



**URL Shortener
(CPSC 476: Programming Project- 3)**

December 12nd, 2016

**Submitted by,
Husamaldeen Naji
Neha Gawade
Karthik Karunanithi**

**Prepared for
Dr. Kenytt Avery
California State University Fullerton**

Version 1.1

- **Spring MVC integration: using Spring Framework's Controllers**
- **Data storage: HSQL database**
- **Design: Bootstrap CSS framework**

Purpose

The purpose of this project is to build a Client server application using Spring MVC concept. that needed re-factoring the code of project 2 to use Spring Framework's @Controllers instead of using the servlets. Jsp pages are still the view method. HSQLDB is also still used as the data store. The project talks about shortening a URL when a user inputs a long URL and the reverse of it when the user enters a shortened URL.

How it works

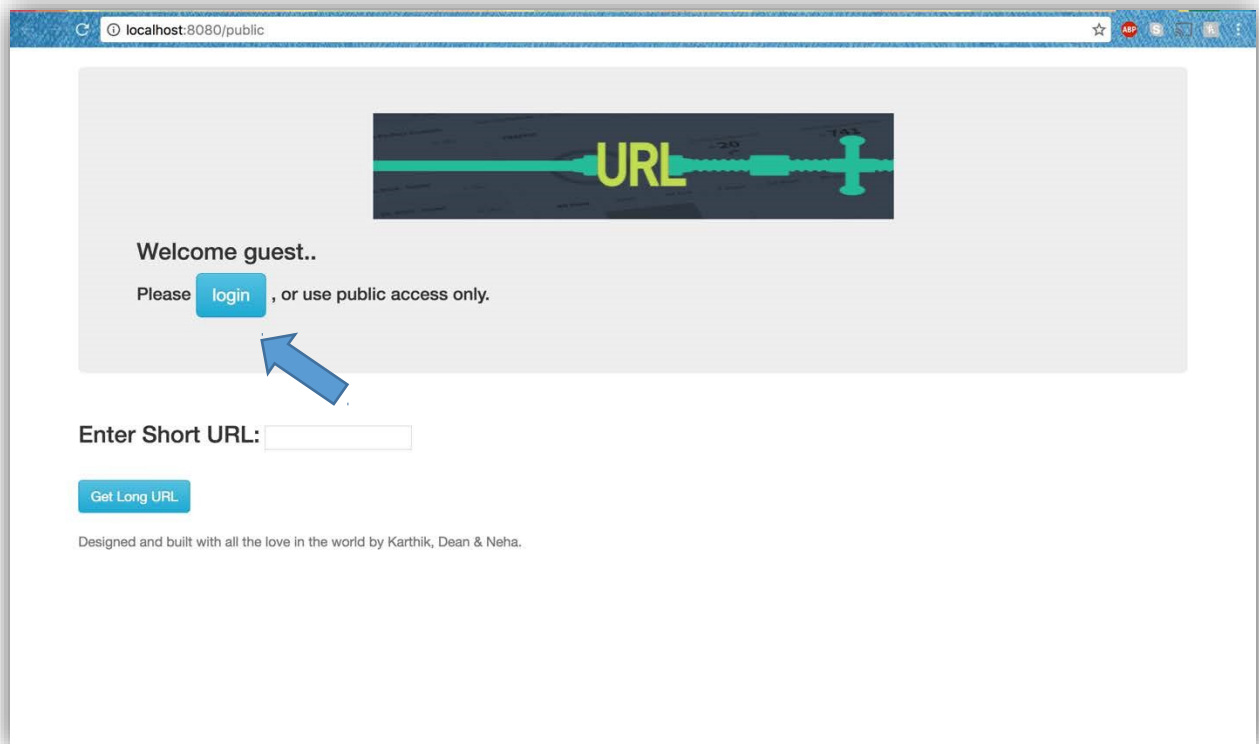


Figure 1 Homepage

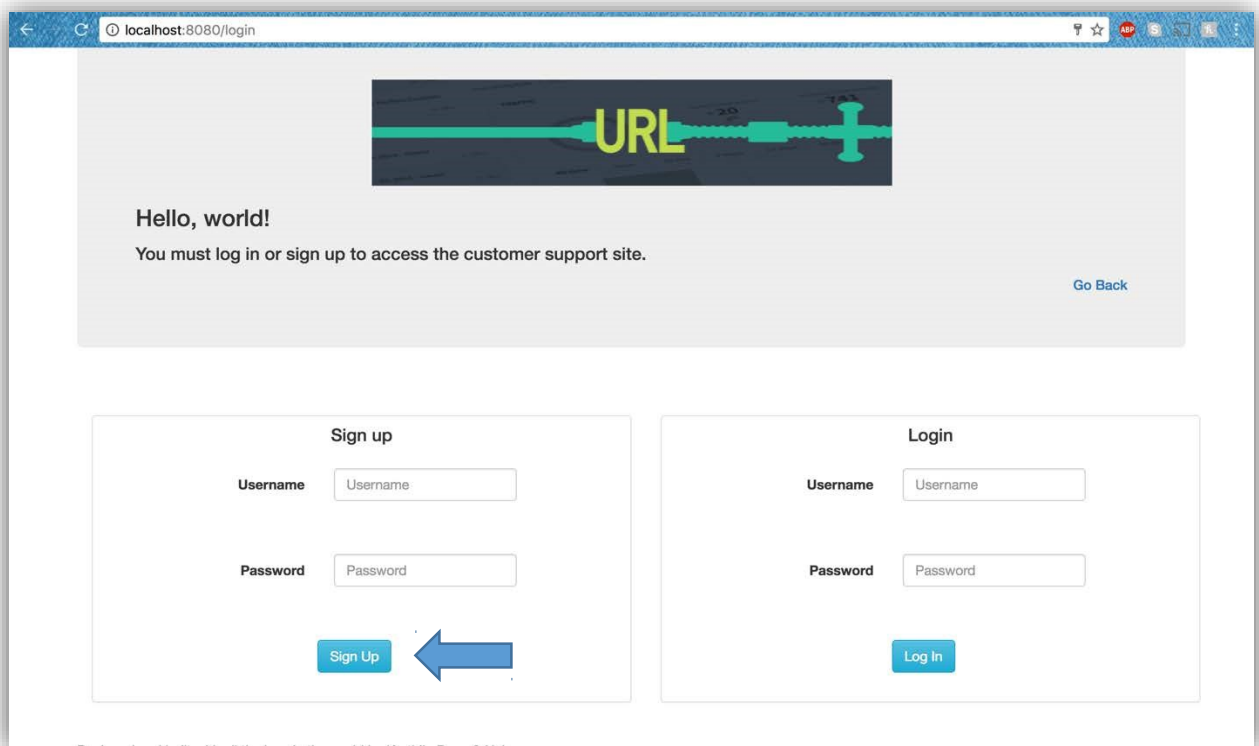


Figure 2 Login & Signup page

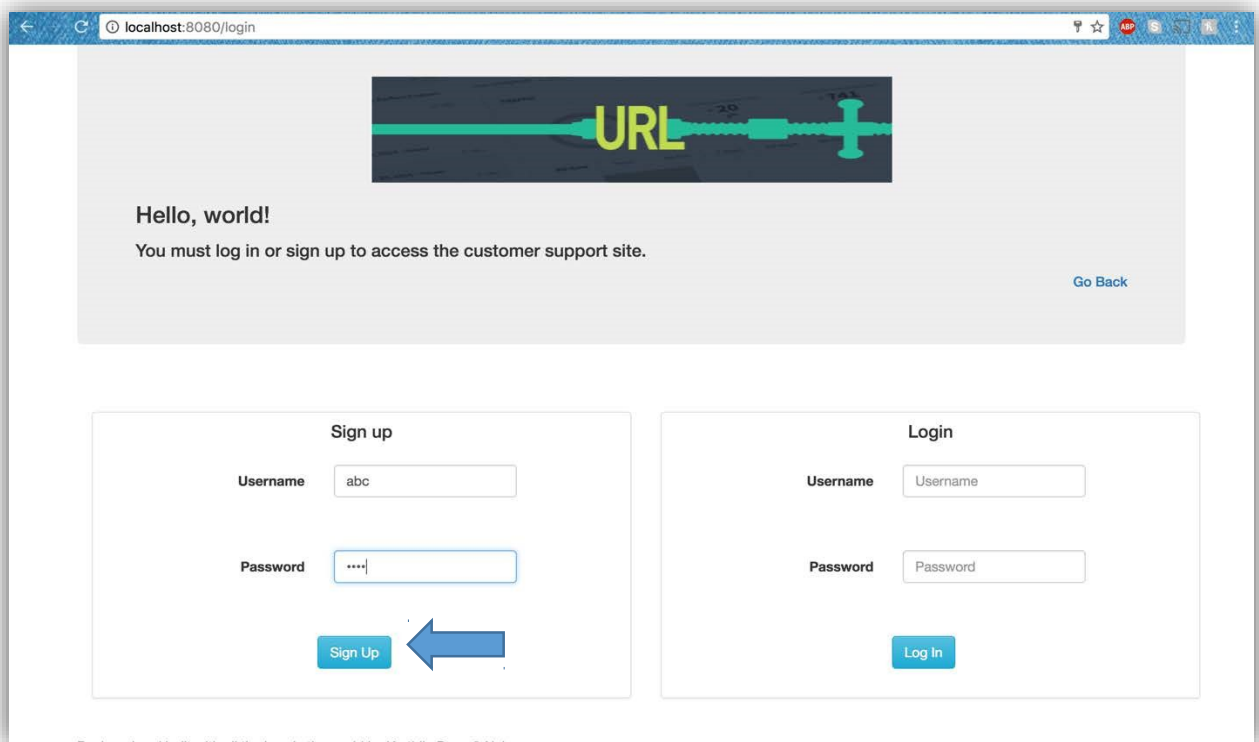


Figure 3 Enter details for Signup

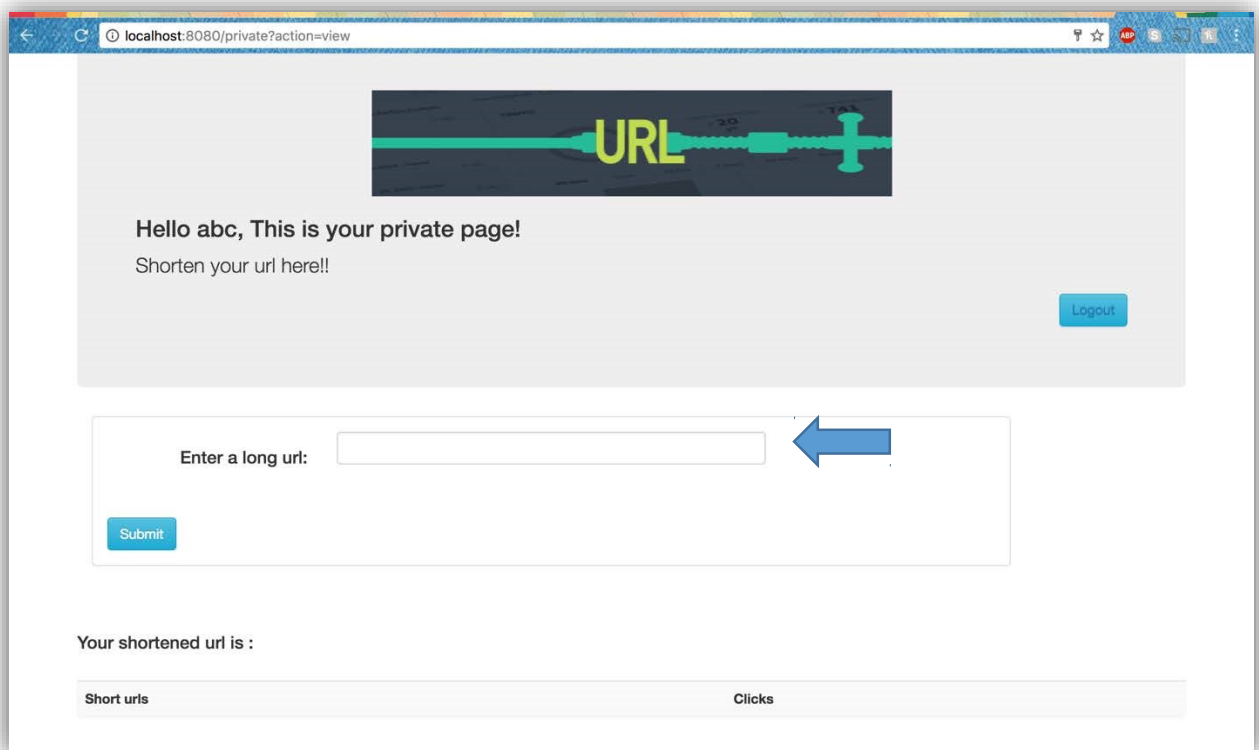


Figure 4 After Signup- Private Page

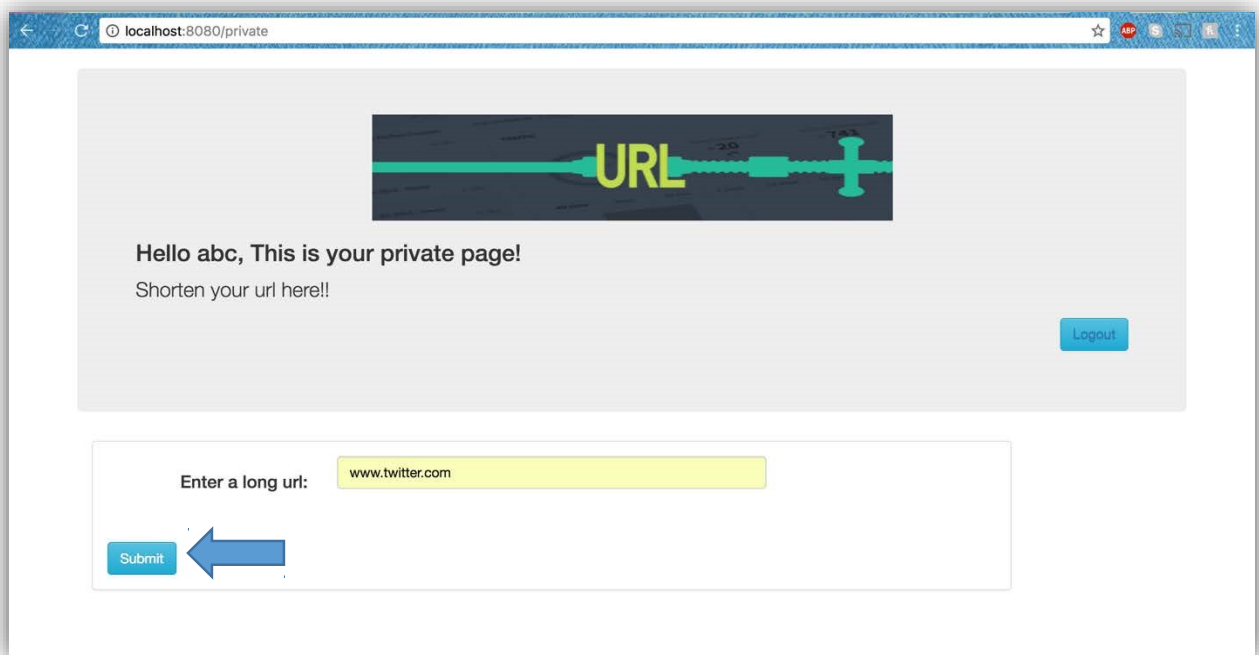


Figure 5 Enter URL to shorten

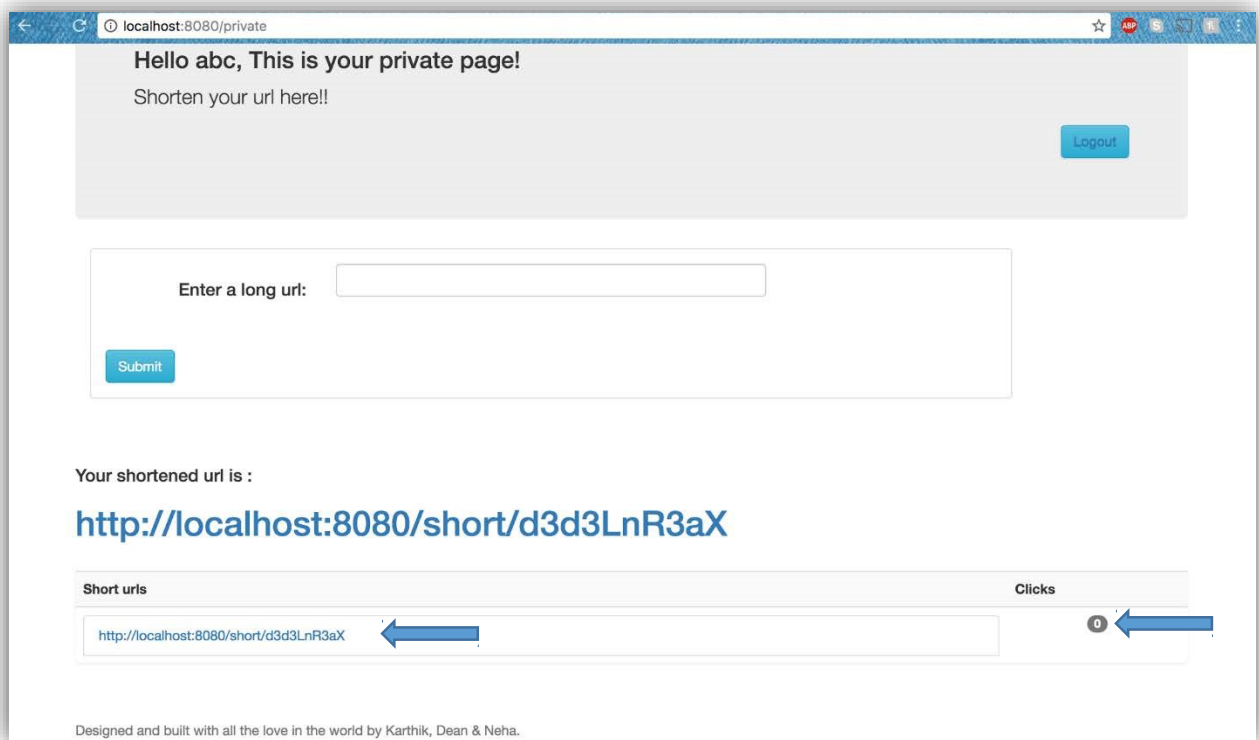


Figure 6 Get shortened url and public clicks count for that url (Url shortening is done by using `java.util.Base64` class)

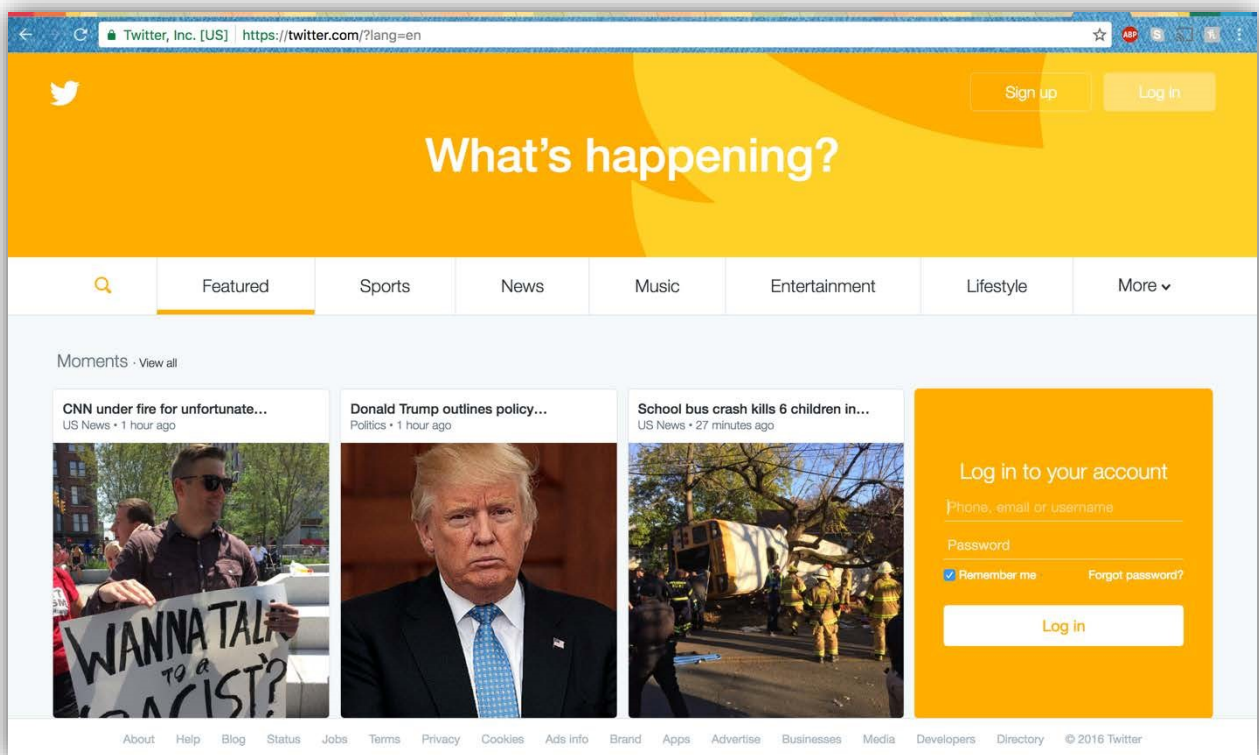


Figure 7 If you click on shortened url link, it will redirect you to long url page

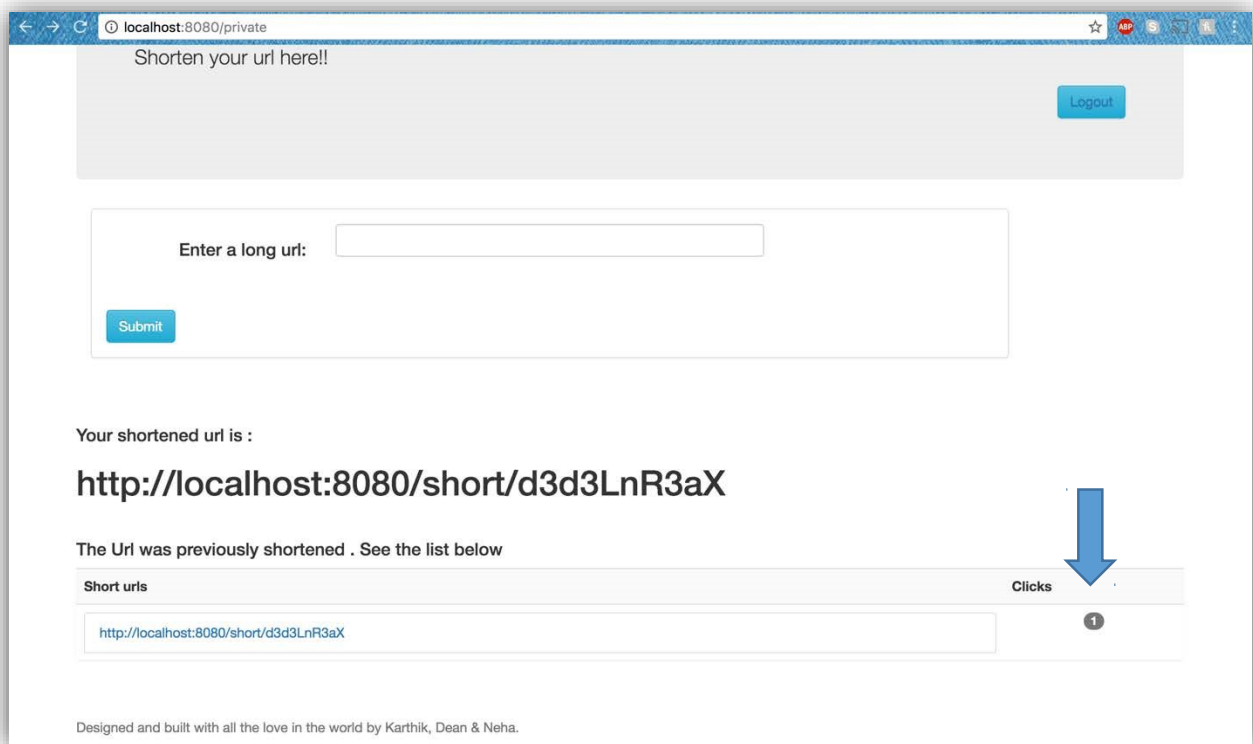


Figure 8 *If you go back and refresh page, clicks count increases by 1*

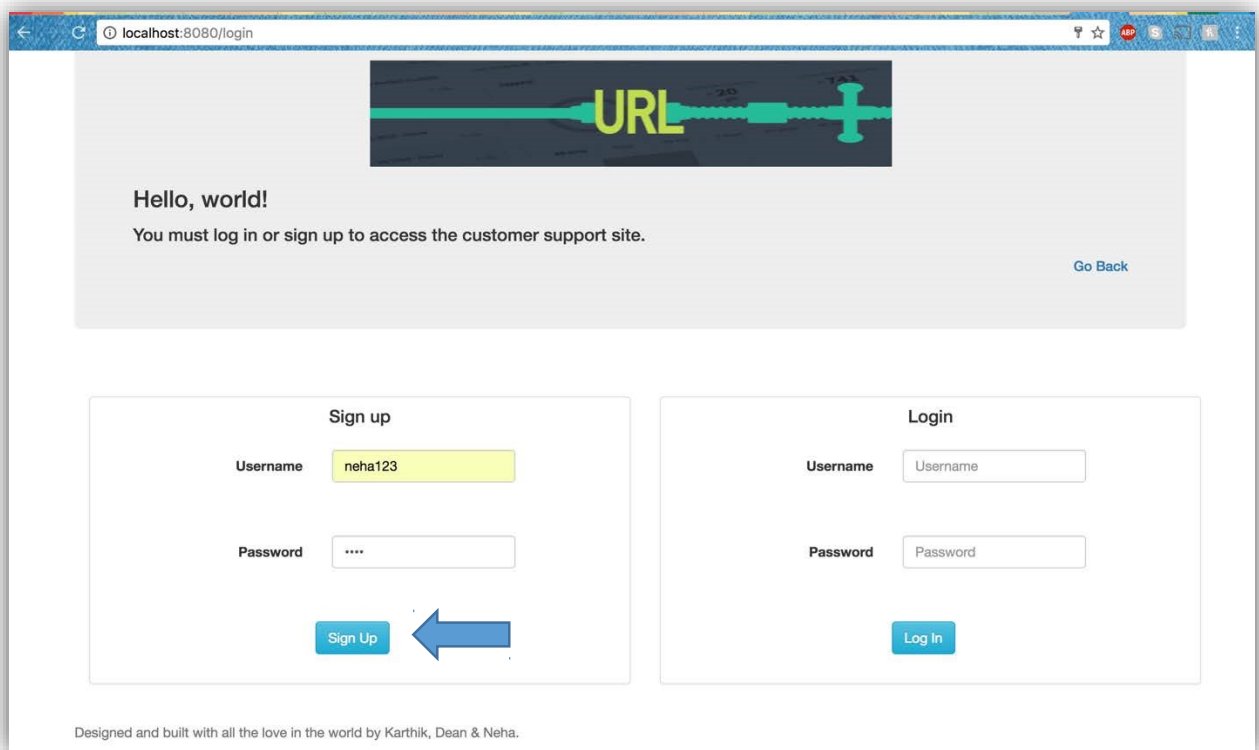


Figure 9 Signup as other user

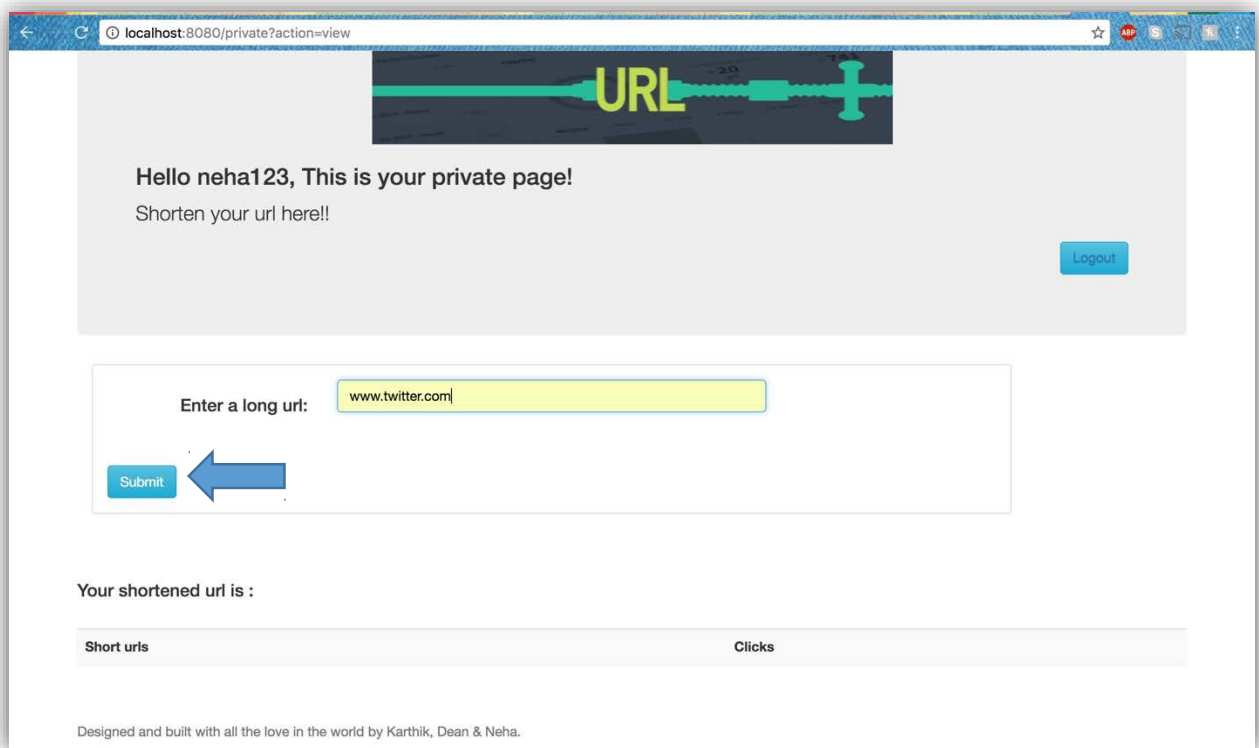


Figure 10 Shorten same url as the other user did

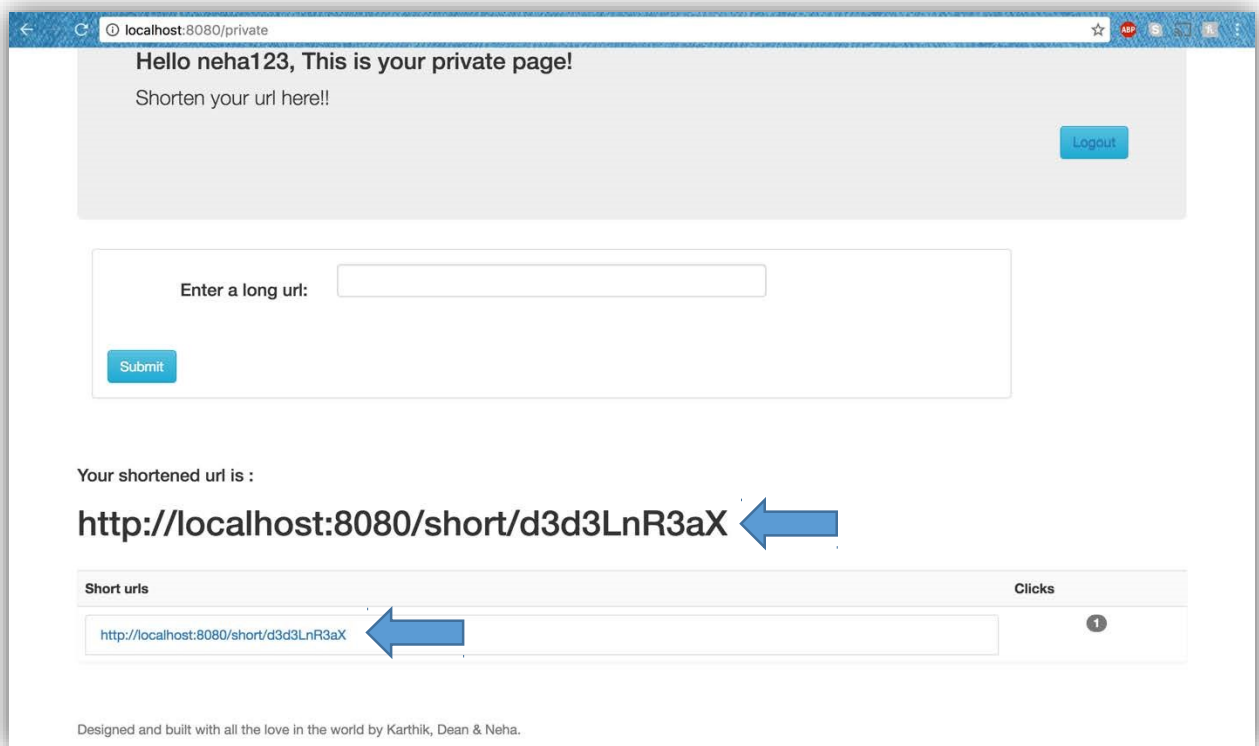


Figure 11 You will get shortened url and the click count same as other user

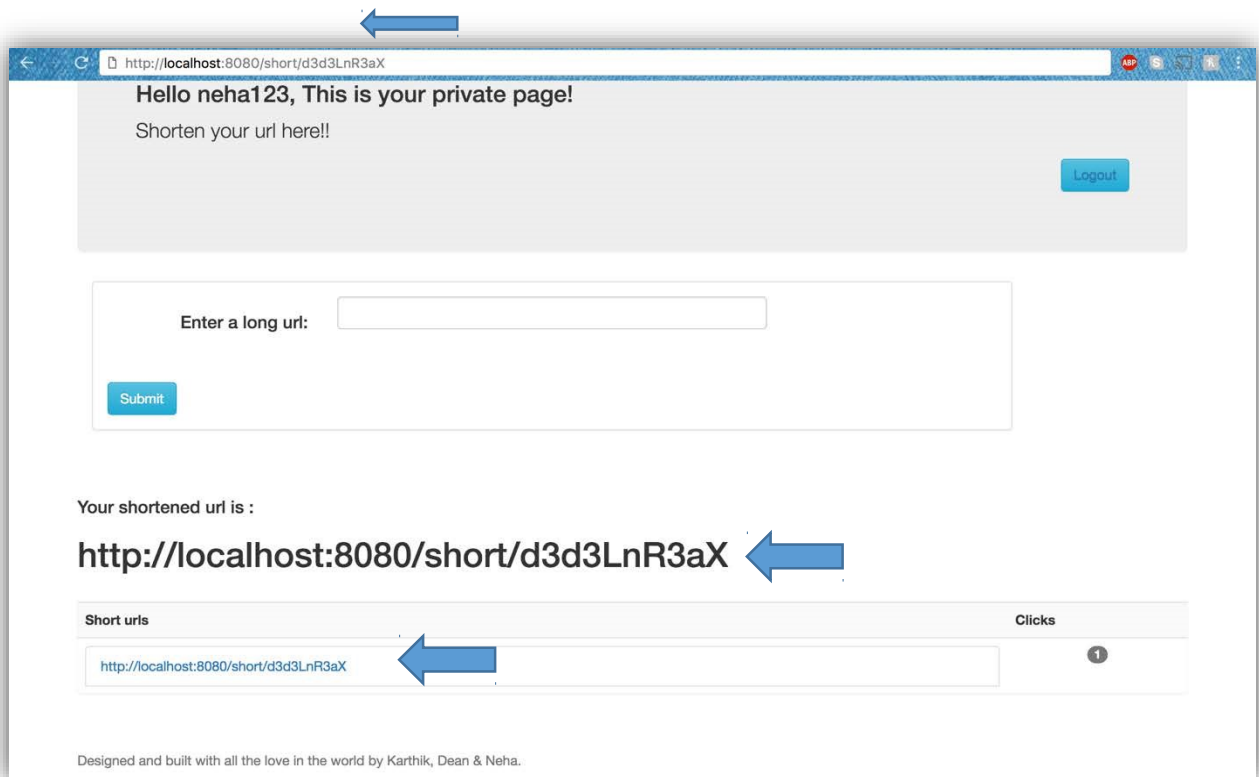


Figure 12 Copy paste shortened url in web browser address bar

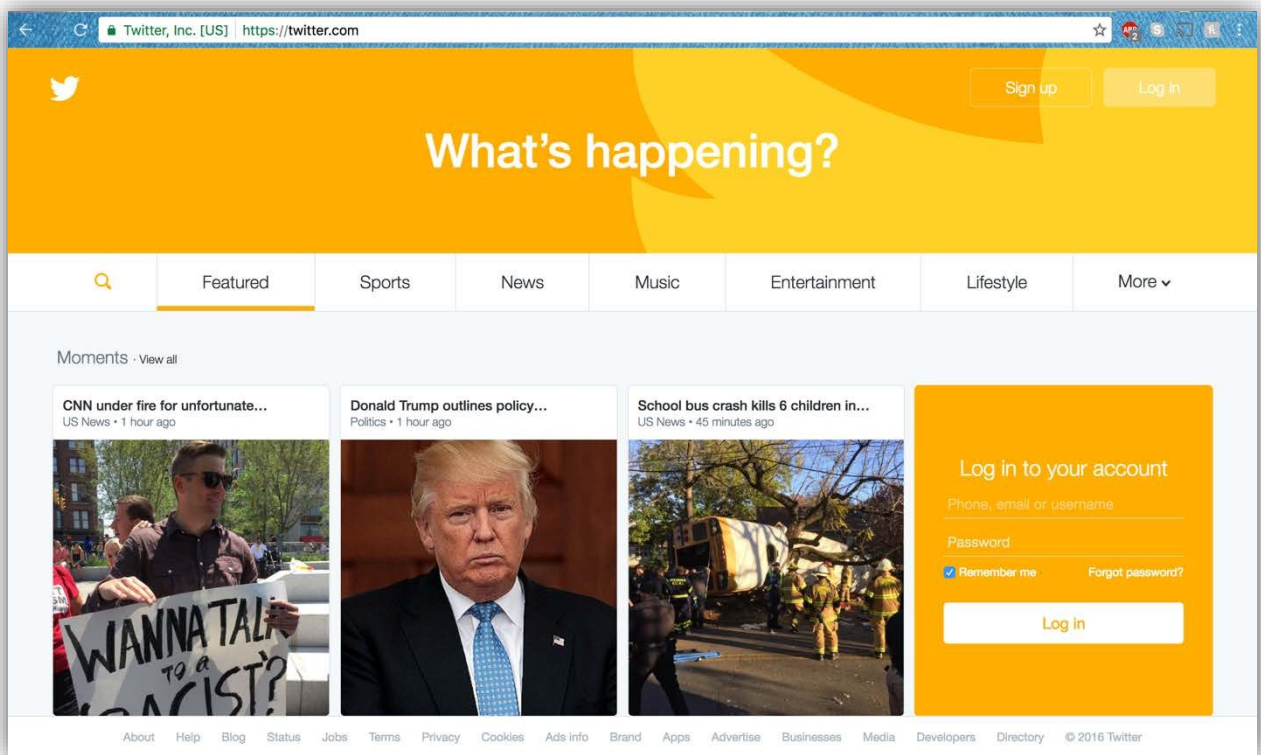


Figure 13 Redirects to long url page

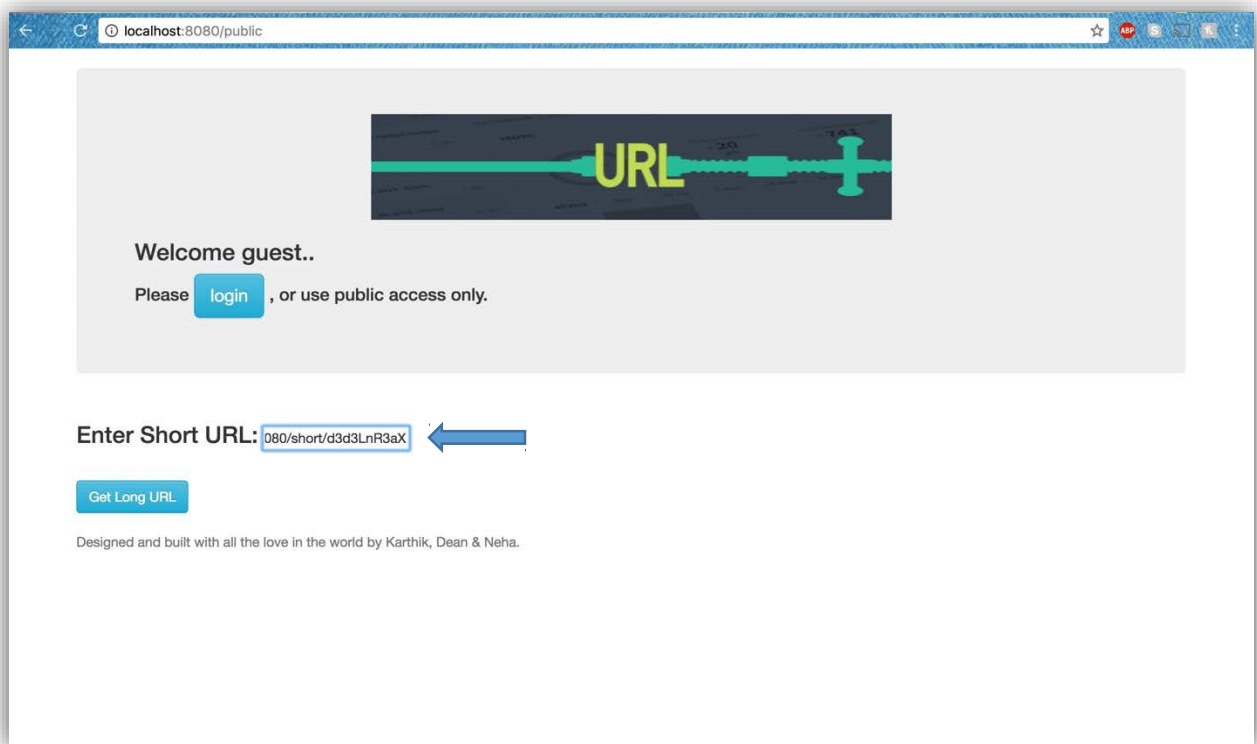


Figure 14 Public page, enter shortened url to get long url

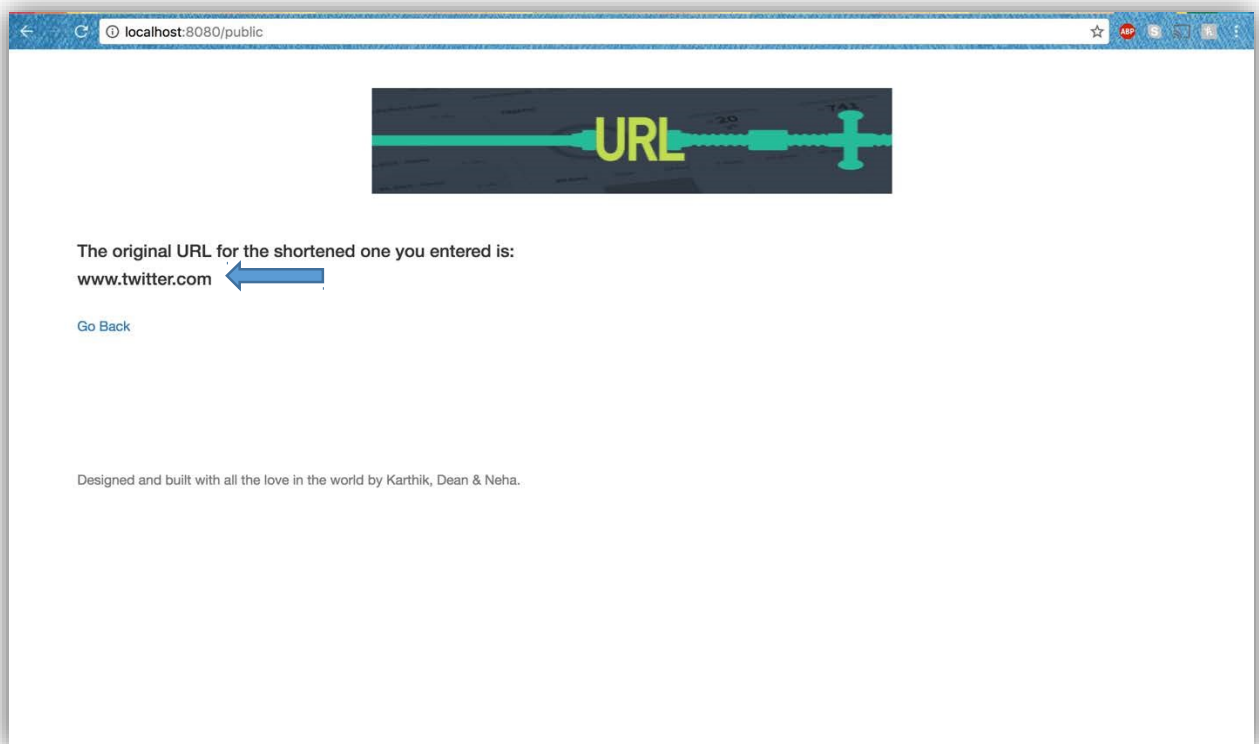


Figure 15 Get long url from shortened url

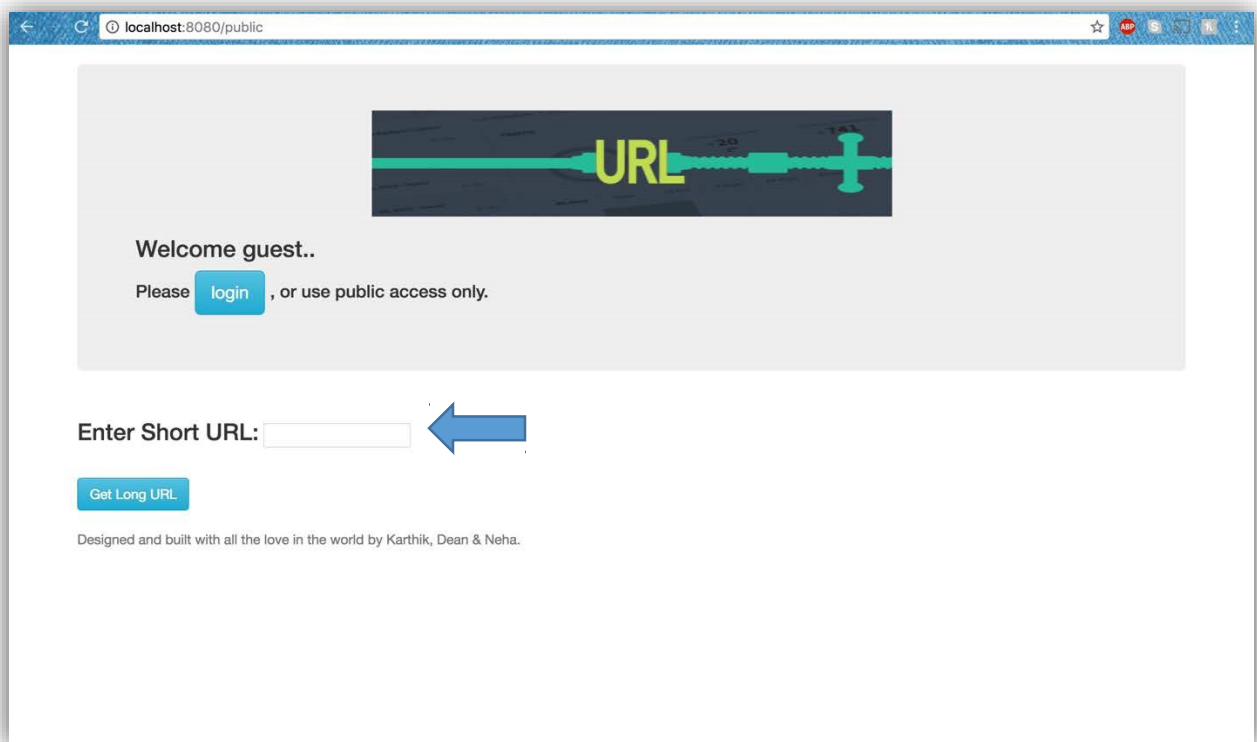


Figure 16 If you enter blank url and submit

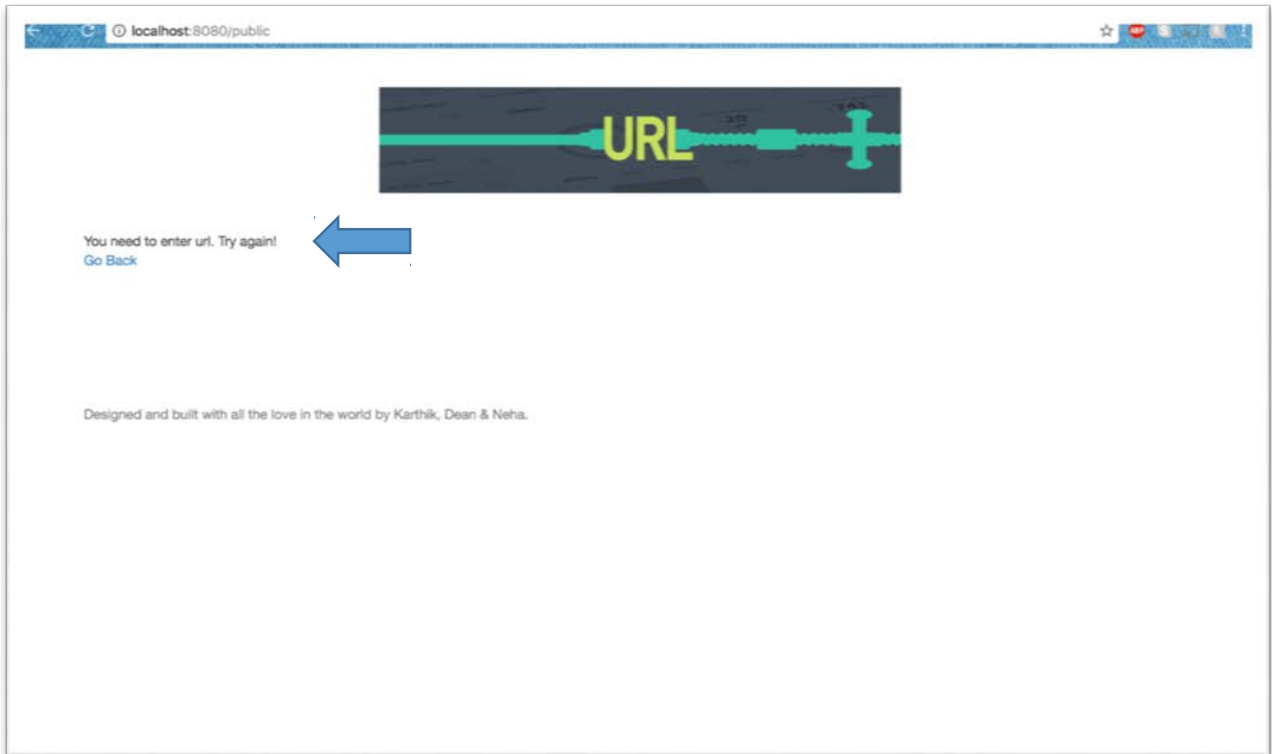


Figure 17 You get an error

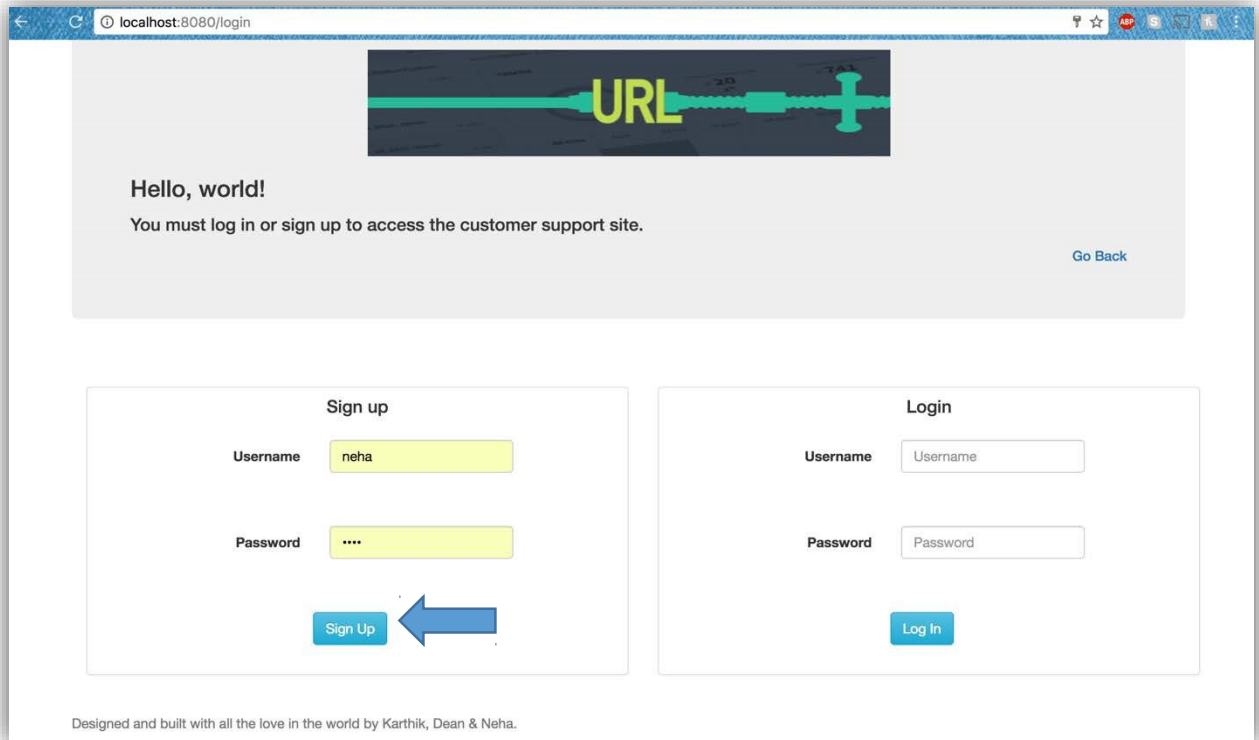


Figure 18 If you try to sign up with existing username

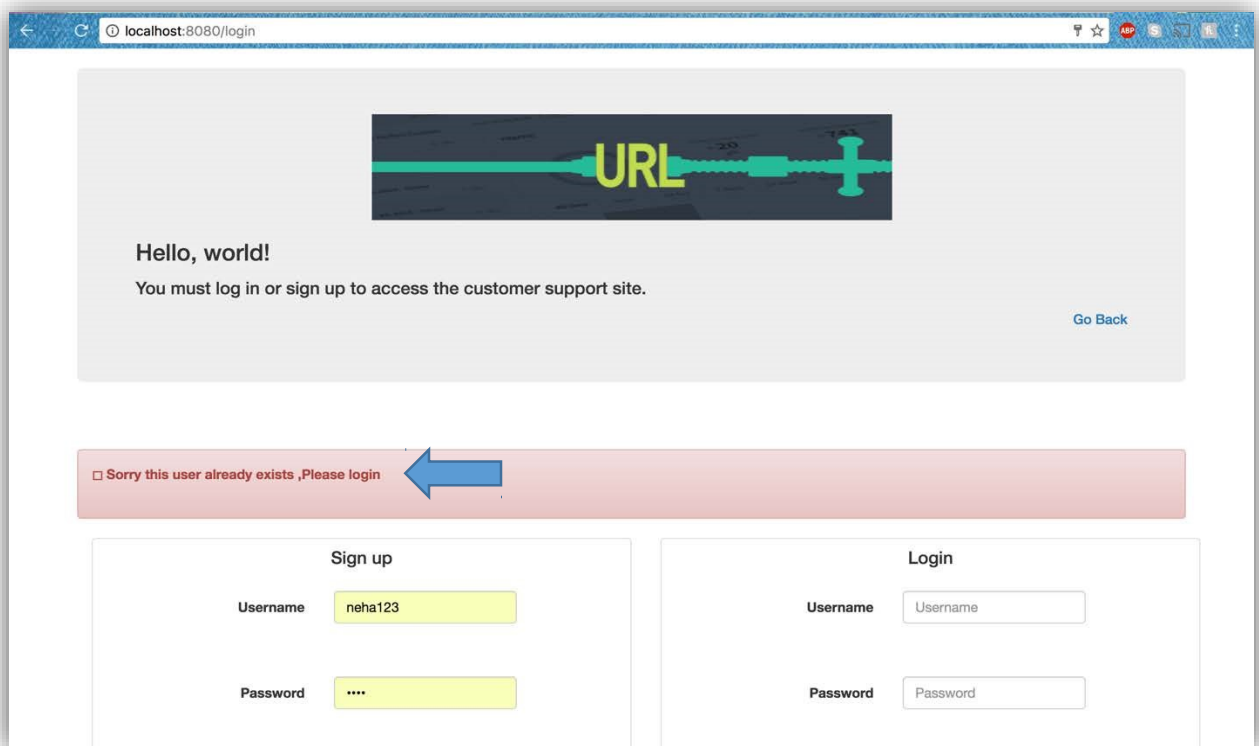


Figure 19 you get an error

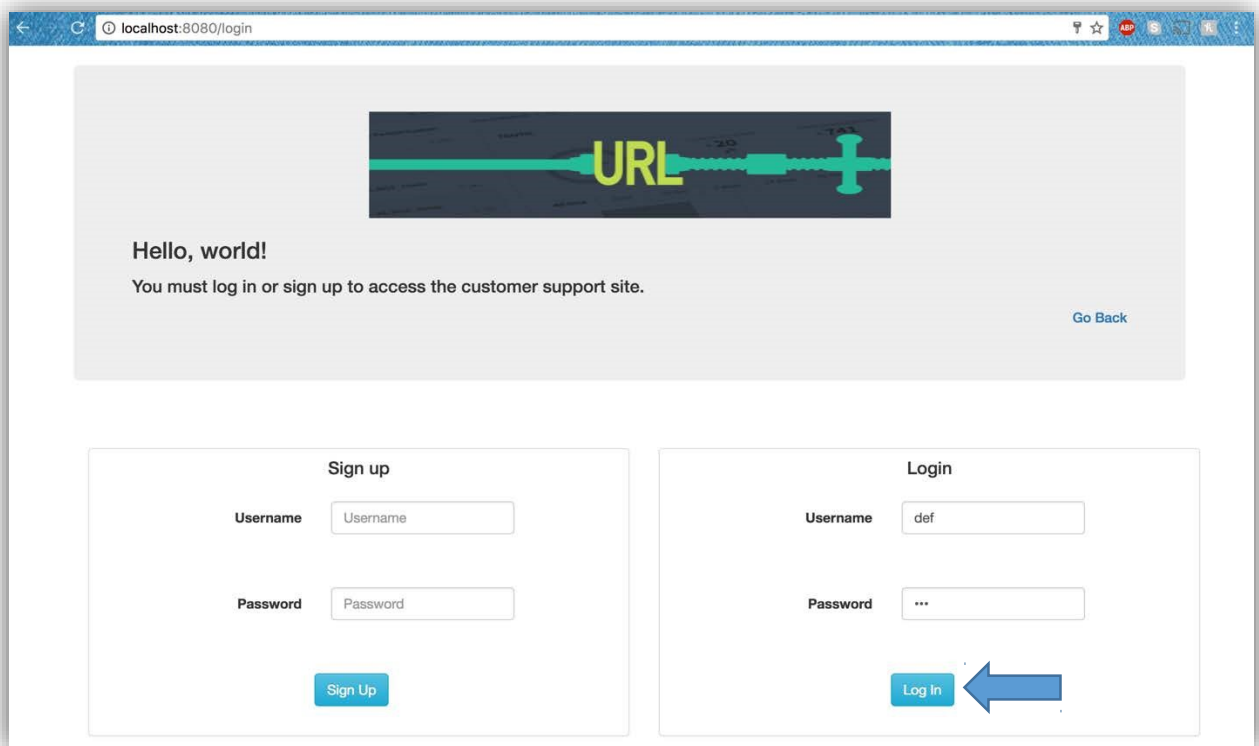


Figure 20 if you login with a username that doesn't exist

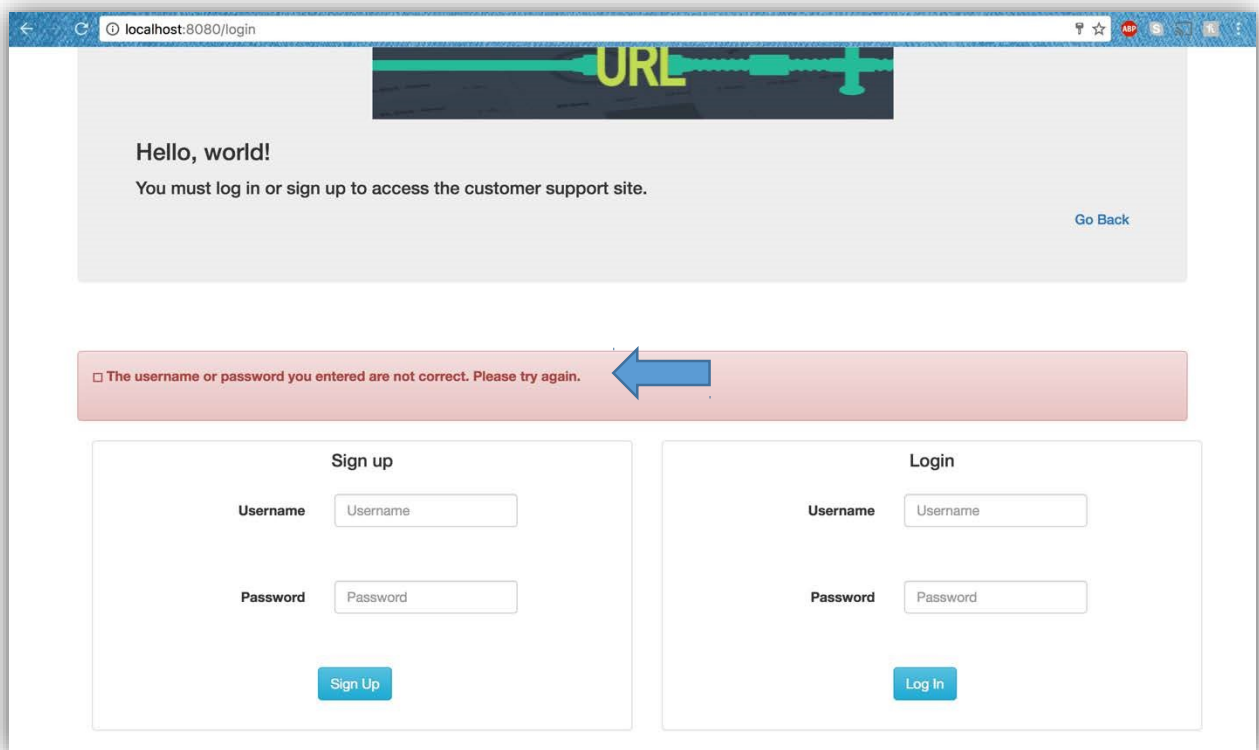


Figure 21 you get an error

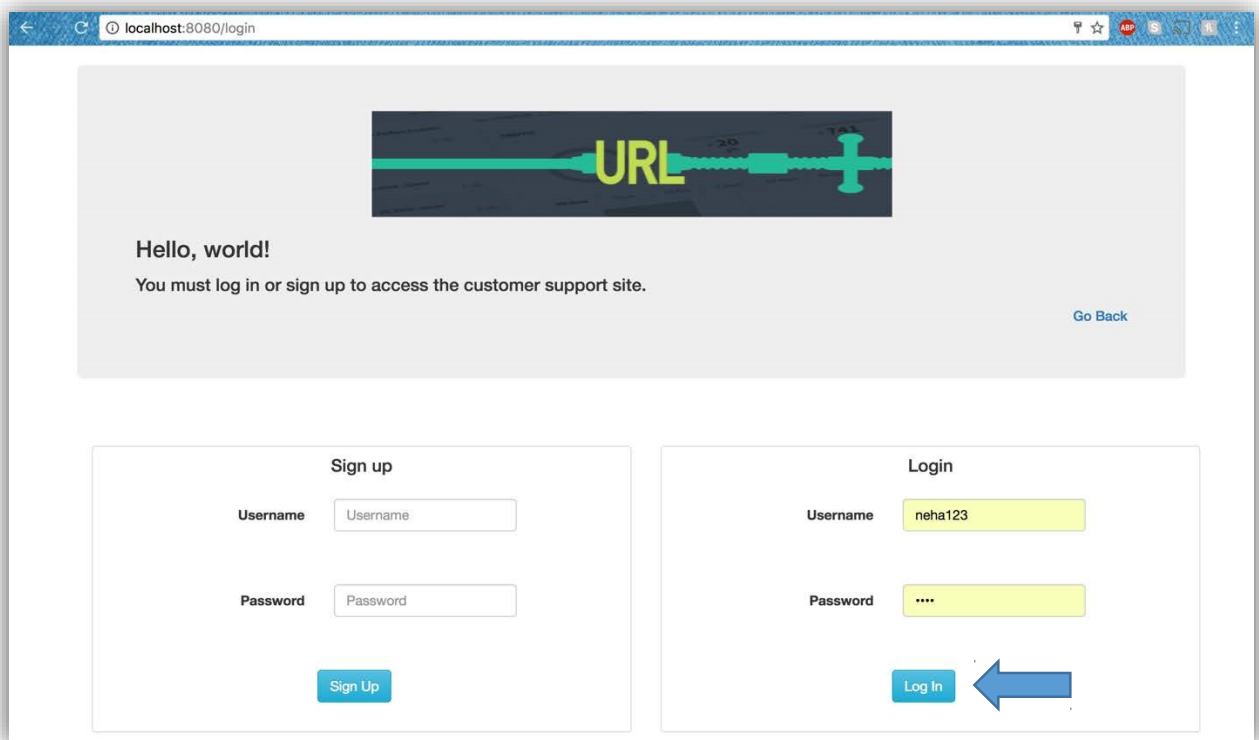


Figure 22 login again with existing username

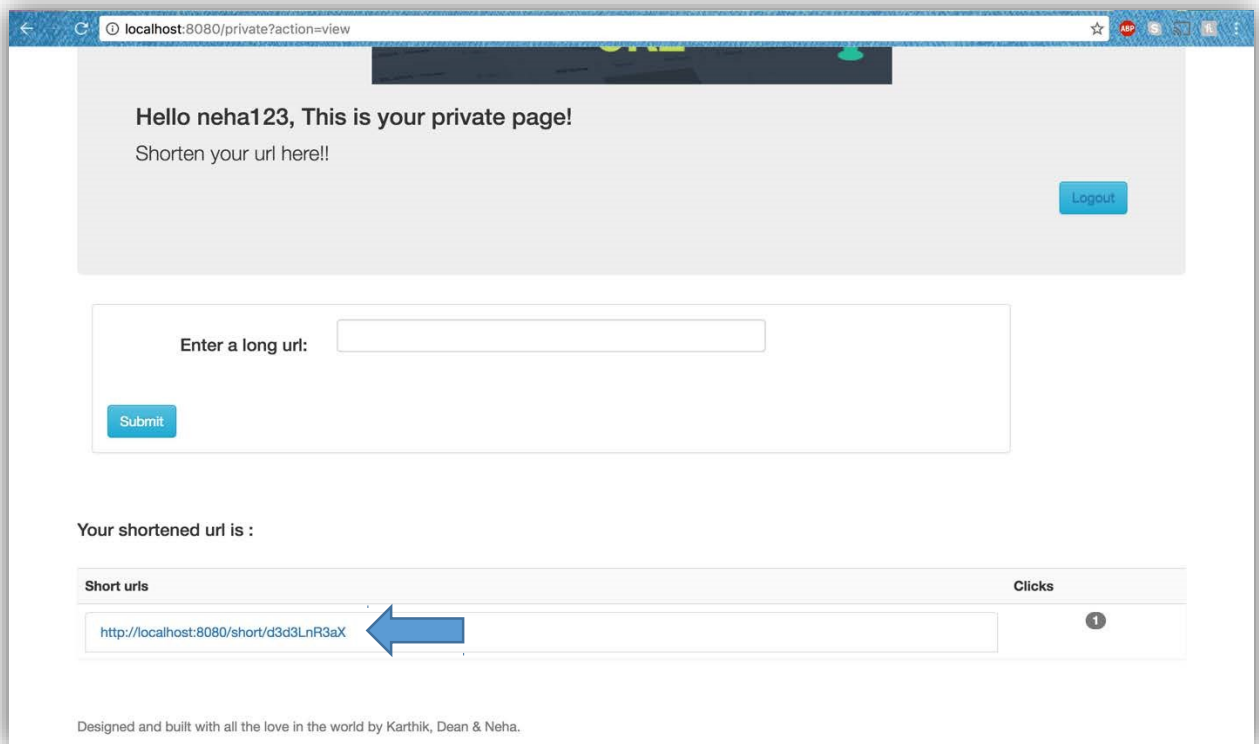


Figure 23 you get the details back

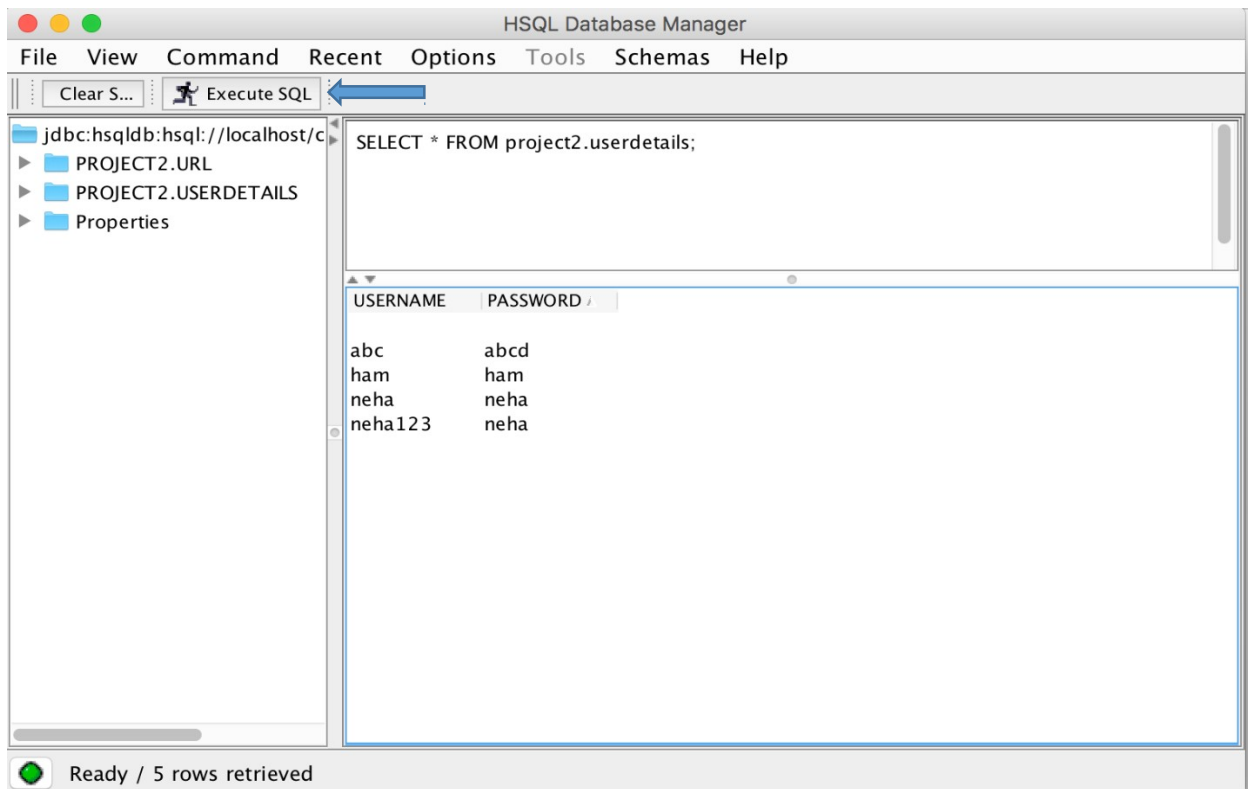


Figure 24 project2.userdetails database

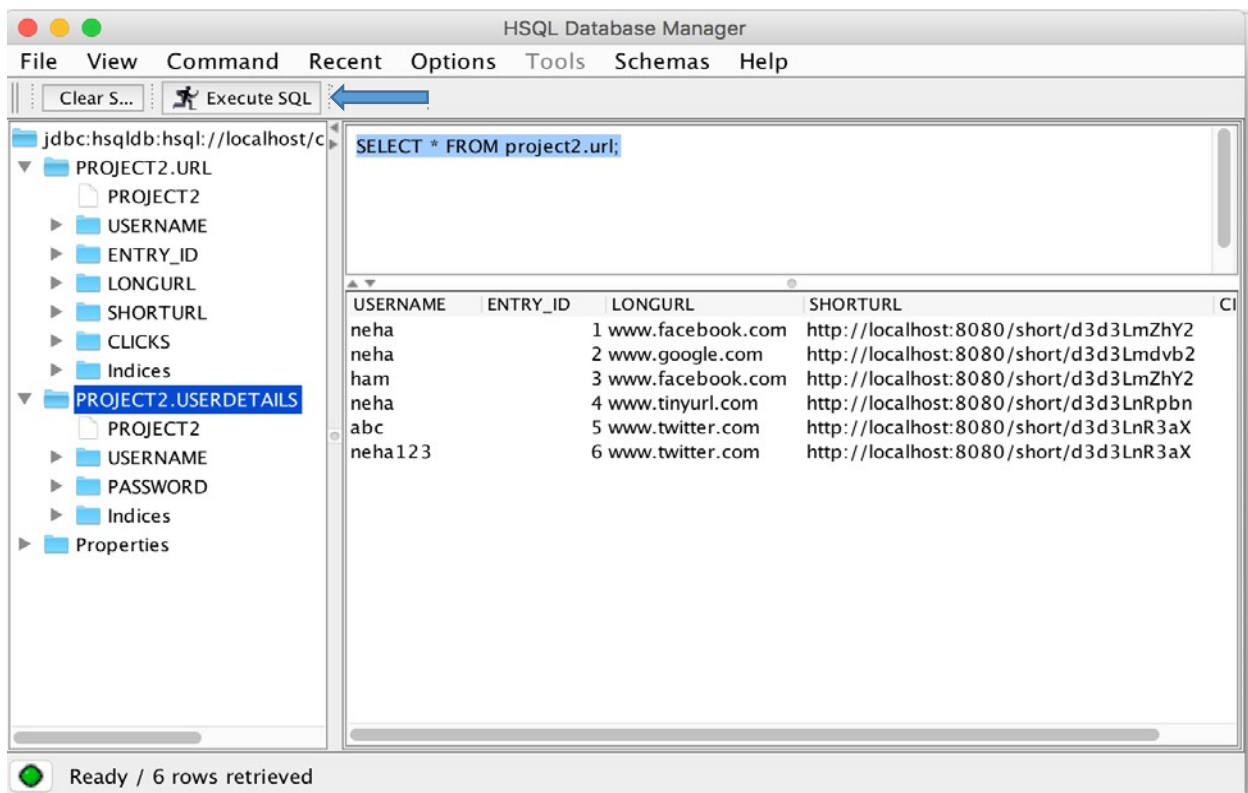


Figure 25 project2.url database

Setup Instructions:

Database

1. Download and install [Apache Ant](#).
2. Download [build.xml](#) and place it in a new, empty directory
3. Go to the directory in command prompt and type **ant start** followed by **ant manage** to open the HSQL Database Manager
4. We have attached a sql file **finalscript.sql** containing the queries for creating all the databases. You can execute them by following steps:

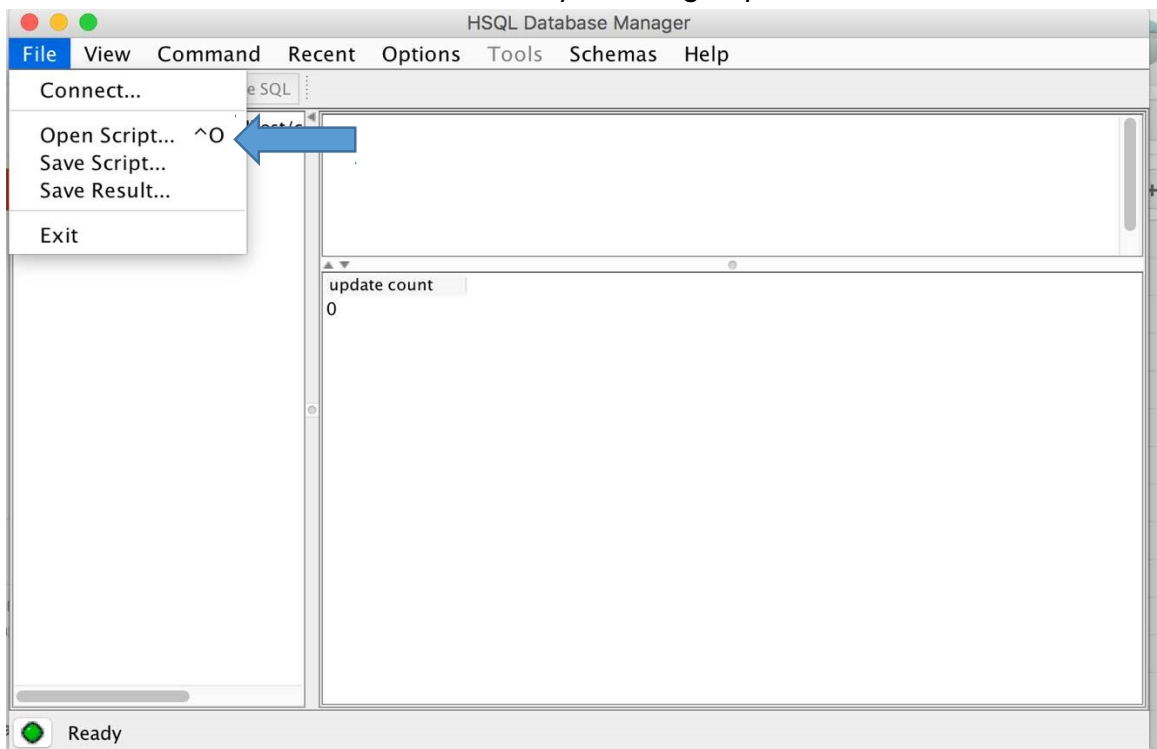


Figure 26 In HSQL Database Manager go to file -> Open Script

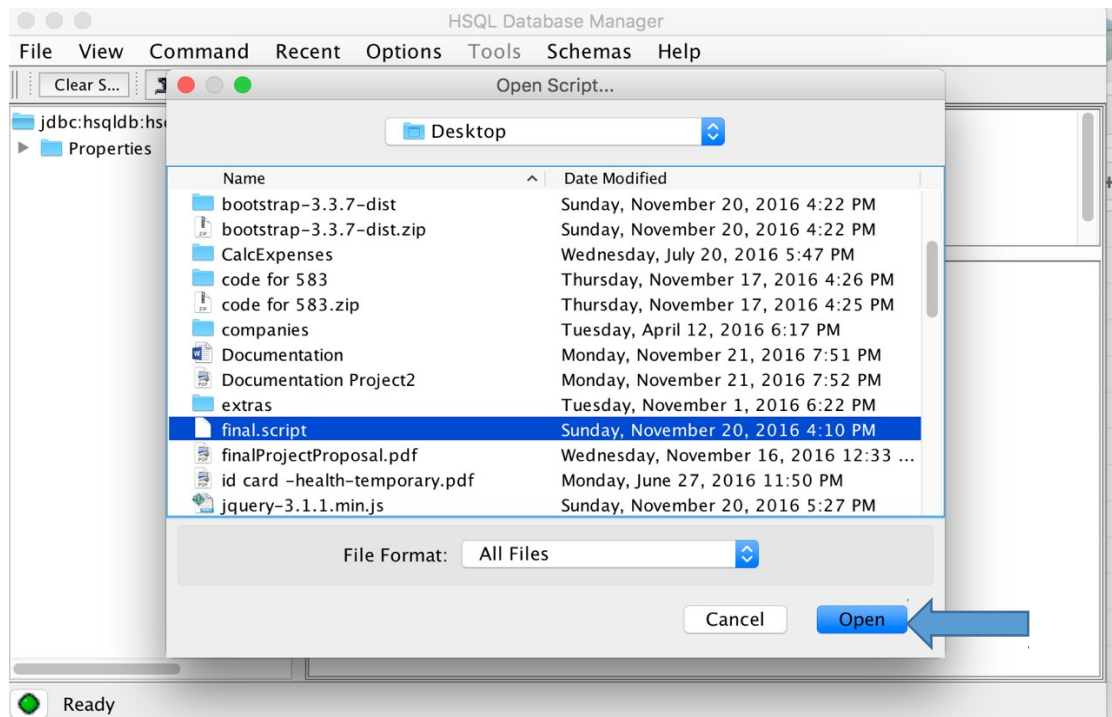


Figure 27 In open script select the *final.script* file attached along with project source code in the email. Then click on open.

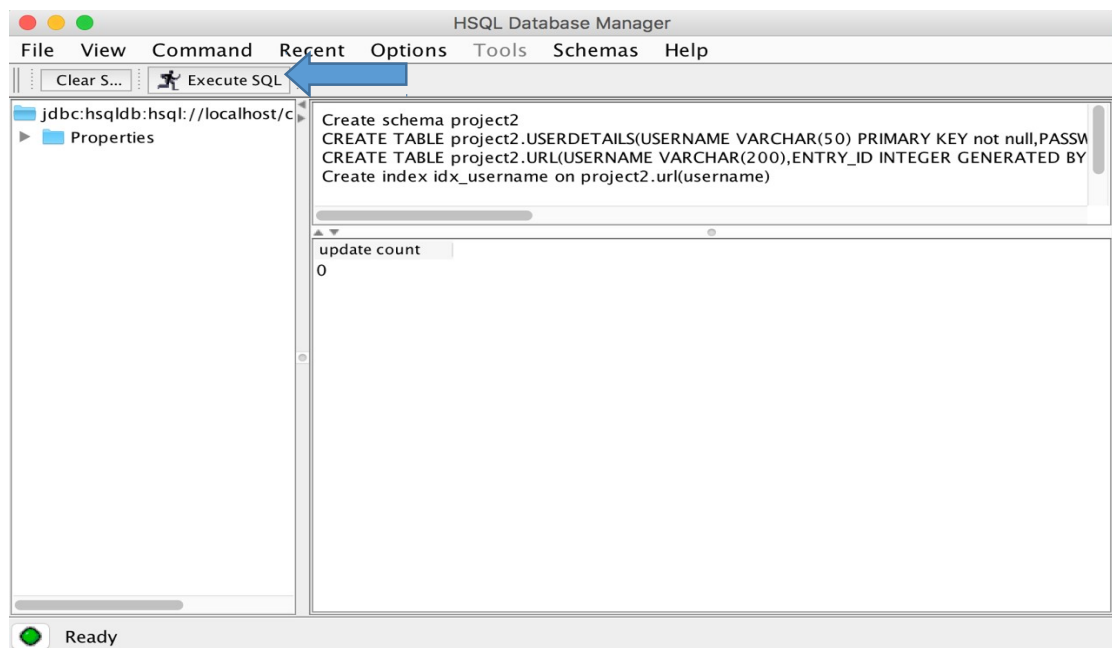


Figure 28 Make sure the queries don't have semicolon at the end, then click on Execute SQL

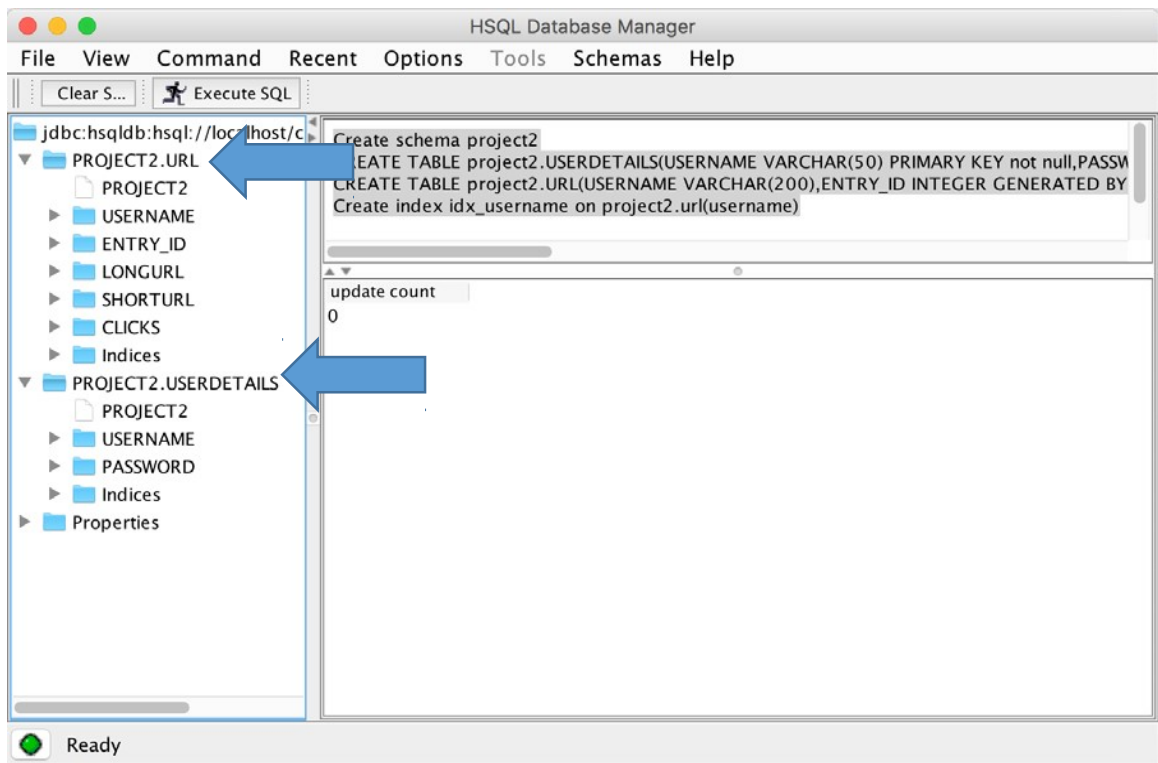


Figure 29 The schema and databases are ready to use

5. Sample insert statement for project3.url database is:

`insert into project3.url (username, longurl, shorturl, clicks) values ('red','www.google.com',
'http://localhost:8080/short/d3d3LmZhY2',0);`

Here, in this database 'entry_id' is generated uniquely on its own and we don't need to insert it. It is the primary key of the table.

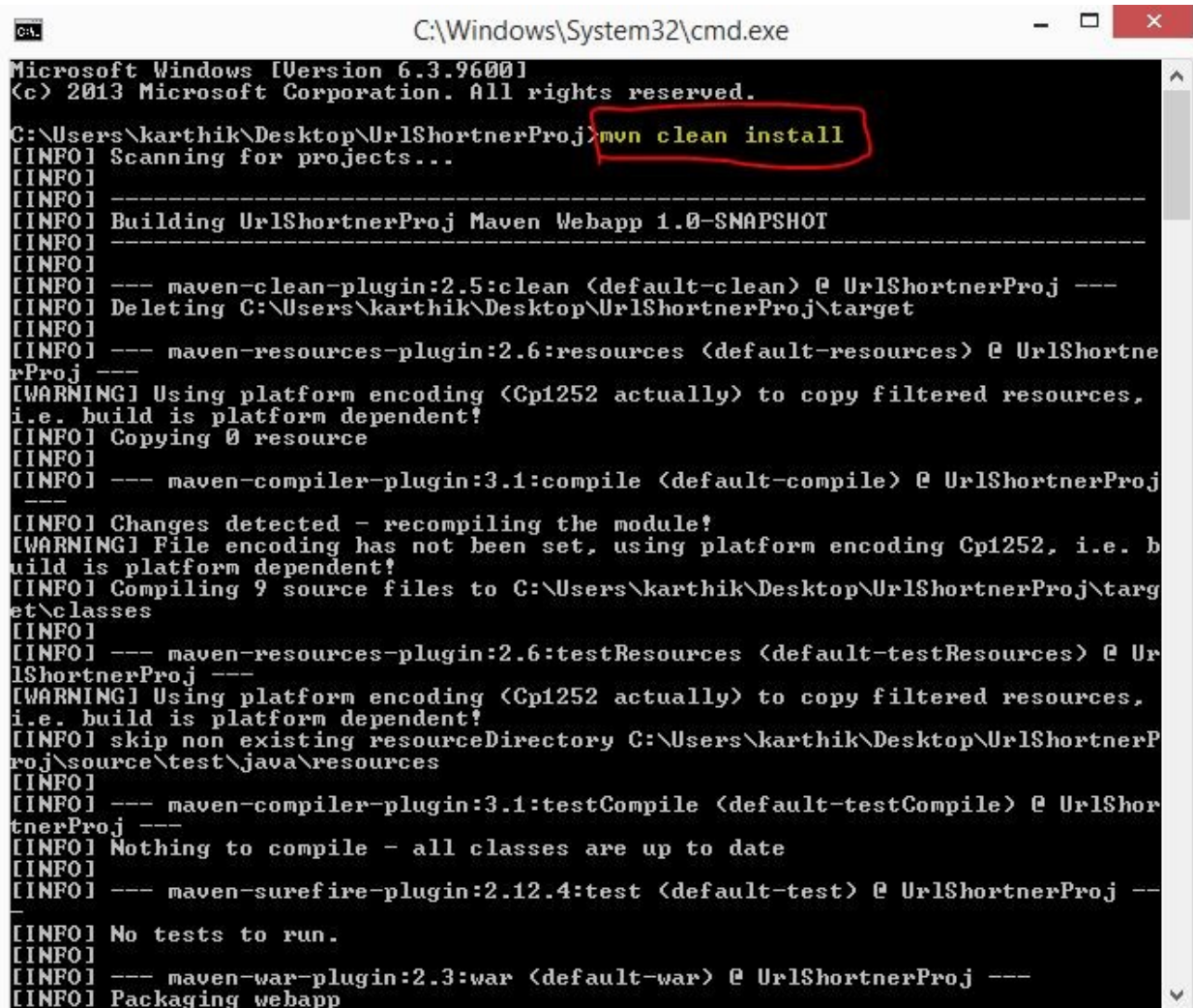
6. In project3.userdetails table 'username' is the primary key. It is also the foreign key in project3.url database.

Web Application

1. We have implemented the project using the Eclipse IDE. So importing the ZIP file and also adding the JSTL jar file (if missing) to WEB-INF/lib folder and JAVAX servlet jar to the eclipse build path should help the application running 1.1 In eclipse import the zip file as maven java EE file.
- 1.2 Make sure the context root is '/'. This can be modified using the web project settings from the eclipse project properties.
- 1.3 Right click on the project root folder and choose Run on server to get the application running on the browser.

- 1.4 Make sure to start the hsql db server by typing ant start and open the script file attached for the program to work
2. The other way to do it is by navigating to the project directory and typing the command **mvn clean install** where the pom.xml is located See the steps below:

Installation STEP 1: Navigate to directory and type mvn clean install

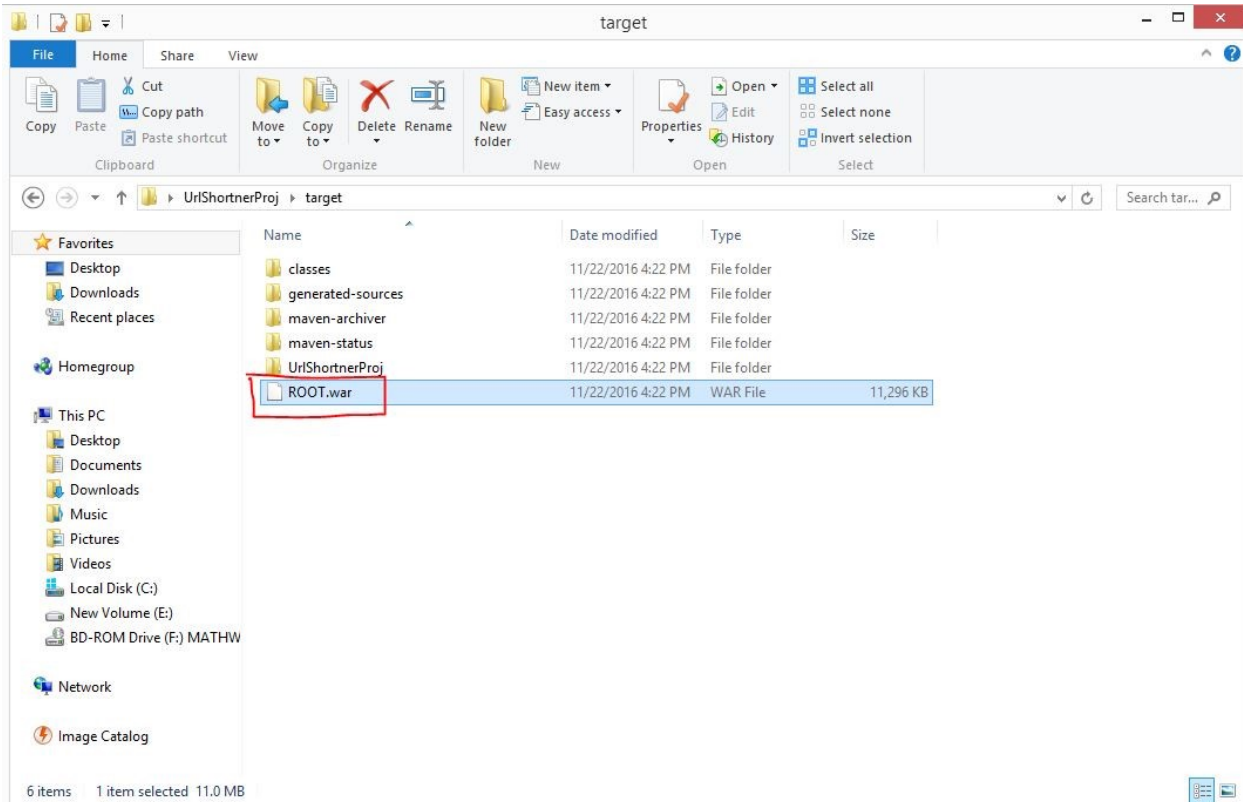


```
C:\Windows\System32\cmd.exe

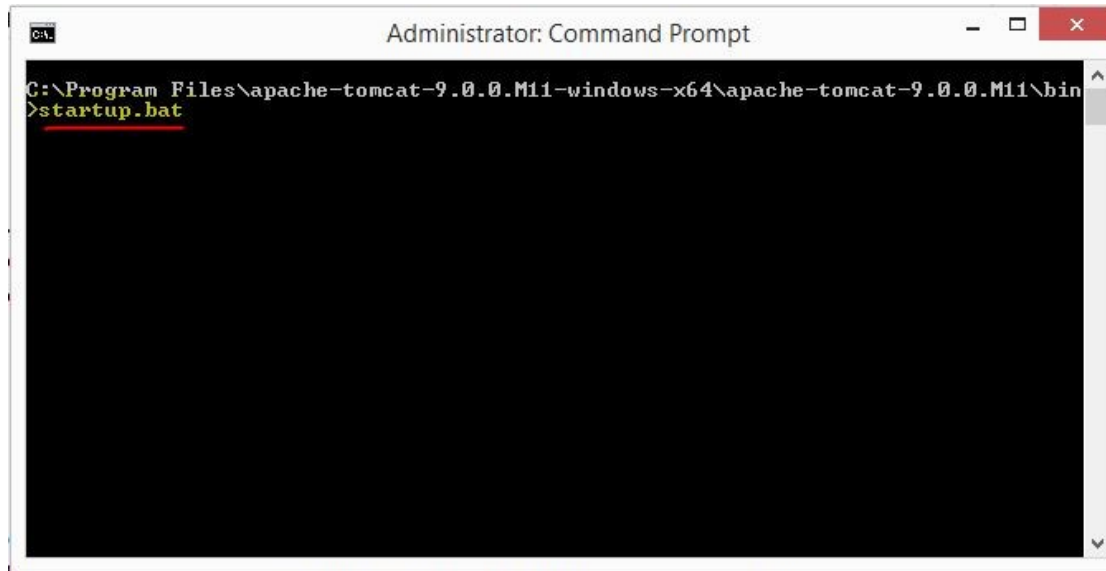
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\karthik\Desktop\UrlShortnerProj>mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building UrlShortnerProj Maven Webapp 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ UrlShortnerProj ---
[INFO] Deleting C:\Users\karthik\Desktop\UrlShortnerProj\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ UrlShortnerProj ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ UrlShortnerProj ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 9 source files to C:\Users\karthik\Desktop\UrlShortnerProj\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ UrlShortnerProj ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\karthik\Desktop\UrlShortnerProj\source\test\java\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ UrlShortnerProj ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ UrlShortnerProj ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-war-plugin:2.3:war (default-war) @ UrlShortnerProj ---
[INFO] Packaging webapp
```

Installation Step2: Once the war file is generated rename the file to ROOT.war

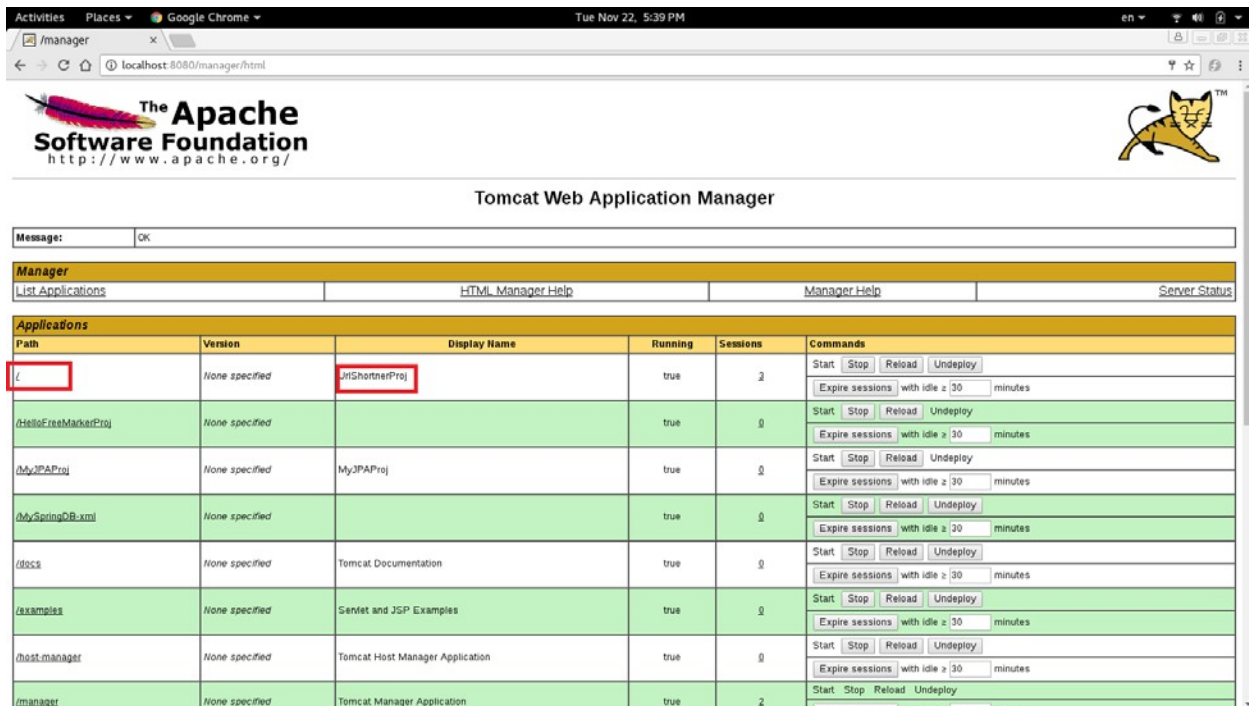


Installation Step3: Launch apache tomcat manager after starting the apache tomcat server



Installation Step4 : If the ROOT.war file is copied in to the tomcat webapps directory . The project will be deployed in the tomcat manager.

Click on the link ["/](#) which defines the ROOT.war file .After Clicking on the link and it should navigate to the home page of the project . Make sure to rename the war file to “ROOT.war” for the application to work with respect to all the functionalities.



Activities Places Google Chrome Tue Nov 22, 5:39 PM

/manager

localhost:8080/manager/html

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

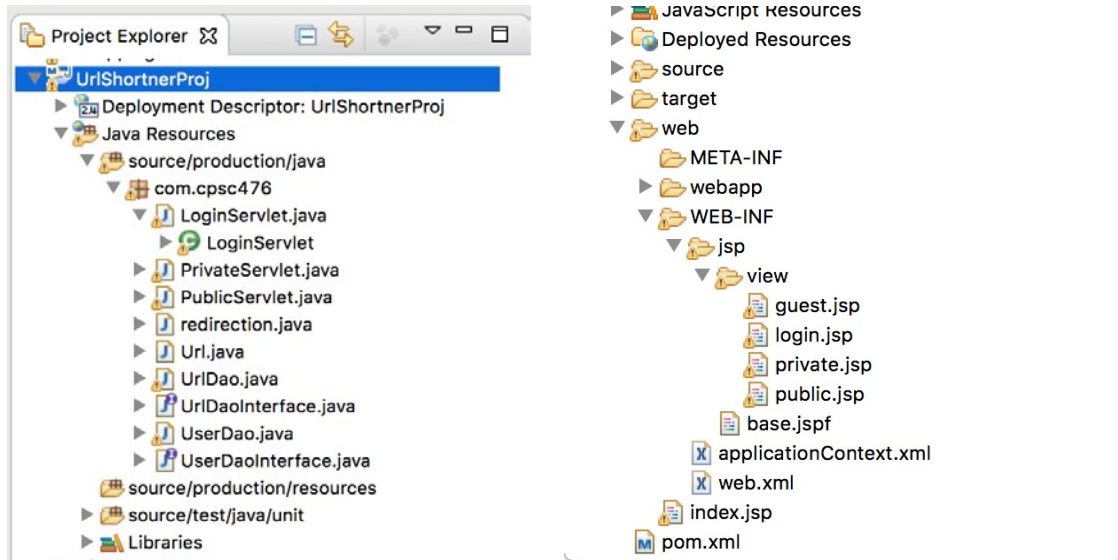
Message: OK

Manager

List Applications HTML Manager Help Manager Help Server Status

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	/ROOT.war	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/HelloFreeMarkerProj	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/MyJPAProj	None specified	MyJPAProj	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/MySpringDB.xml	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	2	Start Stop Reload Undeploy

Architecture of the project:



1. Login and Signup: The login and sign up is built on login.jsp and LoginController.java file
2. Once the user signup or login, he/she is redirected to a private.jsp page
3. The private page where the Short url to long url conversion happens contains the view private.jsp and the controller PrivateController.java. It is mapped to “/private”
4. The public page where the long url to short url conversion happens contains the view public.jsp and guest.jsp which is controlled by the PublicController.java file. guest.jsp contains the actual view mapped to “/public” and public.jsp contains the page for error conditions that occur in the guest.jsp.
5. The user details are stored in a project3.userdetails database.
6. The url's and the clicks for a particular user are stored in project3.url database.
7. There are separate Dao objects for Users and URLs and their respective method definitions are present in UserDao and UrlDao respectively. There are interfaces for the same.
8. Redirection of short url's when entered in address bar of browser is taken care of in RedirectionController.java file.
9. The project is packaged as a maven build with all the dependencies mentioned in pom.xml
10. Spring WebApplicationContext for our web application is configured in applicationContext.xml.

References: <http://docs.oracle.com/javaee/6/api/javax/servlet/package-summary.html> <http://tomcat.apache.org/tomcat-7.0-doc/>
<http://proquest.safaribooksonline.com/book/programming/java/9781118909317>
<https://maven.apache.org/what-is-maven.html> <http://getbootstrap.com/>
<http://hsqldb.org/doc/2.0/guide/index.html>