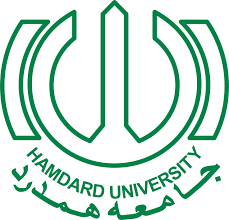
****

**Project Title:** Online Shopping Store System

**Course Name:** Data Structures & Algorithms

**Program:** BSCS-03

**Submitted By :**

YOUSRA ZAHEER(1790-2023)

SYEDA MAIRA ABID GARDAZI(1940-2023)

MUHAMMAD DANIAL MUZAFFAR(1647-2023)

**Submitted To:** Sir Asif Javed

**Submission Date:** 22nd January, 2025

**Institute:** Hamdard University Islamabad

**Table of Content:**

1. **Introduction**
   1. **Significance of the Project**
   2. **Scope of the Project**
2. **Objectives of the Online Shopping Store** 
   1. **User Management**
   2. **Product Categories**
   3. **Display of Products**
   4. **Sorting**
   5. **Searching**
   6. **Cart Management**
   7. **Order and Payment Process**
3. **Literature Review**
   1. **Background Information On Data Structures and Algorithms**
   2. **Related Work**

**4. System Design**

**4.1 Architecture**

* 1. **Data Structures Used**
* **Array**
  + 1. **Phone Array**
    2. **Laptop Array**
    3. **Smartwatches Array**
* **Structures**
  + 1. **Phone Structure**
    2. **Laptop Structure**
    3. **Smartwatches Structure**
  1. **Algorithm Design**
* **Sorting (Bubble sort)**
  + 1. **Sorting Phones**
    2. **Sorting laptops**
    3. **Sorting smartwatches**
* **Searching**
  + 1. **Binary Search**

**5. Implementation**

* 1. **programming language**
  2. **Development Environment**
  3. **Code’s Output Snippets**
     1. **User Registration**
     2. **Displaying Products (Phones)**
     3. **Cart Summary**
     4. **Payment Method**

**6. Testing and Validation**

* **Test cases**
  1. **User Registration Test Case**
  2. **Product Search (Phone Search) Test Case**
  3. **Cart Functionality Test case**
  4. **Payment Selection Test case**
  5. **Order Confirmation Test case**
* **validation**
  1. **Algorithm Testing**
  2. **Code Review**
  3. **Edge Cases**
  4. **Performance Testing**

**7. Conclusion**

* **7.1 Summary**
* **7.2 Future Work**

**1. Introduction**

The Online Shopping Store System is a C++ program that simulates an e-commerce platform. It allows users to browse, sort, and search for products such as phones, laptops, and smartwatches. Additionally, the system provides functionalities for adding items to a cart and completing orders through a payment system. The project showcases the use of structures, arrays, sorting, searching algorithms, and user interaction in C++.

**1.1 Significance of the Project**

* The project introduces key programming principles, such as data structures, sorting, searching, and user interaction.
* It provides a practical context for learning algorithm implementation, such as Bubble Sort and Binary Search.

**1.2 Scope of the Project**

* E-commerce is a rapidly growing industry, and understanding its core functionality is essential for aspiring to build softwares in this domain.

**2. Objectives of the Online Shopping Store**

* 1. **User Management**
  + Users can register by providing their name, email, and password.
  + This data is stored temporarily within the session for interaction.

**2.2 Product Categories**

* + The system manages three product categories:
  + **phones, laptops, and smartwatches.**
  + Each product type is represented by a structure with relevant details such as name, model, price, and specifications like RAM, storage, camera resolution for phones, and processor type for laptops.ok;.

**2.3 Display of Products**

* + Users can view all products in a tabular format. The data for each category (phones, laptops, and smartwatches) is displayed with specific attributes such as price, RAM, processor, etc.

Example output for phones:

+----------+-----------------+----------------------+------------+--------------+---------------+----------------------------------+----------------+

| Phone ID | Phone Name | Model | Price RAM | Storage | Colours Available | Camera Resolution |

+----------+---------------+-----------------------+------------+---------------+---------------+----------------------------------+----------------+

| 1 | iPhone | iPhone 16 Pro Max | 540500 | 8 GB | 1 TB | Black Titanium | 48 MP |

| 2 | Samsung | Samsung A32 | 71000 | 4 GB | 128 GB | Awesome Black | 64 MP |

+----------+---------------+----------------------+------------+----------+---------------+----------------------------------------+----------------+

* 1. **Sorting**
  + The system provides sorting functionalities for each category (phones, laptops, and smartwatches) based on the product name.
  + The sorting algorithm implemented is a simple **bubble sort**.

**2.5 Searching**

* + A **binary search** function is provided to search for a product by name in each category.
  + It returns the product details if found or indicates that the product is not available.

**2.6 Cart Management**

* + Users can add products to their cart by specifying the quantity and product IDs.
  + The system then displays the items added to the cart, along with their quantities and calculated prices.
  + It also calculates the total cost of the cart.

Example cart output for phones:

Category: PHONES

-----------------------------------------------------

Items Quantity Cost

-----------------------------------------------------

iPhone 16 Pro 2 1081000 Rs

Samsung A32 1 71000 Rs

-----------------------------------------------------

Subtotal: 1152000 Rs

**2.7 Order and Payment Process**

* + After viewing their cart, users are prompted to proceed with payment.
  + They can choose between **Cash on Delivery** or **Online Payment**.
  + Shipping details are provided, and the user can choose a shipping option based on the delivery city, affecting the final price.

Example payment screen:

Choose a payment method

1. Cash on Delivery

2. Online Payment

Enter your shipping choice:

Shipping Details:

1. Other city: Rs400

2. Islamabad Delivery: Rs200

3. Self pick: Rs0

### 3. Literature Review

#### 3.1 Background Information on Data Structures and Algorithms

Data structures are fundamental concepts in computer science that allow for efficient storage and manipulation of data. Common data structures include arrays, linked lists, stacks, queues, trees, and hash tables. The choice of data structure impacts the performance of algorithms that operate on the data, such as search, sorting, and update operations.

* **Data Structures**

In this project, the primary data structures used are **arrays** and **structures**. Arrays are used to store multiple instances of products, while structures are employed to define complex types that represent individual products (**phones, laptops, smartwatches**).

* **Algorithms**

Key algorithms like **Bubble Sort** and **Binary Search** are employed in the system. **Bubble Sort** is a simple sorting algorithm that compares and swaps adjacent elements to sort an array. Though not the most efficient in terms of time complexity (O(n²)), it is useful for small-scale applications where simplicity is key. **Binary Search** is a more efficient searching algorithm, with a time complexity of O(log n), used for finding specific products within the sorted arrays.

#### 3.2 Related Work

Several similar projects have implemented e-commerce systems with a focus on product management and user interaction. For instance, many online shopping platforms have used arrays or linked lists to store product details, and data structures like stacks or queues have been applied to manage the cart system or checkout flow.

1. **E-commerce Platforms with Product Management**: In systems such as online book stores or electronics shopping platforms, arrays and structures are commonly used to store product information. These systems typically allow users to browse, search, and filter through products. Similar to the current project, sorting (using algorithms like bubble sort or merge sort) and searching (binary search or hash-based search) techniques are employed to efficiently manage the product listings.
2. **Shopping Cart Systems**: Shopping cart systems also often use arrays or linked lists to manage items in the cart. The cart may be implemented as an array of product structures where each product's ID, name, quantity, and price are stored. Operations like calculating totals, applying discounts, and checking out typically rely on sorting and searching algorithms to ensure correct prices are calculated and the checkout process is efficient.
3. **Payment Systems**: The integration of payment methods, including cash on delivery (COD) or online payment methods, is a standard feature in many online shopping platforms. Such systems store user payment data securely and perform calculations to estimate shipping costs and taxes, just as in this project's payment method implementation.

**4. System Design**

**4.1 Architecture**

**START**   
**Include Libraries**  
**int main()**  
│  
▼  
**Display Main Menu**  
├──> **Option 1: User Registration**  
│ ├── Input User Details (Name, Email, Password)  
│ └── Return to Main Menu  
│  
├──> **Option 2: View Products**  
│ └── Display Phones, Laptops, Smartwatches  
│ └── Return to Main Menu  
│  
├──> **Option 3: Add to Cart**  
│ ├── Input Product ID and Quantity  
│ ├── Calculate Subtotal  
│ └── Proceed to Payment  
│  
├──> **Option 4: Search Items**  
│ └── Input Item Name  
│ └── Perform Binary Search  
│ └── Return Found/Not Found  
│  
├──> **Option 5: Exit**  
│ └── End Program  
│  
▼  
**Payment Method**  
│  
├── Choose: COD or Online Payment  
└── Proceed to Order Details  
│  
▼  
**Order Details**  
│  
├── Input Shipping Details  
│  
▼  
**Confirm Order**  
│  
└── Display "Order Confirmed"  
│  
▼  
**END**

* 1. **Data Structures Used**
* **Arrays**  are used to store collections of products such as phones, laptops, and smartwatches. Each array holds a fixed number of products, with each product represented by a structure containing attributes such as name, price, specifications, etc.
* **Purpose**: Arrays allow easy access and management of product data.

**4.2.1 Phone Array**

Phone p[5] = {

{1, "iPhone", "iPhone 16 Pro Max", 540500, "8 GB", "1 TB", "Black Titanium", "48 MP"},

{2, "Samsung", "Samsung A32", 71000, "4 GB", "128 GB", "Awesome Black", "64 MP"},

{3, "OnePlus", "OnePlus 12", 219960, "2 GB", "256 GB", "Silky Black", "50 MP"},

{4, "Infinix", "Infinix Note 12", 45499, "6 GB", "128 GB", "Ice Blue", "48 MP"},

{5, "Huawei", "Huawei Mate 40", 228999, "8 GB", "512 GB", "Mystic Silver", "50 MP"}

};

**4.2.2 Laptop Array**

Laptop l[5] = {

{1, "Apple", "MacBook Air", "10.13.6", 224999, "16 GB", "512 GB", "Midnight Black", "i7"},

{2, "HP", "Envy x360", "Windows 11", 204500, "16 GB", "512 GB", "Gold", "Ryzen 7"},

{3, "Dell", "XPS 13", "Windows 11", 175000, "8 GB", "256 GB", "Frost", "i5"},

{4, "Lenovo", "ThinkPad X1", "12th Gen", 75000, "16 GB", "256 GB", "Black", "i7"},

{5, "Asus", "ZenBook 14", "Windows 10", 90000, "8 GB", "256 GB", "Ponder Blue", "i5"}

};

**4.2.3 Smartwatch Array**

Smartwatch s[5] = {

{1, "Sveston watch", 11000, "260 mAh", "5-7 days", "240x240"},

{2, "Apple Ultra 2", 10000, "64 GB", "36 hours", "410x410"},

{3, "Samsung Galaxy", 11100, "16 GB", "24 hours", "396x396"},

{4, "Huawei Fit 3", 49000, "1.5 GB", "10 days", "1.82-inch"},

{5, "LG W7 watch", 52000, "4 GB", "100 hours", "360x360"}

};

* **Structures** (structs) are used to represent individual product categories, allowing each product to have a set of attributes. For example, the Phone, Laptop, and Smartwatch structures store details about each respective product.
* **Purpose**: Structures group related information together, making it easy to manage and access each product's details.Functions like displayPhones(), displayLaptops(), and displaySmartwatches() are used to print the product details in a structured format.

**4.2.4 Phone Structure**

struct Phone {

int phone\_id;

string phone\_name;

string model;

int price;

string RAM;

string Storage;

string colours\_available;

string camera\_resolution;

};

**4.2.5 Laptop Structure**

struct Laptop {

int laptop\_id;

string laptop\_name;

string model:

int price;

string RAM;

string SSD;

string colours\_available;

string processor;

};

**4.2.6 Smartwatch Structure**

struct Smartwatch {

int smartwatch\_id;

string product\_name;

int price;

string capacity;

string battery\_health;

string resolution; };

**4.3 Algorithm Design**

* **Sorting Algorithm** the sortPhones(), sortLaptops(), and sortSmartwatches() functions implement sorting using the Bubble Sort algorithm.
* **Bubble Sort Implementation** the system uses bubble sort to sort products by name. Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

**4.3.1 Sorting Phones**

void sortPhones(Phone p[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (p[j].phone\_name > p[j + 1].phone\_name) {

// Swap the two phone elements

Phone temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

}

}

}

}

**4.3.2 Sorting Laptops:**

void sortLaptops(Laptop l[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (l[j].laptop\_name > l[j + 1].laptop\_name) {

// Swap the two laptop elements

Laptop temp = l[j];

l[j] = l[j + 1];

l[j + 1] = temp;

}

}

}

}

**4.3.3 Sorting Smartwatches**

void sortSmartwatches(Smartwatch s[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (s[j].product\_name > s[j + 1].product\_name) {

// Swap the two smartwatch elements

Smartwatch temp = s[j];

s[j] = s[j + 1];

s[j + 1] = temp;

}

}

}

}

* **Searching Algorithm** the binarySearchPhones(), binarySearchLaptops(), and binarySearchSmartwatches() functions allow for efficient searching of products
* **Binary Search Implementation** the system uses binary search to efficiently search for products by name. The binary search function is implemented as follows:

void binarySearchPhones(Phone p[], int size, const string& searchName) {

int low = 0, high = size - 1;

bool found = false;

while (low <= high) {

int mid = (low + high) / 2;

if (p[mid].phone\_name == searchName) {

cout << "Phone Found: " << p[mid].phone\_name << " | Price: Rs. " << p[mid].price << endl;

found = true;

break;

} else if (p[mid].phone\_name < searchName) {

low = mid + 1;

} else {

high = mid - 1;

}

}

if (!found) {

cout << "Phone not found with name: " << searchName << endl;

}

}

**5. Implementation**

#### 5.1 Programming Language

The project is implemented in **C++**. C++ is chosen because it offers powerful features such as object-oriented programming, direct memory management, and efficient handling of complex data structures like arrays and structures. These features make it well-suited for developing console-based applications with product listings, sorting, and searching functionalities.

#### 5.2 Development Environment

The development environment used for this project is **Code::Blocks IDE**. Code::Blocks is an open-source Integrated Development Environment (IDE) that supports C++ programming. It provides various features, such as:

* Syntax highlighting
* Debugging support
* Project management
* Auto-completion
* Easy integration of libraries and debugging tools

This IDE makes development faster and easier by providing an intuitive interface and useful tools for C++ development.

#### 5.3 Code’s Output Snippets

**5.3.1** **User Registration**

\*\*\*\*\*\*\*\*\* Register TO Online Shopping Store \*\*\*\*\*\*\*\*\*

Enter your Name:

Enter your Email:

Enter your Password:

* + 1. **Displaying Products (Phones)**

+----------+---------------+----------------------+------------+----------+---------------+----------------------------------------+--------------------+

| Phone ID | Phone Name | Model | Price (Rs) | RAM | Storage | Colours Available | Camera Resolution |

+----------+---------------+----------------------+------------+----------+---------------+----------------------------------------+--------------------+

| 1 | iPhone | iPhone 16 Pro Max | 540500 | 8 GB | 1 TB | Black Titanium | 48 MP |

+----------+---------------+----------------------+------------+----------+---------------+----------------------------------------+--------------------+

* + 1. **Cart Summary**

Category: PHONES

-----------------------------------------------------

Items Quantity Cost

-----------------------------------------------------

iPhone 16 Pro 2 1081000 Rs

-----------------------------------------------------

Subtotal: 1081000 Rs

* + 1. **Payment Process**

Choose a payment method

1. Cash on Delivery

2. Online Payment

Enter your choice:

Shipping Details

1. Other city: Rs400

2. Islamabad Delivery: Rs200

3. Self pick: Rs0

### 6. Testing and Validation

#### Test Cases: Below are examples of the test cases, including the expected results and actual results for different functionalities of the system:

* 1. **User Registration Test Case**:
  + **Test Case 1**:
    - Input:
      * Name: John Doe
      * Email: john.doe@example.com
      * Password: password123
    - Expected Output:
      * Confirmation message: "User registration successful."
    - Actual Output:
      * "User registration successful."

**6.2 Product Search (Phone Search) Test Case**:

* + **Test Case 1**:
    - Input:
      * Search Name: "Samsung A32
    - Expected Output:
      * Product details: "Samsung A32 | Price: Rs. 71,000"
    - Actual Output:
      * "Phone Found: Samsung A32 | Price: Rs. 71,000"
  + **Test Case 2**:
    - Input:
      * Search Name: "Nokia 3310"
    - Expected Output:
      * "Phone not found"
    - Actual Output:
      * "Phone not found with name: Nokia 3310"

**6.3 Cart Functionality Test Case**:

* + **Test Case 1**:
    - Input:
      * Product ID: 2 (Samsung A32)
      * Quantity: 2
    - Expected Output:
      * Cart updated with 2 Samsung A32 phones and total cost calculated as Rs. 142,000
    - Actual Output:
      * "Added to Cart: 2 Samsung A32. Total Cost: Rs. 142,000"
  1. **Payment Selection Test Case**:
  + **Test Case 1**:
    - Input:
      * Choice: 1 (COD)
    - Expected Output:
      * "You have selected Cash on Delivery."
    - Actual Output:
      * "You have selected Cash on Delivery."
  + **Test Case 2**:
    - Input:
      * Choice: 2 (Online Payment)
    - Expected Output:
      * "You have selected Online Payment."
    - Actual Output:
      * "You have selected Online Payment."

**6.5 Order Confirmation Test Case**:

* + **Test Case 1**:
    - Input:
      * Confirmation choice: Yes
    - Expected Output:
      * "Order Confirmed!"
    - Actual Output:
      * "Order Confirmed!"

#### Validation:

To validate the correctness of the algorithms, the following steps were followed:

* 1. **Algorithm Testing**:
  + Each algorithm (such as Bubble Sort for sorting, Binary Search for searching, and Payment Method selection) was tested with multiple edge cases (e.g., searching for a non-existent product, sorting an already sorted array, etc.) to ensure that they behave as expected in all scenarios.
  + The correctness of sorting was validated by checking if the phone list was in ascending order after the sorting algorithm was applied.
  + Binary Search results were verified by confirming if the searched item was found at the correct index or was identified as "not found."

**6.5 Code Review**:

* + Code was reviewed to ensure that all functions were implemented correctly and efficiently.
  + Manual testing was conducted by running the program with different inputs to observe the output and verify that it meets the requirements.

**6.6 Edge Cases**:

* + The system was tested with edge cases, such as entering empty fields during registration, searching for products that don't exist, and entering invalid payment choices to ensure that the system handles these situations gracefully.

**6.7 Performance Testing**:

* + The algorithms were tested on small and large datasets to ensure they perform well in terms of time complexity. The Bubble Sort algorithm, while simple, was able to sort a reasonably sized product list efficiently, and Binary Search was used for quick lookups.

**7. Conclusion**

**7.1 Summary** this project successfully implements a console-based shopping system that offers core features such as user registration, product search, cart management, and order processing. It utilizes basic algorithms like Bubble Sort and Binary Search to ensure efficient product management and searching. The system effectively meets the functional requirements and demonstrates the practical application of foundational programming concepts of data structures and algorithms.

**7.2** **For future** **enhancements**

* Graphical User interface (GUI)
* Database Integration
* Secure authentication
* Personalized Recommendations
* Mobile Version