BDSE – WFS – Web Development Using Platforms

Project Report

Project Title	Design and Develop a website portal using Liferay
Qualification Name (NICF)	Advanced Certificate in Web Development using Platforms
Product Name	Triple-A (AAA) Company
Module Name (NICF)	Web Development using Platforms

Student name	**Assessor name**
Francis Roel L. Abarca	

Date issued	**Completion date**	**Submitted on**
March 9,2023	March 31, 2023	April 13, 2023

Project Title	**Design and Develop a website portal using Liferay**

Learner declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Student signature: [illegible] Date: 4/13/2023

Content

Project Background

Scenario of the Project:

The company, AAA, is widely known to be the best web hosts solutions company located in the Philippines. They provide web services such as domain name registration, Shared Hosting, Reseller Hosting, Cloud Hosting, VPS Hosting, Dedicated Hosting and Colocation services.

Purpose of this Project:

This Project is used for Summative Assessment of student in the Module 'Web Development using Platforms' of the NICF Course "Applied Degree in Software Engineering".

This Project considers the skills required to Design, Implement, Test & Document a website for a Used Car Sales Portal using Spring Framework, MySQL server and test the system by adopting Risk Based Testing (RBT).

Project Objectives

Functional Requirements:

The AAA website consists of the following Key pages:

1. Home Page
2. Registration Page
3. Login Page
4. Our Services Page
    a) Domain Name Service Page
    b) Shared Hosting Service Page
    c) Reseller Hosting Service Page
    d) Cloud Hosting Service Page
    e) VPS Hosting Service Page
    f) Dedicated Hosting Service Page
    g) Colocation Service Page

5. Contact Us Page
6. About Us Page
7. Terms and Conditions Page

Customers can purchase their services from AAA's branch offices and needs to provide personal information such as name, email, address, national identity card number and contact number.

Customers' data is a very important factor for AAA Company. To maintain and manage all of their customer information, AAA has been decided to develop an application with an object-oriented approach.

An application is required to keep track of their customer's info and services given to customers. The well-organized customer data can help the company to select the correct recipients for promotions and new services.

The Scope of the Project is to build a Liferay Framework website with customer's data management application.

The overview of the project is as below. There are two types of users in this portal. They are:

1. Administrator
2. Site Member (Staff)

Administrator should be able to perform following functions in the portal:

1. Manage all of the site contents and pages.
2. Update the theme and layout.
3. Manage the customer data portlet and control permission.
4. Manage all users (Site Member and Customer) roles and permission.

Site Member should be able to perform following functions in the portal:

1. Update AAA company site contents.
2. Add and Update the AAA's customer data.

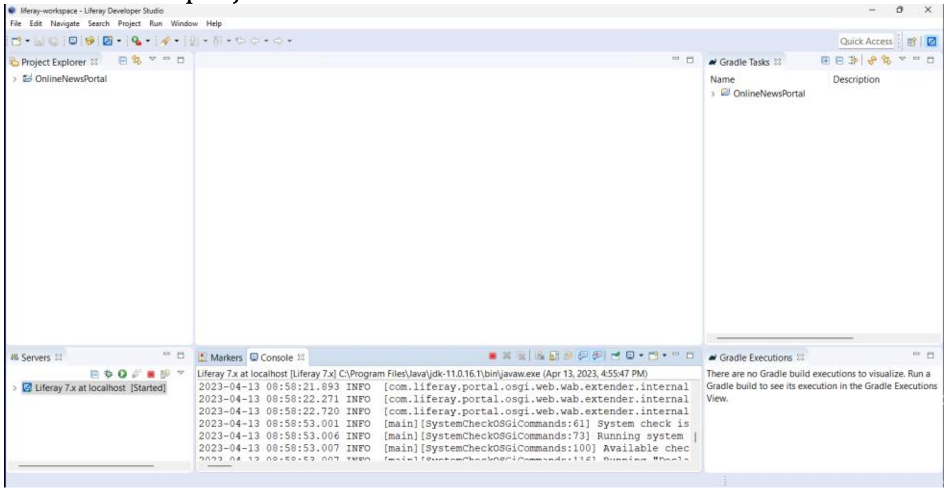
Project Outcomes and Deliverables:

We should be able to perform all the tasks in the Project Task List and prepare the following during the project:

- Implement the project on the project technical environment.
- Prepare a Project Report as per pre-defined template.
- Prepare a Project Presentation as per pre-defined template.
- Prepare Website Documentation.

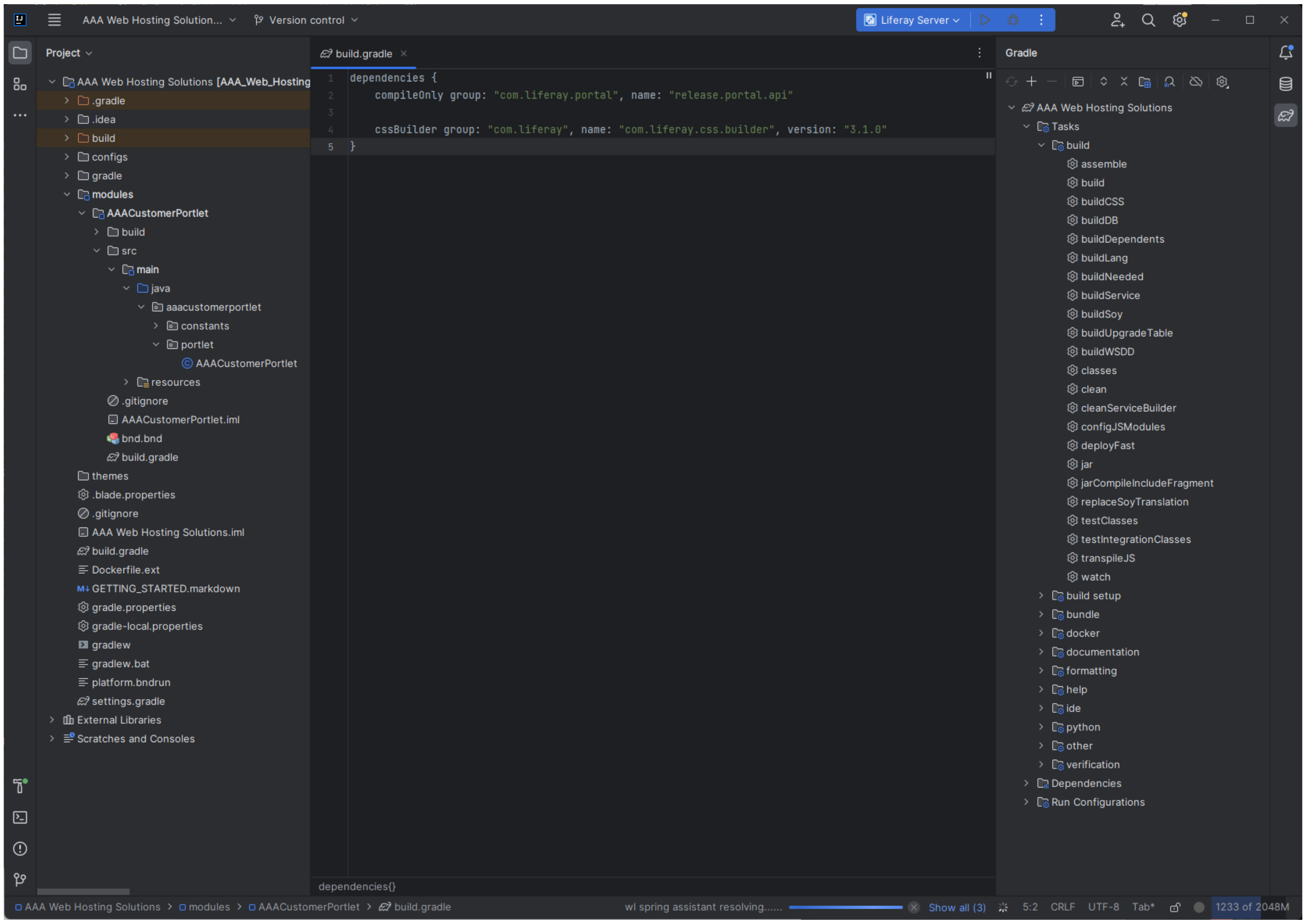
Tools and Platforms Used

Liferay Developer Studio – For hosting Liferay and creating the portlets necessary for the AAA website project.


liferay-workspace - Liferay Developer Studio
File Edit Navigate Search Project Run Window Help
Quick Access
Project Explorer
OnlineNewsPortal
Gradle Tasks
Name
Description
OnlineNewsPortal
Servers
Liferay 7.x at localhost [Started]
Markers
Console
Gradle Executions
There are no Gradle build executions to visualize. Run a Gradle build to see its execution in the Gradle Executions View.

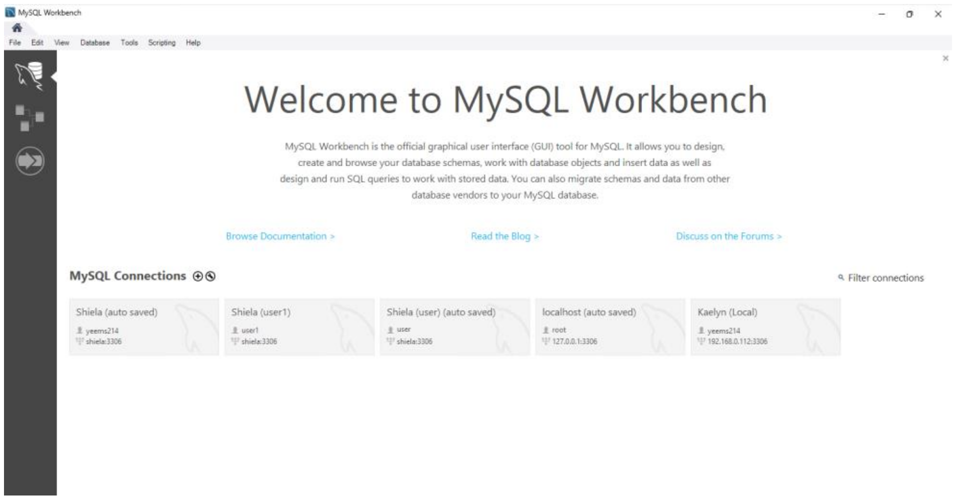
```
Liferay 7.x at localhost [Liferay 7.x] C:\Program Files\Java\jdk-11.0.16.1\bin\javaw.exe (Apr 13, 2023, 4:55:47 PM)
2023-04-13 08:58:21.893 INFO  [com.liferay.portal.osgi.web.wab.extender.internal
2023-04-13 08:58:22.271 INFO  [com.liferay.portal.osgi.web.wab.extender.internal
2023-04-13 08:58:22.720 INFO  [com.liferay.portal.osgi.web.wab.extender.internal
2023-04-13 08:58:53.001 INFO  [main][SystemCheckOSGiCommands:61] System check is
2023-04-13 08:58:53.006 INFO  [main][SystemCheckOSGiCommands:73] Running system
2023-04-13 08:58:53.007 INFO  [main][SystemCheckOSGiCommands:100] Available chec
```

IntelliJ IDEA Ultimate 2023.1 – As a secondary IDE for hosting Liferay and creating the portlets necessary for the AAA website Project



```
dependencies {
    compileOnly group: "com.liferay.portal", name: "release.portal.api"

    cssBuilder group: "com.liferay", name: "com.liferay.css.builder", version: "3.1.0"
}
```

MySQL Workbench – For the MySQL Database Management



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

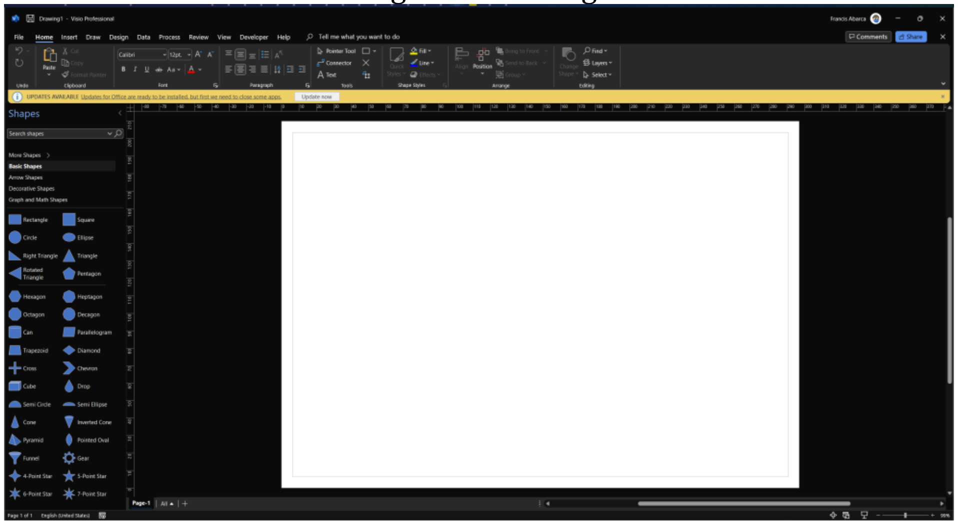
Browse Documentation > Read the Blog > Discuss on the Forums >

MySQL Connections

Filter connections

Shiela (auto saved)	Shiela (user1)	Shiela (user) (auto saved)	localhost (auto saved)	Kaelyn (Local)
yeems214	user1	user	root	yeems214
shiela:3306	shiela:3306	shiela:3306	127.0.0.1:3306	192.168.0.112:3306

Microsoft Visio – For creating the UML Diagrams



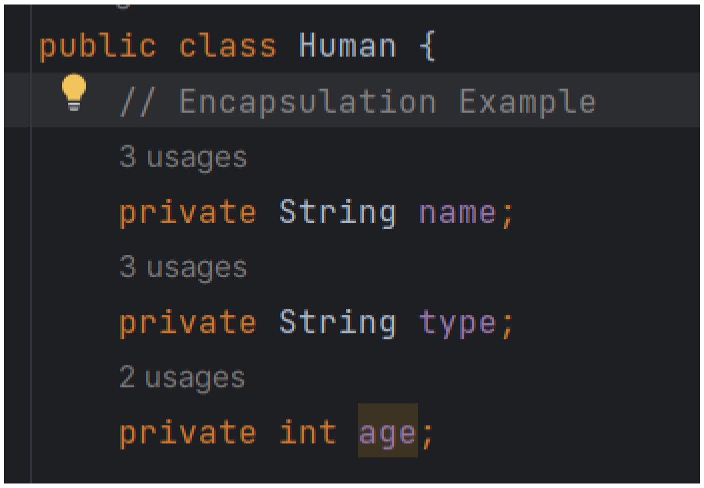
Task 1

Program Paradigm and Design Pattern

Solution:

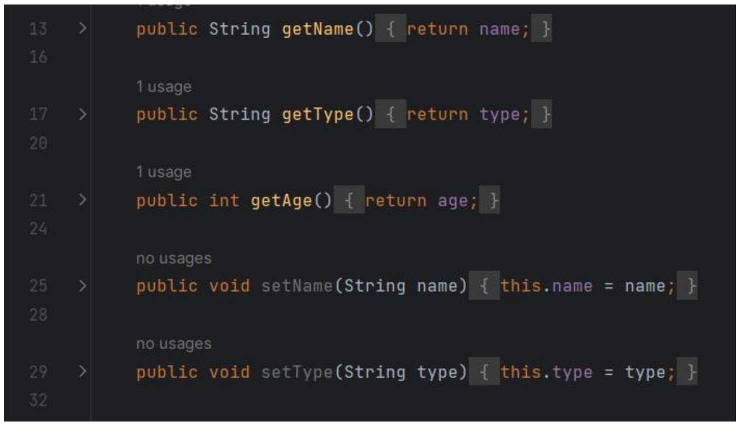
1. **Identify and explain the characteristics of the object-oriented paradigm and relationship between the various classes from a given code scenario.**
   a. **Explain briefly Object-Oriented Paradigm.**
   - Object Oriented Programming (OOP) is a programming paradigm that utilizes the concept of objects. Objects are then defined as a field of data with their own unique behavior and attributes. Object Oriented Programming simplifies development by enclosing most of the code in objects with their own methods and functions that relate to the same object.

   b. **Examine Characteristics of Object-Oriented Programming Paradigm**
   Encapsulation
   - This involves the bundling of data and methods that operate on the data within its own class thus restricting this class from running it on other classes.



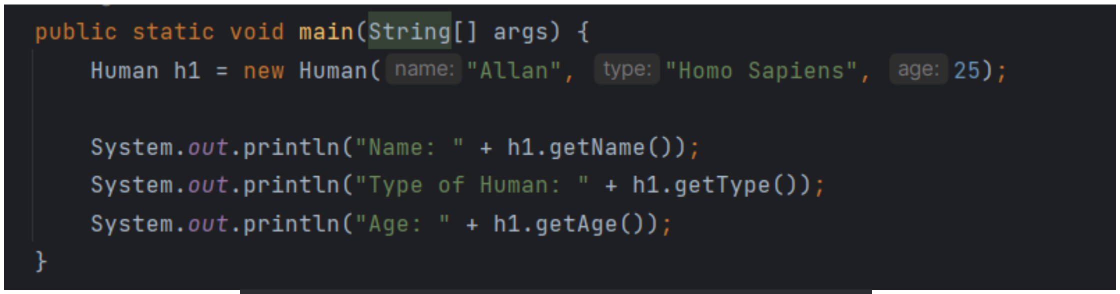
```
public class Human {
    // Encapsulation Example
    3 usages
    private String name;
    3 usages
    private String type;
    2 usages
    private int age;
```

In this example above, both String name, String type and int age are all encapsulated due to the use of private fields therefore, it can only be accessed on the same class and cannot be accessed outside.

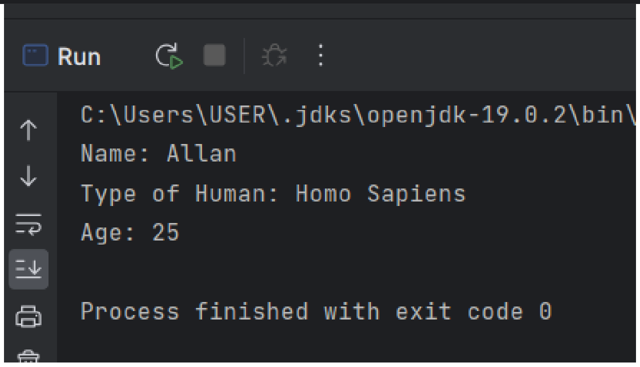


```
    public String getName() { return name; }

    1 usage
    public String getType() { return type; }

    1 usage
    public int getAge() { return age; }

    no usages
    public void setName(String name) { this.name = name; }

    no usages
    public void setType(String type) { this.type = type; }
```

Due to the use of encapsulation, String name, String type and int age needs to be configured to have get and set methods for it to be used outside the class.



```
public static void main(String[] args) {
    Human h1 = new Human( name: "Allan",  type: "Homo Sapiens",  age: 25);

    System.out.println("Name: " + h1.getName());
    System.out.println("Type of Human: " + h1.getType());
    System.out.println("Age: " + h1.getAge());
}
```

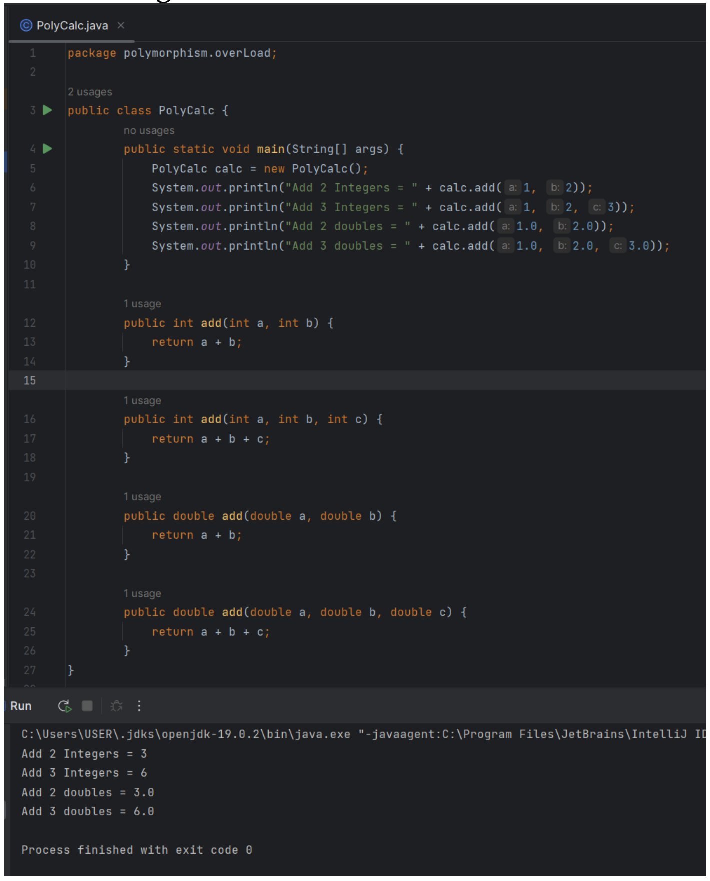


```
C:\Users\USER\.jdks\openjdk-19.0.2\bin\
Name: Allan
Type of Human: Homo Sapiens
Age: 25

Process finished with exit code 0
```

Polymorphism

- This allows objects of distinct classes to be viewed as objects of a shared superclass and allows a single interface to represent numerous types or support multiple method implementations, allowing for code design flexibility and extensibility.

Overloading

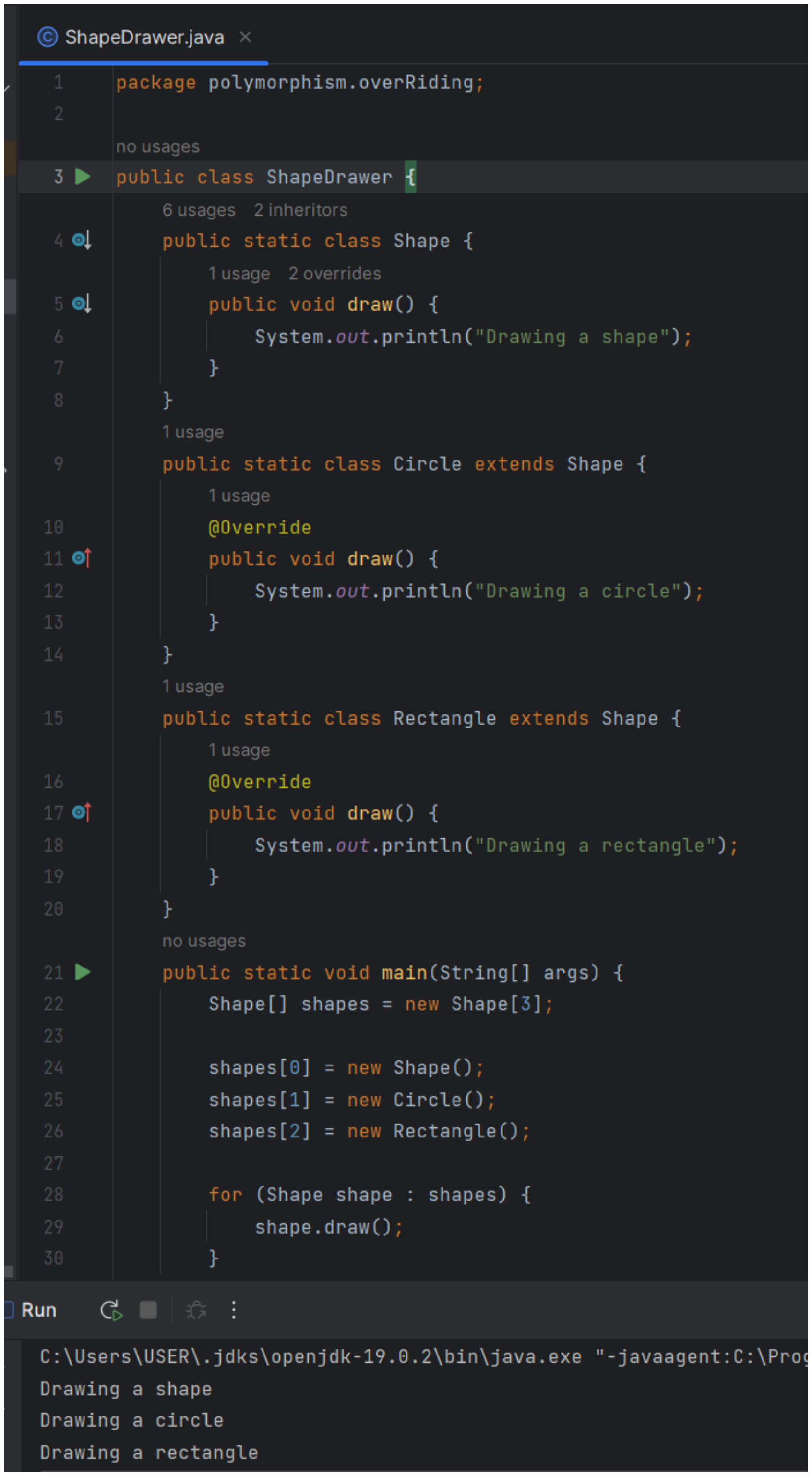


```
package polymorphism.overLoad;

2 usages
public class PolyCalc {
        no usages
        public static void main(String[] args) {
            PolyCalc calc = new PolyCalc();
            System.out.println("Add 2 Integers = " + calc.add( a: 1,  b: 2));
            System.out.println("Add 3 Integers = " + calc.add( a: 1,  b: 2,  c: 3));
            System.out.println("Add 2 doubles = " + calc.add( a: 1.0,  b: 2.0));
            System.out.println("Add 3 doubles = " + calc.add( a: 1.0,  b: 2.0,  c: 3.0));
        }

        1 usage
        public int add(int a, int b) {
            return a + b;
        }

        1 usage
        public int add(int a, int b, int c) {
            return a + b + c;
        }

        1 usage
        public double add(double a, double b) {
            return a + b;
        }

        1 usage
        public double add(double a, double b, double c) {
            return a + b + c;
        }
}
```

```
C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ ID
Add 2 Integers = 3
Add 3 Integers = 6
Add 2 doubles = 3.0
Add 3 doubles = 6.0

Process finished with exit code 0
```

There are 4 methods inside PolyCalc with their own parameters. Judging by these parameters, the main class calls the method with the right parameters given.

Overriding



```java
package polymorphism.overRiding;

public class ShapeDrawer {
    public static class Shape {
        public void draw() {
            System.out.println("Drawing a shape");
        }
    }
    public static class Circle extends Shape {
        @Override
        public void draw() {
            System.out.println("Drawing a circle");
        }
    }
    public static class Rectangle extends Shape {
        @Override
        public void draw() {
            System.out.println("Drawing a rectangle");
        }
    }
    public static void main(String[] args) {
        Shape[] shapes = new Shape[3];

        shapes[0] = new Shape();
        shapes[1] = new Circle();
        shapes[2] = new Rectangle();

        for (Shape shape : shapes) {
            shape.draw();
        }
```

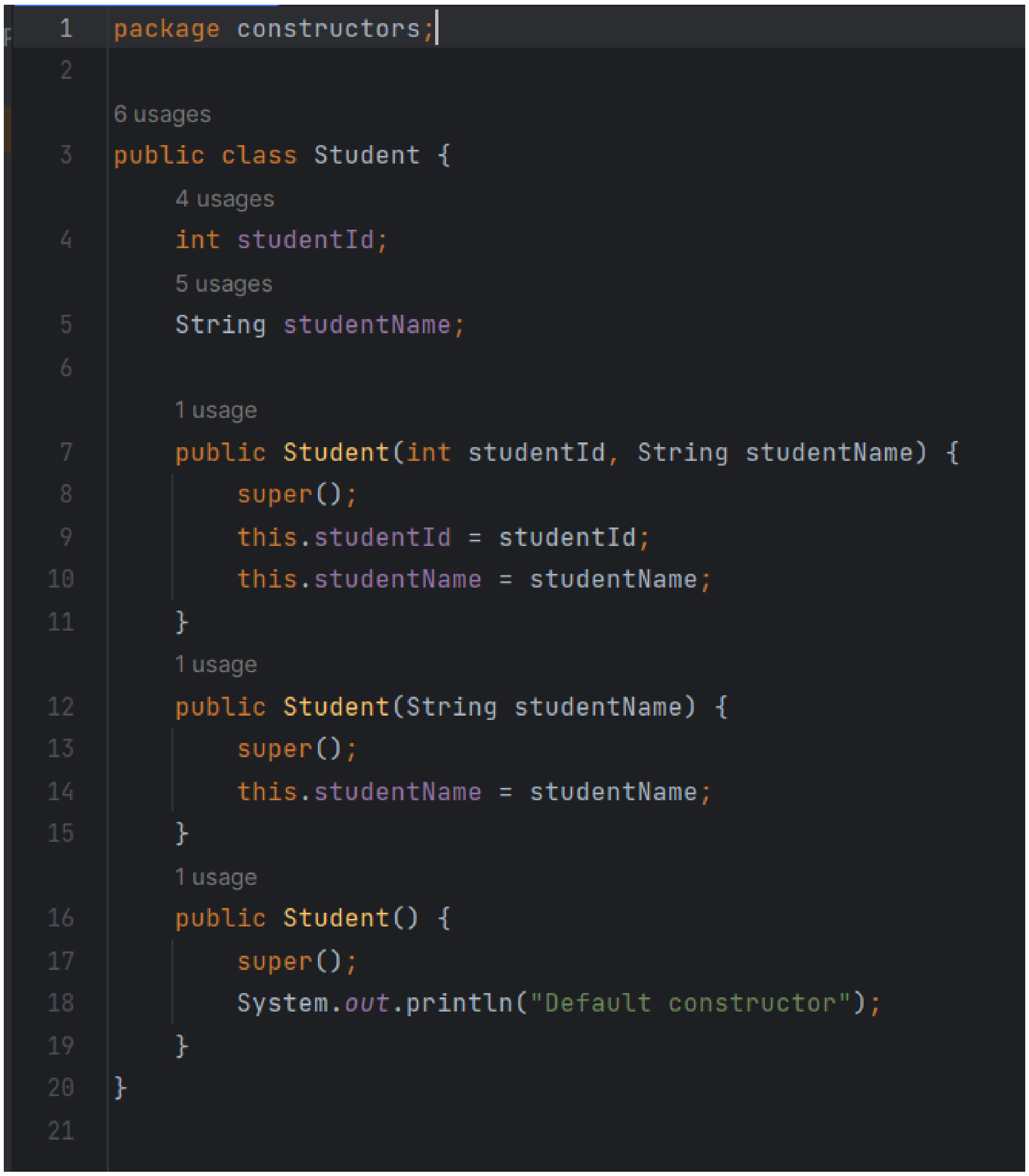
```
C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Prog
Drawing a shape
Drawing a circle
Drawing a rectangle
```

In this example, both the Circle and Rectangle subclasses are extending to the Shape class which then overrides the draw() method.

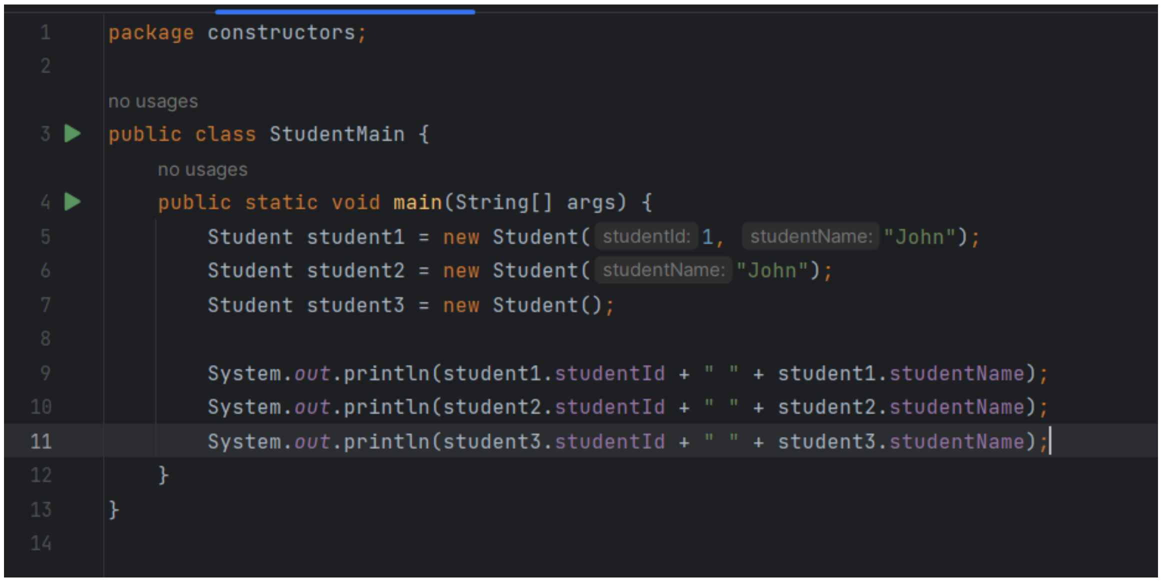
Constructors/Destructors

- A Java Constructor is used to initialize an object, whereas a Java Destructor is used to de-allocate an object at the end of its lifecycle.

Java Constructor



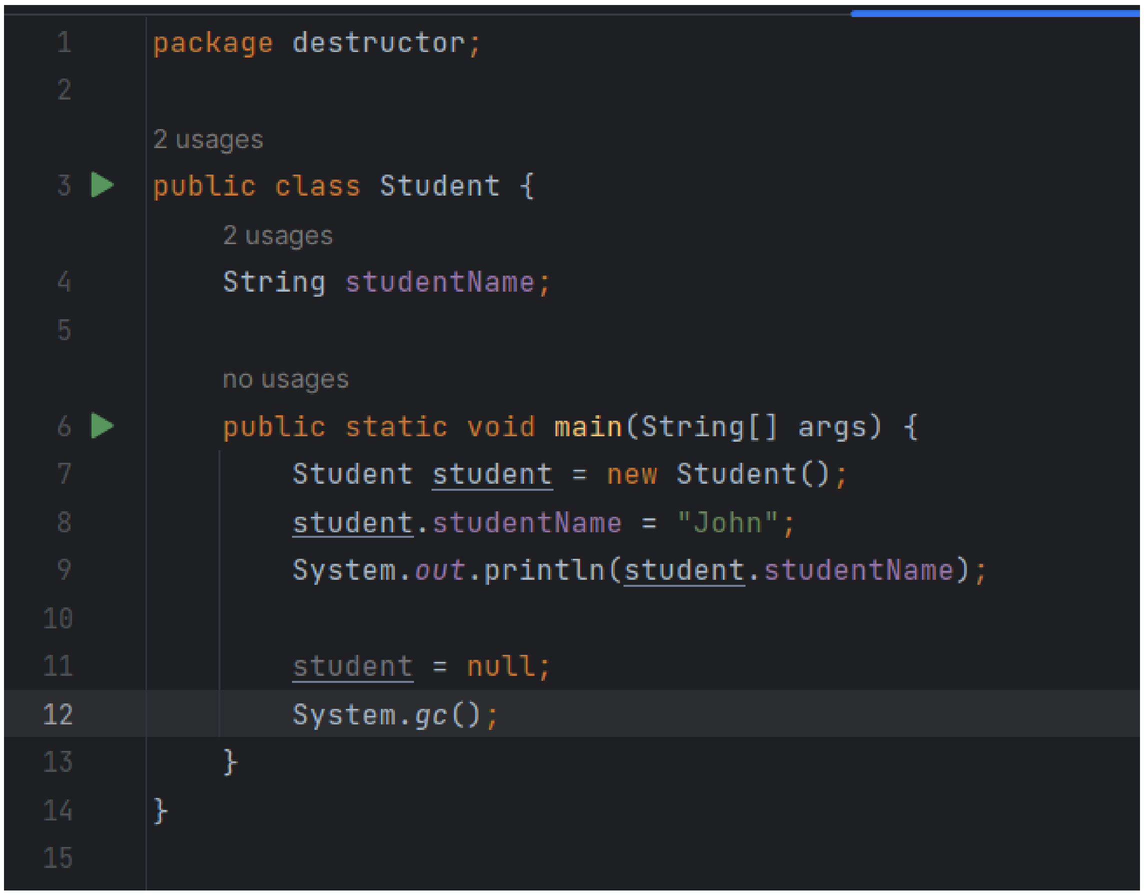
```java
package constructors;

6 usages
public class Student {
    4 usages
    int studentId;
    5 usages
    String studentName;

    1 usage
    public Student(int studentId, String studentName) {
        super();
        this.studentId = studentId;
        this.studentName = studentName;
    }
    1 usage
    public Student(String studentName) {
        super();
        this.studentName = studentName;
    }
    1 usage
    public Student() {
        super();
        System.out.println("Default constructor");
    }
}

```



```
package constructors;

no usages
public class StudentMain {
    no usages
    public static void main(String[] args) {
        Student student1 = new Student( studentId: 1,  studentName: "John");
        Student student2 = new Student( studentName: "John");
        Student student3 = new Student();

        System.out.println(student1.studentId + " " + student1.studentName);
        System.out.println(student2.studentId + " " + student2.studentName);
        System.out.println(student3.studentId + " " + student3.studentName);
    }
}
```

These constructors above show 2 types of constructors which have no arguments (Default) and with one or more arguments (Parametrized).

Java Destructor

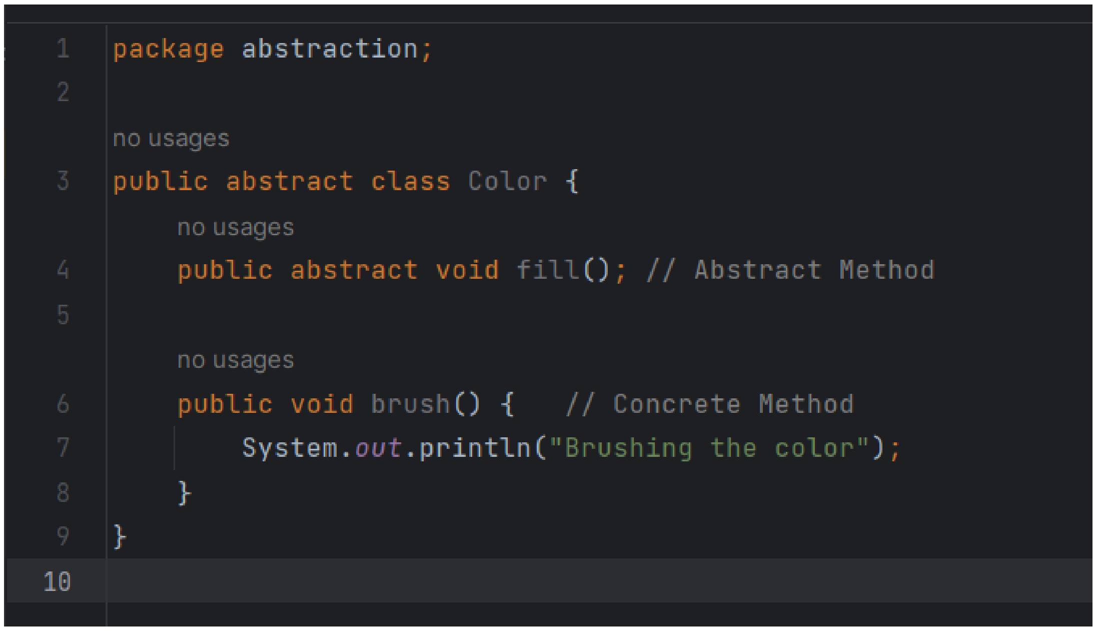


```
package destructor;

2 usages
public class Student {
    2 usages
    String studentName;

    no usages
    public static void main(String[] args) {
        Student student = new Student();
        student.studentName = "John";
        System.out.println(student.studentName);

        student = null;
        System.gc();
    }
}
```

The System.gc() method runs the garbage collector and prevents from running the object by deleting its instance.

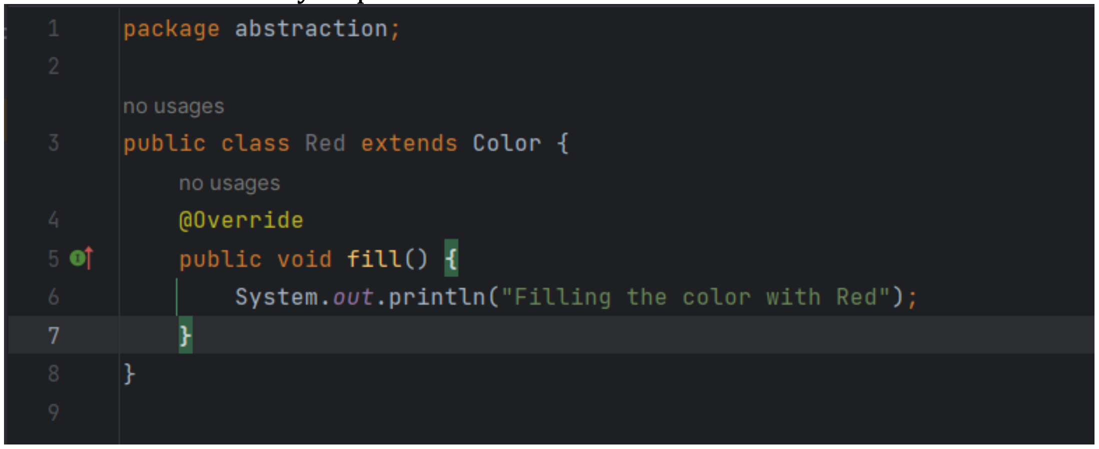
Abstract

- Abstraction is used to hide certain details of a specific class and show its necessary functionalities only. It is achieved by making abstract classes and interfaces.



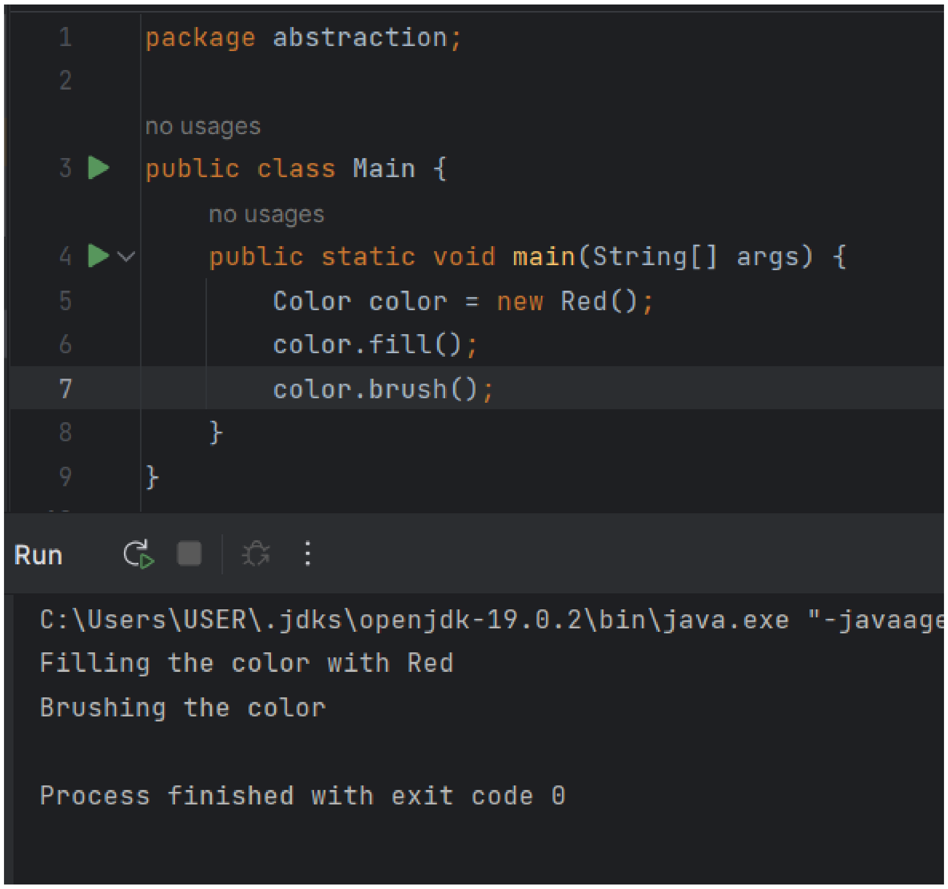
```java
package abstraction;

no usages
public abstract class Color {
    no usages
    public abstract void fill(); // Abstract Method

    no usages
    public void brush() {   // Concrete Method
        System.out.println("Brushing the color");
    }
}

```

The Color class above is declared as an abstract class with the keyword. It is also made without any implementations.



```java
package abstraction;

no usages
public class Red extends Color {
    no usages
    @Override
    public void fill() {
        System.out.println("Filling the color with Red");
    }
}

```

The Red class is declared which extends from Color and in this class is where the fill() color is ran.



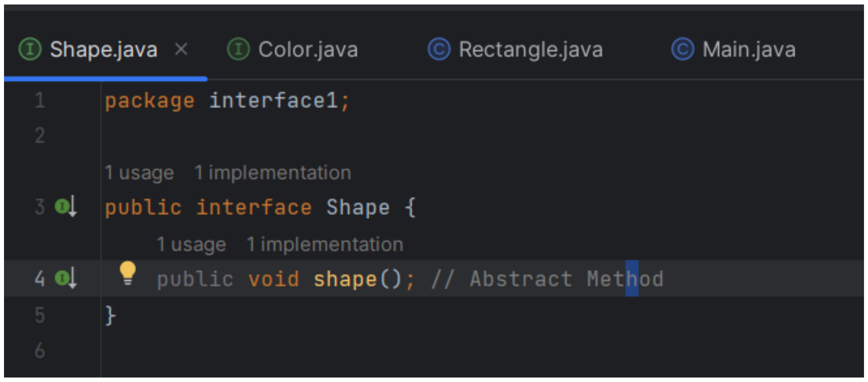
```
package abstraction;

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Color color = new Red();
        color.fill();
        color.brush();
    }
}
```

```
C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe "-javaage
Filling the color with Red
Brushing the color

Process finished with exit code 0
```

Once the fill() method is ran inside color, this implementation is called.

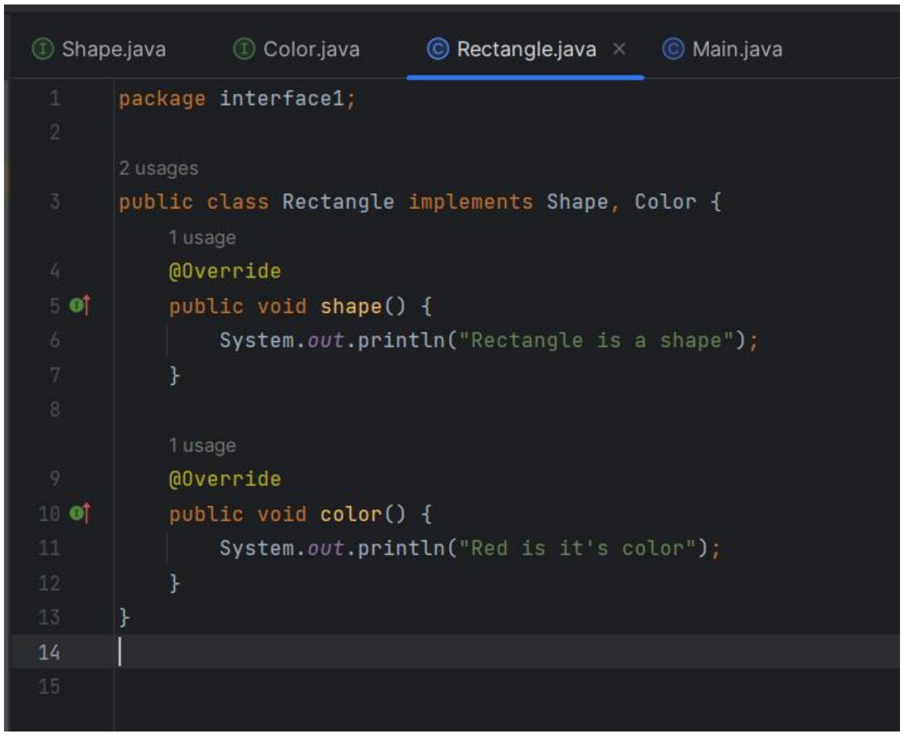
Interface

- These are abstract classes used to contain methods with no body.



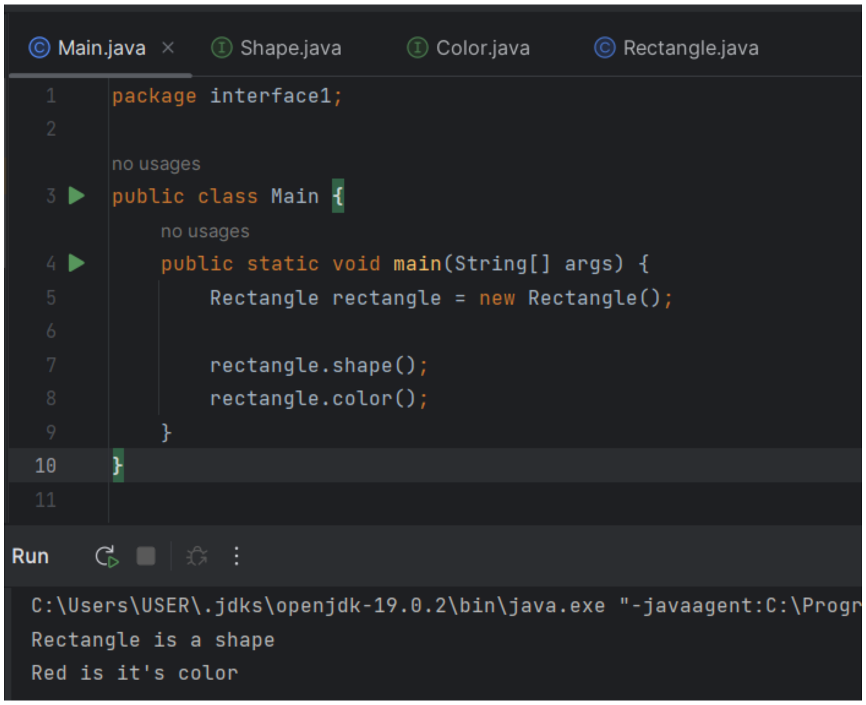
```
package interface1;

1 usage   1 implementation
public interface Shape {
    1 usage   1 implementation
    public void shape(); // Abstract Method
}
```

The shape() method doesn't have a body in the Shape interface.



```java
package interface1;

2 usages
public class Rectangle implements Shape, Color {
    1 usage
    @Override
    public void shape() {
        System.out.println("Rectangle is a shape");
    }

    1 usage
    @Override
    public void color() {
        System.out.println("Red is it's color");
    }
}
```

Inside the Rectangle class, it implements both the shape and color interface which then provides the body for the interface method.



```java
package interface1;

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();

        rectangle.shape();
        rectangle.color();
    }
}
```

```
C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Progr
Rectangle is a shape
Red is it's color
```

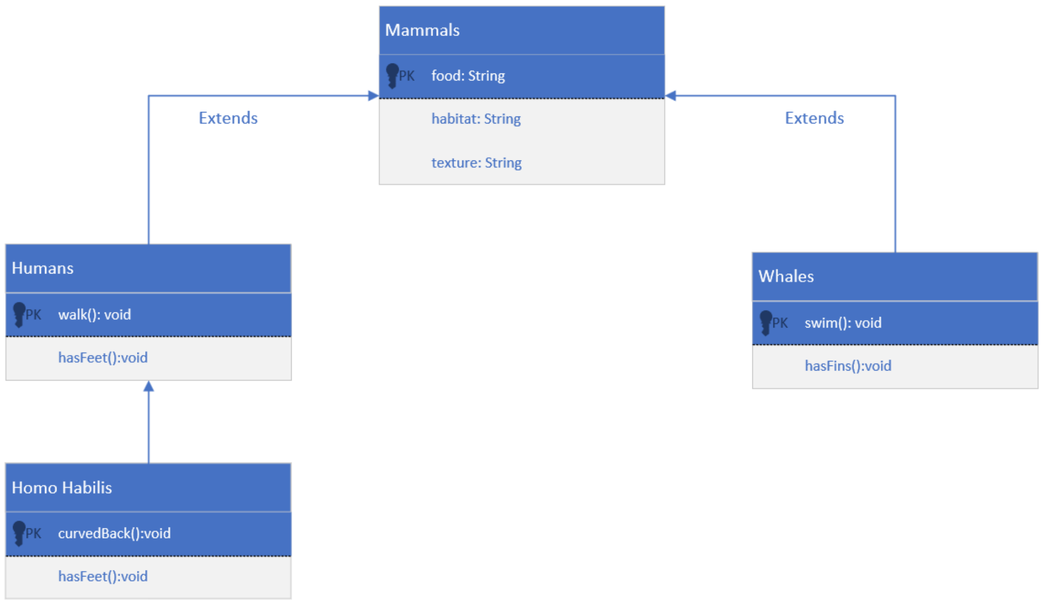
The color() method is then accessed by the Rectangle class which implements the color interface alongside shape.

What are the differences between abstract classes and interfaces?

Abstract Class	Interface
Utilizes the "abstract" keyword	Utilizes the "interface" keyword
Ability to extend classes	Ability to implement classes
It can utilize non-abstract methods	It can only utilize abstract methods
Multiple classes can only inherit one abstract class	One interface can implement multiple classes
It can utilize non-static and non-final variables	It can only utilize static and final variables.
It can have both a private and a protected class member	It can only have a public class
It can implement interfaces	It can't implement abstract classes

c. Examine Class Relationships using UML diagrams.

Inheritance
- It is a relationship between 2 classes where a child class can inherit attributes created by the parent class.



Inheritance can have 3 types:

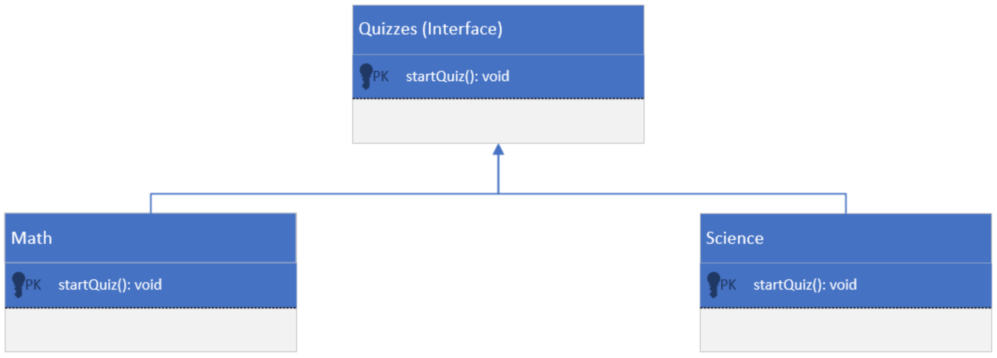
Single Level Inheritance – This is where 1 class only inherits the other class on a single level view. For example, the Whales class is inheriting the Mammals class.

Multi-Level Inheritance – This is where a class then inherits another class which then inherits another class in multiple levels. For example, the Homo Habilis class is inheriting from Humans which is also inheriting from Mammals.

Hierarchical Inheritance – This is where both the Humans and Whales extends and are under the Mammals class.

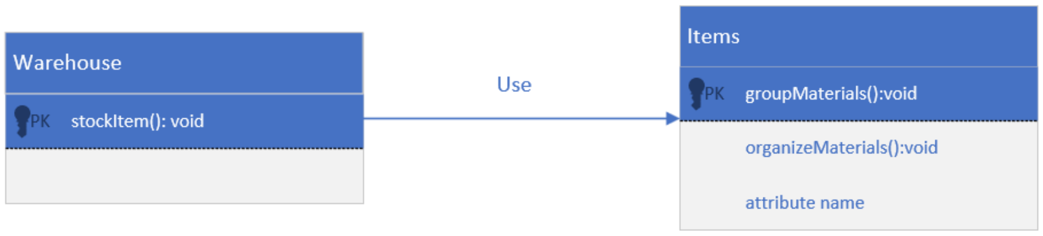
Realization

- The relationship between an interface and an object class in which the object class implements the method declared in the interface.



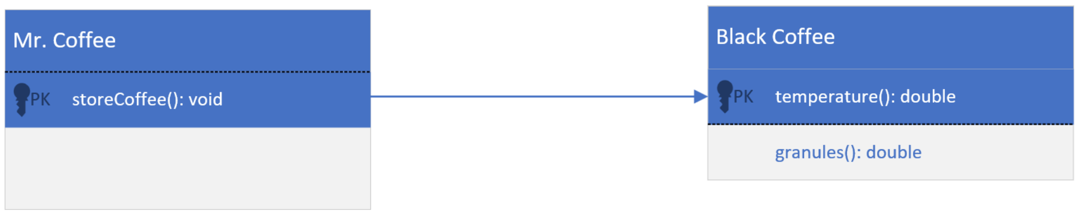
Dependency

- A relationship in which one class (a) is dependent on another class (b), and as class (b) changes, so does class (a).



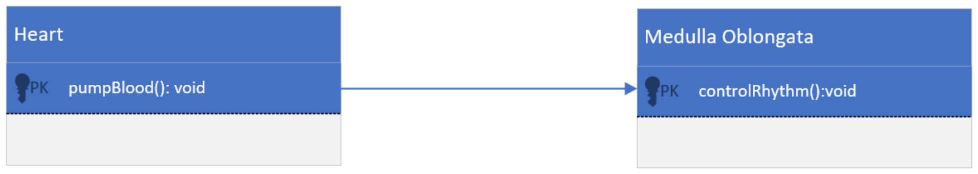
Aggregation

- A connection in which one class is a subset of another, but if the parent class is deleted, the child class is not deleted.



Composition

- A more extreme kind of aggregation in which if the parent class is deleted, the child class is also deleted.



d. **Examine and list out class relationships**
- Hierarchical Inheritance
- Association

2. **Identify the suitable design patterns based on the project scenario.**

a. **What is a Creational Design Pattern? List out creational design patterns.**
- The creational design pattern is concerned with class instantiation, or the creation of a class's object.

Abstract Factory

- An interface for creating a family of objects using a super factory class.

Builder

- Using builder classes, separates an object's step-by-step creation code from its representation.

Factory

- An interface that manages object creation, but a subclass determines which class to instantiate.

Prototype

- Generates an object prototype that can be duplicated to generate new objects.

Singleton

- Makes a single instance object and only one instance of that object.

b. **Identify suitable design pattern for "A3 Portlet"**
- Factory Method

c. **What is structural design pattern? List out structural design pattern.**
- The structural design pattern is concerned with the design of the structure between objects.

Adapter

- A class that connects conflicting interfaces' functions.

Bridge

- Separates an interface from its implementation, allowing them to function independently.

Composite

- This is when a set of objects must be regarded as a single object.

Decorator

- Allows the addition of new functions to an object without causing it to change in structure.

Façade

- Hides the system's complexity while allowing the client to access it.

Proxy

- Serves as a placeholder for the functionality of another class.

d. Identify suitable design pattern for "A3 Portlet"

- Façade Pattern

e. What is behavioral design pattern? List out behavioral design patterns.

- A behavioral design pattern describes how things communicate with one another.

Chain of Responsibility

- Passing a request through a series of classes until it can no longer.

Command

- The encapsulation of a request in the form of an object that is handed to an invoker object.

Iterator

- A method of gaining access to the elements of a collection without knowing its structure.

Mediator

- A class that manages all inter-class communication.

Memento

- Used to return an object to its original state.

Observer

- Ensures that dependent objects are notified when their primary object changes.

State

- Determines how an item should behave internally based on its status.

Template Method

- Generates a template with steps for executing its methods, which subclasses might override as needed.

f. Identify suitable design pattern for "A3 Portlet"

- Template Method

3. Examine how the object-orientated paradigm and its key principle is identified in each of the design patterns.

- To function, Design Patterns use several OOP principles such as encapsulation, inheritance, and abstraction. Encapsulation, for example, is required for creational design patterns to work properly. All of its construction logic is encapsulated in a method that will create the object. It also use abstraction to conceal all of its complexity while enabling the logic to be utilized. Then there are Structural and Behavioral design patterns, which are concerned with the interactions between items. Without inheritance, most of these items would have nothing in common. Following the development and interaction of objects, these design patterns were depicted. Objects are at the heart of OOP.

Task 2

System Design

Solution:

1. **Draw class diagrams that represent a simple structure based on the project scenarios using a UML tool.**
    a. **Draw class diagrams showing class relationships**

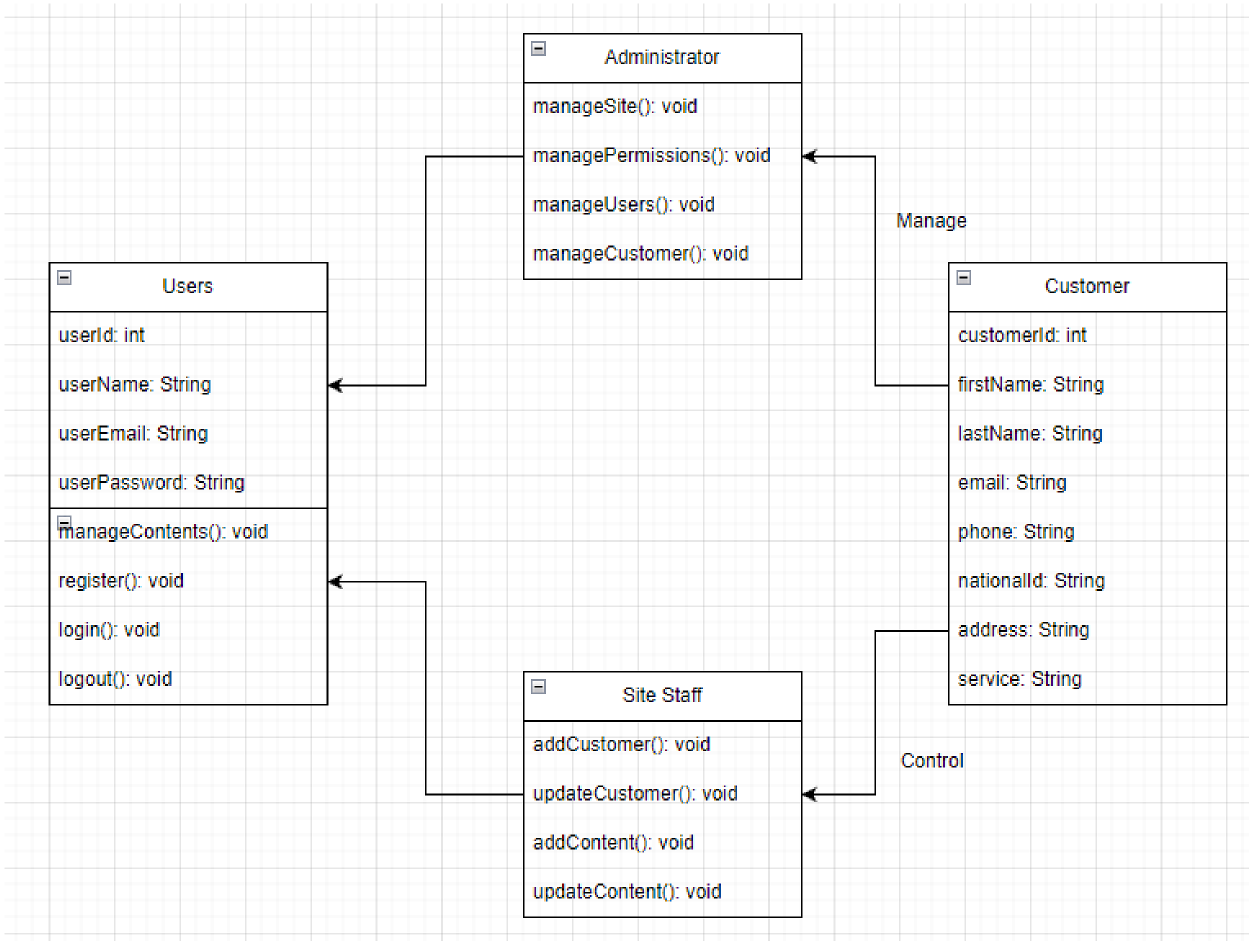
What is a Class Diagram?

- A class diagram depicts the connection between objects and their characteristics. It is used to view the structure of a program before implementing it.

Based on the scenario, define the classes, attributes and methods

Classes	Purpose	Attributes	Methods
Users	Represent's the site's 2 primary users.	userId, userName, userEmail, userPassword	manageContents(), registerUser(), login(), logout()
Administrator	1. Manage the whole site's content and pages. 2. Refresh the theme and layout. 3. Control permissions and manage the customer data portlet. 4. Manage all user roles and permissions (Site Member and Customer).	Inherited from Users	manageSites(), managePermissions(), manageUsers(), manageCustomer()
Site Member (Staff)	1. Update the A3 website's content. 2. Add and update customer information for the A3 site.	Inherited from Users	updateCustomer (), deleteCustomer()
Customers	Site Visitors	customerID, name, email, address, nationalID, contact	giveRegister()

Class Diagram



Explain relationships between classes

The connection between two subcategories, or child classes, Administrator and Site Member, with their superclass, or parent class Users, is denoted by a solid arrowed line. Both Administrator and Site Staff fall under the category of Users within the Triple A (AAA) platform since they share the same attributes and functions as the User class, in addition to having their own exclusive characteristics. An association link exists between the Administrator and Customer, as well as between Site Staff and Customer, represented by a solid line linking the classes. In the project context, both the Administrator and Site Staff are responsible for overseeing and managing the clients.

2. Identify possible situations where design patterns would be beneficial and then develop the UML diagrams reflecting the design patterns.

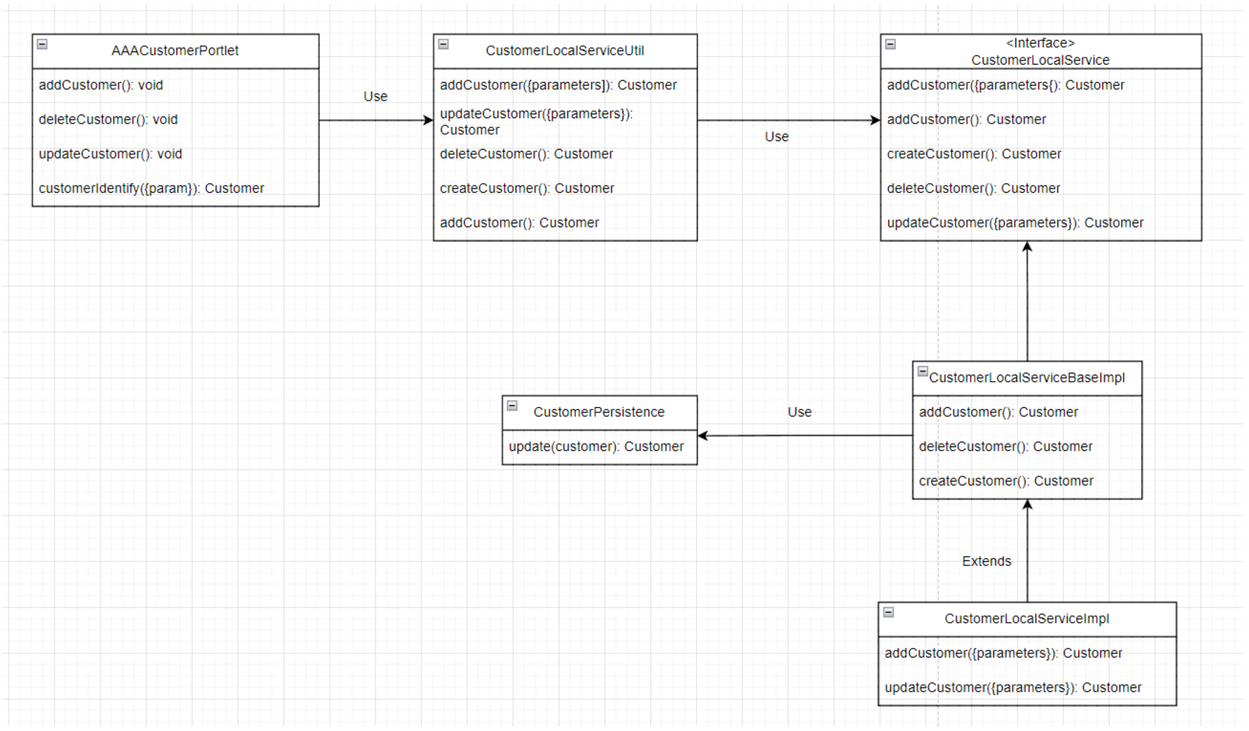
a. Define class diagrams for specific design patterns which have been drawn in previous task (AAA Customer Management Portlet) using a UML tool.

b. Benefits of Design Patterns

c. Develop UML diagrams which you can use in “AAA Portlet”

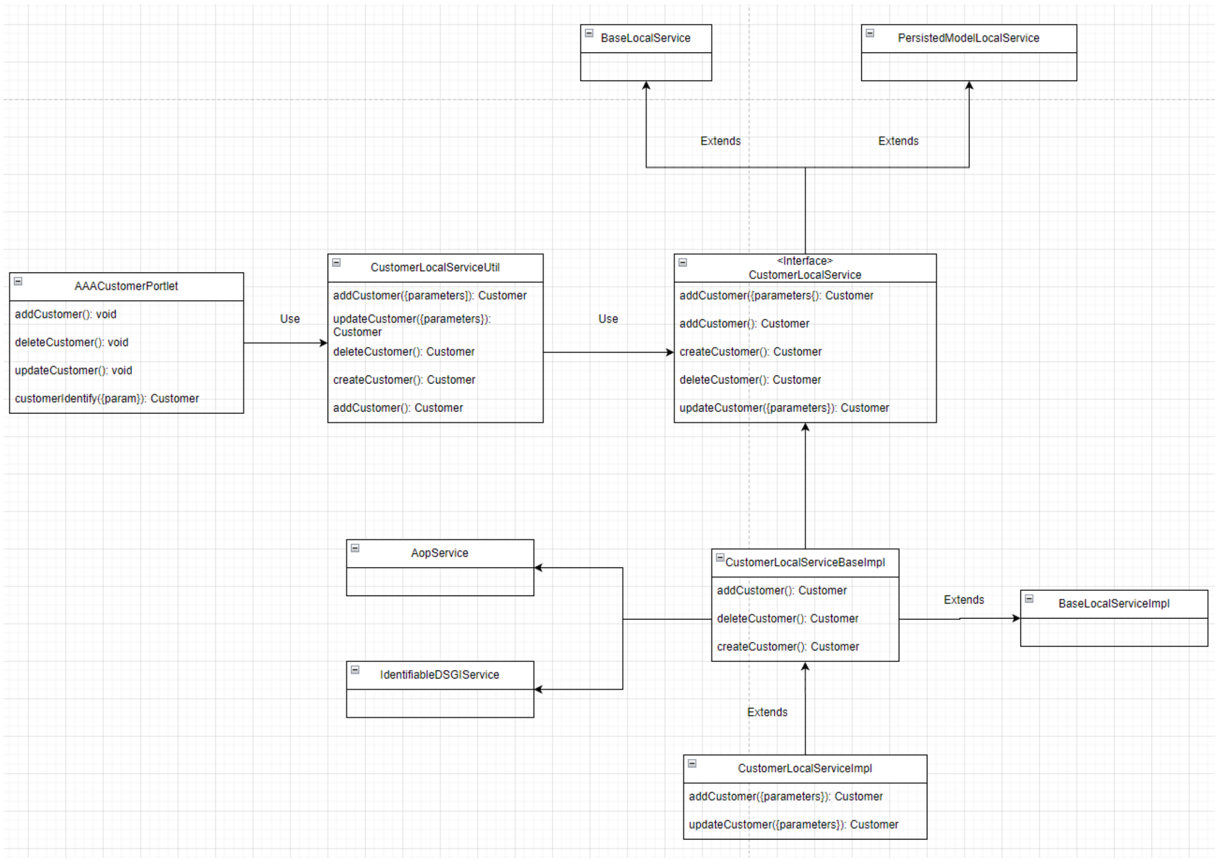
Creational Design Pattern (Factory Method)

The Factory Method is beneficial as it serves as a knowledge mechanism for generating the necessary object instance. In the TripleACustomerPortlet, the object is created through the addCustomer() method invocation, which in turn calls the customerIdentity method. This method subsequently calls the addCustomer(parameters) method in CustomerLocalServiceUtil, which also triggers the addCustomer(parameters) method in CustomerLocalService. The implementation for the addCustomer(parameters) method can be found in the CustomerLocalServiceImpl class.


AAACustomerPortlet
addCustomer(): void
deleteCustomer(): void
updateCustomer(): void
customerIdentify({param}): Customer
Use
CustomerLocalServiceUtil
addCustomer({parameters]): Customer
updateCustomer({parameters}): Customer
deleteCustomer(): Customer
createCustomer(): Customer
addCustomer(): Customer
Use
<Interface> CustomerLocalService
addCustomer({parameters{): Customer
addCustomer(): Customer
createCustomer(): Customer
deleteCustomer(): Customer
updateCustomer({parameters}): Customer
CustomerLocalServiceBaseImpl
addCustomer(): Customer
deleteCustomer(): Customer
createCustomer(): Customer
Use
CustomerPersistence
update(customer): Customer
Extends
CustomerLocalServiceImpl
addCustomer({parameters}): Customer
updateCustomer({parameters}): Customer

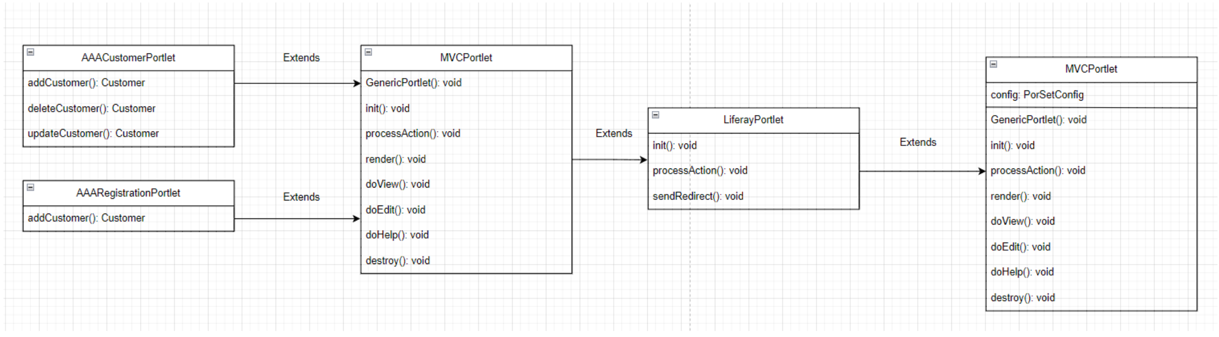
Structural Design Pattern (Façade Pattern)

The facade pattern is advantageous for concealing or streamlining subcategories. The CustomerLocalServiceUtil class can be employed to execute the logic for CRUD operations. While masking the intricacies of the logic, it still permits access to the functionality. In this instance, it serves as the Facade class.


BaseLocalService
PersistedModelLocalService
Extends
Extends
AAACustomerPortlet
addCustomer(): void
deleteCustomer(): void
updateCustomer(): void
customerIdentify({param}): Customer
Use
CustomerLocalServiceUtil
addCustomer({parameters}): Customer
updateCustomer({parameters}): Customer
deleteCustomer(): Customer
createCustomer(): Customer
addCustomer(): Customer
Use
<Interface> CustomerLocalService
addCustomer({parameters(): Customer
addCustomer(): Customer
createCustomer(): Customer
deleteCustomer(): Customer
updateCustomer({parameters}): Customer
AopService
IdentifiableDSGIService
CustomerLocalServiceBaseImpl
addCustomer(): Customer
deleteCustomer(): Customer
createCustomer(): Customer
Extends
BaseLocalServiceImpl
Extends
CustomerLocalServiceImpl
addCustomer({parameters}): Customer
updateCustomer({parameters}): Customer

Behavioral Design Pattern (Template Method)

The template method proves beneficial when multiple variations of an object exist. This is achieved by reusing code, thereby maintaining a consistent object structure. It demonstrates how TripleACustomerPortlet and TripleARegistrationPortlet can possess distinct functions while sharing a common structure. Portlet classes are created by extending MVCPortlet, which extends LiferayPortlet, which further extends GenericPortlet. Consequently, all constructed portlets conform to the GenericPortlet's structure.



Benefits of Design Patterns

Design patterns are advantageous and beneficial for multiple reasons:

- They provide solutions to common programming problems.
    - Design patterns have been proven time and time again solutions to common program design challenges. Developers save time and effort by utilizing these patterns rather than starting from scratch.
- They facilitate communication and debugging.
    - Utilizing design patterns also help developers use a standard vocabulary that they can use to readily express their ideas and intentions whilst reducing misinterpretation and uncertainty.
- They help create scalable programs.
    - Design patterns help developers build more scalable software that is designed to handle extreme amounts of data. Patterns such as Observer and Strategy are in widespread use while developing enterprise-grade applications.
- They encourage code usability.
    - When using design patterns, there's no abrupt end to how many times you can use it. Design patterns also help developers create versatile and adaptive code that they can also use to implement other projects.
- They improve code quality.
    - With Design Patterns, developers can write code that is easy to understand, maintain and extend thanks to its utilization of the best practices in software design principles such as Modularity, Encapsulation and Separation.

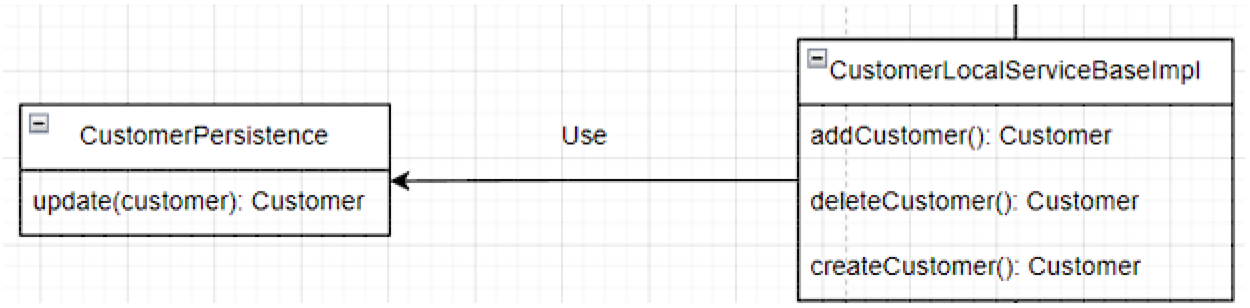
All in all, design patterns can help developers significantly by creating more reliable, high quality, successful software with its use of high-quality practices, ease of communication, standardization of the programming vocabulary and modularity.

3. Observe how the class diagrams are reflected from a given scenario by using the UML tool.

a. Observe how class diagrams are reflected from "AAA Customer Management Portlet" using the UML tool.

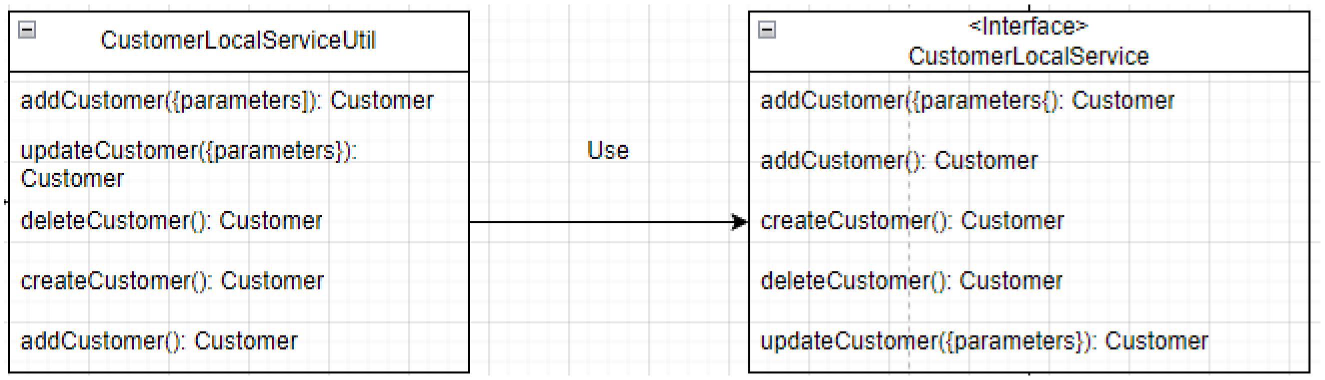
Creational Design Pattern (Factory Method)

In the UML diagram, when the CustomerLocalServiceBaseImpl executes its addCustomer() method, it triggers the update(customer) function from CustomerPersistence. This interface can be employed to initiate the necessary logic for generating an object instance. Upon creating an instance, the TripleACustomerPortlet can assign relevant data to the newly established object. This functionality is achievable due to the interdependence of these classes.


CustomerPersistence
update(customer): Customer
Use
CustomerLocalServiceBaseImpl
addCustomer(): Customer
deleteCustomer(): Customer
createCustomer(): Customer

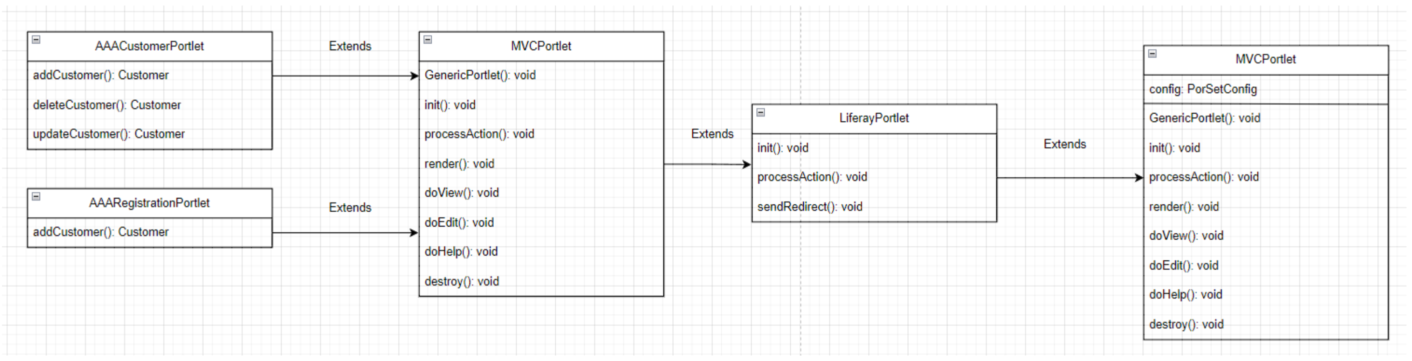
Structural Design Pattern (Façade Pattern)

As depicted in the UML diagram for the Façade Pattern, the CustomerLocalServiceUtil relies on the CustomerLocalService. To invoke the methods, the CustomerLocalService must traverse a sequence of classes; yet, with the help of CustomerLocalServiceUtil, these complex methods can be accessed. The façade class in this context is the CustomerServiceUtil class.


CustomerLocalServiceUtil
addCustomer({parameters]): Customer
updateCustomer({parameters}): Customer
deleteCustomer(): Customer
createCustomer(): Customer
addCustomer(): Customer
Use
<Interface>
CustomerLocalService
addCustomer({parameters{): Customer
addCustomer(): Customer
createCustomer(): Customer
deleteCustomer(): Customer
updateCustomer({parameters}): Customer

Behavioral Design Pattern (Template Method)

The template design pattern illustrates the generalization relationship between the abstract class GenericPortlet and its subclasses. Subclasses modify the method implementation without altering the structure of the abstract class.



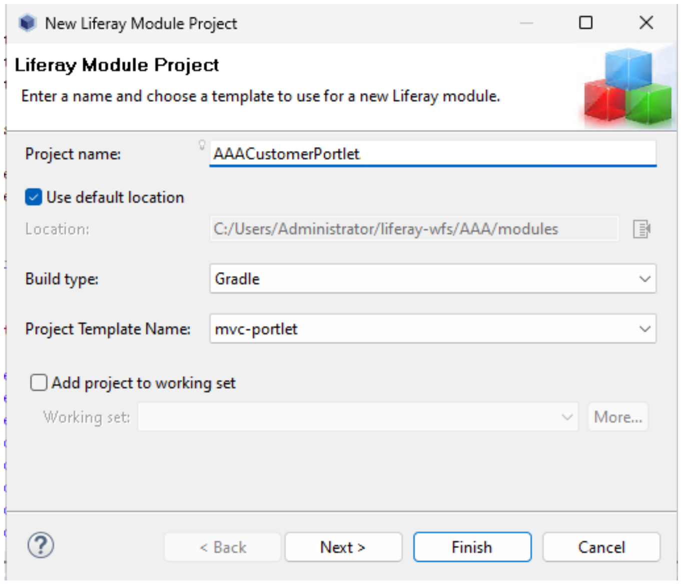
Task 3

System Development

Solution:

1. Set up a project, build an application based on your derived UML class diagrams and design patterns. Provide screenshots as evidence.

a. Create MVC Portlet to manage AAA customer's data.



New Liferay Module Project

Liferay Module Project

Enter a name and choose a template to use for a new Liferay module.

Project name: AAACustomerPortlet

[✓] Use default location

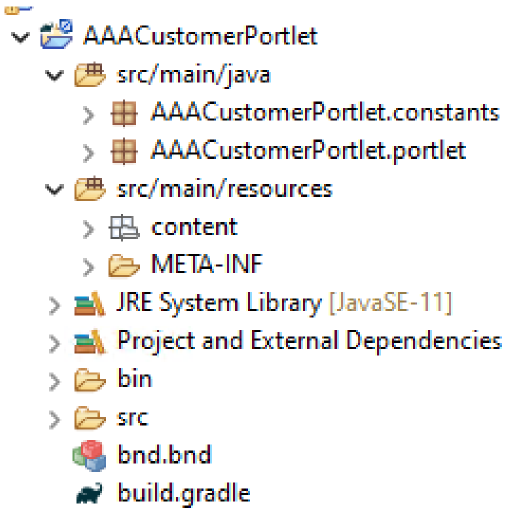
Location: C:/Users/Administrator/liferay-wfs/AAA/modules

Build type: Gradle

Project Template Name: mvc-portlet

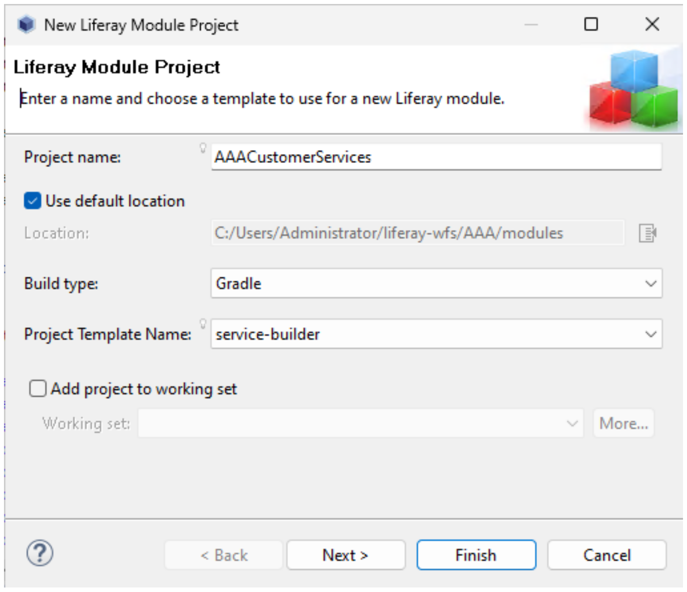
[ ] Add project to working set

Working set:



```
AAACustomerPortlet
  src/main/java
    AAACustomerPortlet.constants
    AAACustomerPortlet.portlet
  src/main/resources
    content
    META-INF
  JRE System Library [JavaSE-11]
  Project and External Dependencies
  bin
  src
  bnd.bnd
  build.gradle
```

b. Build the required entities using service-builder.



New Liferay Module Project

Liferay Module Project

Enter a name and choose a template to use for a new Liferay module.

Project name: AAACustomerServices

[✓] Use default location

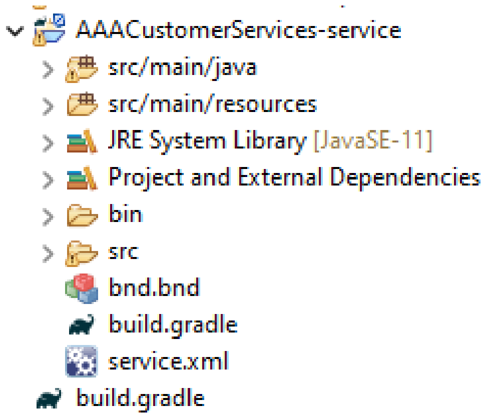
Location: C:/Users/Administrator/liferay-wfs/AAA/modules

Build type: Gradle

Project Template Name: service-builder

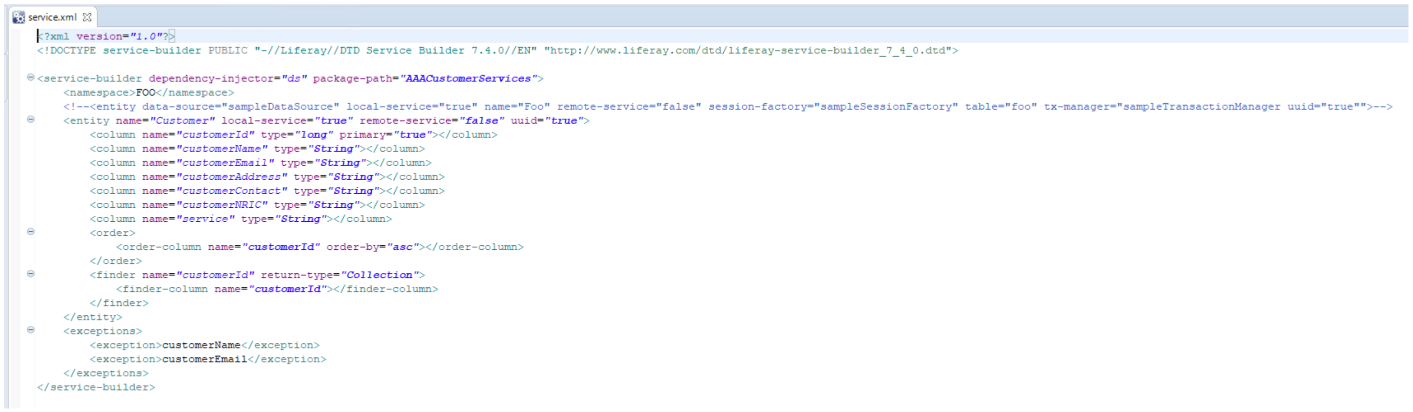
[ ] Add project to working set

Working set:



```
AAACustomerServices-service
  src/main/java
  src/main/resources
  JRE System Library [JavaSE-11]
  Project and External Dependencies
  bin
  src
  bnd.bnd
  build.gradle
  service.xml
build.gradle
```

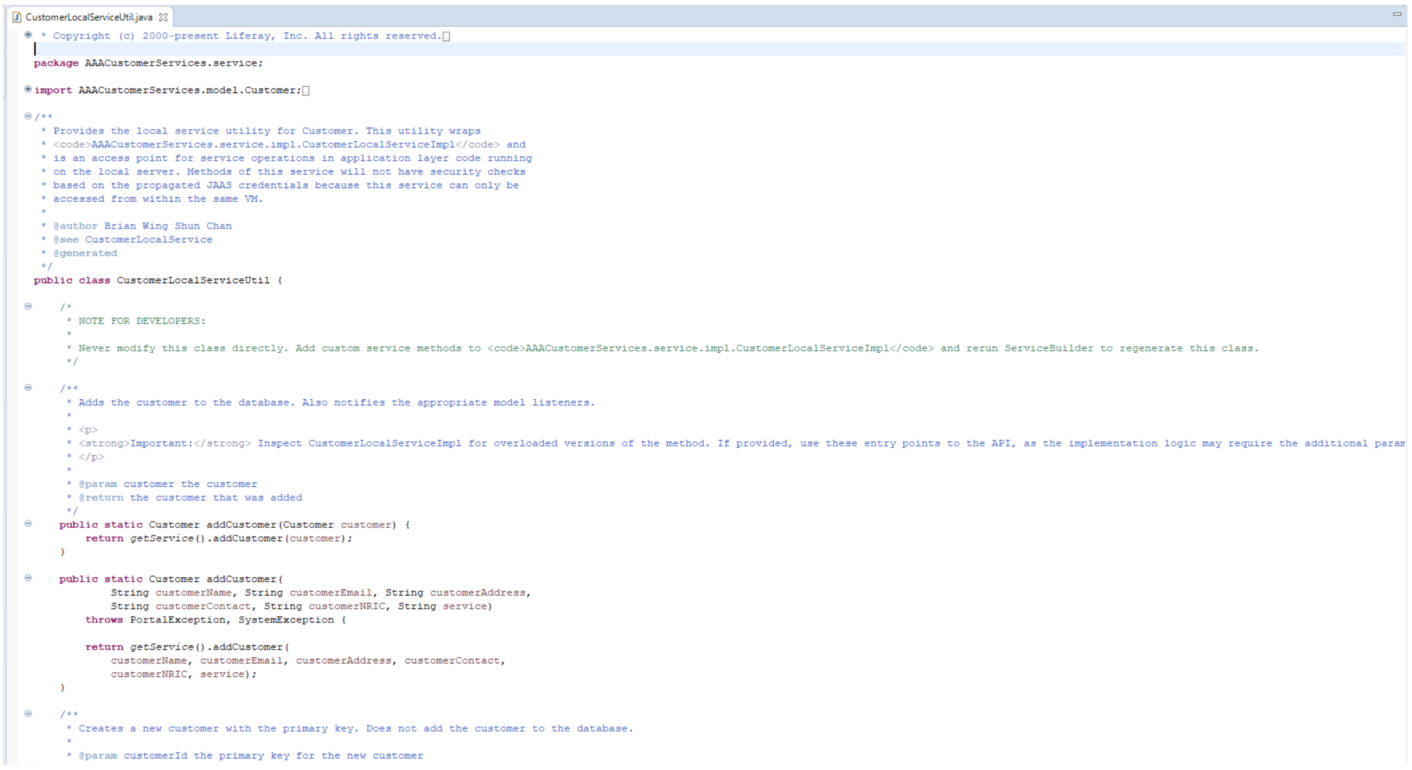
c. Provide screen capture of service.xml.



service.xml

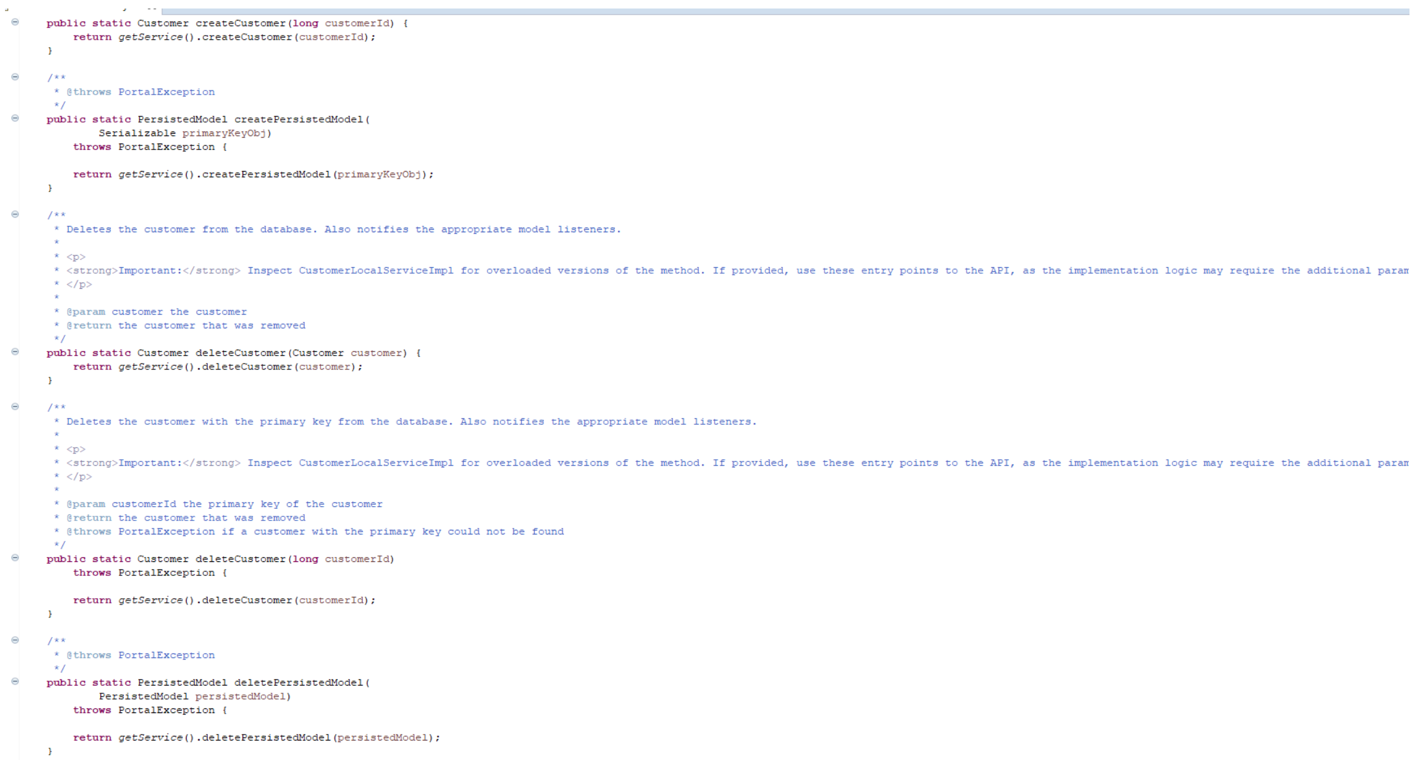
```
<?xml version="1.0"?>
<!DOCTYPE service-builder PUBLIC "-//Liferay//DTD Service Builder 7.4.0//EN" "http://www.liferay.com/dtd/liferay-service-builder_7_4_0.dtd">

<service-builder dependency-injector="ds" package-path="AAACustomerServices">
    <namespace>FOO</namespace>
    <!--<entity data-source="sampleDataSource" local-service="true" name="Foo" remote-service="false" session-factory="sampleSessionFactory" table="foo" tx-manager="sampleTransactionManager uuid="true"">-->
    <entity name="Customer" local-service="true" remote-service="false" uuid="true">
        <column name="customerId" type="long" primary="true"></column>
        <column name="customerName" type="String"></column>
        <column name="customerEmail" type="String"></column>
        <column name="customerAddress" type="String"></column>
        <column name="customerContact" type="String"></column>
        <column name="customerNRIC" type="String"></column>
        <column name="service" type="String"></column>
        <order>
            <order-column name="customerId" order-by="asc"></order-column>
        </order>
        <finder name="customerId" return-type="Collection">
            <finder-column name="customerId"></finder-column>
        </finder>
    </entity>
    <exceptions>
        <exception>customerName</exception>
        <exception>customerEmail</exception>
    </exceptions>
</service-builder>
```

d. Provide screen capture of the service layer.

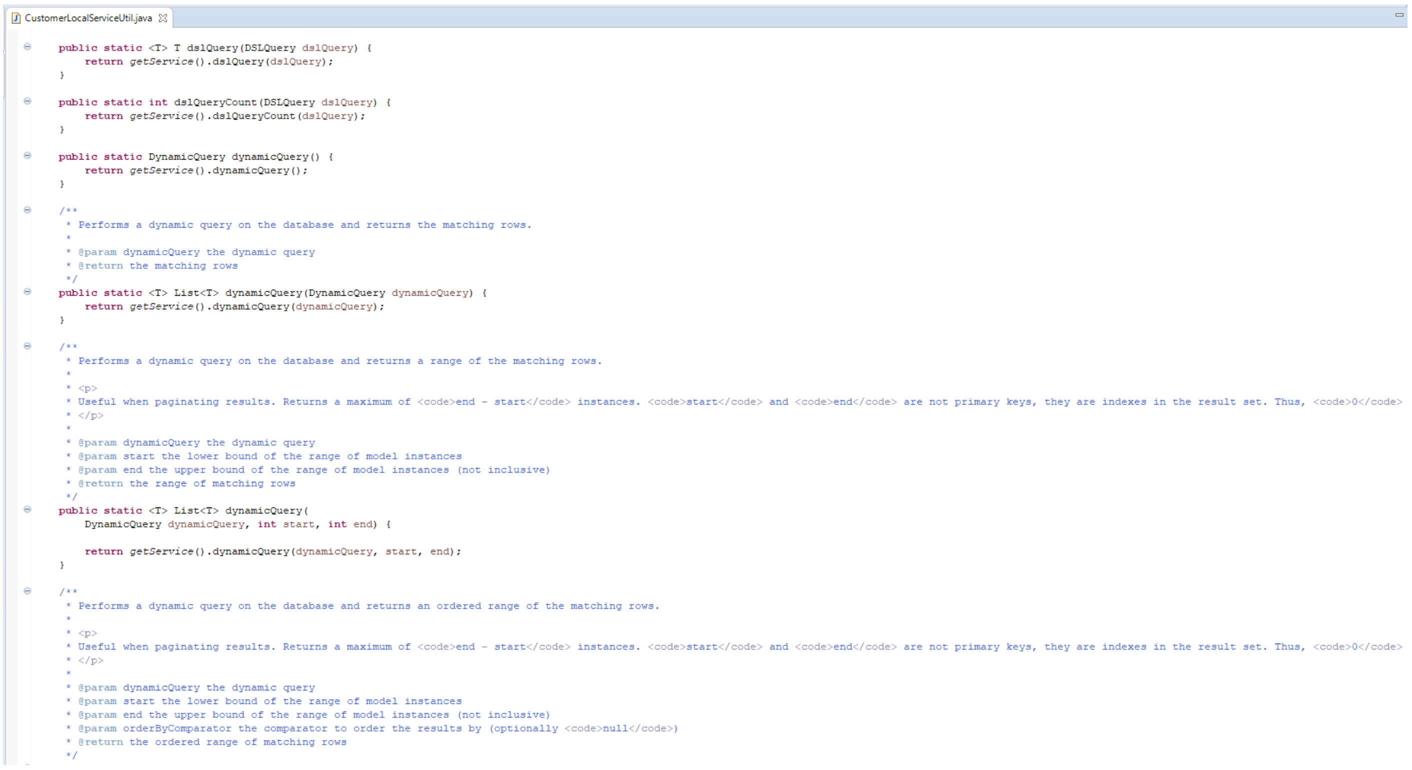


CustomerLocalServiceUtil.java

```
 * Copyright (c) 2000-present Liferay, Inc. All rights reserved.

package AAACustomerServices.service;

import AAACustomerServices.model.Customer;

/**
 * Provides the local service utility for Customer. This utility wraps
 * <code>AAACustomerServices.service.impl.CustomerLocalServiceImpl</code> and
 * is an access point for service operations in application layer code running
 * on the local server. Methods of this service will not have security checks
 * based on the propagated JAAS credentials because this service can only be
 * accessed from within the same VM.
 *
 * @author Brian Wing Shun Chan
 * @see CustomerLocalService
 * @generated
 */
public class CustomerLocalServiceUtil {

    /*
     * NOTE FOR DEVELOPERS:
     *
     * Never modify this class directly. Add custom service methods to <code>AAACustomerServices.service.impl.CustomerLocalServiceImpl</code> and rerun ServiceBuilder to regenerate this class.
     */

    /**
     * Adds the customer to the database. Also notifies the appropriate model listeners.
     *
     * <p>
     * <strong>Important:</strong> Inspect CustomerLocalServiceImpl for overloaded versions of the method. If provided, use these entry points to the API, as the implementation logic may require the additional param
     * </p>
     *
     * @param customer the customer
     * @return the customer that was added
     */
    public static Customer addCustomer(Customer customer) {
        return getService().addCustomer(customer);
    }

    public static Customer addCustomer(
            String customerName, String customerEmail, String customerAddress,
            String customerContact, String customerNRIC, String service)
        throws PortalException, SystemException {

        return getService().addCustomer(
            customerName, customerEmail, customerAddress, customerContact,
            customerNRIC, service);
    }

    /**
     * Creates a new customer with the primary key. Does not add the customer to the database.
     *
     * @param customerId the primary key for the new customer
```

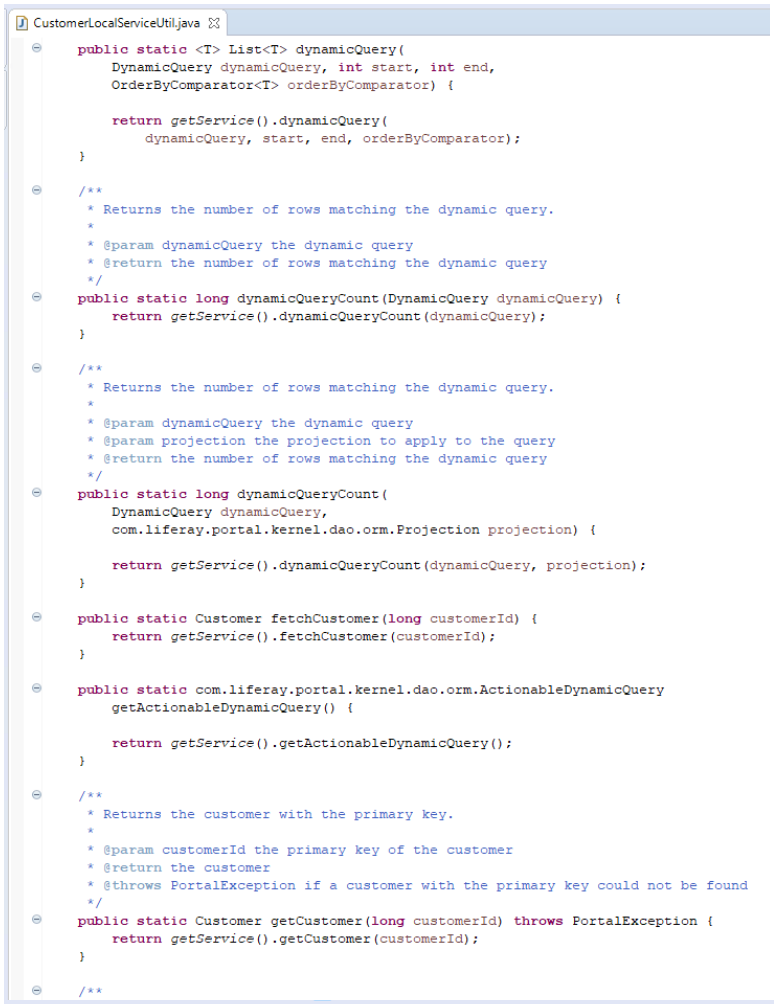


```java
	public static Customer createCustomer(long customerId) {
		return getService().createCustomer(customerId);
	}

	/**
	 * @throws PortalException
	 */
	public static PersistedModel createPersistedModel(
			Serializable primaryKeyObj)
		throws PortalException {

		return getService().createPersistedModel(primaryKeyObj);
	}

	/**
	 * Deletes the customer from the database. Also notifies the appropriate model listeners.
	 *
	 * <p>
	 * <strong>Important:</strong> Inspect CustomerLocalServiceImpl for overloaded versions of the method. If provided, use these entry points to the API, as the implementation logic may require the additional param
	 * </p>
	 *
	 * @param customer the customer
	 * @return the customer that was removed
	 */
	public static Customer deleteCustomer(Customer customer) {
		return getService().deleteCustomer(customer);
	}

	/**
	 * Deletes the customer with the primary key from the database. Also notifies the appropriate model listeners.
	 *
	 * <p>
	 * <strong>Important:</strong> Inspect CustomerLocalServiceImpl for overloaded versions of the method. If provided, use these entry points to the API, as the implementation logic may require the additional param
	 * </p>
	 *
	 * @param customerId the primary key of the customer
	 * @return the customer that was removed
	 * @throws PortalException if a customer with the primary key could not be found
	 */
	public static Customer deleteCustomer(long customerId)
		throws PortalException {

		return getService().deleteCustomer(customerId);
	}

	/**
	 * @throws PortalException
	 */
	public static PersistedModel deletePersistedModel(
			PersistedModel persistedModel)
		throws PortalException {

		return getService().deletePersistedModel(persistedModel);
	}
```



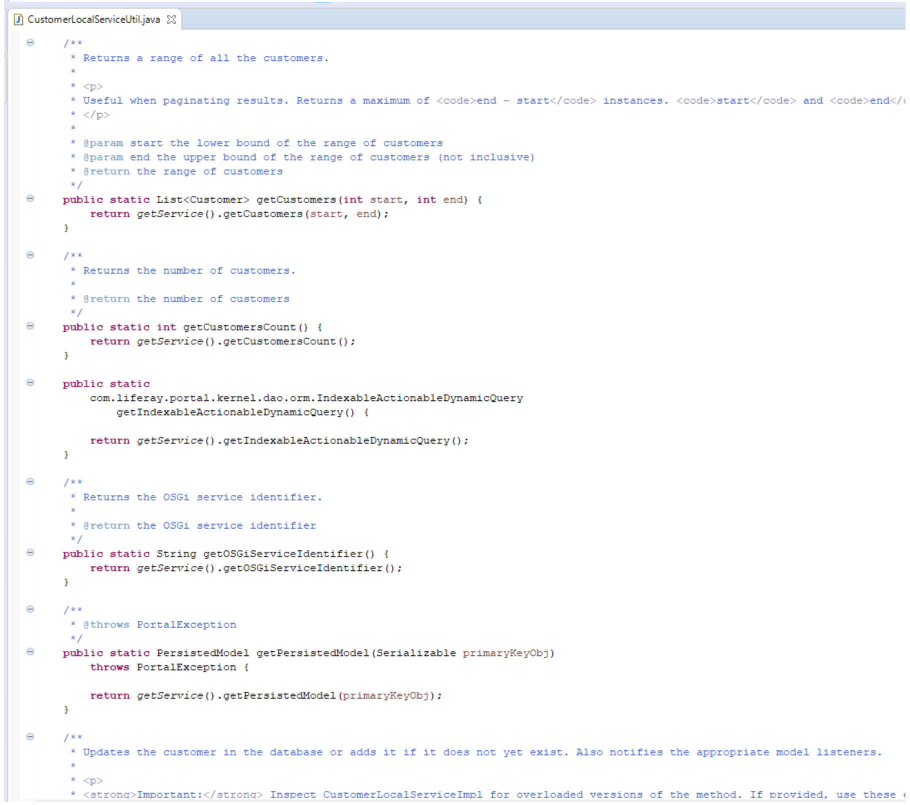
CustomerLocalServiceUtil.java

```java
	public static <T> T dslQuery(DSLQuery dslQuery) {
		return getService().dslQuery(dslQuery);
	}

	public static int dslQueryCount(DSLQuery dslQuery) {
		return getService().dslQueryCount(dslQuery);
	}

	public static DynamicQuery dynamicQuery() {
		return getService().dynamicQuery();
	}

	/**
	 * Performs a dynamic query on the database and returns the matching rows.
	 *
	 * @param dynamicQuery the dynamic query
	 * @return the matching rows
	 */
	public static <T> List<T> dynamicQuery(DynamicQuery dynamicQuery) {
		return getService().dynamicQuery(dynamicQuery);
	}

	/**
	 * Performs a dynamic query on the database and returns a range of the matching rows.
	 *
	 * <p>
	 * Useful when paginating results. Returns a maximum of <code>end - start</code> instances. <code>start</code> and <code>end</code> are not primary keys, they are indexes in the result set. Thus, <code>0</code>
	 * </p>
	 *
	 * @param dynamicQuery the dynamic query
	 * @param start the lower bound of the range of model instances
	 * @param end the upper bound of the range of model instances (not inclusive)
	 * @return the range of matching rows
	 */
	public static <T> List<T> dynamicQuery(
		DynamicQuery dynamicQuery, int start, int end) {

		return getService().dynamicQuery(dynamicQuery, start, end);
	}

	/**
	 * Performs a dynamic query on the database and returns an ordered range of the matching rows.
	 *
	 * <p>
	 * Useful when paginating results. Returns a maximum of <code>end - start</code> instances. <code>start</code> and <code>end</code> are not primary keys, they are indexes in the result set. Thus, <code>0</code>
	 * </p>
	 *
	 * @param dynamicQuery the dynamic query
	 * @param start the lower bound of the range of model instances
	 * @param end the upper bound of the range of model instances (not inclusive)
	 * @param orderByComparator the comparator to order the results by (optionally <code>null</code>)
	 * @return the ordered range of matching rows
	 */
```



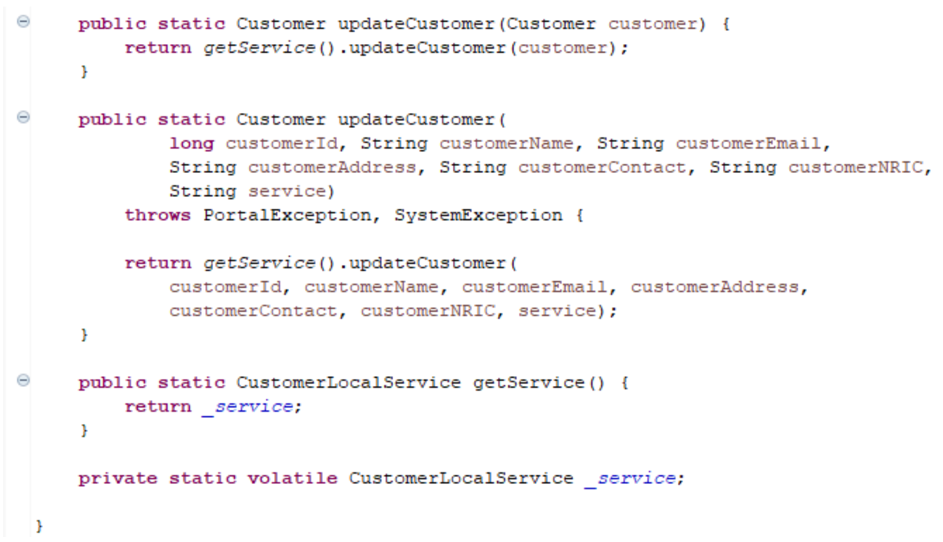
CustomerLocalServiceUtil.java

```java
	public static <T> List<T> dynamicQuery(
		DynamicQuery dynamicQuery, int start, int end,
		OrderByComparator<T> orderByComparator) {

		return getService().dynamicQuery(
			dynamicQuery, start, end, orderByComparator);
	}

	/**
	 * Returns the number of rows matching the dynamic query.
	 *
	 * @param dynamicQuery the dynamic query
	 * @return the number of rows matching the dynamic query
	 */
	public static long dynamicQueryCount(DynamicQuery dynamicQuery) {
		return getService().dynamicQueryCount(dynamicQuery);
	}

	/**
	 * Returns the number of rows matching the dynamic query.
	 *
	 * @param dynamicQuery the dynamic query
	 * @param projection the projection to apply to the query
	 * @return the number of rows matching the dynamic query
	 */
	public static long dynamicQueryCount(
		DynamicQuery dynamicQuery,
		com.liferay.portal.kernel.dao.orm.Projection projection) {

		return getService().dynamicQueryCount(dynamicQuery, projection);
	}

	public static Customer fetchCustomer(long customerId) {
		return getService().fetchCustomer(customerId);
	}

	public static com.liferay.portal.kernel.dao.orm.ActionableDynamicQuery
		getActionableDynamicQuery() {

		return getService().getActionableDynamicQuery();
	}

	/**
	 * Returns the customer with the primary key.
	 *
	 * @param customerId the primary key of the customer
	 * @return the customer
	 * @throws PortalException if a customer with the primary key could not be found
	 */
	public static Customer getCustomer(long customerId) throws PortalException {
		return getService().getCustomer(customerId);
	}

	/**
```



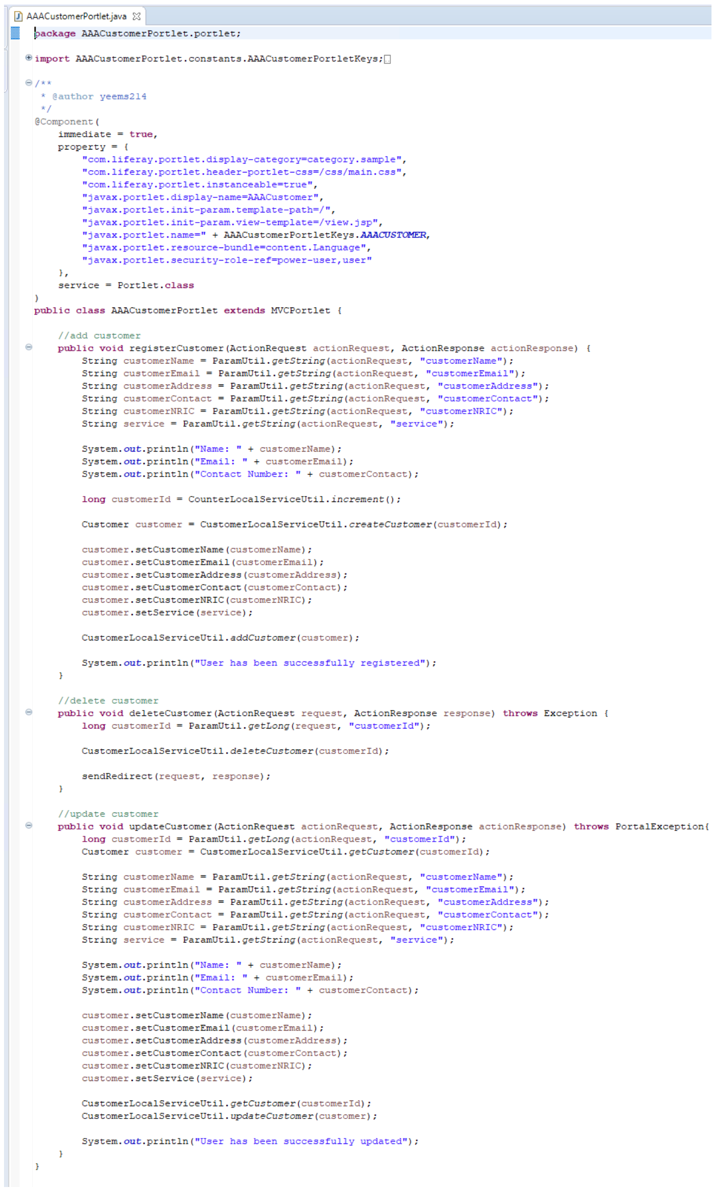
CustomerLocalServiceUtil.java

```java
	/**
	 * Returns a range of all the customers.
	 *
	 * <p>
	 * Useful when paginating results. Returns a maximum of <code>end - start</code> instances. <code>start</code> and <code>end</
	 * </p>
	 *
	 * @param start the lower bound of the range of customers
	 * @param end the upper bound of the range of customers (not inclusive)
	 * @return the range of customers
	 */
	public static List<Customer> getCustomers(int start, int end) {
		return getService().getCustomers(start, end);
	}

	/**
	 * Returns the number of customers.
	 *
	 * @return the number of customers
	 */
	public static int getCustomersCount() {
		return getService().getCustomersCount();
	}

	public static
		com.liferay.portal.kernel.dao.orm.IndexableActionableDynamicQuery
			getIndexableActionableDynamicQuery() {

		return getService().getIndexableActionableDynamicQuery();
	}

	/**
	 * Returns the OSGi service identifier.
	 *
	 * @return the OSGi service identifier
	 */
	public static String getOSGiServiceIdentifier() {
		return getService().getOSGiServiceIdentifier();
	}

	/**
	 * @throws PortalException
	 */
	public static PersistedModel getPersistedModel(Serializable primaryKeyObj)
		throws PortalException {

		return getService().getPersistedModel(primaryKeyObj);
	}

	/**
	 * Updates the customer in the database or adds it if it does not yet exist. Also notifies the appropriate model listeners.
	 *
	 * <p>
	 * <strong>Important:</strong> Inspect CustomerLocalServiceImpl for overloaded versions of the method. If provided, use these
```



```java
    public static Customer updateCustomer(Customer customer) {
        return getService().updateCustomer(customer);
    }

    public static Customer updateCustomer(
            long customerId, String customerName, String customerEmail,
            String customerAddress, String customerContact, String customerNRIC,
            String service)
        throws PortalException, SystemException {

        return getService().updateCustomer(
            customerId, customerName, customerEmail, customerAddress,
            customerContact, customerNRIC, service);
    }

    public static CustomerLocalService getService() {
        return _service;
    }

    private static volatile CustomerLocalService _service;

}
```

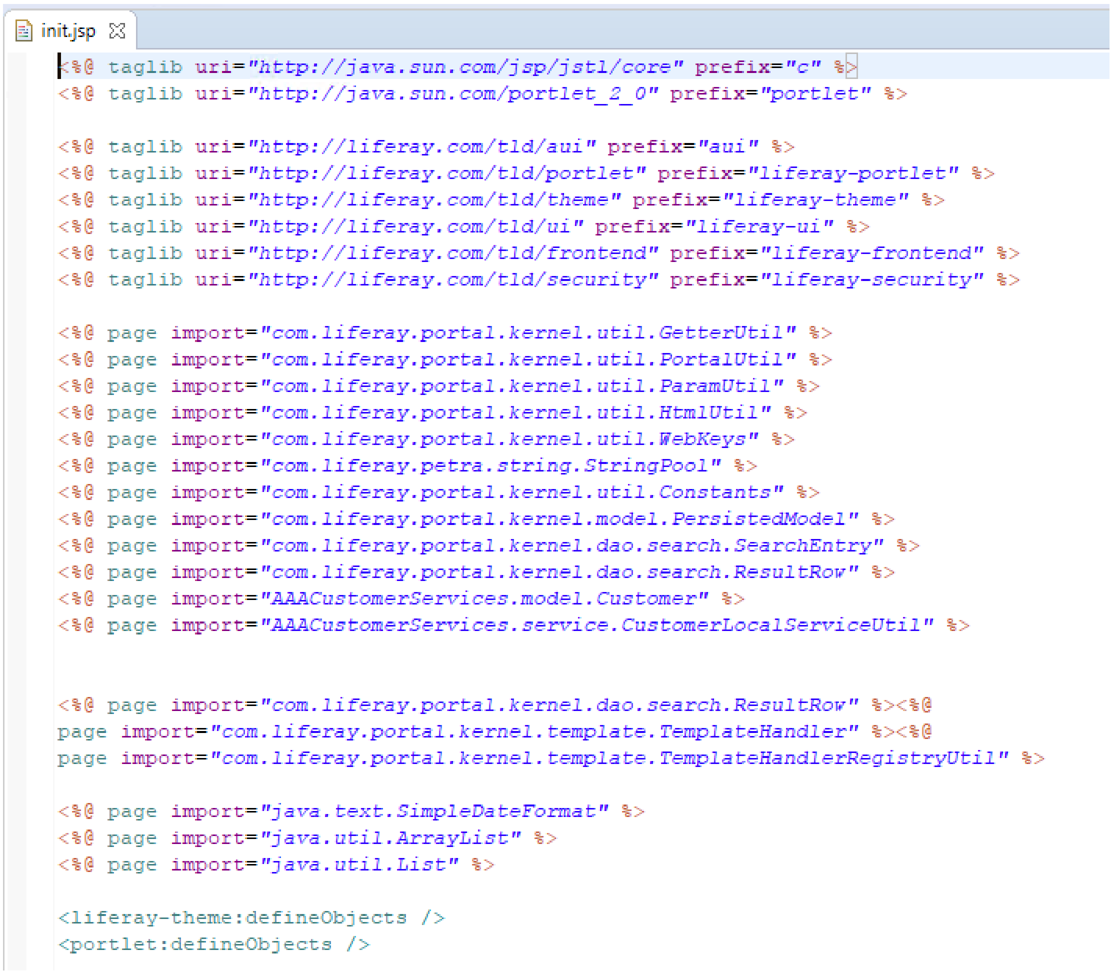
e. Provide screen capture of the controller class.



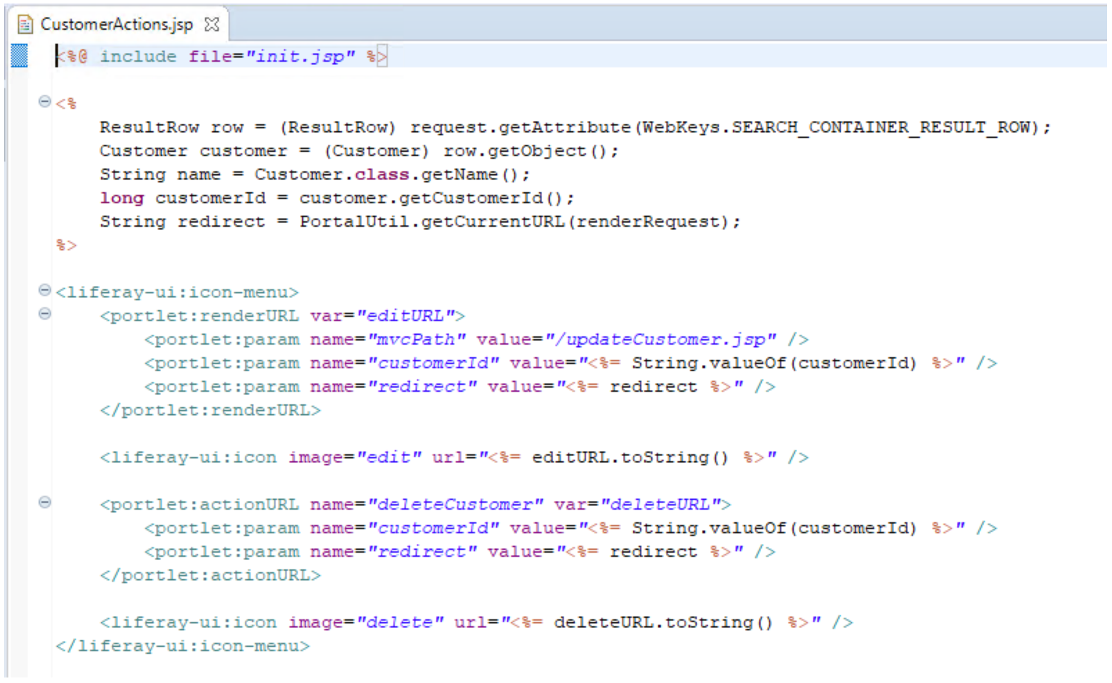
AAACustomerPortlet.java

```java
package AAACustomerPortlet.portlet;

import AAACustomerPortlet.constants.AAACustomerPortletKeys;

/**
 * @author yeems214
 */
@Component(
    immediate = true,
    property = {
        "com.liferay.portlet.display-category=category.sample",
        "com.liferay.portlet.header-portlet-css=/css/main.css",
        "com.liferay.portlet.instanceable=true",
        "javax.portlet.display-name=AAACustomer",
        "javax.portlet.init-param.template-path=/",
        "javax.portlet.init-param.view-template=/view.jsp",
        "javax.portlet.name=" + AAACustomerPortletKeys.AAACUSTOMER,
        "javax.portlet.resource-bundle=content.Language",
        "javax.portlet.security-role-ref=power-user,user"
    },
    service = Portlet.class
)
public class AAACustomerPortlet extends MVCPortlet {

    //add customer
    public void registerCustomer(ActionRequest actionRequest, ActionResponse actionResponse) {
        String customerName = ParamUtil.getString(actionRequest, "customerName");
        String customerEmail = ParamUtil.getString(actionRequest, "customerEmail");
        String customerAddress = ParamUtil.getString(actionRequest, "customerAddress");
        String customerContact = ParamUtil.getString(actionRequest, "customerContact");
        String customerNRIC = ParamUtil.getString(actionRequest, "customerNRIC");
        String service = ParamUtil.getString(actionRequest, "service");

        System.out.println("Name: " + customerName);
        System.out.println("Email: " + customerEmail);
        System.out.println("Contact Number: " + customerContact);

        long customerId = CounterLocalServiceUtil.increment();

        Customer customer = CustomerLocalServiceUtil.createCustomer(customerId);

        customer.setCustomerName(customerName);
        customer.setCustomerEmail(customerEmail);
        customer.setCustomerAddress(customerAddress);
        customer.setCustomerContact(customerContact);
        customer.setCustomerNRIC(customerNRIC);
        customer.setService(service);

        CustomerLocalServiceUtil.addCustomer(customer);

        System.out.println("User has been successfully registered");
    }

    //delete customer
    public void deleteCustomer(ActionRequest request, ActionResponse response) throws Exception {
        long customerId = ParamUtil.getLong(request, "customerId");

        CustomerLocalServiceUtil.deleteCustomer(customerId);

        sendRedirect(request, response);
    }

    //update customer
    public void updateCustomer(ActionRequest actionRequest, ActionResponse actionResponse) throws PortalException{
        long customerId = ParamUtil.getLong(actionRequest, "customerId");
        Customer customer = CustomerLocalServiceUtil.getCustomer(customerId);

        String customerName = ParamUtil.getString(actionRequest, "customerName");
        String customerEmail = ParamUtil.getString(actionRequest, "customerEmail");
        String customerAddress = ParamUtil.getString(actionRequest, "customerAddress");
        String customerContact = ParamUtil.getString(actionRequest, "customerContact");
        String customerNRIC = ParamUtil.getString(actionRequest, "customerNRIC");
        String service = ParamUtil.getString(actionRequest, "service");

        System.out.println("Name: " + customerName);
        System.out.println("Email: " + customerEmail);
        System.out.println("Contact Number: " + customerContact);

        customer.setCustomerName(customerName);
        customer.setCustomerEmail(customerEmail);
        customer.setCustomerAddress(customerAddress);
        customer.setCustomerContact(customerContact);
        customer.setCustomerNRIC(customerNRIC);
        customer.setService(service);

        CustomerLocalServiceUtil.getCustomer(customerId);
        CustomerLocalServiceUtil.updateCustomer(customer);

        System.out.println("User has been successfully updated");
    }
}
```

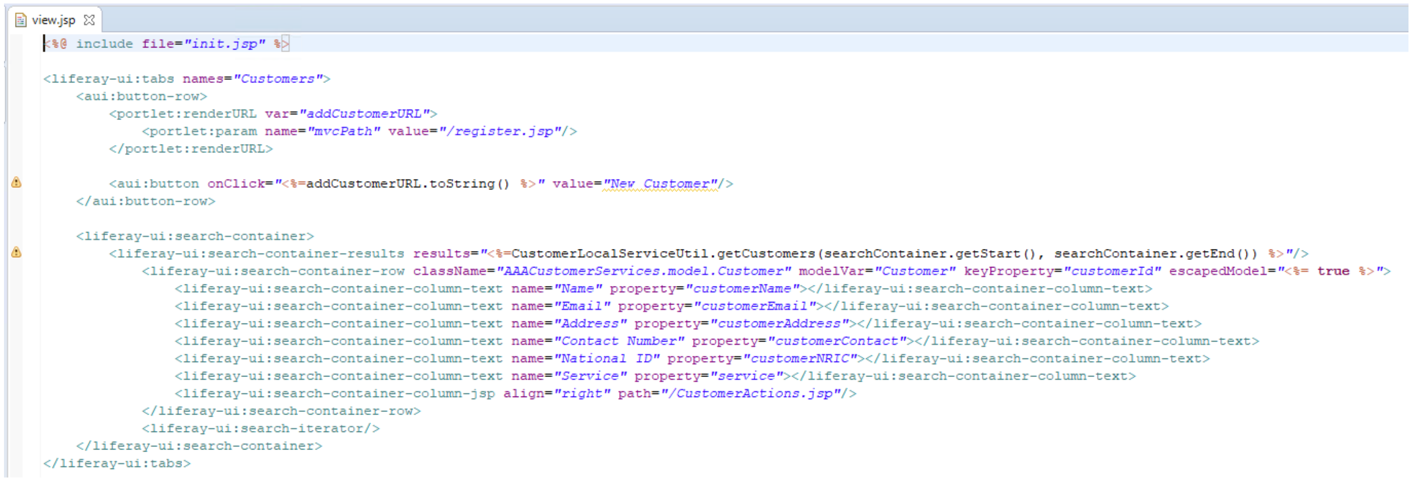
f. Provide screen capture of the required .jsp files for “AAA Customer Portlet”



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>

<%@ taglib uri="http://liferay.com/tld/aui" prefix="aui" %>
<%@ taglib uri="http://liferay.com/tld/portlet" prefix="liferay-portlet" %>
<%@ taglib uri="http://liferay.com/tld/theme" prefix="liferay-theme" %>
<%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>
<%@ taglib uri="http://liferay.com/tld/frontend" prefix="liferay-frontend" %>
<%@ taglib uri="http://liferay.com/tld/security" prefix="liferay-security" %>

<%@ page import="com.liferay.portal.kernel.util.GetterUtil" %>
<%@ page import="com.liferay.portal.kernel.util.PortalUtil" %>
<%@ page import="com.liferay.portal.kernel.util.ParamUtil" %>
<%@ page import="com.liferay.portal.kernel.util.HtmlUtil" %>
<%@ page import="com.liferay.portal.kernel.util.WebKeys" %>
<%@ page import="com.liferay.petra.string.StringPool" %>
<%@ page import="com.liferay.portal.kernel.util.Constants" %>
<%@ page import="com.liferay.portal.kernel.model.PersistedModel" %>
<%@ page import="com.liferay.portal.kernel.dao.search.SearchEntry" %>
<%@ page import="com.liferay.portal.kernel.dao.search.ResultRow" %>
<%@ page import="AAACustomerServices.model.Customer" %>
<%@ page import="AAACustomerServices.service.CustomerLocalServiceUtil" %>


<%@ page import="com.liferay.portal.kernel.dao.search.ResultRow" %><%@
page import="com.liferay.portal.kernel.template.TemplateHandler" %><%@
page import="com.liferay.portal.kernel.template.TemplateHandlerRegistryUtil" %>

<%@ page import="java.text.SimpleDateFormat" %>
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.List" %>

<liferay-theme:defineObjects />
<portlet:defineObjects />
```

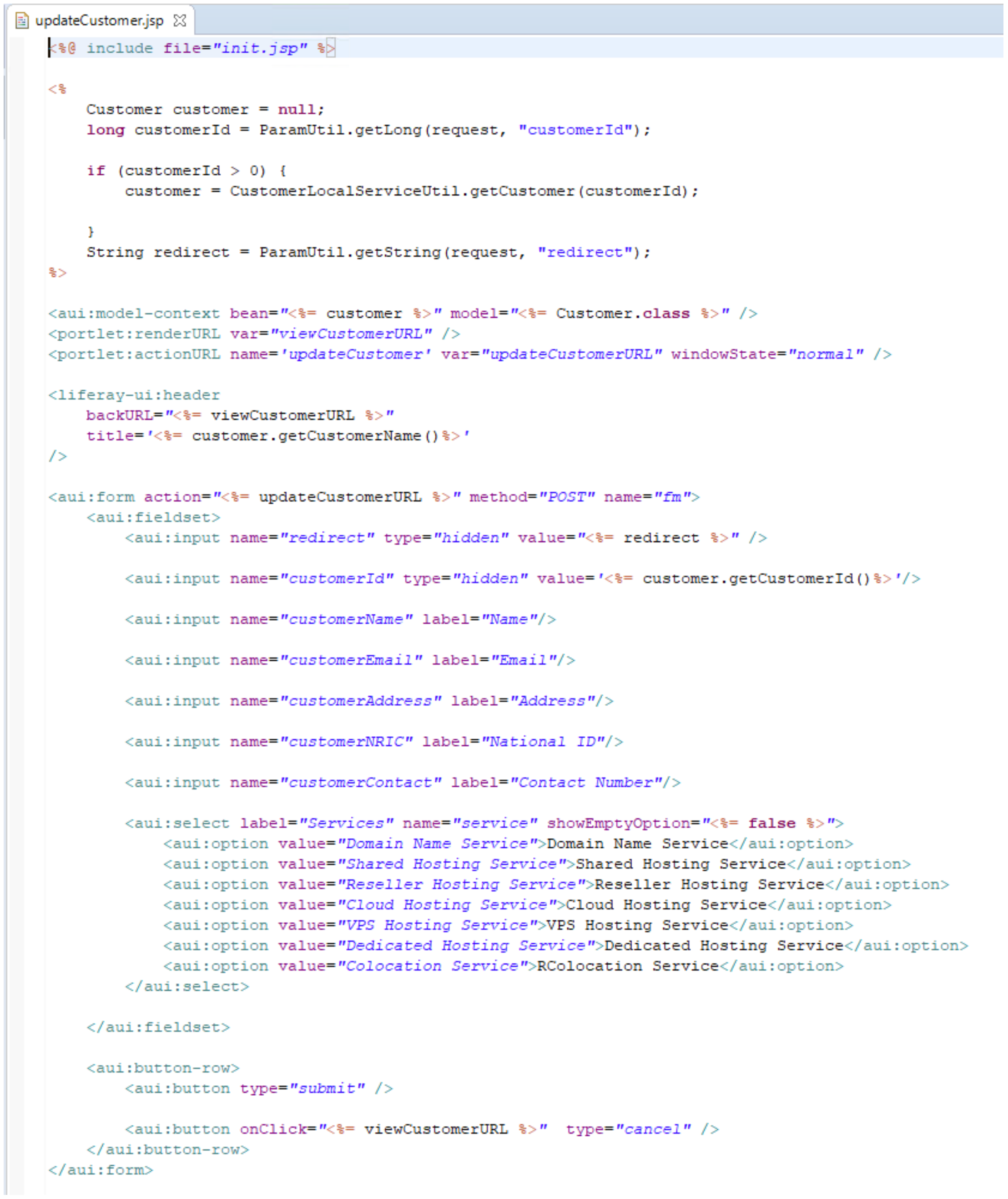


```
<%@ include file="init.jsp" %>

<%
    ResultRow row = (ResultRow) request.getAttribute(WebKeys.SEARCH_CONTAINER_RESULT_ROW);
    Customer customer = (Customer) row.getObject();
    String name = Customer.class.getName();
    long customerId = customer.getCustomerId();
    String redirect = PortalUtil.getCurrentURL(renderRequest);
%>

<liferay-ui:icon-menu>
    <portlet:renderURL var="editURL">
        <portlet:param name="mvcPath" value="/updateCustomer.jsp" />
        <portlet:param name="customerId" value="<%= String.valueOf(customerId) %>" />
        <portlet:param name="redirect" value="<%= redirect %>" />
    </portlet:renderURL>

    <liferay-ui:icon image="edit" url="<%= editURL.toString() %>" />

    <portlet:actionURL name="deleteCustomer" var="deleteURL">
        <portlet:param name="customerId" value="<%= String.valueOf(customerId) %>" />
        <portlet:param name="redirect" value="<%= redirect %>" />
    </portlet:actionURL>

    <liferay-ui:icon image="delete" url="<%= deleteURL.toString() %>" />
</liferay-ui:icon-menu>
```



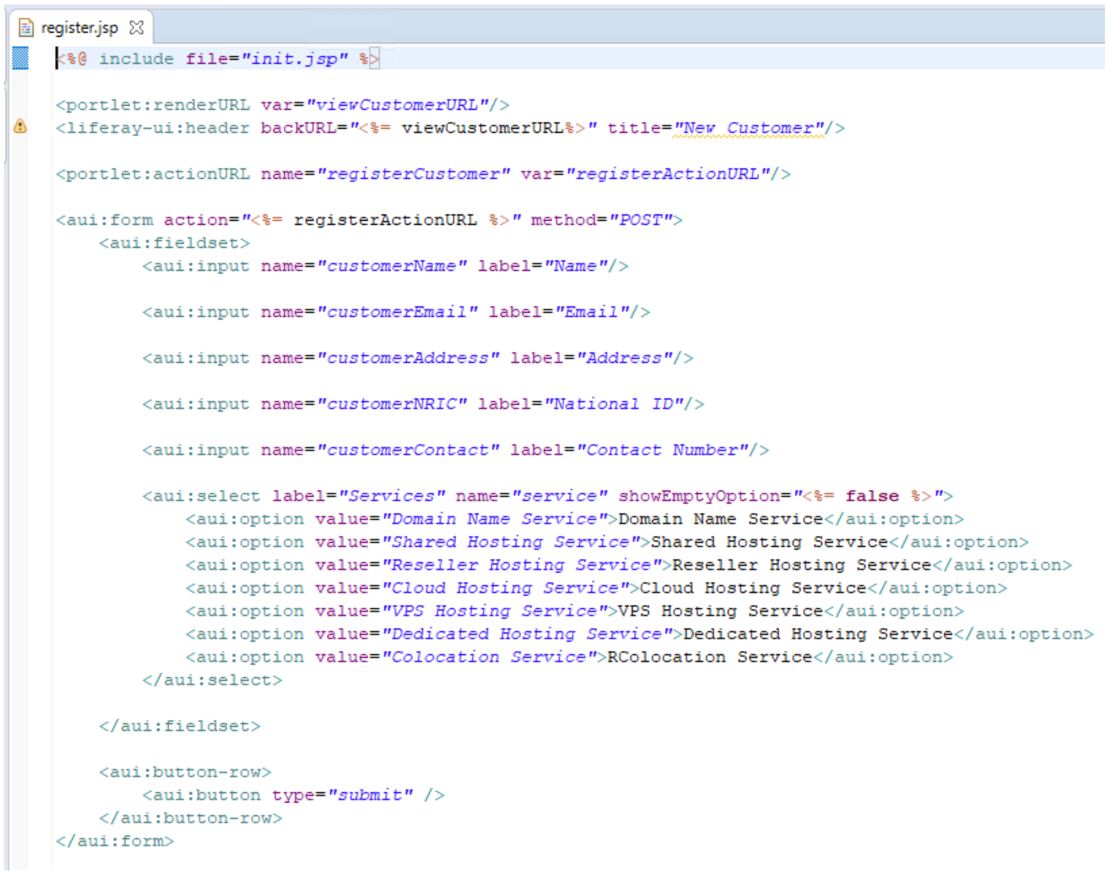
view.jsp

```
<%@ include file="init.jsp" %>

<liferay-ui:tabs names="Customers">
    <aui:button-row>
        <portlet:renderURL var="addCustomerURL">
            <portlet:param name="mvcPath" value="/register.jsp"/>
        </portlet:renderURL>

        <aui:button onClick="<%=addCustomerURL.toString() %>" value="New Customer"/>
    </aui:button-row>

    <liferay-ui:search-container>
        <liferay-ui:search-container-results results="<%=CustomerLocalServiceUtil.getCustomers(searchContainer.getStart(), searchContainer.getEnd()) %>"/>
            <liferay-ui:search-container-row className="AAACustomerServices.model.Customer" modelVar="Customer" keyProperty="customerId" escapedModel="<%= true %>">
                <liferay-ui:search-container-column-text name="Name" property="customerName"></liferay-ui:search-container-column-text>
                <liferay-ui:search-container-column-text name="Email" property="customerEmail"></liferay-ui:search-container-column-text>
                <liferay-ui:search-container-column-text name="Address" property="customerAddress"></liferay-ui:search-container-column-text>
                <liferay-ui:search-container-column-text name="Contact Number" property="customerContact"></liferay-ui:search-container-column-text>
                <liferay-ui:search-container-column-text name="National ID" property="customerNRIC"></liferay-ui:search-container-column-text>
                <liferay-ui:search-container-column-text name="Service" property="service"></liferay-ui:search-container-column-text>
                <liferay-ui:search-container-column-jsp align="right" path="/CustomerActions.jsp"/>
            </liferay-ui:search-container-row>
            <liferay-ui:search-iterator/>
    </liferay-ui:search-container>
</liferay-ui:tabs>
```



updateCustomer.jsp

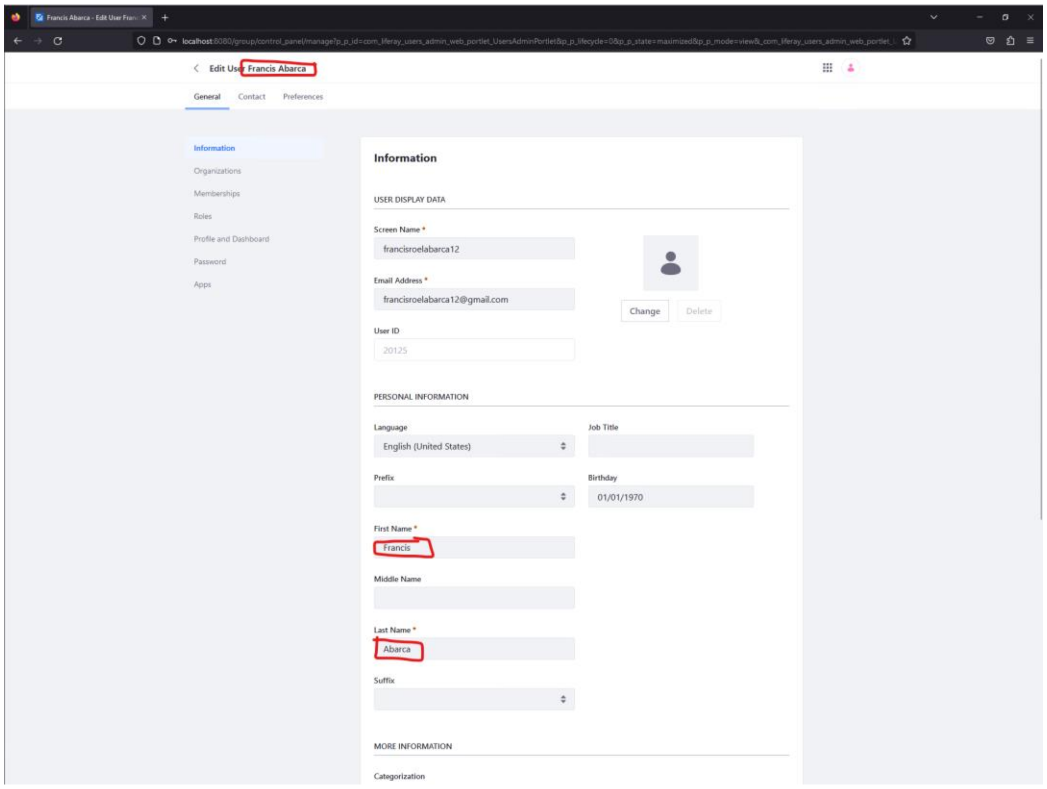
```
<%@ include file="init.jsp" %>

<%
    Customer customer = null;
    long customerId = ParamUtil.getLong(request, "customerId");

    if (customerId > 0) {
        customer = CustomerLocalServiceUtil.getCustomer(customerId);

    }
    String redirect = ParamUtil.getString(request, "redirect");
%>

<aui:model-context bean="<%= customer %>" model="<%= Customer.class %>" />
<portlet:renderURL var="viewCustomerURL" />
<portlet:actionURL name='updateCustomer' var="updateCustomerURL" windowState="normal" />

<liferay-ui:header
    backURL="<%= viewCustomerURL %>"
    title='<%= customer.getCustomerName()%>'
/>

<aui:form action="<%= updateCustomerURL %>" method="POST" name="fm">
    <aui:fieldset>
        <aui:input name="redirect" type="hidden" value="<%= redirect %>" />

        <aui:input name="customerId" type="hidden" value='<%= customer.getCustomerId()%>'/>

        <aui:input name="customerName" label="Name"/>

        <aui:input name="customerEmail" label="Email"/>

        <aui:input name="customerAddress" label="Address"/>

        <aui:input name="customerNRIC" label="National ID"/>

        <aui:input name="customerContact" label="Contact Number"/>

        <aui:select label="Services" name="service" showEmptyOption="<%= false %>">
            <aui:option value="Domain Name Service">Domain Name Service</aui:option>
            <aui:option value="Shared Hosting Service">Shared Hosting Service</aui:option>
            <aui:option value="Reseller Hosting Service">Reseller Hosting Service</aui:option>
            <aui:option value="Cloud Hosting Service">Cloud Hosting Service</aui:option>
            <aui:option value="VPS Hosting Service">VPS Hosting Service</aui:option>
            <aui:option value="Dedicated Hosting Service">Dedicated Hosting Service</aui:option>
            <aui:option value="Colocation Service">RColocation Service</aui:option>
        </aui:select>

    </aui:fieldset>

    <aui:button-row>
        <aui:button type="submit" />

        <aui:button onClick="<%= viewCustomerURL %>"  type="cancel" />
    </aui:button-row>
</aui:form>
```



register.jsp

```
<%@ include file="init.jsp" %>

<portlet:renderURL var="viewCustomerURL"/>
<liferay-ui:header backURL="<%= viewCustomerURL%>" title="New Customer"/>

<portlet:actionURL name="registerCustomer" var="registerActionURL"/>

<aui:form action="<%= registerActionURL %>" method="POST">
    <aui:fieldset>
        <aui:input name="customerName" label="Name"/>

        <aui:input name="customerEmail" label="Email"/>

        <aui:input name="customerAddress" label="Address"/>

        <aui:input name="customerNRIC" label="National ID"/>

        <aui:input name="customerContact" label="Contact Number"/>

        <aui:select label="Services" name="service" showEmptyOption="<%= false %>">
            <aui:option value="Domain Name Service">Domain Name Service</aui:option>
            <aui:option value="Shared Hosting Service">Shared Hosting Service</aui:option>
            <aui:option value="Reseller Hosting Service">Reseller Hosting Service</aui:option>
            <aui:option value="Cloud Hosting Service">Cloud Hosting Service</aui:option>
            <aui:option value="VPS Hosting Service">VPS Hosting Service</aui:option>
            <aui:option value="Dedicated Hosting Service">Dedicated Hosting Service</aui:option>
            <aui:option value="Colocation Service">RColocation Service</aui:option>
        </aui:select>

    </aui:fieldset>

    <aui:button-row>
        <aui:button type="submit" />
    </aui:button-row>
</aui:form>
```

g. Create the AAA Services webpages. Take screen captures of the following pages along with your Admin Account Name (Make sure that admin must be registered with Your Name)



Francis Abarca - Edit User Fran

localhost:8080/group/control_panel/manage?p_p_id=com_liferay_users_admin_web_portlet_UsersAdminPortlet&p_p_lifecycle=0&p_p_state=maximized&p_p_mode=view&_com_liferay_users_admin_web_portlet_

Edit User Francis Abarca

General | Contact | Preferences

Information

USER DISPLAY DATA

Screen Name *
francisroelabarca12

Email Address *
francisroelabarca12@gmail.com

User ID
20125

PERSONAL INFORMATION

Language
English (United States)

Job Title

Prefix

Birthday
01/01/1970

First Name *
Francis

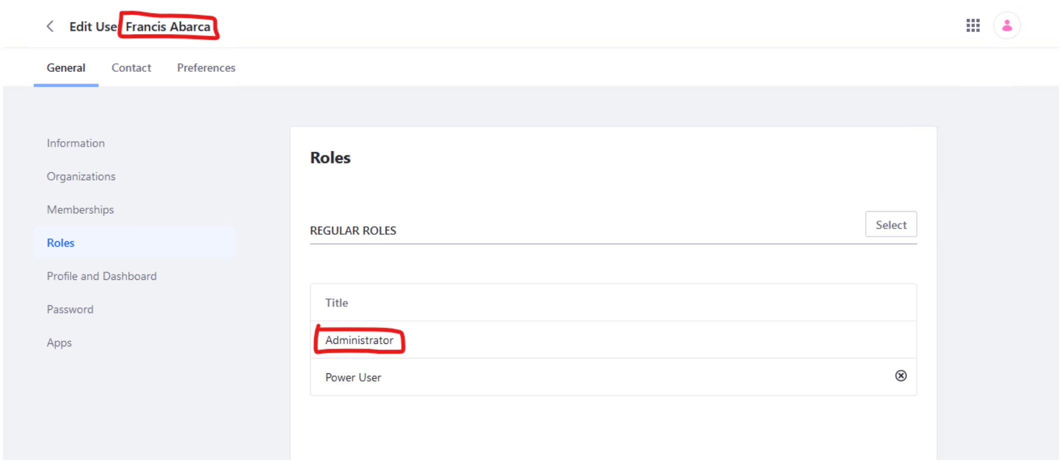
Middle Name

Last Name *
Abarca

Suffix

MORE INFORMATION

Categorization



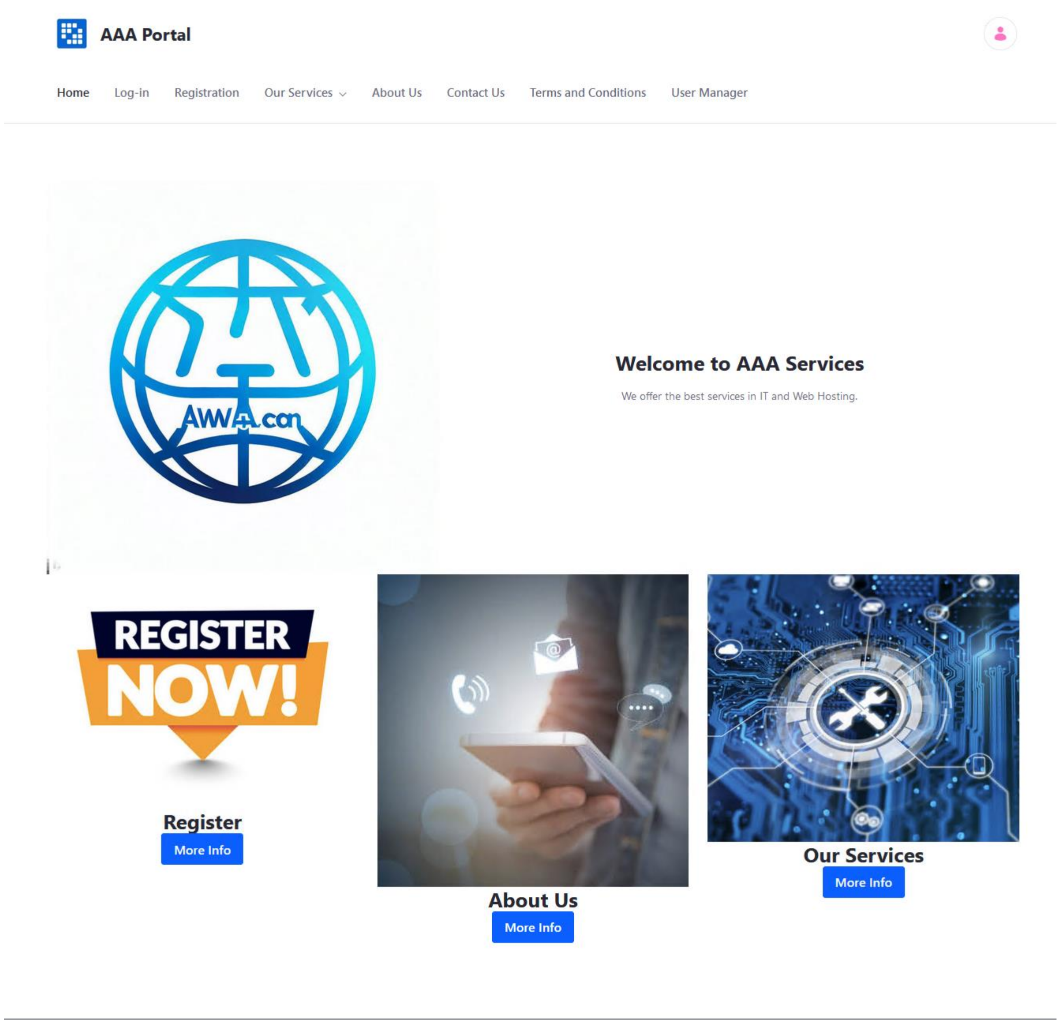
Edit User Francis Abarca

Roles

REGULAR ROLES

Title
Administrator
Power User

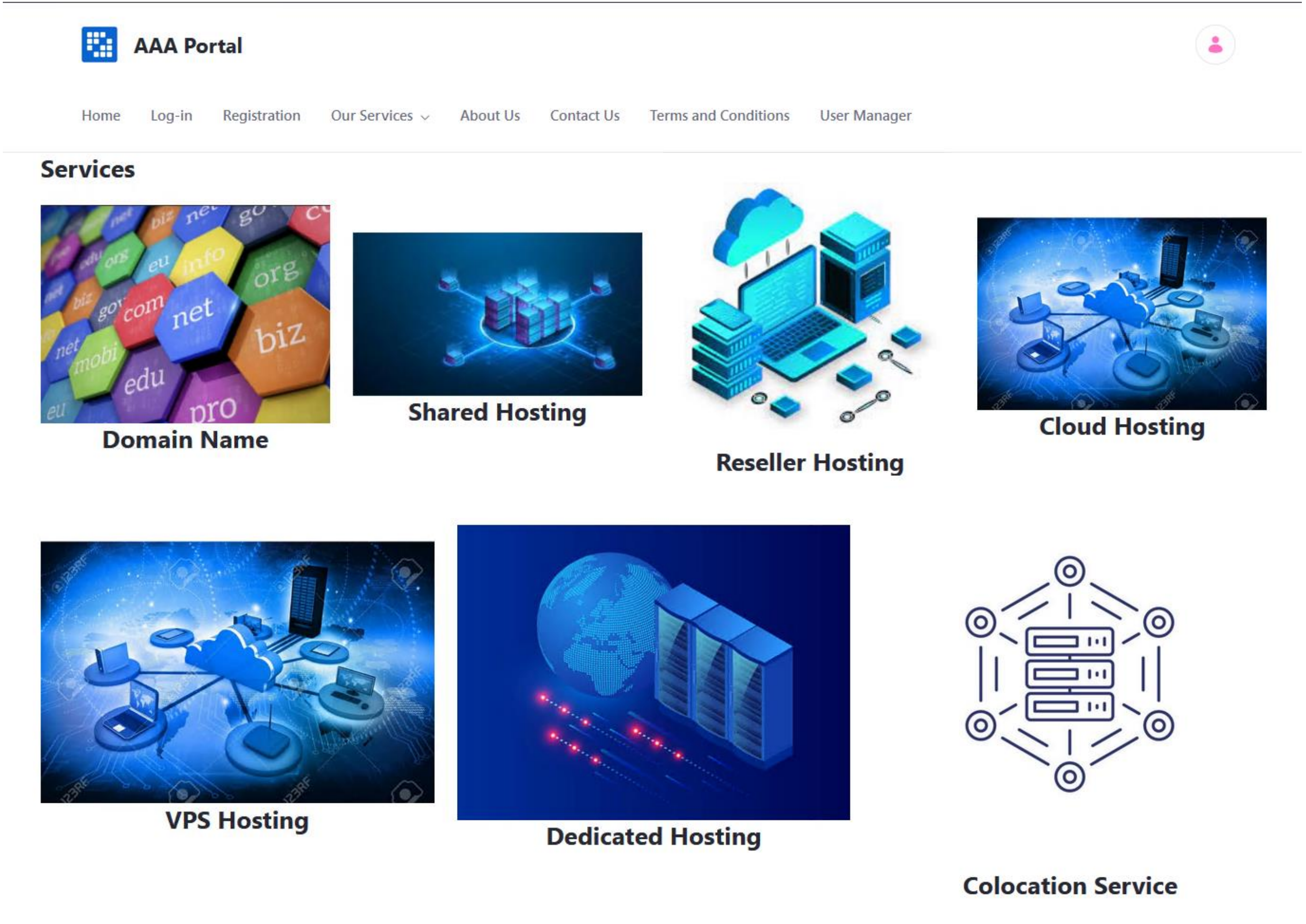
i. Home Page



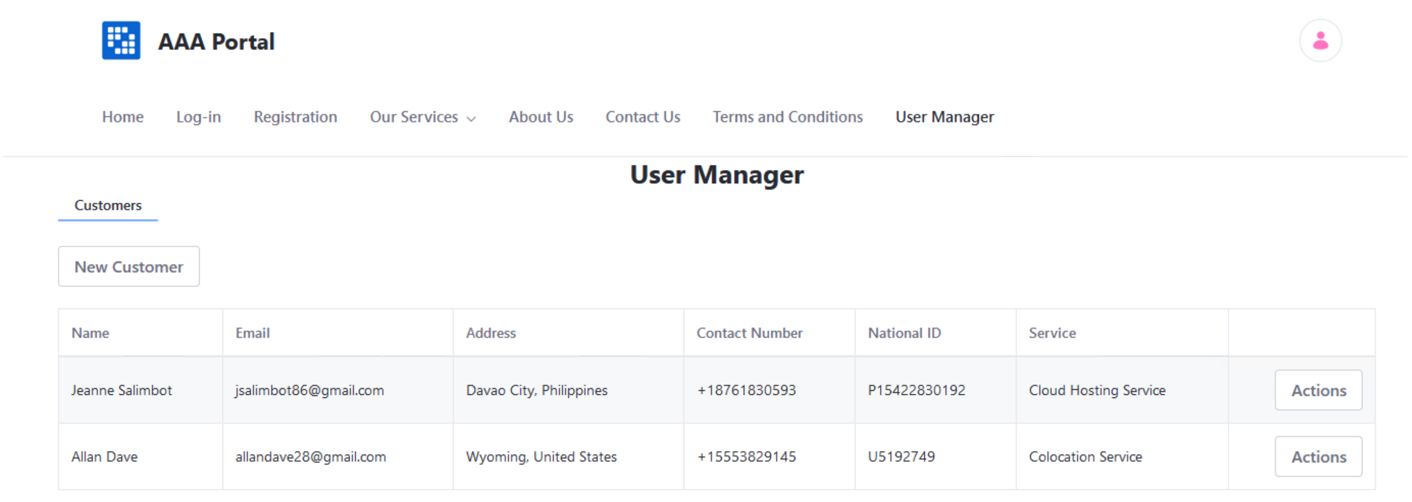
Welcome to AAA Services

We offer the best services in IT and Web Hosting.

ii. Our Services Page

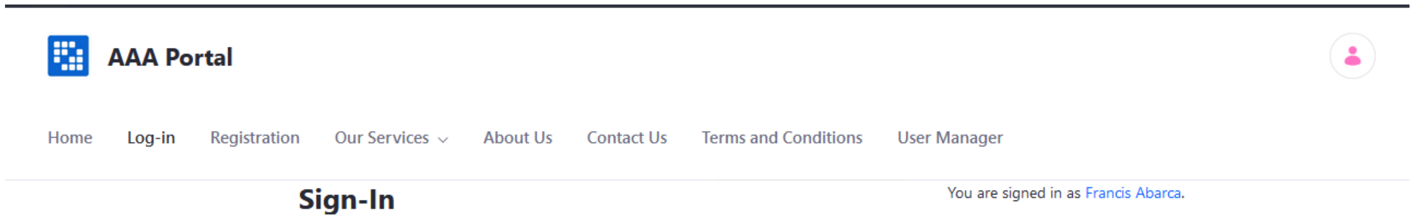


iii. Customer Management



Name	Email	Address	Contact Number	National ID	Service	
Jeanne Salimbot	jsalimbot86@gmail.com	Davao City, Philippines	+18761830593	P15422830192	Cloud Hosting Service	Actions
Allan Dave	allandave28@gmail.com	Wyoming, United States	+15553829145	U5192749	Colocation Service	Actions

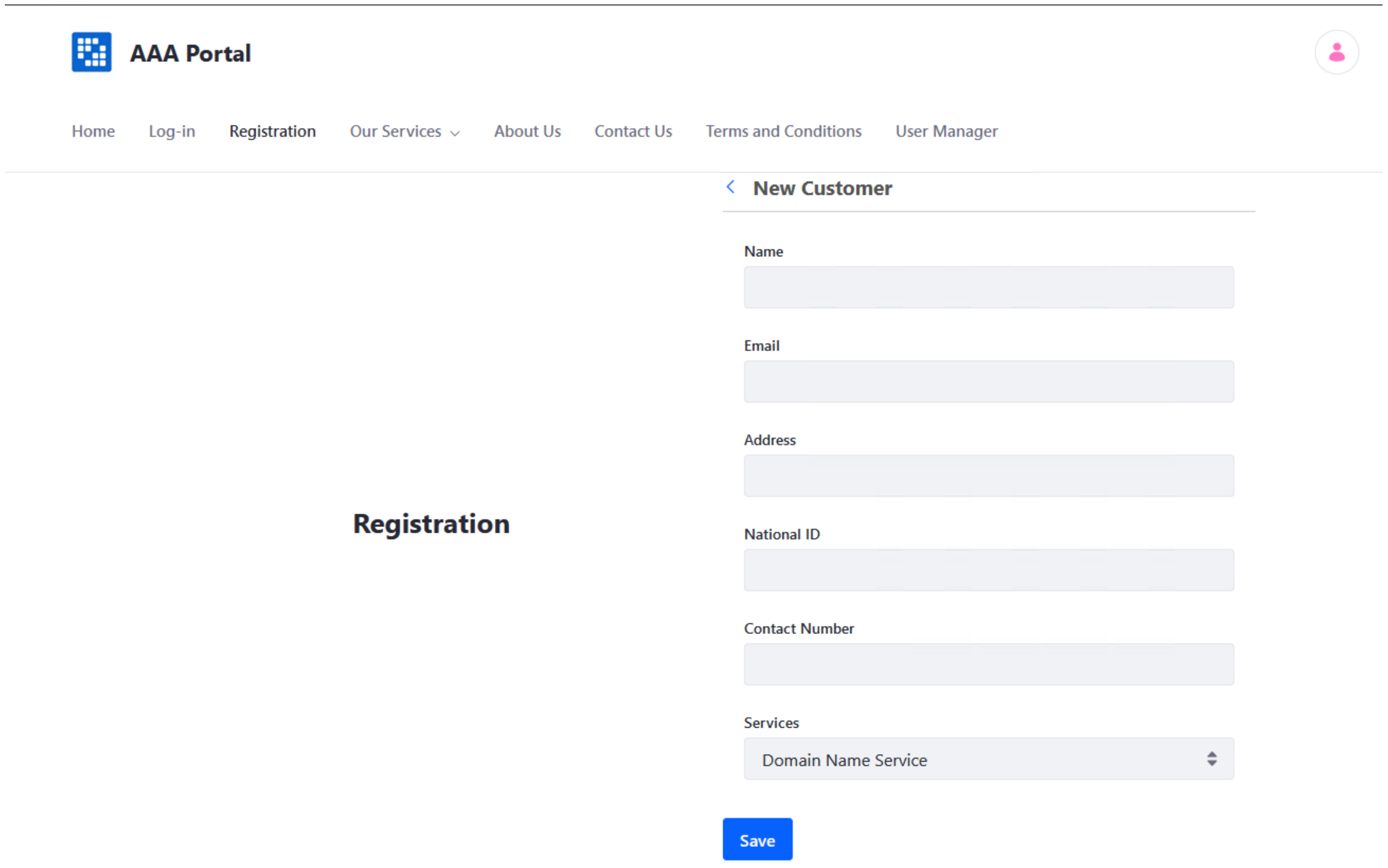
iv. Login Page



Sign-In

You are signed in as Francis Abarca.

v. Registration Page

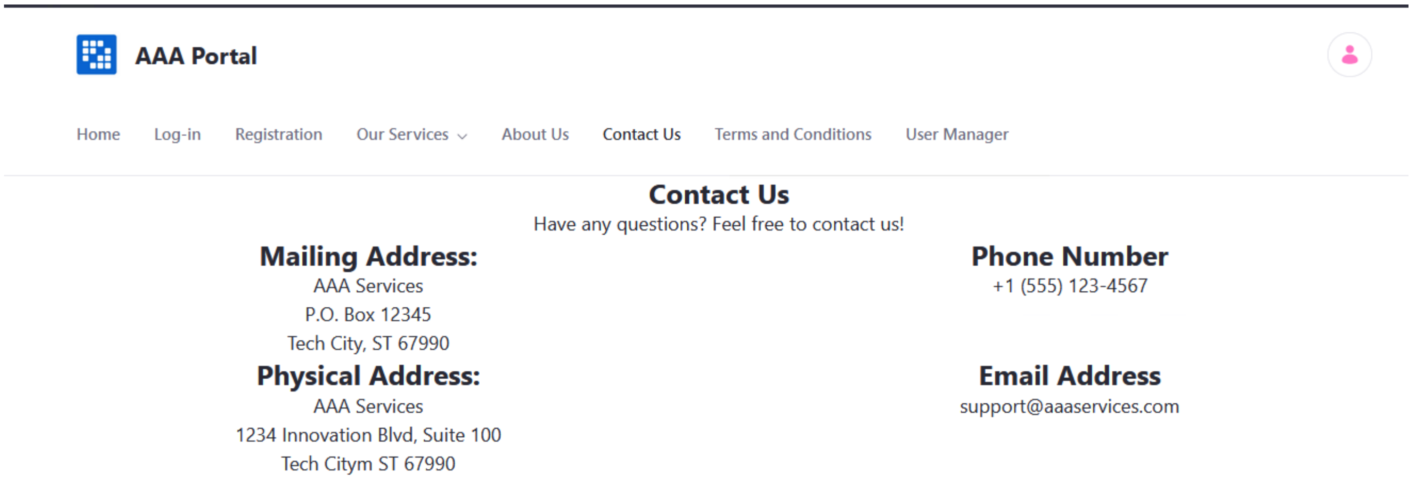


Registration

New Customer

Name	
Email	
Address	
National ID	
Contact Number	
Services	Domain Name Service

vi. Contact Us Page



Contact Us

Have any questions? Feel free to contact us!

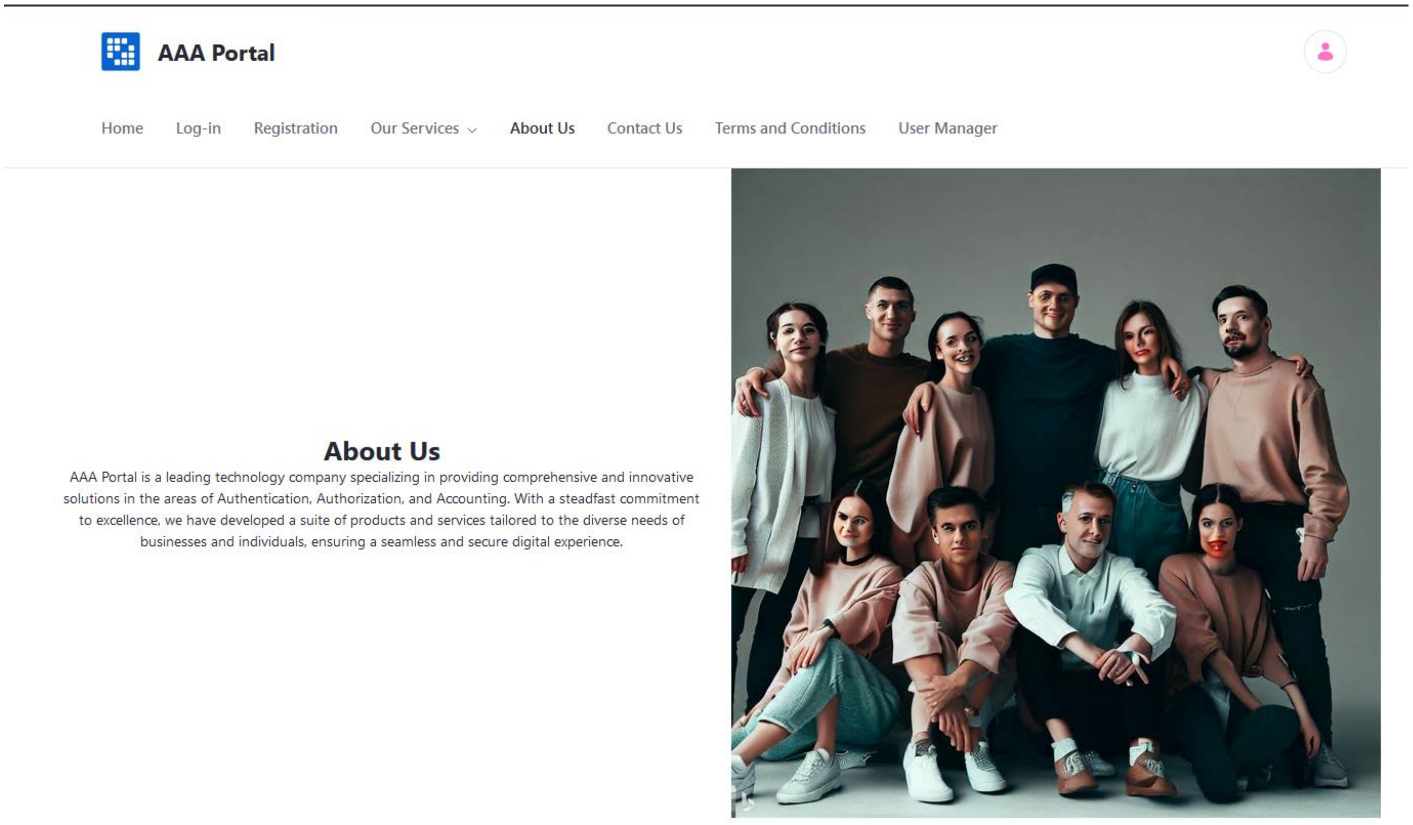
Mailing Address:
AAA Services
P.O. Box 12345
Tech City, ST 67990

Physical Address:
AAA Services
1234 Innovation Blvd, Suite 100
Tech Citym ST 67990

Phone Number
+1 (555) 123-4567

Email Address
support@aaaservices.com

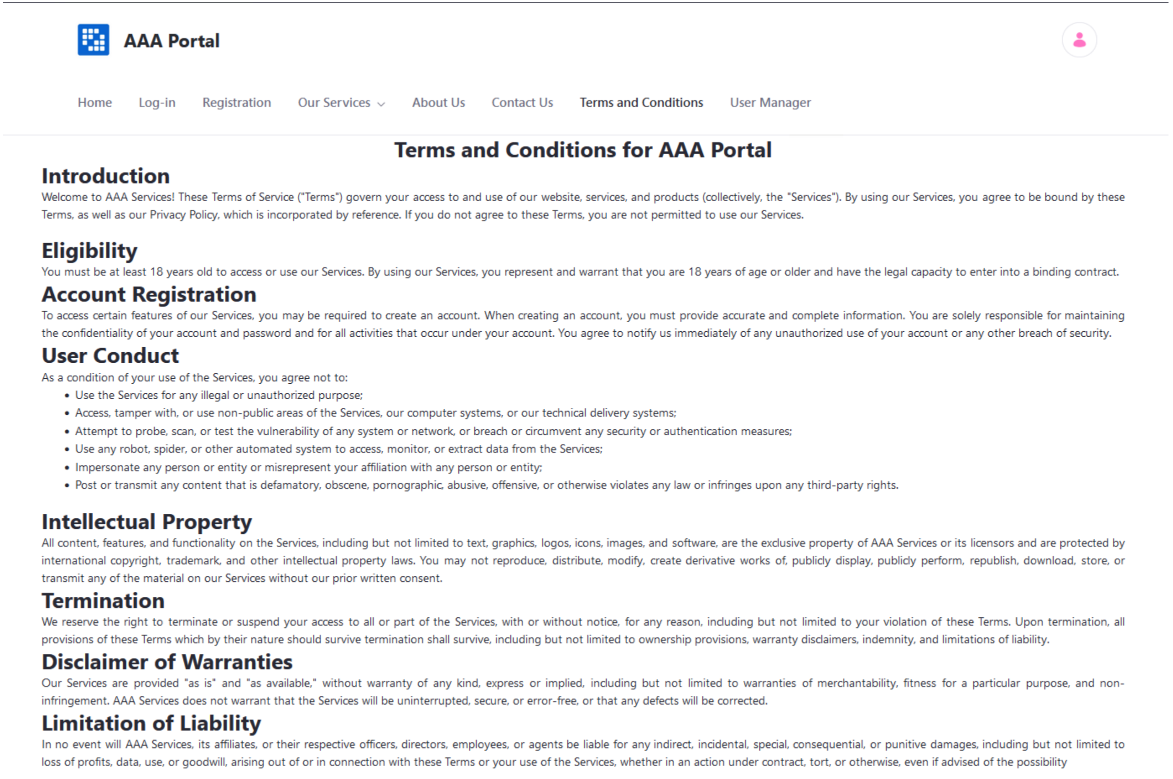
vii. About Us Page



About Us

AAA Portal is a leading technology company specializing in providing comprehensive and innovative solutions in the areas of Authentication, Authorization, and Accounting. With a steadfast commitment to excellence, we have developed a suite of products and services tailored to the diverse needs of businesses and individuals, ensuring a seamless and secure digital experience.

viii. Terms and Conditions Page



AAA Portal

Home Log-in Registration Our Services About Us Contact Us Terms and Conditions User Manager

Terms and Conditions for AAA Portal

Introduction

Welcome to AAA Services! These Terms of Service ("Terms") govern your access to and use of our website, services, and products (collectively, the "Services"). By using our Services, you agree to be bound by these Terms, as well as our Privacy Policy, which is incorporated by reference. If you do not agree to these Terms, you are not permitted to use our Services.

Eligibility

You must be at least 18 years old to access or use our Services. By using our Services, you represent and warrant that you are 18 years of age or older and have the legal capacity to enter into a binding contract.

Account Registration

To access certain features of our Services, you may be required to create an account. When creating an account, you must provide accurate and complete information. You are solely responsible for maintaining the confidentiality of your account and password and for all activities that occur under your account. You agree to notify us immediately of any unauthorized use of your account or any other breach of security.

User Conduct

As a condition of your use of the Services, you agree not to:

- Use the Services for any illegal or unauthorized purpose;
- Access, tamper with, or use non-public areas of the Services, our computer systems, or our technical delivery systems;
- Attempt to probe, scan, or test the vulnerability of any system or network, or breach or circumvent any security or authentication measures;
- Use any robot, spider, or other automated system to access, monitor, or extract data from the Services;
- Impersonate any person or entity or misrepresent your affiliation with any person or entity;
- Post or transmit any content that is defamatory, obscene, pornographic, abusive, offensive, or otherwise violates any law or infringes upon any third-party rights.

Intellectual Property

All content, features, and functionality on the Services, including but not limited to text, graphics, logos, icons, images, and software, are the exclusive property of AAA Services or its licensors and are protected by international copyright, trademark, and other intellectual property laws. You may not reproduce, distribute, modify, create derivative works of, publicly display, publicly perform, republish, download, store, or transmit any of the material on our Services without our prior written consent.

Termination

We reserve the right to terminate or suspend your access to all or part of the Services, with or without notice, for any reason, including but not limited to your violation of these Terms. Upon termination, all provisions of these Terms which by their nature should survive termination shall survive, including but not limited to ownership provisions, warranty disclaimers, indemnity, and limitations of liability.

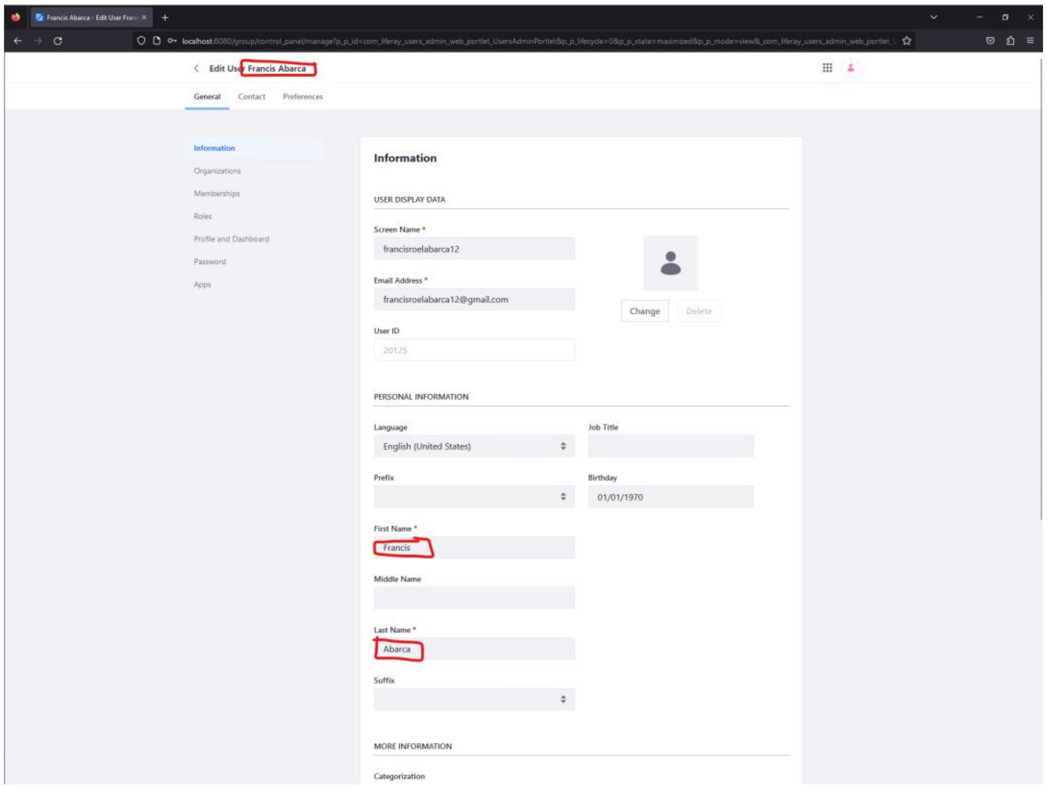
Disclaimer of Warranties

Our Services are provided "as is" and "as available," without warranty of any kind, express or implied, including but not limited to warranties of merchantability, fitness for a particular purpose, and non-infringement. AAA Services does not warrant that the Services will be uninterrupted, secure, or error-free, or that any defects will be corrected.

Limitation of Liability

In no event will AAA Services, its affiliates, or their respective officers, directors, employees, or agents be liable for any indirect, incidental, special, consequential, or punitive damages, including but not limited to loss of profits, data, use, or goodwill, arising out of or in connection with these Terms or your use of the Services, whether in an action under contract, tort, or otherwise, even if advised of the possibility

h. Provide screen capture logging in with two different users (Administrator, Site Member) showing permission for customer portlet



Francis Abarca - Edit User Fran

localhost:8080/group/control_panel/manage?p_p_id=com_liferay_users_admin_web_portlet_UsersAdminPortlet&p_p_lifecycle=0&p_p_state=maximized&p_p_mode=view&_com_liferay_users_admin_web_portlet_

Edit User Francis Abarca

General Contact Preferences

Information
Organizations
Memberships
Roles
Profile and Dashboard
Password
Apps

Information

USER DISPLAY DATA

Screen Name *	francisroelabarca12
Email Address *	francisroelabarca12@gmail.com
User ID	20125

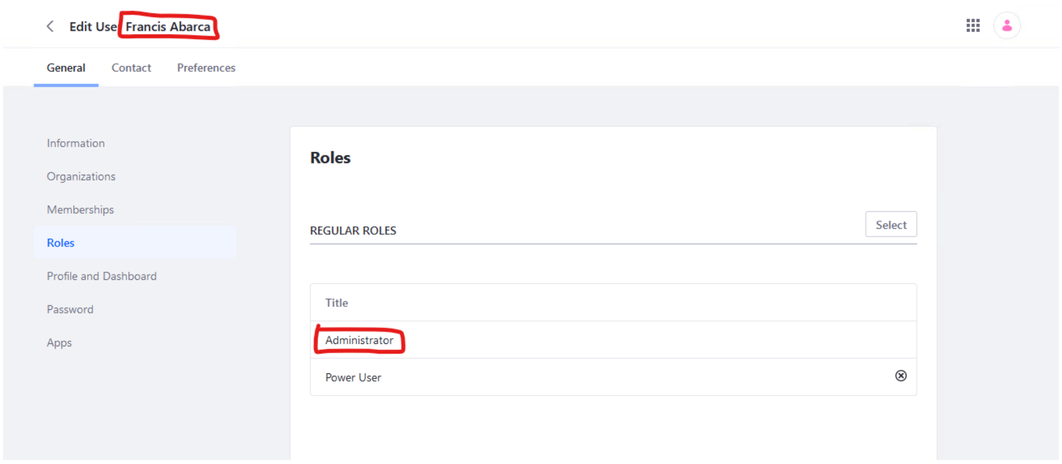
Change Delete

PERSONAL INFORMATION

Language	English (United States)
Job Title	
Prefix	
Birthday	01/01/1970
First Name *	Francis
Middle Name	
Last Name *	Abarca
Suffix	

MORE INFORMATION

Categorization



Edit User Francis Abarca

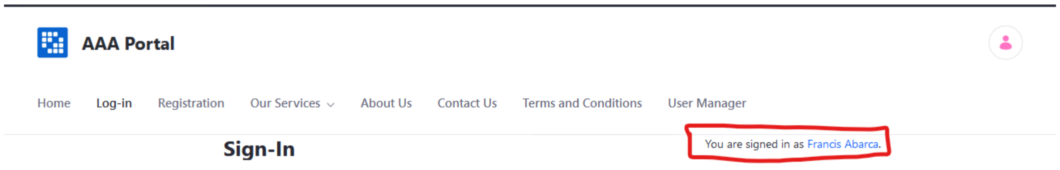
General Contact Preferences

Information
Organizations
Memberships
Roles
Profile and Dashboard
Password
Apps

Roles

REGULAR ROLES — Select

Title
Administrator
Power User



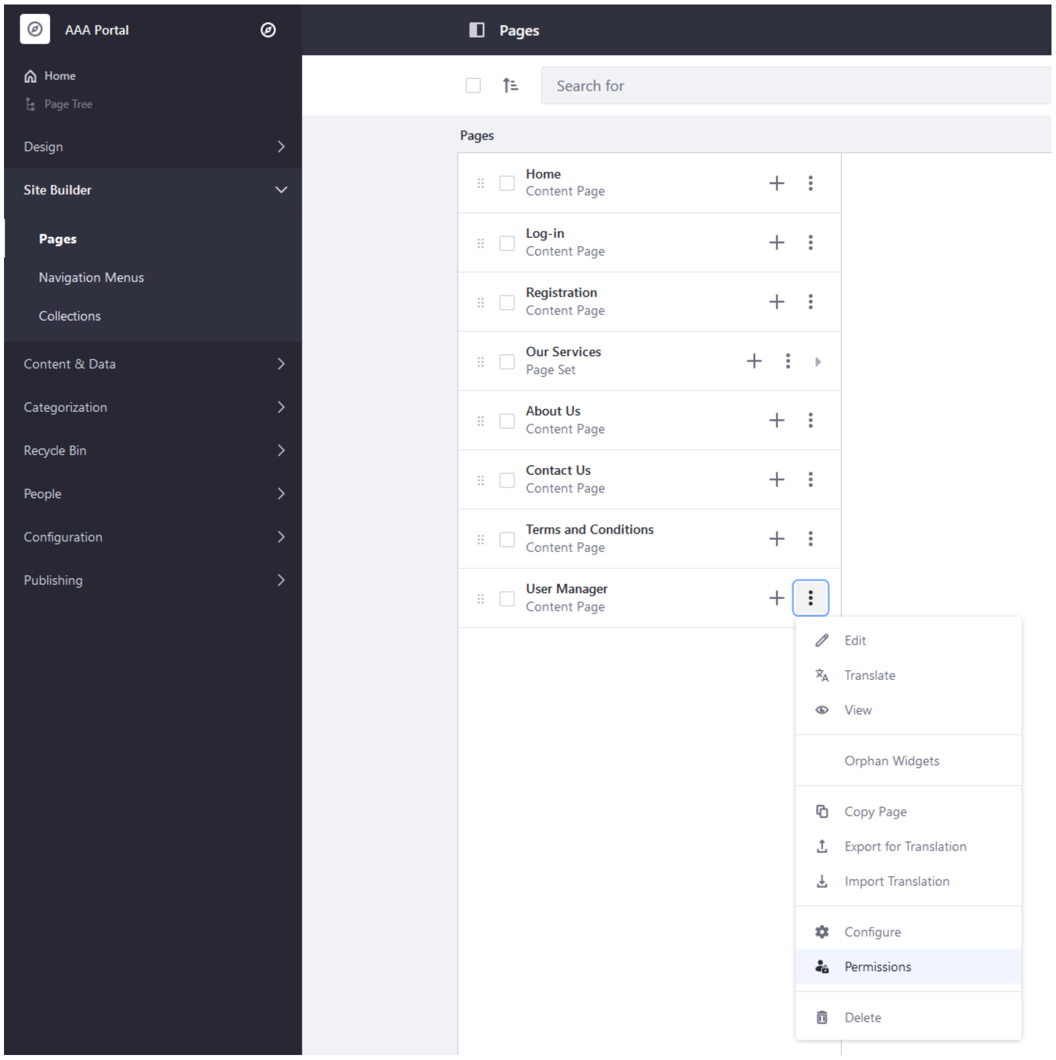
AAA Portal

Home | Log-in | Registration | Our Services | About Us | Contact Us | Terms and Conditions | User Manager

Sign-In

You are signed in as Francis Abarca.

Logging in with the Admin account

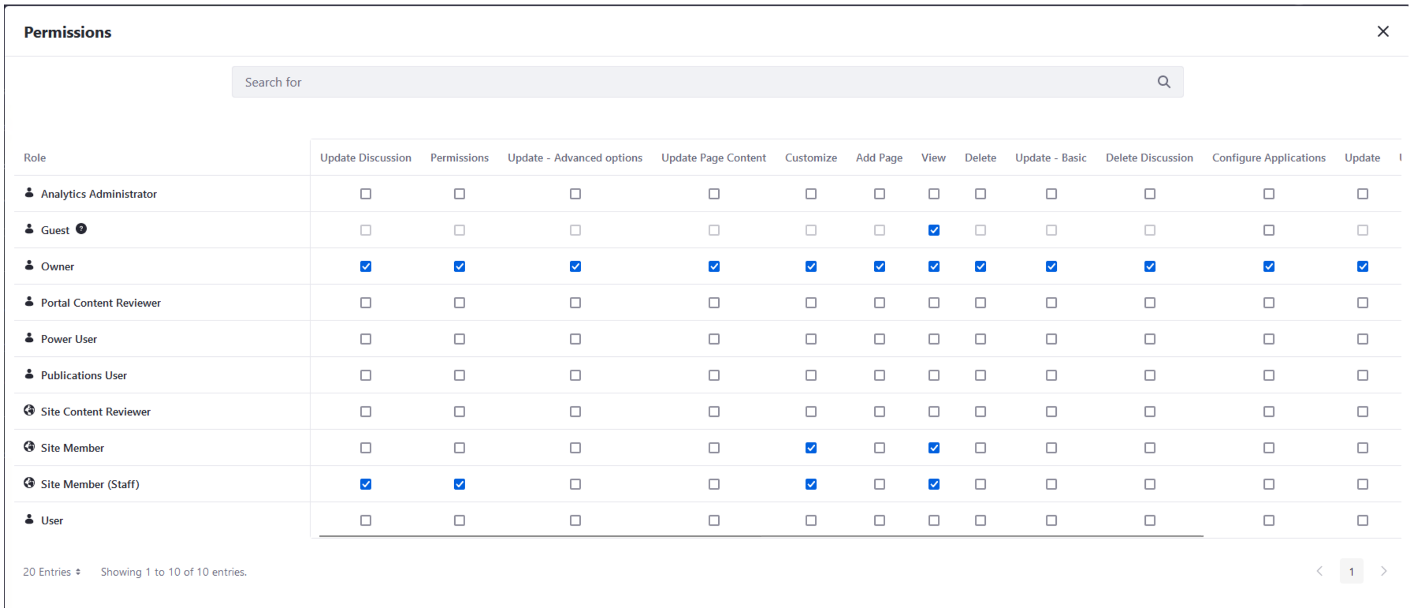


Pages

Page	Type
Home	Content Page
Log-in	Content Page
Registration	Content Page
Our Services	Page Set
About Us	Content Page
Contact Us	Content Page
Terms and Conditions	Content Page
User Manager	Content Page

BDSE – WFS – Web Development Using Platforms

Permission for Customer Management / Admin



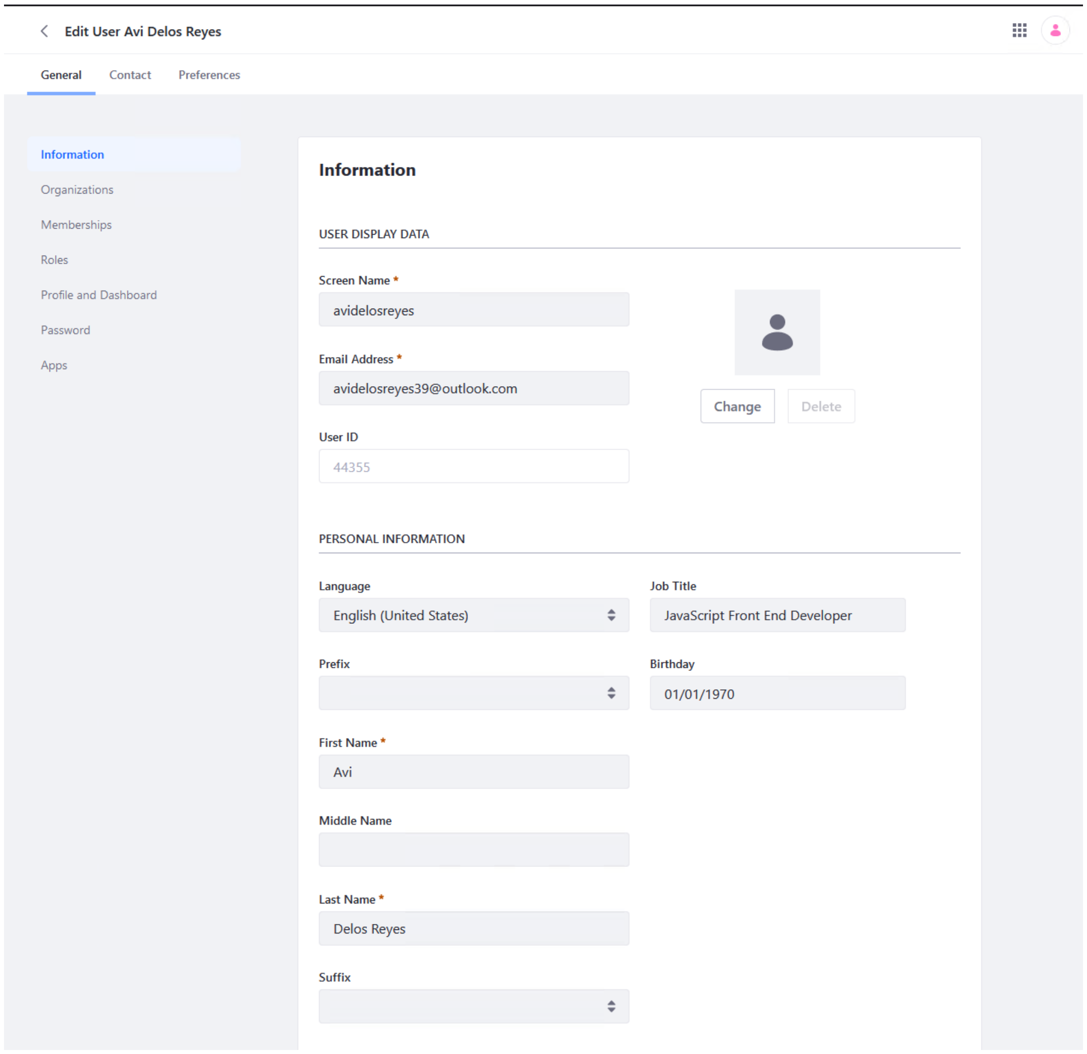
Permissions

Search for

Role	Update Discussion	Permissions	Update - Advanced options	Update Page Content	Customize	Add Page	View	Delete	Update - Basic	Delete Discussion	Configure Applications	Update
Analytics Administrator	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Guest	☐	☐	☐	☐	☐	☐	✓	☐	☐	☐	☐	☐
Owner	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Portal Content Reviewer	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Power User	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Publications User	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Site Content Reviewer	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Site Member	☐	☐	☐	☐	✓	☐	✓	☐	☐	☐	☐	☐
Site Member (Staff)	✓	✓	☐	☐	✓	☐	✓	☐	☐	☐	☐	☐
User	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

20 Entries Showing 1 to 10 of 10 entries.

Site Staff Account



Edit User Avi Delos Reyes

General | Contact | Preferences

Information
Organizations
Memberships
Roles
Profile and Dashboard
Password
Apps

Information

USER DISPLAY DATA

Screen Name *: avidelosreyes

Email Address *: avidelosreyes39@outlook.com

User ID: 44355

Change | Delete

PERSONAL INFORMATION

Language: English (United States)

Job Title: JavaScript Front End Developer

Prefix:

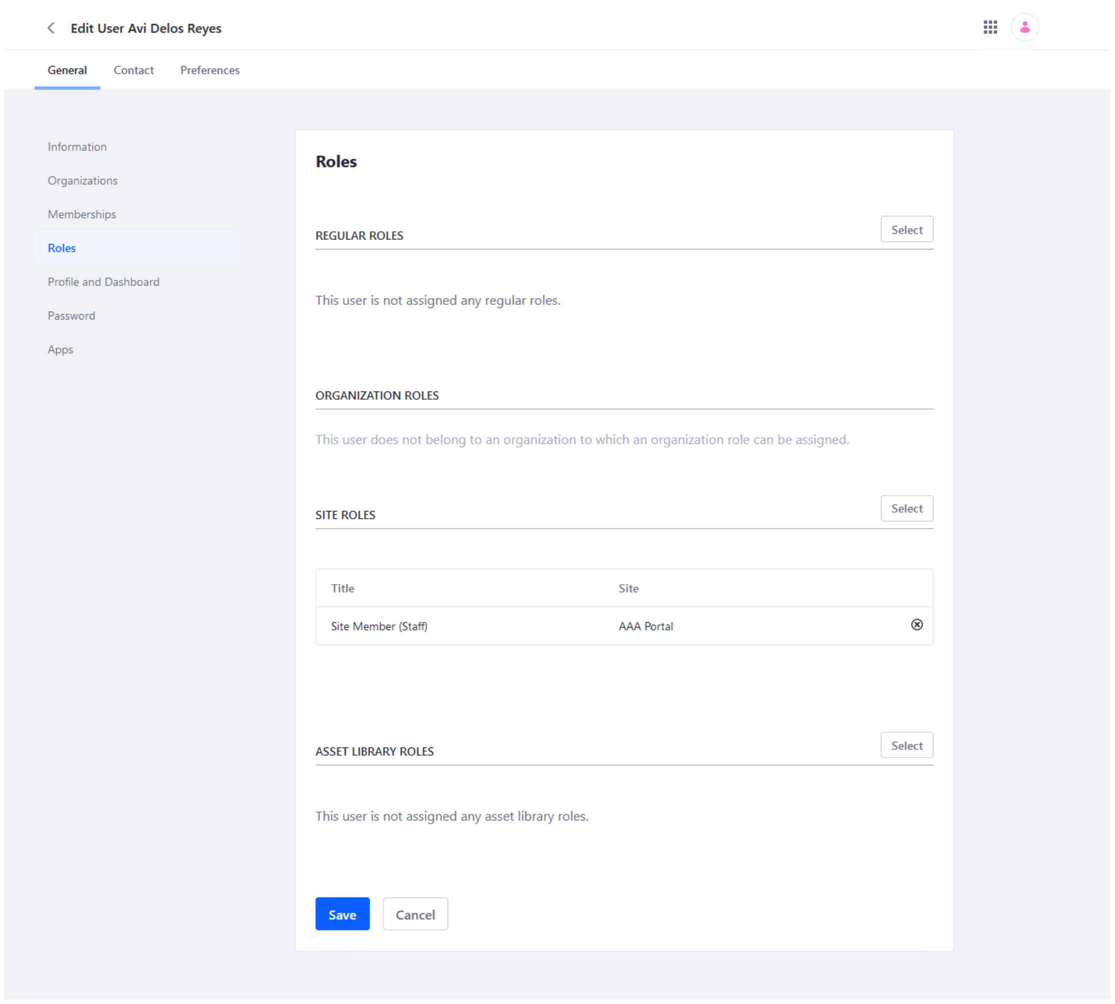
Birthday: 01/01/1970

First Name *: Avi

Middle Name:

Last Name *: Delos Reyes

Suffix:



Edit User Avi Delos Reyes

General | Contact | Preferences

- Information
- Organizations
- Memberships
- Roles
- Profile and Dashboard
- Password
- Apps

Roles

REGULAR ROLES — Select

This user is not assigned any regular roles.

ORGANIZATION ROLES

This user does not belong to an organization to which an organization role can be assigned.

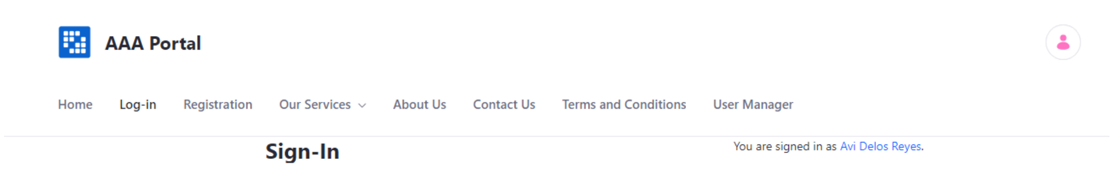
SITE ROLES — Select

Title	Site
Site Member (Staff)	AAA Portal

ASSET LIBRARY ROLES — Select

This user is not assigned any asset library roles.

Save | Cancel



AAA Portal

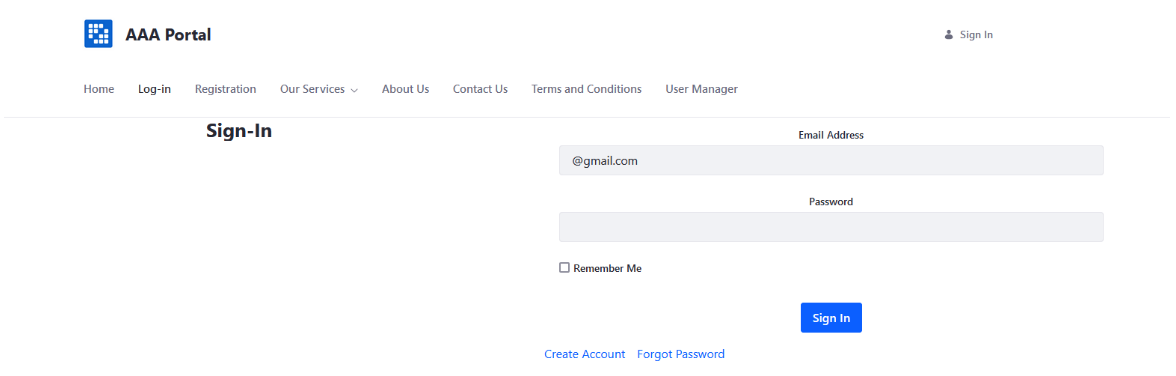
Home | Log-in | Registration | Our Services | About Us | Contact Us | Terms and Conditions | User Manager

Sign-In

You are signed in as Avi Delos Reyes.

Customer / Guest Website

Non-accessible admin.



AAA Portal — Sign In

Home | Log-in | Registration | Our Services | About Us | Contact Us | Terms and Conditions | User Manager

Sign-In

Email Address: @gmail.com

Password

Remember Me

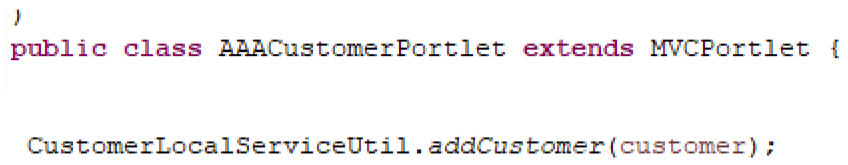
Sign In

Create Account | Forgot Password

2. Develop an application that implements design patterns and utilizes techniques to produce secure code. Provide the implemented code as evidence.

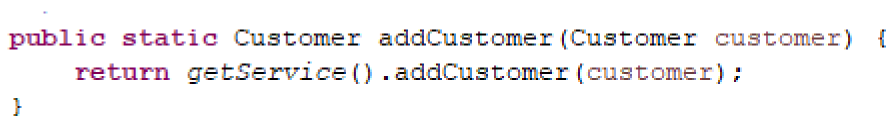
Creational Design Pattern (Factory Method)

AAACustomerPortlet



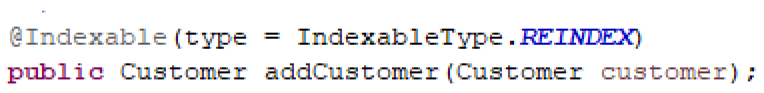
```
)
public class AAACustomerPortlet extends MVCPortlet {


  CustomerLocalServiceUtil.addCustomer(customer);
```

CustomerLocalServiceUtil



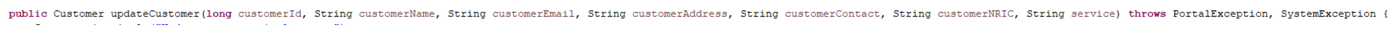
```
public static Customer addCustomer(Customer customer) {
     return getService().addCustomer(customer);
}
```

CustomerLocalService



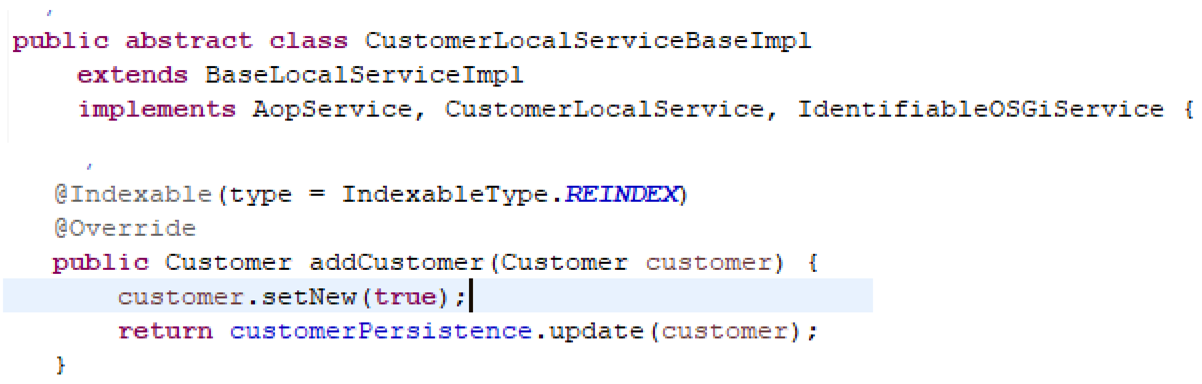
```
@Indexable(type = IndexableType.REINDEX)
public Customer addCustomer(Customer customer);
```

CustomerLocalServiceImpl



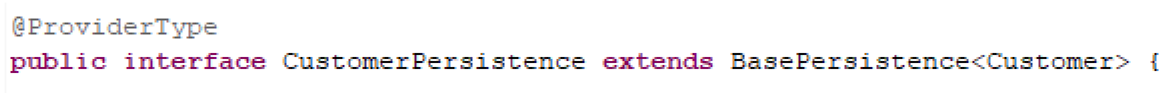
```
public Customer updateCustomer(long customerId, String customerName, String customerEmail, String customerAddress, String customerContact, String customerNRIC, String service) throws PortalException, SystemException {
```

CustomerLocalServiceBaseImpl



```
public abstract class CustomerLocalServiceBaseImpl
    extends BaseLocalServiceImpl
    implements AopService, CustomerLocalService, IdentifiableOSGiService {

   @Indexable(type = IndexableType.REINDEX)
   @Override
   public Customer addCustomer(Customer customer) {
       customer.setNew(true);
       return customerPersistence.update(customer);
   }
```

CustomerPersistence

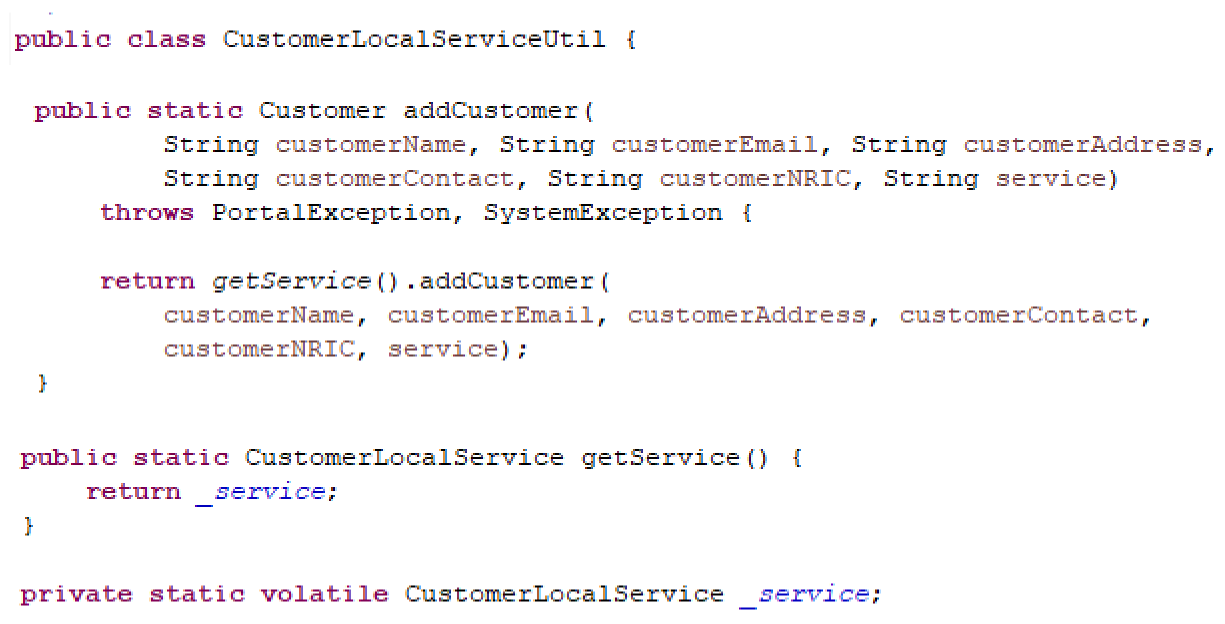


```
@ProviderType
public interface CustomerPersistence extends BasePersistence<Customer> {
```

The Factory Method pattern is employed to construct a new Customer object. The CustomerLocalServiceUtil class calls the createCustomer(parameter) function of the CustomerLocalService interface to build the customer object, which is implemented in the CustomerLocalServiceImpl class. After creating the Customer object, the AAACustomerPortlet can assign the necessary data. The Factory Method design guarantees accurate object creation while concealing the implementation specifics behind the interfaces needed to execute the appropriate functions.

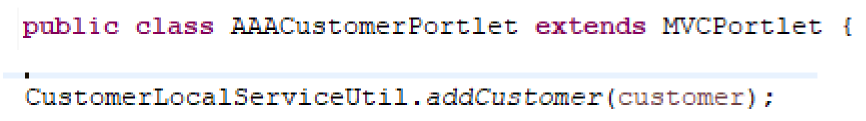
Structural Design Pattern (Façade Method)

CustomerLocalServiceUtil



```
public class CustomerLocalServiceUtil {

 public static Customer addCustomer(
         String customerName, String customerEmail, String customerAddress,
         String customerContact, String customerNRIC, String service)
     throws PortalException, SystemException {

     return getService().addCustomer(
         customerName, customerEmail, customerAddress, customerContact,
         customerNRIC, service);
 }

 public static CustomerLocalService getService() {
     return _service;
 }

 private static volatile CustomerLocalService _service;
```

AAACustomerPortlet

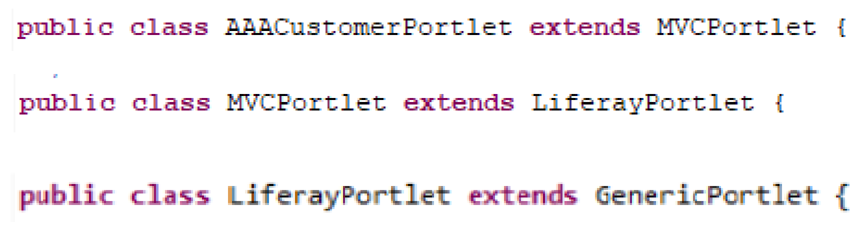


```
public class AAACustomerPortlet extends MVCPortlet {

CustomerLocalServiceUtil.addCustomer(customer);
```

To access its methods, the CustomerLocalService has to navigate through multiple classes; yet, the CustomerLocalServiceUtil offers a means to access these intricate methods without understanding the underlying implementations. This demonstrates encapsulation, where internal specifics are concealed behind a more approachable interface.

Behavioral Design Pattern (Template Method)

The templates of the MVCPortlet class are employed to generate portlets. Consequently, all newly created portlets will extend MVCPortlet. The MVCPortlet class extends LiferayPortlet, which further extends GenericPortlet. This exemplifies multi-level inheritance. Due to the abstract nature of the GenericPortlet class, abstraction plays a crucial role in this scenario.



```
public class AAACustomerPortlet extends MVCPortlet {

public class MVCPortlet extends LiferayPortlet {

public class LiferayPortlet extends GenericPortlet {
```

Task 4

Evaluation of the Design Pattern

Solution:

1. Discuss the use of design patterns for the given purpose and consequences by applying design patterns.

a. Discuss usage of design patterns in “AAA Customer Management Portlet.”

b. What are the consequences of design patterns which you have used in your project.

The **Factory method**, a creational design pattern, is employed to initialize an object by generating a Customer instance using the createCustomer() method from the CustomerLocalService. This approach allows the creation of multiple object instances without exposing the underlying creation logic. However, it can raise system complexity and demand significant time and resources.

The **Facade pattern**, a Structural design pattern, assists in providing database modification access to the system. Utilizing a facade class conceals the code complexity and simplifies system access by invoking suitable methods on the client's behalf, negating the need for multiple method calls. However, the facade class's implementation might be extensive since it requires integration with existing code, and relying exclusively on the facade class for business logic access may pose risks.

Behavioral design patterns facilitate communication between classes, while the template pattern enables the development of various portlets from a single template, enhancing code reusability. Nonetheless, using the same template for all portlets could restrict some functionalities or specific designs tailored to certain portlets.

2. Investigate how different design patterns can work within a range of different scenarios.

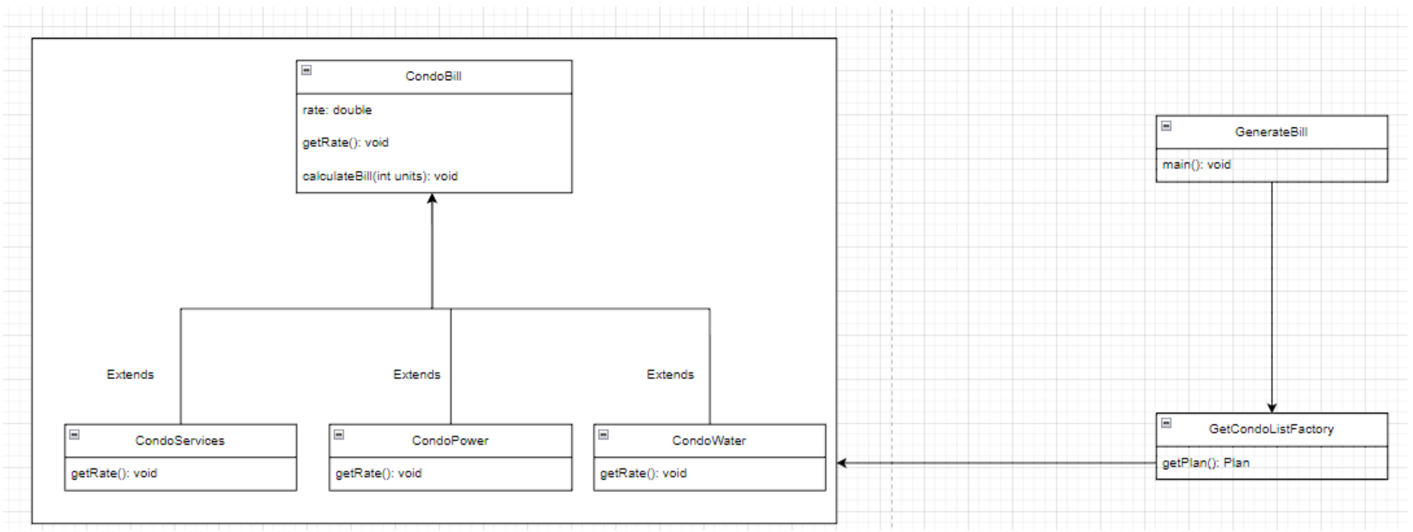
Design Patterns

1. Creational Design Pattern

Investigate at least 2 different design patterns in creational with scenarios and relevant diagrams.

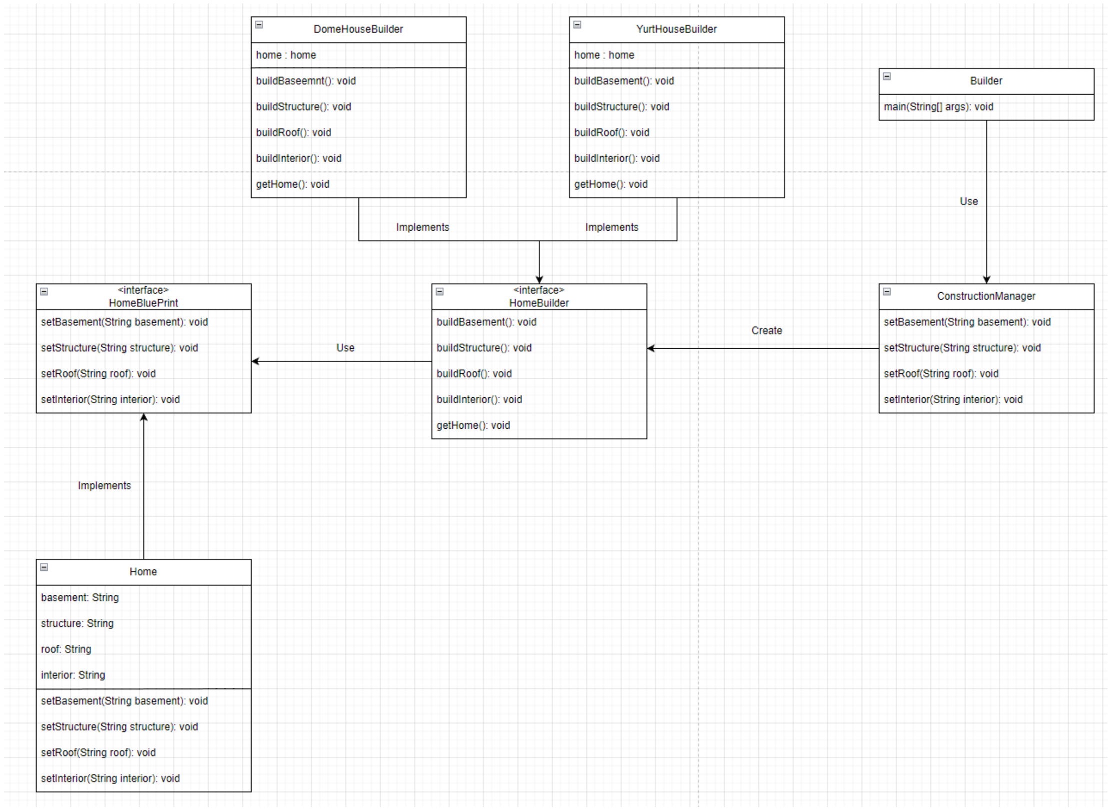
Factory Method

The GetCondoListFactory is utilized by GenerateBill to produce a wide range of condolists by passing information to the GetCondoListFActory method to obtain the requested type of condo list bill.



Builder Pattern

The ConstructionManager utilizes the Builder to assemble a HomeBuilder. DomeHouseBuilder and YurtHouseBuilder implement the interface, and these subclasses decide the appropriate HomeBuilder implementation to use. Home implements the HomeBlueprint interface, which the HomeBuilder interface employs. The ConstructionManager will build the requested HomeBuilder.

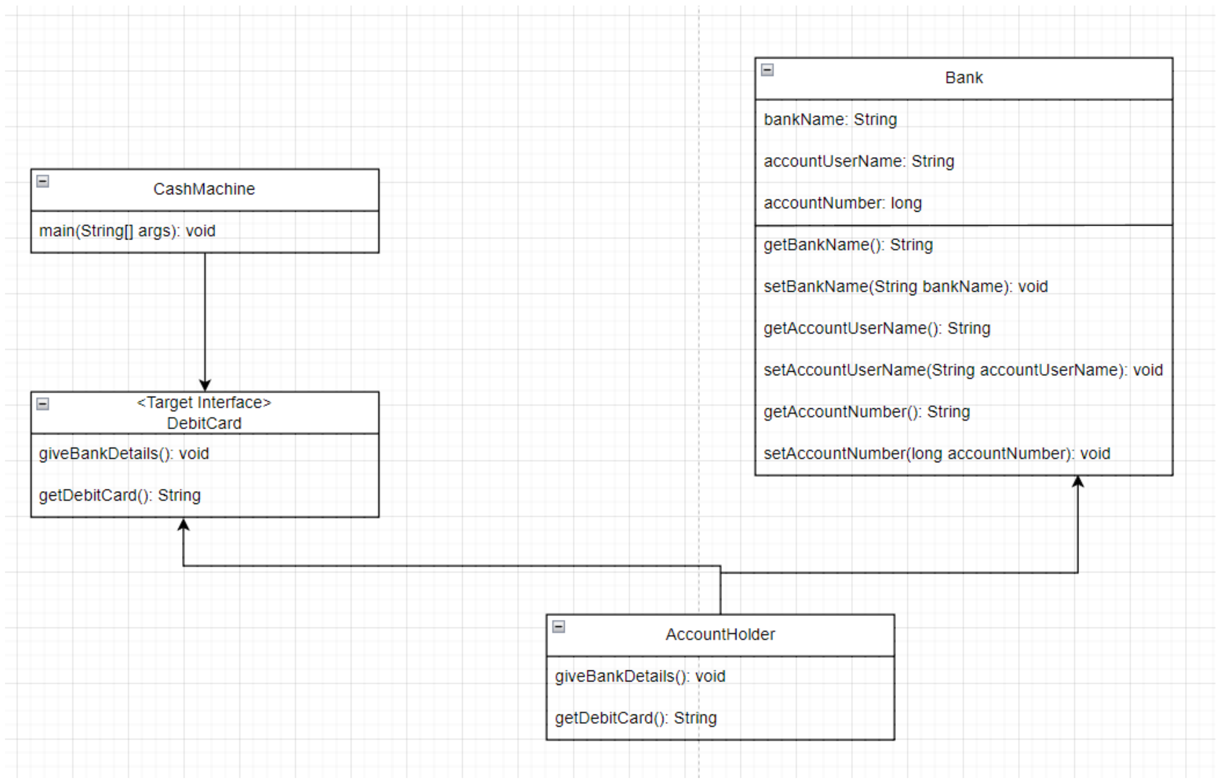


2. Structural Design Pattern

Investigate at least 2 different design patterns in structural with scenarios and relevant diagrams.

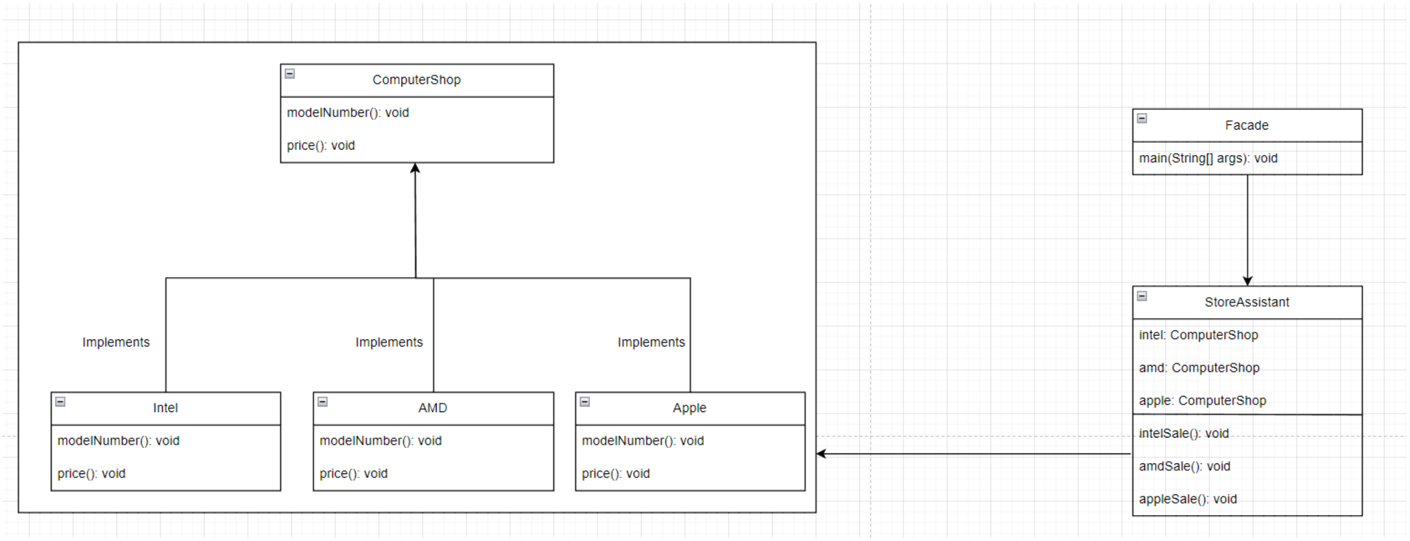
Adapter Pattern

The CashMachine class employs the DebitCard interface, which is implemented by the AccountHolder class. The AccountHolder class serves as a wrapper, adapting specific methods available from the Bank class, as it extends that class.



Façade Pattern

In order to discover the available PC options in the ComputerShop, the facade class, known as the StoreAssistant, compiles and stores all accessible computers whenever a request is obtained from the client class, named Facade. The StoreAssistant class manages the intricate implementation and delivers the requested information to the client.

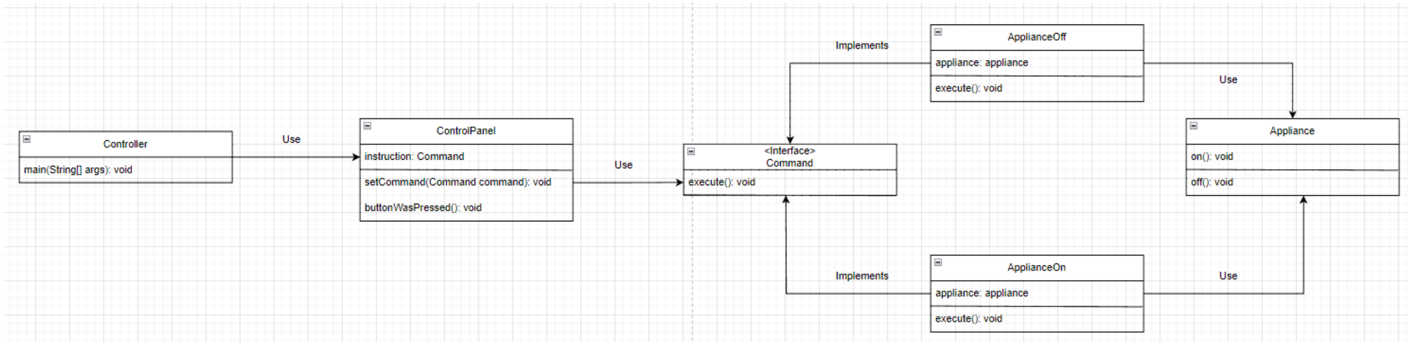


3. Behavioral Design Pattern

Investigate at least 2 different design patterns in behavioral with scenarios and relevant diagrams.

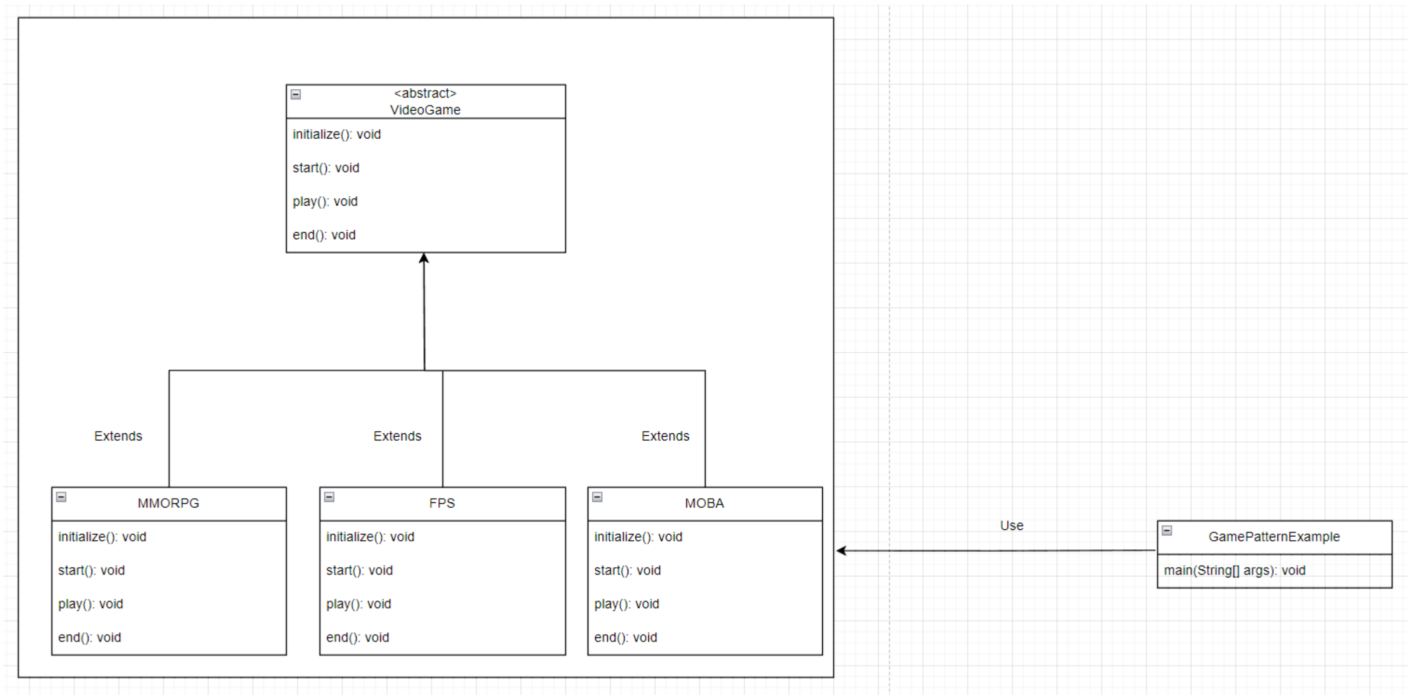
Command Pattern

The Controller class utilizes the ControlPanel class to transmit a directive or command, while the class implementing the Command interface carries out the command, manipulating or managing the Appliance class in the process.



Template Method Pattern

The GamePatternExample class employs the abstract class, Game, with the specific game depending on the requested subclass.



3. Identify the appropriate design pattern from the investigation.

a. Among 2 different scenarios in creational design which you have analyzed in P4, reconcile the most appropriate design pattern.

- Out of the 2 creational design patterns, the **Factory Method** design fits the best.

b. Among 2 different scenarios in structural design which you have analyzed in P4, reconcile the most appropriate design pattern.

- Out of the 2 structural design patterns, the **Façade Method** makes the most sense.

c. Among the 2 different scenarios in behavioral design which you have analyzed in P4, reconcile the most appropriate design pattern.

- Between the Command Pattern and the Template Method, the **Template Method** makes the more sense as the most appropriate design pattern.

4. Evaluate and justify the design patterns that you had to identify in each of the scenarios.

Creational Design Patterns aim to offer flexible, efficient, and reusable approaches for object creation while promoting loose coupling between classes. The Factory Method enables a class to delegate object creation to its subclasses, allowing for implementation changes regardless of the class. The capacity to extend or implement an interface or abstract class facilitates the development of additional classes with the same structure, which is beneficial when the objects to be created are unknown or not predetermined.

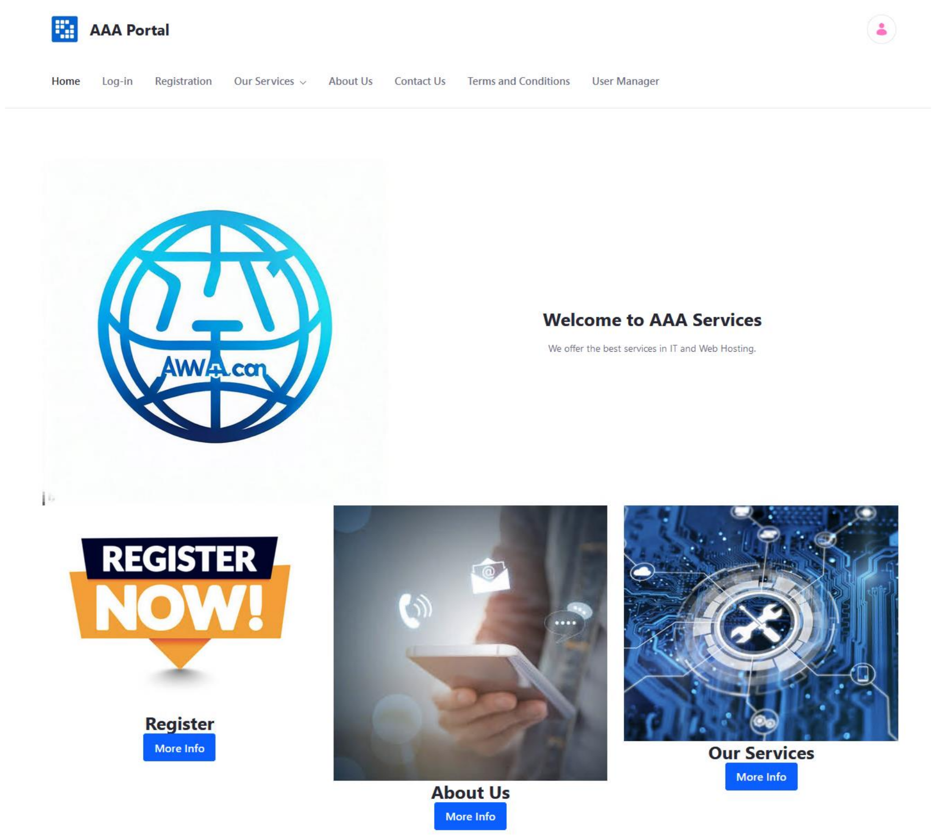
Structural Design Patterns focus on the organization and amalgamation of objects and classes to construct larger structures with enhanced functionality and features. The Facade Pattern, specifically, assists in streamlining a complex structure. Accessing a system's complexity through a single facade class simplifies the client's use of all the code's methods without exposing its complexity. Since code structure can occasionally be intricate and daunting, concealing it behind a facade class eases management without compromising the code's structure.

Behavioral Design Patterns concentrate on the communication and interaction between objects and classes, particularly regarding the allocation of responsibilities and algorithms. The Template Method permits variations in some methods while maintaining a consistent overall process and structure. Consequently, it can be employed to establish a uniform step for executing code with varying request parameters, simplifying the procedure and enhancing system efficiency.

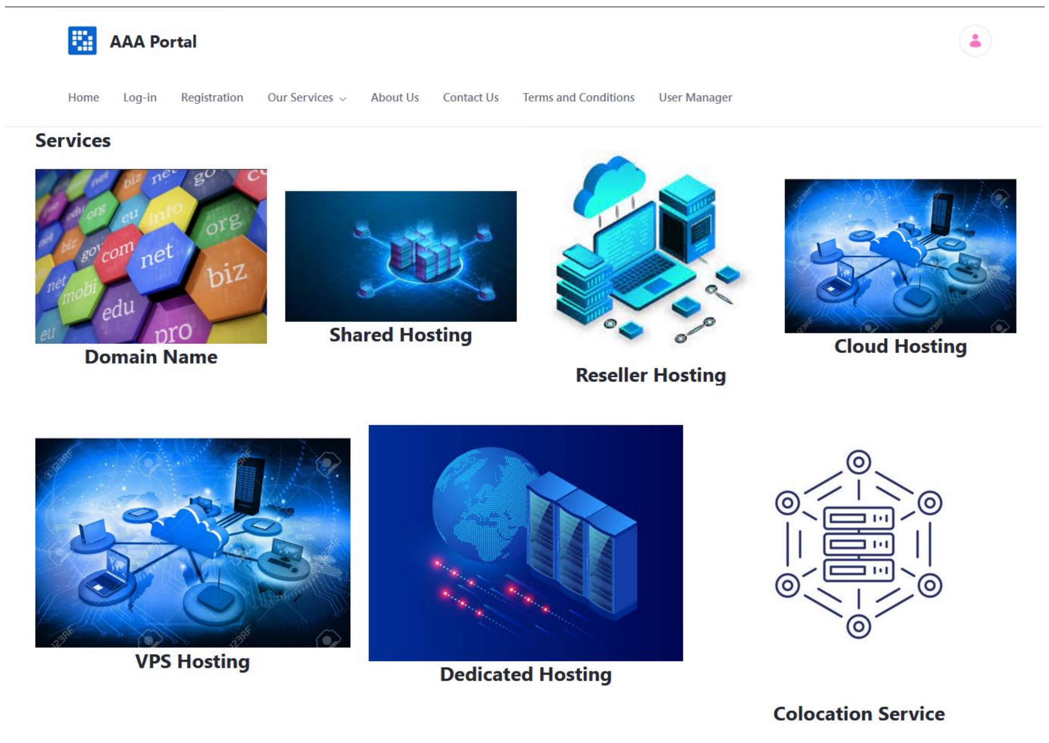
Task 5

Final Project

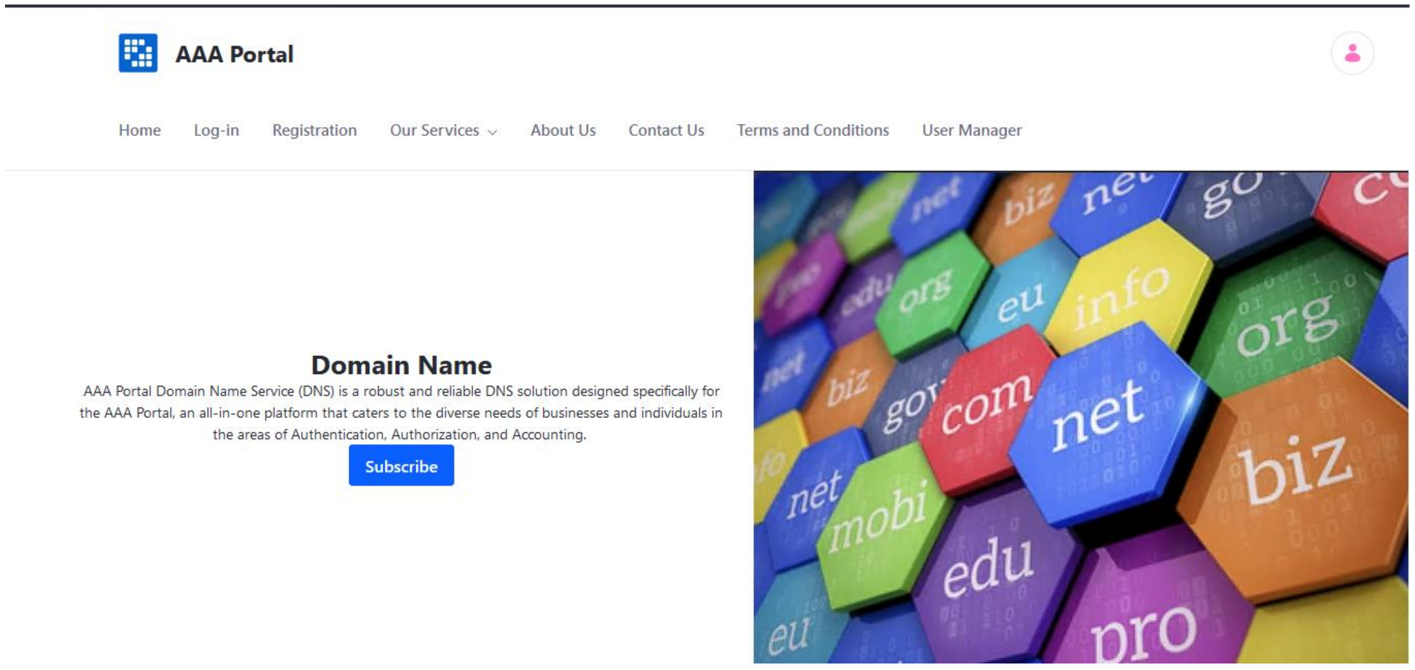
i. Home Page



ii. Services Page



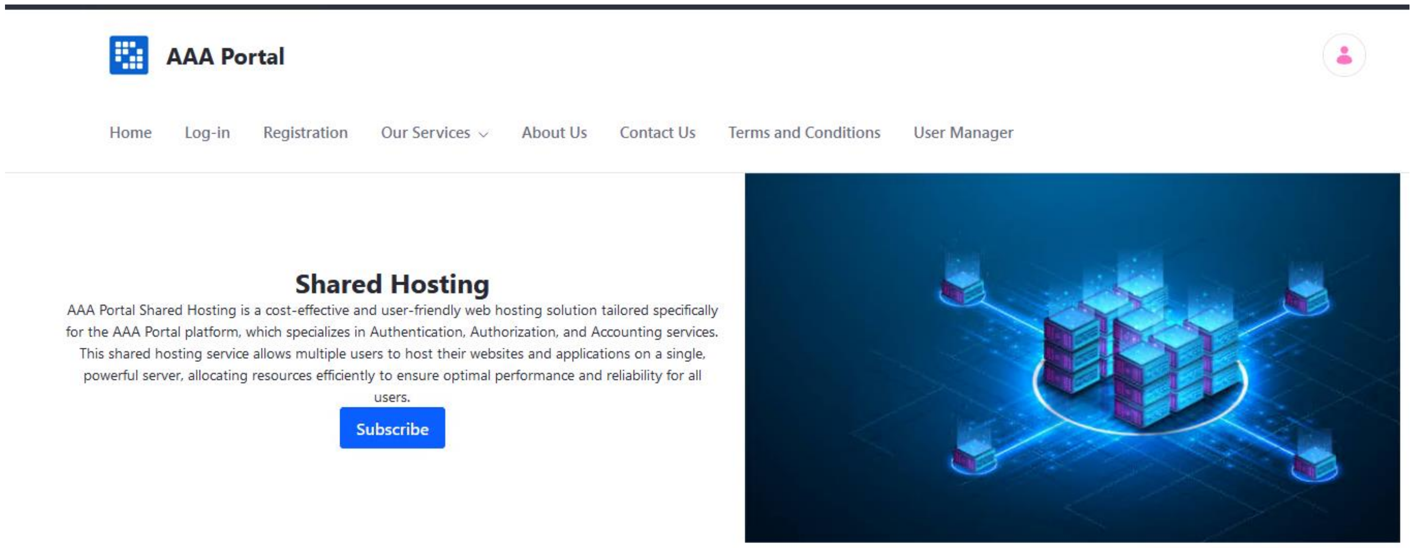
iii. Domain Name Service Page



Domain Name

AAA Portal Domain Name Service (DNS) is a robust and reliable DNS solution designed specifically for the AAA Portal, an all-in-one platform that caters to the diverse needs of businesses and individuals in the areas of Authentication, Authorization, and Accounting.

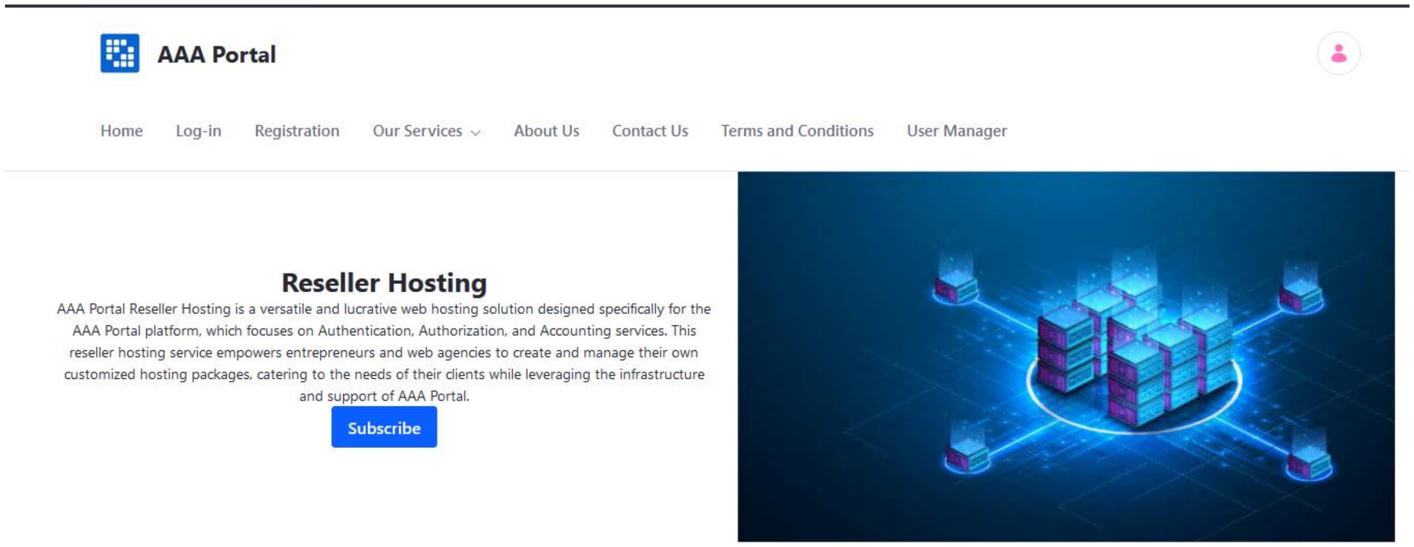
iv. Shared Hosting Service Page



Shared Hosting

AAA Portal Shared Hosting is a cost-effective and user-friendly web hosting solution tailored specifically for the AAA Portal platform, which specializes in Authentication, Authorization, and Accounting services. This shared hosting service allows multiple users to host their websites and applications on a single, powerful server, allocating resources efficiently to ensure optimal performance and reliability for all users.

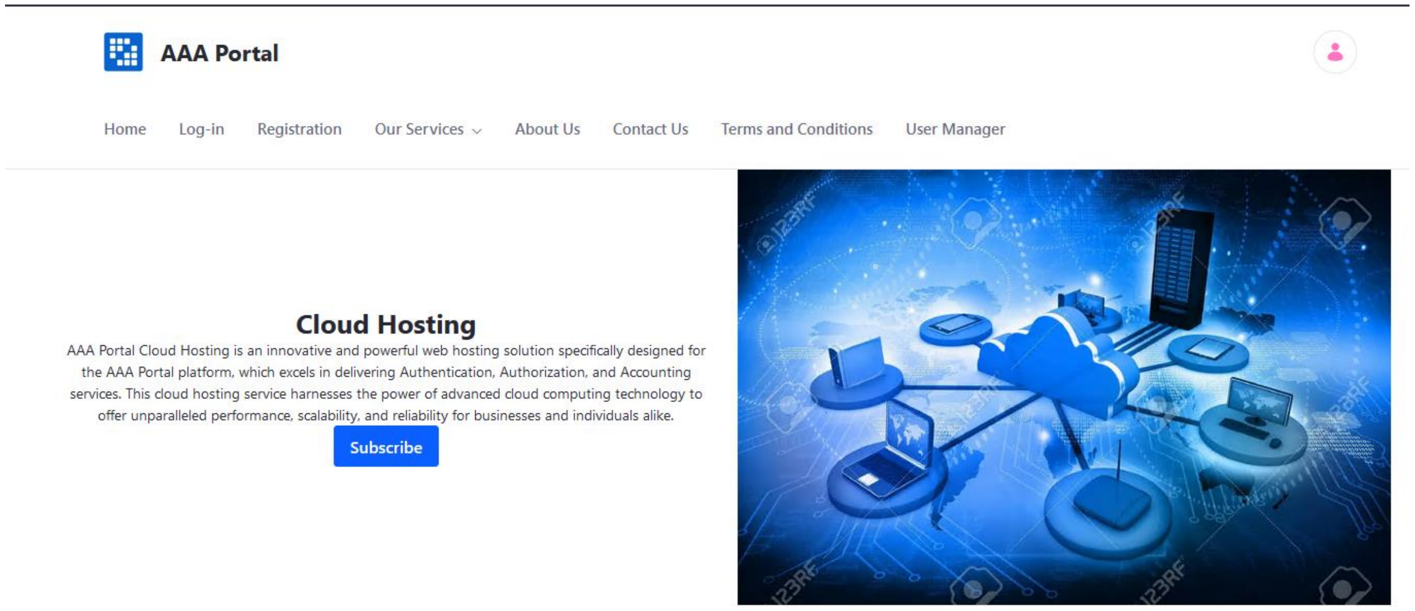
v. Reseller Hosting Service Page



Reseller Hosting

AAA Portal Reseller Hosting is a versatile and lucrative web hosting solution designed specifically for the AAA Portal platform, which focuses on Authentication, Authorization, and Accounting services. This reseller hosting service empowers entrepreneurs and web agencies to create and manage their own customized hosting packages, catering to the needs of their clients while leveraging the infrastructure and support of AAA Portal.

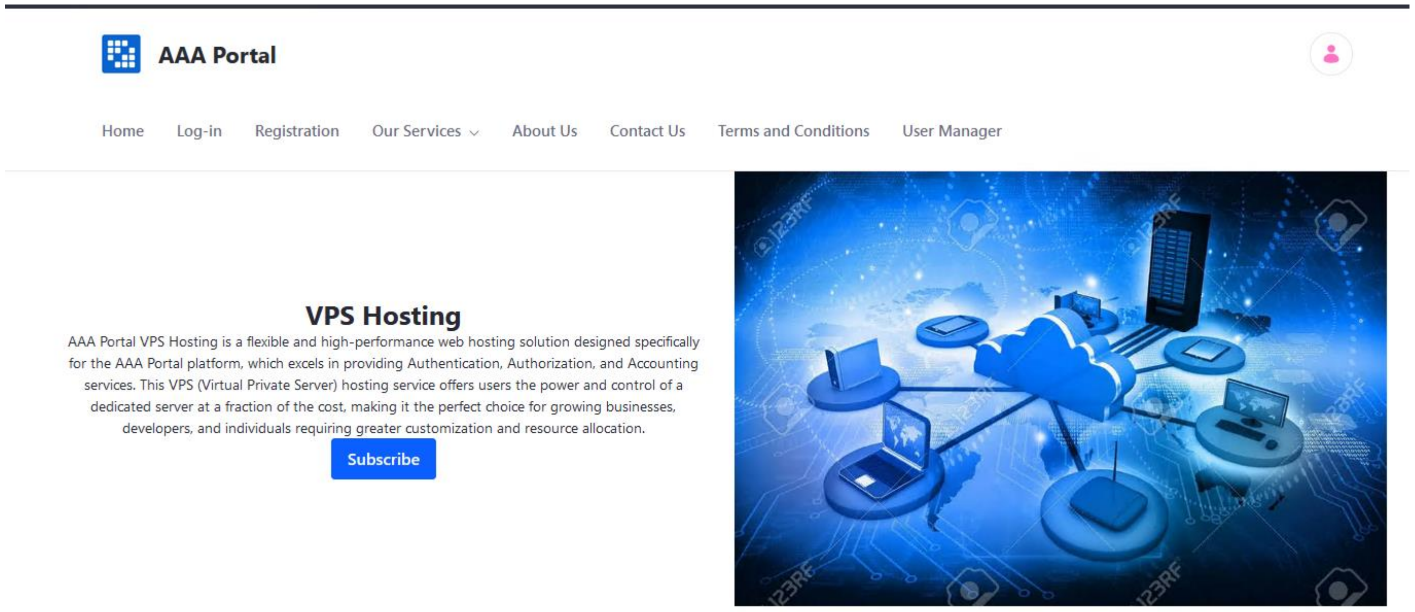
vi. Cloud Hosting Service Page



Cloud Hosting

AAA Portal Cloud Hosting is an innovative and powerful web hosting solution specifically designed for the AAA Portal platform, which excels in delivering Authentication, Authorization, and Accounting services. This cloud hosting service harnesses the power of advanced cloud computing technology to offer unparalleled performance, scalability, and reliability for businesses and individuals alike.

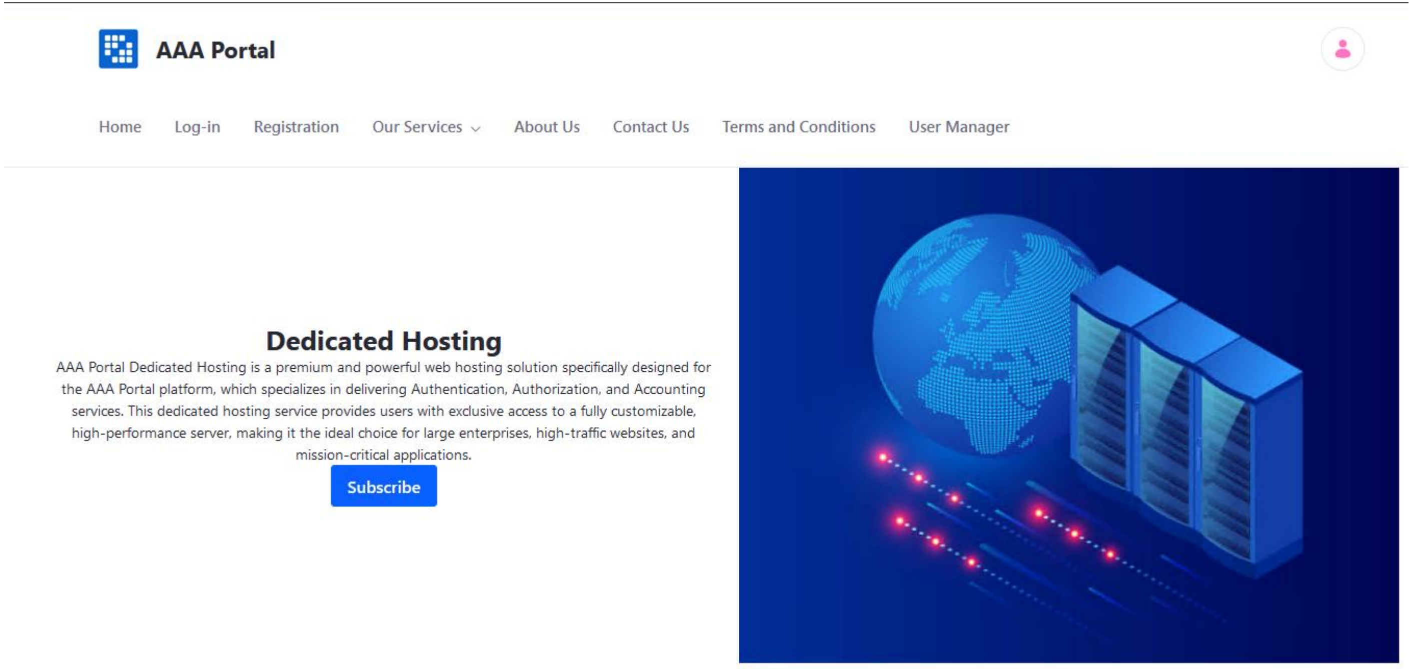
vii. VPS Hosting Service Page



VPS Hosting

AAA Portal VPS Hosting is a flexible and high-performance web hosting solution designed specifically for the AAA Portal platform, which excels in providing Authentication, Authorization, and Accounting services. This VPS (Virtual Private Server) hosting service offers users the power and control of a dedicated server at a fraction of the cost, making it the perfect choice for growing businesses, developers, and individuals requiring greater customization and resource allocation.

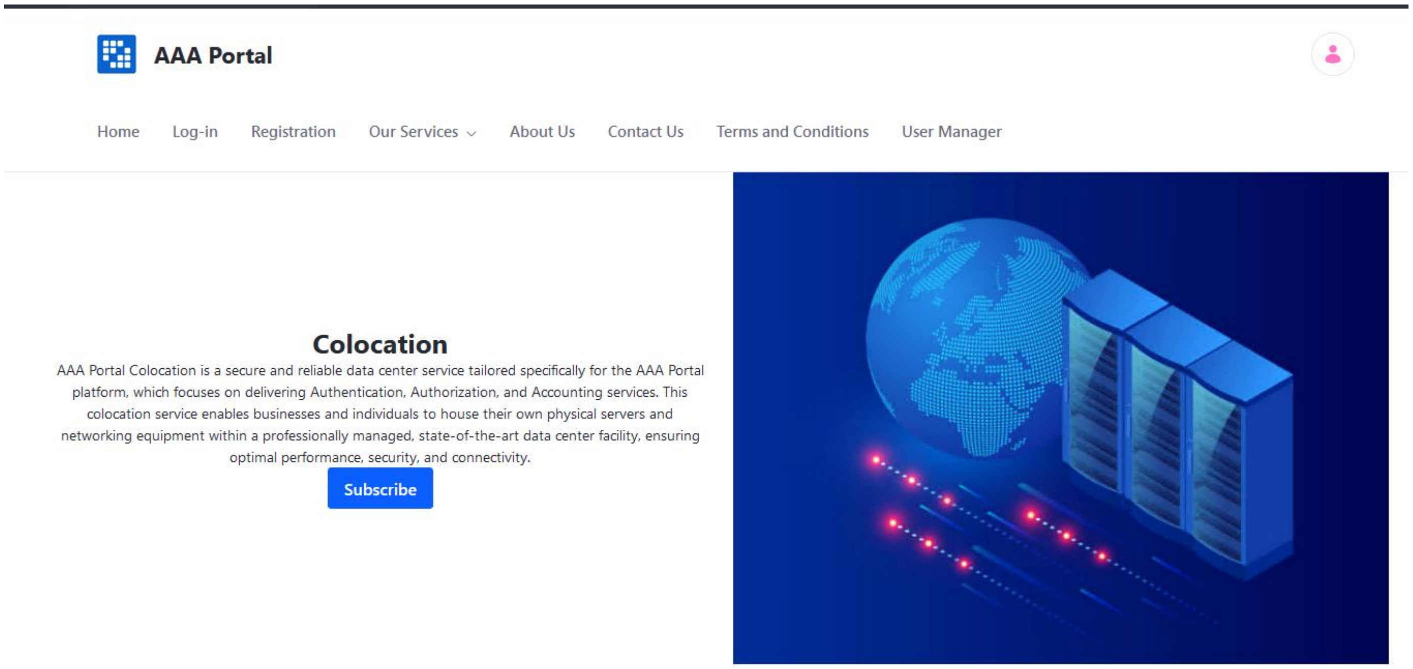
viii. Dedicated Hosting Service Page



Dedicated Hosting

AAA Portal Dedicated Hosting is a premium and powerful web hosting solution specifically designed for the AAA Portal platform, which specializes in delivering Authentication, Authorization, and Accounting services. This dedicated hosting service provides users with exclusive access to a fully customizable, high-performance server, making it the ideal choice for large enterprises, high-traffic websites, and mission-critical applications.

ix. Colocation Service Page



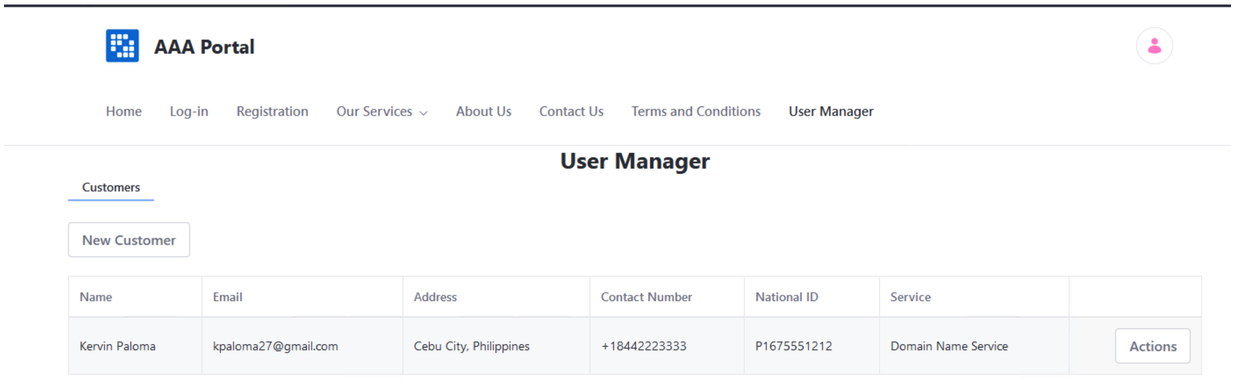
AAA Portal

Home Log-in Registration Our Services About Us Contact Us Terms and Conditions User Manager

Colocation

AAA Portal Colocation is a secure and reliable data center service tailored specifically for the AAA Portal platform, which focuses on delivering Authentication, Authorization, and Accounting services. This colocation service enables businesses and individuals to house their own physical servers and networking equipment within a professionally managed, state-of-the-art data center facility, ensuring optimal performance, security, and connectivity.

x. User Manager



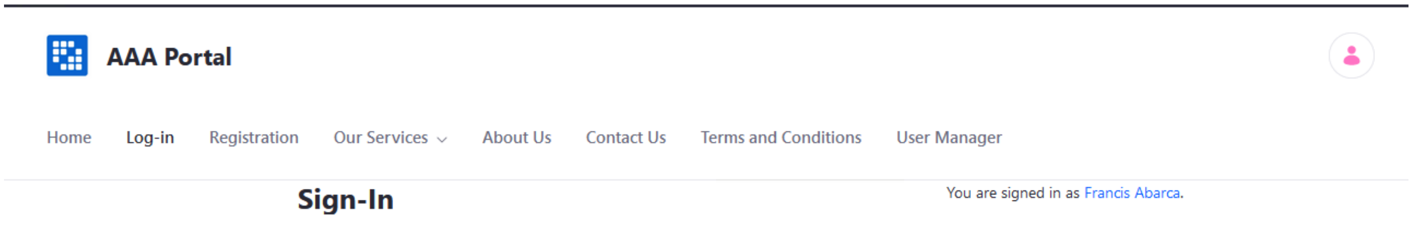
AAA Portal

Home Log-in Registration Our Services About Us Contact Us Terms and Conditions User Manager

User Manager

Name	Email	Address	Contact Number	National ID	Service	
Kervin Paloma	kpaloma27@gmail.com	Cebu City, Philippines	+18442223333	P1675551212	Domain Name Service	Actions

xi. Login Page



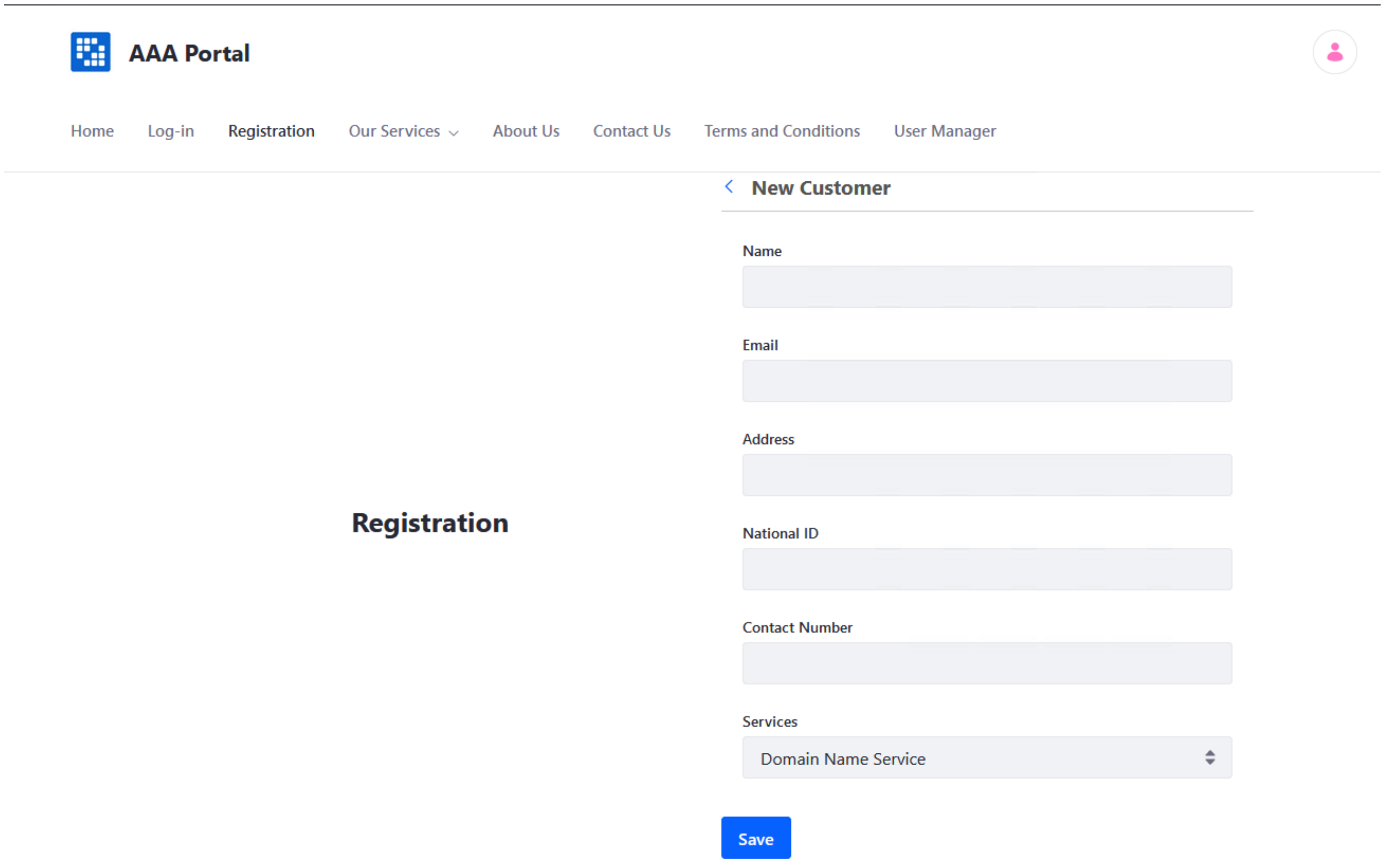
AAA Portal

Home Log-in Registration Our Services About Us Contact Us Terms and Conditions User Manager

Sign-In

You are signed in as Francis Abarca.

xii. Registration Page

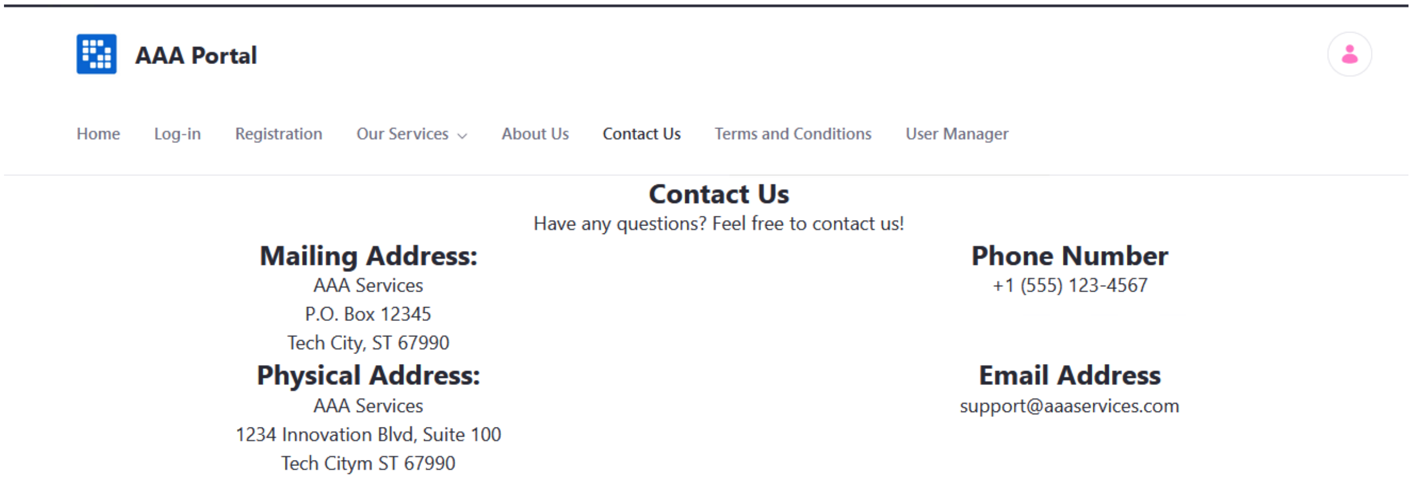


Registration

New Customer

Name	
Email	
Address	
National ID	
Contact Number	
Services	Domain Name Service

xiii. Contact Us Page



Contact Us

Have any questions? Feel free to contact us!

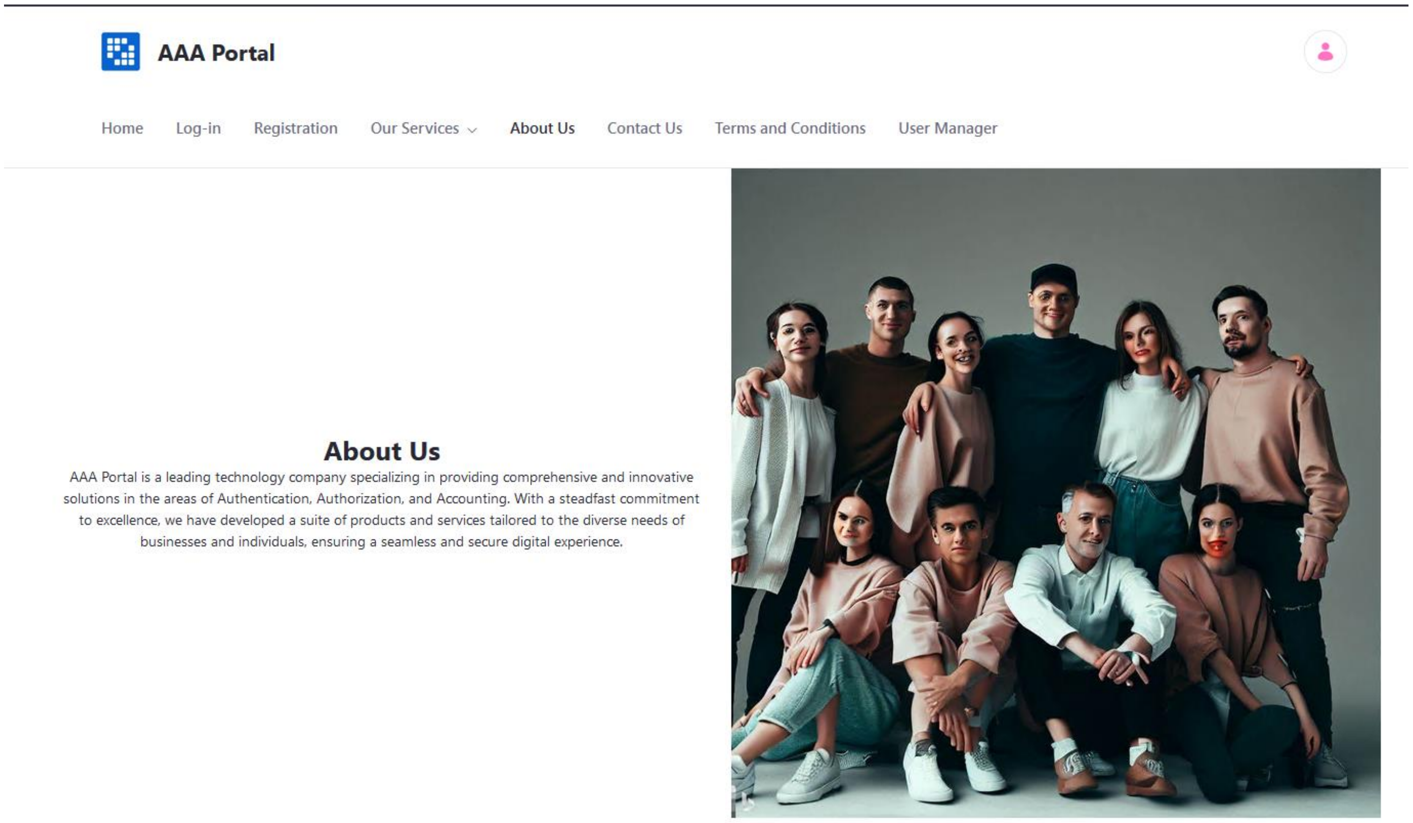
Mailing Address:
AAA Services
P.O. Box 12345
Tech City, ST 67990

Physical Address:
AAA Services
1234 Innovation Blvd, Suite 100
Tech Citym ST 67990

Phone Number
+1 (555) 123-4567

Email Address
support@aaaservices.com

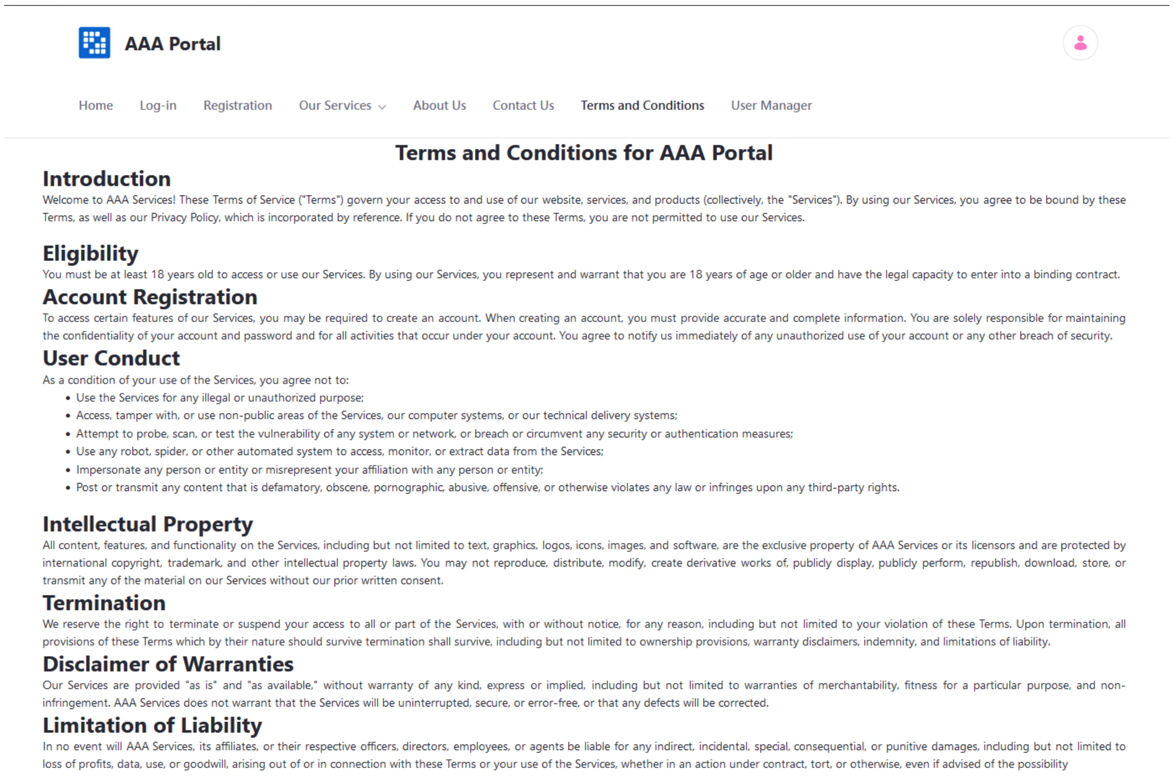
xiv. About Us Page



About Us

AAA Portal is a leading technology company specializing in providing comprehensive and innovative solutions in the areas of Authentication, Authorization, and Accounting. With a steadfast commitment to excellence, we have developed a suite of products and services tailored to the diverse needs of businesses and individuals, ensuring a seamless and secure digital experience.

xv. Terms and Conditions Page



Terms and Conditions for AAA Portal

Introduction

Welcome to AAA Services! These Terms of Service ("Terms") govern your access to and use of our website, services, and products (collectively, the "Services"). By using our Services, you agree to be bound by these Terms, as well as our Privacy Policy, which is incorporated by reference. If you do not agree to these Terms, you are not permitted to use our Services.

Eligibility

You must be at least 18 years old to access or use our Services. By using our Services, you represent and warrant that you are 18 years of age or older and have the legal capacity to enter into a binding contract.

Account Registration

To access certain features of our Services, you may be required to create an account. When creating an account, you must provide accurate and complete information. You are solely responsible for maintaining the confidentiality of your account and password and for all activities that occur under your account. You agree to notify us immediately of any unauthorized use of your account or any other breach of security.

User Conduct

As a condition of your use of the Services, you agree not to:

- Use the Services for any illegal or unauthorized purpose;
- Access, tamper with, or use non-public areas of the Services, our computer systems, or our technical delivery systems;
- Attempt to probe, scan, or test the vulnerability of any system or network, or breach or circumvent any security or authentication measures;
- Use any robot, spider, or other automated system to access, monitor, or extract data from the Services;
- Impersonate any person or entity or misrepresent your affiliation with any person or entity;
- Post or transmit any content that is defamatory, obscene, pornographic, abusive, offensive, or otherwise violates any law or infringes upon any third-party rights.

Intellectual Property

All content, features, and functionality on the Services, including but not limited to text, graphics, logos, icons, images, and software, are the exclusive property of AAA Services or its licensors and are protected by international copyright, trademark, and other intellectual property laws. You may not reproduce, distribute, modify, create derivative works of, publicly display, publicly perform, republish, download, store, or transmit any of the material on our Services without our prior written consent.

Termination

We reserve the right to terminate or suspend your access to all or part of the Services, with or without notice, for any reason, including but not limited to your violation of these Terms. Upon termination, all provisions of these Terms which by their nature should survive termination shall survive, including but not limited to ownership provisions, warranty disclaimers, indemnity, and limitations of liability.

Disclaimer of Warranties

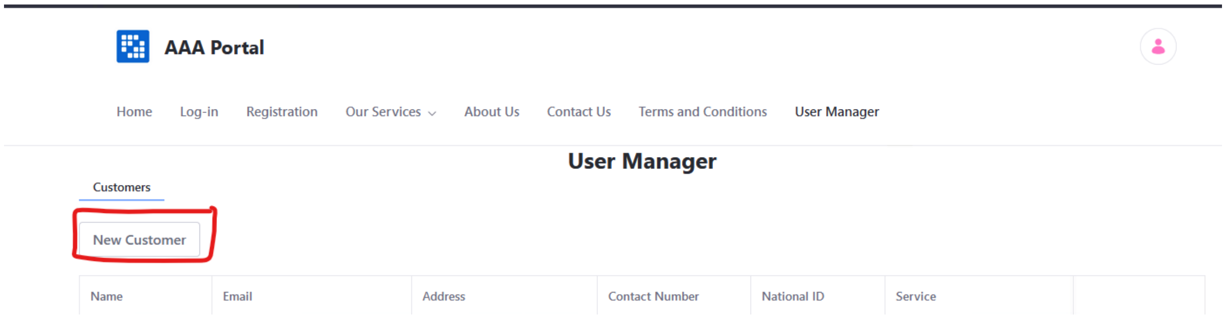
Our Services are provided "as is" and "as available," without warranty of any kind, express or implied, including but not limited to warranties of merchantability, fitness for a particular purpose, and non-infringement. AAA Services does not warrant that the Services will be uninterrupted, secure, or error-free, or that any defects will be corrected.

Limitation of Liability

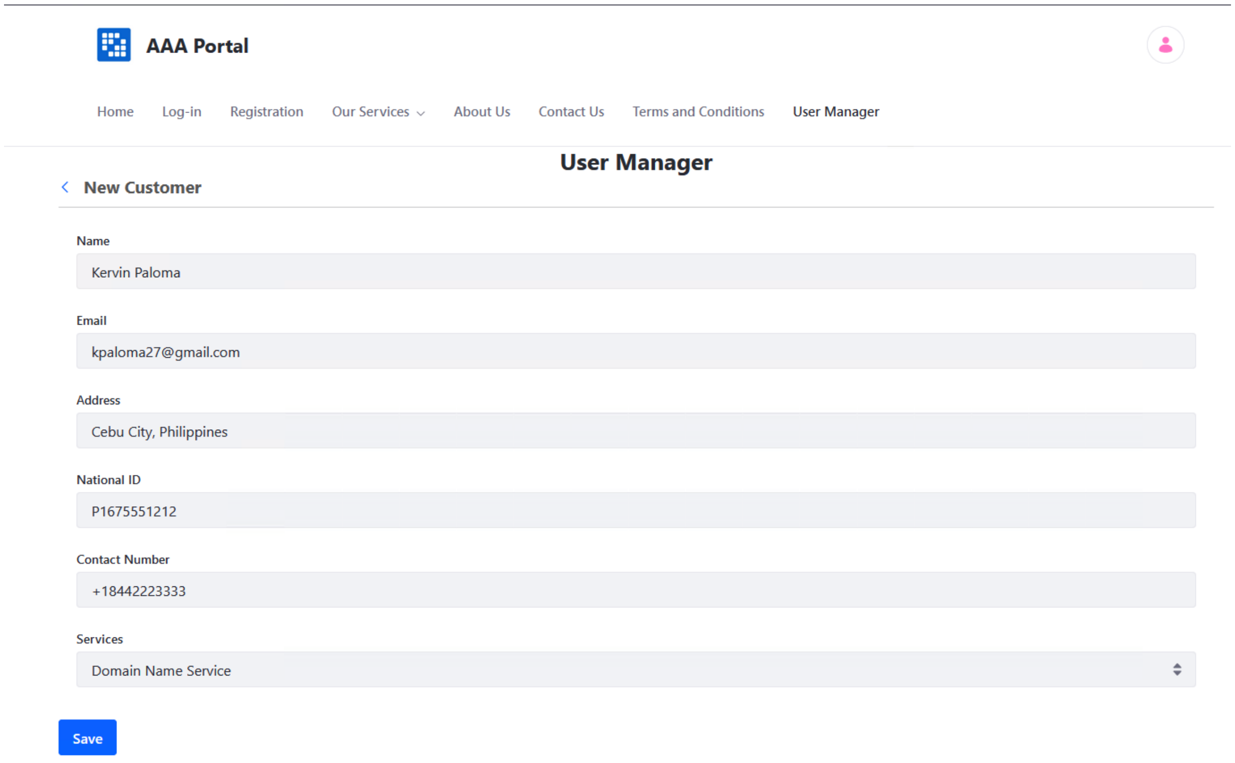
In no event will AAA Services, its affiliates, or their respective officers, directors, employees, or agents be liable for any indirect, incidental, special, consequential, or punitive damages, including but not limited to loss of profits, data, use, or goodwill, arising out of or in connection with these Terms or your use of the Services, whether in an action under contract, tort, or otherwise, even if advised of the possibility

xvi. Scenarios:

i. Add Customer



Name	Email	Address	Contact Number	National ID	Service	



Name
Kervin Paloma

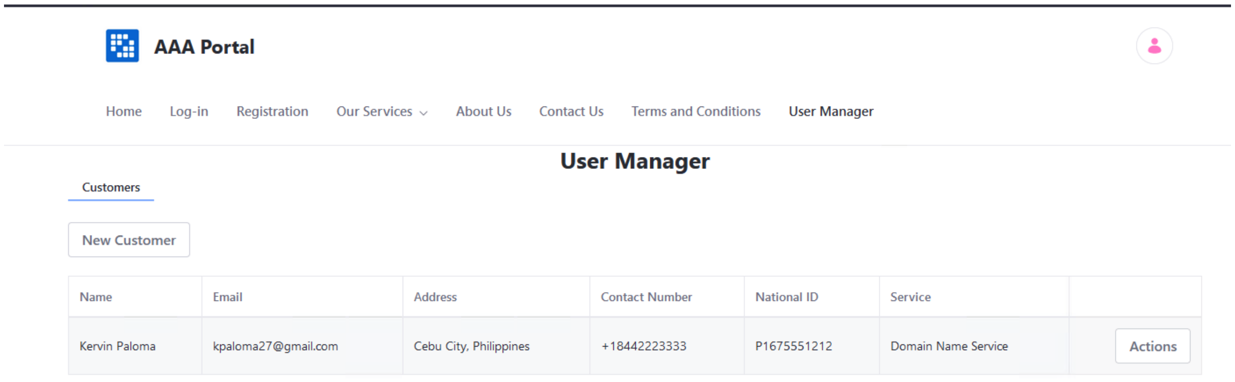
Email
kpaloma27@gmail.com

Address
Cebu City, Philippines

National ID
P1675551212

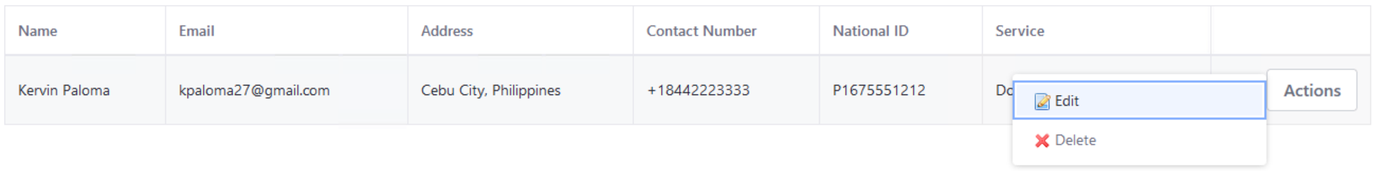
Contact Number
+18442223333

Services
Domain Name Service

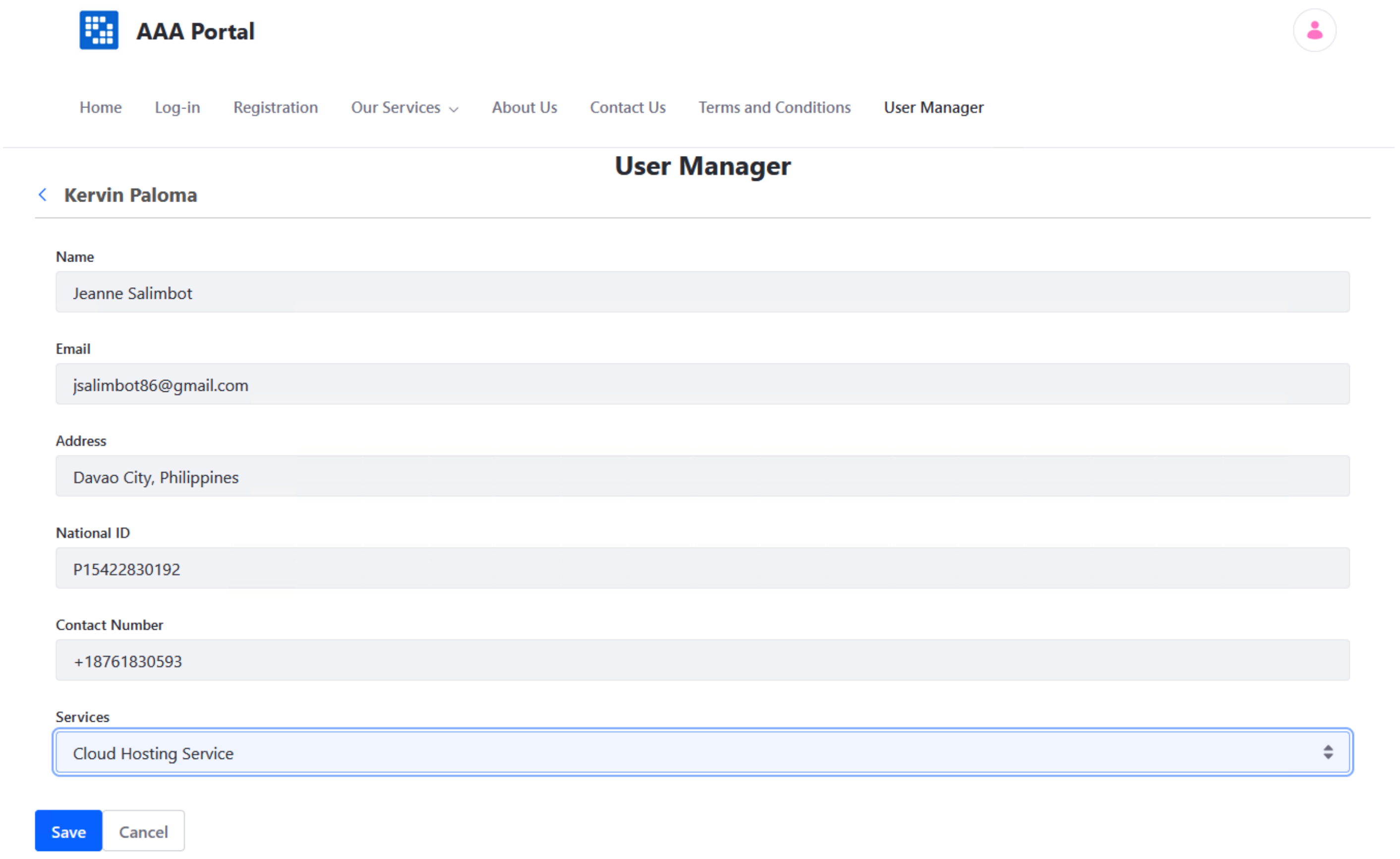


Name	Email	Address	Contact Number	National ID	Service	
Kervin Paloma	kpaloma27@gmail.com	Cebu City, Philippines	+18442223333	P1675551212	Domain Name Service	Actions

ii. Update Customer



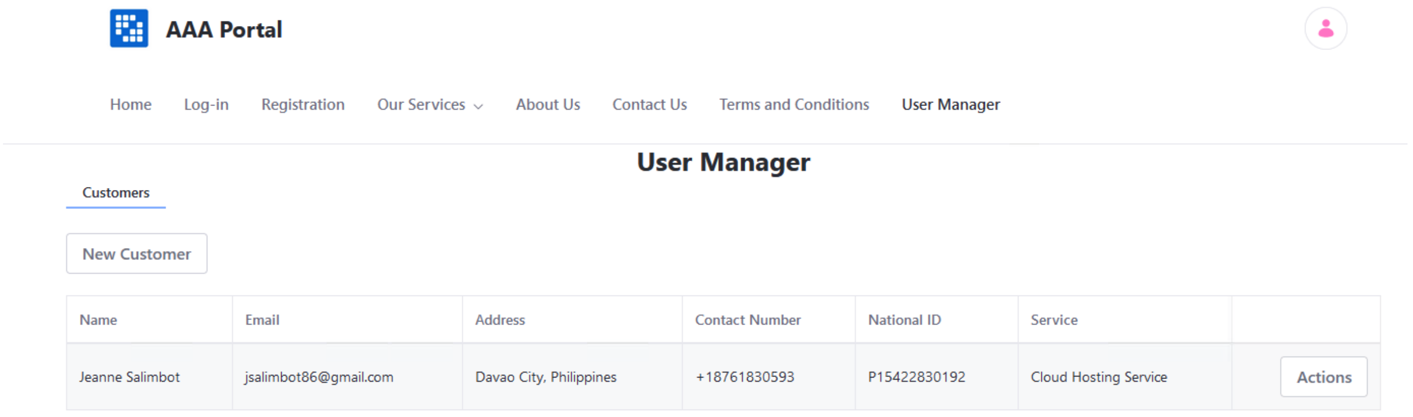
Name	Email	Address	Contact Number	National ID	Service	
Kervin Paloma	kpaloma27@gmail.com	Cebu City, Philippines	+18442223333	P1675551212	Dc	Actions



User Manager

Kervin Paloma

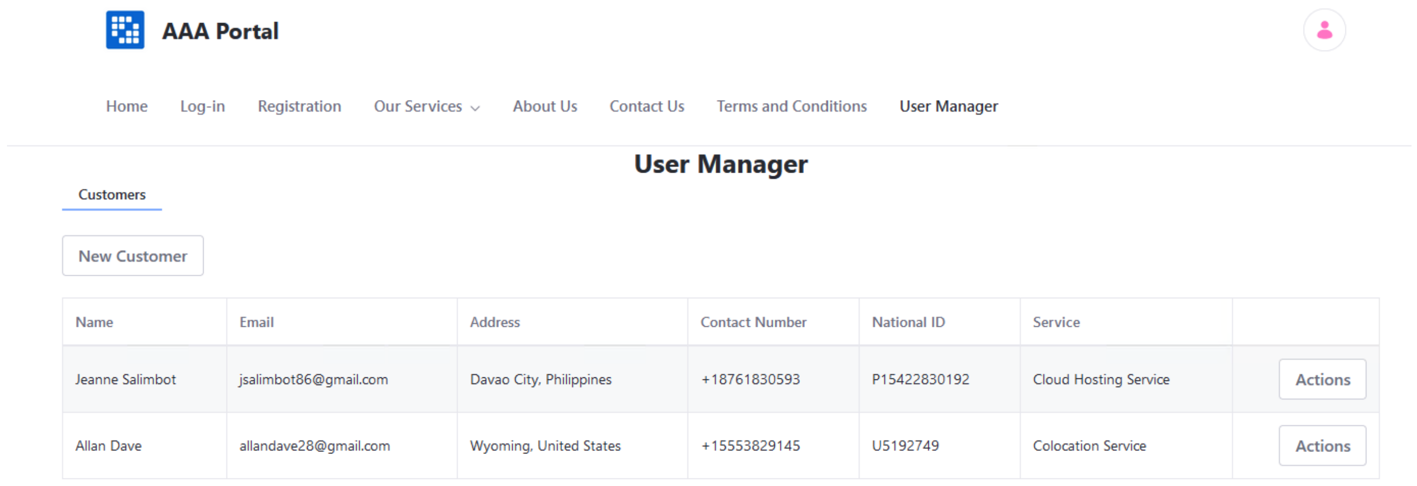
Name	Jeanne Salimbot
Email	jsalimbot86@gmail.com
Address	Davao City, Philippines
National ID	P15422830192
Contact Number	+18761830593
Services	Cloud Hosting Service



User Manager

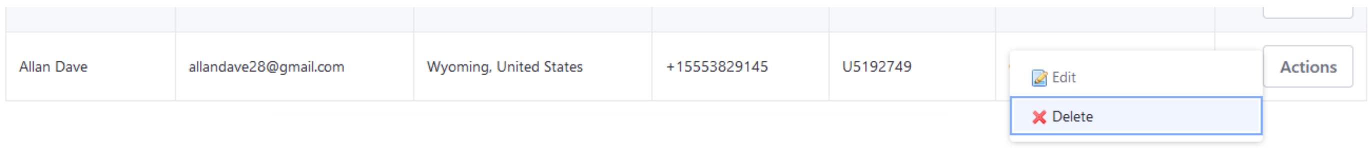
Name	Email	Address	Contact Number	National ID	Service	
Jeanne Salimbot	jsalimbot86@gmail.com	Davao City, Philippines	+18761830593	P15422830192	Cloud Hosting Service	Actions

iii. Delete Customer

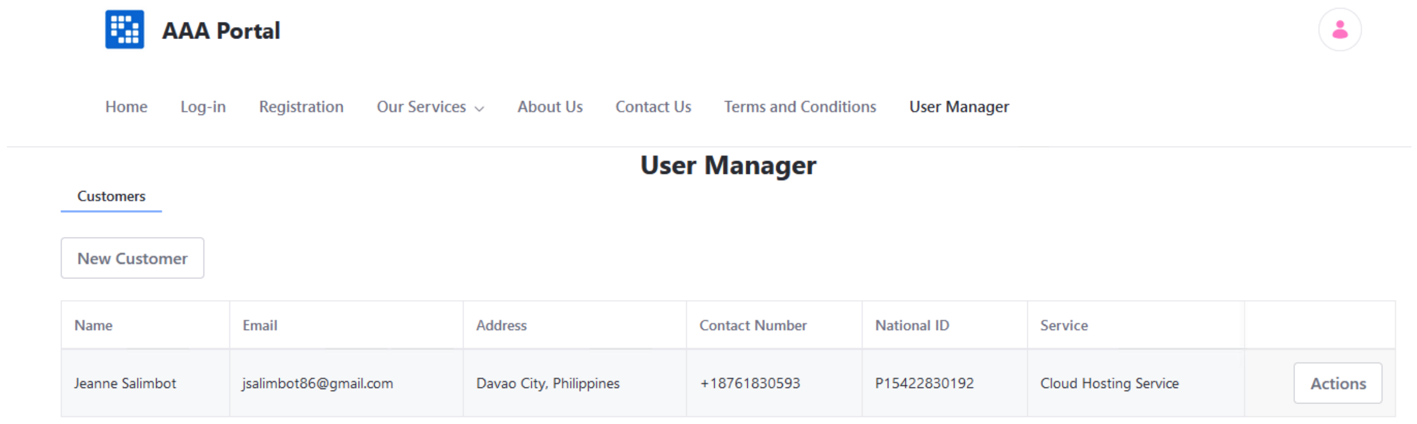


User Manager

Name	Email	Address	Contact Number	National ID	Service	
Jeanne Salimbot	jsalimbot86@gmail.com	Davao City, Philippines	+18761830593	P15422830192	Cloud Hosting Service	Actions
Allan Dave	allandave28@gmail.com	Wyoming, United States	+15553829145	U5192749	Colocation Service	Actions



Allan Dave	allandave28@gmail.com	Wyoming, United States	+15553829145	U5192749		Actions



User Manager

Name	Email	Address	Contact Number	National ID	Service	
Jeanne Salimbot	jsalimbot86@gmail.com	Davao City, Philippines	+18761830593	P15422830192	Cloud Hosting Service	Actions