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

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 account as auctioneer 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	

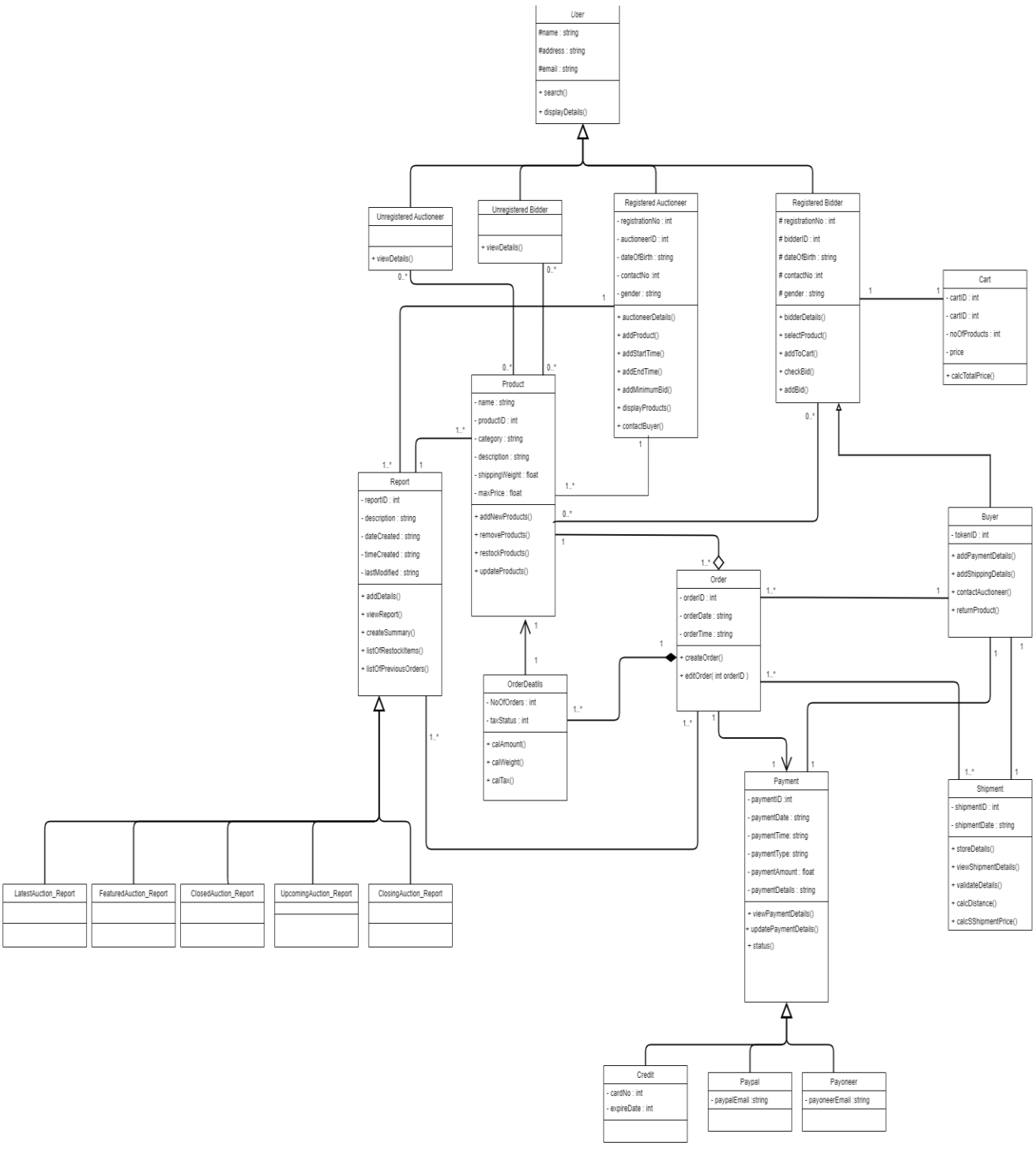
Class name: Report	
Responsibilities	**Collaborations**
Generate list of products	
Generate reports about auctions	Auctioneer
View reports	

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**



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;
}
```