PROJECT

WEB PROGRAMMING LANGUAGES

FALL 2015

“Online Course Registration”

Group: ‘3-idiots’

1. Venkatesh Avula - vxa141230

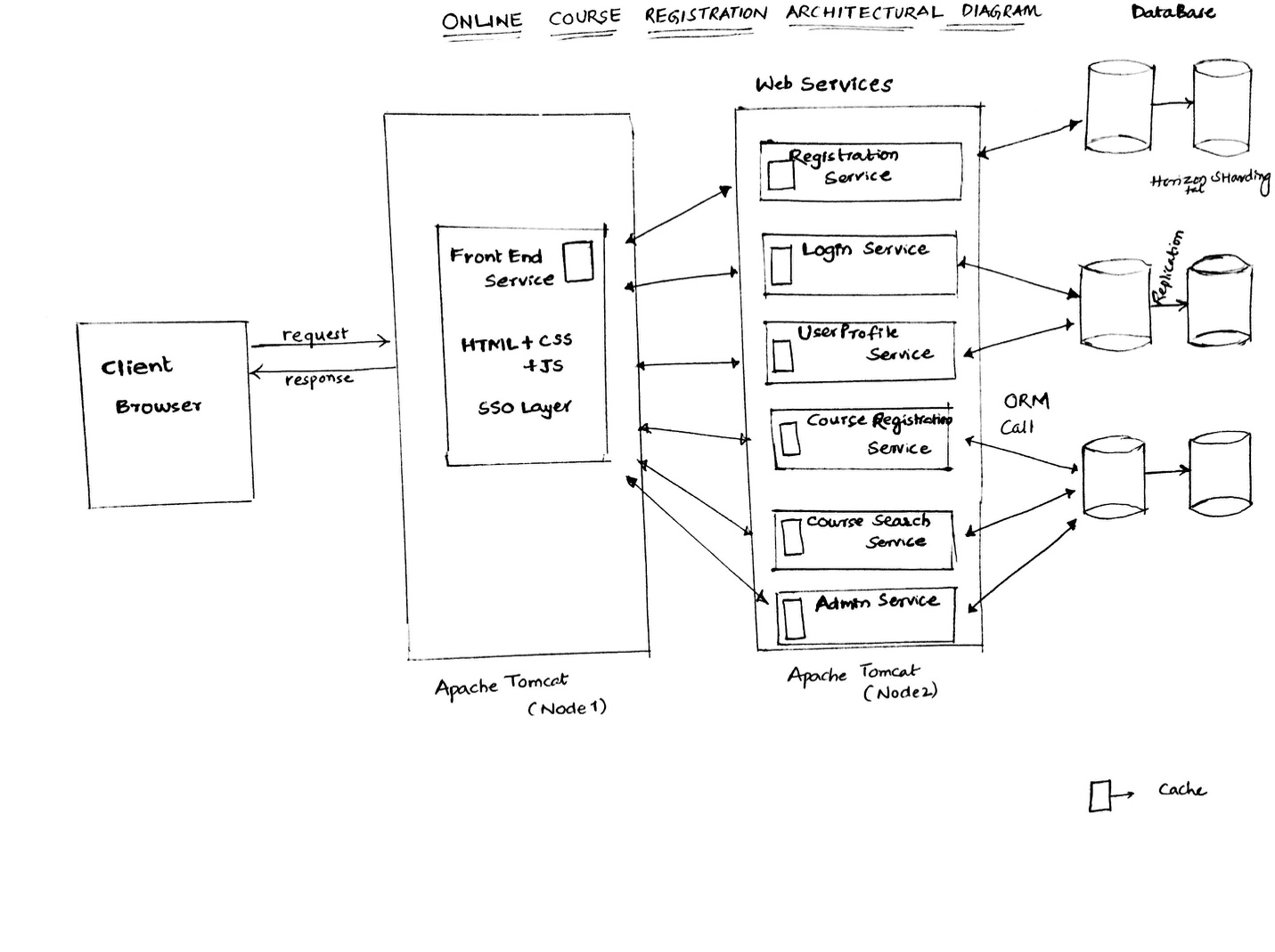
2. Sashidhar Evuru - sxe140630

3. Vamsi Katepalli - vxk142730

**Introduction:** “Online Course Registration System”

**Objective**: The objective of this project is to develop a web application that performs the online course registration. The users in the context of this application have two roles, one as student and other as admin.

Following is the architectural diagram used for the project:



**Website:**

**Description:**

Website is the first point of contact for the user. A user who wishes to register in the website, login or access all the functionalities come to the website by using a browser. Initially, the user lands on the login and registration page. In this page, a user can either register or login if he already has credentials.

The application is hosted on Apache Tomcat. In website, we have segregated the access to API’s and static html by using a mapping in web.xml and by using spring MVC features, we were able to get the request processed for the API URL’s. All the HTML’s are placed in pages folder in WEB-INF directory and the view resolver rule is mentioned in mvc-dispatcher-static-servlet.xml. These static URL’s are processed with keyword static in the URL. Ex: <http://localhost:8080/ocrs/static/site/login> . For the API URL’s, we have a different URL for example: <http://localhost:8080/ocrs/api/login>. All the other code is embedded in HTML’s to achieve the functionality.

Website also requests data from web services by using spring remoting and returns the data to user. Thus website’s main functionality is to redirect the requests of the user to backend and return the response to front End, and JavaScript will try to render the change in the browser. We have also implemented all the functionalities in the web by using Ajax calls.

**Technologies:**

Many technologies have been considered to implement the website functionality including client side scripting frameworks like JQuery and Angular JS. In the webserver, we have tried with technologies like spring, spring MVC, servlets, JSP, gauva, and Rest Client. Finally following are the technologies used in implementing the website:

1. JQuery
2. HTML
3. JSON
4. Spring
5. Spring MVC
6. Jackson JSON
7. Maven
8. REST Client for Java
9. Servlet and JSP’s
10. Spring Remoting
11. Log4J
12. Apache Tomcat
13. Java

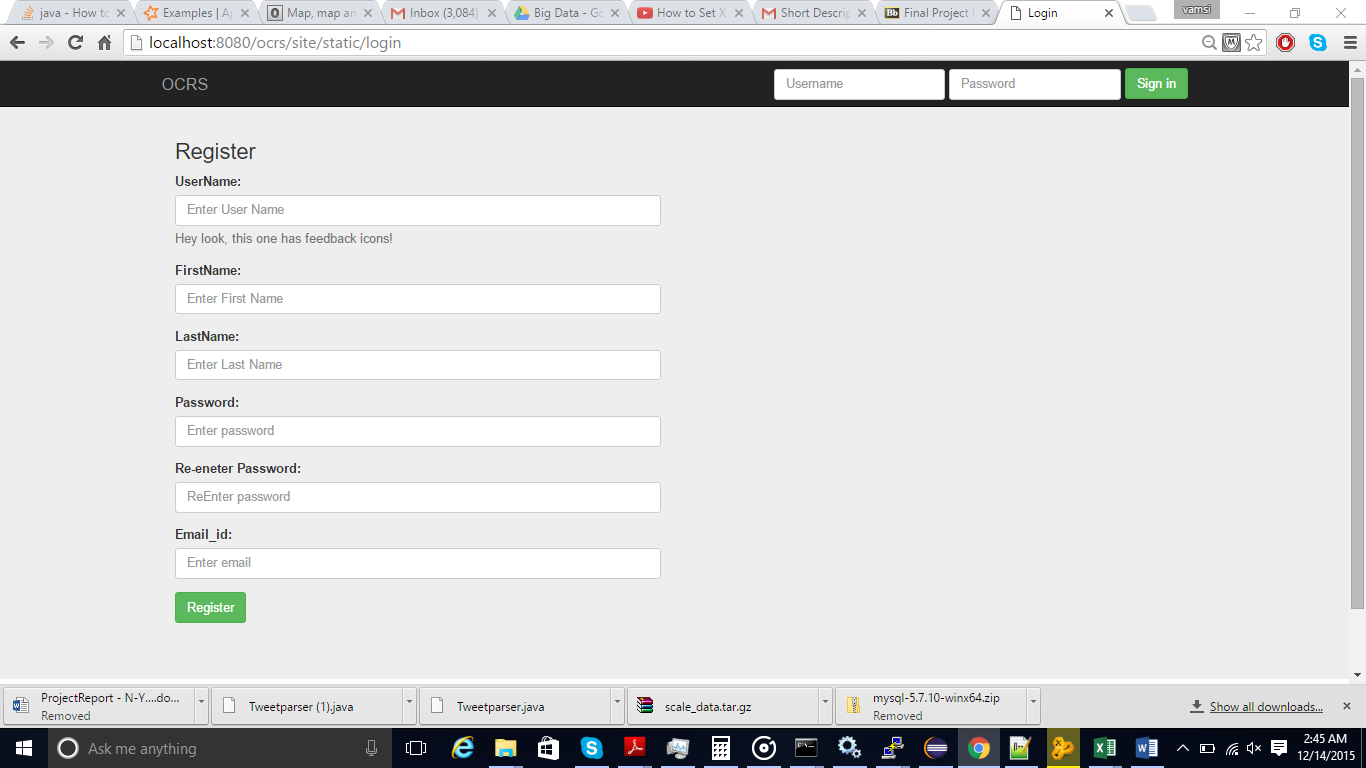
**Features:**

Along with the basic functionality for client side processing, and redirecting the calls to the user, we have also implemented the following required functionalities:

1. **SSL encryption**:
   1. We enabled SSL in Apache Tomcat by using the following configuration:
      1. <Connector SSLEnabled="true" clientAuth="false" keystoreFile="C:\My\_folder\Fall15\WPL\ocrs\keystore" keystorePass="3idiots" maxThreads="150" port="8443" protocol="org.apache.coyote.http11.Http11Protocol" scheme="https" secure="true" sslProtocol="TLS"/>
   2. We used keytool for generating the keystore file by using a keystorepass.
   3. This keystore file is stored in the folder where server.xml can access the file.
   4. After adding the configuration, we can access the website by using https protocol and on port 8443
2. **Compression:**
   1. We used compression for the requests between servers. We also used this for webservices.
   2. Following is the configuration used:
      1. <Connector compressableMimeType="text/html, text/xml, text/plain, text/css, text/javascript, text/json, application/x-javascript, application/javascript, application/json" compression="on" compressionMinSize="2048" connectionTimeout="20000" noCompressionUserAgents="gozilla, traviata" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
3. **Separation of Static and API’s**
   1. Static and dynamic URL’s are separated to achieve good URL’s for the website.
   2. Since website is hosted and accessed using localhost, there is no issue of CORS
4. **Session based access for API’s**
   1. In website every page is protected from access by using a sessoinKey.
   2. Initially when user will login to the website, a session Key is generated by the login service which is saved in the localStorage by the browser. This sessonKey is used to validate and check if user is logged in.
   3. If user logsout of the website, we clear the sessionkey present in the localStorag, and thus access is restricted for the user.
   4. Also**,** user can login using SSO functionality by using his Gmail credentials in to the website.

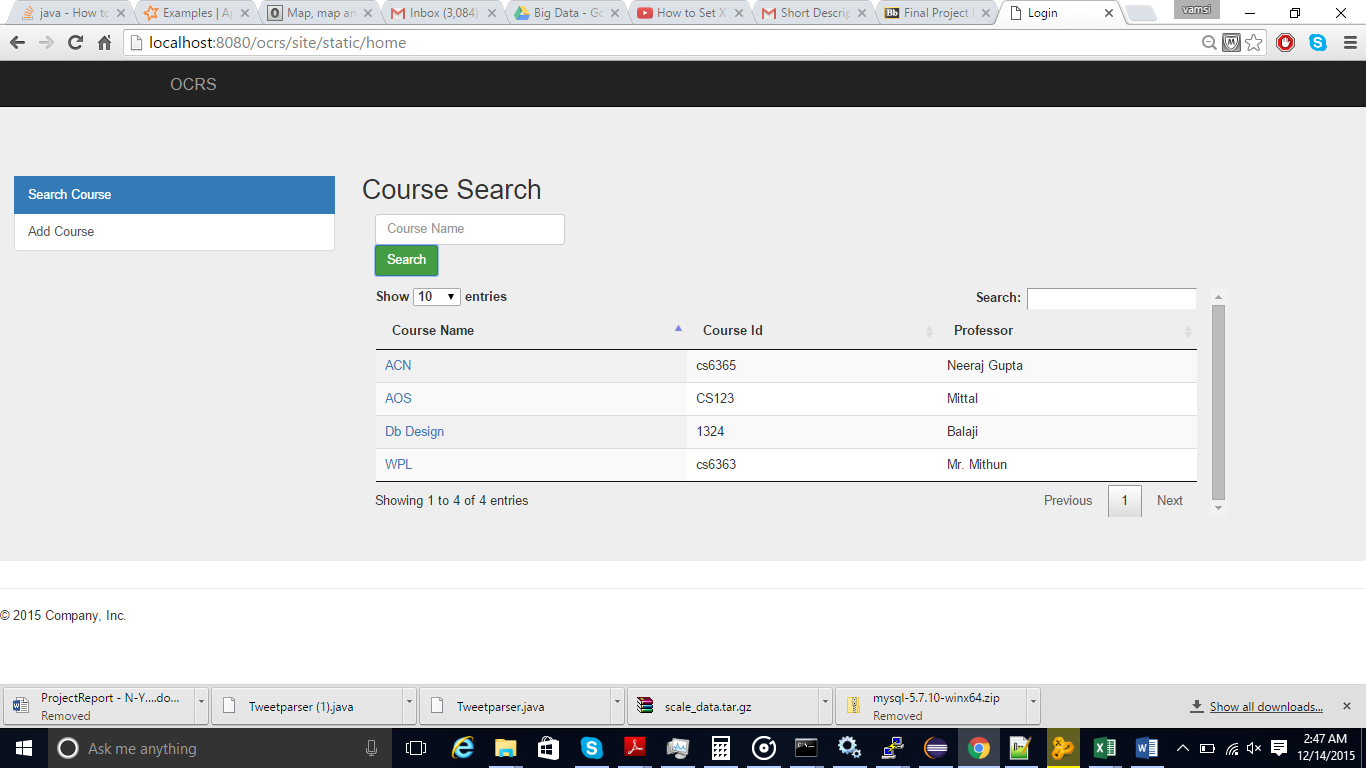
**User Features:**

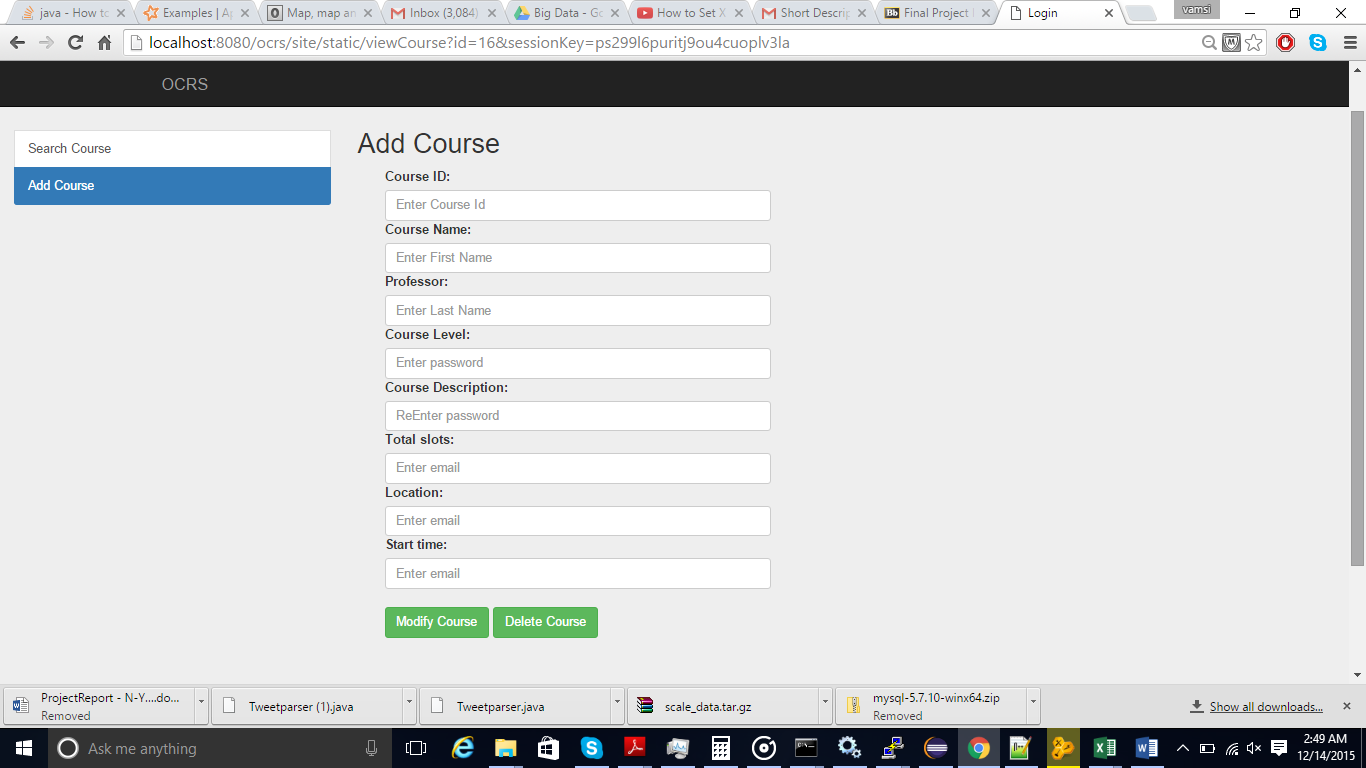
**Login and Registration**: This section explains features from user perspective. Following is the home page of the user once he reaches the application where user can either register or login to the application using his credentials.



**Admin Home:**

In the Admin screen, admin can search for a course and modify the course or drop the course. Following are the screens:





**WebServices:**

Webservices are used to process the information and implement the business functionality of the application. We have developed 5 services which has different functionalities. Website accesses these API’s and this is the only means to access the data. Websrvices access data from Database using hibernate.

**Technologies:**

We have tried the same set of technologies for server side similar to website. We considered mainly Spring framework as it has more documentation and Java based framework. Following are the technologies used on the server side:

1. Spring
2. Spring MVC
3. Jackson JSON
4. Maven
5. REST Client for Java
6. Servlet and JSP’s
7. Spring Remoting
8. Log4J
9. Apache Tomcat
10. Java

**Features:**

In this section we explain in detail the functionality of each web service and business logic they process:

1. **LoginService:**

LoginService is used to login the user in to the application by using the user credentials ie username and password and if the credentials are valid, it creates a sessionKey for the user which is random and returns this key to the user. Login interacts with the database and stores the password using the hash.

* 1. Login service authenticates the user using username and password.
  2. Generates sessionKey.
  3. The sesionKeys generated in loginservice are stored in a map which is a spring bean. This is required because when we receive the subsequent requests for other services, we check if the sessionkey received is present in session and get the user information. If the sessionKey is not valid, we return an error to the user mentioning that the session is not valid and in website, user is redirected to login page.
  4. It also has the functionality to logout the user from the application. In this Scenario, we remove the entry in map stored in the session.
  5. We have also added an Interceptor to so that requests to login and registration does not need a sessionKey and all remaining URL’s need sessionKey and we validate the sessionKey. If it is valid we allow the call else return with error.

1. **Registration Service:**

Registration service is used to register the user in the application based on the data entered by the user. Some of the fields include username, password, Email.

1. **Course Search Service:**

Course Search Service is the main service for searching a course for a user and also some admin related services. It has many API’s as mention below:

* 1. List Courses:
     1. In this API, we fetch all Courses in the Database and return to user. This is used in listing courses for a user.
     2. URL: <https://localhost:8080/CourseRegistrationService/course/list>
  2. Search Courses:
     1. In this API, we search for a course based on the keyword in the input. The Keywords correspond to the courseName in the course Object. This is used in the Course Search page in website.
     2. <https://localhost:8080/CourseRegistrationService/course/search>
  3. Add Course:
     1. This API will be used to add a course. This is mainly used by administrators ie users with Admin roles.
     2. User can add the course based on different fields like Course id, professor etc.
     3. <https://localhost:8080/CourseRegistrationService/course/add>
  4. Remove Corse:
     1. This API will be used to remove a course.
     2. <https://localhost:8080/CourseRegistrationService/course/remove>
  5. getCourse:
     1. This API is used to get a specific course for a user based on courseId ex: 6363
     2. <https://localhost:8080/CourseRegistrationService/course/getCourse>
  6. updateCourse:
     1. This API is used to update a course. This is mainly used by Admin users.
     2. <https://localhost:8080/CourseRegistrationService/course/updateCourse>

1. **Course Registration Service:**

Course Registration service is used to register a course for a user. Following are the API’s

* 1. Register course:
     1. Registers a course for a user
     2. <https://localhost:8080/CourseRegistrationService/registerCourse/add>
  2. Unregister Course:
     1. Unregister a course
     2. <https://localhost:8080/CourseRegistrationService/registerCourse/remove>
  3. getRegCourse:
     1. This API is used to get registered courses

<https://localhost:8080/CourseRegistrationService/course/getRegCourse>

1. **Memcached:**

We have used memcache to implement distributed caching in our application. Course related information for search API’s are stored in cache.

1. Extra Features Implemented :
   1. **Hibernate ORM** :

We have used HIBERNATE - ORM for back-end operations. We have created Java POJOS and data base mappings and used hibernate queries for various operations.

* 1. We have implemented **SSO authentication** by using gmail api. In this module, we have used user profile information provided by gmail to register into our site. Once user login to site with his/her gmail credentials, user will be redirected to home page. If the user trying to login for the first time we have created user in the backed. For this we have created a profile in google developer tools to get the client id.

**Problems Encountered:**

1. During memcache implemnentation, we were having issue with accessing the resource from cache, and after making the class serializable, we were able to get the data.
2. We tried to implement Data Sharding in the application but there was no sufficient documentation for MySQL server over the web. We also tried for other databases but could not get any information.
3. During SSO implementation we have face some issues like ,For the first time google link is not being loaded.
4. We have used jquery table to display course details. There is some issue with the data table because of that some time java script error is being thrown.

References:

1. Lecture Notes by Dr. Mithun Balakrishnan
2. <http://memcached.org/>
3. <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>
4. <https://jquery.com/>
5. <http://www.w3schools.com/jquery/>