\_id, item\_id)

VALUES (i\_user\_id, i\_item\_id);

END;

/

-- The procedure inserts a new row into the cart table with the given user and item IDs.

CREATE OR REPLACE PROCEDURE delete\_cart (

i\_item\_id NUMBER

) IS

BEGIN

DELETE FROM cart

WHERE item\_id = i\_item\_id;

END;

/

-- The procedure deletes all rows from the cart table that match the given item ID.

CREATE OR REPLACE PROCEDURE save\_for\_later (

i\_item\_id NUMBER

) IS

cart\_items item\_list\_type;

BEGIN

SELECT item\_type(item\_id, price)

BULK COLLECT INTO cart\_items

FROM cart

WHERE item\_id = i\_item\_id;

IF cart\_items.COUNT > 0 THEN

FORALL i IN 1..cart\_items.COUNT

INSERT INTO wishlist (item\_id, price)

VALUES (cart\_items(i).item\_id, cart\_items(i).price);

DELETE FROM cart

WHERE item\_id = i\_item\_id;

END IF;

END;

/

-- If cart\_items is not empty, the procedure then uses a FORALL statement to insert each item in cart\_items into the wishlist table, and finally deletes all rows from the cart table that match the given item ID.

CREATE OR REPLACE FUNCTION get\_sum (

i\_cart\_items item\_list\_type

) RETURN NUMBER IS

l\_sum NUMBER := 0;

BEGIN

FOR i IN 1..i\_cart\_items.COUNT LOOP

l\_sum := l\_sum + i\_cart\_items(i).price;

END LOOP;

RETURN l\_sum;

END;

/

-- The function calculates the sum of all prices in i\_cart\_items and returns the result as a NUMBER.

-- Example usage

DECLARE

l\_cart\_items item\_list\_type;

BEGIN

l\_cart\_items := item\_list\_type(item\_type(1, 10), item\_type(2, 20), item\_type(3, 30));

DBMS\_OUTPUT.PUT\_LINE('Total: ' || get\_sum(l\_cart\_items));

END;

/

**DBController:-**

CREATE OR REPLACE PACKAGE DBController IS

PROCEDURE connect\_to\_database;

PROCEDURE close\_database\_connection;

END DBController;

CREATE OR REPLACE PACKAGE BODY DBController IS

conn SYS\_REFCURSOR;

PROCEDURE connect\_to\_database IS

BEGIN

conn := SYS\_REFCURSOR(SQL.STRING('mysql:host=localhost;dbname=shopee'), 'root', '');

END connect\_to\_database;

-- The connection is opened using a SYS\_REFCURSOR object, which is a type of Oracle cursor that can handle multiple result sets.

PROCEDURE close\_database\_connection IS

BEGIN

IF conn%ISOPEN THEN

conn.close;

END IF;

END close\_database\_connection;

END DBController;

-- The connection is opened using a SYS\_REFCURSOR object, which is a type of Oracle cursor that can handle multiple result sets.

**Product:-**

CREATE OR REPLACE TYPE item\_type AS OBJECT (

item\_id NUMBER,

item\_name VARCHAR2(100),

item\_price NUMBER

);

/

CREATE OR REPLACE TYPE item\_type\_table AS TABLE OF item\_type;

/

CREATE OR REPLACE FUNCTION get\_product\_data (p\_table\_name IN VARCHAR2)

RETURN item\_type\_table PIPELINED

IS

v\_item item\_type;

BEGIN

FOR item IN (SELECT item\_id, item\_name, item\_price FROM p\_table\_name)

LOOP

v\_item := item\_type(item.item\_id, item.item\_name, item.item\_price);

PIPE ROW(v\_item);

END LOOP;

RETURN;

END;

/

-- The get\_product\_data function takes a table name as input and returns a pipelined table of item\_type objects. The function uses a cursor to select all rows from the specified table and constructs an item\_type object for each row. The PIPE ROW statement is used to pipe each object out of the function and into the calling environment, allowing the caller to iterate over the result set.

CREATE OR REPLACE FUNCTION get\_product (p\_item\_id IN NUMBER, p\_table\_name IN VARCHAR2)

RETURN item\_type

IS

v\_item item\_type;

BEGIN

SELECT item\_type(item\_id, item\_name, item\_price)

INTO v\_item

FROM p\_table\_name

WHERE item\_id = p\_item\_id;

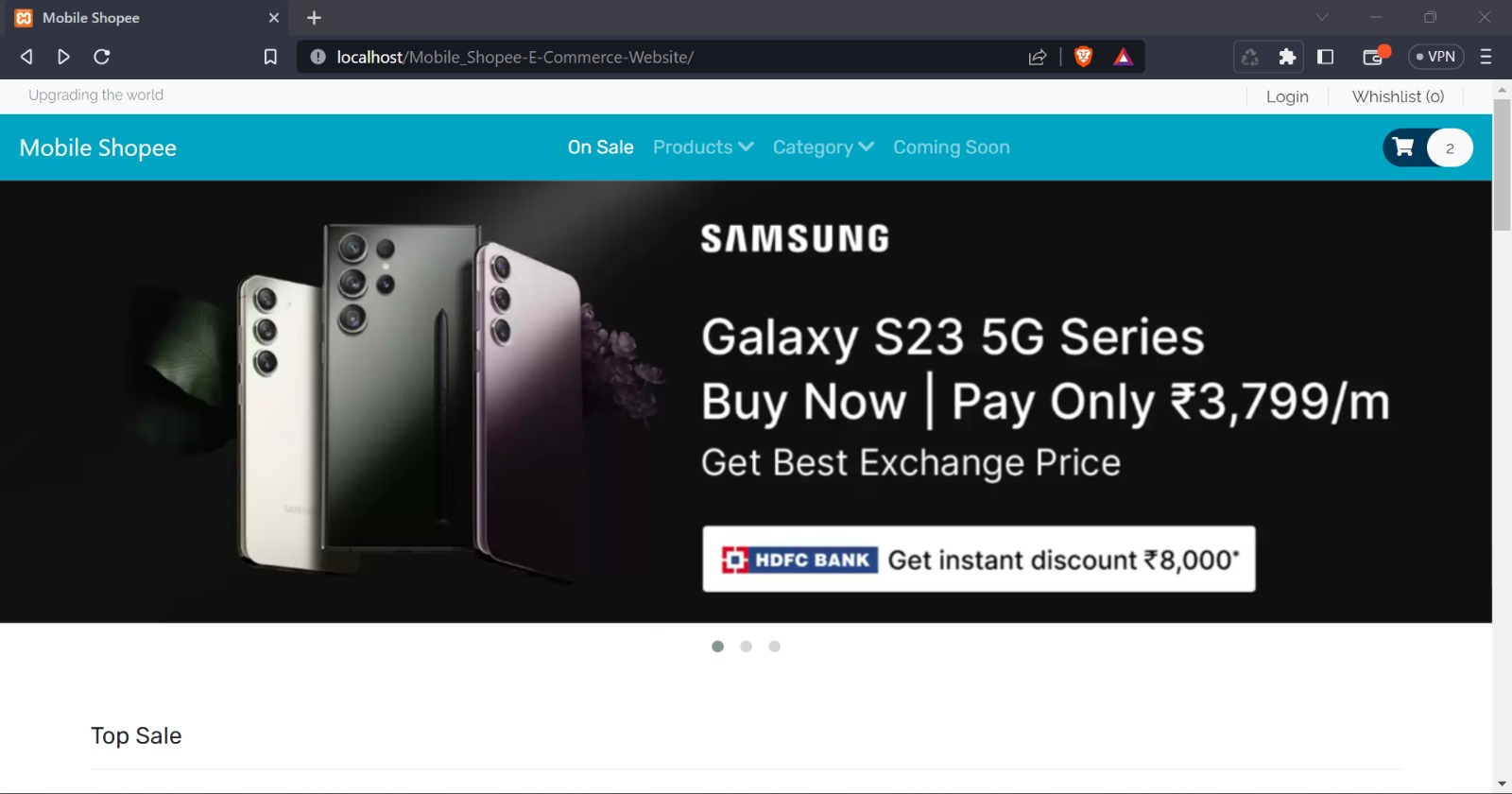
RETURN v\_item;

END;

/

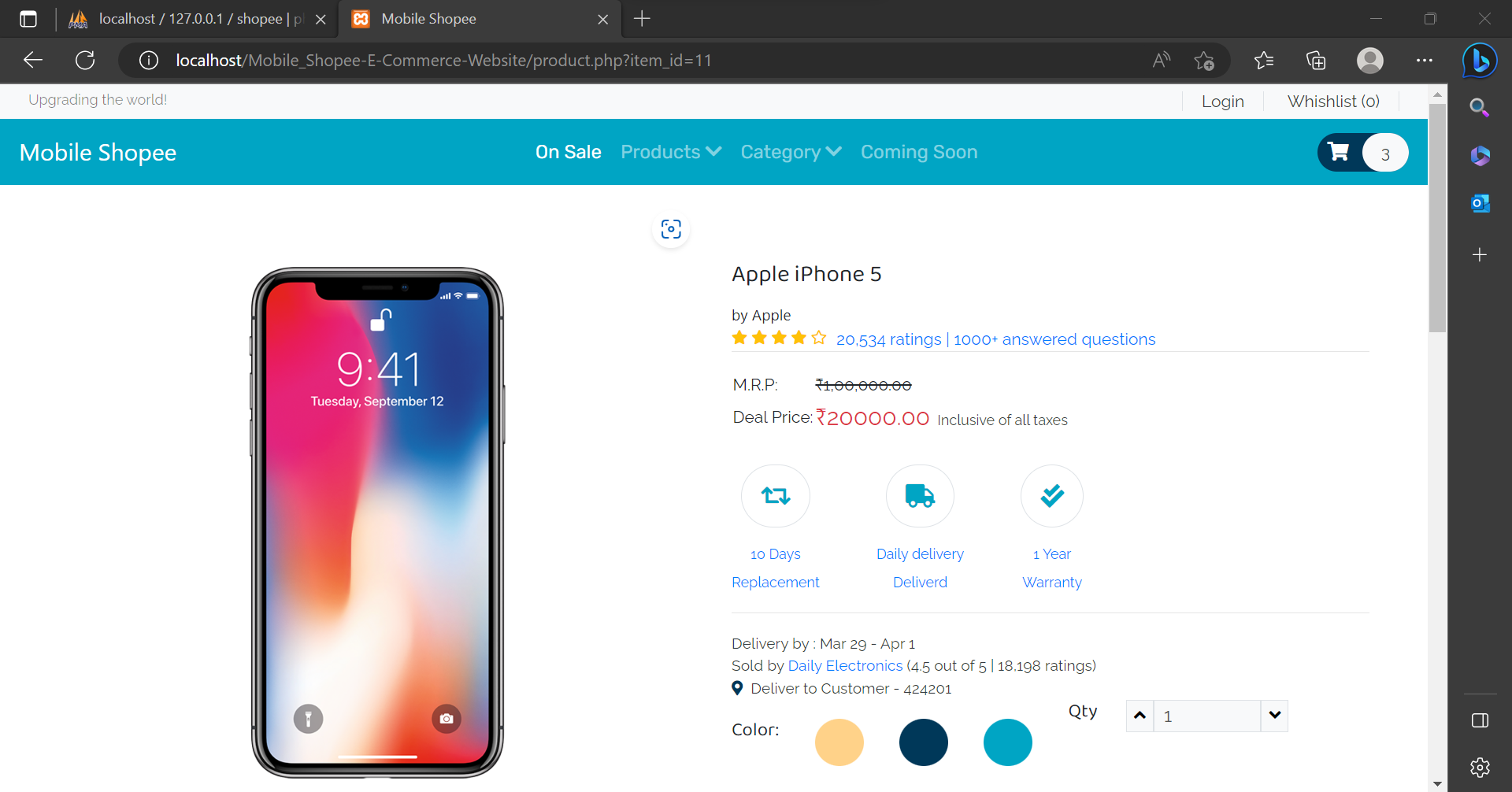
-- The get\_product function takes an item\_id and a table name as inputs and returns a single item\_type object that matches the specified item\_id.

**UI/UX:-**

****

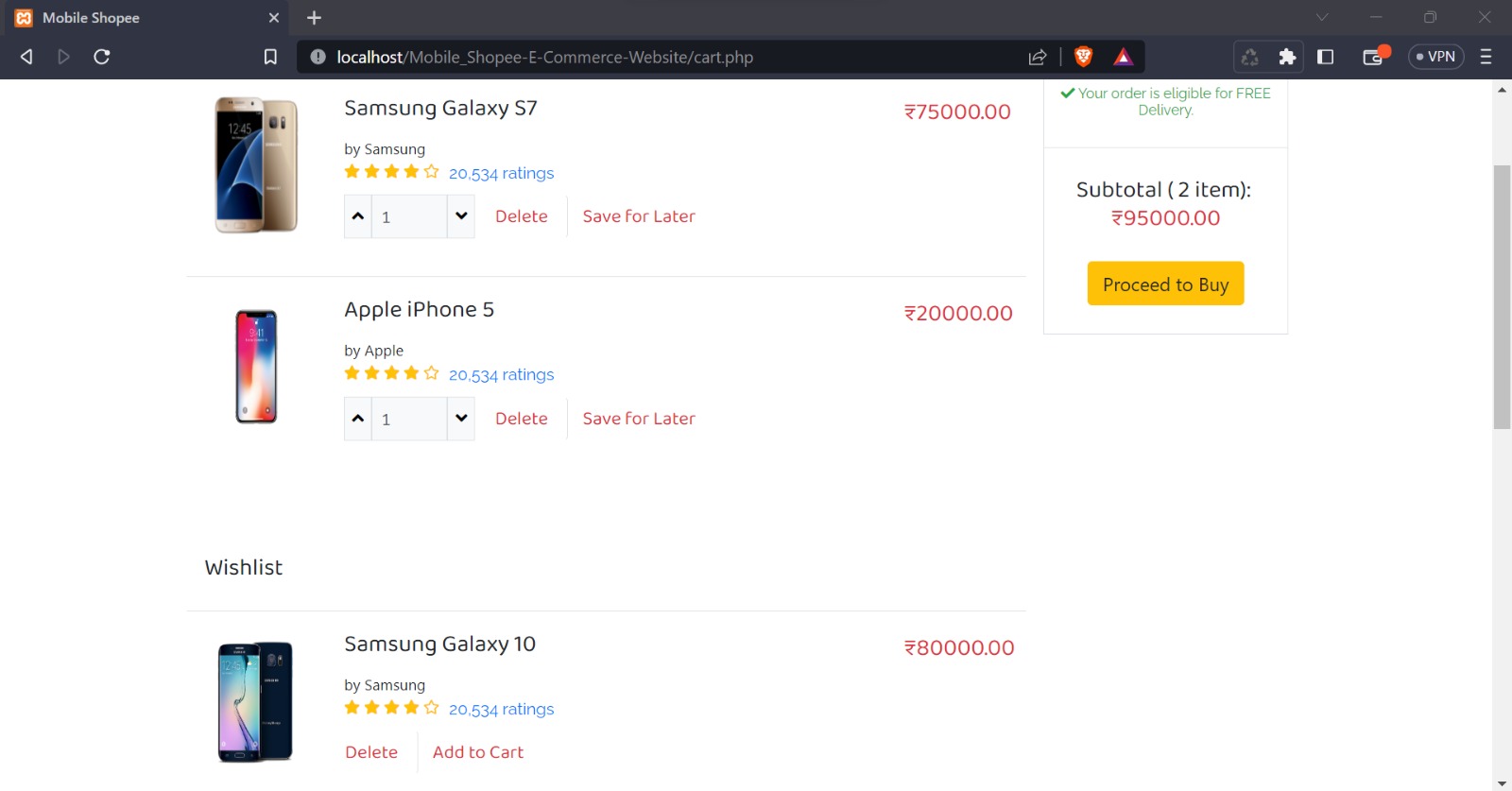
**A screenshot of a cell phone

Description automatically generated with low confidence**

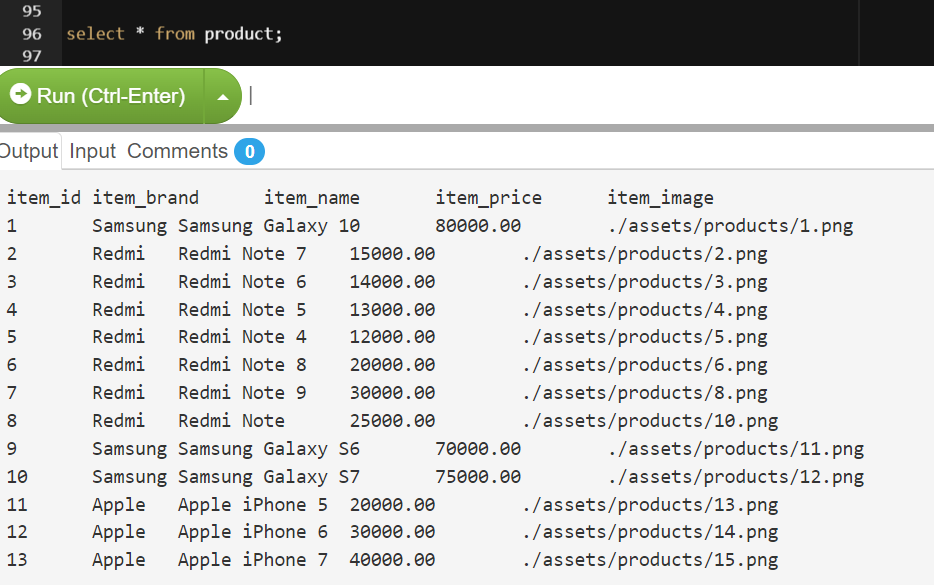
****

**A screenshot of a computer

Description automatically generated with medium confidence**

****

**RESULTS AND SNAPSHOTS:**

****

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

**REFERENCES: -**

1. Node.js documentation: The official Node.js documentation provides detailed information on how to use the Node.js platform, including instructions on how to create web applications and interact with databases using Node.js.
2. PHP documentation: The official PHP documentation provides detailed information on how to use the PHP programming language, including instructions on how to create web applications and interact with databases using PHP.
3. MySQL documentation: The official MySQL documentation provides detailed information on how to use the MySQL database management system, including instructions on how to create and manage databases, tables, and data.
4. Dahl, R. (2009). Node.js: Asynchronous event-driven JavaScript runtime. Retrieved from <https://nodejs.org/en/about/>
5. Dubois, P. (2014). MySQL Cookbook: Solutions for Database Developers and Administrators. O'Reilly Media.
6. PHP. (n.d.). In PHP: Hypertext Preprocessor. Retrieved from <https://www.php.net/>