**Technical Project Report on**

**Market Place Application**

**Name: Keyur Kirti Mehta**

Professor: Ryan[Rybarczyk](https://iu.instructure.com/courses/1613565/users/5588844)

Contents

[**1.** **Introduction** 3](#_Toc505111037)

[**2.** **Technical Architecture** 3](#_Toc505111038)

[**I.** **Java RMI** 3](#_Toc505111039)

[**II.** **MVC Architecture** 3](#_Toc505111040)

[**3.** **UML Diagram** 4](#_Toc505111041)

[**4.** **Screenshots** 5](#_Toc505111042)

[**I.** **Server start and Login view** 5](#_Toc505111043)

[**II.** **Register User** 5](#_Toc505111044)

[**III.** **Admin Login** 6](#_Toc505111045)

[**IV. Admin View** 6](#_Toc505111046)

[**V.** **Customer Login** 7](#_Toc505111047)

[**VI.** **Customer View** 8](#_Toc505111048)

[**5.** **Conclusion** 8](#_Toc505111049)

[**6.** **References** 8](#_Toc505111050)

# **Introduction**

Marketplace Application is an e-commerce platform to purchase goods and get it delivered at the door step. This application is hosted on server which makes it accessible to all its user online from anywhere. This application gives user to browse different products based on different categories and will be available for purchase. Also administrator of this application will have authority to control the details of the products.

This document will give detailed technical and general information about the application. Also it will give details about its architecture and its usage. This report will also have screenshot of functionality which will give idea to user about its different modules.

# **Technical Architecture**

## **Java RMI**

Java RMI is API which is used to communicate between client and server. It uses stub and skeleton framework for the communication. So, basically stub is an object on the client side which initiates the communication with the server. It lookup for the similar string in server as of in the client. Server will rebind with the clients. [2]

In this project, views on the client side will lookup for the connection string on the server. Each view will have different connection string, which will then point to different interfaces on the server side. Server is rebind it with the connection to the client.

## **MVC Architecture**

MVC is Model-View-Controller architecture. It is a design pattern used in software designing. This architecture separates the different modules with each other, so as changes in one will not or have little impact on the other. [1]

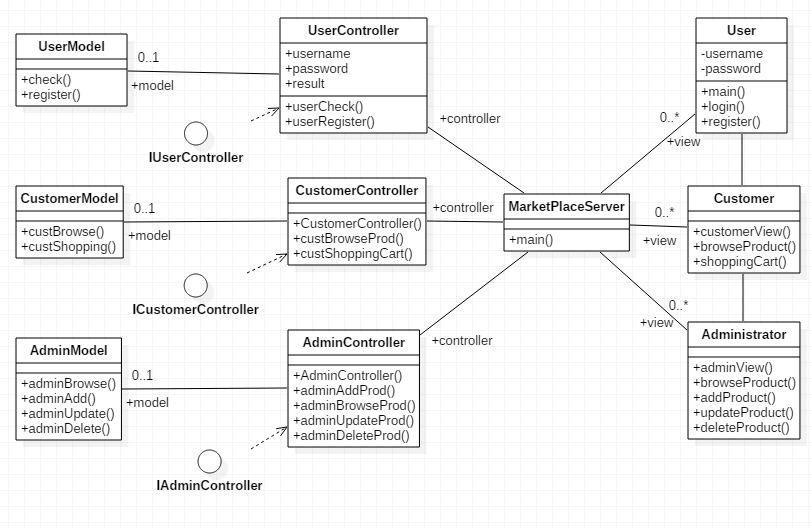
**Model** – This component holds to core business logic of the functionality. It processes all the requests received from the views.

**View** – This is the actual display which user can see. It will have the data along with the information. Views will be different for different type of the users. This can be GUI or the console where user can interact with it.

**Controller** – It acts as a mediator between the model and views. It takes request from views and send it to the model for processing. And similarly it response (processed request/ result) from server to the client.

In this project, we have MVC architecture in client server environment. The details explanation will be given later in the Domain model section.

# **UML Diagram**



The domain model designed is based on the MVC architecture.[3] Where view is on client side and controller and the model are on server side. There can be multiple clients. So, if controller is on client side and is coupled with it, changes in views will result in controller update. So, because of this, the controller is kept on the server side.

**Views:** Once client starts, user will be displayed with general user view. And there are 2 additional views created which will be displayed based on the type of the customer logged into the application. These are Customer and Administrator views. Any requests from these clients will go to the server through RMI. Views will pass the data (request) through the objects of interfaces. Interfaces will hide the actual information from the client and will provide abstraction layer to the application.

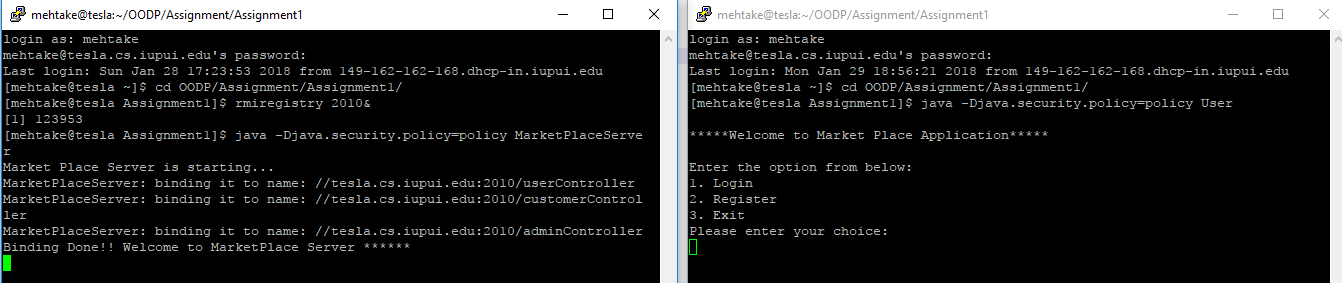
**Controller:** Controller are designed on the server side. Interfaces for the controllers are implemented on the server side. Any requests (RMI) from views will come to controllers where interface methods are implemented from server class where RMI connection is bind. Controller than will forward the request to the models through the objects of models. Each view has its own controller that will handle its request.

**Model:** Similar to controller model are designed on the server side. And for controller have its own model which will process the request forwarded by the controller. Here request will be processed and the result will be sent back to views through controller. And views will display / take necessary action based on the response.

In this designed all the components are loosely coupled with each other. So, the changes in one will have no or littler effect on the other component. i.e. changes in any 1 particular model will have no effect on other models or the controller or views. Similarly for changes in view will not make model to update its own logic.

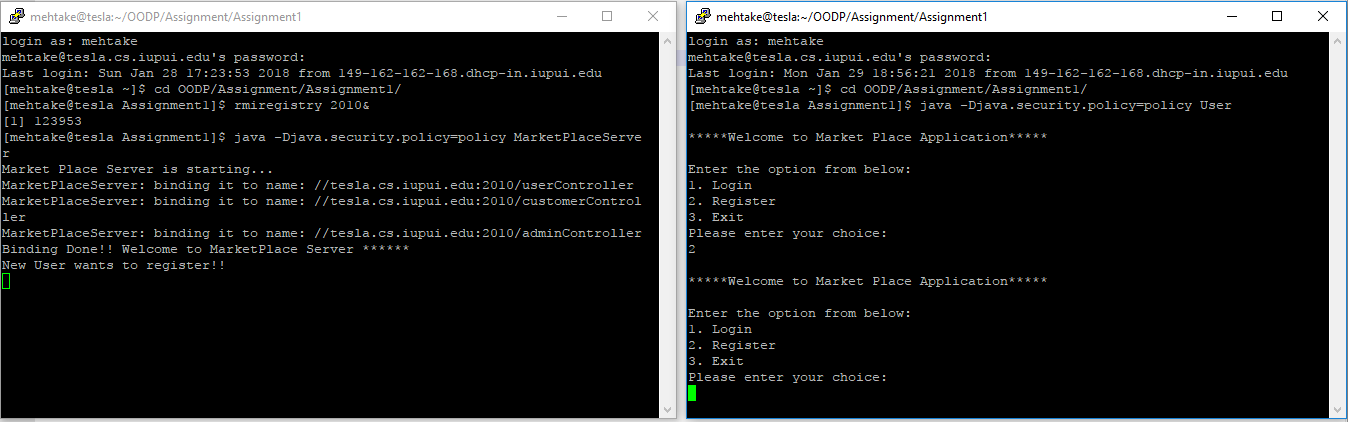
# **Screenshots**

## **Server start and Login view**



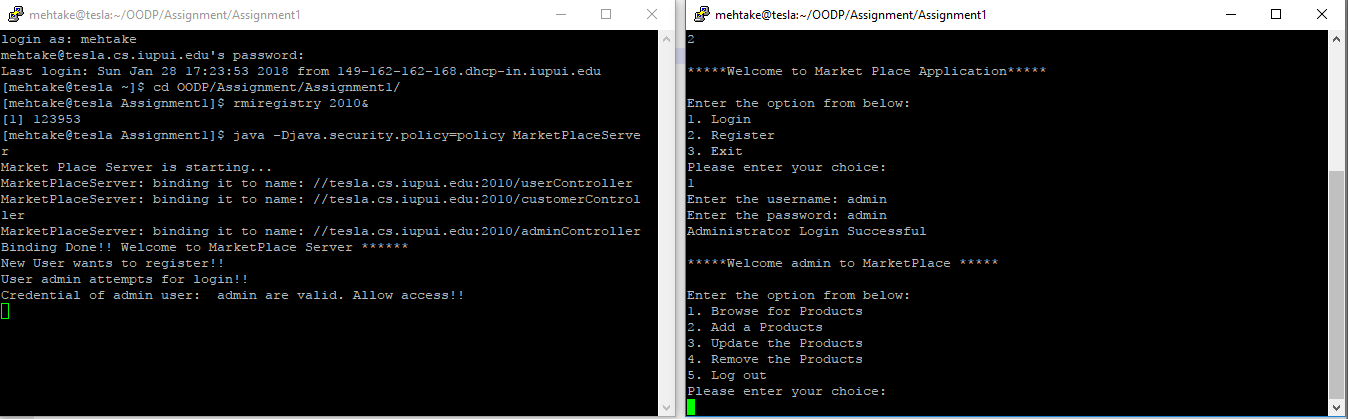
Once the server start, it will bind with the view on the client. And server will display start message. And on the other console, client will display initial user view with option to login and register.

## **Register User**



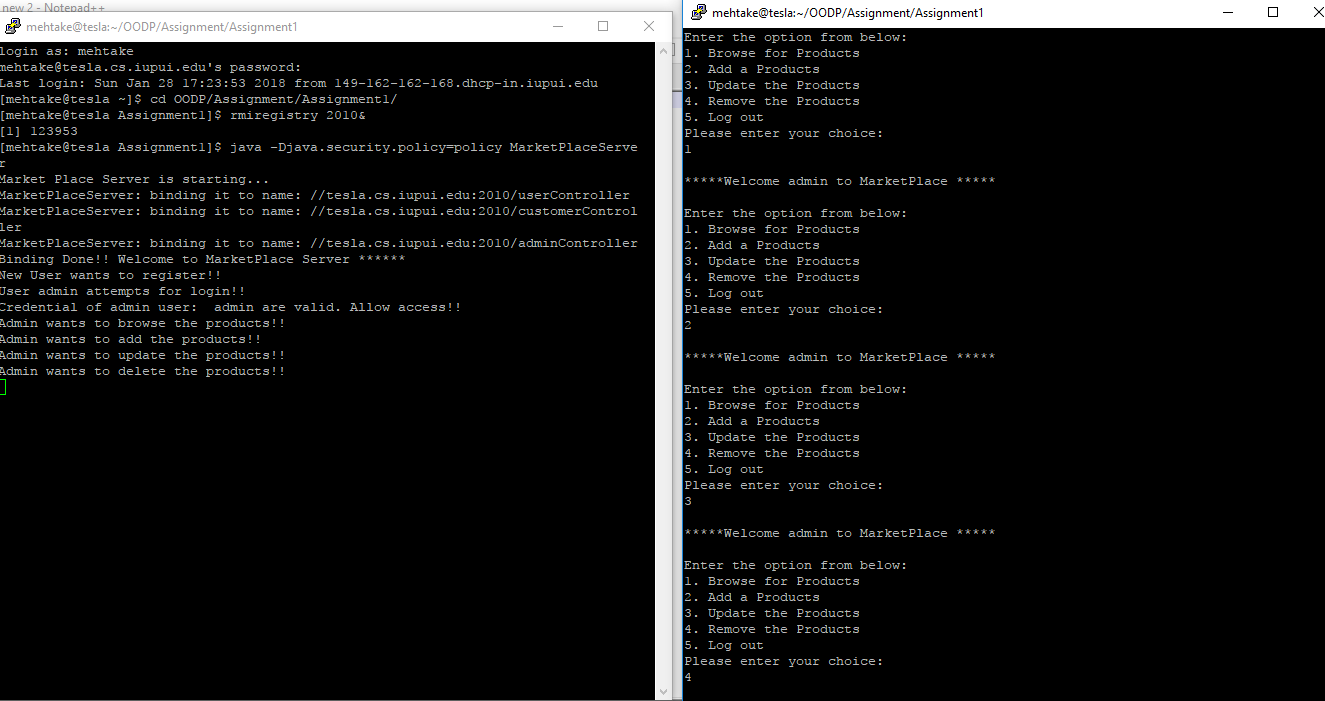
Once Register option is selected, server will get the request. But the complete functionality is not implemented yet.

## **Admin Login**



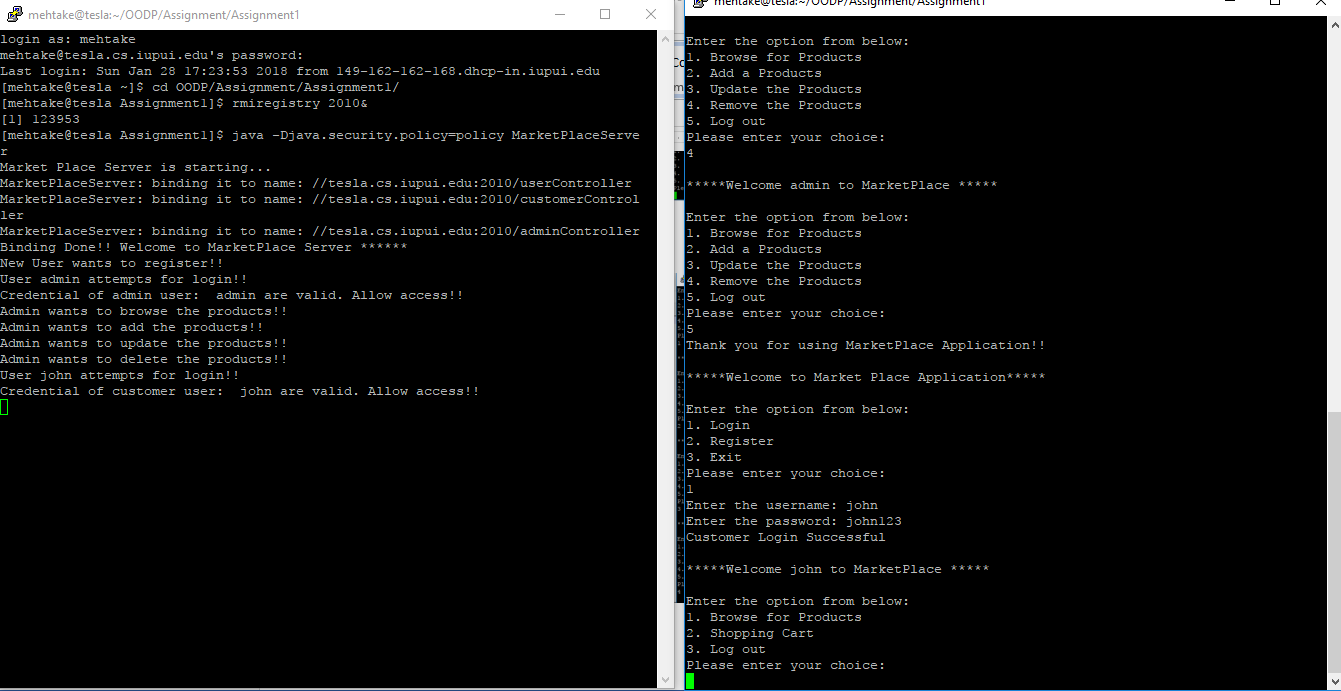
There are 2 kinds of login provided in this application. One is admin and other is Customer user. On entering valid credentials of admin, it displays the admin view. And on server side, will get message showing access.

## **Admin View**



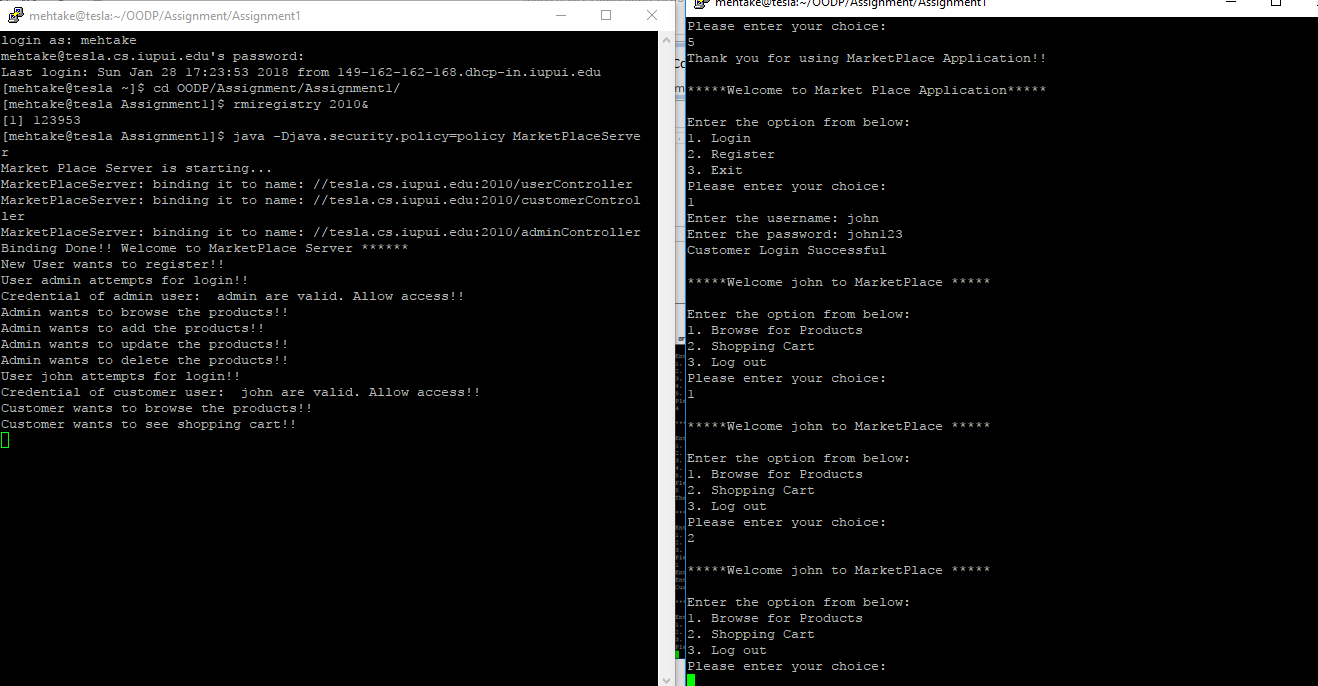
Admin view will display all the options of admin have in MarketPlace Application. Based on the selection, model will display respective message. Functionality is not implemented.

## **Customer Login**



On entering valid credentials of customer, it displays the customer view. And on server side, will get message showing access.

## **Customer View**



Customer view will display all the options customers have in MarketPlace Application. Based on the selection, model will display respective message. Functionality is not implemented.

# **Conclusion**

The agile methodology of the designing and developing the MarketPlace Application will result in good and flawless design. Also this first iteration created the basic structure of the application using MVC architecture and the communication between different models are setup through RMI.

# **References**

1. [website] <https://developer.chrome.com/apps/app_frameworks>
2. [website] <https://www.javatpoint.com/RMI>
3. [website] <https://www.smartdraw.com/class-diagram/>
4. [website] <https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/>