# Project build and Deployment document

## 1. How to generate war file from source code in a build machine.

**Prerequisites:**

1. JDK 1.7 should be installed (Type "java -version" from command prompt to verify).
2. Maven 3.2.5 should be installed (Type "mvn --version" from command prompt to verify).

Steps to build the source code are as follows:

* Extract the source code and pom file from the attached zip file to a local folder. Ex: c:\temp\BankApp
* Open command prompt. Navigate to the folder containing the source code. This folder should contain "src" folder and a "pom.xml" file.
* From command prompt, type "mvn clean install" and press enter.
* Wait till the build process is successful.
* Upon successful build, a new folder "target" will be created inside the folder containing the source code.
* Navigate inside to this "target" folder. You will find "BankApp.war" file which is ready to be deployed.

**Note:** I newly installed Maven 3.2.5 and JDK 1.7 in a new build machine and then followed the above steps to generate the war file and was able to generate the war file successfully.

## 2. How OOPS concept is used in this application.

Following are the OOPS concepts in Java

**a. Abstraction:** Hiding internal details and showing functionality is known as abstraction. Abstraction in java is achieved using "Abstract class" and "Interfaces".I am using an interface "MonthlyPaymentCalculator" in this application. The EMI calculation method "calculateMonthlyPayment" is present inside this interface.

The "MonthlyPaymentCalculatorImpl" class implements this "MonthlyPaymentCalculator" interface and providing the EMI calculation.So internal working details is hidden for the user, thus achieving Abstraction.

**b. Inheritance:** One object acquires all the properties and behaviors of parent object.Inheritance in java is achieved either by extending "class" or "interfaces".

My application has a class "MonthlyPaymentCalculatorImpl" which implements the "MonthlyPaymentCalculator" interface.This "MonthlyPaymentCalculatorImpl" class implements the "calculateMonthlyPayment" and provides the implementation for the "calculateMonthlyPayment" method, thus achieving Inheritance.

**c. Encapsulation**: Binding (or wrapping) code and data together into a single unit is known as encapsulation.POJO's (Plain Old Java Objects) in java is the best examples for encapsulation where properties are always private and their getters and setters are always public.

My application has a business model called "LoanEntities" which has private properties and public methods, thus achieving encapsulation.

**d. Polymorphism:** One interface, multiple methods.Polymorphism is achieved using method overloading(static) and method overriding(dynamic/runtime).

Due to simplicity of the requirement I have not implemented polymorphism in this application.

## 3. How coding standards are met in this application.

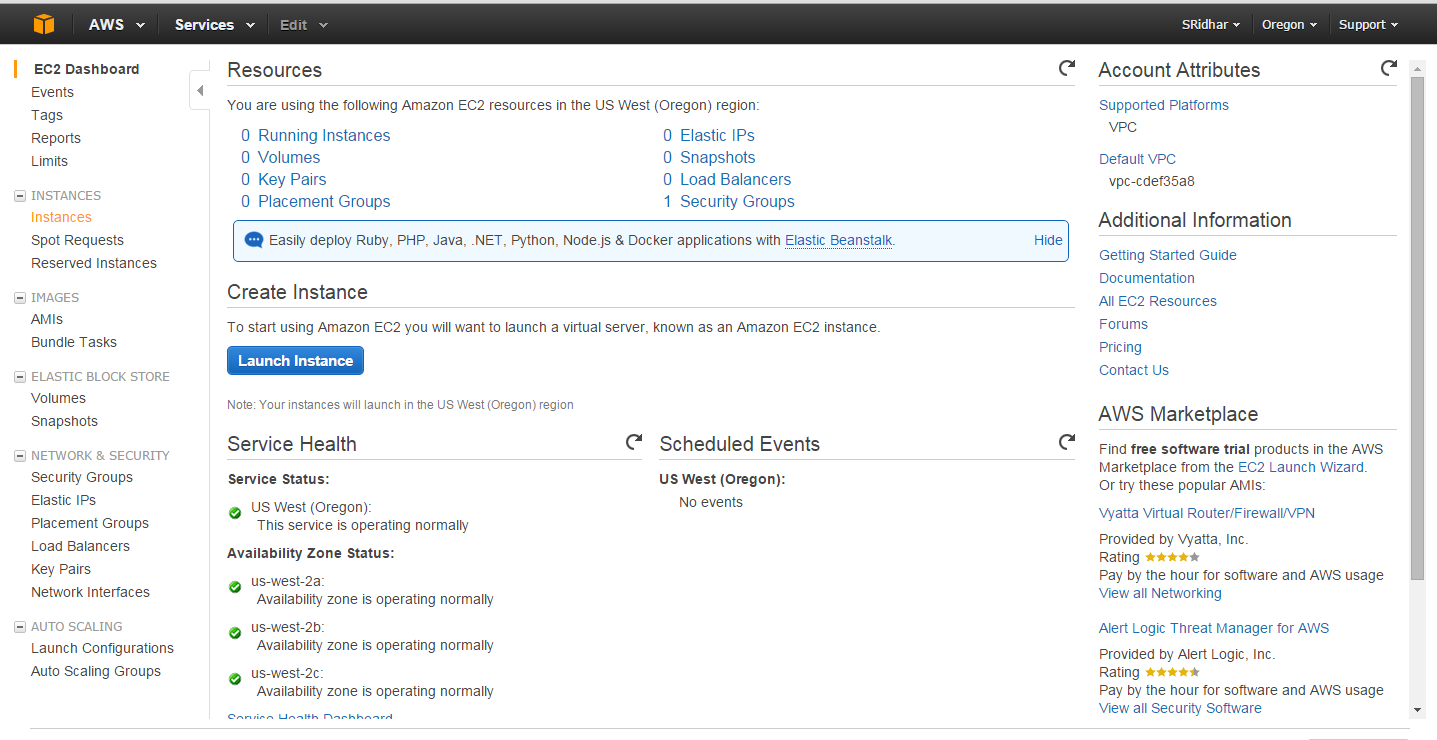
* Added comments about the working details of class, interfaces, methods and variables used in this application.
* Followed naming conventions and camel casing for classes, interfaces and variables.
* Unused code/variables are not present in this application. Every class/variable has its own purpose.
* "Coding against the interface" is done in this application. This allows to change the implementation of EMI calculation method easily.
* Proper package name is provided in this application.

## 4. How to create a Free Amazon EC2 instance.

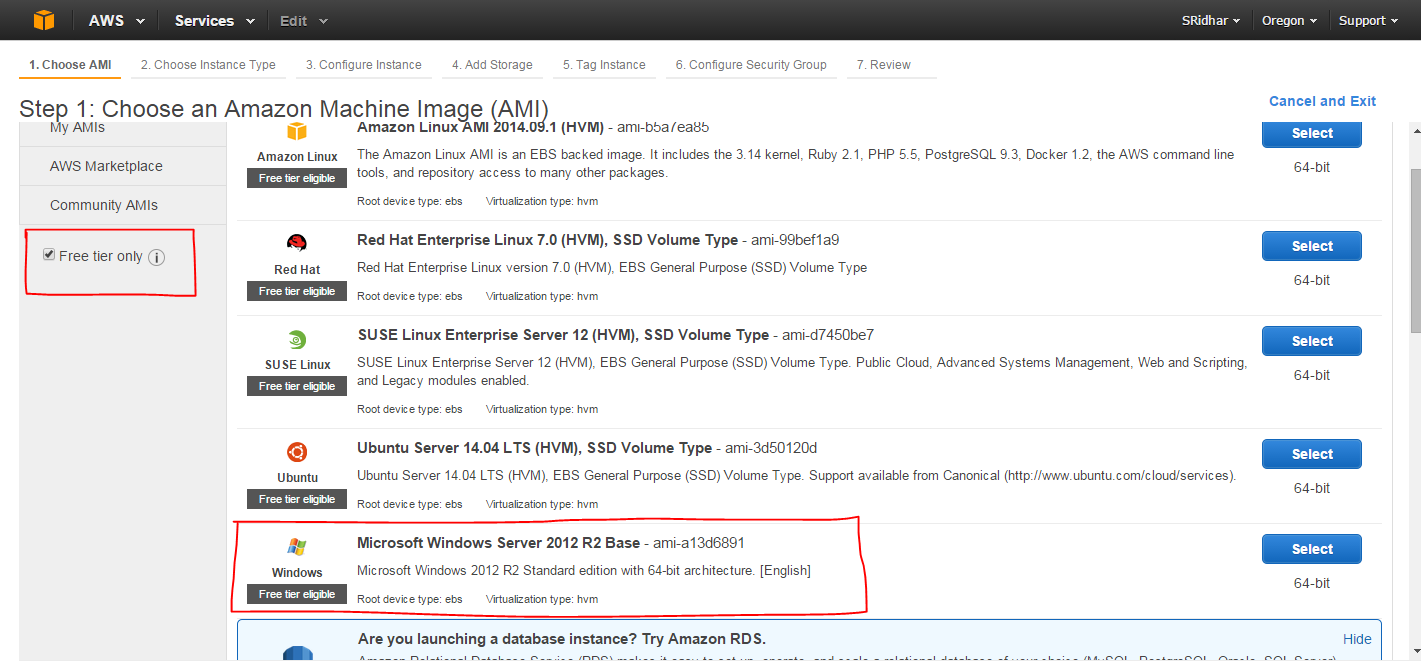
**Note:** Below details are only to create a new windows instance. If you wish to see only the war deployment in amazon instance, please skip to next section.

Also the below details explain about creating and launching a free windows instance (which is free of cost) and not an enterprise instance.

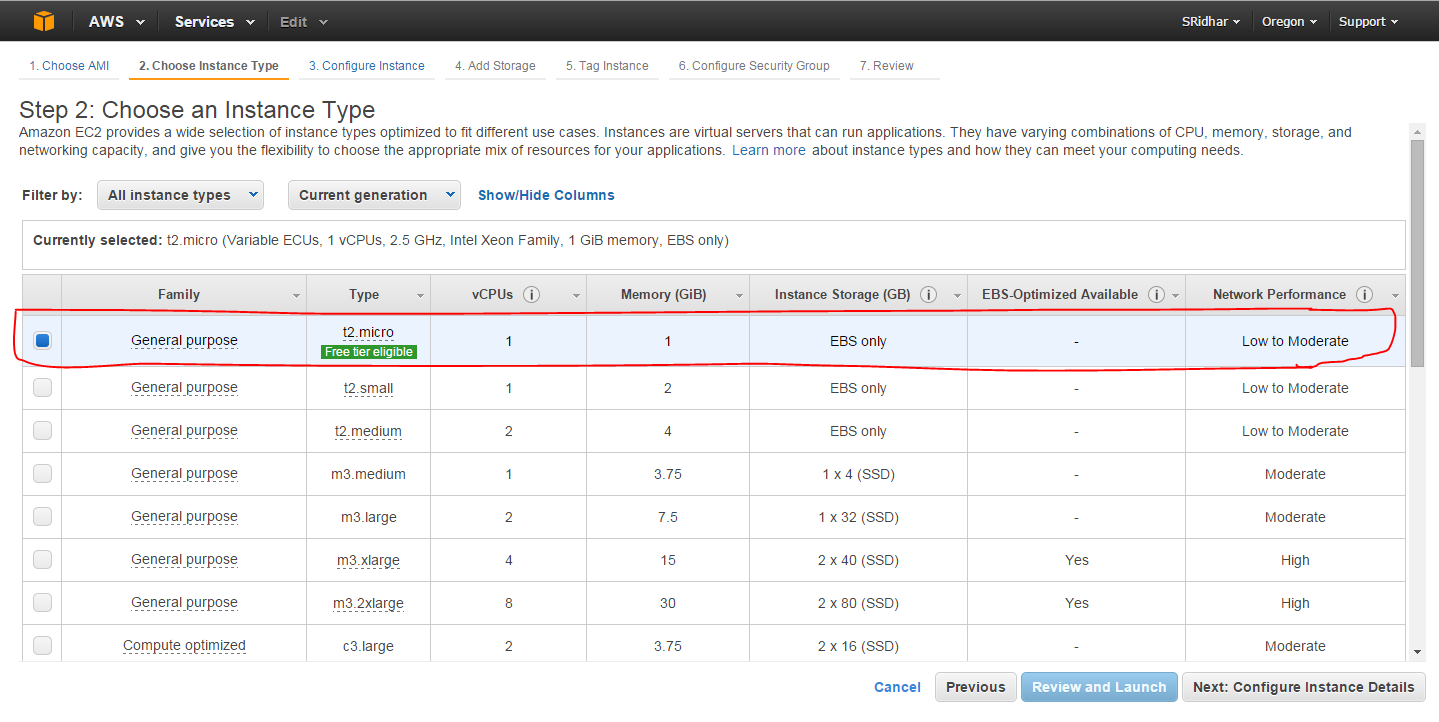
1. Login to the Amazon account and navigate to EC2 dashboard. See below snapshot.



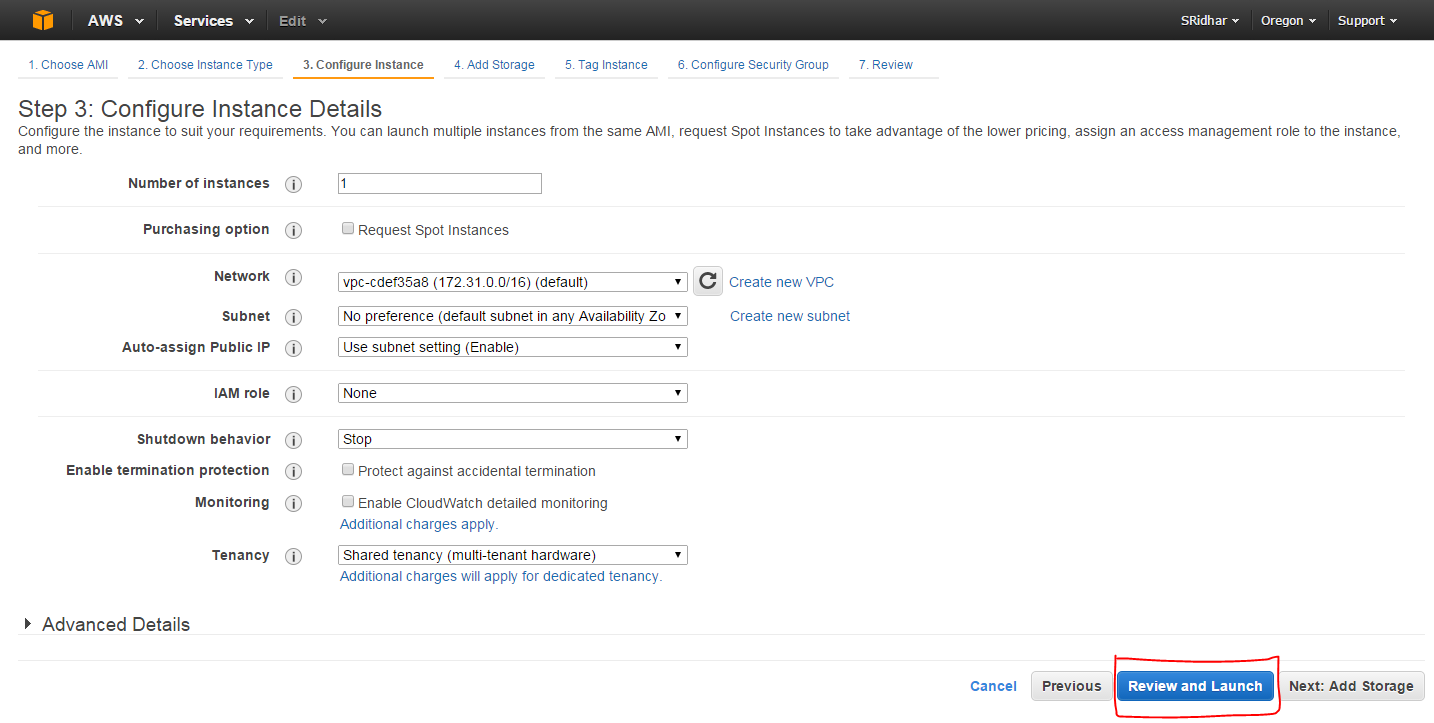
1. Click on Launch Instance button. Click on “Free tier only” option from the dashboard. It will list all the free instances (both windows and linux instances). Select “Microsoft Windows Server 2012 R2 Base” to launch a windows instance. See snapshot below



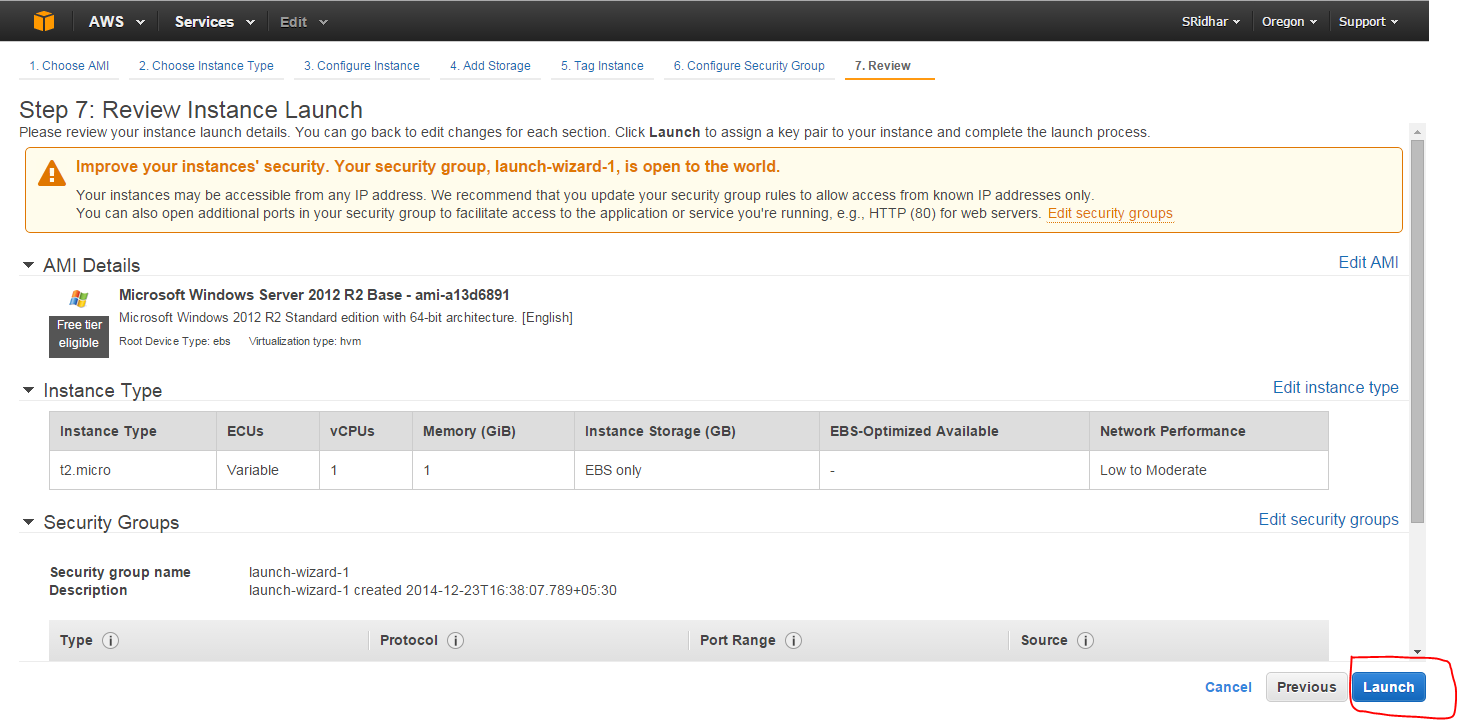
1. Select the free “General purpose” instance. Click on “Next: Configure Instance Details” button. See below snapshot.



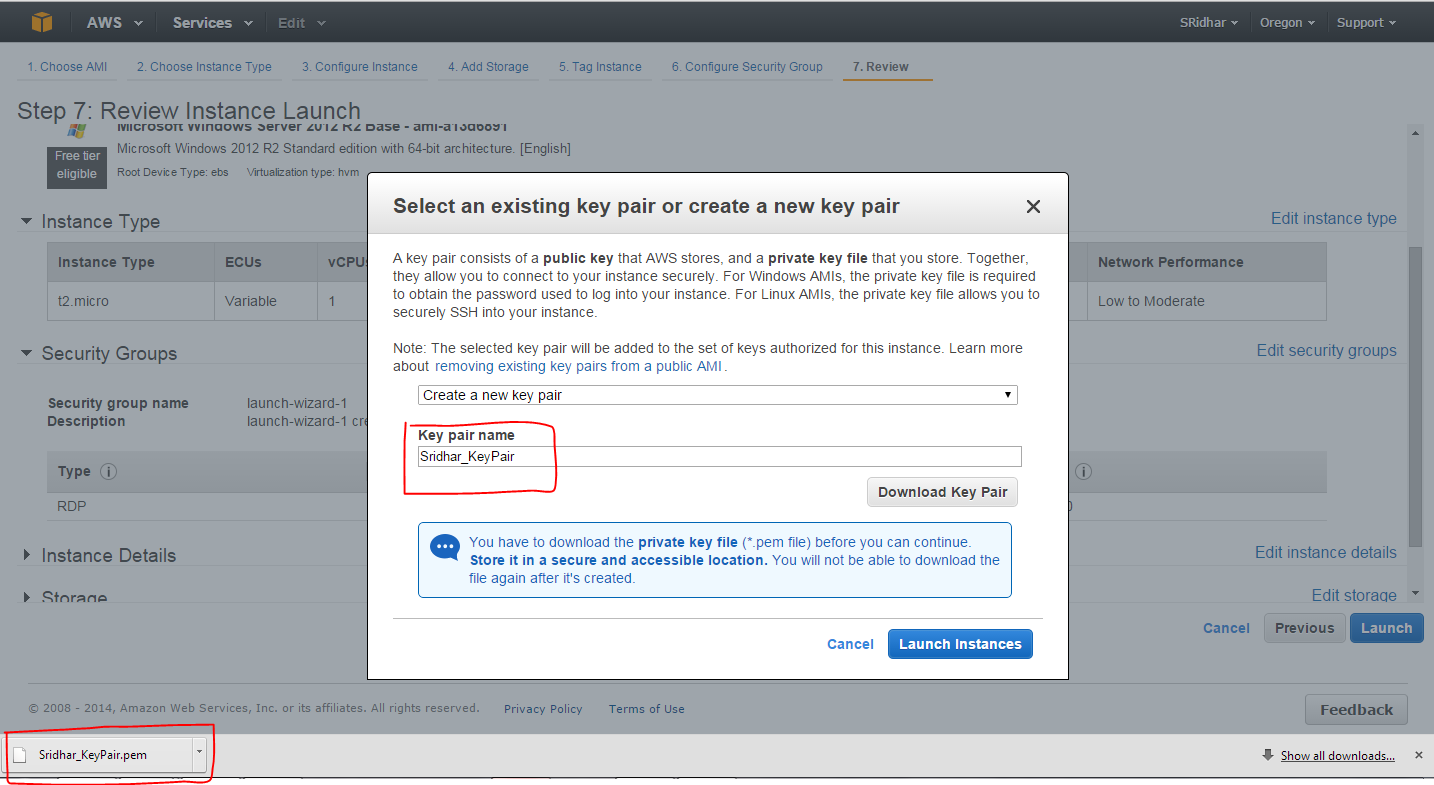
1. Do not change any setting in this below page. Click on “Review and Launch” button. See below snapshot.



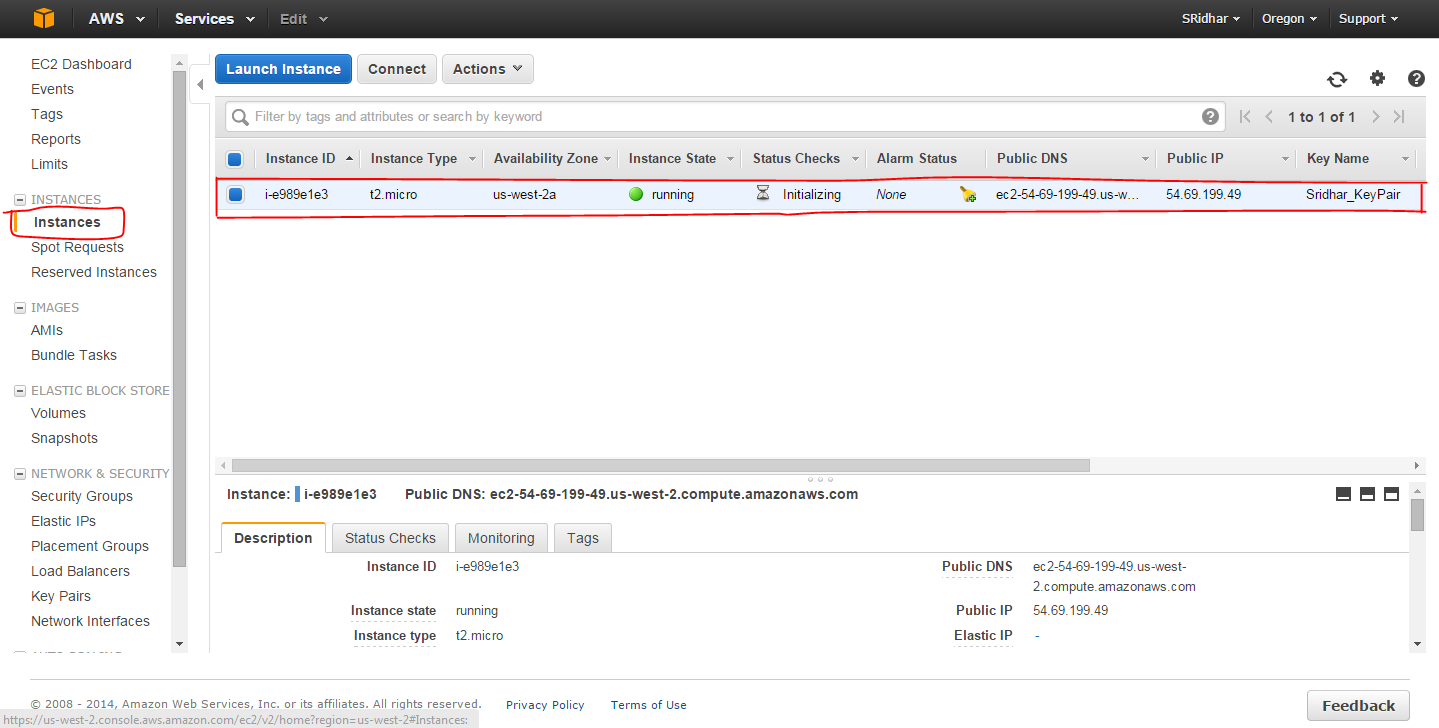
1. Click on launch button. See below snapshot.



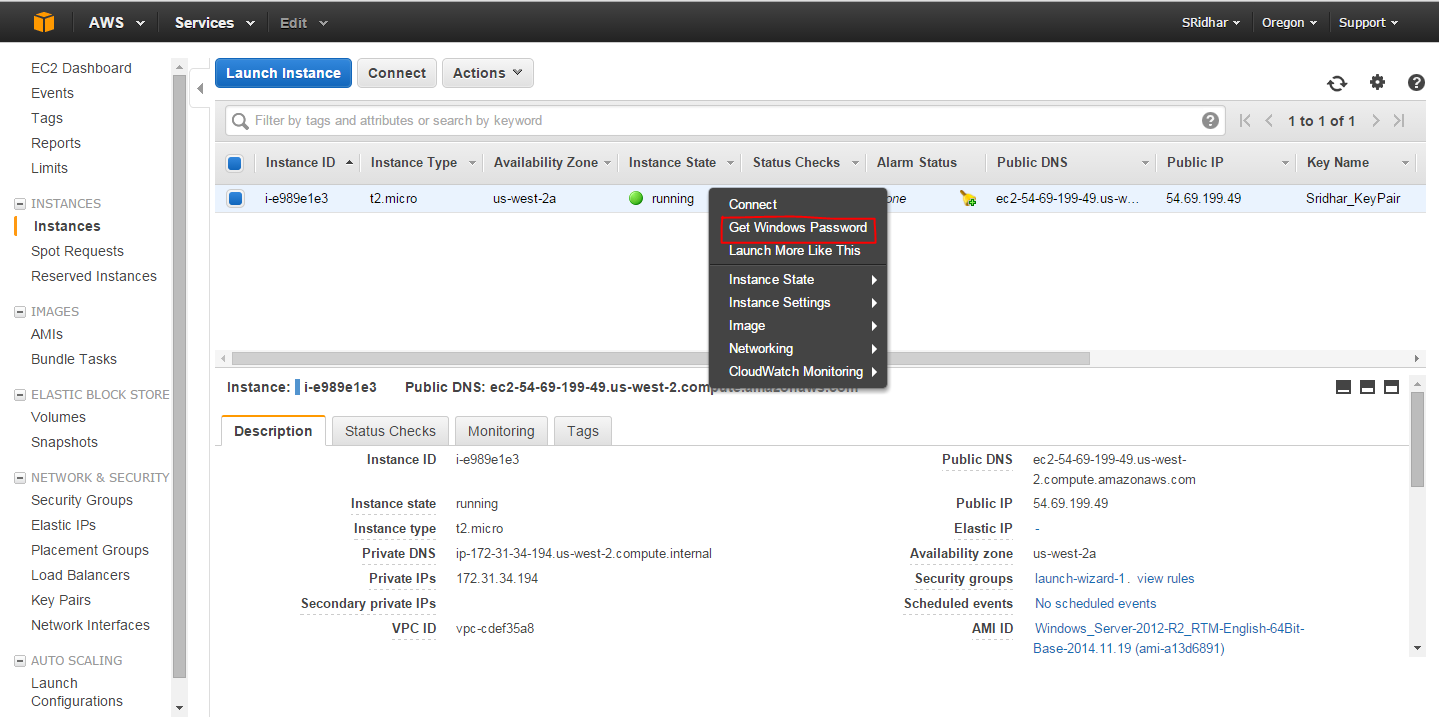
1. You will be asked to enter a key pair. As you are a new user, create a new key pair, give a name and click “download key pair” button.Key pair will be downloaded. Now click on “launch” button to launch the instance. See below snapshot.



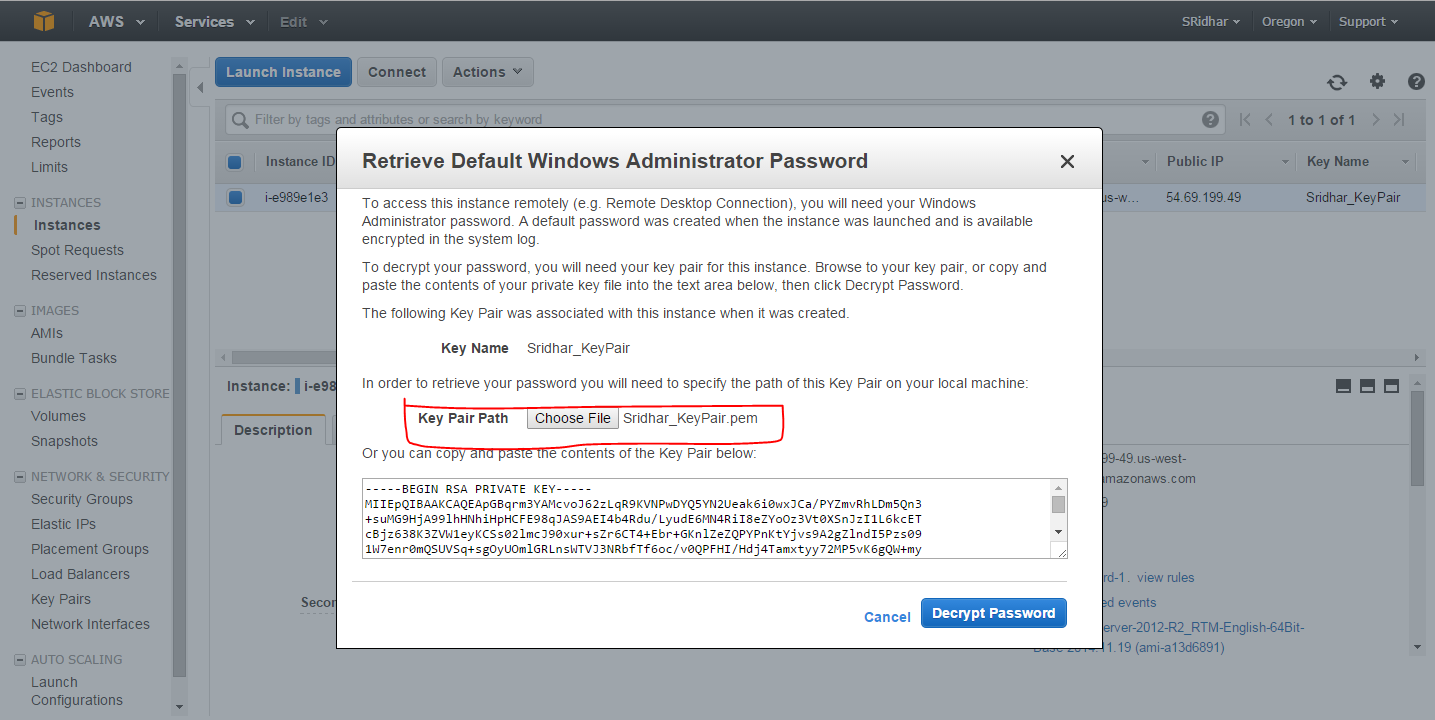
1. Click on the launched instance. You will be navigated to EC2 dashboard. Click on “Instances” tab. Following sceen will show the details of the launched instance.



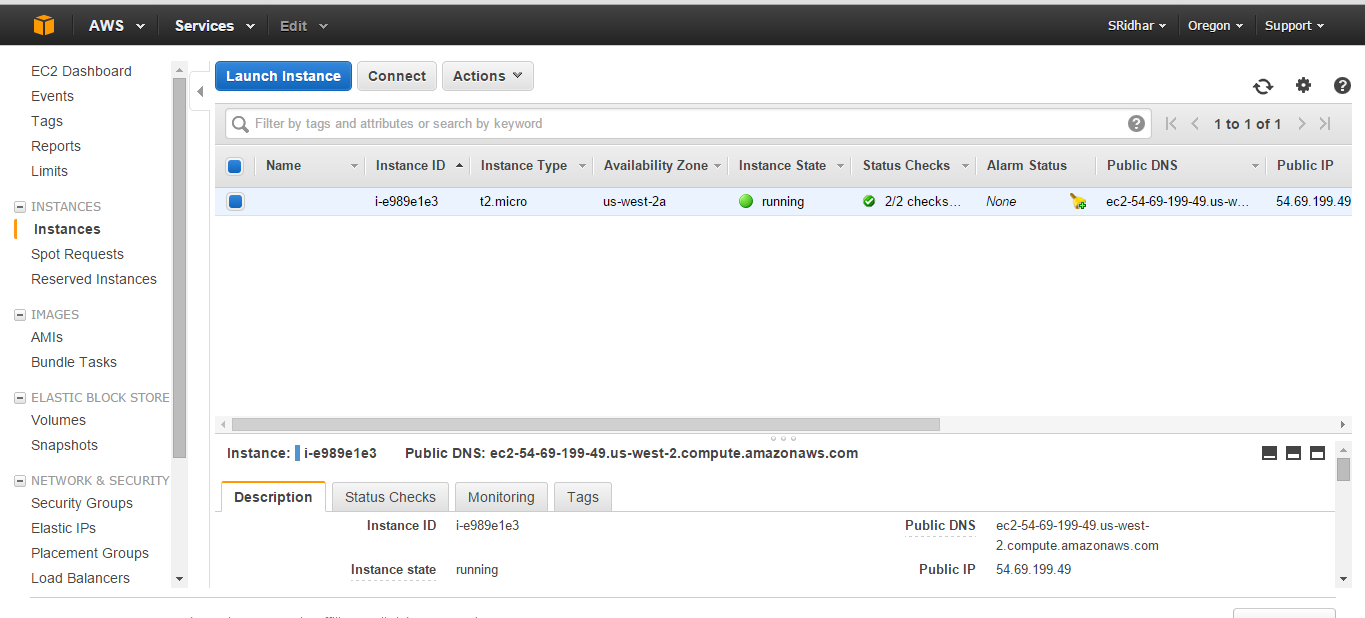
1. To get the Remote Desktop Connection password, right click on the running instance and select “Get Windows Password”. See below snapshot.



1. You will be asked to give the path of the previously downloaded pem key. After giving pem key path, click on “Decrypt Password” button. See below snapshot.



1. Once password is generated you will be able to login to this instance using remote desktop connection.



## 5. How to deploy war file in Amazon instances.

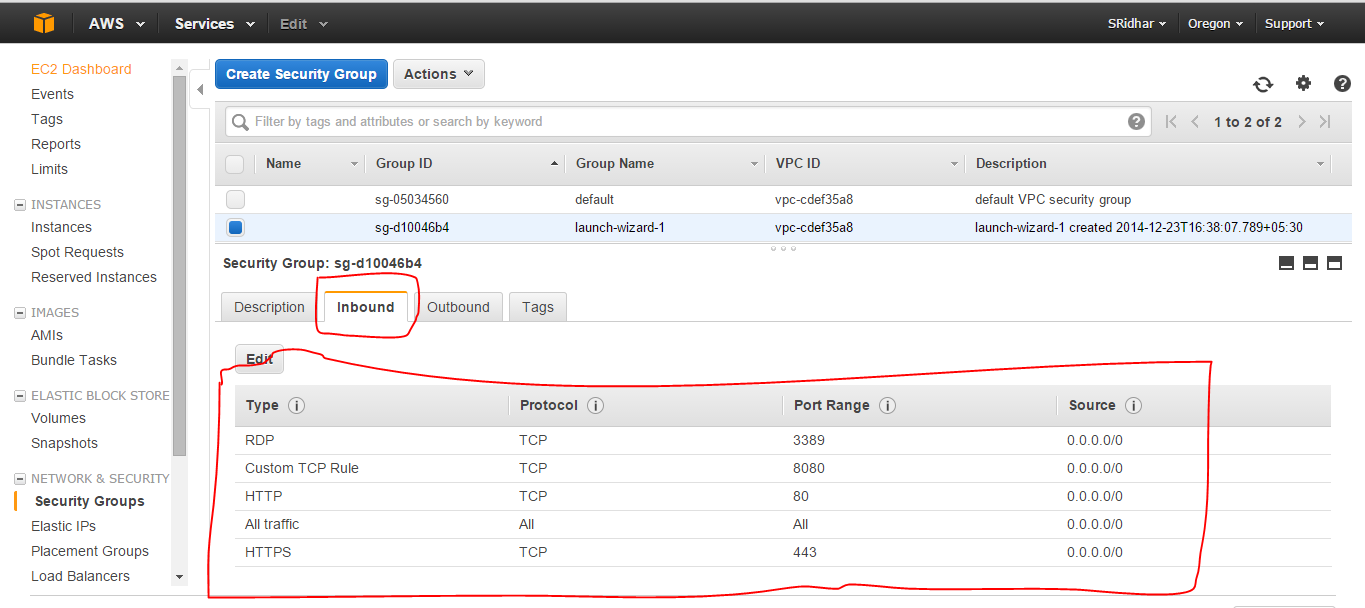
**Prerequisites:**

* To access any application deployed inside an EC2 instances, the windows firewall of the EC2 instance should be configured to receive requests from any external IP and through particular port.

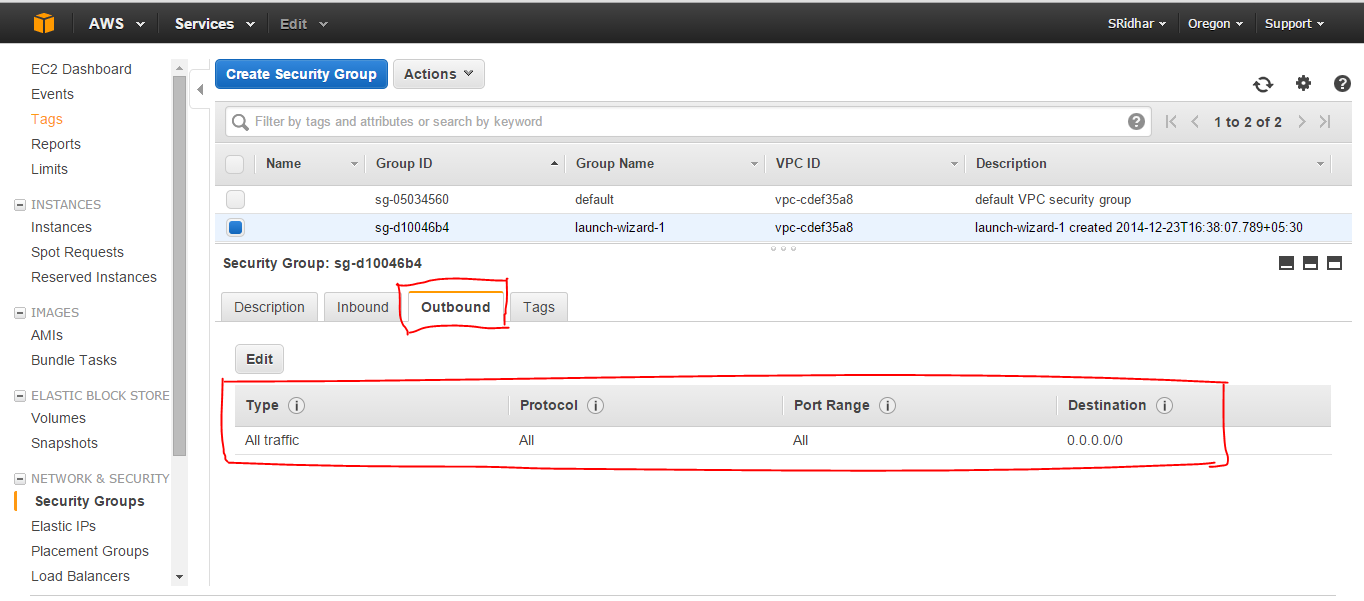
Ex: port 8080 should be opened to receive requests from users.

* Also from the EC2 console, the security rules (inbound and outbound) request IP’s and ports should be configured. Or else users cannot access the application deployed in EC2. See the below snapshots.

The below snapshot shows “inbound” rule of an EC2.



The below snapshot shows “outbound” rule of an EC2.



Once these things are done, we can proceed to deploy our war file.

Steps to deploy war file in a newly created Amazon instance.

1. Log in to this windows instance.
2. Download and install JDK 1.7 from <http://www.oracle.com/technetwork/java/javase/downloads/index.html> and add the JAVA\_HOME “ex: C:\Program Files (x86)\Java\jdk1.7.0\_01\bin” location and add the JRE\_HOME location “ex: C:\Program Files (x86)\Java\jre7\bin” to the environmental variables.
3. Download the apache tomcat 7 from <http://tomcat.apache.org/download-70.cgi>. Extract the zip file.
4. Copy the war file generated war file into extracted “apache-tomcat-7.0.42\webapps” folder using the details from section 1. Navigate to “apache-tomcat-7.0.42\bin” folder and double click the “startup.bat” file of tomcat.
5. Open browser and give the following URL : <http://localhost:8080/EMICalculatingApp/loandetailspage> to display the loan details page.

The following application can also be accessed from this below URL. This application is hosted by me in a windows EC2 instance.

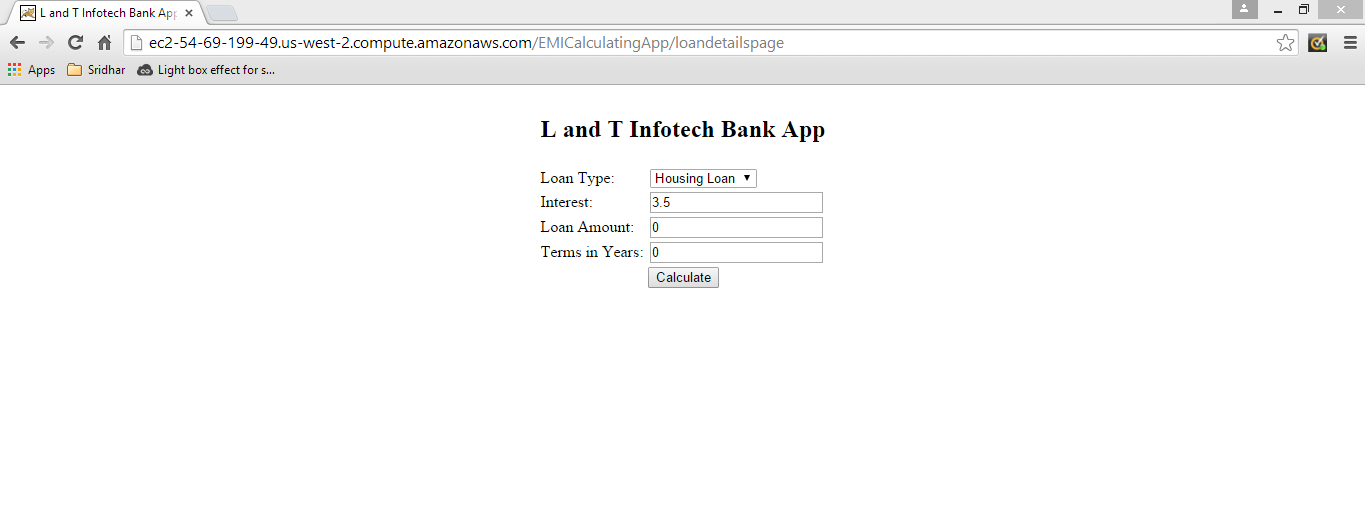
<http://ec2-54-69-199-49.us-west-2.compute.amazonaws.com/EMICalculatingApp/loandetailspage>

## 6. Snapshots from the application

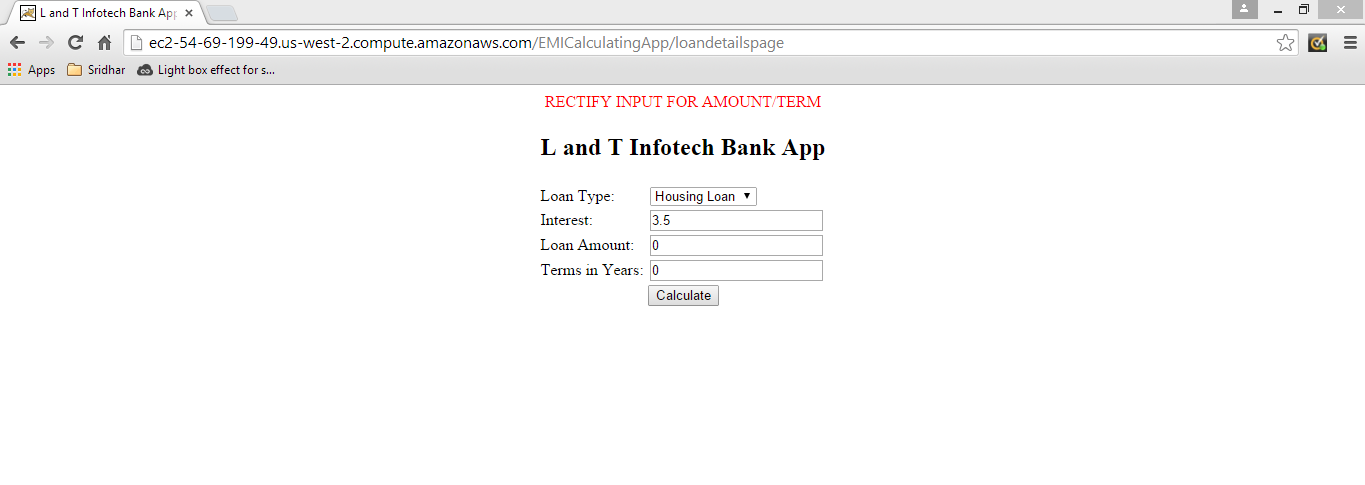
Note: For sake of simplicity I have given loan amount range (minimum 1rs and maximum 10lakhs) and loan term a minimum of 1 year and maximum of 10 years.

The loan amount only accepts integer (1 - 100000) as input. Inputs like (1000.00, 25000.567, 100000.00 etc) will be considered as wrong input for this application. Also the loan term is integer (1 - 10). Inputs like (1.5, 5.5, 6.9, 7.5 etc) is considered as wrong input for this application.

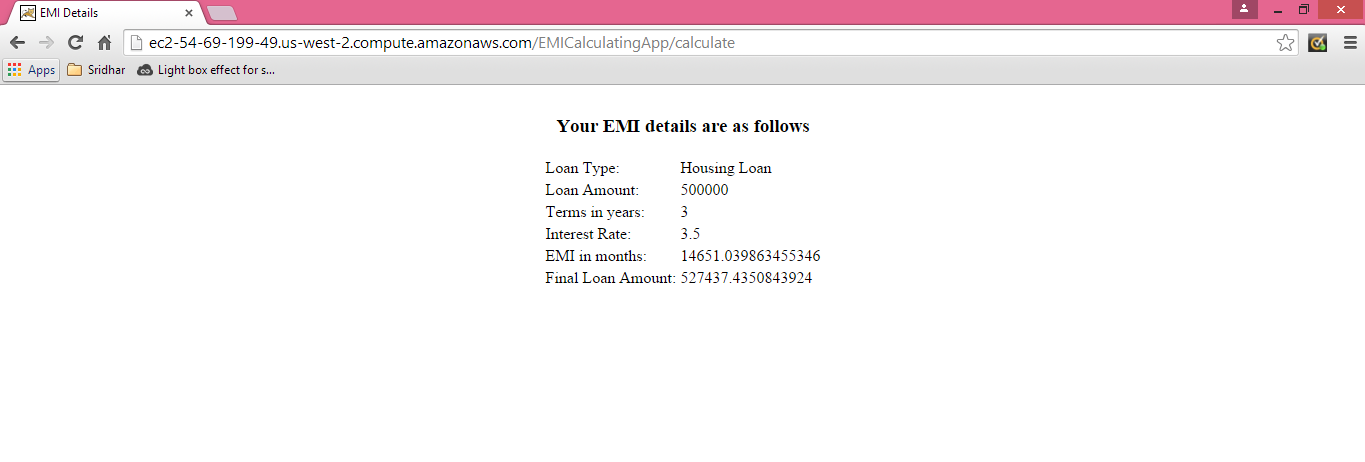
Also basic validations like, without entering any loan amount or loan terms and clicking on calculate button, only providing loan amount and not providing loan term, only providing loan term and not providing loan amount is validated using spring validations.

Screen showing main page.

Screen showing invalid input error message.



Screen showing EMI calculation details.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*The End\*\*\*\*\*\*\*\*\*\*\*\*