

Topic : Online Action System

Group no : MLB\_WE\_01.01\_12

Campus : Malabe

Submission Date:

We declare that this is our own work, and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

|  |  |  |
| --- | --- | --- |
| **REGISTRATION NO** | **NAME** | **CONTACT NUMBER** |
| IT21279898 | Kulasekara M. P.G. G | 0766481411 |
| IT21312908 | Fernando K.R.A. W | 0760636278 |
| IT21327544 | Pathiraja Y.P.M. A | 0765700835 |
| IT21324574 | Dissanayake H.S.B. N | 0715838074 |
| IT21268458 | Dissanayaka A.V. R | 0776646602 |

**PART 1**

• **REQUIREMENTS**

Bidder is an online auction system that provides services to both bidders and auctioneers all over the world. This online platform supports bidders to bid for products and for auctioneers to place auctions. People all over the world can use our website as an online business platform. In our system there is an admin who is responsible for generating reports on latest auctions, featured auctions, upcoming auctions, closing auctions, closed auctions and generating charts on trending list per month, monthly income, and customer interaction per month. To do these things, there are some requirements that need to be considered. The requirements are as follows.

1. A Guest can visit the online auction website using URL and browse the website
2. User can enter the system and use his/her user accountasauctioneer account or bidder account
3. Unregistered bidder needs to register to the Auction system by providing details such as name, address, contact number and country
4. Unregistered auctioneer needs to register to the Auction System by providing details such as name, address, contact number and product category
5. Registered member should login to the system by providing login credentials.
6. Both auctioneers and bidders can edit / update input details.
7. System must validate the inputs.
8. System can delete, update, store data on the website.
9. User can search for products by category/brand name/product name.
10. Bidder can save searched items.
11. Bidder Can use add to cart option.
12. Bidder Can check bid date and time.
13. Bidder Can change/cancel his/her bid product.
14. Buyer Can select shipping method (standard shipping, Economy shipping, Expedited shipping).
15. Buyer Can select payment method (debit card, credit card, pay pal).
16. Buyer Can contact the auctioneer.
17. Buyer can return products.
18. Add new order details
19. Update orders
20. Auctioneer Can add new products.
21. Auctioneer Can cancel/ start auctions
22. Auctioneer Can set the starting time and the ending time of an auction.
23. Auctioneer Can set the minimum amount for a product to bid
24. Auctioneer Can select shipping method.
25. Auctioneer Can contact the buyer.
26. Auctioneer/bidder/buyer can send feedback to the system.
27. Administrator can add/remove users according to the feedback given by bidders/auctioneers.
28. Manager can analyse reports and charts.
29. Support service can handle inquiries.
30. Auctioneer can handle return products.

* **The classes**
* User
* Unregistered bidder
* Unregistered auctioneer
* Registered bidder
* Unregistered auctioneer
* Buyer
* Product
* Payment
* Shipment
* Report
* Order
* Order details
* **CRC cards for possible classes**

|  |  |
| --- | --- |
| **Class name: User** | |
| **Responsibilities** | **Collaborations** |
| Input name |  |
| Input NIC |  |
| Search products | Product |

|  |  |
| --- | --- |
| **Class name: Unregistered bidder** | |
| **Responsibilities** | **Collaborations** |
| Register as a bidder |  |
| Search products | Product |

|  |  |
| --- | --- |
| **Class name: Registered bidder** | |
| **Responsibilities** | **Collaborations** |
| Save searched products | product |
| Add to cart | Product |
| Check bid date and time | Auctioneer |
| Update status of the bid product | Product |

|  |  |
| --- | --- |
| **Class name: Buyer** | |
| **Responsibilities** | **Collaborations** |
| Select shipping method | shipping |
| Select payment method | Payment |
| Contact auctioneer | Auctioneer |
| Return product | Product |

|  |  |
| --- | --- |
| **Class name: Unregistered auctioneer** | |
| **Responsibilities** | **Collaborations** |
| Register as an auctioneer |  |
| Search products | Product |

|  |  |
| --- | --- |
| **Class name: Registered auctioneer** | |
| **Responsibilities** | **Collaborations** |
| Add new products for the auction | product |
| Set auction |  |
| Select shipping method | Shipping |
| Display list of products | Report |
| Contact the buyer | buyer |

|  |  |
| --- | --- |
| **Class name: product** | |
| **Responsibilities** | **Collaborations** |
| Add products | Auctioneer |
| Update products |  |
| Restock products |  |

|  |  |
| --- | --- |
| **Class name : Payment** | |
| **Responsibilities** | **Collaborations** |
| Store payment details |  |
| Validate payment details | Buyer |
| Display payment methods |  |

|  |  |
| --- | --- |
| **Class name: Shipment** | |
| **Responsibilities** | **Collaborations** |
| Store shipping details |  |
| Validate shipping details | Buyer |
| Display shipping methods |  |

|  |  |
| --- | --- |
| **Class name: Report** | |
| **Responsibilities** | **Collaborations** |
| Generate list of products |  |
| Generate reports about auctions | Auctioneer |
| View reports |  |

|  |  |
| --- | --- |
| **Class name: Order** | |
| **Responsibilities** | **Collaborations** |
| Create orders | Buyer |
| Edit orders |  |

|  |  |
| --- | --- |
| **Class name: OrderDetails** | |
| **Responsibilities** | **Collaborations** |
| Calculate order amount | order |
| Calculate weight of orders | product |
| Calculate tax for each product |  |

**Exercise 1**

* **Draw the class diagram**

**Diagram, schematic

Description automatically generated**

**Exercise 2**

* **Coding**

#include <iostream>

using namespace std;

#define SIZE 2

class Product;

class Registered\_Auctioneer;

class Cart;

class Order\_Details;

class Order;

class Payment;

class Shipment;

class Buyer ;

class Report;

class User{ // User class (parent class )

protected:

string name;

string address;

string email;

public:

User(); //default constructor

User(string NA,string Add,string EM); // overloaded Constructors

void search() {}

void dislayDeatils();

};

User::User(){

cout <<endl<< "User Class " << endl;

}

User::User(string NA,string Add,string EM){

name=NA;

address=Add;

email=EM;

}

class Registered\_Auctioneer : public User{ // Registered\_Auctioneer class(Child Class)

private:

int registrationNo;

int auctioneerID;

int dateOfBirth ;

int contactNo;

string gender;

Product\* product;

Report\* report;

public:

Registered\_Auctioneer(); //default constructor

Registered\_Auctioneer(string NA,string Add,string EM,int RNo,int AID, int ADOB,int ACNo,string Gen) ;//overloaded Constructors

void auctioneerDetails();

void addProduct();

void setStartTime();

void setEndTime();

float setMinimumBid();

void displayProducts();

void contactBuyer();

};

Registered\_Auctioneer::Registered\_Auctioneer(){

cout << "Registered\_Auctioneer Class " << endl;

}

Registered\_Auctioneer::Registered\_Auctioneer(string NA,string Add,string EM,int RNo,int AID,int ADOB,int ACNo,string Gen) : User(NA , Add , EM){

name=NA;

address=Add;

email=EM;

registrationNo=RNo;

auctioneerID=AID;

dateOfBirth= ADOB;

contactNo=ACNo;

gender=Gen;

}

void Registered\_Auctioneer::auctioneerDetails(){

cout <<endl<<"Name : "<<name<<endl;

cout <<"Address : "<<address<<endl;

cout <<"E-mail : "<<email<<endl;

cout <<"Registration No : "<<registrationNo<<endl;

cout <<"Auctioneer ID : "<<auctioneerID<<endl;

cout <<"Date Of Birth: "<<dateOfBirth<<endl;

cout <<"Contact No : "<< contactNo<<endl;

cout <<"Gender : "<< gender<<endl;

}

void Registered\_Auctioneer::addProduct() {};

void Registered\_Auctioneer::setStartTime() {};

float Registered\_Auctioneer::setMinimumBid() {};

void Registered\_Auctioneer::setEndTime() {};

void Registered\_Auctioneer::displayProducts() {};

void Registered\_Auctioneer::contactBuyer() {};

class Unregistered\_Auctioneer : public User{ // Unregistered\_Auctioneer class(Child Class)

private:

Product \*Products;

public:

Unregistered\_Auctioneer(); //default constructor

void viewDetails();

};

Unregistered\_Auctioneer::Unregistered\_Auctioneer(){

cout << " Unregistered\_Auctioneer Class " << endl;

}

class Unregistered\_Bidder : public User{ // Unregistered\_Bidder class(Child Class)

private:

Product \*Products;

public:

Unregistered\_Bidder(); //default constructor

void viewDetails();

};

Unregistered\_Bidder::Unregistered\_Bidder(){

cout << " Unregistered\_Bidder Class " << endl;

}

class Registered\_Bidder : public User{ // Registered\_Bidder class(Child Class)

private:

Cart \*Carts; //an object of Cart as attribute

Product \*Products; //an object of Product as attribute

int B\_registrationNo;

int bidderID;

int B\_dateOfBirth ;

int B\_contactNo;

string B\_gender;

public:

Registered\_Bidder(); //default constructor

Registered\_Bidder(string NA,string Add,string EM,int RNo,int BID,int DOB,int CNo,string Gen );//overloaded Constructors

void bidderDetails() ;

void selectProduct(Product \*Pd);

void addToCart(Cart \*c);

void checkBid();

float addBid() ;

};

Registered\_Bidder::Registered\_Bidder(){

cout << "Registered\_Bidder Class " << endl;

}

Registered\_Bidder::Registered\_Bidder(string NA,string Add,string EM,int RNo,int BID,int DOB,int CNo,string Gen): User(NA , Add , EM){

name=NA;

address=Add;

email=EM;

B\_registrationNo=RNo;

bidderID=BID;

B\_dateOfBirth= DOB;

B\_contactNo=CNo;

B\_gender=Gen;

}

void Registered\_Bidder::bidderDetails() {

cout <<endl<<"Name : "<<name<<endl;

cout <<"Address : "<<address<<endl;

cout <<"E-mail : "<<email<<endl;

cout <<"Registration No : "<<B\_registrationNo<<endl;

cout <<"bidderID ID : "<<bidderID<<endl;

cout <<"Date Of Birth: "<<B\_dateOfBirth<<endl;

cout <<"Contact No : "<<B\_contactNo<<endl;

cout <<"Gender : "<< B\_gender<<endl;

}

void Registered\_Bidder::selectProduct(Product \*Pd){

Products=Pd;

}

void Registered\_Bidder::addToCart(Cart \*c){

Carts=c;

}

class Cart{ // Cart class

private:

int cartID;

int noOfProducts;

float price;

Registered\_Bidder \*RB;

public:

Cart(); //default constructor

Cart(int ID,int Products,float amount,Registered\_Bidder \*RBider );//overloaded Constructors

float calcTotalPrice();

};

Cart::Cart(){

cout << endl<< "Cart Class " << endl;

}

Cart::Cart(int ID,int Products,float amount,Registered\_Bidder \*RBider){

cartID=ID;

noOfProducts=Products;

price=amount;

RB=RBider;

RB->addToCart(this);

}

class Product{ // ProductCart class

private:

string name;

int productID ;

string category;

string description;

float shippingWeight;

float maxPrice;

Registered\_Bidder \*RB; //an object of Registered\_Bidder as attribute

Registered\_Auctioneer\* auctioneer;

Unregistered\_Auctioneer\* unregauctioneer;

Report\* report;

public:

Product(); //default constructor

Product(string Na,int PID,string Cap,string Des,float shipWeigh,float MAXPrice,Registered\_Bidder \*RBider );//overloaded Constructors

void displayProducts( );

void addNewProducts();

void removeProducts();

void restockProducts();

void updateProducts();

};

Product::Product(string Na,int PID,string Cap,string Des,float shipWeigh,float MAXPrice,Registered\_Bidder \*RBider ){

name=Na;

productID=PID ;

category=Cap;

description=Des;

shippingWeight=shipWeigh;

maxPrice=MAXPrice;

RB=RBider;

RB->selectProduct( this);

}

void Product::displayProducts() {

cout <<"Name : "<<name<<endl;

cout <<"Product ID : "<<productID<<endl;

cout <<"Description: "<<description<<endl;

cout <<"Shipping Weight : "<<shippingWeight<<endl;

cout <<"Max Price : "<<maxPrice <<endl;

cout <<"Bid amount : "<<RB <<endl;

}

Product::Product(){

cout<< endl << "Product Class " << endl;

}

//Order Details class

class Order\_Details {

private:

int noOfOrders;

int taxStatus;

Product\* product;

public:

Order\_Details(); //default constructor

Order\_Details(int NoOfOrders, int taxStatus); //overloading constructor

~Order\_Details(); //destructor

float calAmount();

float calWeight();

float calTax();

};

Order\_Details::Order\_Details() {

int noOfOrders = 0;

int taxStatus = 0;

}

Order\_Details::Order\_Details(int NoOfOrders, int TaxStatus) {

noOfOrders = NoOfOrders;

taxStatus = TaxStatus;

};

Order\_Details::~Order\_Details() {

cout << "Deleting Order Details" << endl;

};

float Order\_Details::calAmount() {

}

float Order\_Details::calWeight() {

}

float Order\_Details::calTax() {

}

//Order Class

class Order{

private:

int orderID;

int orderDate;

int orderTime;

Order\_Details\* details;

Product\* product;

Payment\* payment;

Shipment \*ship;

public:

Order(); //Default Constuctor

Order(int OrderID, int OrderDate, int orderTime, int NoOfOrder, int TaxStatus); //Overloading contructor

~Order(); //destructor

void createOrder(Product \*product1,Payment\* payment1);

void editOrder(int orderID);

};

Order::Order()

{

orderID = 0;

orderDate = 0;

orderTime = 0;

details = new Order\_Details(0, 0);

}

Order::Order(int OrderID, int OrderDate, int orderTime, int NoOfOrder, int TaxStatus) {

orderID = OrderID;

orderDate = OrderDate;

orderTime = orderTime;

details = new Order\_Details(NoOfOrder, TaxStatus);

}

Order::~Order() {

cout << "Deleting Order " << orderID << endl;

delete details;

}

void Order::createOrder(Product\* product1,Payment\* payment1) {

product = product1;

payment=payment1;

}

void Order::editOrder(int orderID) {

}

// Report Class

class Report {

private:

int reportId;

string description;

string dateCreated;

string timeCreated;

string lastModified;

Registered\_Auctioneer\* auctioneer;

Product \* product;

public:

Report();

void addDetails();

void viewReports();

void createSummary();

void listOfRestockItems();

void listOfPreviousItems();

~Report();

};

// Methods for Report Class

Report::Report() {

reportId = 0;

description = "No Description";

dateCreated = "2022/01/01";

timeCreated = "00:00:00";

lastModified = "00:00:00";

}

void Report::addDetails() {

}

void Report::viewReports() {

}

void Report::createSummary() {

}

void Report::listOfRestockItems() {

}

void Report::listOfPreviousItems() {

}

Report::~Report() {

cout << "Report Destructor runs" << endl;

}

//Latest Auction Report Class (Child Class to the Report Class)

class latestAuction\_report : public Report {

private:

int latestId;

public:

latestAuction\_report();

~latestAuction\_report();

};

// Methods for Latest Auction Report Class

latestAuction\_report::latestAuction\_report() {

latestId = 0;

}

latestAuction\_report::~latestAuction\_report() {

cout << "Latest Auction Report Destructor runs" << endl;

}

//Featured Auction Report Class (Child Class to the Report Class)

class featuedAuction\_report : public Report {

private:

int featuedId;

public:

featuedAuction\_report();

~featuedAuction\_report();

};

// Methods for Featured Auction Report Class

featuedAuction\_report::featuedAuction\_report() {

featuedId = 0;

}

featuedAuction\_report::~featuedAuction\_report() {

cout << "Featured Auction Report Destructor runs" << endl;

}

//Closed Auction Report Class (Child Class to the Report Class)

class closedAuction\_report : public Report {

private:

int closedId;

public:

closedAuction\_report();

~closedAuction\_report();

};

// Methods for Closed Auction Report Class

closedAuction\_report::closedAuction\_report() {

closedId = 0;

}

closedAuction\_report::~closedAuction\_report() {

cout << "Closed Auction Report Destructor runs" << endl;

}

//Upcomming Auction Report Class (Child Class to the Report Class)

class upcommingAuction\_report : public Report {

private:

int upcommingId;

public:

upcommingAuction\_report();

~upcommingAuction\_report();

};

// Methods for Upcomming Auction Report Class

upcommingAuction\_report::upcommingAuction\_report() {

upcommingId = 0;

}

upcommingAuction\_report::~upcommingAuction\_report() {

cout << "Upcomming Auction Report Destructor runs" << endl;

}

//Closing Auction Report Class (Child Class to the Report Class)

class closingAuction\_report : public Report {

private:

int closingId;

public:

closingAuction\_report();

~closingAuction\_report();

};

// Methods for Closing Auction Report Class

closingAuction\_report::closingAuction\_report() {

closingId = 0;

}

closingAuction\_report::~closingAuction\_report() {

cout << "Closing Auction Report Destructor runs" << endl;

}

class Payment {

protected :

int paymentId;

string paymentDate;

string paymentTime;

float paymentAmount;

string paymentDetails;

Buyer \*buyer;

public:

Payment();

Payment(int id,string Date,string Time,float Amount ,string Details);

void viewPaymnetDetails(Buyer \*buyer1);

void updatePaymnetDetails();

void status();

};

Payment::Payment(){

cout <<"Payment class"<< endl;

}

Payment::Payment(int id,string Date,string Time,float Amount ,string Details){

paymentId=id;

paymentDate=Date;

paymentTime=Time;

paymentAmount=Amount;

paymentDetails=Details;

}

void Payment::viewPaymnetDetails(Buyer \*buyer1) {}

void Payment::updatePaymnetDetails() {}

void Payment::status() {}

class Credit : public Payment{

private:

int cardNO;

int expirDate;

public:

Credit();

Credit(int id,string Date,string Time,float Amount ,string Details, int C\_NO, int ExDate);

};

Credit::Credit(){

cout <<"Credit class"<< endl;

}

Credit::Credit(int id,string Date,string Time,float Amount ,string Details, int C\_NO, int ExDate) : Payment(id,Date,Time,Amount,Details){

paymentId=id;

paymentDate=Date;

paymentTime=Time;

paymentAmount=Amount;

paymentDetails=Details;

cardNO=C\_NO;

expirDate=ExDate;

}

class Paypal : public Payment{

private:

string paypalEmail;

public:

Paypal();

Paypal(int id,string Date,string Time,float Amount,string Details,string Email);

};

Paypal::Paypal(){

cout <<"Paypal class"<< endl;

}

Paypal::Paypal(int id,string Date,string Time,float Amount ,string Details,string Email) : Payment(id,Date,Time,Amount,Details){

paymentId=id;

paymentDate=Date;

paymentTime=Time;

paymentAmount=Amount;

paymentDetails=Details;

paypalEmail=Email;

}

class Payoneer : public Payment{

private:

string PayoneerEmail;

public:

Payoneer();

Payoneer(int id,string Date,string Time,float Amount,string Details,string Email);

};

Payoneer::Payoneer(){

cout <<"Paypal class"<< endl;

}

Payoneer::Payoneer(int id,string Date,string Time,float Amount ,string Details,string Email) : Payment(id,Date,Time,Amount,Details){

paymentId=id;

paymentDate=Date;

paymentTime=Time;

paymentAmount=Amount;

paymentDetails=Details;

PayoneerEmail=Email;

}

class Buyer : public Registered\_Bidder

{

private:

int tokenID;

Payment \*pay;

Shipment \*ship;

Order \*ord[SIZE];

public :

Buyer();

Buyer(char \*uname, char \*uaddress, char \*uemail, int regBiNo, int bidID, char \*b\_dob, int conNum, char \*Bgen, int tokID);

void addPayementDetails();

void addShippingDetails();

void contactAuctioneer();

void returnProduct();

};

Buyer::Buyer() {};

Buyer::Buyer(char \*uname, char \*uaddress, char \*uemail, int regBiNo, int bidID, char \*b\_dob, int conNum, char \*Bgen, int tokID)

{

tokenID = tokID;

}

void addPayementDetails() {};

void addShippingDetails() {};

void contactAuctioneer() {};

void returnProduct() {};

class Shipment {

private:

int shipmentID;

string shipment;

Buyer\* buyer;

Order\* order;

public:

Shipment();

void storeDetails(int shipmentID, string shipment);

void viewShipmentDetails();

void validateDetails();

void calcDistance();

void calcShipmentPrice();

};

Shipment::Shipment() {

shipmentID = 0;

shipment = "";

cout << "Shipment class begins" << endl;

}

void Shipment::storeDetails(int ShipmentID,string Shipment) {

shipmentID = ShipmentID;

shipment = Shipment;

}

void Shipment::viewShipmentDetails() {

cout << "ShipmentID : " << shipmentID << endl

<< "Shipment : " << shipment << endl;

}

void Shipment::validateDetails() {

}

void Shipment::calcDistance() {

}

void Shipment::calcShipmentPrice() {

}

int main(void) {

Product p;

Product();

cout<<"----------------------------"<<endl;

Order\_Details OD;

Order\_Details();

cout<<"----------------------------"<<endl;

Order O;

Order();

cout<<"----------------------------"<<endl;

Payment P;

Payment();

cout<<"----------------------------"<<endl;

Shipment S;

Shipment();

cout<<"----------------------------"<<endl;

Buyer B;

Buyer();

cout<<"----------------------------"<<endl;

Report R;

Report();

cout<<"----------------------------"<<endl;

User();

User u1("Githma","Homagama","githam@gmail.com"); // create Static Objects

Registered\_Auctioneer();

Registered\_Auctioneer A1("Githma","Homagama","githam@gmail.com",001,200,2001,07664 ,"Male"); // create Static Objects

A1.auctioneerDetails();

User u2("Sujith","Colombo","Sujith@gmail.com");

Registered\_Bidder();

Registered\_Bidder B1("Sujith","Colombo","Sujith@gmail.com", 1000,500,2000,075023,"Male" ); // create Static Objects

B1.bidderDetails();

Buyer \*buy;

buy = new Buyer("Ruwani","Jaffna","ruwani@gmail.com", 100,200,"1997-06-07",0777111111,"female",600);

Unregistered\_Auctioneer();

Unregistered\_Bidder();

Registered\_Bidder \*B2=new Registered\_Bidder("Sujith","Colombo","Sujith@gmail.com", 1000,500,2000,075023,"Male" ); //create Dynamic Objects

Cart \*c1=new Cart (300,2,1000.00,B2); //create Dynamic Objects

Product \*P1 = new Product("phone",900,"Electrical","good",200,20000.00,B2); //create Dynamic Objects

Cart();

P1->displayProducts();

Product();

return 0;

}