**Jenkins API Automation Dashboard**

1. **Overview ------------------------------------------------------------------- 3**
2. **Dashboard Understanding -------------------------------------------- 4**
   1. **Login Page**
   2. **Home Page**
   3. **JobCreation Page**
   4. **JobRetrive Page**

**2.4.1 Fetch Job Details**

**2.4.2 Display Console Output**

**2.4.3 Display Build Job**

**2.4.4 Delete Job**

**2.4.5 Build Count**

1. **How to start the JenkinsService API ---------------------------------- 14**
2. **Code Explanation --------------------------------------------------------- 15**

**4.1.1 Index.jsp**

**4.1.2 Login.jsp**

**4.1.3 Home.jsp**

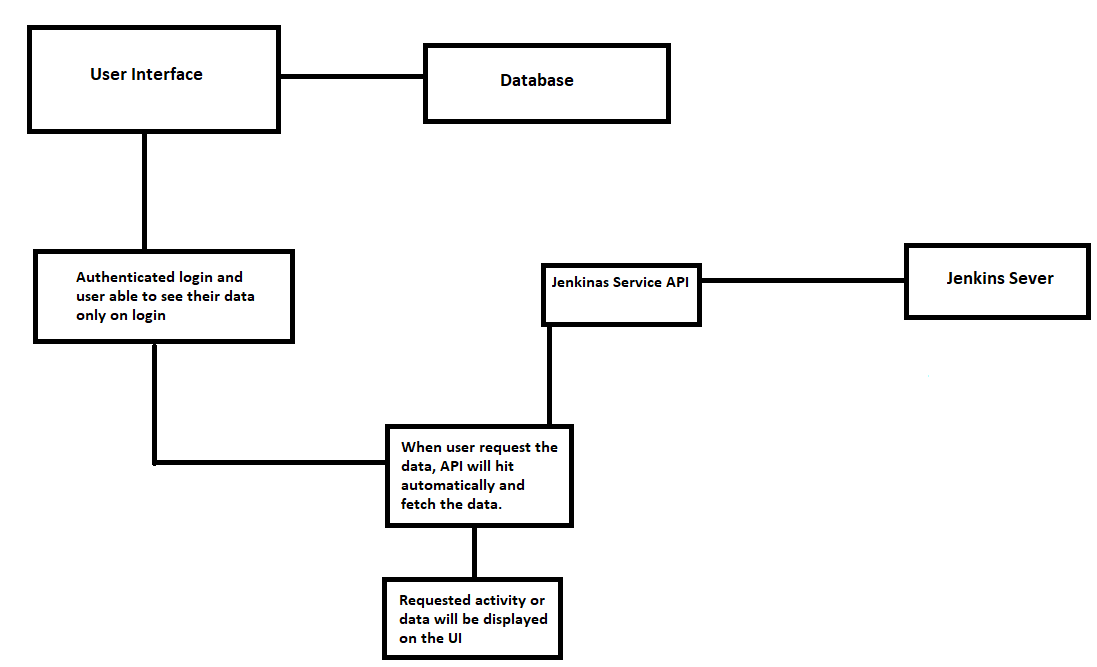
**4.1.4 Buildjob.jsp**

**4.1.5 Jobretrive.jsp**

**4.1.6 Main.js**

1. **Overview:**

* Basically the main purpose of the dashboard is to automate as much work as we are doing on Jenkins. For example, creating Job, adding configuration, deleting job, showing the Job details and many more.
* To provide this functionality we crated one dashboard for user from where user can perform this activity by just one click. Each and every user provided with user id and password which were stored in database and also each user can see the job which is created by him/her and perform necessary actions.
* To perform all this activity, we have created one Jenkins service API which will fetch the data from Jenkins server and display it on the dashboard.
* Basic workflow of the system is displayed below to understand more.

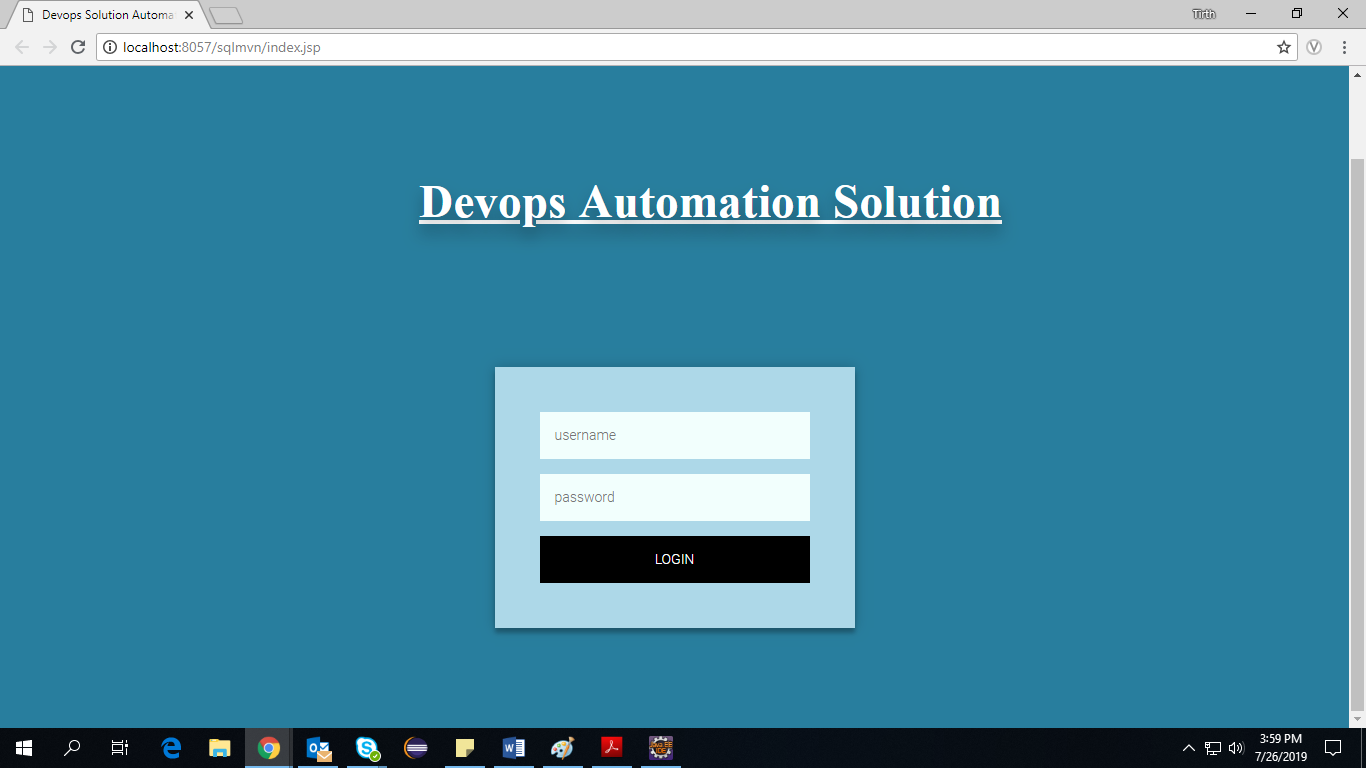
****

1. **Dashboard Understanding:**

**So basically here I am going to brief about each and every feature of the dashboard.**

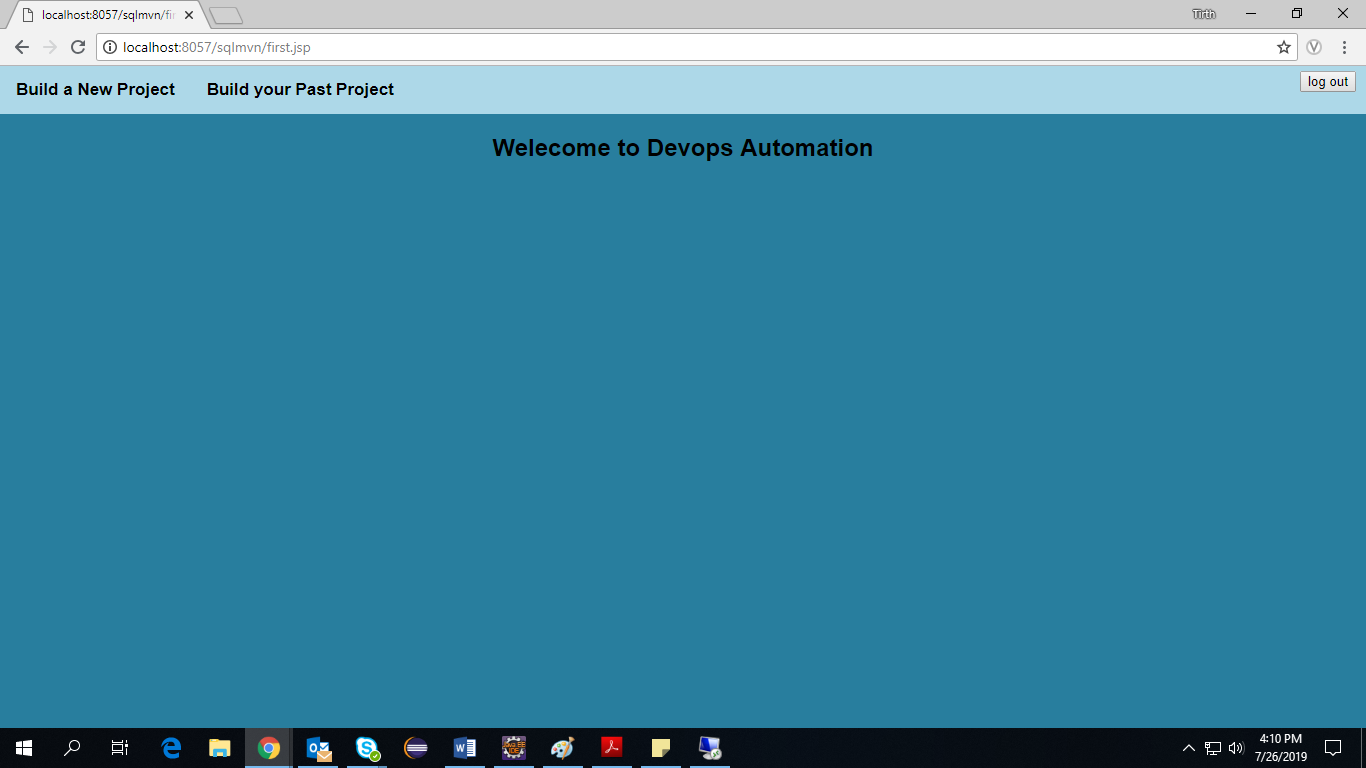
**Below all pages are created using HTML, CSS, Java Script and Jsp.**

**2.1 Login Page**

****

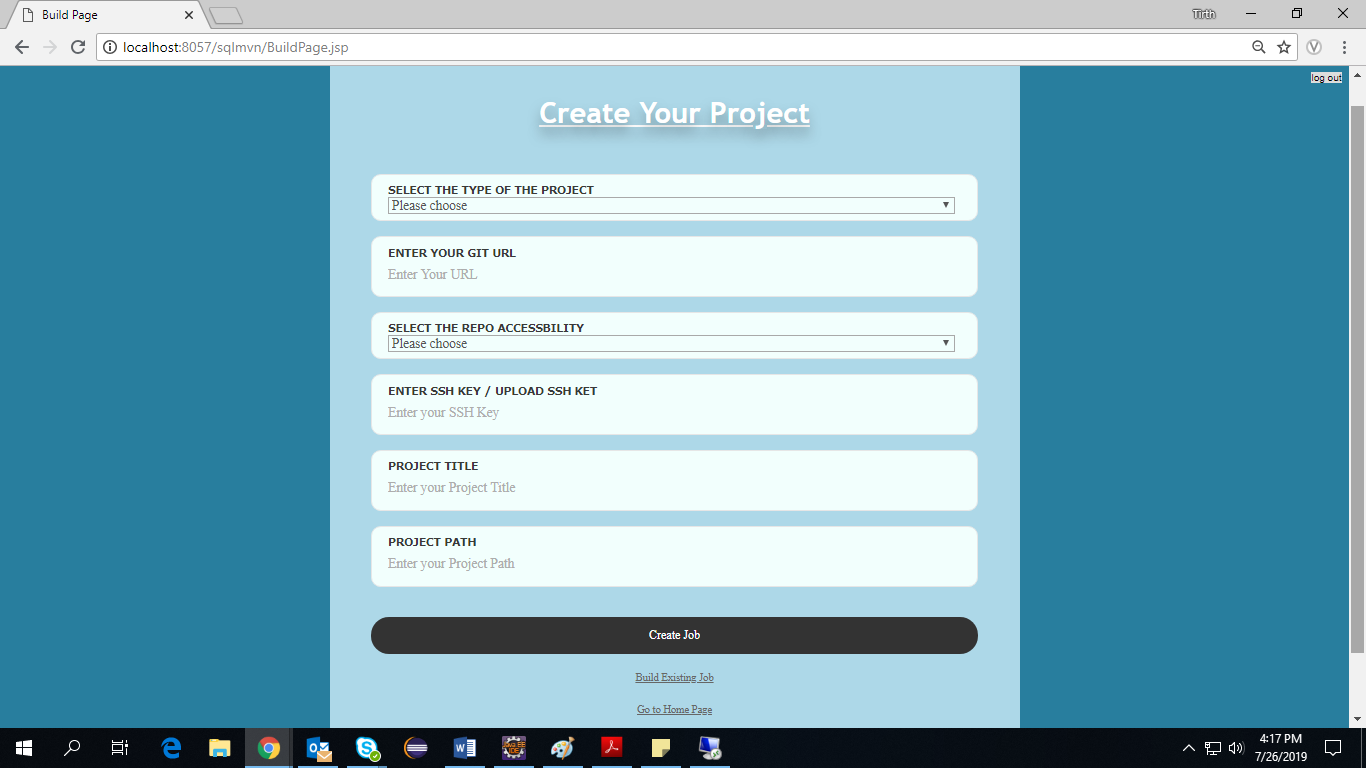
* This is the first page of the Dashboard. Through which user are able to login in the dashboard and the User Id and password provided to the user. User id and password will be validated from the database and if both are correct then it will do successful login otherwise it will give error message.

**2.2 Home Page :**



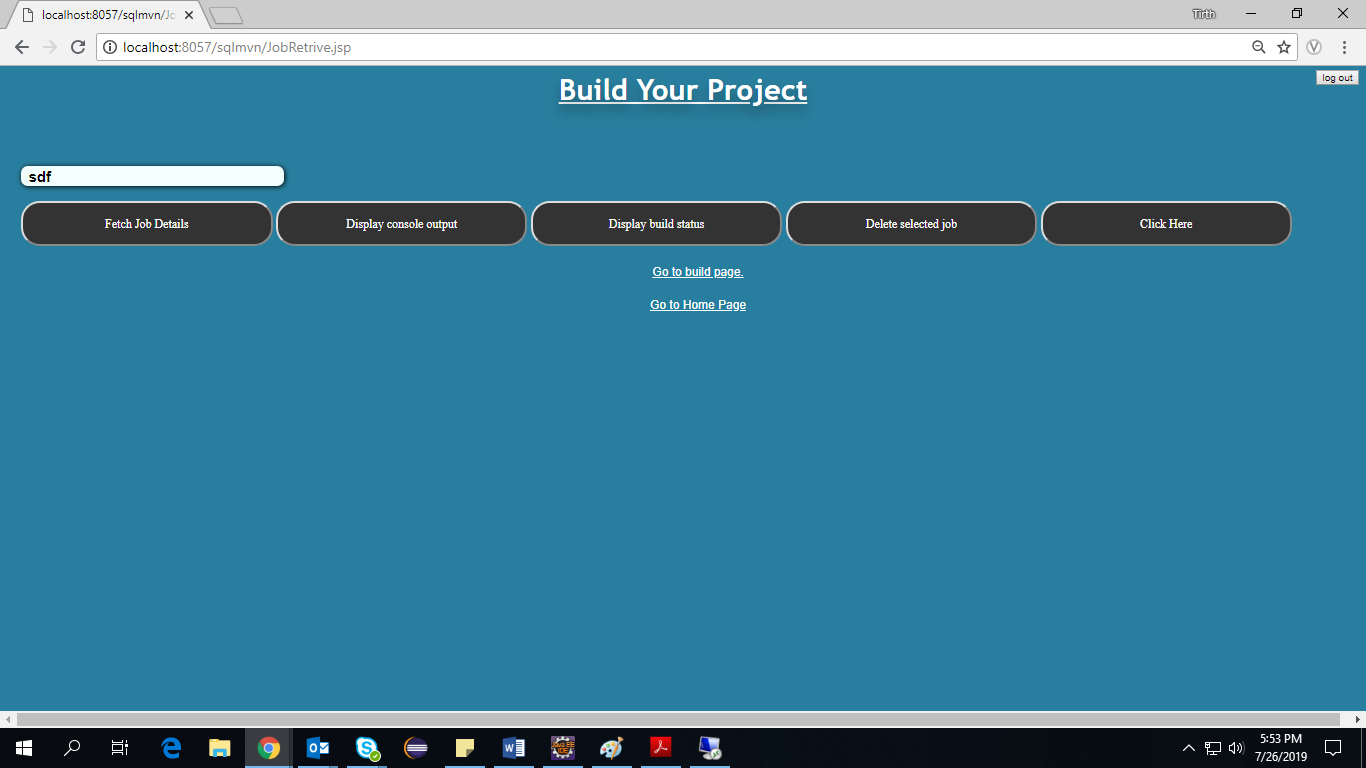
* Once you login into your account you will able to this home page. Here I added two functionalities. If you want to create a new Job the you can click on the first tab. If you want to perform actions on existing job, then you can click on the second tab.

**2.3 Job Creation Page**



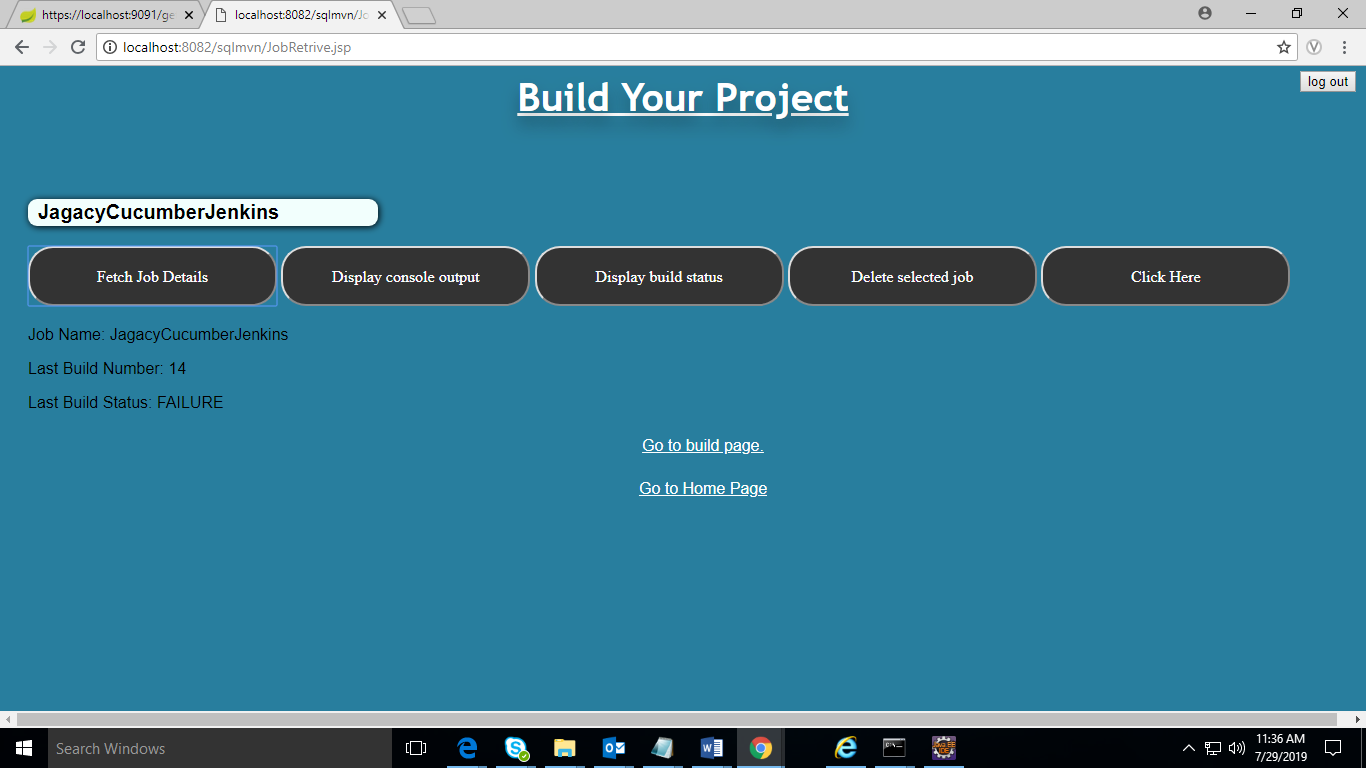
* This is the create project page in which it is asking for few of the field to create the project like type of project (Maven, Python etc.), GIT URL from where you have to fetch the project, GIT repo type like private or public and if it is private then it will ask for the SSH key and if its public there is no need for SSH key, project title and project path.
* And once you click on the create project button it will create your project and it will be reflected in the second tab in which you can perform actions on created job. For example, building a job, displaying output etc.

**2.4 JobRetrive Page:**

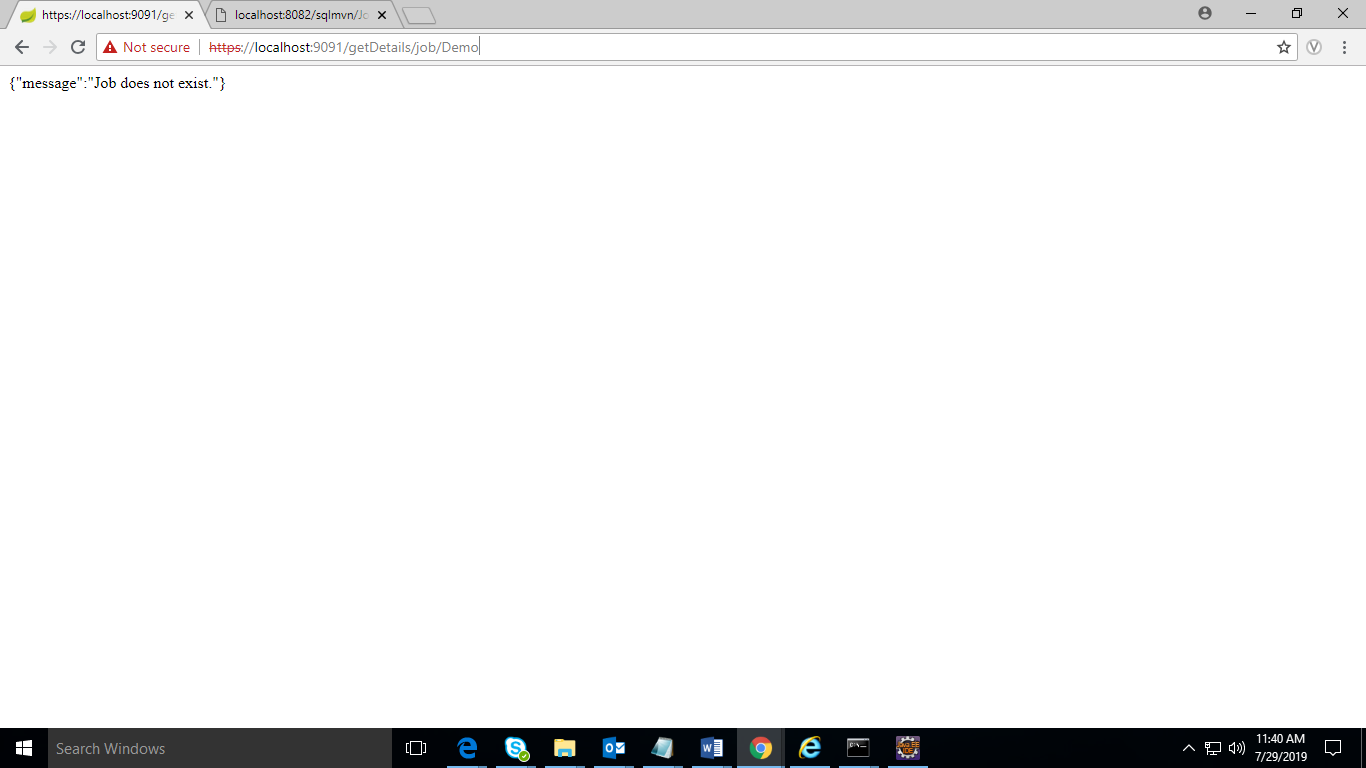


* Above one is job retrieve page in which it showing the dropdown of the jobs which you have created and which are in your Jenkins dashboard. From the dropdown you can select any job and perform the below actions on it. First feature is the fetch job detail which will display the details of the job.
* Here when you select a particular Job and click on the “Fetch Job Details” button immediately it will hit the API and at the same time api will fetch the details from Jenkins server and diplay in your GUI.

2.4.1 Fetch Job Details

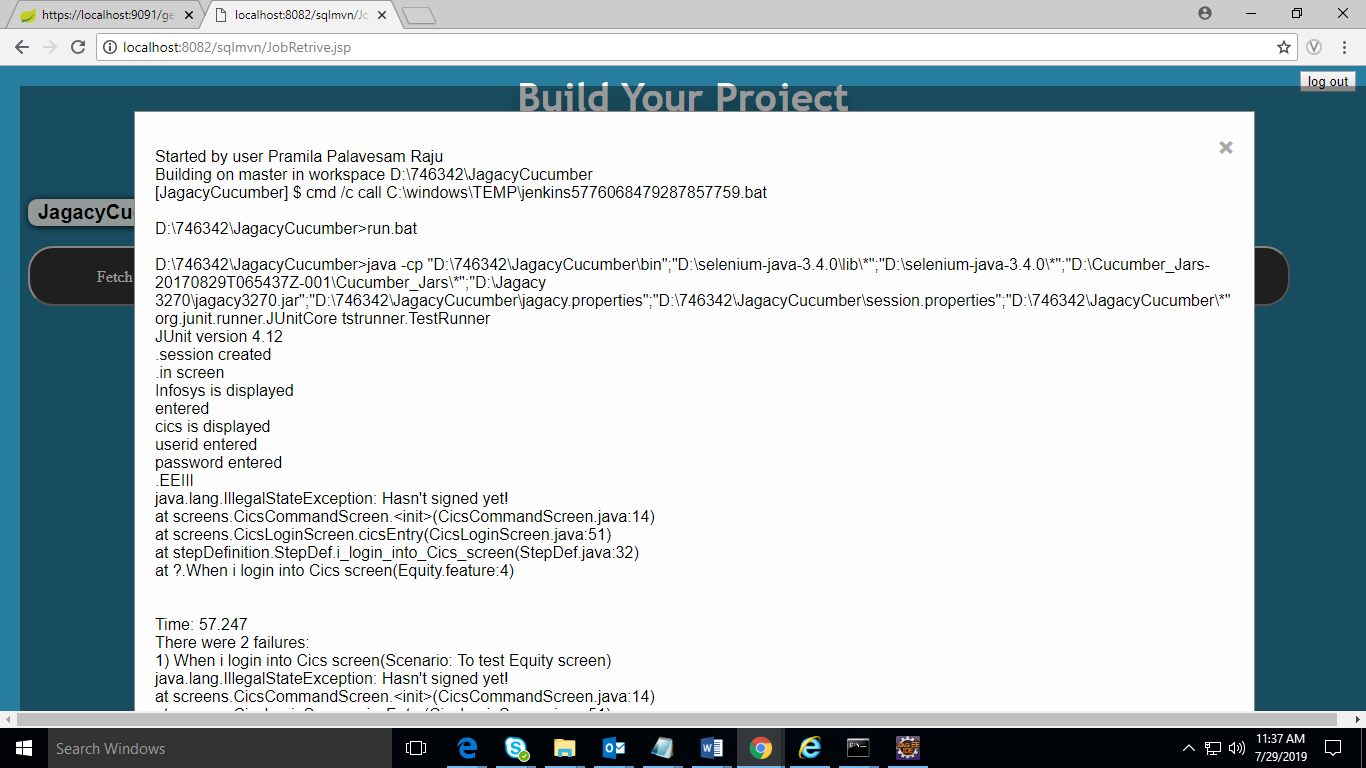


* API will fetch the details in the JSON format. Below image show the fetch details by API in JSON format.

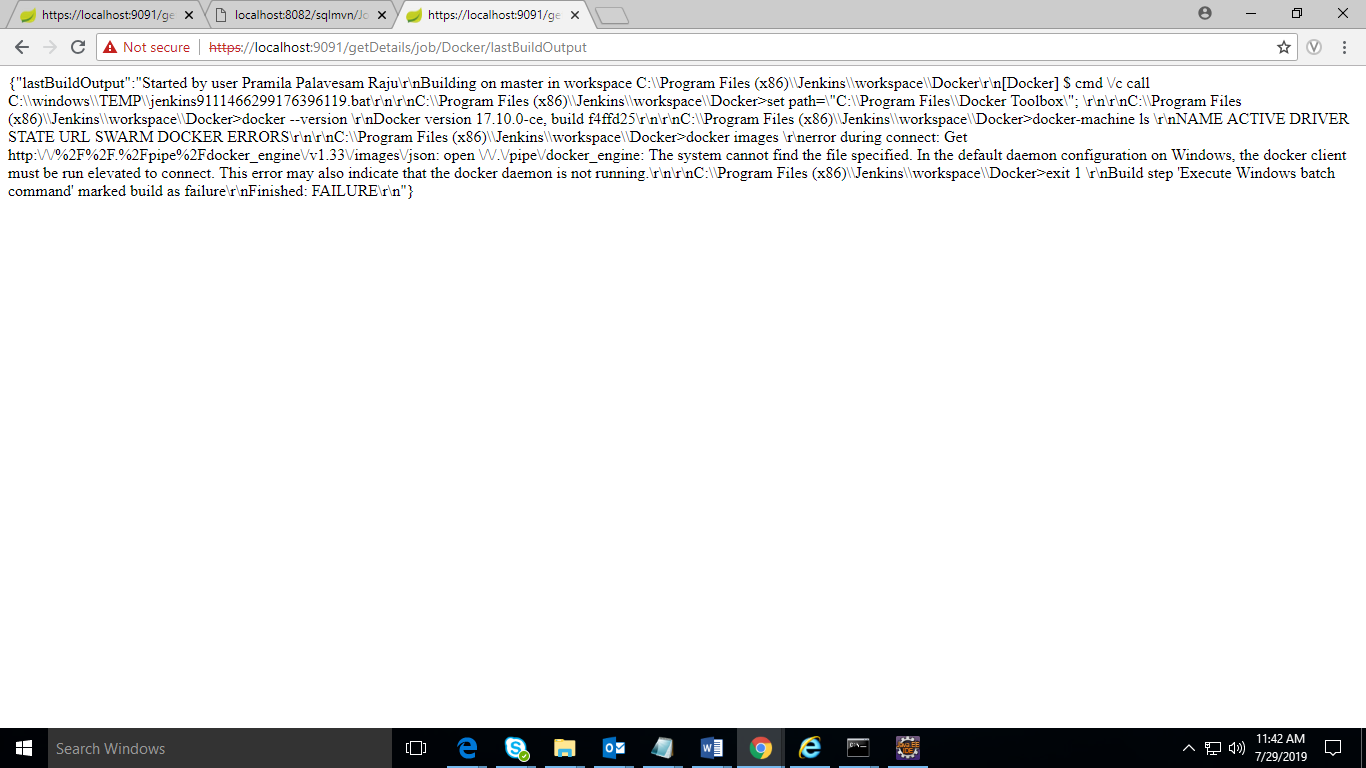


* Now the second feature to “Display Console Output” which is use to show the console output of the last build of Job. So we can see the failure reason or execution of the Job. Once you click on this button it will pop up the window in which you can see the console output.

2.4.2 Display Console Output

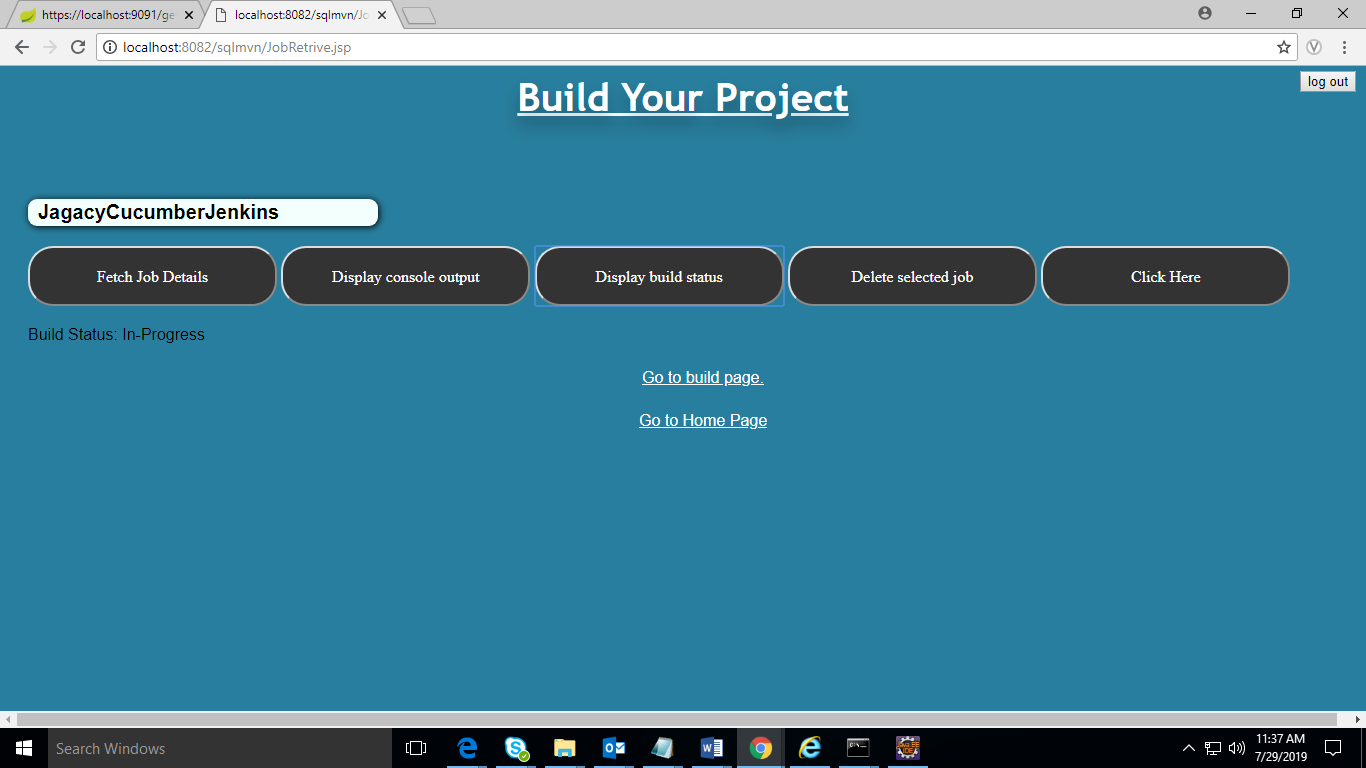


* API will fetch this details in JSON format as below.



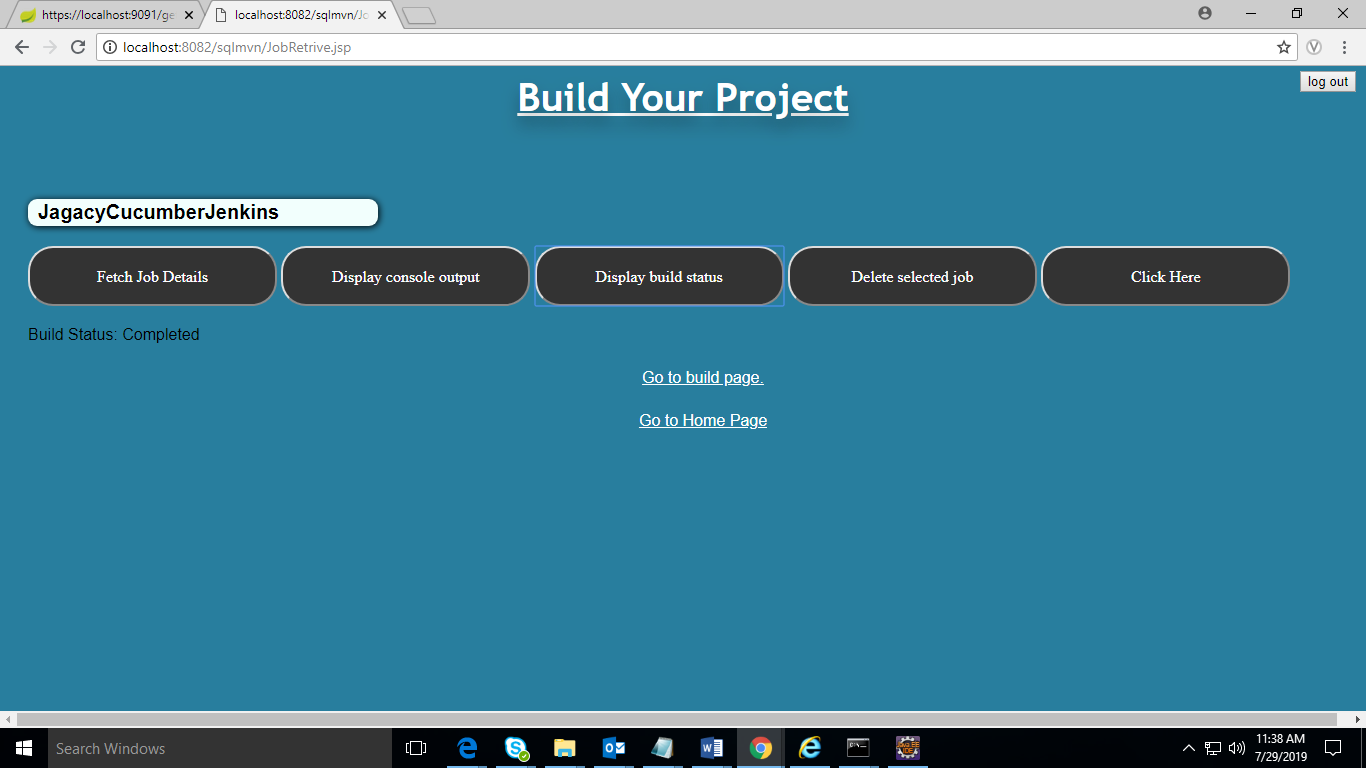
* Third feature is “Display Build Status”. When you select the Job and click on this button it will start the building process of Job in background. In that period it will show the status as in-progess. When Job building is done it shows the status as complete in spite of it failed or sucessfull.

2.4.3 Display Build Status

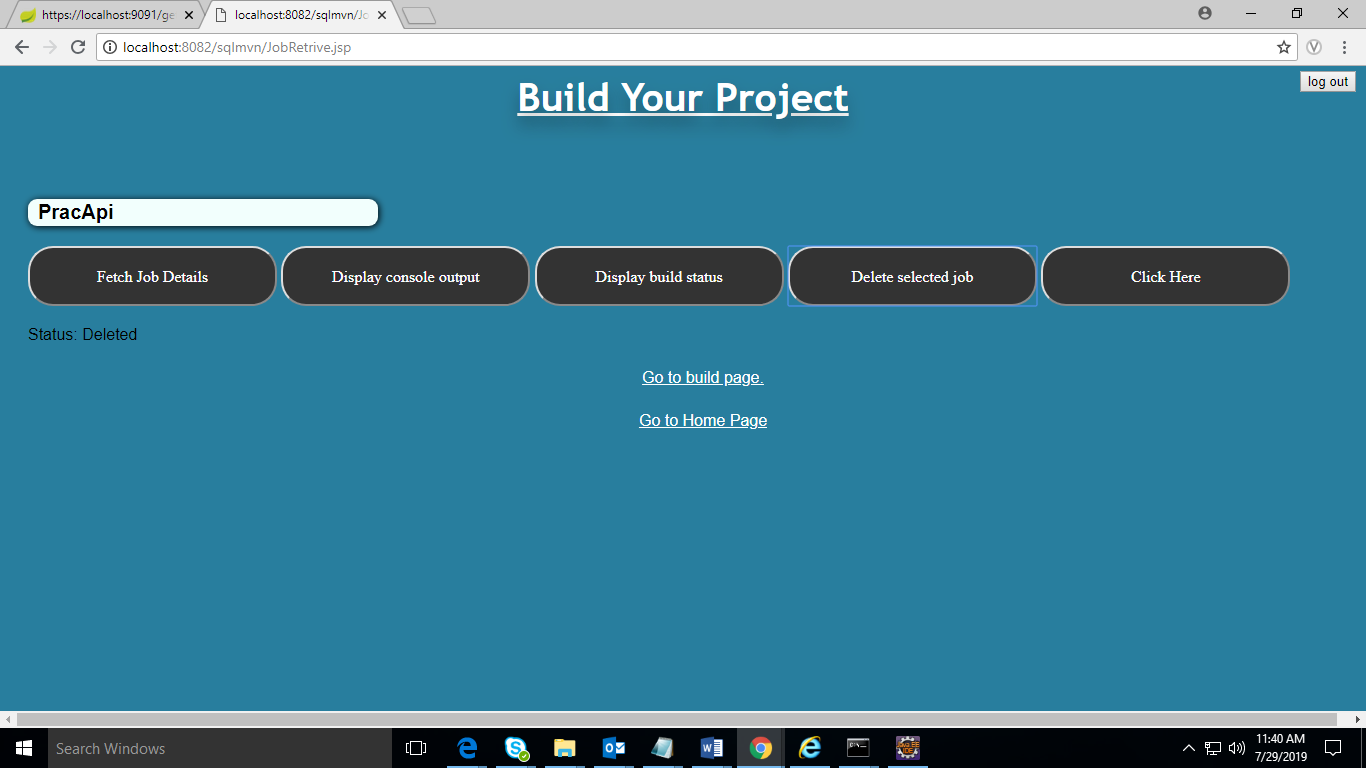


* You can see that your job is building in Jenkins when you press Display Job Build button. Once Job is completed it showing the below result.

2.4.4 Delete Selected Job

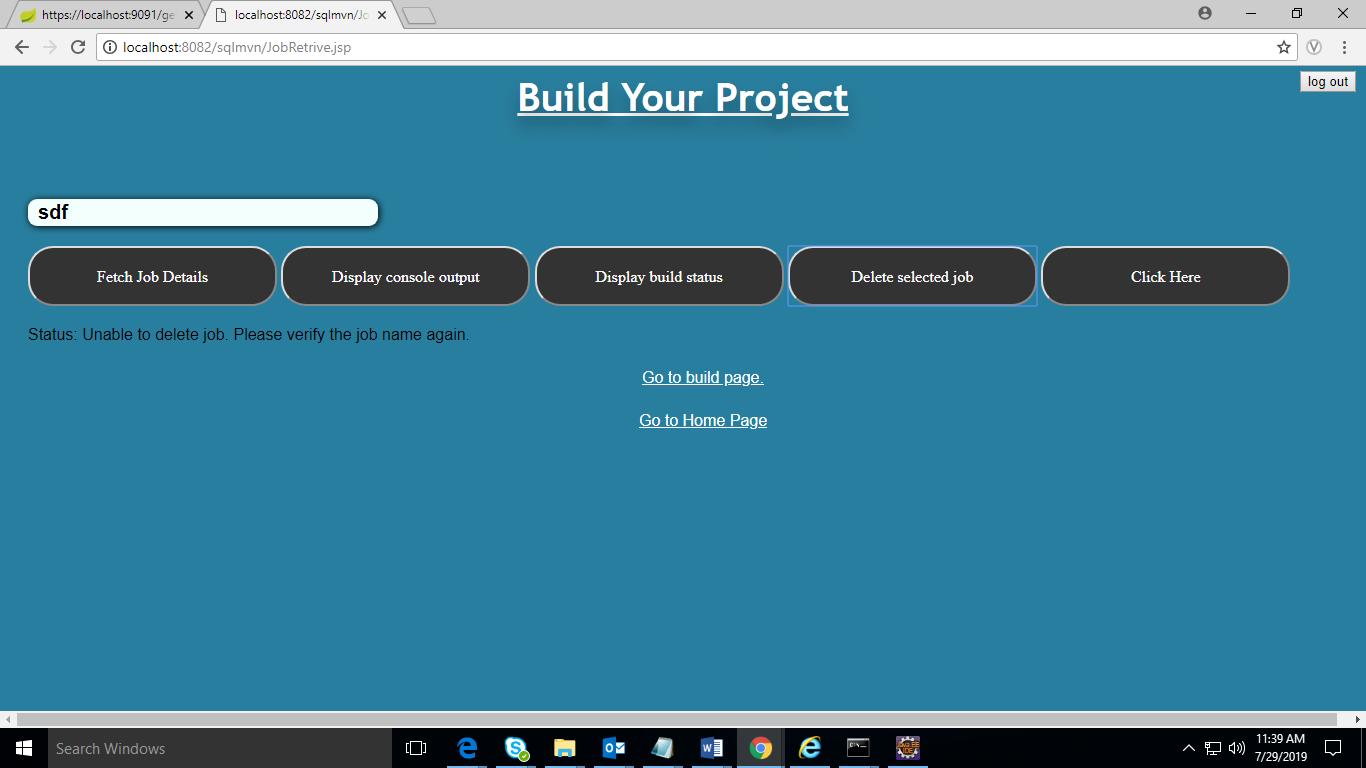


* Next feature is “Delete Selected Job”. Once you select particular job from drop down and click on the Delete Selected Job button again it will hit the API and delete the Job from Jenkins server permanently same as you delete job from Jenkins but more easy away through this dashboard.



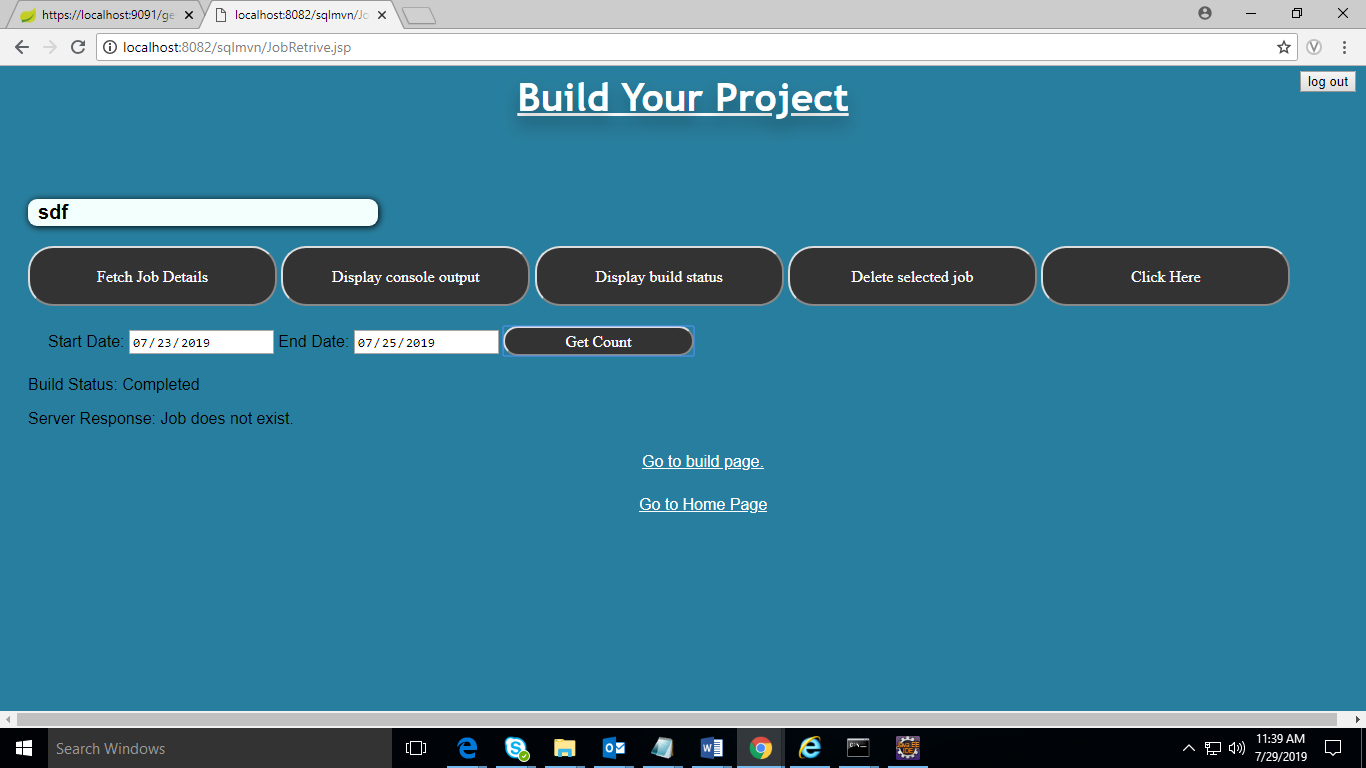
* Once the job is delete it will show the result as Deleted. And if you select the Job from drop down which is not present in the Jenkins server and then also you click on the Delete this Job then it will show the bellow output.

2.4.5 Build Count



* The next feature is to get the count of the number of builds then you can use this. Once you select the particular job and click on Click Here button it will display with Start date and End date in which you have to enter your input based on your requirement.

Once you gave the input of the dates and click on the Get count button it will give the number of count of builds that happens in that particular time.

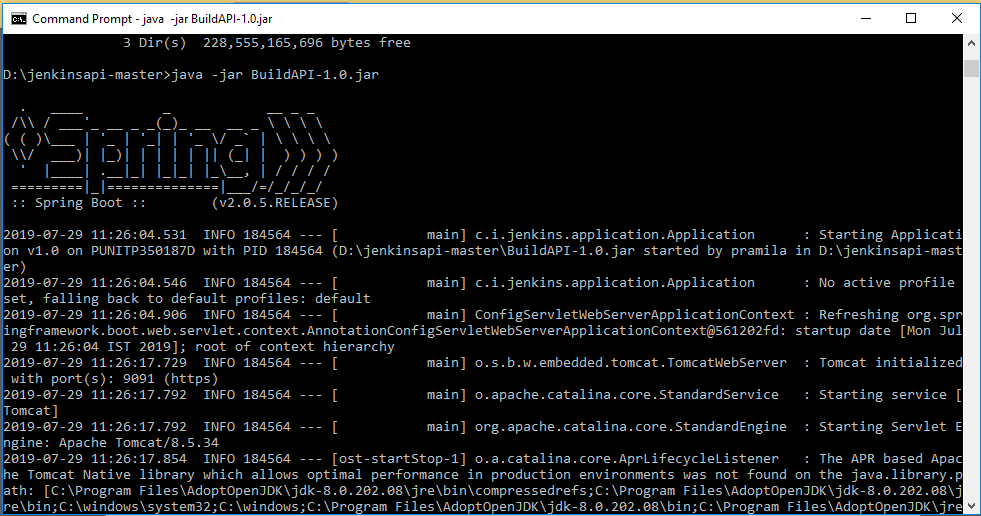


And from any page you can go to any page and also logout your account from any page.

1. **How to start the JenkinsServiceAPI:**

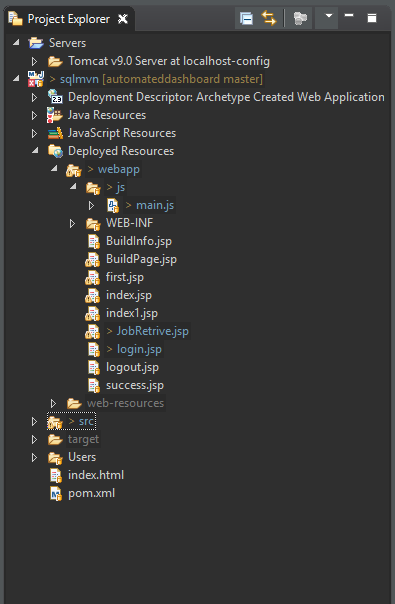
* When you run the Jar file of Service API it will start the service API. Until you start API you cant use any of the feature of dashboard.
* Command to run the Jar file is
* Go to the location where your Jar is present and then run below command

**Java - jar Build-API-1.0.jar**

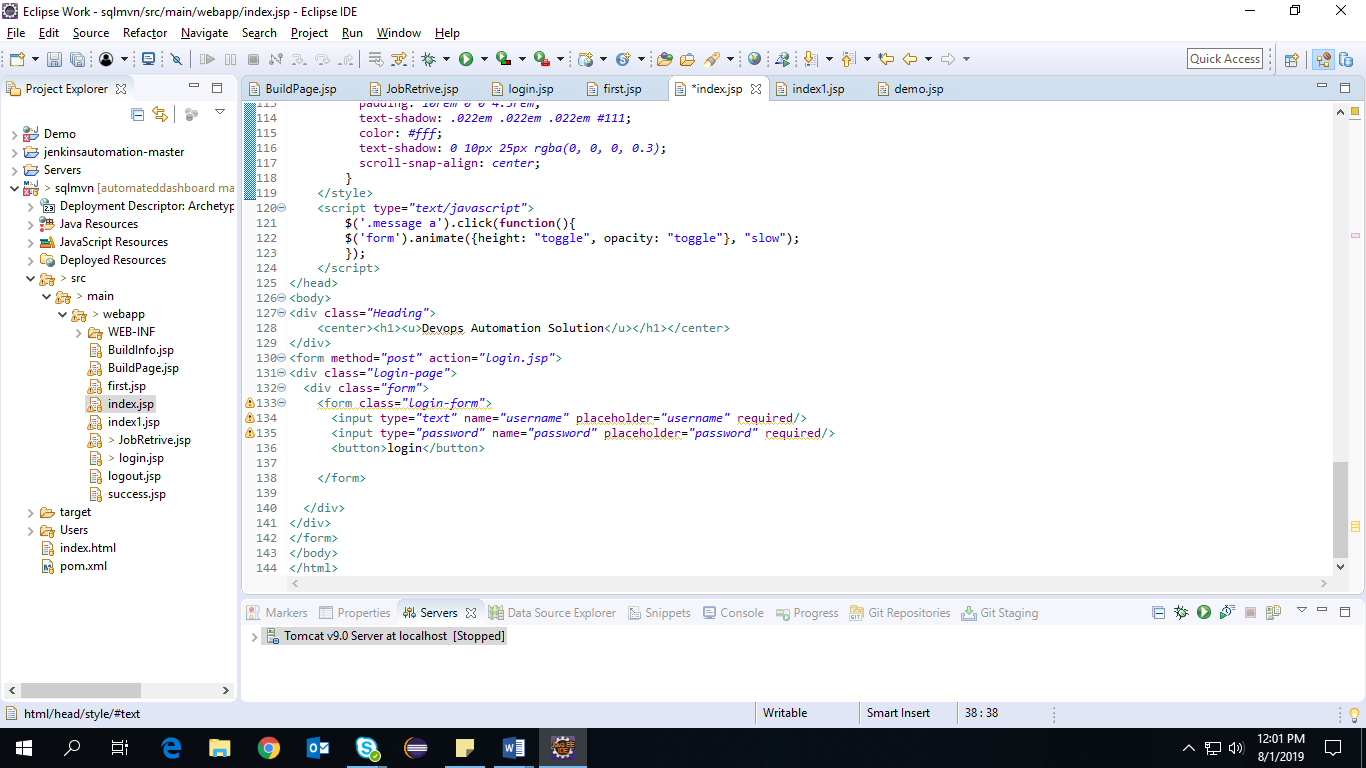


1. **Code Explanations:-**

Structure:

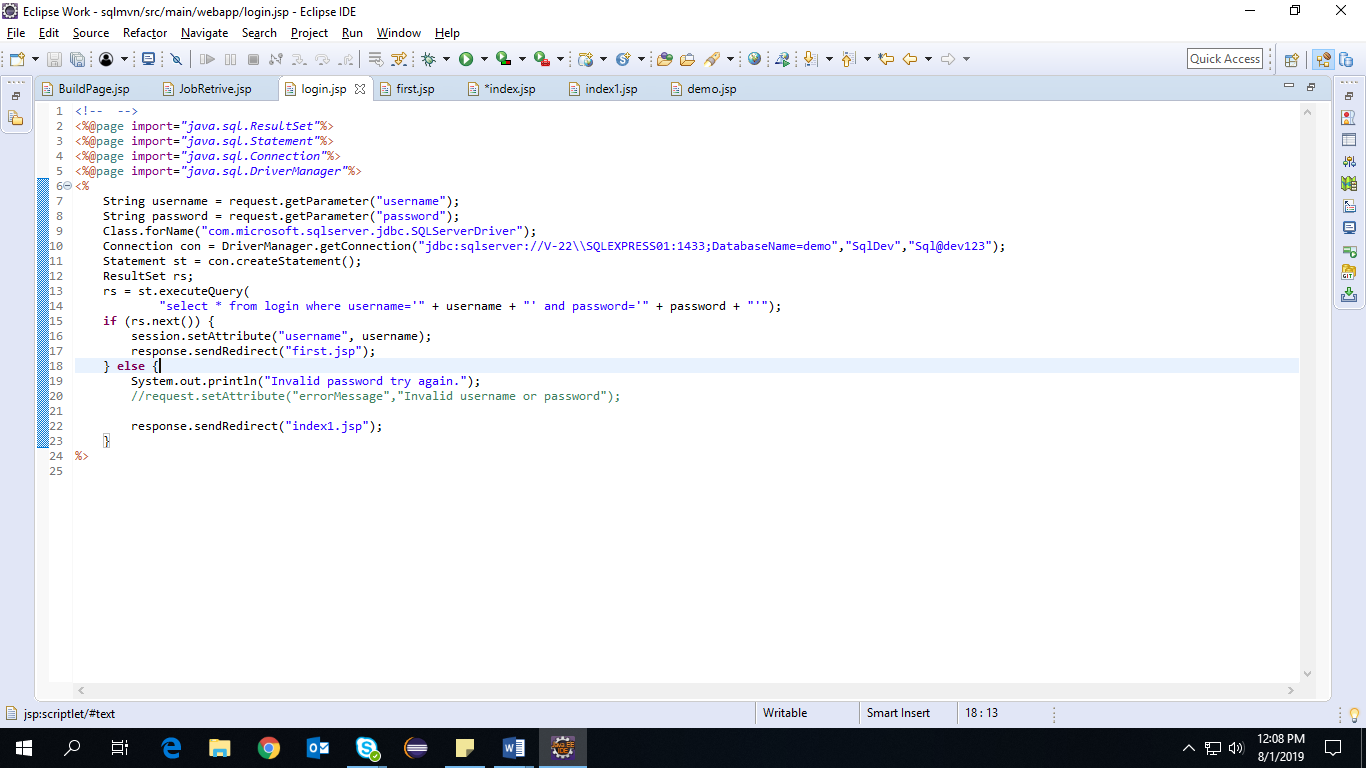


4.1.1 Index.jsp



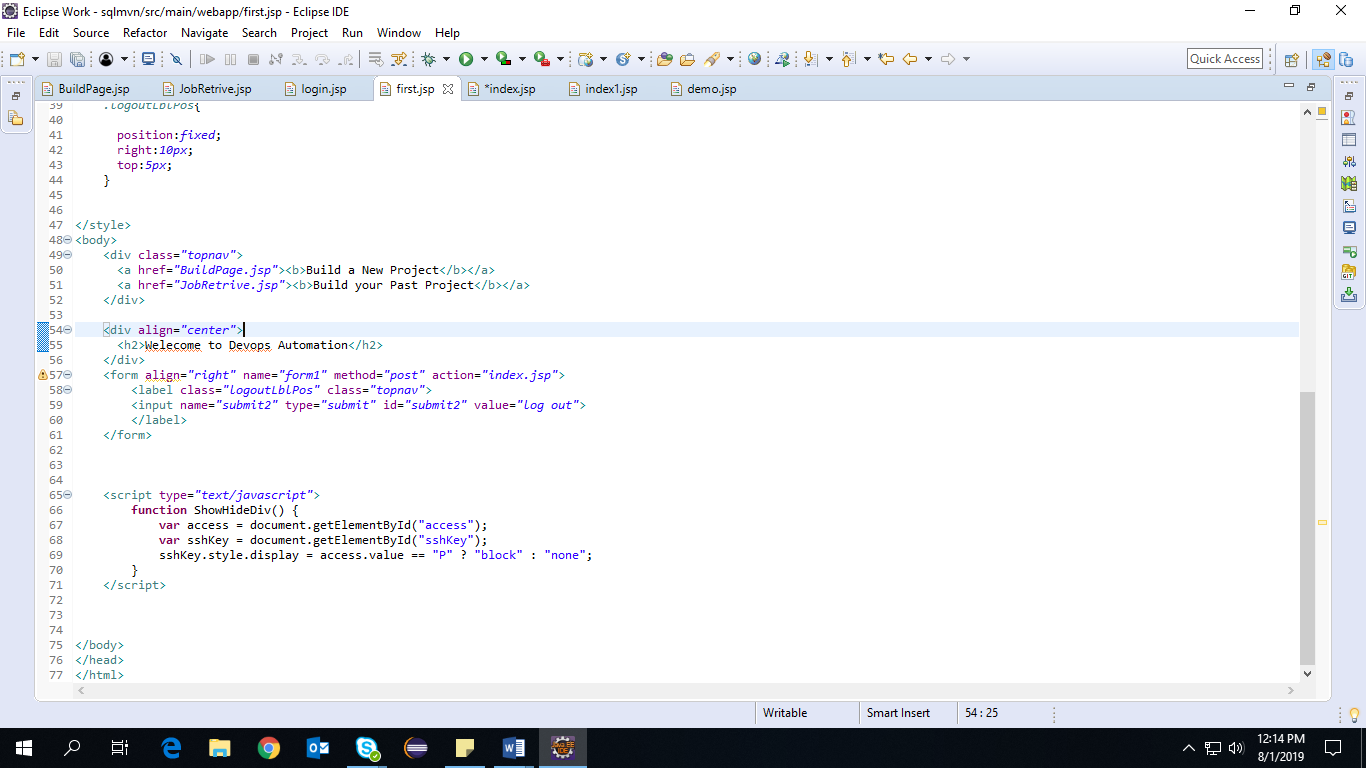
* Here it’s showing the first page “index.jsp” page which call the the “login.jsp” page. This “index.jsp” page mostly include html, css and Javascript part.

4.1.2 Login.jsp



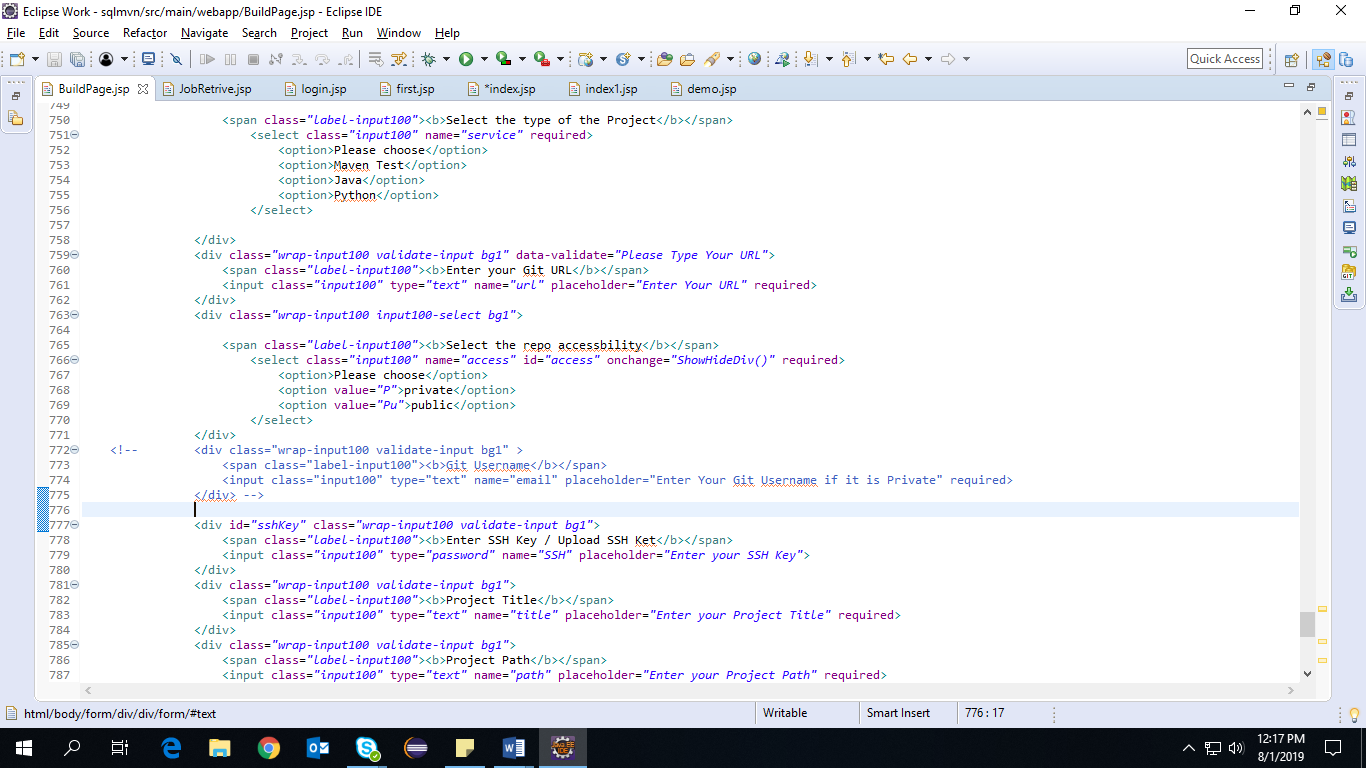
* Above image showing the “login.jsp” page which create the connetion with database and also check in the database that the given Username and Password is in the database or not. If any of this is not match with database, it will show the error message.

4.1.3 First.jsp



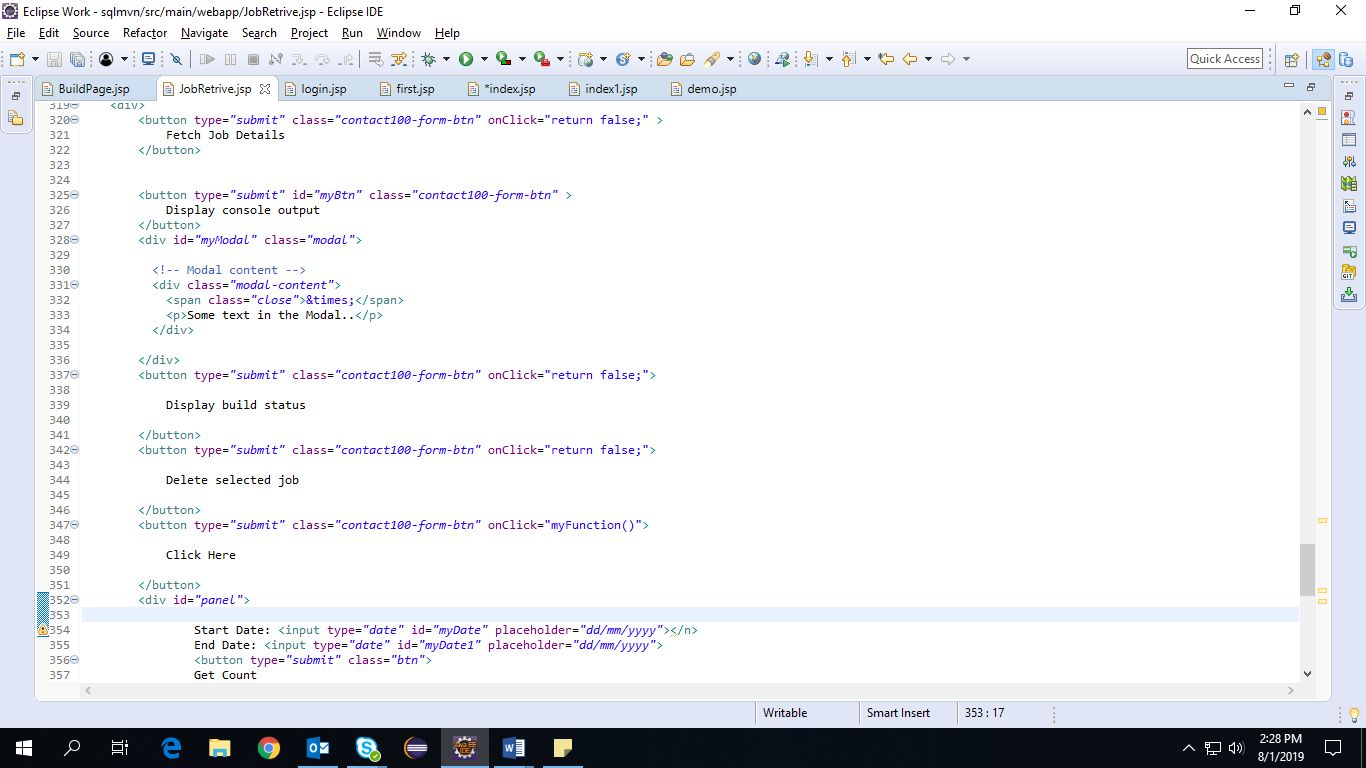
* It’s showing “first.jsp” page which is the home page of the GUI which includes html, CSS and JavaScript part.

4.1.4 Buildpage.jsp



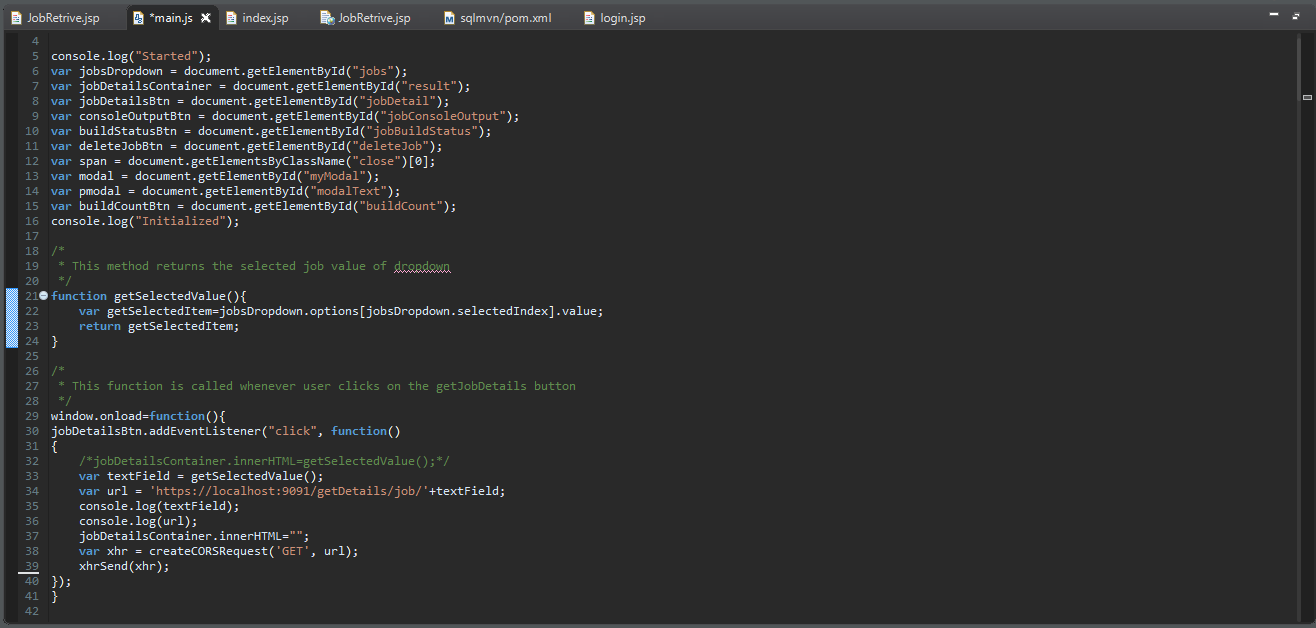
* Above one is “BuildPage.jsp” page which take the input from user in the form of form. Once user add the details it will be stored in database and also it create the Job.

4.1.5 Jobretrive.jsp

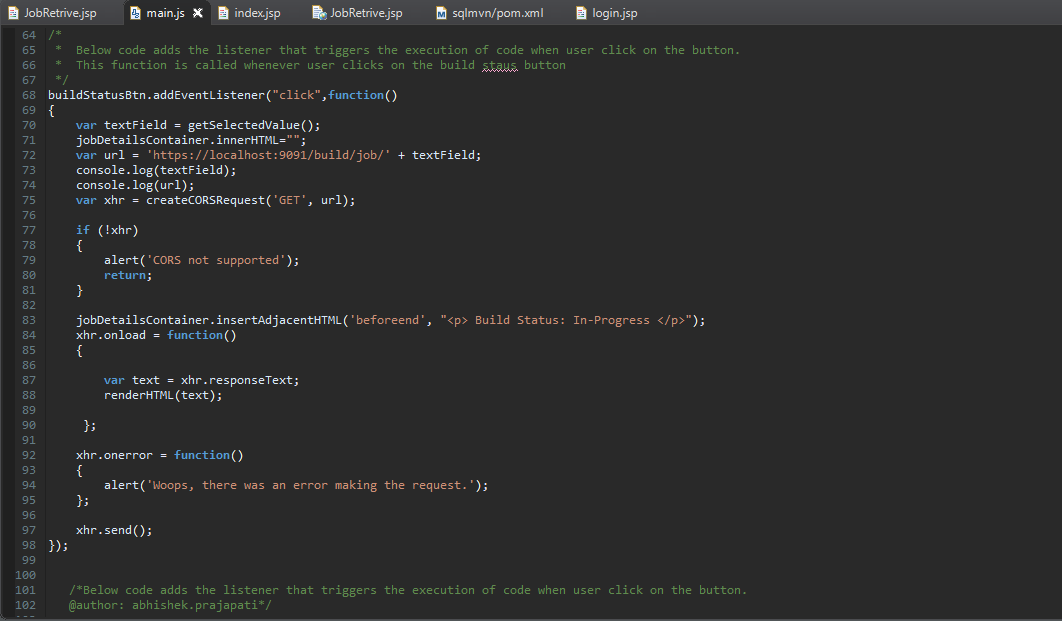


* Above one is “JobRetrive.jsp” page which include the different features and contains html, css and JS.
* Now below is the “main.js”. It will run when we run our service API jar file. It includes different functions which is used to integrate the service API with GUI.

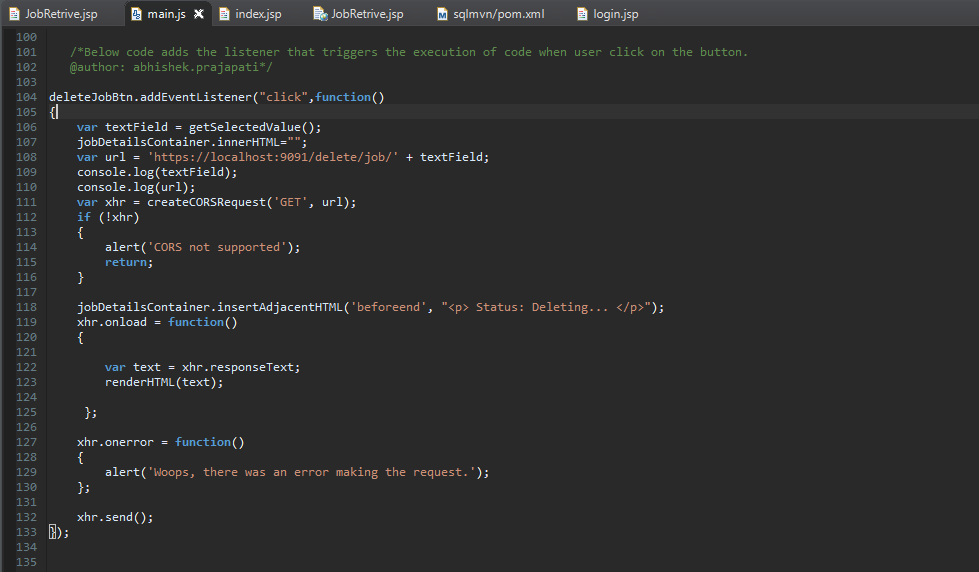
4.1.6 Main.js



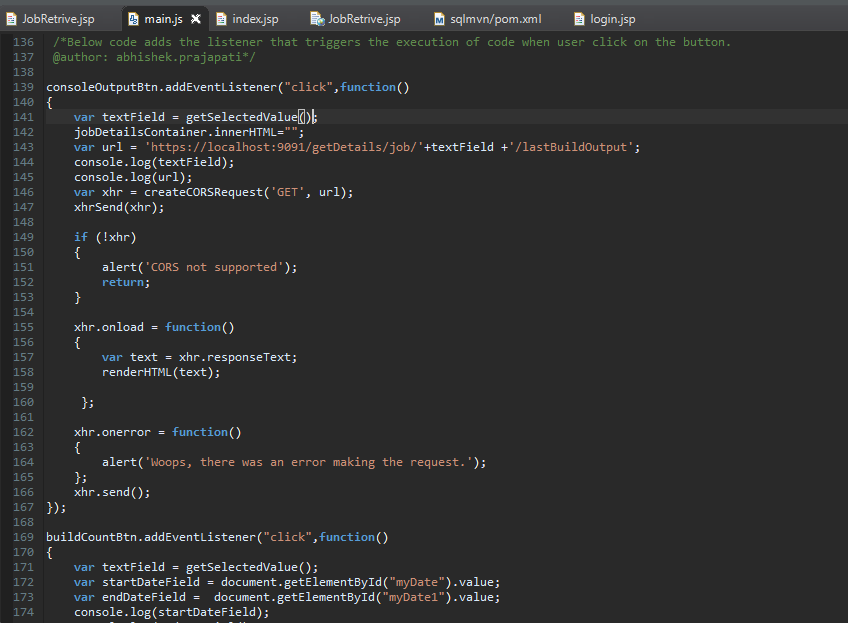
* Here different variables are defined which fetch the UI element using GetElement.
* Here the function which is implemented is JobDetails which hit the service API once click on the button and display the details which you can see on localhost:9091 port.
* Here getSelectedValue method will return the value which user select from the dropdown box.



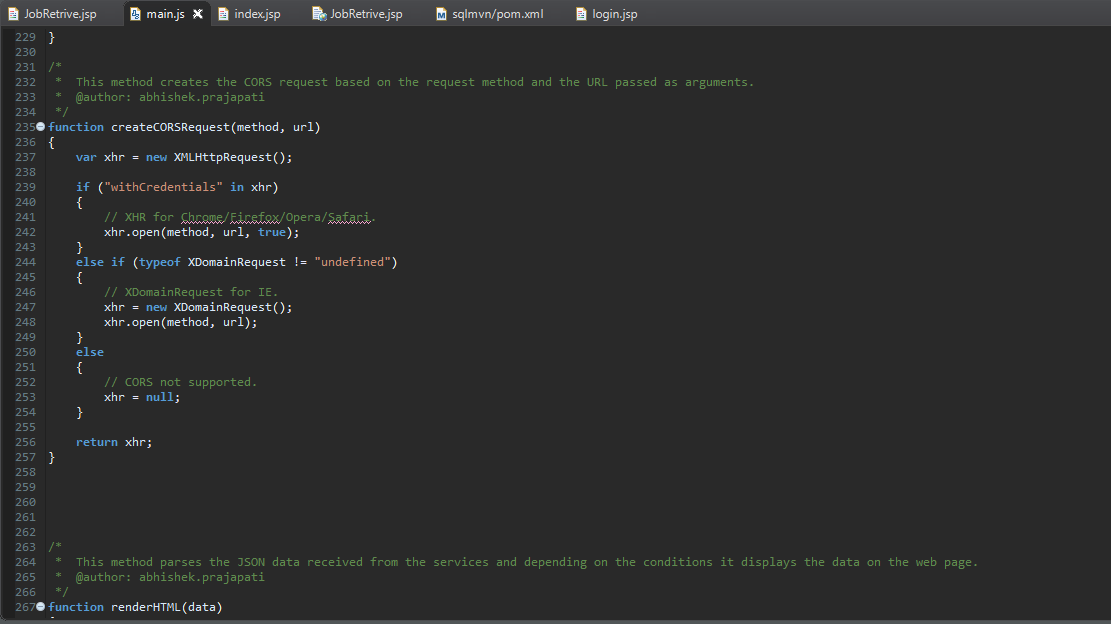
* Below code adds the listener that triggers the execution of code when user clicl on the button and it is called when user click on the build status button.
* Second function which implemented is buildStatus. Once user click on this button it will hit the API and create the GET request and in that time it will show the In-Progress status. And if any issue is occurring during connection or creation of the job it will show the error message.



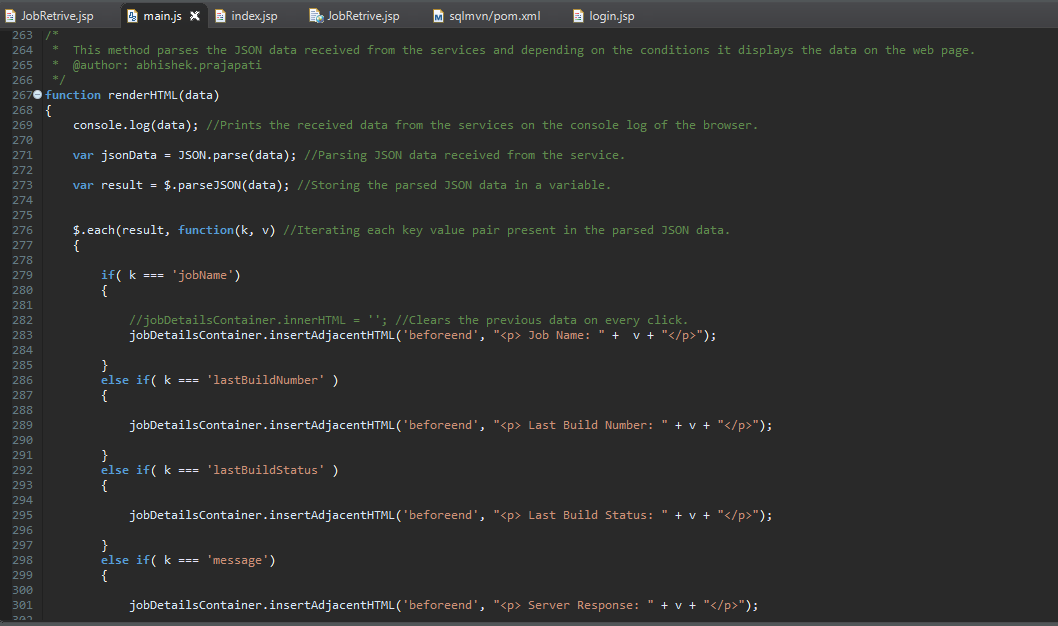
* Third function is delete Job which is working same as Build Job function.



* Consoleoutputbtn function fetch the details from the URL given and and display it in text form. And also display an error.



* As XHR is required for chrome, Firefox here createCORsRequest method create the cors request based on the request method and the URL passed as argument.



* Above method parse the JSON data received from the services and depending on the conditions it displays the data on the web page.
* That console.log prints the received data from the services on the console log of the browser.

