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) Hosting |
| Module Name (NICF) | Development using Platforms |

|  |  |  |  |
| --- | --- | --- | --- |
| Student name | | Assessor name | |
| Ida Bagus Ketut Yoghantara | |  | |
| Date issued | Completion date | | Submitted on |
| 10 November 2022 | 16 November 2022 | | 16 November 2022 |
|  | |  | |
| 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:  Date: 14 November 2022 |

Content

1. Project background
2. Project Objectives
3. Project Requirement Specifications
4. Task 1
5. Task 2
6. Task 3
7. Task 4
8. Task 5

Project Background

Triple-A (AAA) company is the best web host solutions service company located in downtown of Myanmar. It provides all types of services which include domain name registration, Shared Hosting, Reseller Hosting, Cloud Hosting, VPS Hosting,  
Dedicated Hosting, and Colocation service, etc.

Project Objective

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).

Tools & platform used

1. Microsoft Word
2. Microsoft Power Point
3. Liferay Developer Studio
4. MySQL Workbench
5. Web Browser Mozilla Firefox
6. Diagram.net

Project Requirements Specifications

**3.1 Project Scope**

**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 user (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.

**3.2 Functional Requirement**

**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

Task 1

Task Statement:

Create the following items Under “Program Paradigm and Design Pattern” in Project Presentation

1. Identify and explain the characteristics of the object-orientated paradigm and relationship between the various classes from a given code scenario.
2. Identity the suitable design patterns base on the project scenario.
3. Examine how the object-orientated paradigm and its key principle is identified in each of the design patterns.

Solution:

1.

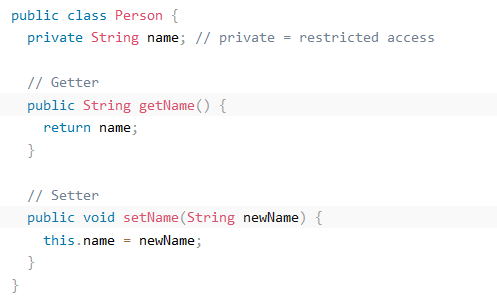
1. Explain briefly Object-Oriented Programming Paradigm

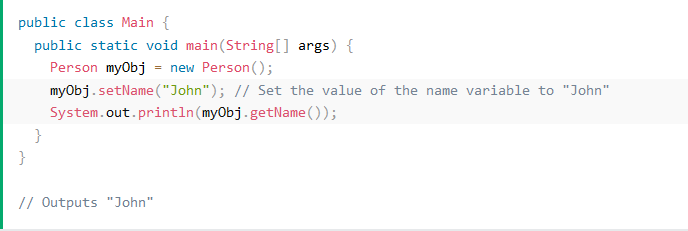
OOP is a programming paradigm where relies on the concept of classes and objects. It is used to structure a software program into simple, reusable pieces of code blue prints, which are used to create a individual instances of object.

1. Examine characteristics of Object-Oriented Programming Paradigm
2. Encapsulation

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, that it is a protective shield that prevents the data from being accessed by the code outside this shield.

Example:





1. Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

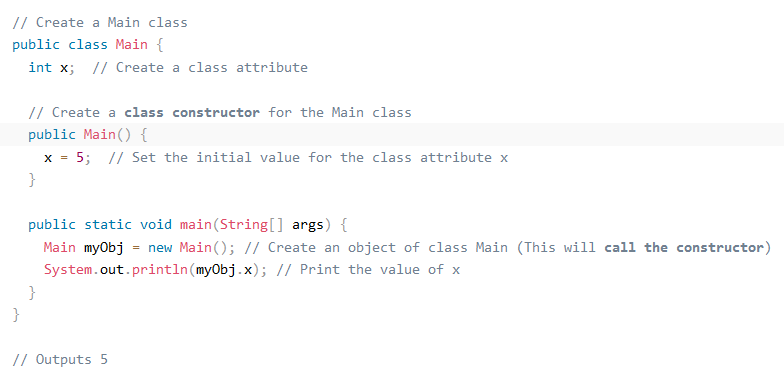
Example:



1. Constructors/ Destructors

Constructors or constructors in Java is a terminology been used to construct something in our programs. A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.

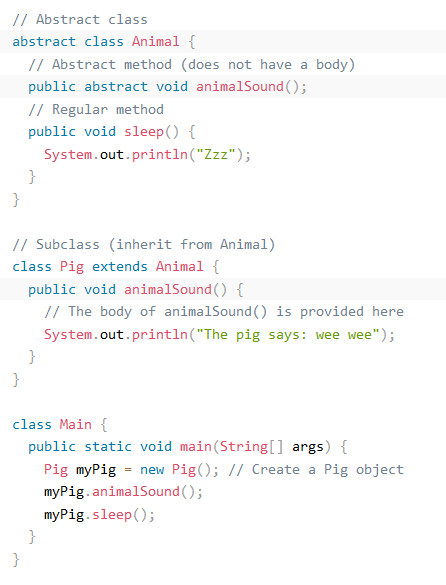
Example:



1. Abstract

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery

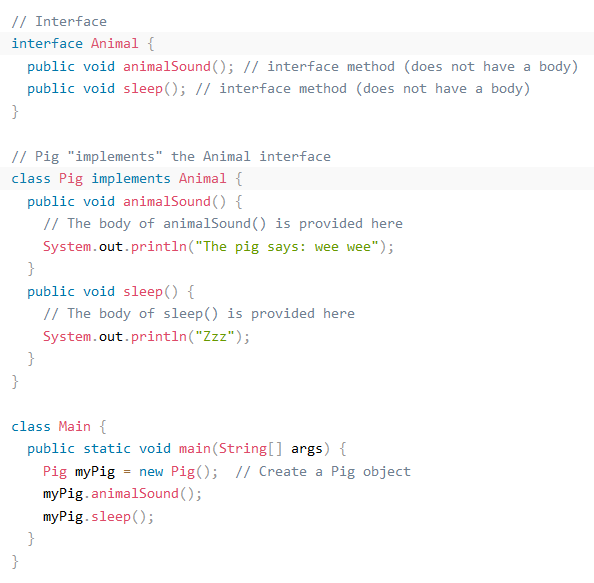
Example:



1. Interface

A totally abstract class is what an interface is. It consists of several abstract techniques (methods without a body). In Java, an interface is created with the interface keyword.

Example:



1. What are the differences between abstract classes and interfaces?

|  |  |  |
| --- | --- | --- |
| NO | Abstract | Interface |
| 1 | Abstractclasscan **have abstract and non-abstract** methods**.** | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2 | Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3 | Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4 | Abstract class **can provide the implementation of interface.** | Interface **can't provide the implementation of abstract class**. |
| 5 | The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 6 | An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |

2. Identify suitable behavioral design pattern for “AAA Portlet”

* Template method / pattern

Because we use many classes, most of which share relatively similar code, we could create a template for an abstract class with the majority of the functions predefined, and then create another class with a slight modification in accordance with the desired outcome.

* Observer method

Because in the portlet itself contains a one-to-many relationship, for example one customer could have more than one service, that’s where the observer pattern could come in handy to keep track of the state of the object in the portlet

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

OOP (Object Oriented Paradigm)

The Problem – solving methodology used by the OOP is entirely different. Because it concentrates parts of the system rather than be solved. In real world we can use object like cars and cat are good to examples.

Because that OOP is a programming technique or philosophy that divides code into object and their connections. Design patterns will provide tried-and-true methods of creating types or objects to handle a certain problem in a software

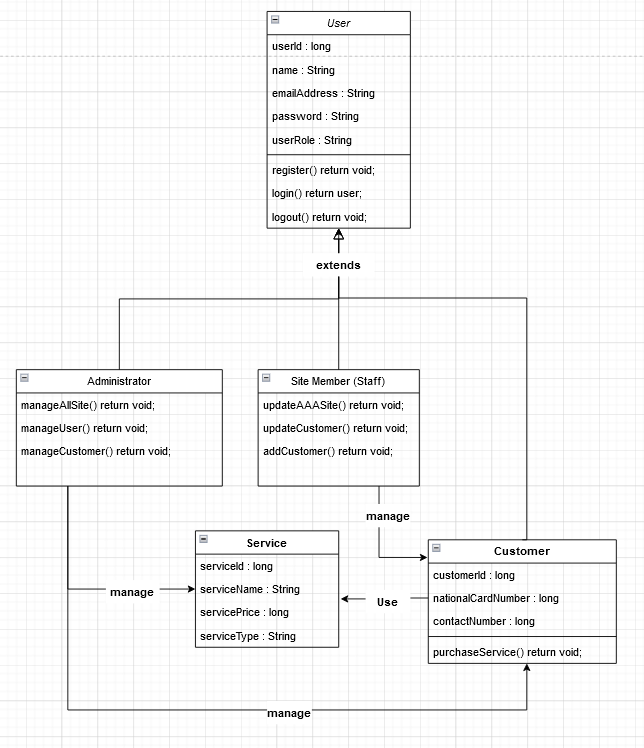
Task 2

Task Statement:

Create the following items in “System Design” Section in Project Report

1. Draw class diagrams that represent a simple structure based on the project scenarios using a UML tool.
2. Identify possible situations where design patterns would be beneficial and then develop the UML diagrams reflecting the design patterns.
3. Observe how the class diagrams are reflected from a given scenario by using the UML tool.

Solution:

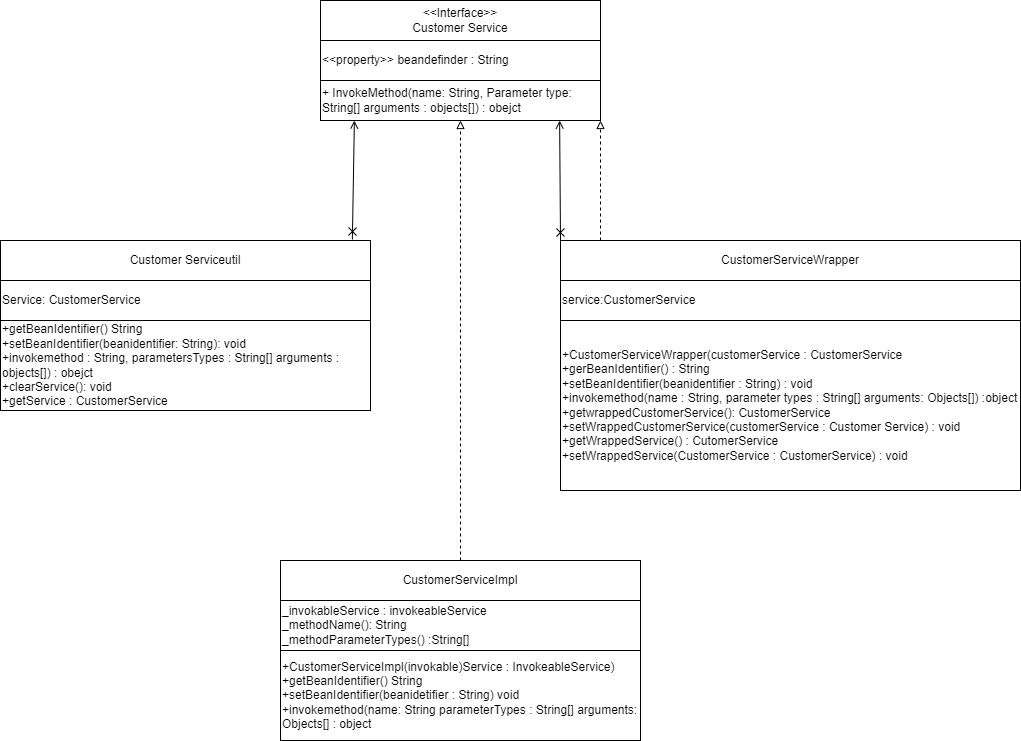
1. Draw class diagrams that represent a simple structure based on the project scenarios using an UML tool  
     
   
2. Identify possible situations where design patterns would be beneficial and then develop the UML diagrams reflecting the design patterns.

Benefits of Design Patterns

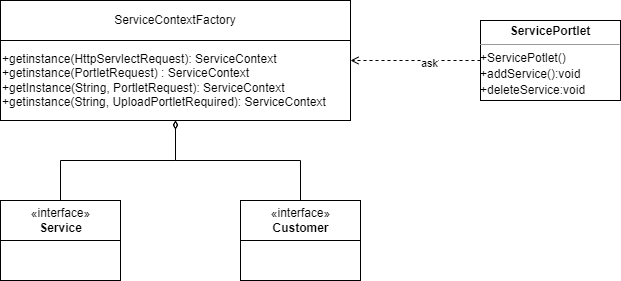
Design Patterns make it easier to modify software and reuse objects. It is easier to reuse and modify loosely connected items. Design patterns are created using a variety of tiny specialized objects. The system object performing the requested function is given functional responsibilities.

* This feature makes it easier to maintain binary compatibility with later versions.
* It is simple to create and solve future problems.
* Designers and developers can communicate more clearly and precisely when using design patterns.

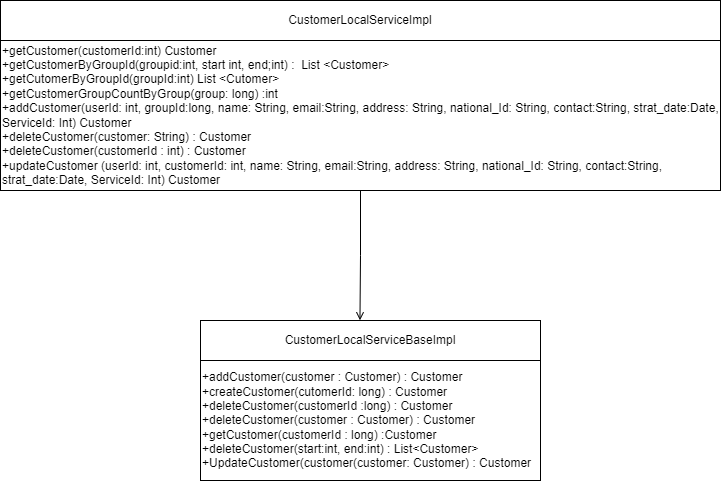
Façade Patterns



Factory Patterns



Template Patterns



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

The authoring design pattern in the service portlet is the factory design pattern code. The ServiceContext of a service portlet obtains a service class from a ServiceContextFactory. Similarly, the customer portlet employs the factory design pattern. A class diagram contains class names and properties, as well as class links and methods. As a result, the class diagram displays the application's actual source code. The prototype design pattern code clones an existing object rather than creating a new object as the next type of design pattern code for creation.

CustomerModelImpl clones the parent class CustomerModel's clone() function. CustomerModelImpl clones the parent class CustomerModel's clone() function. After analyzing the code, I quickly created a class diagram using UML tools. Because of structural design patterns. In source code, the observer design pattern is used. The LocalServiceImpl class, for starters, extends and overrides the methods of the hyper-abstract CustomerLocalServiceBaseImpl class. The Observer pattern is the last design pattern I created, as shown in the diagram.

Task 3

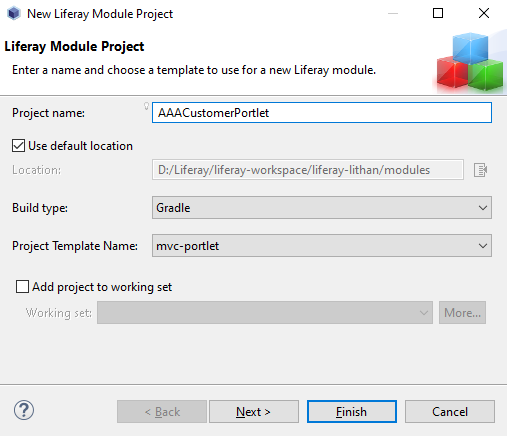
Task Statement:

Create the following items under “System Development” in Project Report

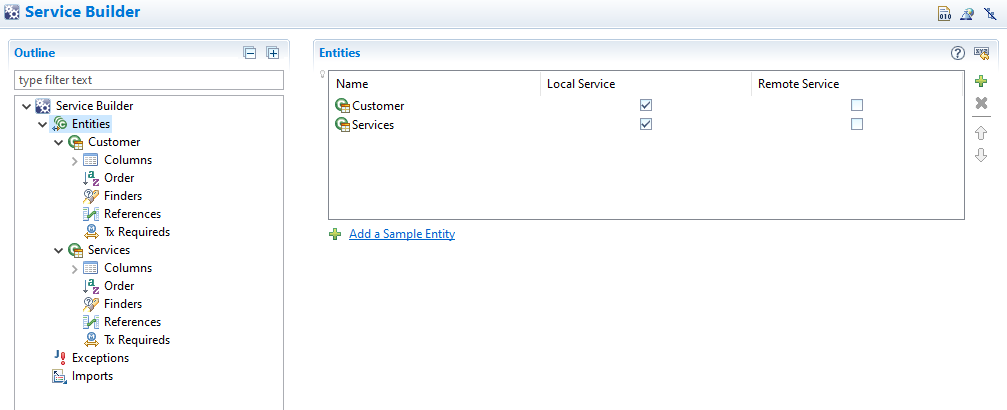
1. Set up a project, build an application based on your derived UML class diagrams and design patterns. Provide screenshots as evidence.
2. Develop an application that implements design patterns and utilizes techniques to produce secure code. Provide the implemented code as evidence.

Solution:

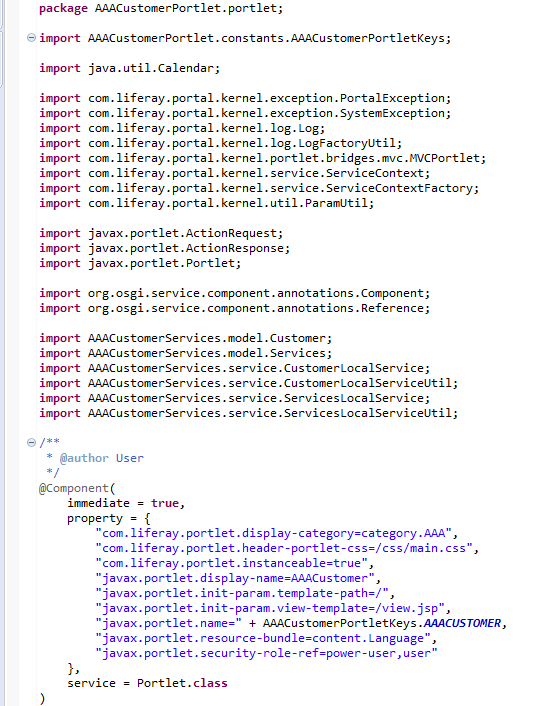
Create MVC Portlet and Service Builder to manage AAA Customer’s data



Create required entities and their properties

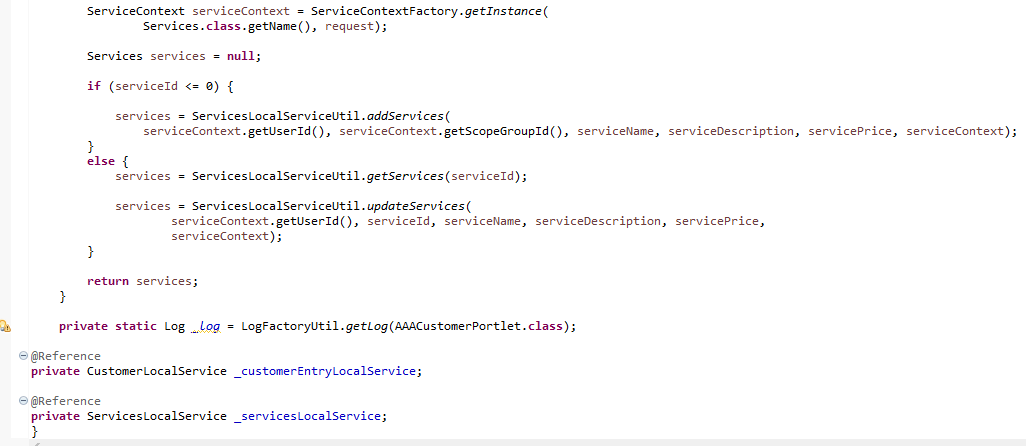


Portlet Controller

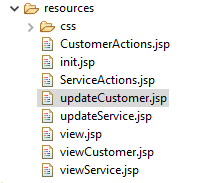








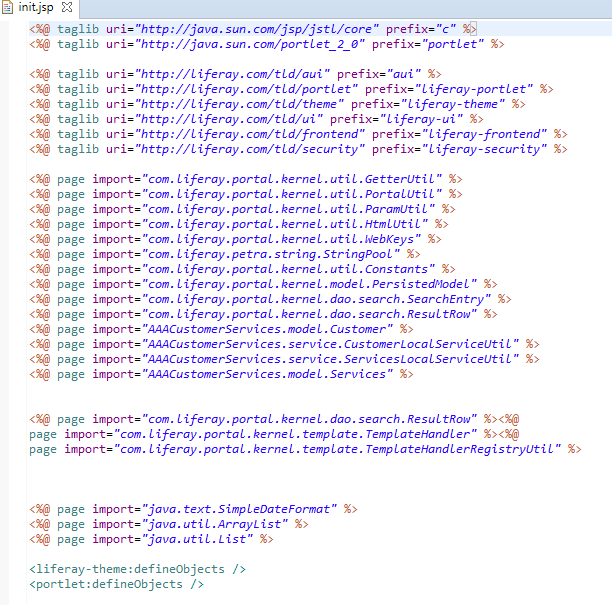
View (Jsp Files)



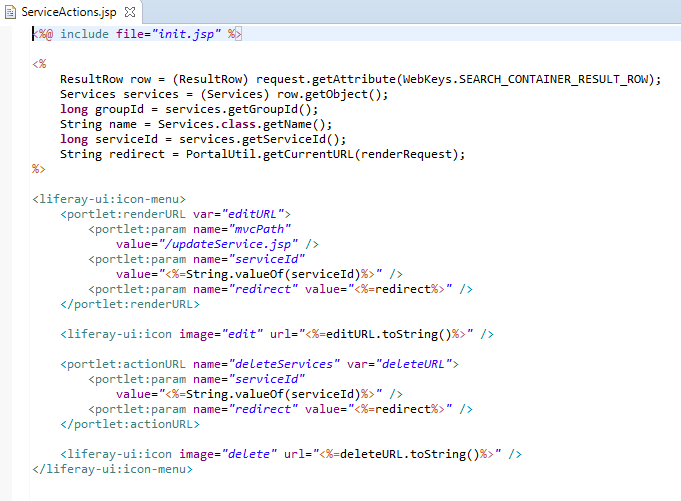
CustomerActions.jsp



Init.jsp



ServiceActions.jsp



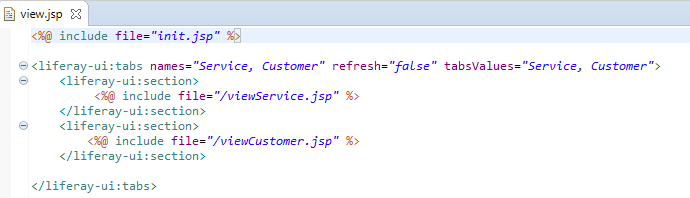
updateCustomer.jsp



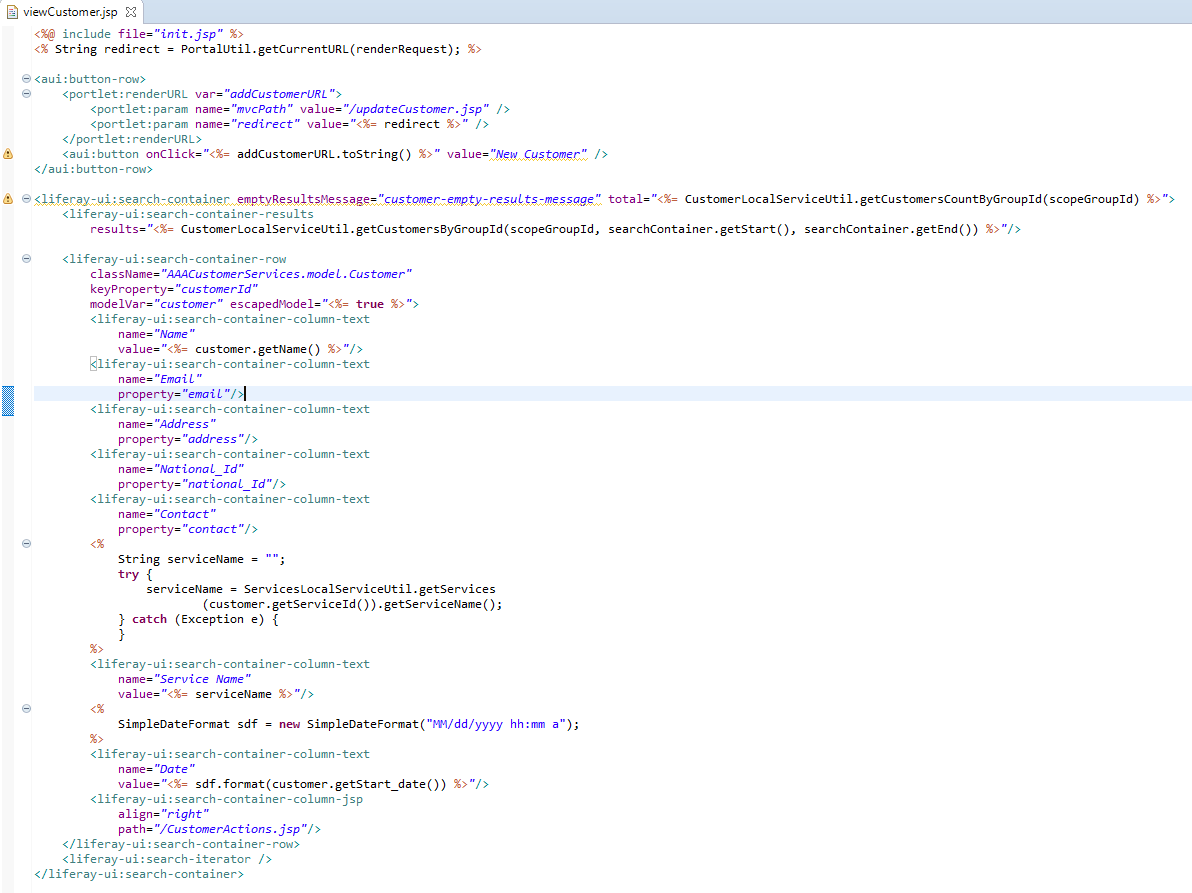
updateService.jsp



view.jsp



viewCustomer.jsp



viewService.jsp



1. Task 4

Task Statement

Create the following items under “Evaluation of Design Pattern” in Project Report

1. Discuss the use of design patterns for the given purpose and consequences by applying design patterns.
2. Investigate how different design patterns can work within a range of different scenarios.
3. Identify the appropriate design pattern from the investigation.
4. Evaluate and justify the design patterns that you had identify in each of the scenarios.

Solution

1. Discuss the use of design patterns for the given purpose and consequences by applying design patterns.
2. Discuss usage of design pattern in “AAA Customer Management Portlet”

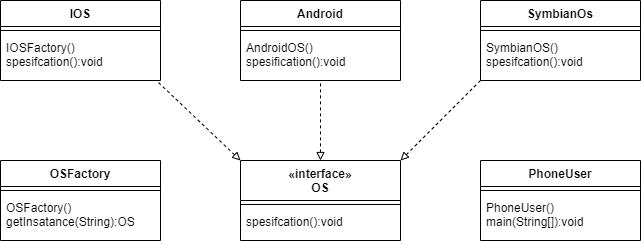
* The use of design patterns in the Triple A Portlet is critical for maintaining neat and clean code.
* Even though the portlet has many layers of services and classes, not every design pattern will be useful when used by the portlet.
* We must use this information to determine the correct and optimal design pattern for the portlet. Because of the multiple layers of java classes, the portlet has already used the Factory pattern. And, because we don't use any filtering functions in the portlet, using the filter pattern is probably not the best pattern to adapt into it.
* Especially given how the portlet implements the method and the fact that there are multiple layers of java classes.

1. The consequences of design patterns which used in the project

* Using the factory pattern in a portlet or any OOP project will result in cleaner and tidier code, but if used excessively will result in a large number of required classes for main class implementation.

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

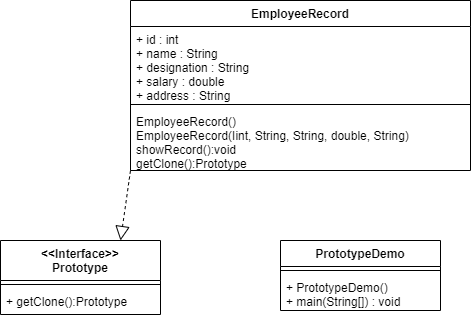
**Factory Pattern**



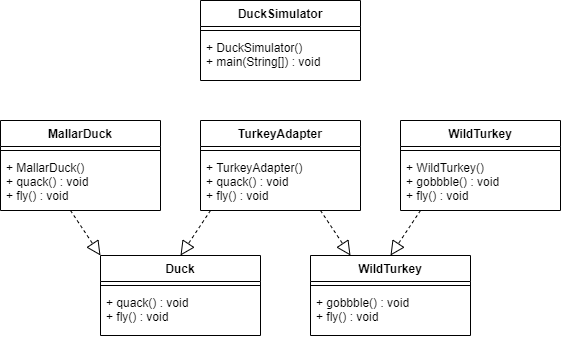
**Singleton**



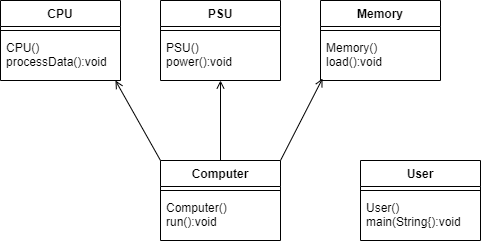
**Prototype**



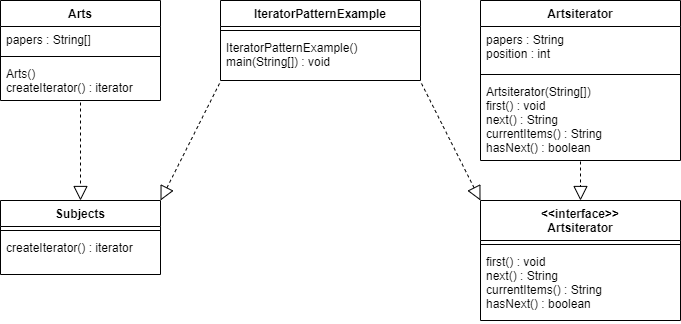
**Adapter Pattern**



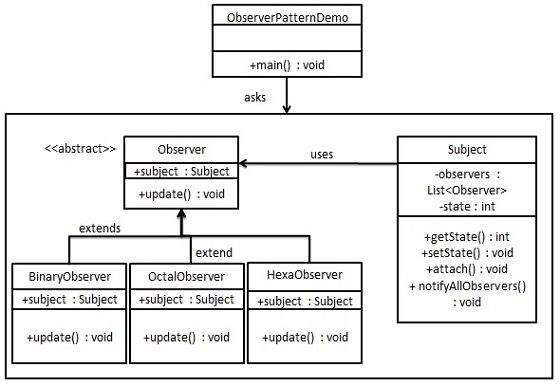
**Façade Pattern**



**Iterator pattern**



**Observer Pattern**



1. Identify the appropriate design pattern from the investigation.  
     
   Factory Pattern

The Factory Method pattern will be used in a project to create an object by creating a runtime interface. Factory patterns make code more resilient, easier to modify, and less dependent. For example, we could simply change the implementation of a class because the client is unaware of it. After all, it only provides interface code, not implementation.

Prototype

Is used to describe the complexities involved in creating new client instances. The goal is to clone an existing object rather than creating a new instance, which may require costly actions, and this technique saves time and money, especially if the creation of objects is a large operation.

Façade Pattern

The facade pattern is a design pattern that simplifies the operation of a program by combining all of its functionalities into a single interface. This pattern would be useful in the AAA Portlet because it simplifies the creation of user objects and their subclasses. For example, we could create a new administrator object and hide its properties and methods within an interface.

Observer Pattern

Because the portlet contains a one-to-many relationship, for example, one customer may have more than one service, the observer pattern may be useful in keeping track of the state of the object in the portlet.

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

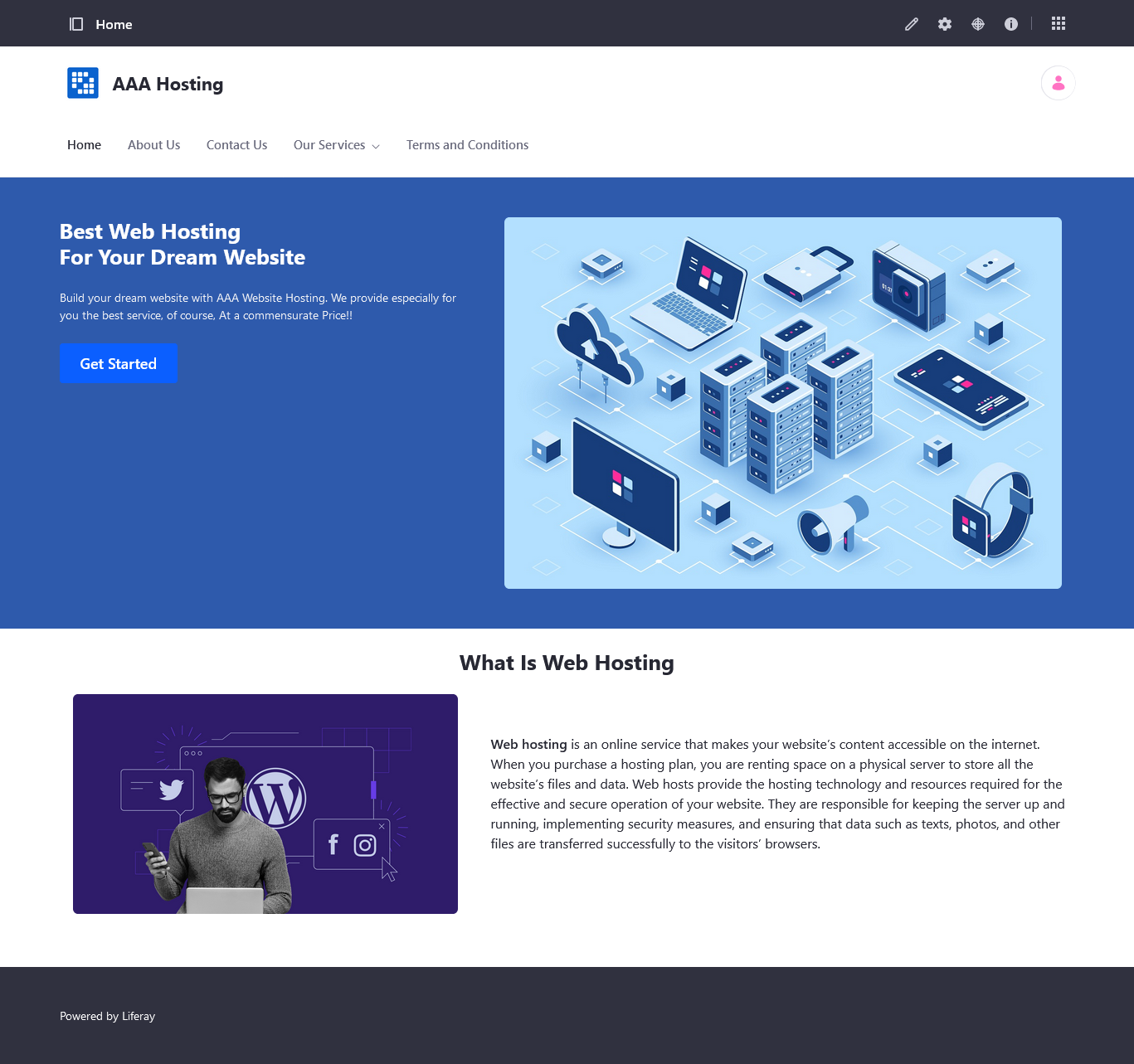
* Factory methods, in this pattern the creation of class instances is the responsibility of the subclass. Even if the interface provided when the project is built is a superclass, subclasses can make changes to the object being built.
* Prototype, this pattern makes it possible to clone existing objects without code. This means that this type of operation is not class dependent.
* Facade pattern hides the complexity of the system and provides an interface to the customer using which the customer can access the system. This type of design pattern belongs to the structural pattern because it adds an interface to the existing system to hide its complexity.
* Observer pattern can contain a one-to-many Relationship

1. Task 5

Task Statement

1. Provide screen capture of developed pages and hosted application in Project Presentation

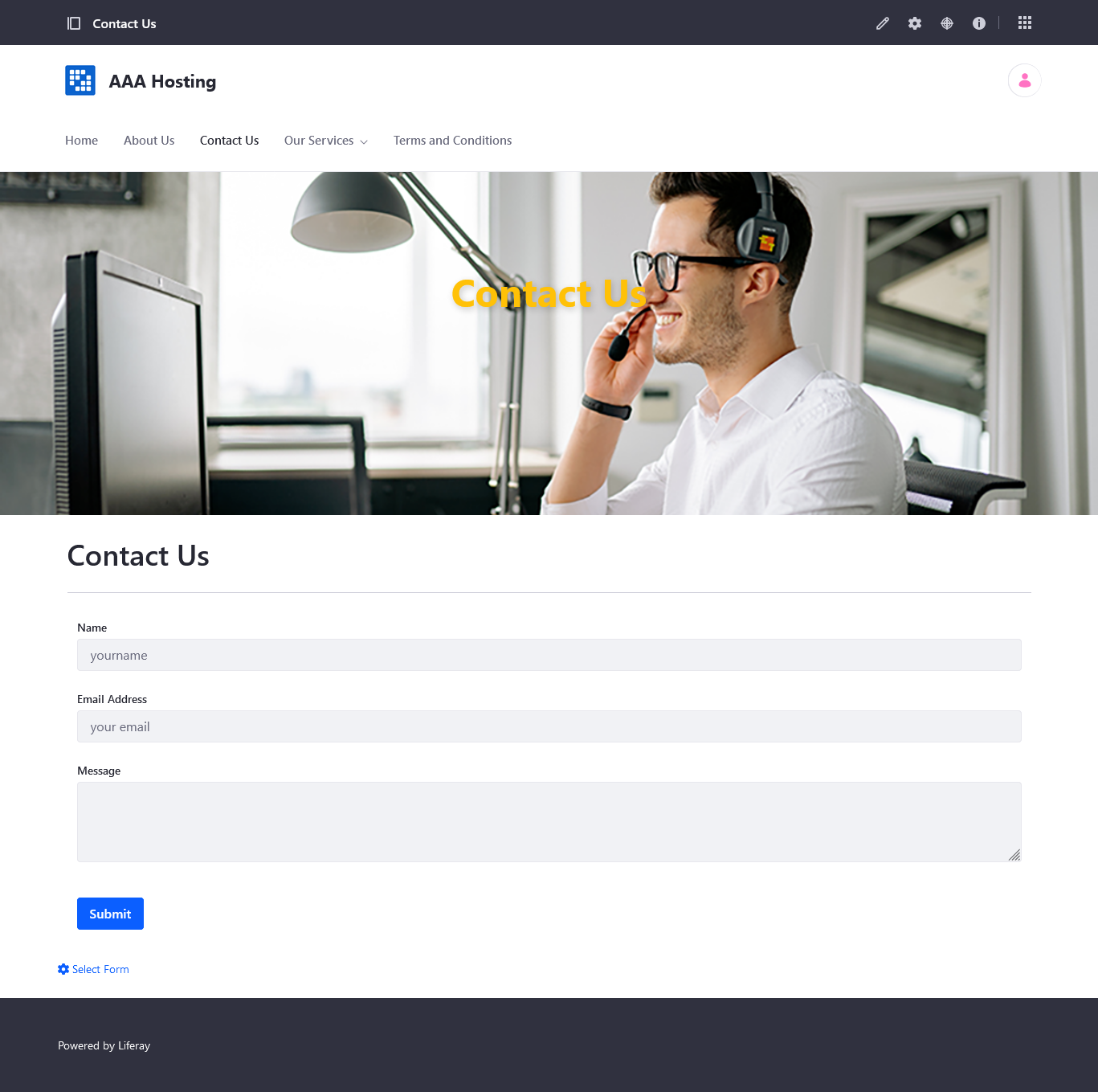
Solution

Home  


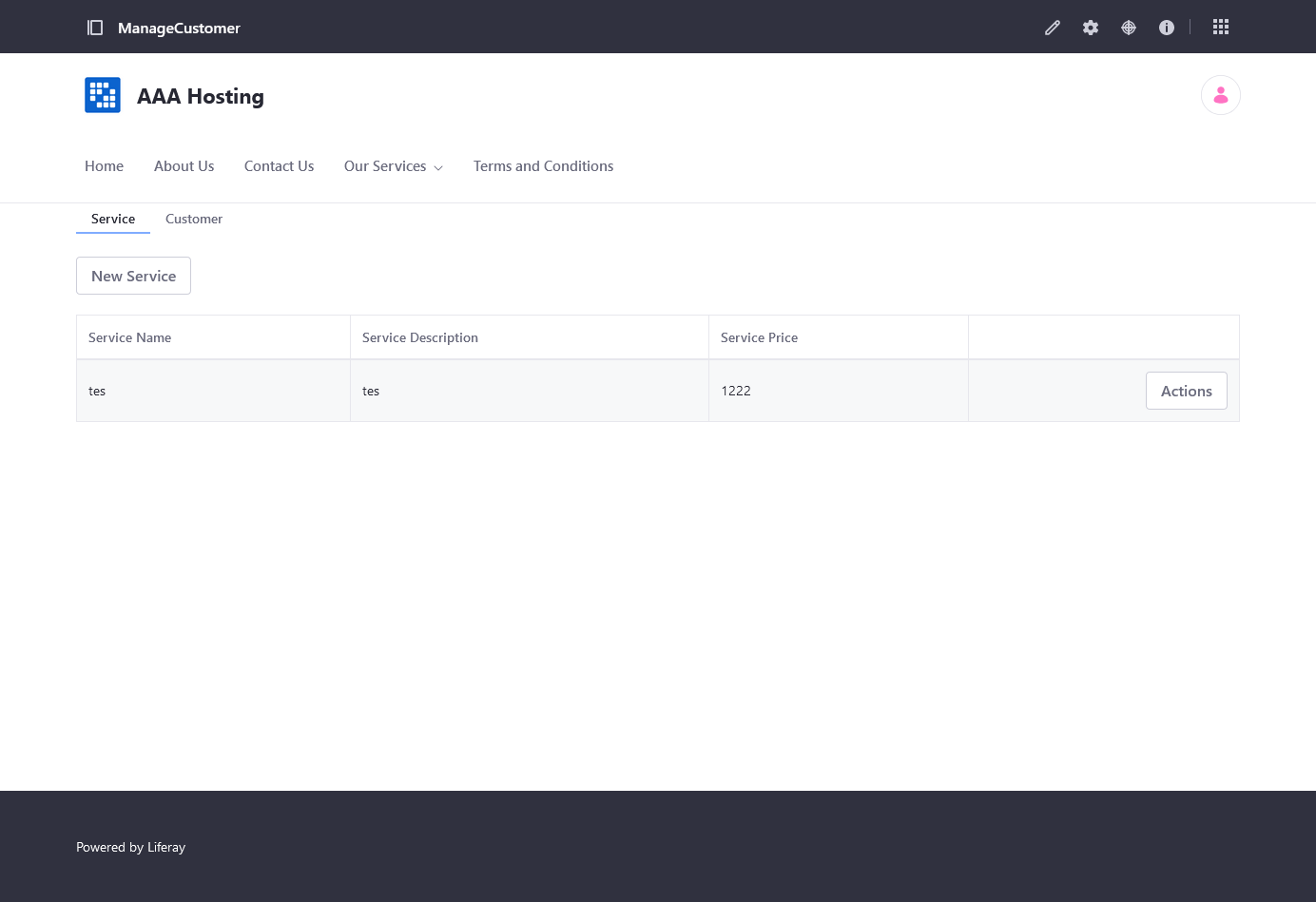
About Us

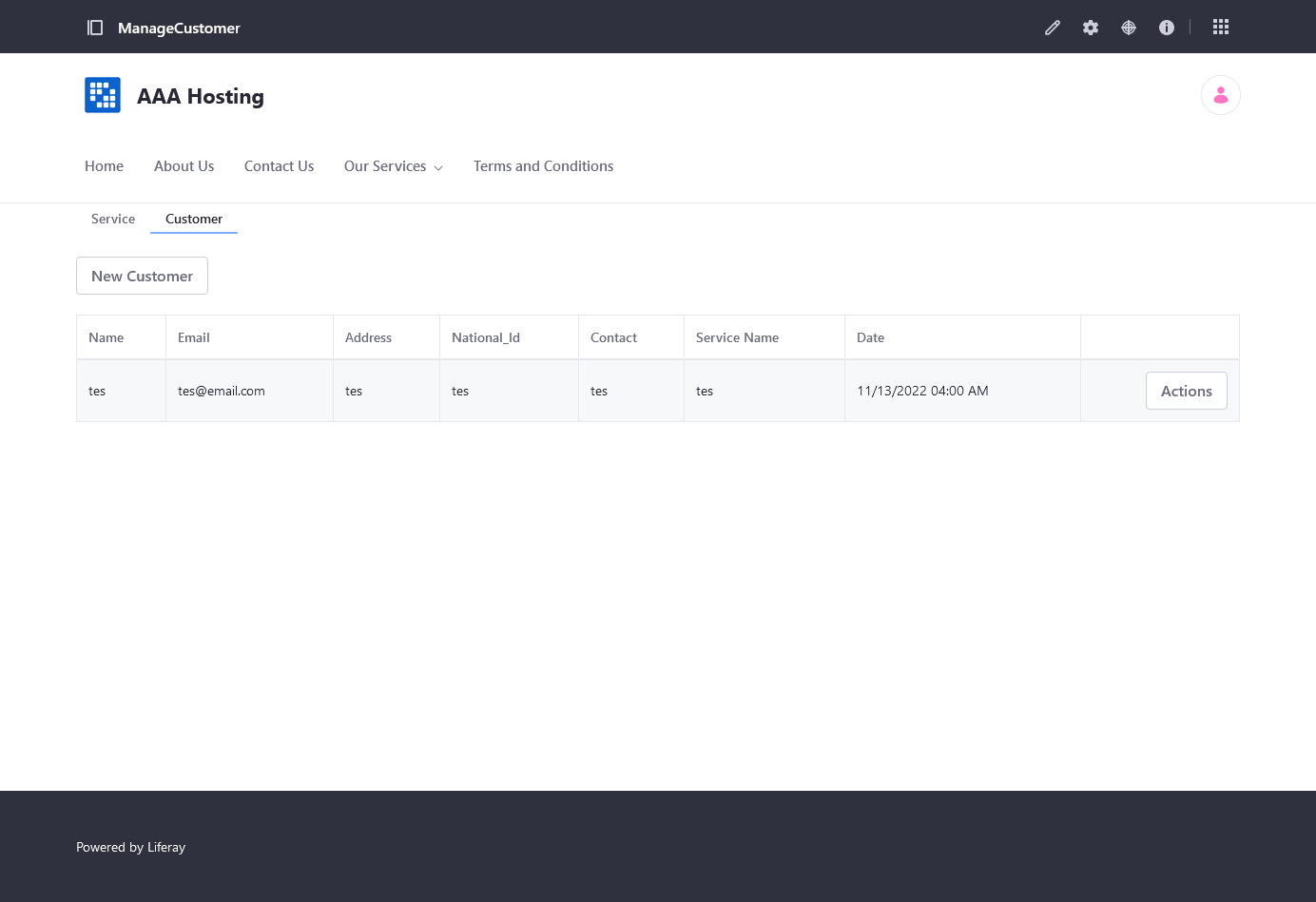


Contact Us

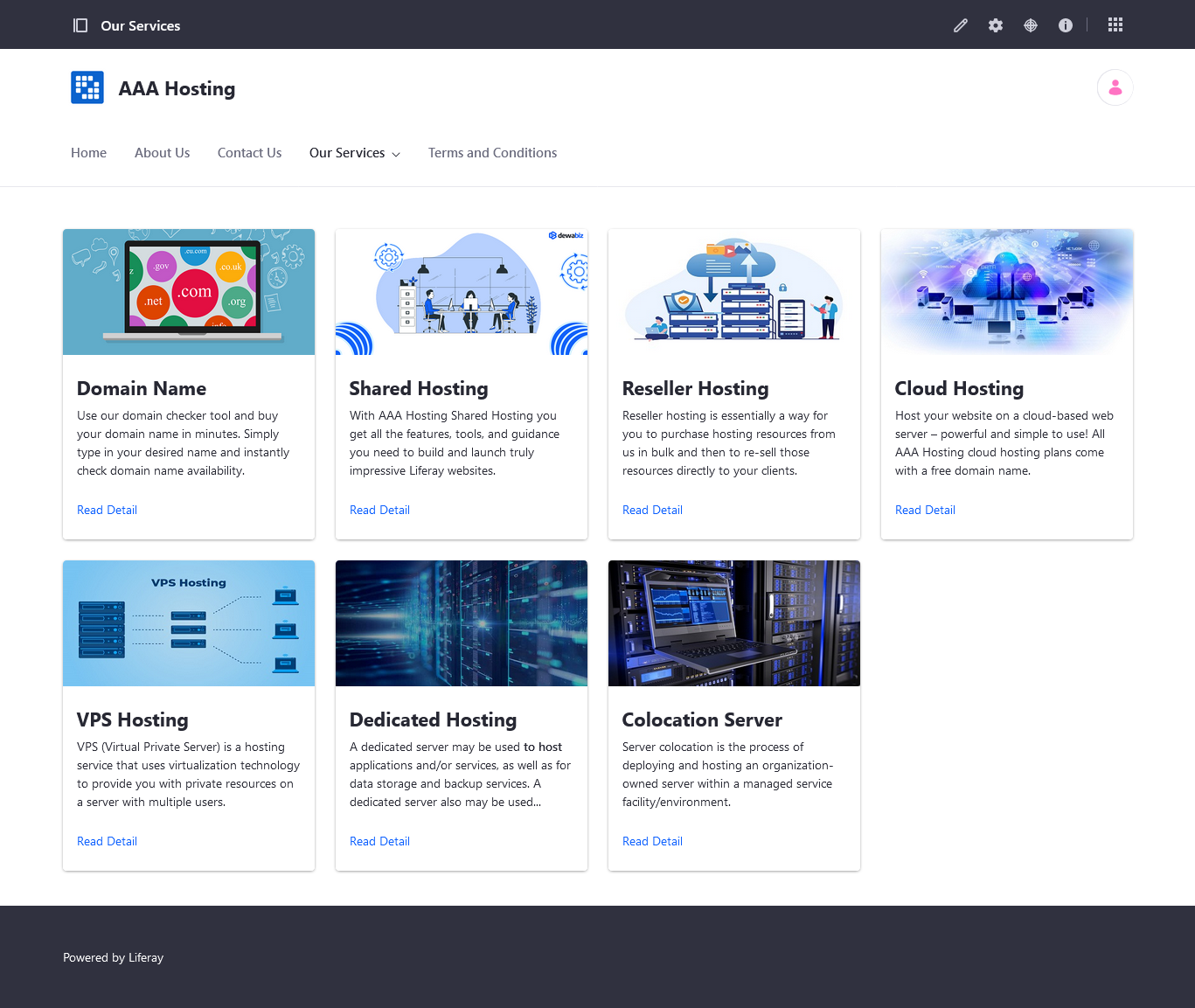


Manage Service & Customer





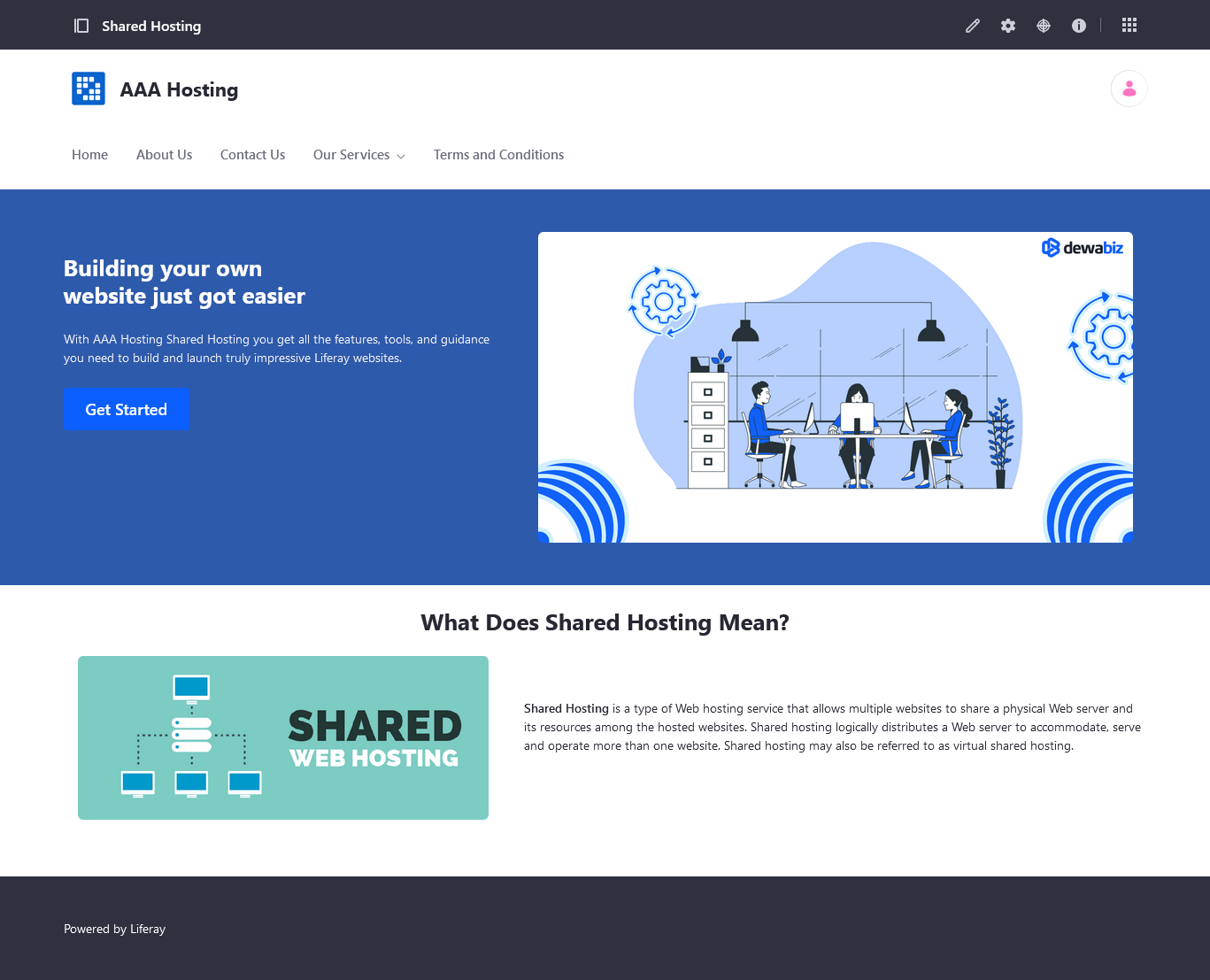
Our Services



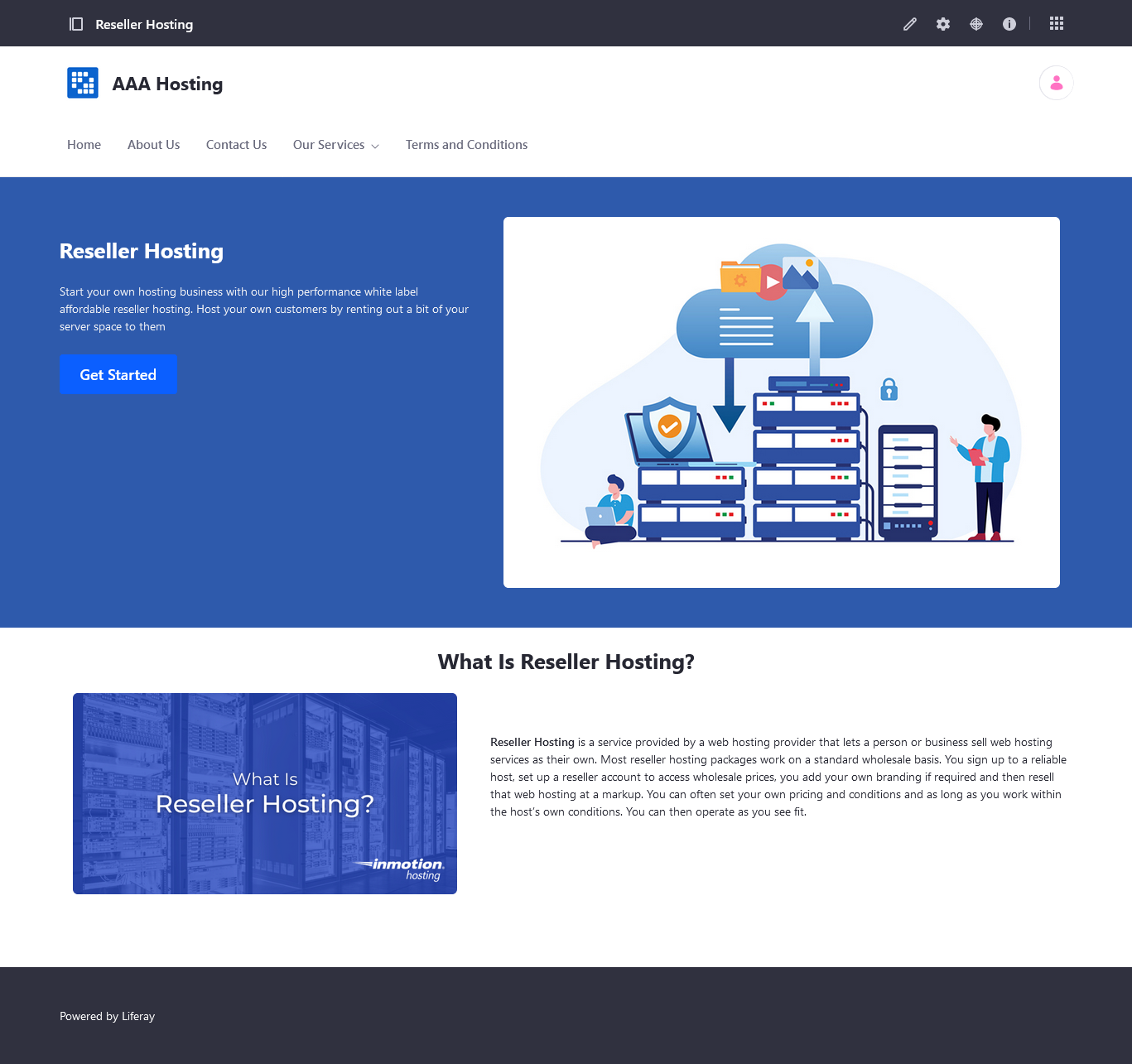
Domain Name



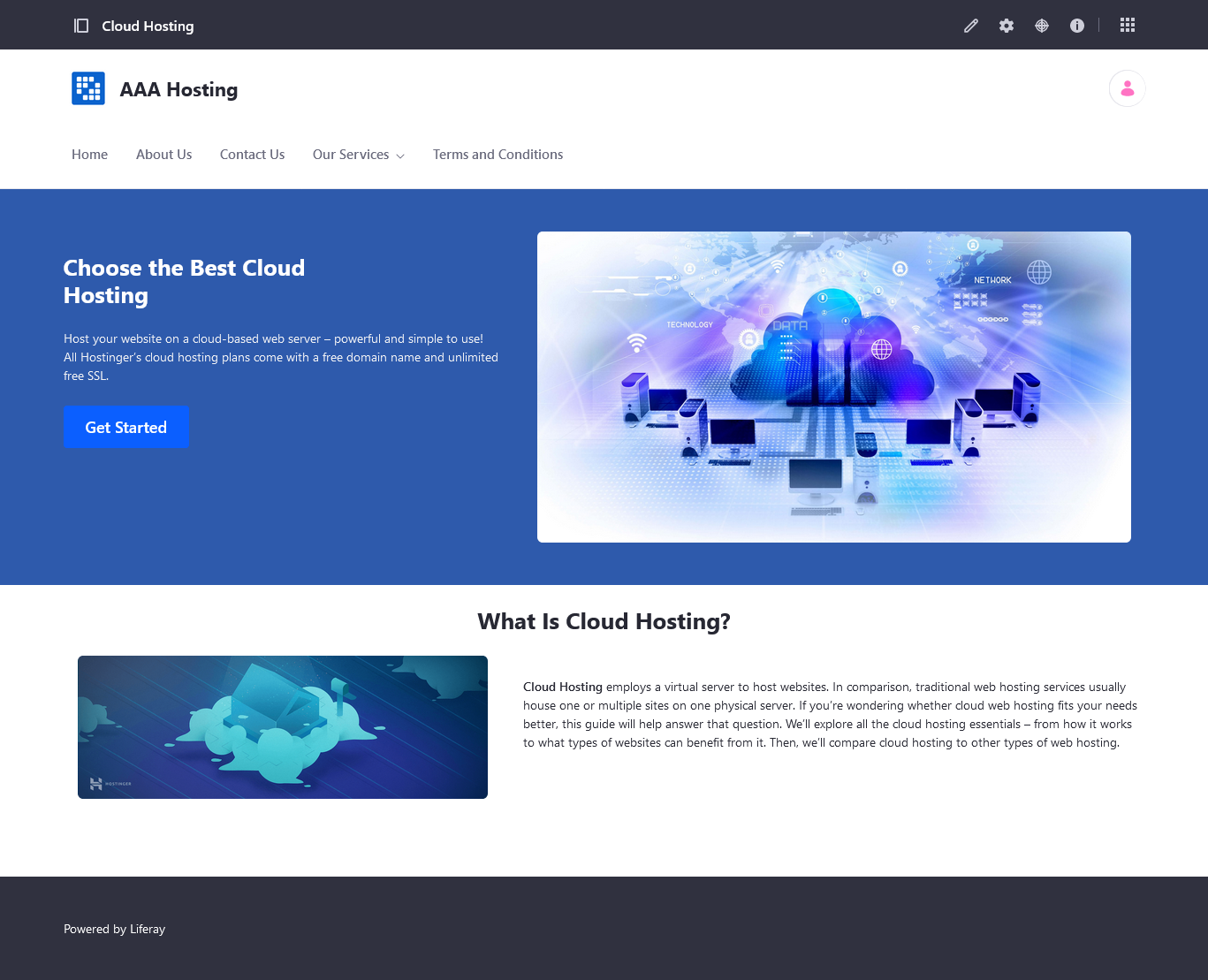
Shared Hosting



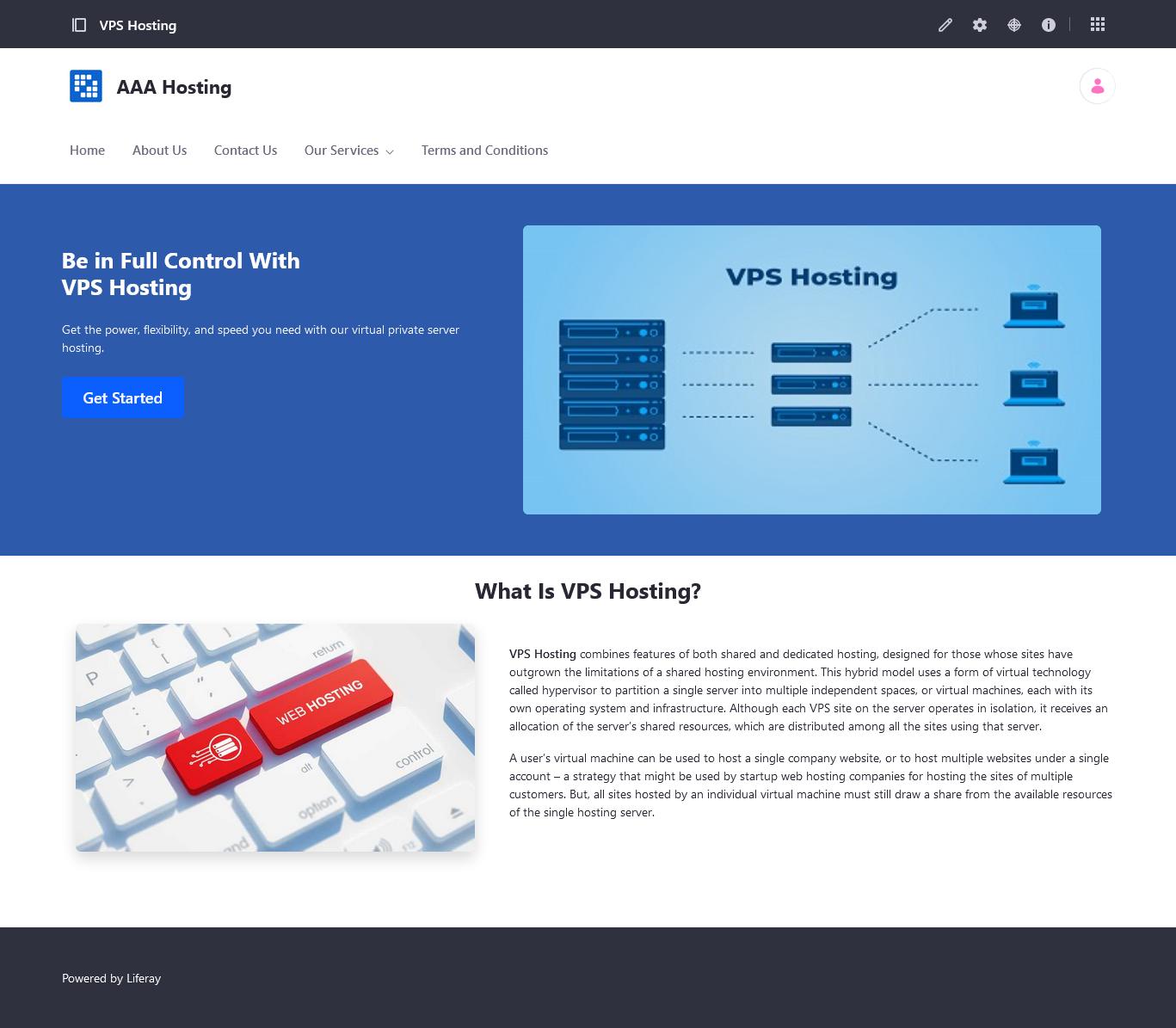
Reseller Hosting



Cloud Hosting



VPS Hosting



Dedicated Hosting



Colocation Server



Terms and Conditions

