

CS268

Supporting Multi Tenants on a Cloud Platform

Final Report

Ravee Khandagale
Sahil Motadoo

Advisor: Dr. Melody Moh

Topics

Objective

Research on Technologies

High Level description on implementation

Explanation with screenshots on working product

Objective

- Understand the concept of multitenancy - Part 1
- Bring up a web service using Apache Tomcat - Part 1
- Implementing RBAC using Tomcat web app. (Java) - Part 2
- Support user login to create a project. (has a name and description) - Part 2
- Store the project in a persistent database. - Part 2
- Ensure project created by different users cannot be seen by other users by default. - Part 2
- Support a special role that can create project and invite others to the project - Part 2
- Extend and enhance RBAC implemented in Part 2 to support a large amount of tenants. - Part 3

Research on Technologies

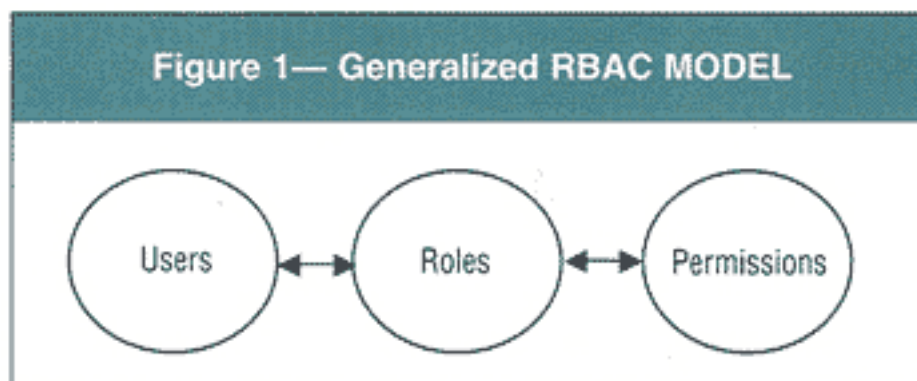
Multitenancy

- Multitenancy helps tenants to use a common application workflow at the same time.
- Tenants have their own personal instance of the application that they use or deploy
- They ideally should not have knowledge or access to other tenants and their instances of the application
- This includes the data and configuration details about the instances of the other tenant's application
- Tenants can modify the following attributes:
 - UI
 - Workflow
 - Data Schema
 - Access Privileges
- Characteristics
 - The behaviour of one tenant does not affect the other's → Security
 - Scalability
 - Pay per usage
 - Tenants can have individual databases and/or schemas → Recovery

Apache Tomcat

- Apache tomcat is an open source software which is used as an application server.
- It is a platform which is used to host Java based JSP/HTML pages as well as fully-developed web application in the form of a WAR file (Web Archive).
- Tomcat hosts web applications in a container called Catalina which is present in tomcat install location.
- Tomcat also has a GUI which can be accessed using "http://localhost:8080/". Using this GUI we can deploy WAR files and also specify the deployment descriptors as well as the necessary context path.

RBAC



- Traditionally, it means assigning resources based on the role in a particular organization
- Similarly, on a multitenancy platform, it means assigning resources to a particular tenant based on the role that is defined to him/her.

- First, we need to study what resources should be accessed by what group of people/tenants.
- Accordingly, the roles that are defined are given a certain set of resources.
- This means that those set of users which have a particular role are given a particular set of resources
- For example, say, there are two applications- Airbnb and Tripping. There are many user accounts associated to these two applications.
- The users of Airbnb should not have access to the resources in Tripping.
- Similarly, the users of Tripping should not be able to access the resources of Airbnb.
- Moreover, each user has various roles such as *renter*, *home-host*, *admin* etc.
- A renter cannot have write permissions for the house listings given in a particular area, only the home-host has that permission.
- This shows that each user is given a role and each of those roles has permissions associated with it.
- Role based access helps in doing so.
- Moreover, it provides a strict couple to access given for each role
- Thereby, access is clean to implement.
- Now, one way to do provide clean access is to provide a list of users to each resource and whenever a query is fired to access that resource, check in the control list of that resource to see if the user who is firing the query has access to that resource.
- Now, this will work fine for a limited number of users and a good amount of resources in comparison to the number of resources.
- But, in a system where the number of users is much greater than the number of resources, it would be wise to define resources for a particular user.
- To optimize this more, we can define roles for each user and give access to a role instead of a user.
- We can do this by using a lot of technologies.
- This has various implementation ideas based on the application that you have or based on the kind of database design that you already have.
- If an application has to be designed from scratch, this functionality can be kept in mind while designing the database.

Apache Tomcat- Realm Configuration

- A database of users who have authority along with their passwords in a web application/s.
- There are also various roles that are assigned to each user.
- A particular user can have many roles associated to them, but for simplicity, we have defined only one role for each user.
- Each role will then have access to the list of resources which are mentioned in the database.
- The high level description on implementation will describe realm configuration.

- On a high level, this gives tomcat access to interact with the database and authenticate the users based on their role and give them access based to the various resources in their system.

Part 3:

Scalability in RBAC Applications/Systems

There are two ways in which an application or system can be scalable:

Scale-up:

- This is also called as vertical scaling.
- It involves faster CPU, more RAM, more disk space
- Here, one has to test the limit of the application in terms of how many resources or users are being added in the system.
- Adding more users or resources is going to stress your system more and the application is going to scale up only till a limit due to the limited number of resources.
- For example if you scale up by putting more users, at some point you're going to reach the hard disk limits.

Scale-out:

- This is also called horizontal scaling.
- It involves adding more cores CPUs and servers
- Data can be parallelly processed and it might be distributed physically
- The communication between the nodes should be limited.
- Actually, communication in nodes means sabotaging some parts of ACID
- One cannot have entire ACID and scale up
- NoSQL Solutions are the popular solutions
- The key-value are scalable
- However, you can query only on primary key.

-

High Level Description on Implementation

Part 1

Launch server

In order to run our application, we need to use a stable version of Tomcat. We use Tomcat 7 for our purposes.

1. Download the .tar file for tomcat version 7 using either manual download or wget command
2. Extract the contents to /usr/local/tomcat7
3. Go to the bin folder in tomcat7
4. To start the server execute command: ./startup.sh
5. Tomcat will now start up and listen on default port 8080
6. To stop the server execute command: ./shutdown.sh

Deploy Web App

For the sake of setting up the tomcat server, we have deployed a sample web application consisting of a login page

1. Copy the web pages to an arbitrary path. In our case, we copied the necessary HTML files to /usr/share/tomcat7-myapp/myapp/
2. Now go to the location where tomcat is installed. Go specifically to /conf/Catalina/localhost/
3. Create an XML file, myapp.xml. This XML file is the deployment descriptor which is read by the tomcat server. Tomcat can then render the HTML content.
4. Write the following in myapp.xml
<Context path="/myapp" docBase="/usr/share/tomcat7-myapp/myapp"/>
5. Now shutdown (using ./shutdown.sh) or restart tomcat for the changes to take place
6. Go to "http://localhost:8080/myapp/" to access the webpage.

Note: For the actual application we would be deploying a WAR(Web Archive) in a similar fashion

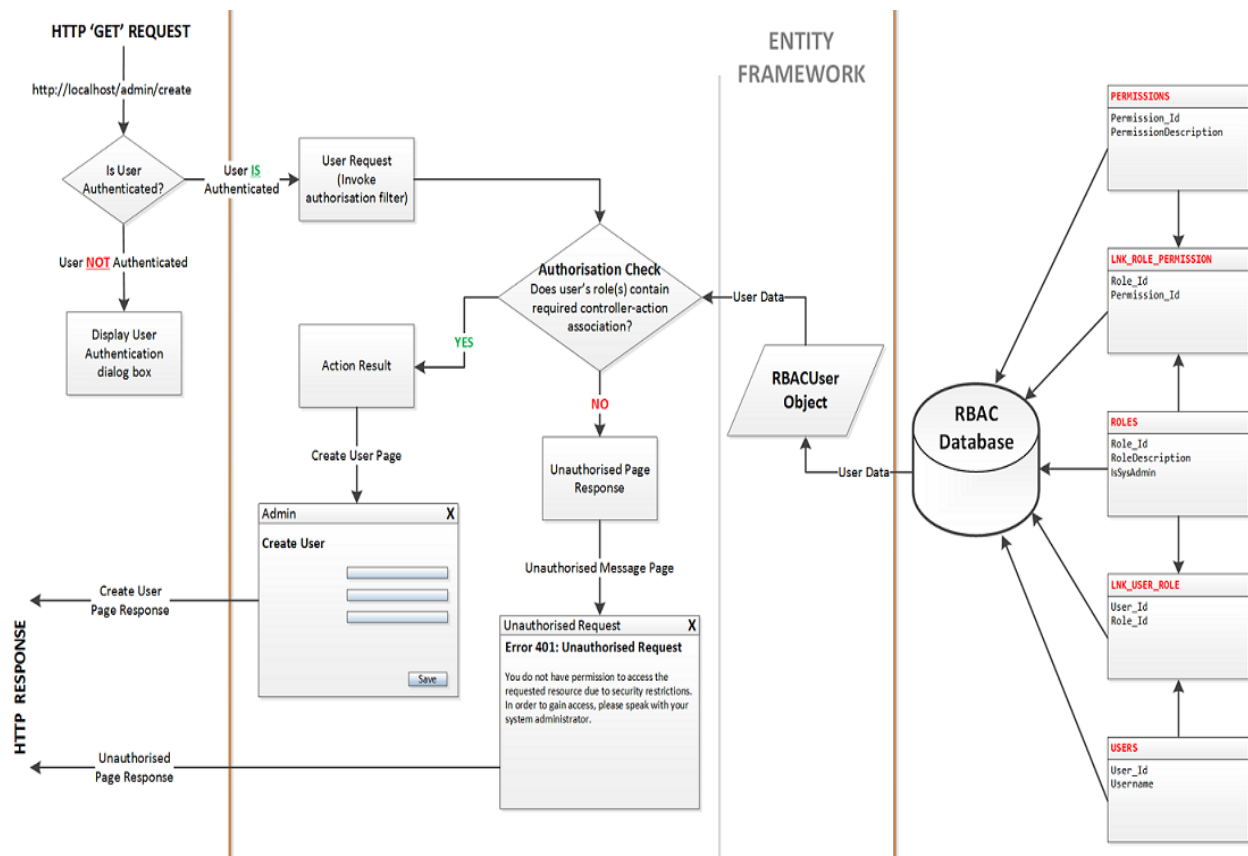
Part 2

- We first set up the realm configurations in Apache tomcat
- We then Created Tables in Mysql database (which is a persistent database)
 - 1)Users
This has username and password associated with a user account
 - 2)Projects
This has the project name and the project description associated with it
 - 3)Roles
This has the rolename and the project name (as a foreign key) that the role can access in the system
 - 4)Users_Roles
This has the username and the role assigned to that user(rolename is foreign key here)
- We then create a login page- login.jsp
- This page will have the form from which the user can enter the system.

- He/she has to provide username and password details and the realm definition authenticates the user using form based authentication.
- Then the user is directed to a page in which he can create his own project
- This user is the owner of the project the user creates

Part 3:

- For making the application more scalable:
- We add a functionality called “Add user”
- The admin (in our case, the role is manager) has full access to all the application’s resources (that is, projects)
- The manager clicks “Add User” to add more users (tenants) in the system.
- Addition of a lot of demonstrates scalability.
- Compared to a cloud environment, this is a way to assign tenants to a system by abstracting the details of the underlying infrastructure.



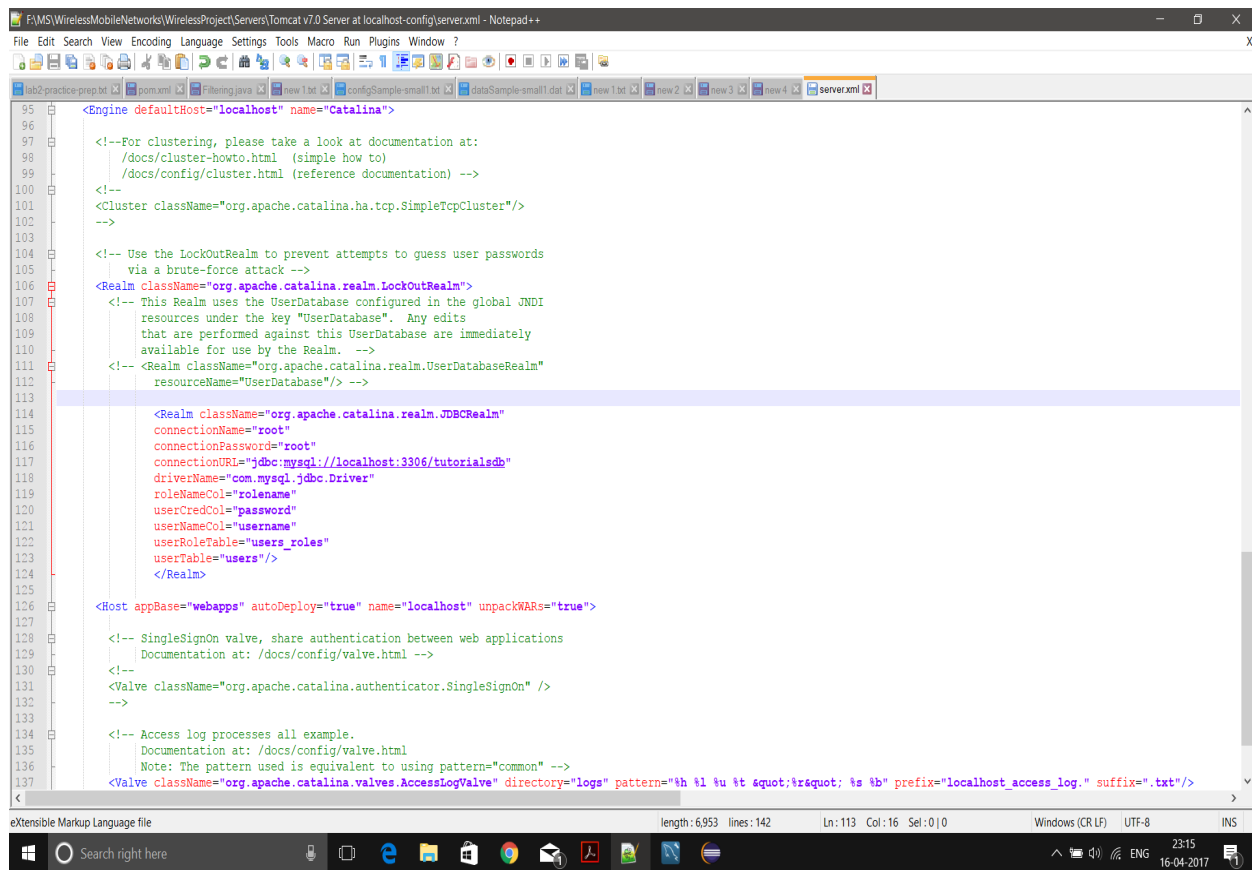
Explanation with screenshots of Working Product

Steps 1-10 explain the process of launching a tomcat server.

1. We downloaded Apache Tomcat from the official download page of Apache Tomcat:
<https://tomcat.apache.org/download-70.cgi>
2. We then unzipped and then started tomcat service
3. The following screenshot shows how we started the service
4. We then defined the admin role in tomcat-users.xml to log in to the tomcat server as shown in the below screenshot.
5. The following screenshot shows the details of the listening port (port 80)
6. The following is the GUI which can be used by users of different roles like admin, user etc.
7. After an admin login, the following page is displayed:
8. You can deploy a WAR (Web Archive) file and can specify a context path to the WAR file
9. It can also be used to specify the deployment descriptor which is read by tomcat in order to render the web pages
10. We created a sample web page login to demonstrate the use of Tomcat server It goes Login->Submit->Success

Part 2

The realm configuration for tomcat server.xml file is done as following:



```
<!--For clustering, please take a look at documentation at:
/docs/cluster-howto.html (simple how to)
/docs/config/cluster.html (reference documentation) -->
<!--
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
-->

<!-- Use the LockOutRealm to prevent attempts to guess user passwords
via a brute-force attack -->
<Realm className="org.apache.catalina.realm.LockOutRealm">
  <!-- This Realm uses the UserDatabase configured in the global JNDI
resources under the key "UserDatabase". Any edits
that are performed against this UserDatabase are immediately
available for use by the Realm. -->
  <!-- <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/> -->

  <Realm className="org.apache.catalina.realm.JDBCRealm"
    connectionName="root"
    connectionPassword="root"
    connectionURL="jdbc:mysql://localhost:3306/tutorialsdB"
    driverName="com.mysql.jdbc.Driver"
    roleNameCol="rolename"
    userCredCol="password"
    userNameCol="username"
    userRoleTable="users_roles"
    userTable="users"/>
</Realm>

<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">

  <!-- SingleSignOn valve, share authentication between web applications
Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
Documentation at: /docs/config/valve.html
Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" pattern="%h %l %u %t &quot;%s %b&quot; prefix=localhost_access_log. suffix=.txt"/>
</Host>
```

- The realm configuration uses JDBC to connect to the database.
- Therefore, there needs no new connection to the database and this feature can be plugged in as is using JDBC
- The realm configuration also ensures access to the database according to the role that you have in the system.
- We have to mention the column and table names that we are using in our database in the realm configuration.
- Then accordingly, we edit this file- server.xml to include the authentications that we want in order to give access

- After setting up, the realm configuration, we created and hosted simple web pages which will demonstrate the role based access in the system.
- We have created web pages using jsp and hosted them on tomcat.
- Initially, there is a login page.

The login page has two buttons-

1)Login Button- The user can log in using his credentials.

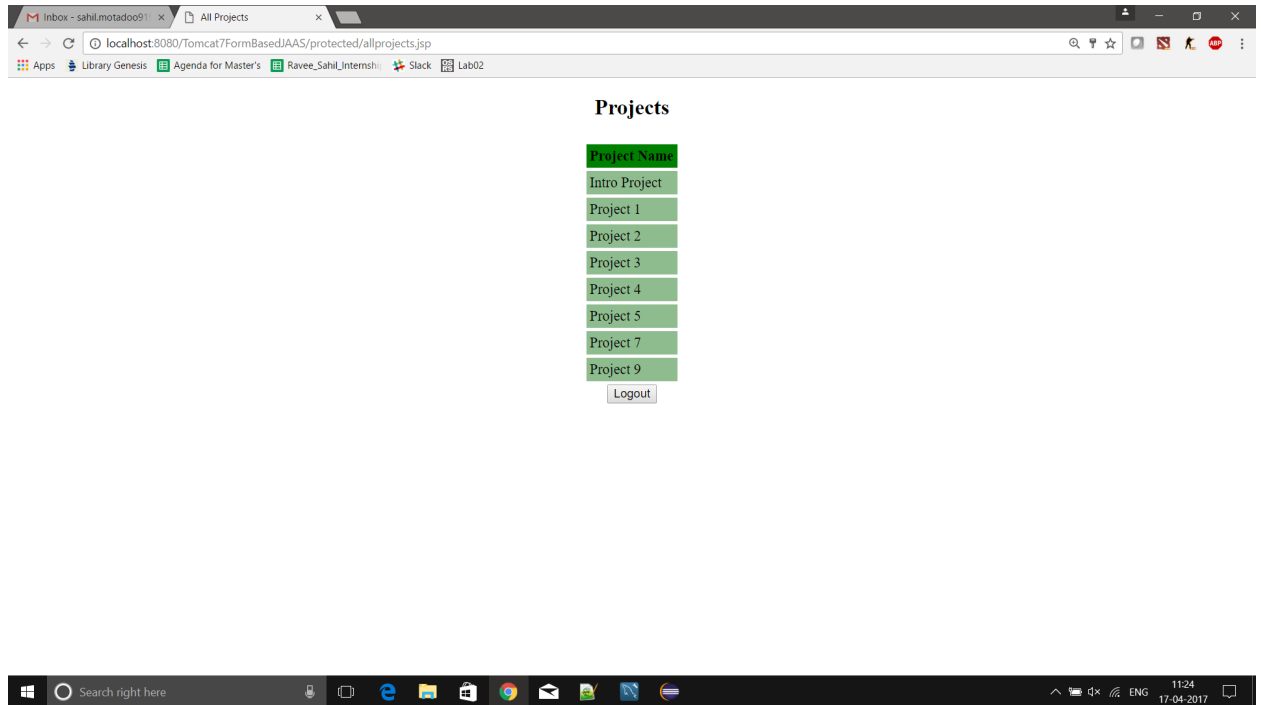
After clicking this, you will be directed to your own page which displays your information and the projects you are authorized to see

2)Display Projects Button

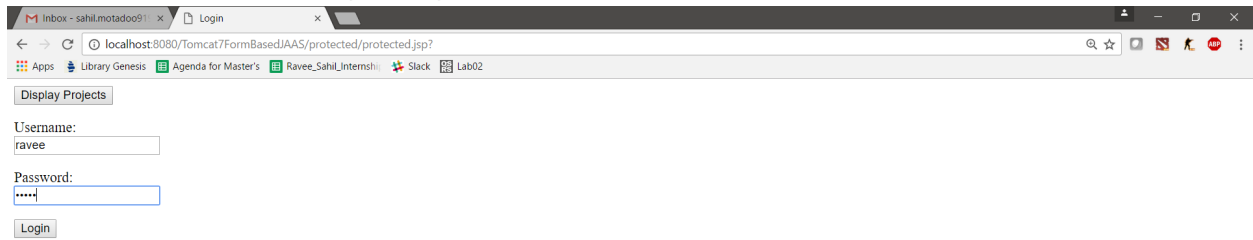
- Only the users of the system can see the Projects name and nothing more than the project name.
- All users in the system can see the project name by default, that is, all users have access to see the project name.
- Role based access will be given to view the project description in addition to seeing the project names.

After clicking this, you will be authorized to see all the project names (not the description).

Screenshot of all Projects

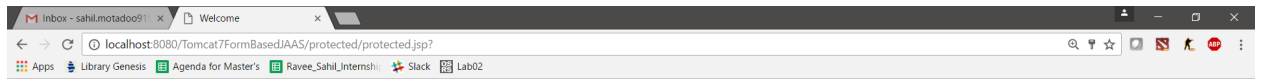


Screenshot of Login Page



- After logging in, you will see a list of **only those projects that you own** along with their descriptions.
- You will see a message saying that the you are authenticated to your role and your role is displayed
- There is also a feature of creating a project that you want to own. That details of that project can only be seen by you.

Screenshot of user details after Log in for all roles



Hello ravee,

[Logout](#)

Authentication Successful.

Role: memberlv11

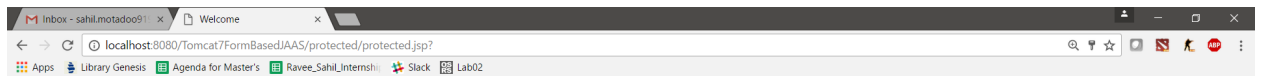
Projects Owned

Project Name	Project Description
Project 1	abc
Project 2	xyz
Project 3	atyu
Project 4	yu
Project 5	Creating multitenants

Create Project:

Project Name:

Project Description:



Hello sahil,

[Logout](#)

Authentication Successful.

Role: manager

Projects Owned

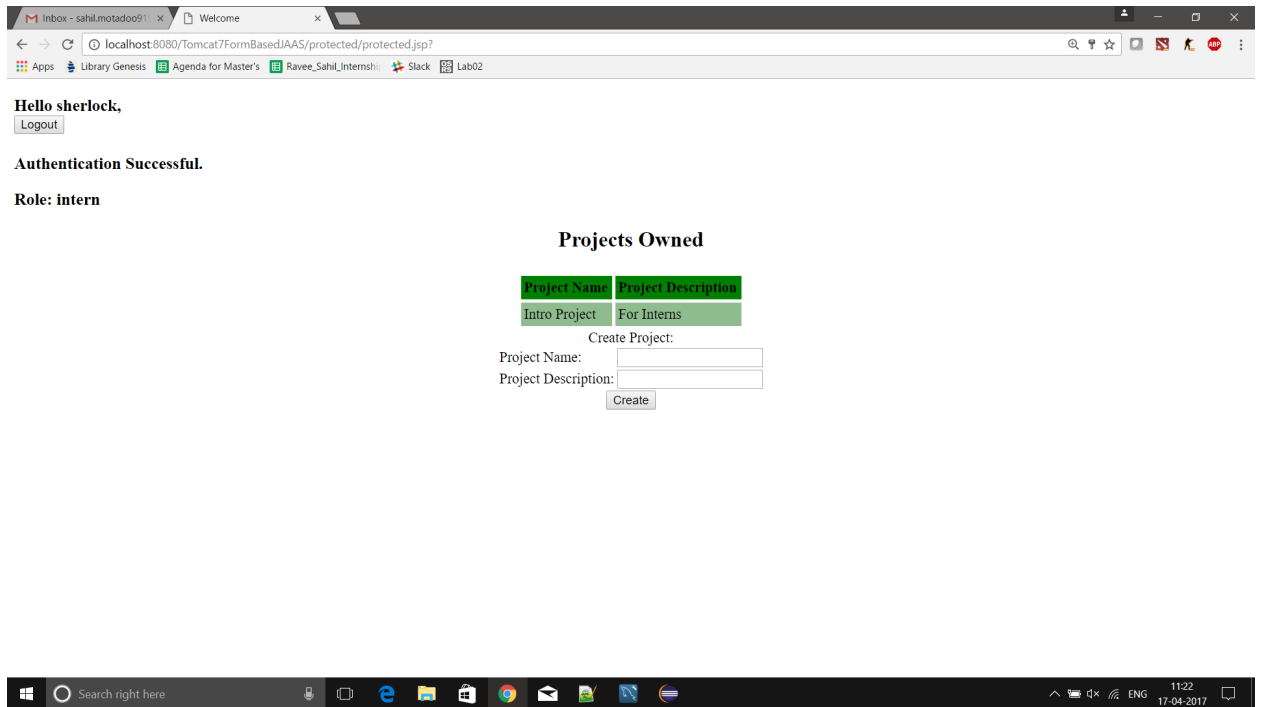
Project Name	Project Description
Project 7	iop

Create Project:

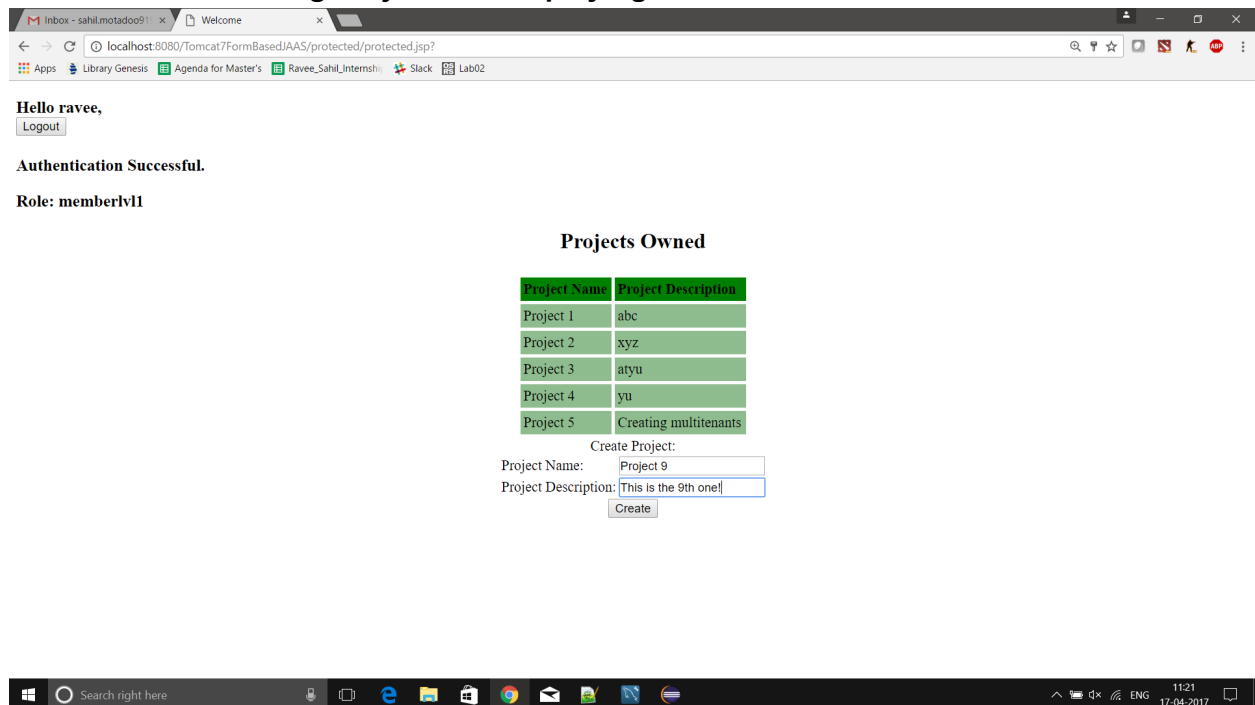
Project Name:

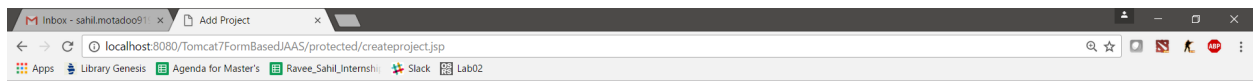
Project Description:





Screenshot of creating Project and displaying it





Project Created

[Main menu](#)



Hello ravee,
[Logout](#)

Authentication Successful.

Role: memberlv11

Projects Owned

Project Name	Project Description
Project 1	abc
Project 2	xyz
Project 3	atyu
Project 4	yu
Project 5	Creating multitenants
Project 9	This is the 9th one!

Create Project:

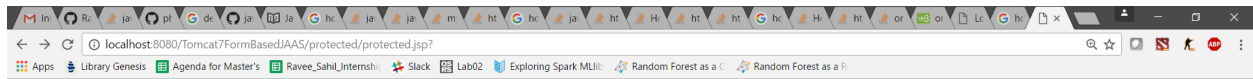
Project Name:

Project Description:



Part 3

“Sahil” is the user who has the role of manager (analogous to admin/cloud service provider). Only the manager has access to add user.



Hello sahil,
[Logout](#)

Authentication Successful.

Role: manager

Projects Owned

Project Name	Project Description
Project 7	iop

Create Project:

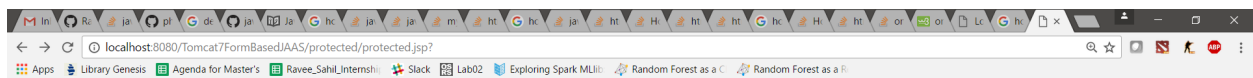
Project Name:
Project Description:

Assign Project:

Project Name:
Roles:

Add Users:

Username:
Password: Roles:



Hello sahil,
[Logout](#)

Authentication Successful.

Role: manager

Projects Owned

Project Name	Project Description
Project 7	iop

Create Project:

Project Name:
Project Description:

Assign Project:

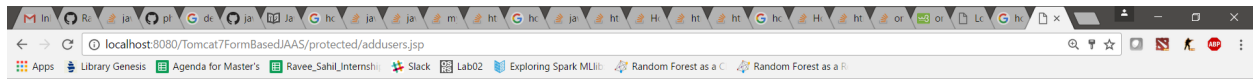
Project Name:
Roles:

Add Users:

Username:
Password: Roles:



On successful adding, we are redirected to a new page, saying the user is added.



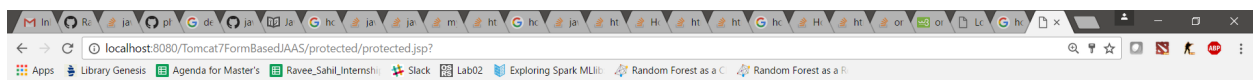
User added!

[Main menu](#)



Now, “Ravee” who has only memberlv1 access (that is, she is not an admin) does not have the authority to add user.

She is now denied access if she tries to add user



Role: memberlv1

Projects Owned

Project Name	Project Description
Project 1	abc
Project 2	xyz
Project 3	atyu
Project 4	yu
Project 5	Creating multitenants
Project 9	This is the 9th one!
Project 11	This is 11

Create Project:

Project Name:
Project Description:

Assign Project:

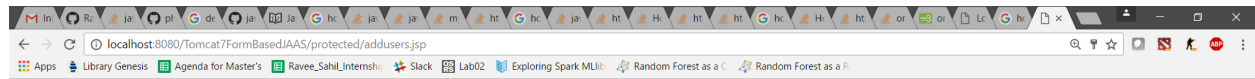
Project Name:
Roles:

Add Users:

Username:
Password: Roles:



Ravee is denied access while adding users

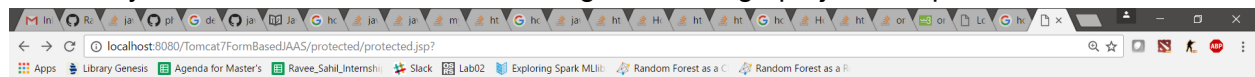


You are not Authorized!



Similarly, only the manager can assign project.

Here, only, the user “Sahil” who is the manager can assign project to a particular user.



Hello sahil,

[Logout](#)

Authentication Successful.

Role: manager

Projects Owned

Project Name	Project Description
Project 7	iop

Create Project:

Project Name:
Project Description:

Assign Project:

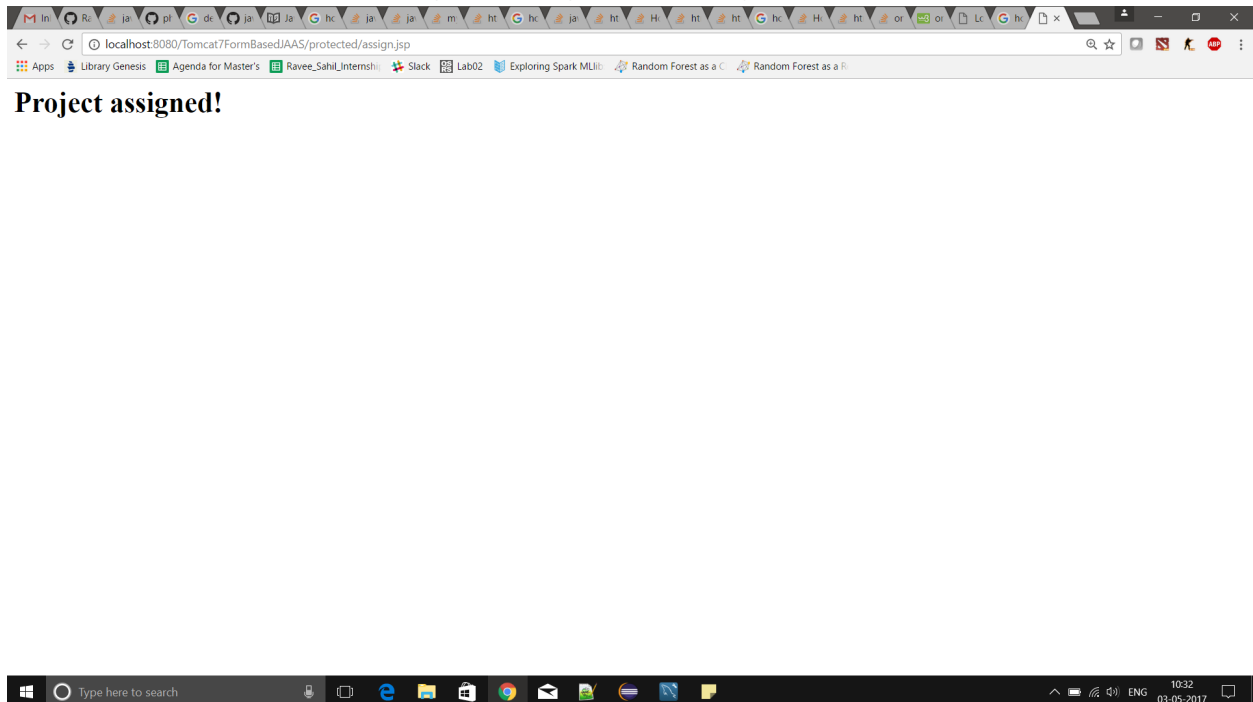
Project Name:
Roles:

Add Users:

Username:
Password: Roles:



You are redirected to another page on assignment of project.



Code Snippets

