Software Engineering Principles (CSI1007)

SLOT: L59 + L60

LAB ASSESSMENT: 3

Title: Build Your Own PC

Team Members:-

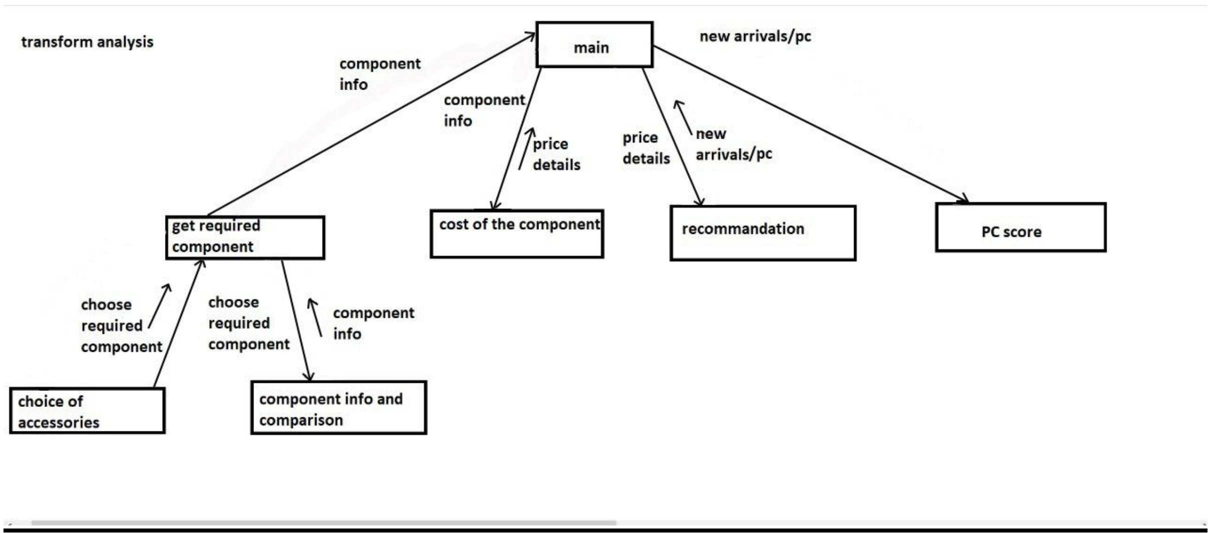
1. **Prathiban V (Co-ordinator)**
2. **Sri HariHaran R**
3. **Chandru M**

Prepare the complete Software Design document of your project. Follow the below mentioned sequence for displaying the various diagrams in the software design document

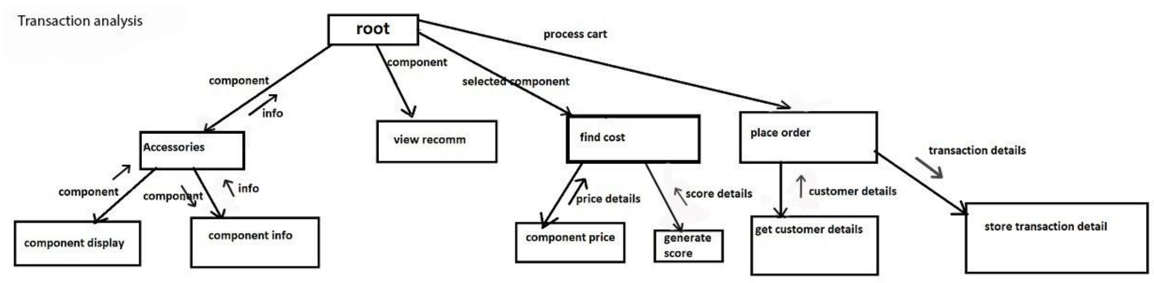
The document should contain the following diagrams.

<u>Structure Chart</u>

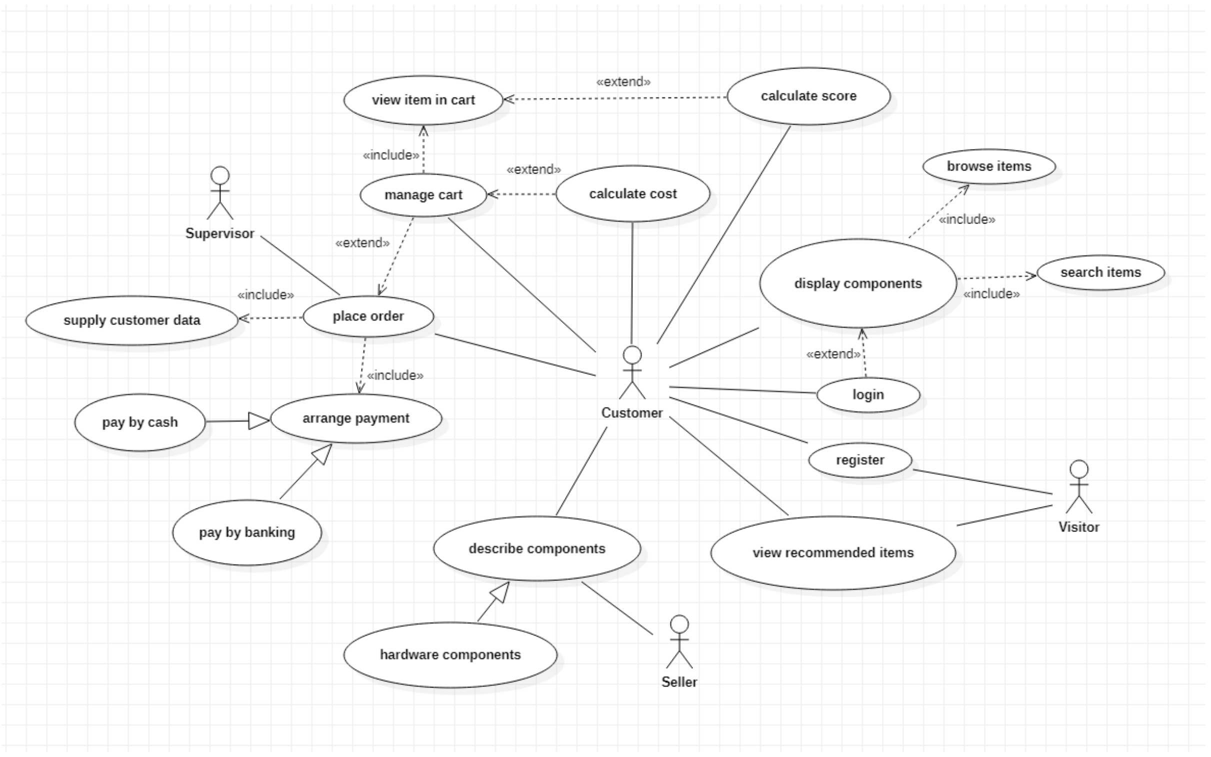
Transform Analysis:-



Transaction Analysis:-



Use Case Diagram (Properly display the include and extend relationship among use cases)



Use Case Description for all the use cases (Use the appropriate Template – for each use case shown in the Use Case Diagram)

Use Case Template

Use Case ID:	UC1001		
Use Case Name:	display components		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	17-10-2021

Actor:	Customer
Description:	The use case begins when the actor intents to view the each components.
Preconditions:	Viewing each component of PC when the actor is logged in
Postconditions:	NA
Priority:	1
Frequency of Use:	High
Normal Course of Events:	1. Actor can browse each component they require.
Alternative Courses:	1. Actor may search the component which they require.
Exceptions:	NA
Includes:	browse items , search items
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1002		
Use Case Name:	calculate score		
Created By:	V. Prathiban	Last Updated By:	V. Prathiban
Date Created:	17-10-2021	Date Last Updated:	17-10-2021

Actor:	Customer
Description:	The use case begins when the actor intents to view the score of the components present in cart.
Preconditions:	NA
Postconditions:	NA
Priority:	1
Frequency of Use:	High
Normal Course of Events:	1. The actor gets the score of a entire PC
Alternative Courses:	1. If the entire score is not displayed , actor may get score of each component.
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1003		
Use Case Name:	view recommended items		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	16-10-2021

Actor:	Customer ,Visitor
Description:	The use case begins when the actor intents to visit the recommended items shown.
Preconditions:	NA
Postconditions:	NA
Priority:	2
Frequency of Use:	High
Normal Course of Events:	1. The actor chooses the recommended item.
Alternative Courses:	1. If there is no recommended item, actor may choose the same component with different brand.
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1004		
Use Case Name:	describe components		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	17-10-2021	Date Last Updated:	17-10-2021

Actor:	Customer ,Seller
Description:	The use case begins when the actor intents to view the each component description.
Preconditions:	NA
Postconditions:	NA
Priority:	1
Frequency of Use:	High
Normal Course of Events:	1. The actor gets brief information about components.
Alternative Courses:	1. If the brief information is not available ,they may get summary of that component.
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1005		
Use Case Name:	calculate cost		
Created By:	V. Prathiban	Last Updated By:	V. Prathiban
Date Created:	17-10-2021	Date Last Updated:	17-10-2021

Actor:	Customer
Description:	The use case begins when the actor intents to know the price of total cost present in the cart.
Preconditions:	NA
Postconditions:	NA
Priority:	2
Frequency of Use:	Medium
Normal Course of Events:	1. The actor knows the price of components present in cart which includes EMI option.
Alternative Courses:	1. If the EMI option and price is not available , actor may get the price of that component present in the offline store.
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1006		
Use Case Name:	Login		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	16-10-2021

Actor:	Customer
Description:	The use case begins when the actor intent to log in to the system.
Preconditions:	NA
Postconditions:	NA
Priority:	1
Frequency of Use:	High
Normal Course of Events:	1. The actor logs in successfully
Alternative Courses:	1. If the log in is not successful, actor may use forgot password
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1007		
Use Case Name:	register		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	16-10-2021

Actor:	Customer
Description:	The use case begins when the actor intent to register to the system.
Preconditions:	NA
Postconditions:	NA
Priority:	1
Frequency of Use:	High
Normal Course of Events:	1. The actor register his/her account successfully.
Alternative Courses:	1. If Registration is failed, the actor may choose other way like signup through facebook.
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	UC1008		
Use Case Name:	manage cart		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	16-10-2021

Actor:	Customer
Description:	The use case begins when the actor intent to add the items in cart.
Preconditions:	Place the order item from the calculate cost use case.
Postconditions:	NA
Priority:	3
Frequency of Use:	Medium
Normal Course of Events:	1. Select the required item in cart
Alternative Courses:	1. If there is no required item use recommended item.
Exceptions:	NA
Includes:	view item in cart
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

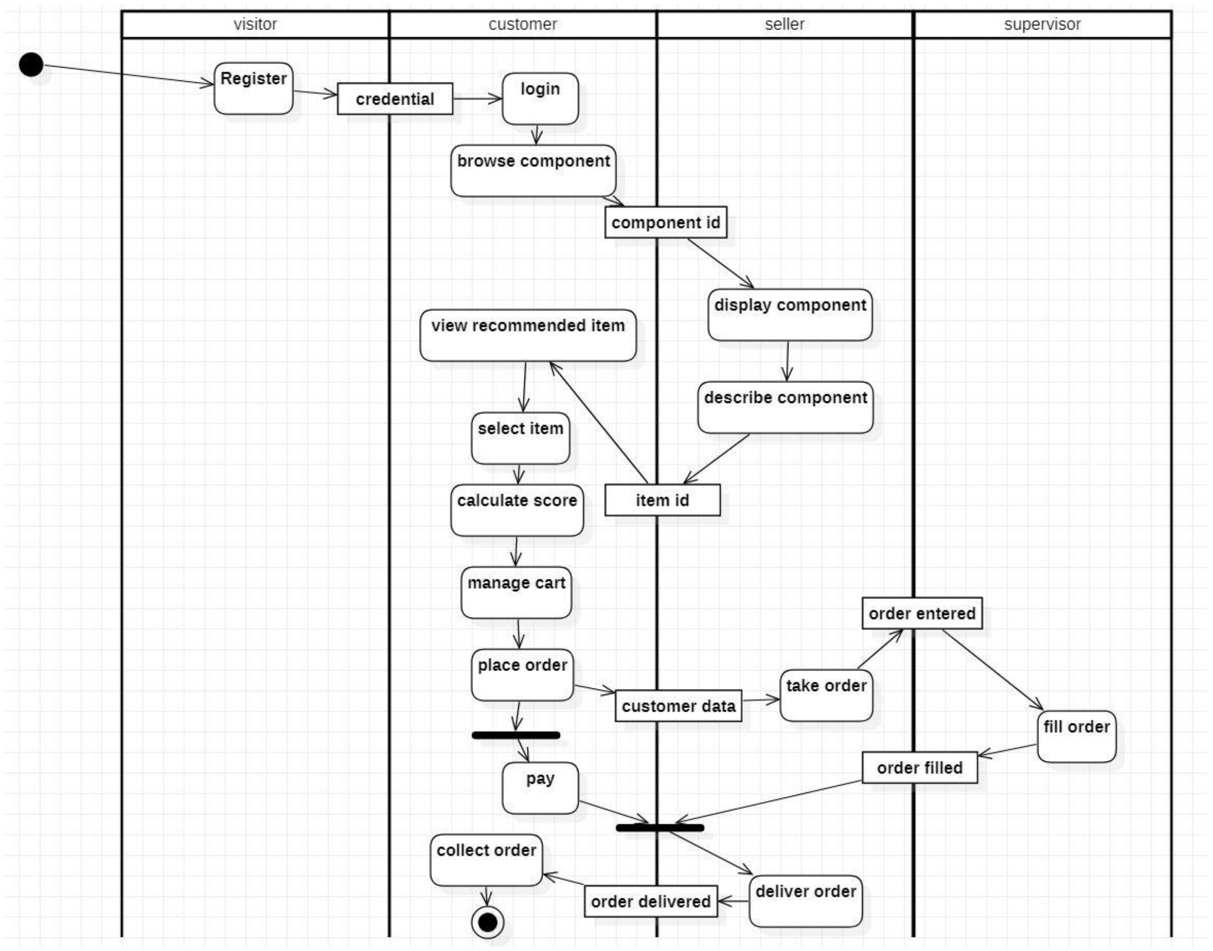
Use Case ID:	UC1009		
Use Case Name:	view item in cart		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	16-10-2021

Actor:	Customer
Description:	The use case begins when the actor intents to view the items in cart.
Preconditions:	View the items in cart using the score value.
Postconditions:	NA
Priority:	3
Frequency of Use:	Medium
Normal Course of Events:	1. The actor may order the item in cart
Alternative Courses:	2. If the actor do not need the item ,actor may choose from similar product
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

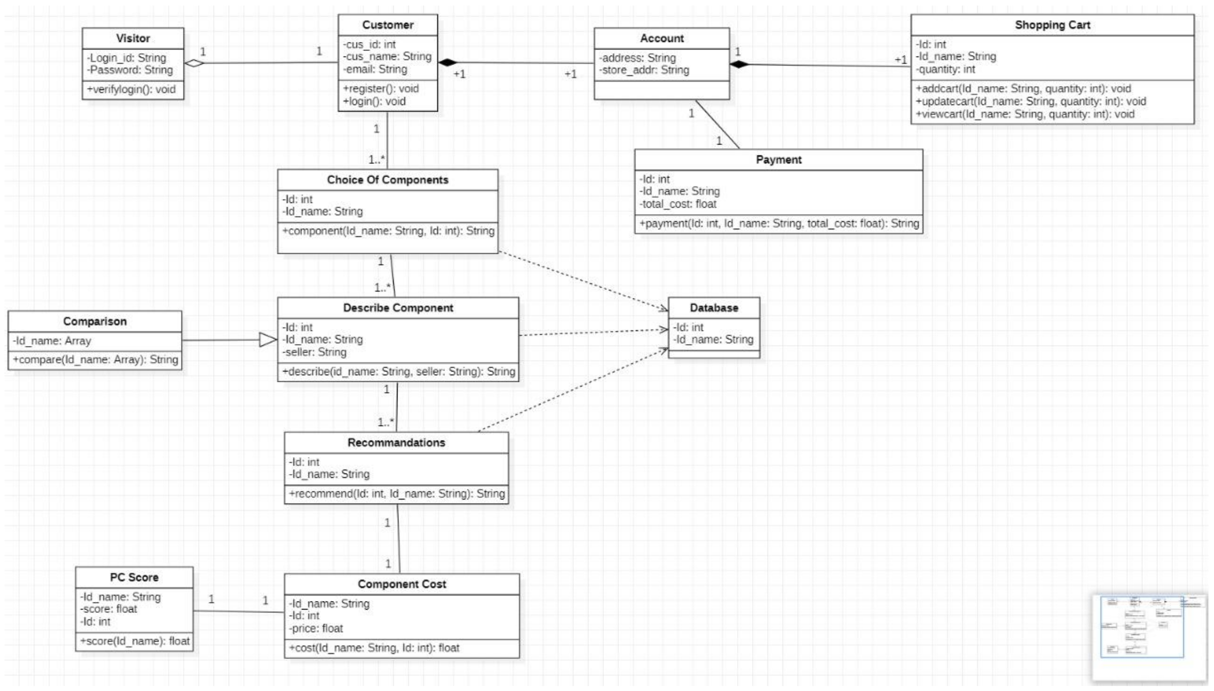
Use Case ID:	UC1010		
Use Case Name:	place order		
Created By:	R.Sri HariHaran	Last Updated By:	R.Sri HariHaran
Date Created:	16-10-2021	Date Last Updated:	16-10-2021

Actor:	Customer ,Supervisor
Description:	The use case begins when the user intents to place the order from cart.
Preconditions:	The items should be in cart to place the order.
Postconditions:	NA
Priority:	4
Frequency of Use:	Medium
Normal Course of Events:	1. Enter the home address for deliverable.
Alternative Courses:	1. If the address is not deliverable, provide the nearby store information.
Exceptions:	NA
Includes:	supply customer data, arrange payment
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Activity Diagram (Use the concept of swim-lane)



Class Diagram (Carefully associate the different classes using proper relationship and display the multiplicity values)



CRC card (for each class shown in the class diagram)

visitor	
• verifiying the user login details	• customer

customer	
• registering the customer and enter the login details by using loginid	• visitor
	• account
	• choice of components

account	
• stores customer's name and address details	• customer
	• shopping cart
	• payment

choice of components	
• selecting the list of components given here	• describe component
	• customer
	• database

describe component	comparison
• description about the components	• choice of component
	• recommandations
	• database

database	
• contains all the details of components and customers	• choice of components • describe component • recommandations

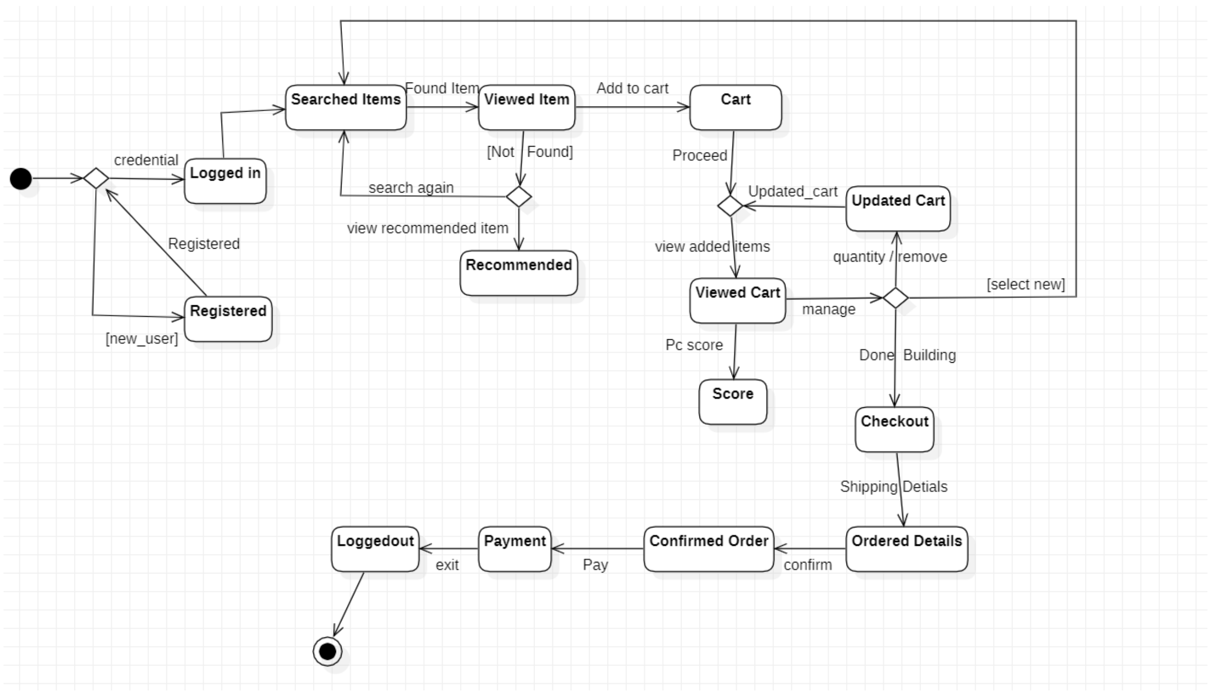
recommandations	
• showing recommanded components to the customer	• describe component • component cost

component cost	
• describe the cost of the component	• recommandations • pc score

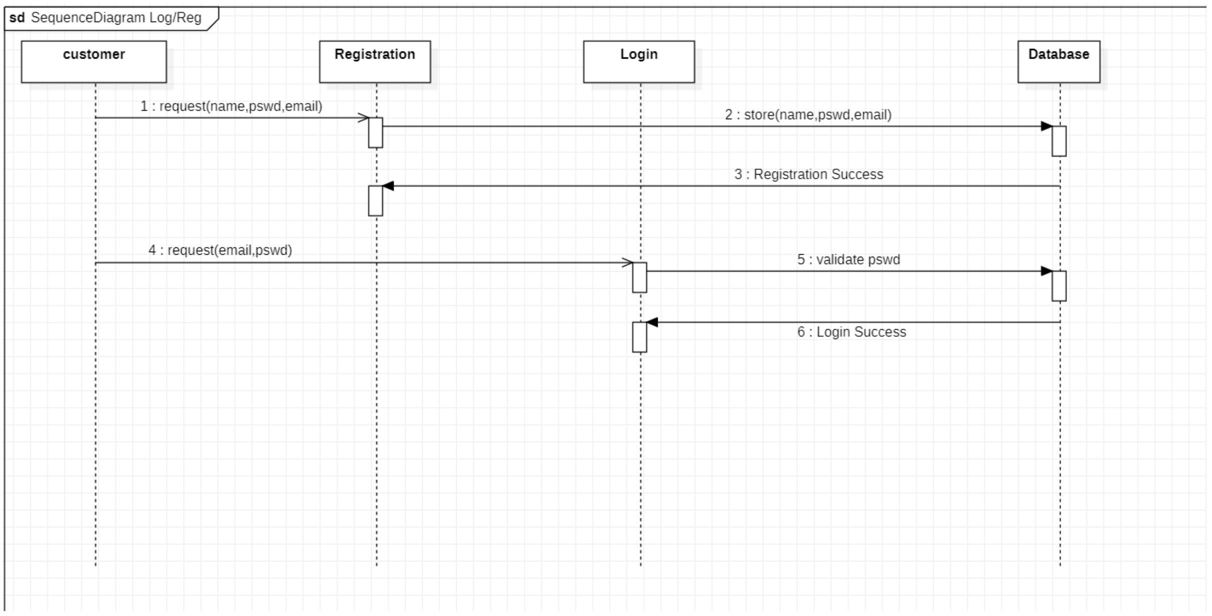
pc score	
• describe the score of the computer	• component cost

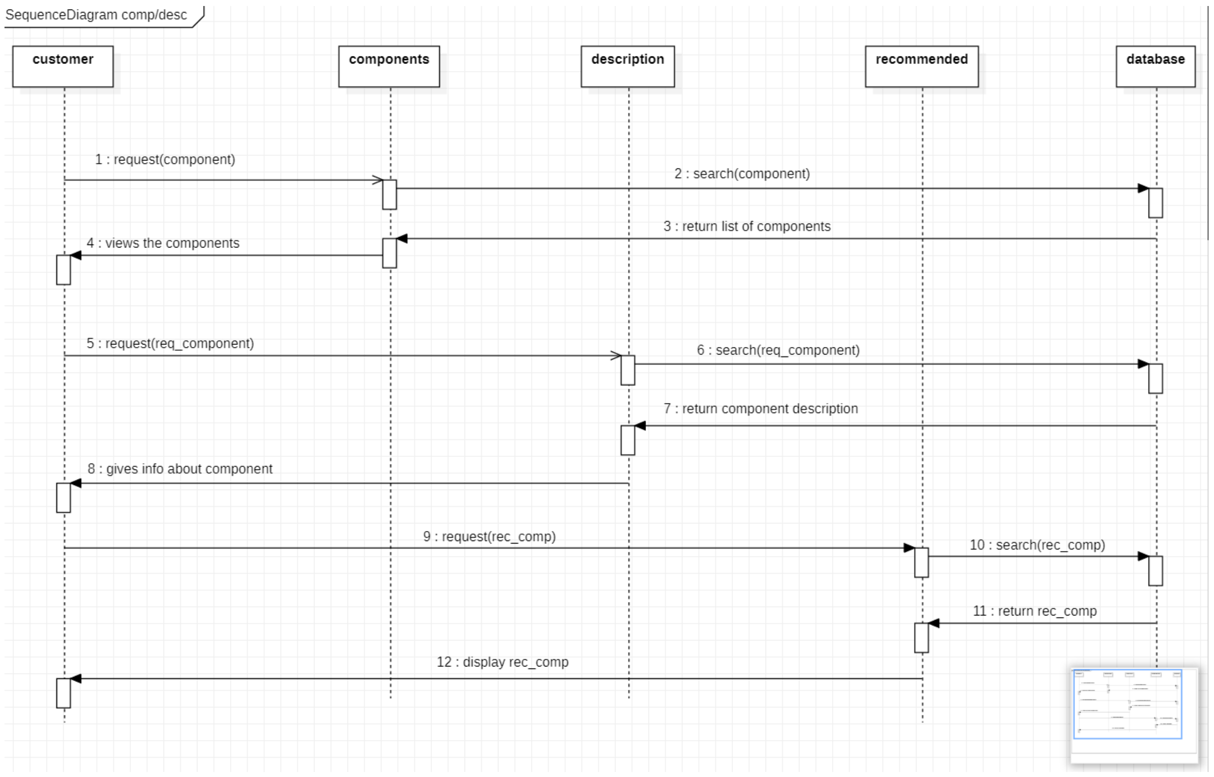
payment	
• accepting the payments from the customers by using id	• account

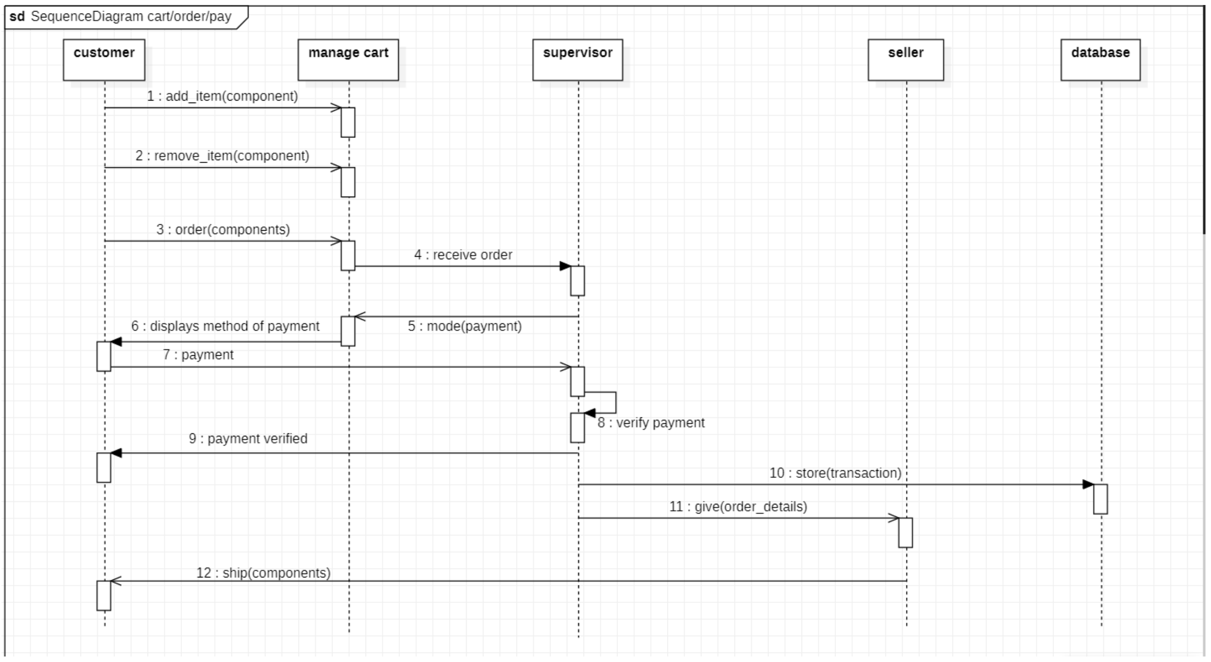
State Chart Diagram (Displaying the all the states and transition among states by firing an event)



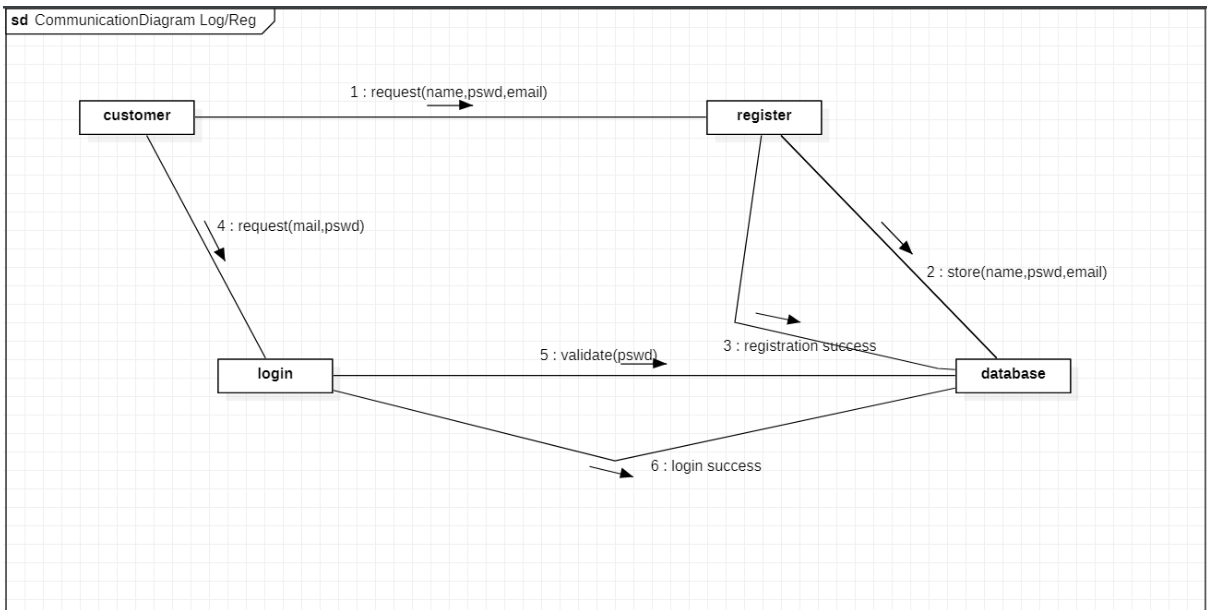
Sequence Diagram (Display the timelines (lifelines) of each objects properly)

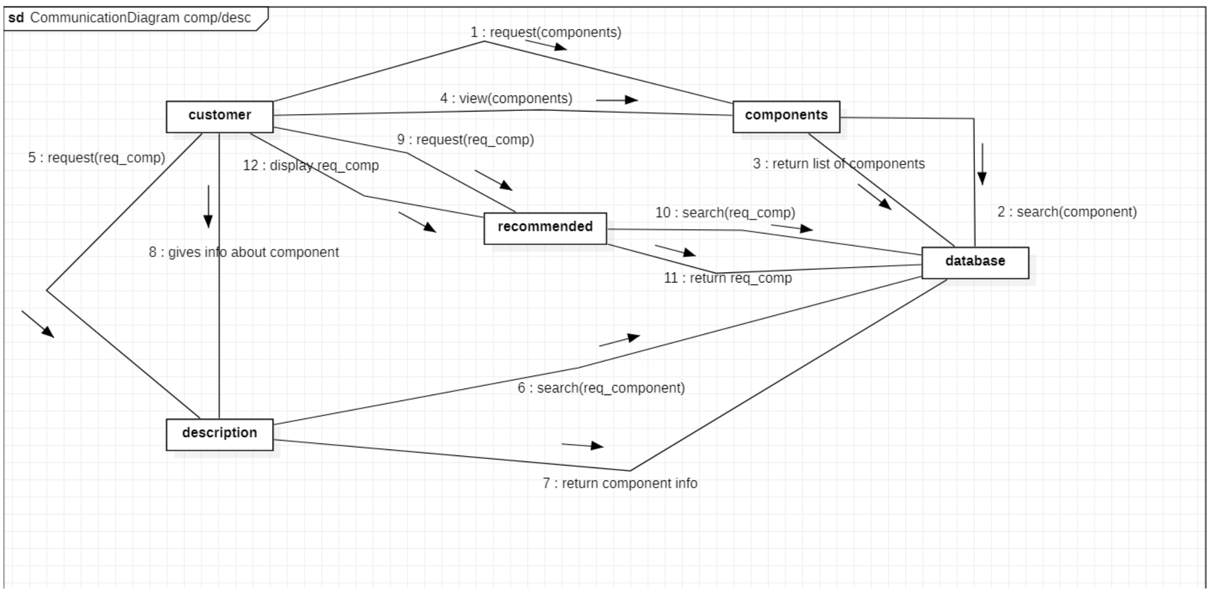


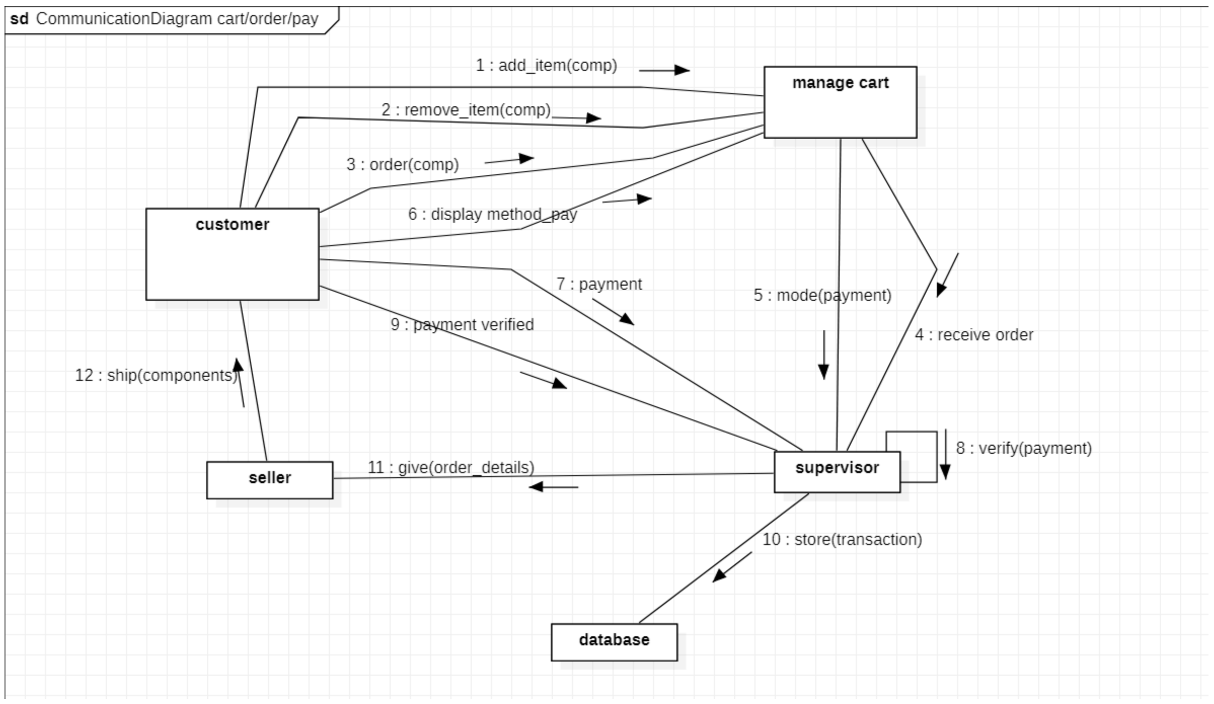

SequenceDiagram comp/desc
customer
components
description
recommended
database
1 : request(component)
2 : search(component)
3 : return list of components
4 : views the components
5 : request(req_component)
6 : search(req_component)
7 : return component description
8 : gives info about component
9 : request(rec_comp)
10 : search(rec_comp)
11 : return rec_comp
12 : display rec_comp


sd SequenceDiagram cart/order/pay
customer
manage cart
supervisor
seller
database
1 : add_item(component)
2 : remove_item(component)
3 : order(components)
4 : receive order
5 : mode(payment)
6 : displays method of payment
7 : payment
8 : verify payment
9 : payment verified
10 : store(transaction)
11 : give(order_details)
12 : ship(components)

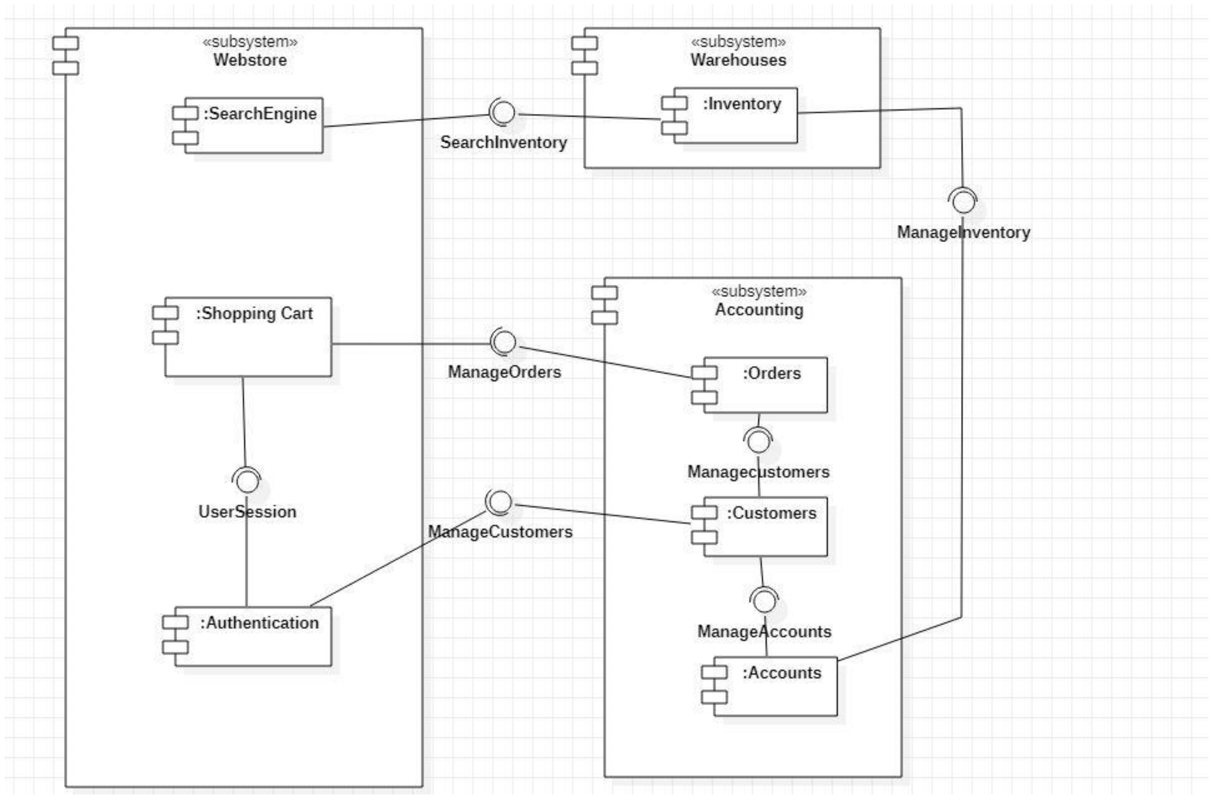
Collaboration / Communication Diagram



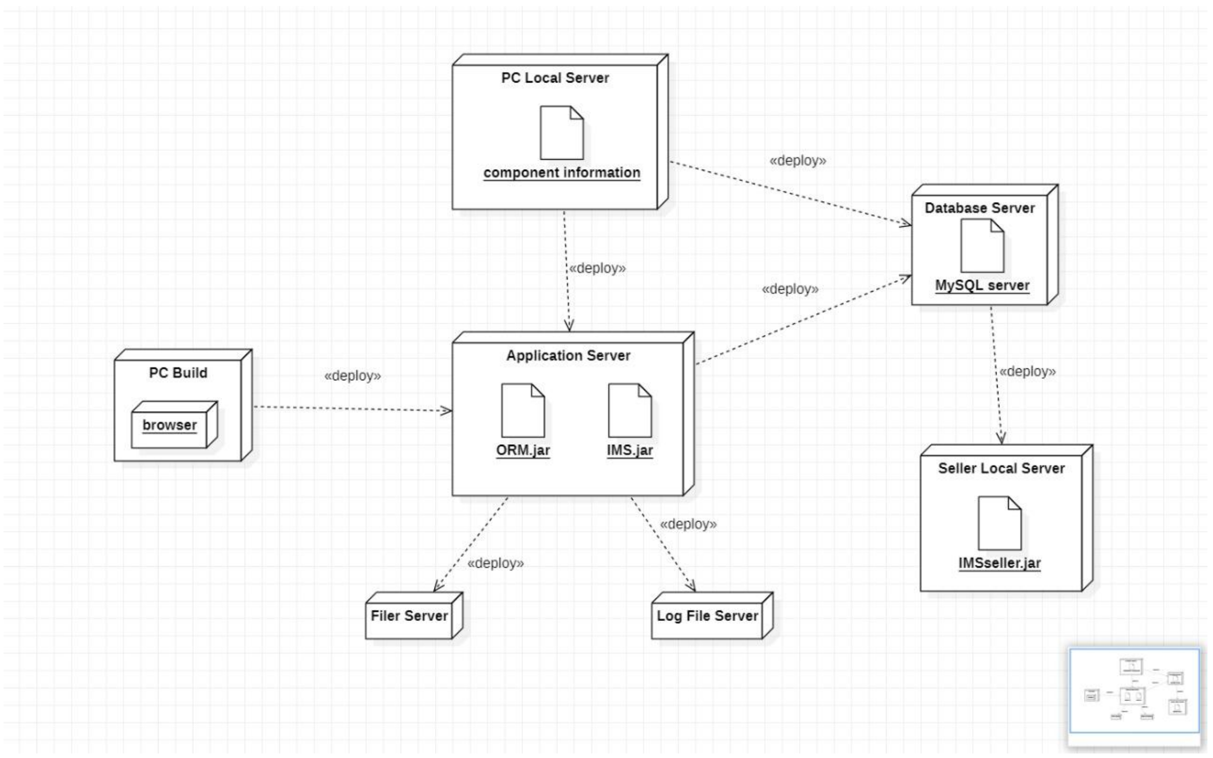




Component Diagram (Join the components through proper interfaces)



Deployment Diagram (Display the relevant Nodes and the interconnection among Nodes as well as artifacts)



Generate the code as per your developed language. (If for your language, the plugins are not available, then generate the code in JAVA). Display your output as File Name and File Content for each file generated.

Acount.java

```
import java.util.*;

/**
 *
 */
public class Account {

   /**
    * Default constructor
    */
   public Account() {
   }

   /**
    *
    */
   private String address;
   /**
    *
    */
   private String store_addr;

   /**
    *
    */
   public Shopping Cart 1;

   /**
    *
    */
   public Customer 1;



   /**
    * @return
    */
   public void Account_updation() {
      // TODO implement here
      return null;
   }}
```

Choiceofcomponents.java

```
import java.util.*;

/**
 *
 */
public class Choice Of Components {

   /**
    * Default constructor
    */
   public Choice Of Components() {
   }

   /**
    *
    */
   private int Id;

   /**
    *
    */
private String Id_name;


/**
 *
 */
private String seller;



/**
 *
 */
public Customer 1..*;



/**
 * @param Id_name
 * @param Id
 * @return
 */
public String component(String Id_name, int Id) {
   // TODO implement here
   return "";
}}
```

Comparison.java

```
import java.util.*;

/**
 *
 */
public class Comparison extends Describe Component {

    /**
     * Default constructor
     */
    public Comparison() {
    }

    /**
     *
     */
    private Array Id_name;

    /**
     * @param Id_name
     * @return
     */
    public String compare(Array Id_name) {
        // TODO implement here
        return "";
    }

}
```

ComponentCost.java

```
import java.util.*;

/**
 *
 */
public class Component Cost {

   /**
    * Default constructor
    */
   public Component Cost() {
   }

   /**
    *
    */
   private String Id_name;

   /**
    *
    */
    private int Id;


    /**
     *
     */
    private float price;




    /**
     * @param Id_name
     * @param Id
     * @return
     */
    public float cost(String Id_name, int Id) {
        // TODO implement here
        return 0.0f;
    }

}
```

Customer.java

```
import java.util.*;

/**
 *
 */
public class Customer {

   /**
    * Default constructor
    */
   public Customer() {
   }

   /**
    *
    */
   private int cus_id;

   /**
    *
    */
```

```
private String cus_name;

/**
 *
 */
private String email;

/**
 *
 */
public Account 1;

/**
 * @return
 */
public void register() {
   // TODO implement here
   return null;
}

/**
 * @return
 */
```

```
    public void login() {
        // TODO implement here
        return null;
    }

}
```

Database.java

```
import java.util.*;

/**
 *
 */
public class Database {

   /**
    * Default constructor
    */
   public Database() {
   }

   /**
    *
    */
   private int Id;

   /**
    *
    */
    private String Id_name;


}
```

DescrieComponent.java

```
import java.util.*;

/**
 *
 */
public class Describe Component {

   /**
    * Default constructor
    */
   public Describe Component() {
   }

   /**
    *
    */
   private int Id;

   /**
    *
```

```
     */
    private String Id_name;

    /**
     *
     */
    private String seller;



    /**
     * @param id_name
     * @param seller
     * @return
     */
    public String describe(String id_name, String seller) {
        // TODO implement here
        return "";
    }

}
```

Payment:

```
import java.util.*;

/**
 *
 */
public class Payment {

    /**
     * Default constructor
     */
    public Payment() {
    }

    /**
     *
     */
    private int Id;

    /**
     *
     */
    private String Id_name;

    /**
     *
     */
    private float total_cost;

    /**
     * @param Id
     * @param Id_name
     * @param total_cost
     * @return
     */
    public String payment(int Id, String Id_name, float total_cost) {
        // TODO implement here
        return "";
    }

}
```

PCScore.java

```
import java.util.*;

/**
 *
 */
public class PC Score {

    /**
     * Default constructor
     */
    public PC Score() {
    }

    /**
     *
     */
    private String Id_name;

    /**
     *
     */
    private float score;


    /**
     *
     */
    private int Id;



    /**
     * @param Id_name
     * @return
     */
    public float score(void Id_name) {
        // TODO implement here
        return 0.0f;
    }

}
```

Recommandations

```
import java.util.*;

/**
 *
 */
public class Recommandations {

    /**
     * Default constructor
     */
    public Recommandations() {
    }

    /**
     *
     */
    private int Id;

    /**
     *
     */
    private String Id_name;

    /**
     * @param Id
     * @param Id_name
     * @return
     */
    public String recommend(int Id, String Id_name) {
        // TODO implement here
        return "";
    }

}
```

ShoppingCart.java

```
import java.util.*;

/**
 *
 */
public class Shopping Cart {

   /**
    * Default constructor
    */
   public Shopping Cart() {
   }

   /**
    *
    */
   private int Id;

   /**
    *
    */
```

```
private String Id_name;

/**
 *
 */
private int quantity;
/**
 * @param Id_name
 * @param quantity
 * @return
 */
public void addcart(String Id_name, int quantity) {
   // TODO implement here
   return null;
}

/**
 * @param Id_name
 * @param quantity
 * @return
 */
public void updatecart(String Id_name, int quantity) {
   // TODO implement here
```

```
        return null;
    }

    /**
     * @param Id_name
     * @param quantity
     * @return
     */
    public void viewcart(String Id_name, int quantity) {
        // TODO implement here
        return null;
    }

}
```

__Visitor.java__

```
import java.util.*;

/**
 *
 */
public class Visitor {

   /**
    * Default constructor
    */
   public Visitor() {
   }

   /**
    *
    */
   private String Login_id;

   /**
    *
    */
    private String Password;

    /**
     * @return
     */
    public void verifylogin() {
        // TODO implement here
        return null;
    }

}
```