UML MODELLING FOR VOGUE E-COMMERCE SYSTEM

Professor: Dr. Silvana Faja

C661

CIS-5661

**Sai Teja Valluru**

**Sathwik Reddy Chennu**

Team Members:

Table of Contents

Page No.

1. SYSTEM DESCRIPTION--------------------------------------------------2

2. REQUIREMENTS DETERMINATION---------------------------------2

3.ANALYSIS -------------------------------------------------------------------8

4. DESIGN ---------------------------------------------------------------------15

5. IMPLEMENTATION------------------------------------------------------18

6. REFLECTION ON PATTERN ADOPTION---------------------------19

7. CONCLUSION--------------------------------------------------------------21

8. APPENDICES---------------------------------------------------------------22

1. **System description:**

The Vogue android system is the e-commerce app which contains all the latest fashion designs for MEN and WOMEN. The system shall allow customer to register and login into the application. Only registered users will be able to shop. The system shall allow the user to select the product and make their selection of their choice like size and add to the cart. If the user wishes to change the selection of the product, the user can remove the product from the cart. Price in the cart is changed every time the user adds or removes the product in the cart. The system shall allow the user to select multiple quantity of the same product. When the user clicks on the payment, it automatically redirects to the payment system. Once the payment is done the user receives a confirmation number of the order.

The project is based on the Object-oriented modeling through UML. The project would include Use case diagram, Class diagrams, Interaction diagrams, Activity diagrams and Pattern adoption

**2.Requirements Determination:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Details** | **Type** | **Priority** |
| F1. | The Vogue E-commerce System shall allow the customer to register into the application. | Functional | Must have |
| F2. | The Vogue E-commerce System shall allow the customer to login into the application | Functional | Must have |
| F3. | Vogue E-commerce System shall display the shopping page to the customer**.** | Functional | Must have |
| F4. | The Vogue E-commerce System shall allow the customer to select their required product. | Functional | Must have |
| F5. | The Vogue E-commerce System shall allow the user to add products to their cart. | Functional | Must have |
| F6. | The Vogue E-commerce System shall allow the customer to change or remove the product. | Functional | Must have |
| F7**.** | The Vogue E-commerce System shall allow the user to select multiple quantities of the same product. | Functional | Must have |
| F8. | The Vogue E-commerce System shall ask for reviewing the order from the customer before the payment. | Functional | Must  Have |
| F9. | The Vogue E-commerce System shall re-direct customer to the payment system after checkout. | Functional | Must have |
| F10. | The Vogue E-commerce System shall generate an order confirmation number after the payment. | Functional | Must have |
| F11. | The Vogue E-commerce System shall allow customers to track their product details. | Functional | Should Have |

|  |  |  |  |
| --- | --- | --- | --- |
| N2. | The Vogue E-commerce System shall compatible with various platforms. | Non-Functional | Must have |
| N3. | The Vogue E-commerce System shall provide good user interface. | Non-Functional | Could have |
| N4. | The Vogue E-commerce System shall have strong security in their payment processing. | Non-Functional | Must have |
| N5. | The Vogue E-commerce System shall handle privacy of the customer data. | Non-Functional | Must have |

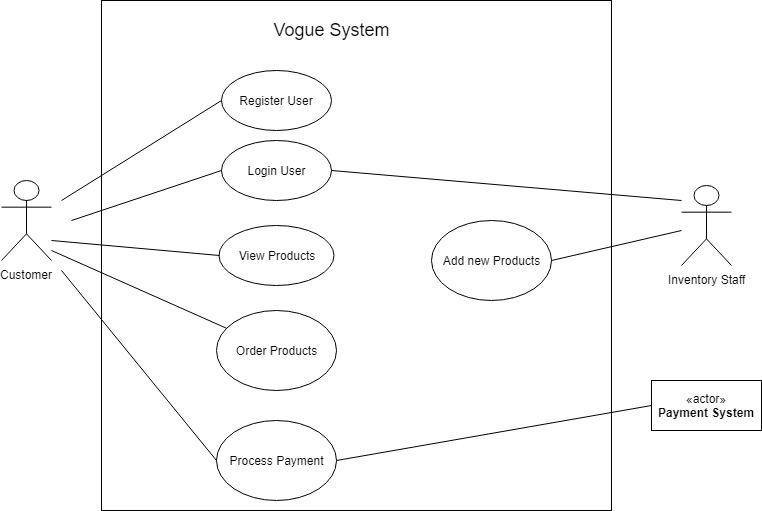
**Requirements gathering methods:**

**Brainstorming:** Gathering of the required methods for getting all possible ideas for the problems to enhance the details in clarity.

**Focus Group:** The Feedback is gathered from the group of people who are related to the system. These people are users of the system.

**Prototyping:** Using the prototype of the preliminary requirements of the initial version of the system helps to make the additional requirements.

**Use case diagram:**



**Use Case Specifications:**

|  |
| --- |
| **Use Case: Register User** |
| **ID:** 1 |
| **Brief Description:**  Customer creates account to access into application and buy products |
| **Primary Actor:**  Customer |
| **Preconditions:**  1. Customer is interested to buy products from the application. |
| **Main flow:**  1. Customer provide details  2. System validates customer details  3. System Creates new account |
| **Postconditions:**  1. A new user has been Registered. |
| **Alternative flows:**  2. Given Invalid user details or empty fields  2.a System notes invalid fields  2.b Returns to Step one |

|  |
| --- |
| **Use Case: Login User** |
| **ID:** 2 |
| **Brief Description:**  User Logins into System to view products. |
| **Primary Actor:**  Customer  Inventory Staff |
| **Preconditions:**  1. User is already registered with the system. |
| **Main flow:**  1. User enters Login credentials.  2. System Validates customer details.  3. System Authenticate User  4. System permits User into the system |
| **Postconditions:**  1. User logged into the System. |
| **Alternative flows:**  2. Given Invalid user details or empty fields  2.a System notes invalid fields  2.b Returns to Step one |

|  |
| --- |
| **Use Case: View Products** |
| **ID:** 3 |
| **Brief Description:**  Customer select the product and view it. |
| **Primary Actor:**  Customer |
| **Preconditions:**  1. Customer logged into the system. |
| **Main flow:**  1. Customer search for the product.  2. System displays list of products.  3. Customer selects the product.  4. For each item that is out of stock  4.1 The system informs the Customer that the item is currently unavailable.  5. Customer view product. |
| **Postconditions:**  1. Product is viewed |
| **Alternative flows:**  2. Customer not entered in search field  2.a System notes invalid fields  2.b Returns to Step one |

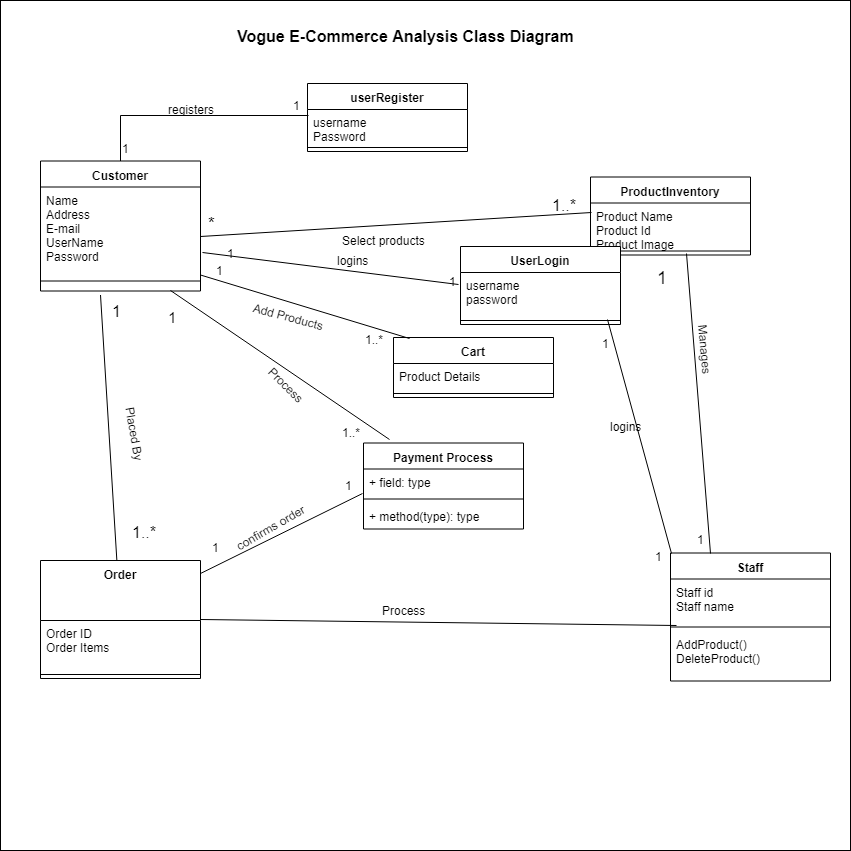
|  |
| --- |
| **Use Case: Order Products** |
| **ID:** 4 |
| **Brief Description:**  Customer selects the products and orders it. |
| **Primary Actor:**  Customer |
| **Preconditions:**   1. Customer logged into the system 2. Customer selected the product. |
| **Main flow:**  1. System calculates the total product price.  2. System displays Total Price to customer.  3. Customer previews the order.  4. Customer confirms the order.  5. System redirect to payment activity.  5.1 On successful Payment  5.2 Order is Placed |
| **Postconditions:**  1. The customer’s order is placed. |

|  |
| --- |
| **Use Case: Process Payment** |
| **ID:** 5 |
| **Brief Description:**  Customer selects the products and orders it. |
| **Primary Actor:**  Customer  Payment System |
| **Preconditions:**   1. Customer logged into the system 2. Customer placed an order |
| **Main flow:**  1. System asks customer to enter payment details.  2. Customer enters the payment details.  3. System validates the details.  4. Payment System authenticates the customer payment details.  5. If the payment is success.  5.1 System confirms the customer’s order.  6. If transaction got cancelled.  6.1 System redirects customer to payment page. |
| **Postconditions:**  1. The customer’s order is Confirmed. |
| **Alternative Flow:**  2. Given Invalid card details or empty fields  2.a System notes invalid fields  2.b Returns to Step one |

|  |
| --- |
| **Use Case: Add Products** |
| **ID:** 6 |
| **Brief Description:**  Inventory Staff add products into the system |
| **Primary Actor:**  Inventory staff |
| Secondary Actor: |
| **Preconditions:**   1. Inventory staff logged into the system |
| **Main flow:**  1. Inventory staff select add product.  2. System ask to enter the product details  3. Inventory staff enters the new product details.  4. Inventory staff confirms the new product.  5. System adds new product. |
| **Postconditions:**  1. The new product is added to inventory. |
| **Alternative Flow:**  3. Given Invalid product details or empty fields  3.a System notes invalid fields  3.b Returns to Step one. |

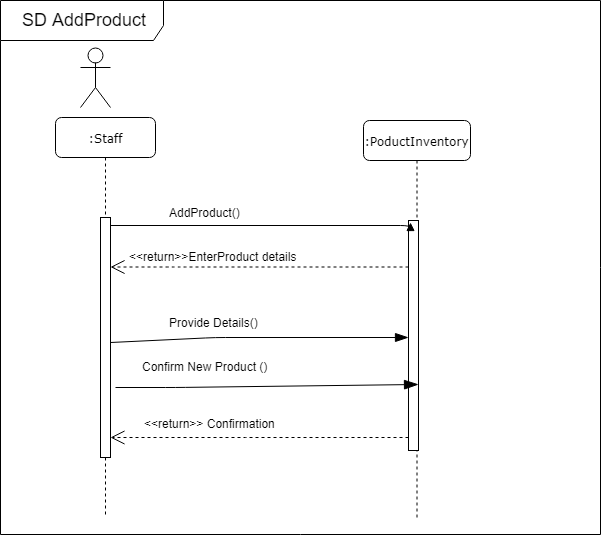
**3.ANALYSIS:**

**Analysis Class diagram:**

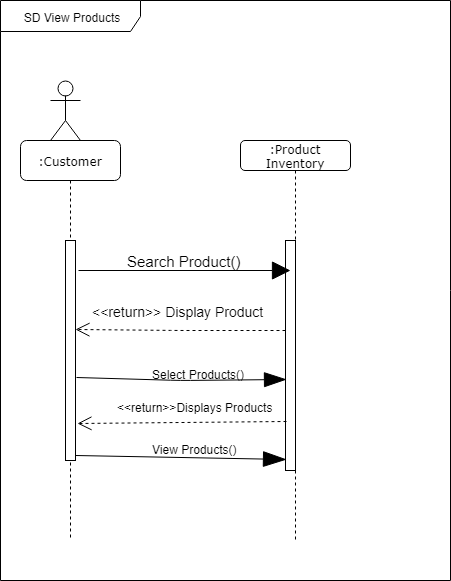


**Sequence Diagram:**

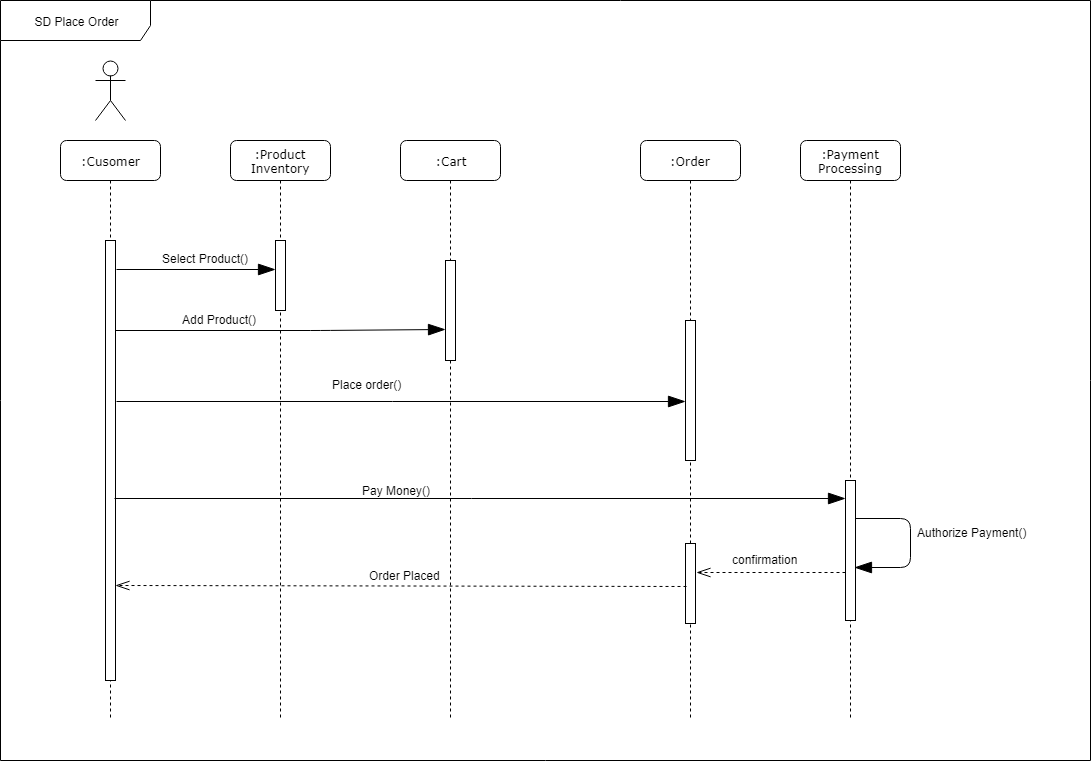
1. **Add Product**



1. **View Product**

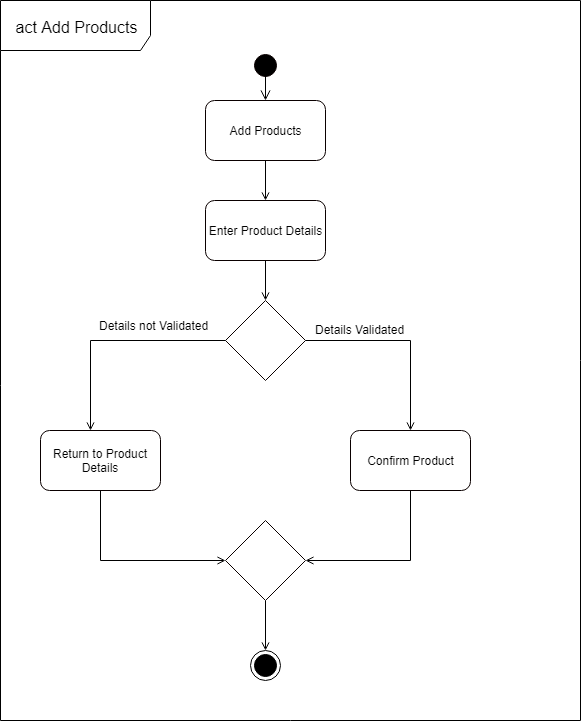


1. **Place order**

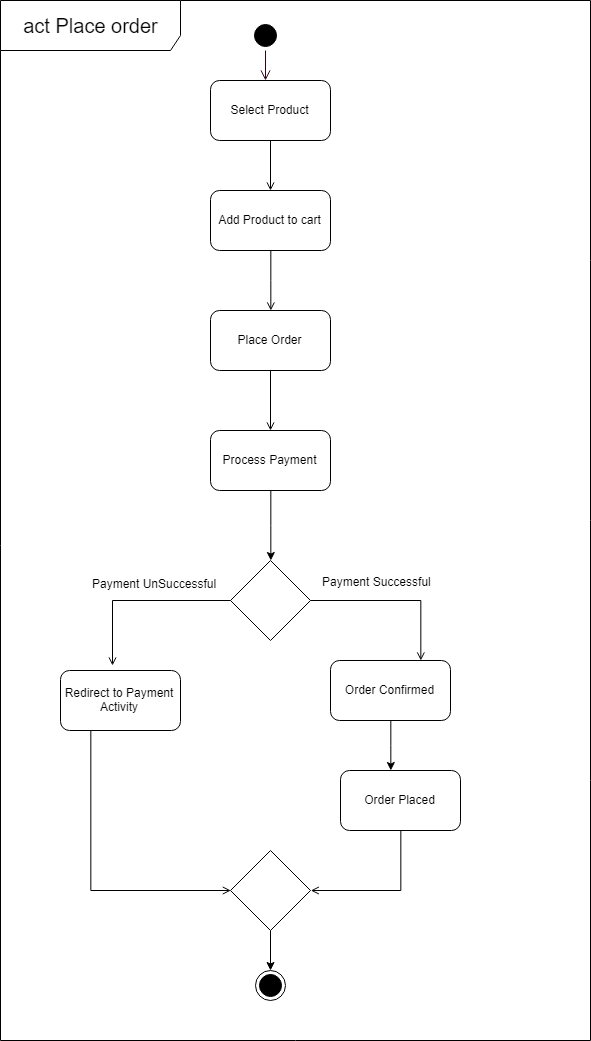


**Activity Diagram:**

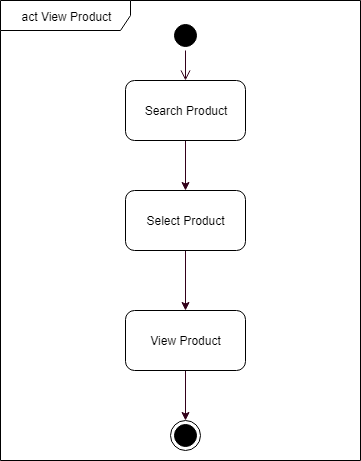
1. **ADD Products**



1. **Place Order**

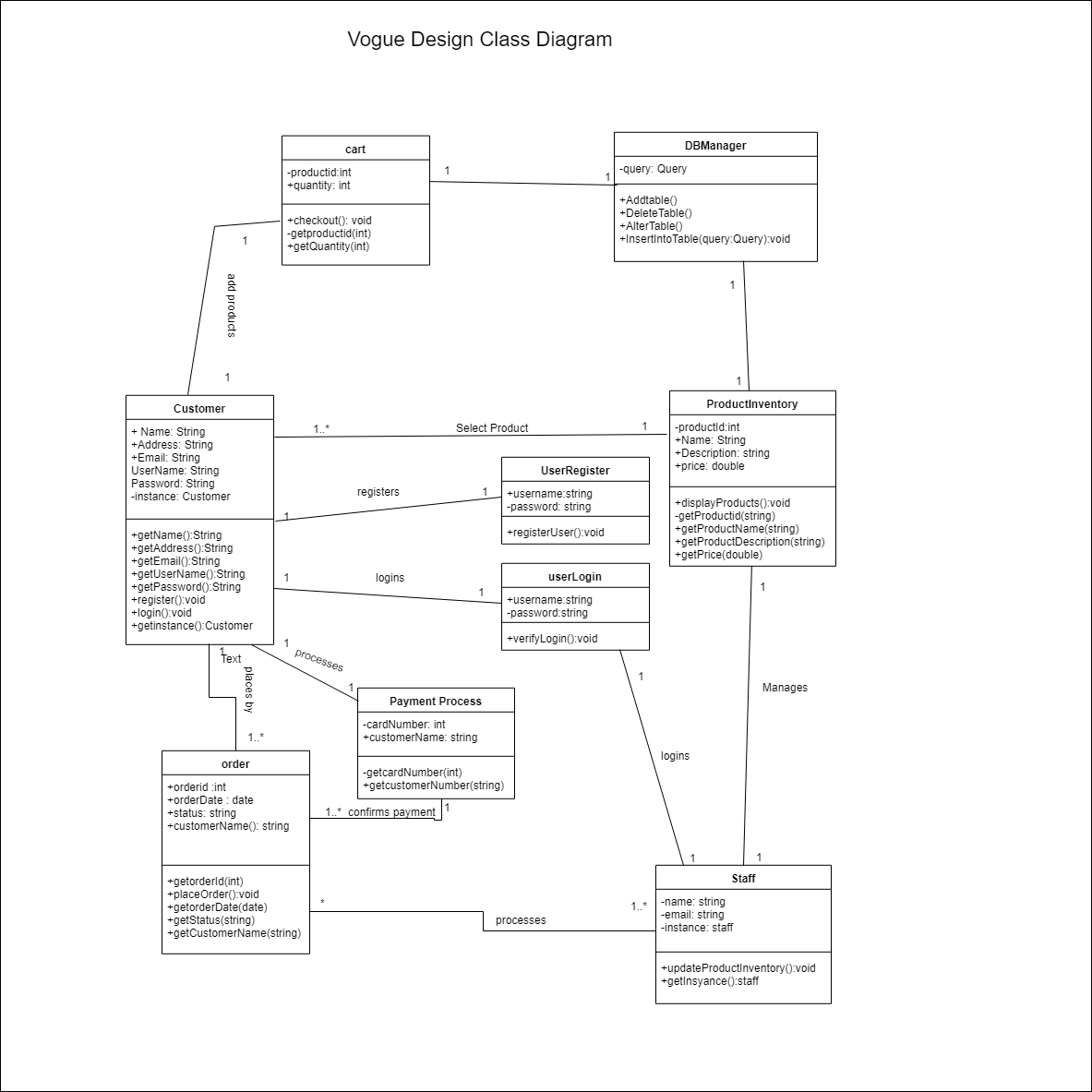


1. **View Product**



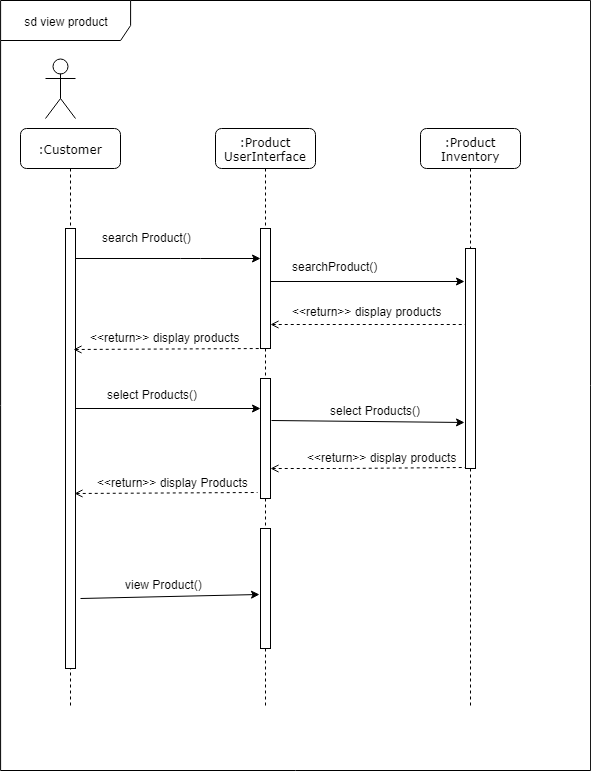
1. **Design:**

**Design Class diagrams:**

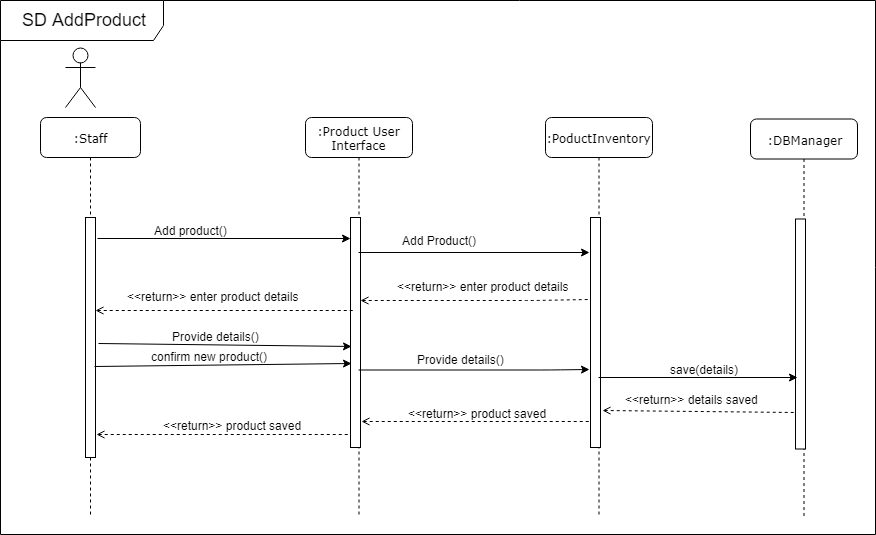


**Sequence Diagrams:**

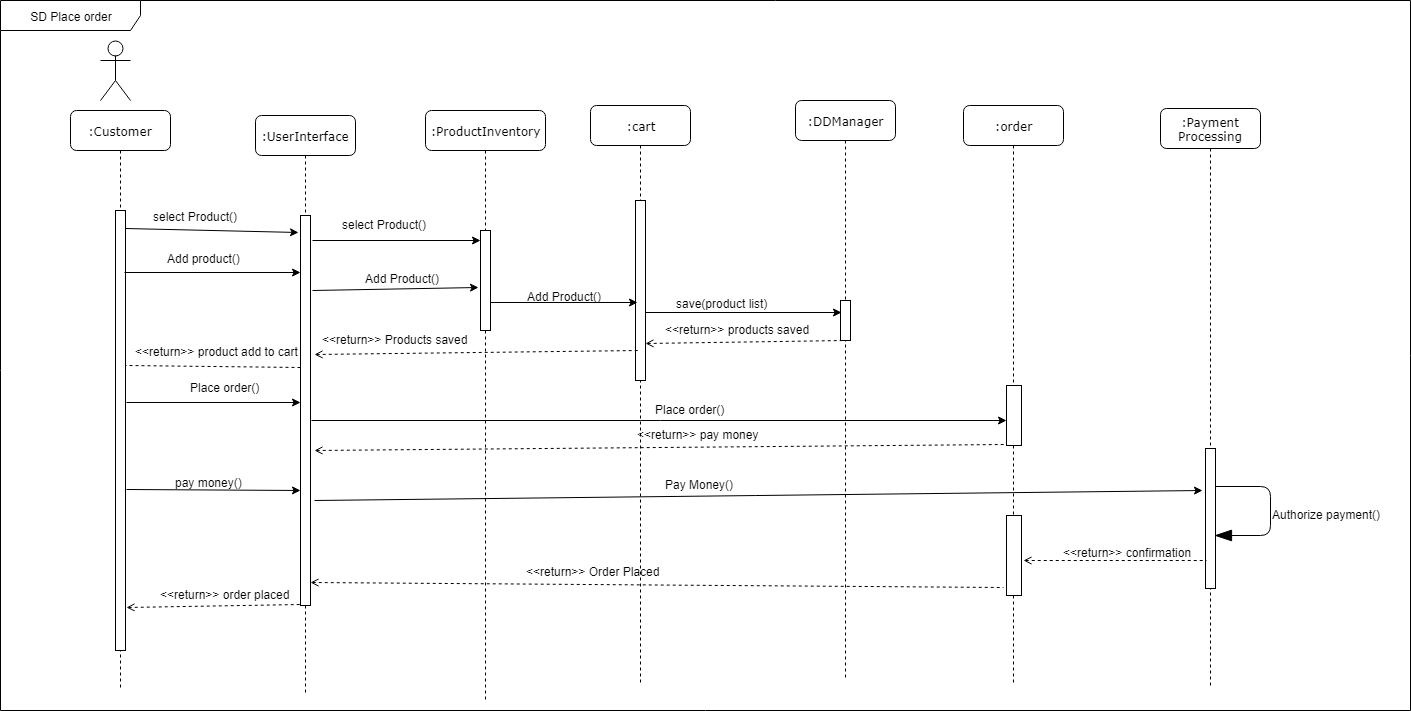
1. **View Products**



1. **Add Product**



1. **Place order**



1. **Implementation:**

**Use case:** Register User

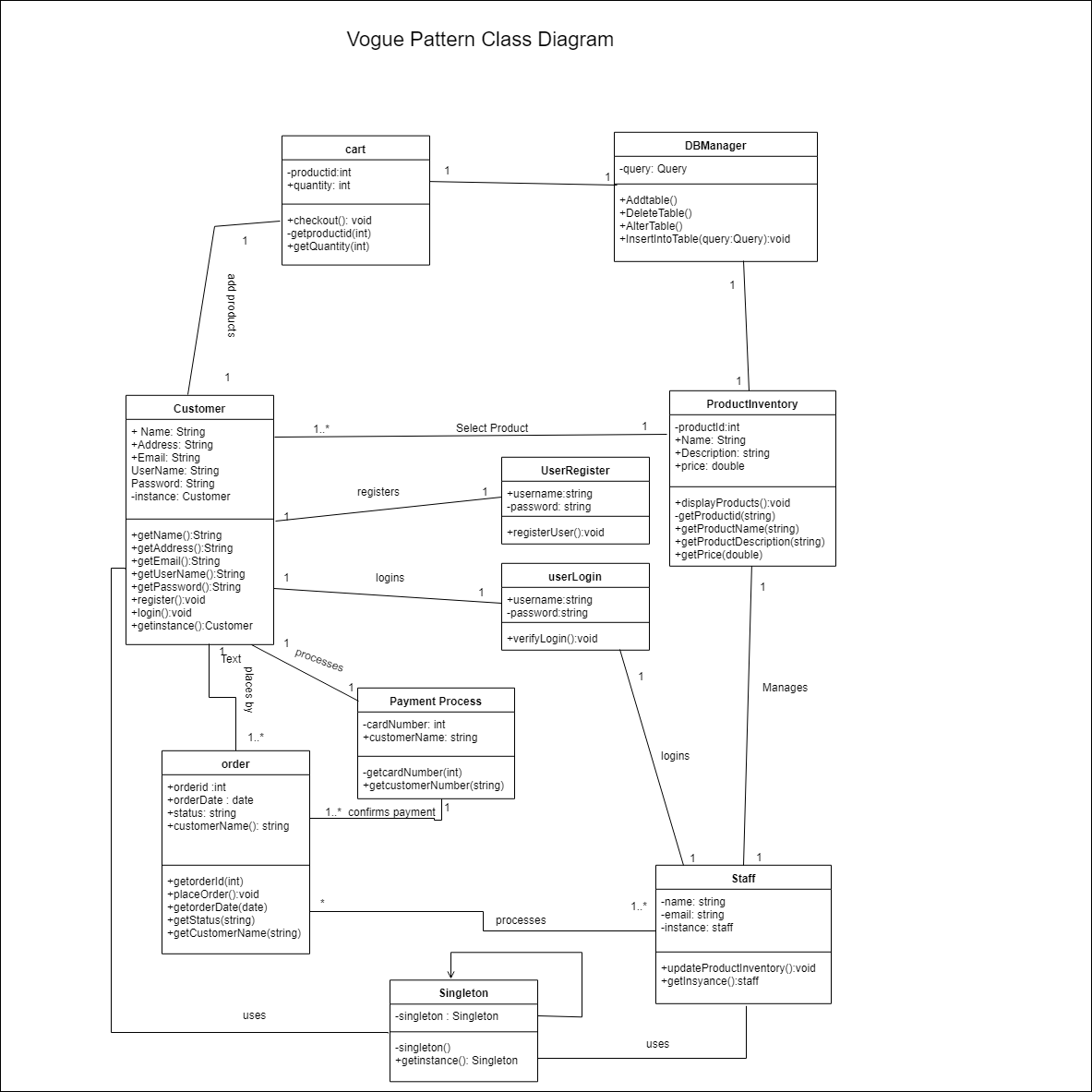
Code to implement

**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.text.TextUtils;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.tasks.OnCompleteListener;  
**import** com.google.android.gms.tasks.Task;  
**import** com.google.firebase.auth.AuthResult;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.auth.FirebaseAuthException;  
  
**public class** RegisterActivity **extends** AppCompatActivity {  
**private** Button **creatbtn**;  
**private** EditText **useremail**,**userpassword**;  
  
 **private** FirebaseAuth **firebaseAuth**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_register***);  
  
 **creatbtn** = (Button) findViewById(R.id.***register\_btn***);  
  
 **useremail** = (EditText) findViewById(R.id.***regemail***);  
 **userpassword** = (EditText) findViewById(R.id.***regpass***);  
 **creatbtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 userRegister();  
 }  
  
 });  
 **firebaseAuth**=FirebaseAuth.*getInstance*();  
  
  
 }  
  
 **public void** userRegister()  
 {  
  
 String email= **useremail**.getText().toString().trim();  
 String pass = **userpassword**.getText().toString().trim();  
  
  
 **if** (TextUtils.*isEmpty*(email))  
 {  
 Toast.*makeText*(getApplicationContext(),**"Enter email"**,Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 **if** (TextUtils.*isEmpty*(pass))  
 {  
 Toast.*makeText*(getApplicationContext(),**"Enter password"**,Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
  
  
 **firebaseAuth**.createUserWithEmailAndPassword(email,pass).addOnCompleteListener(**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task)  
 {  
  
 **if**(task.isSuccessful())  
 {  
 finish();  
 startActivity(**new** Intent(RegisterActivity.**this**,LoginActivity.**class**));  
 Toast.*makeText*(getApplicationContext(),**"Registration Successful"**,Toast.***LENGTH\_SHORT***).show();  
  
 }  
 **else** {  
 Toast.*makeText*(getApplicationContext(),**"Registration UNSuccessful"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
  
 }  
  
  
  
}

1. **Reflection on pattern adoption:**

We are using many instances of a single class in the process of placing order by Customer and adding items into the product inventory by staff Instead of using multiple instances which result in loss of heap memory, Using of Singleton Pattern ensures that there is only one instance of a class created in the Java virtual machine.

**Pattern Class Diagram:**



1. **Conclusion**

The E-Commerce Shopping application is easy to implement in multiple platforms by using all the above diagrams. This application can be accessed from any the global locations with the internet connection. The personal account can be handled by the firebase database which has the strong security features for protecting the user data. All the available products are altered by the staff members, these updates can be view by the customers.

**Future Enhancements**

We can concentrate on the future enhancement by implementing the new requirements like the system itself suggesting products to customer’s desire, Customers sharing the products with friends and family and customers able to utilize the gift cards and promo codes at the time of payment.

**Lessons learned**

With this project, we learned how to design the good system from collecting the requirements by using the correct requirement gathering methods. We also learned how to design the simplest USE cases and Classes and implementing the sequence diagram and activity diagrams from them. We understood that how important the use of patterns in the real time project as it simplifies whole project for the developers.

**Recommendations**

* It is good if have got some more assignments to cover all the diagrams. This would help us to implement more from our side too.
* All the topics that are covered are clear but some topics like we ourselves differentiating actual difference between analysis and design diagrams, but we found it by reading from the course materials that you provided.
* Over all we are well satisfied with the subject.

1. **Appendices:**

**Customer**

Customer is one of the users of the system which uses the e-commerce application to buy products and track status.

**Database**

Database stores the login credentials and used at the time of authentication and this is also used for storing the data of the product inventory.

**Staff**

Staff is one of the users of the system who processes the customer’s orders and manages the product inventory.

**Cart**

Customer adds the products to the cart which the customer likes. These cart data is linked to the database of the system.

**Product Inventory**

Product inventory is accessed by the customer and managed by the staff.

**Payment Process**

This takes the customer’s card details and authenticates. The order is confirmed by successful authentication.