**Calculator**

**Introduction**

Basic calculator allows user to add , subtract , delete and divide any two numbers at given point of time. The calculator provides four operation functionalities along with the validations for number divide by zero and if user enters any character other than number. The calculator provides both server side and client side validation. The advantage of server side validation is that if any user is accessing request with **third party client** the result should show appropriate message at server side.

**System Design**

The calculatortakes the input of two numbers from the user and perform operations like addition , multiplication , subtraction and division. On the client side the calculator checks two basic checks one of them is if the user enters any alphabet or special character apart from number the user is notified by a message that only numbers should be entered. This exception is handled at the server side as well.

The request of division needs two numbers along with constraint that second number should not be zero. This constraint is also handled on both client and server side. For implementation of basic calculator **web services along with JSP and servlets** are used. The validation at server side makes third party client aware about the constraints he/she needs to follow.

**Results**

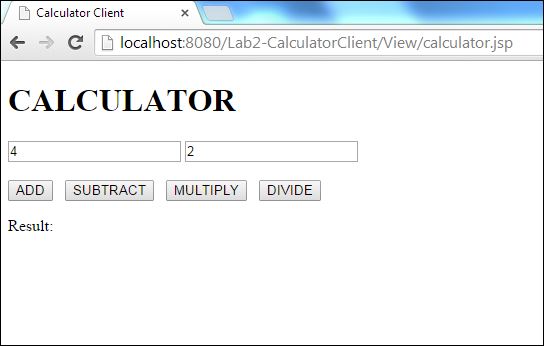
1. **Addition Form**

****

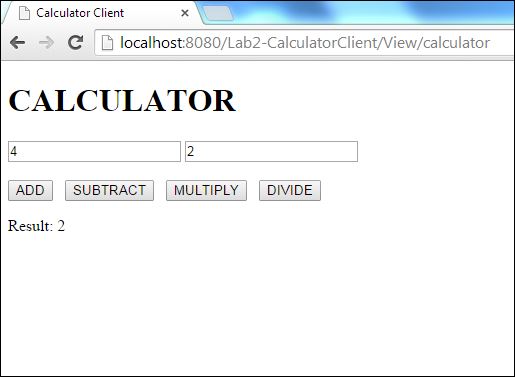
**Addition Result**

****

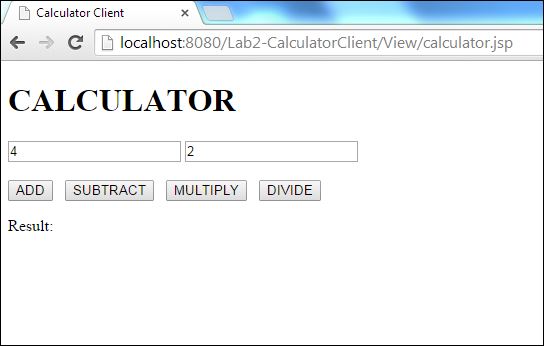
1. **Subtraction Form**

****

**Subtraction Result**

****

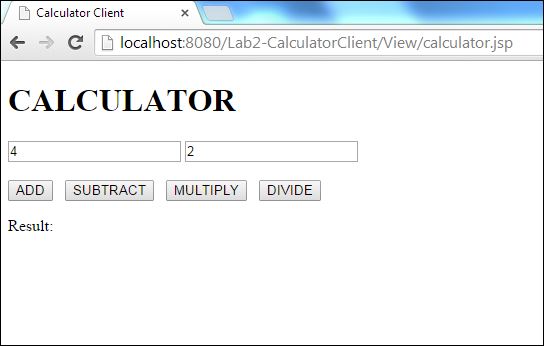
1. **Multiplication Form**

****

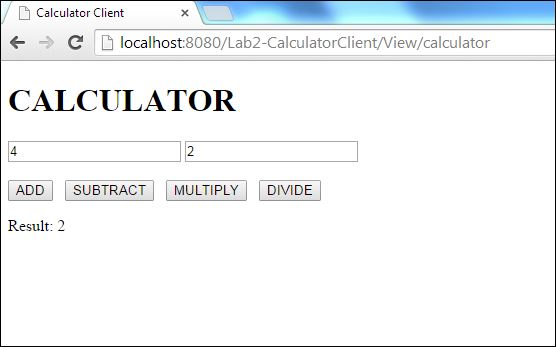
**Multiplication Result**

****

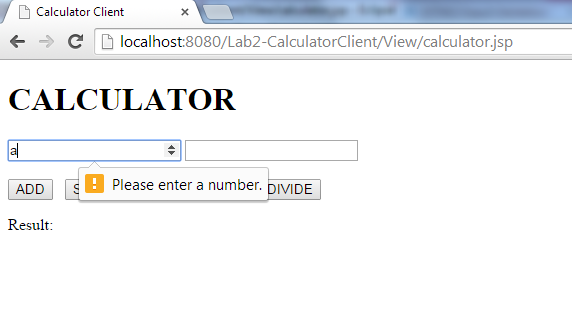
1. **Division Form**

****

**Division Result**

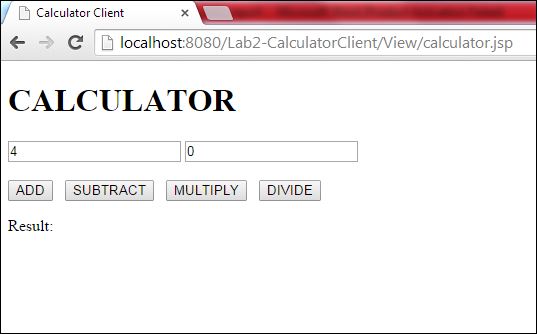
****

**Exception Handling at client side**

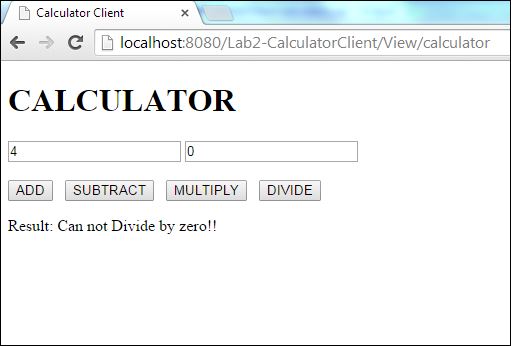
****

**Exception Handling at Server side**

**Divide by zero input**

****

**Error on server side**

****

**Invalid input**

****

**Error on server side**

****

**Calculator Client**

**Results for 1000 random calls**

|  |  |
| --- | --- |
| Operation | Average time in mili seconds |
| add | 8 |
| subtract | 7 |
| multiply | 6 |
| divide | 6 |

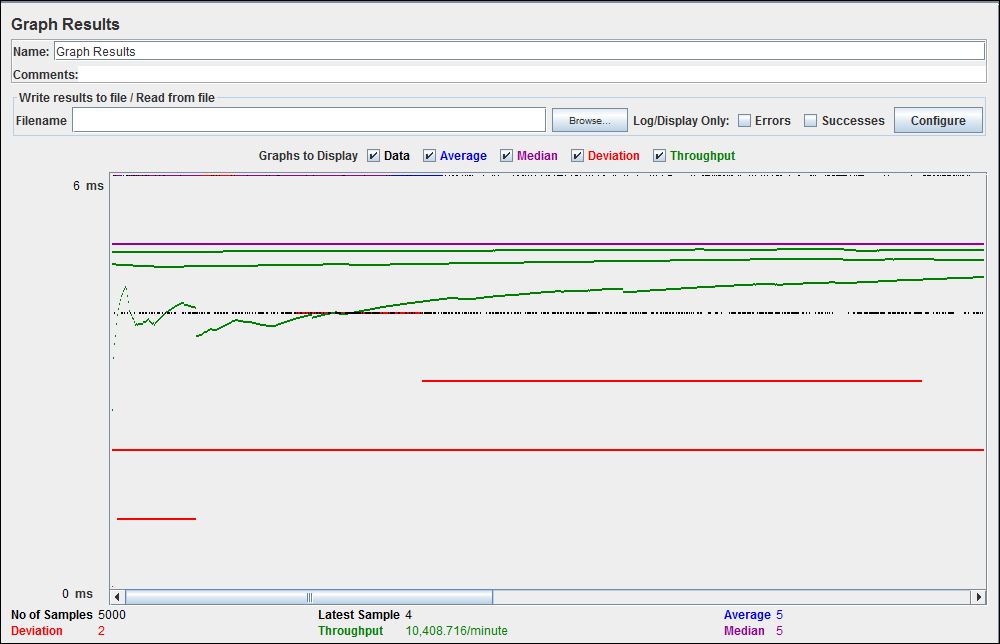
**Graph results for 1000 random calls**

****

**Results for 5000 random calls**

|  |  |
| --- | --- |
| Operation | Average time in mili seconds |
| add | 5 |
| subtract | 4 |
| multiply | 5 |
| divide | 5 |

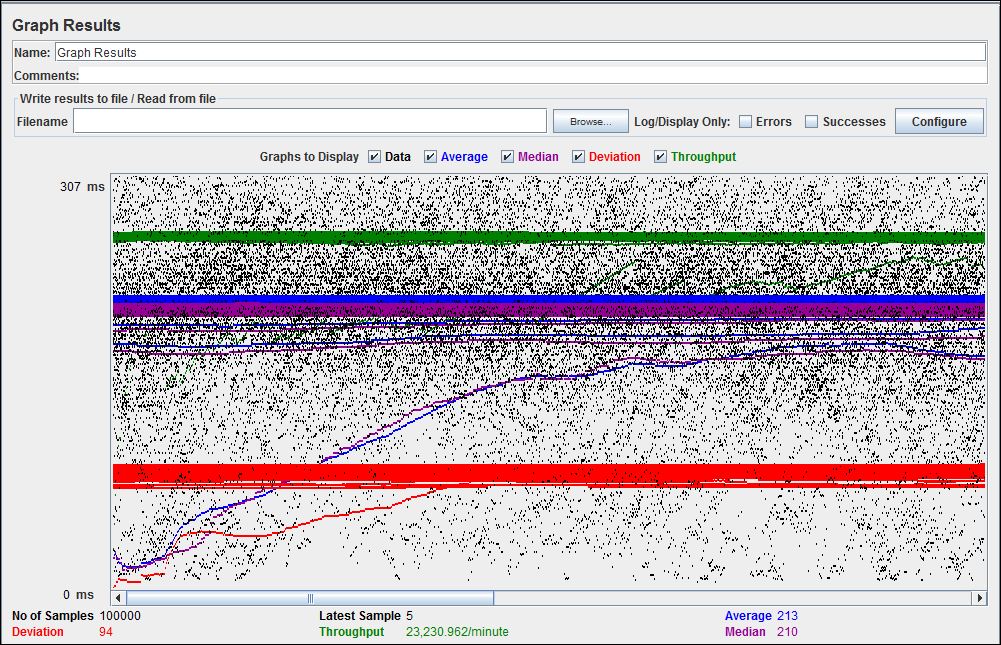
**Graph results for 5000 random calls**

****

**Results for 100 concurrent users with 1000 random calls each**

|  |  |
| --- | --- |
| Operation | Average time in mili seconds |
| add | 220 |
| subtract | 210 |
| multiply | 204 |
| divide | 209 |

**Graph results for 100 concurrent users with 1000 random calls each**

****

**Performance analysis of average time is greater than 10 %**

The application calculator when serving 1000 user the average time is 6 ms for 5000 user the average time is 5 ms and for 100 users with 1000 request the average time is 213 ms. The average time for 100 concurrent users increases because SOAP uses XML which needs to be parsed and its very lengthier which makes SOAP slower than RMI , CORBA and IIOP. Also it has WSDL dependence which does not have dynamic discovery or standardized mechanism for services. **This clearly concludes that serving more number of requests per user increases the average time significantly**. Hence this application is not scalable and it cannot serve concurrent request of multiple users efficiently.

**Comparing results with REST of LAB1 Calculator**

In lab1 calculator was implemented using NodeJS which uses REST which significantly reduced the average time per request than the SOAP based application. The major reason behind this is REST is stateless protocol and information can be cached which makes REST faster than SOAP. Since REST uses standard HTTP hence its simple and faster. Rest is better in performance and scalability. Rest reads can be cached whereas the SOAP based reads cannot be cached. On the other side the SOAP uses XML which is very large and which needs to be parsed each time the request comes hence the SOAP is slower than REST. The REST services don’t have extra xml mark up as that of SOAP based protocol. For REST services you just need HTTP stack or a browser which is almost available in every device. **Hence the calculator implemented in LAB 2 using web SOAP takes more average time than that of calculator using NodeJS and REST.**

**Yelp**

**Introduction**

Yelp is people sourced social networking and business review site. Site has various categories like restaurant , food , coffee and tea. The users in yelp can give reviews and ratings about particular product or service. Yelp allows to Sing In and Sign Up options which allows user to create account and use the account. Administrator can add , remove and update category where as normal user can add , remove and update his/ her reviews and ratings. The yelp also shows reviews of other people and gives chance for user to edit his/ her reviews. User can always able to see the last login time while navigating through yelp. Yelp provides validations for user and the administrator.

**System Design**

The yelp site has home page which gives option to admin / user to login or sign in to the account. The admin can login to the system using admin privileges. If admin does not gives appropriate user name / password / email id / last name / first name the system at server side will do the checks and display appropriate message to the admin. The system checks the admin’s credentials and displays a page that allows admin to add / delete / update category.

Yelp checks users credentials at login and sign up time. If the user does not enters appropriate credentials then system throws the message of required values of fields at server side. Once the user is authenticated he can add / delete / update particular reviews and comments. User can see his comments when he logs in along with the last login date and time. User can see all the comments of other users at home screen.

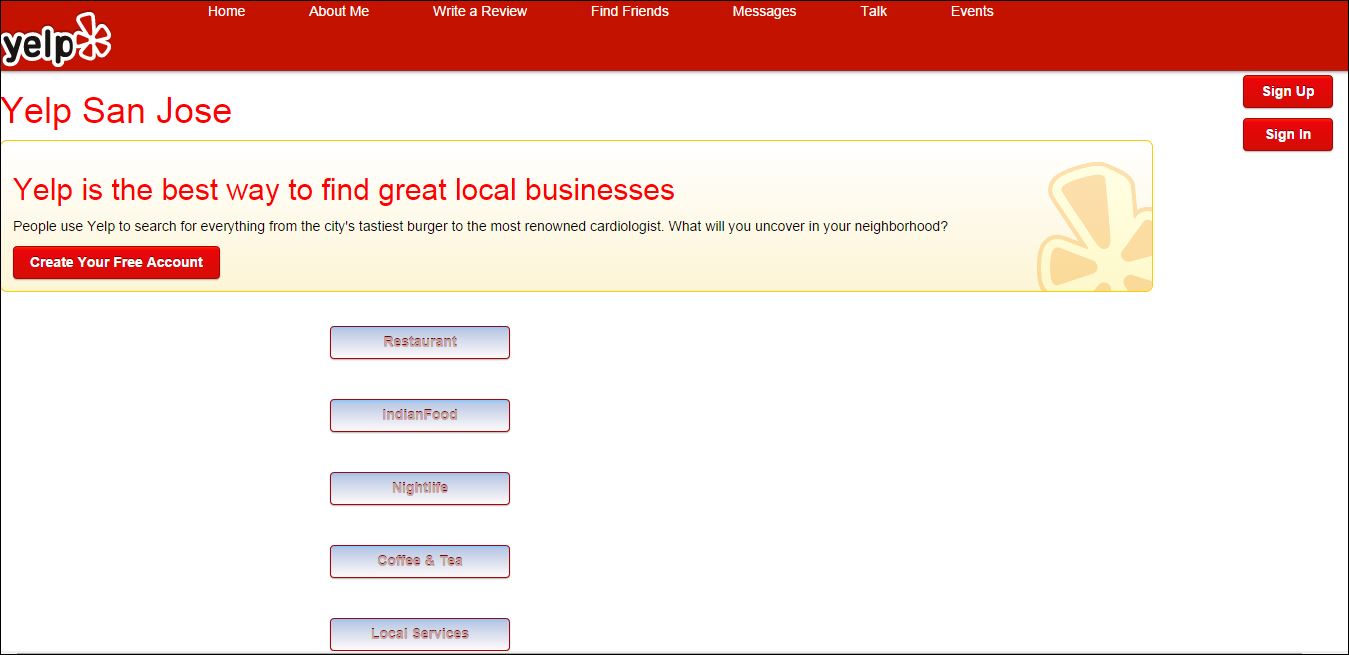
For building Yelp application **web service along with Servlets and JSP** is used. The server consists of **Apache tomcat 7**. The server has **deployment descriptor** which has all the information about the application including services exposed. The server has **service class which is exposed to the client** with the required dependencies. Server side communicates with **MYSQL database.** Server creates a connection for MYSQL database and serves request of databases for client.

Client side consists of **JSP and Servlets** which consumes theservices those are exposed by the server. Stubs are generated at client side. Client side has **proxy methods** , these proxy methods can be called from the servlet and data can be fetched / inserted through using proxy services exposed by server. Client side consists of **Web.xml** which is client side deployment descriptor. The web.xml has mapping of **URL-Pattern** to its corresponding **servlet** name.

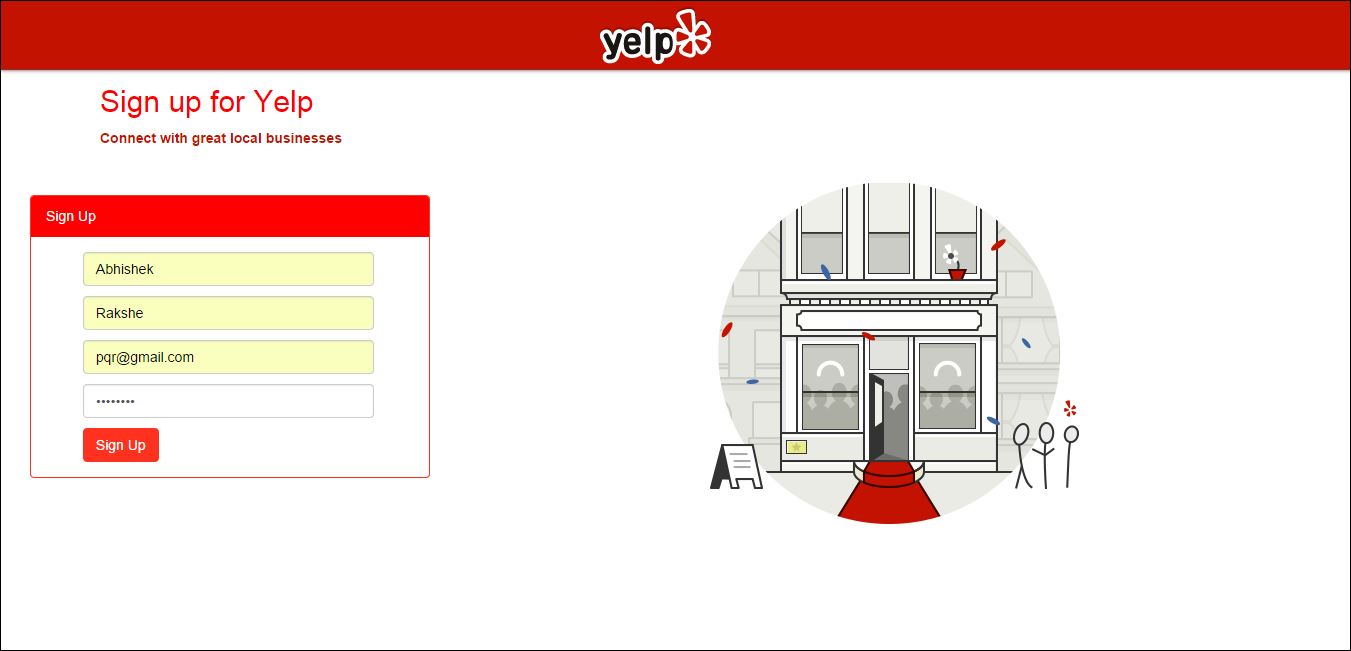
Client has home.jsp file as a welcome file. The flow starts from home.jsp to the individual servlet as per the user interaction with the yelp application. When any action is performed by used the web.xml is checked at client side and corresponding servlet is invoked. The servlet consists of proxy methods which makes a call to actual service exposed by the service class at server side. This service class has connection with database which makes SQL calls and fetches data sends back response to the client JSP file.

**Results**

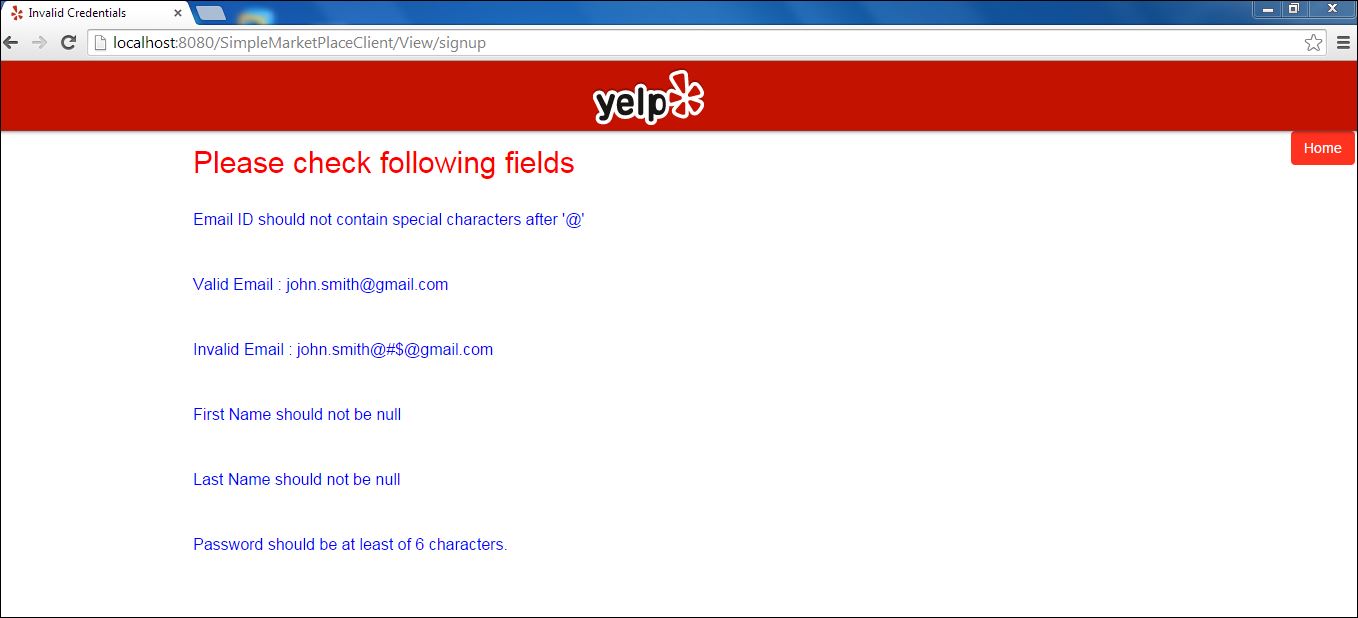
**Home page**

****

**Sign Up**

****

**Sign Up – If credentials are not correct**

****

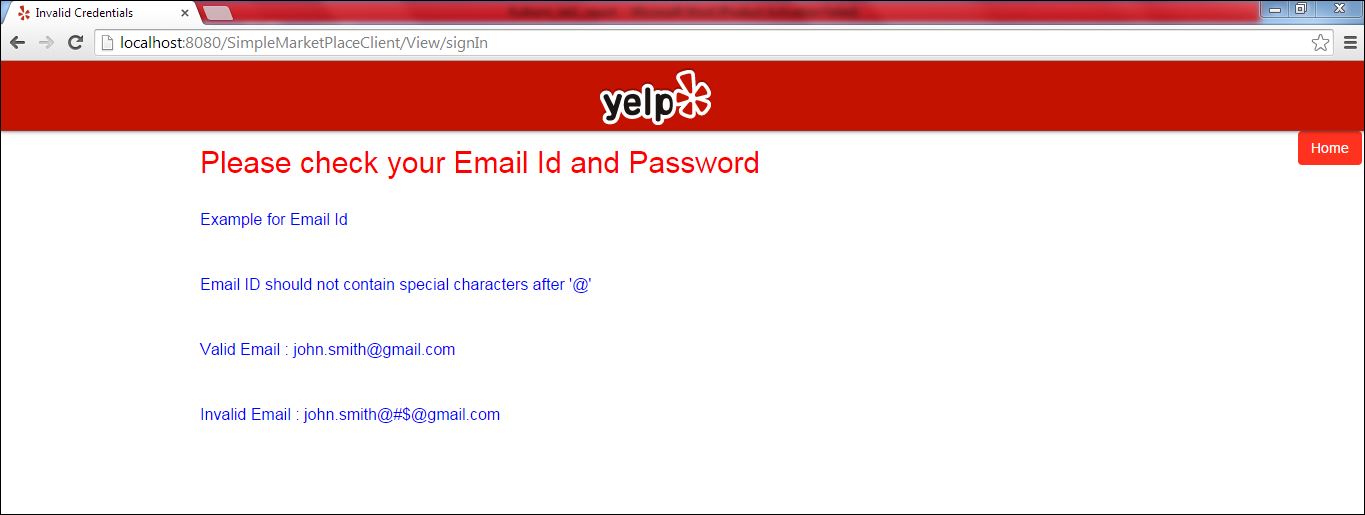
**Sign Up – If credentials are correct the message will be displayed with login creation time**

****

**Sign In**

****

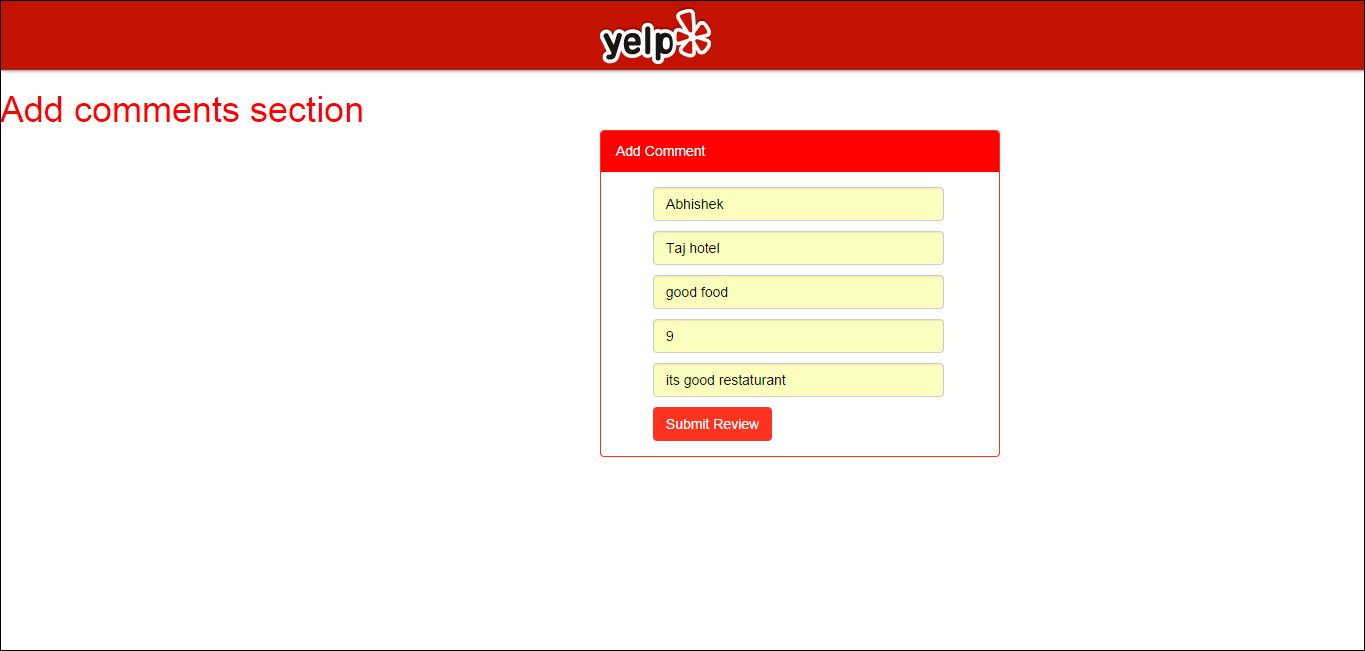
**Sign In – If credentials are not correct**

****

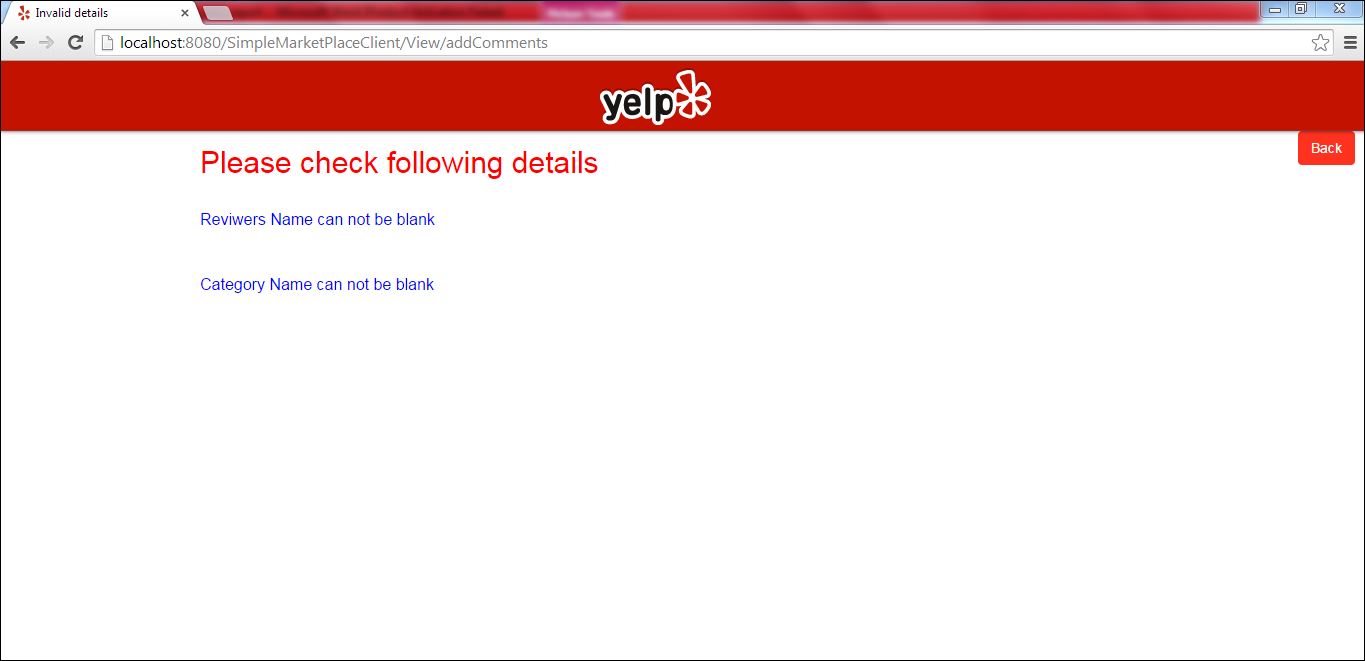
**Sign In – If credentials are correct, the user will see his/ her comments with last login time.**

****

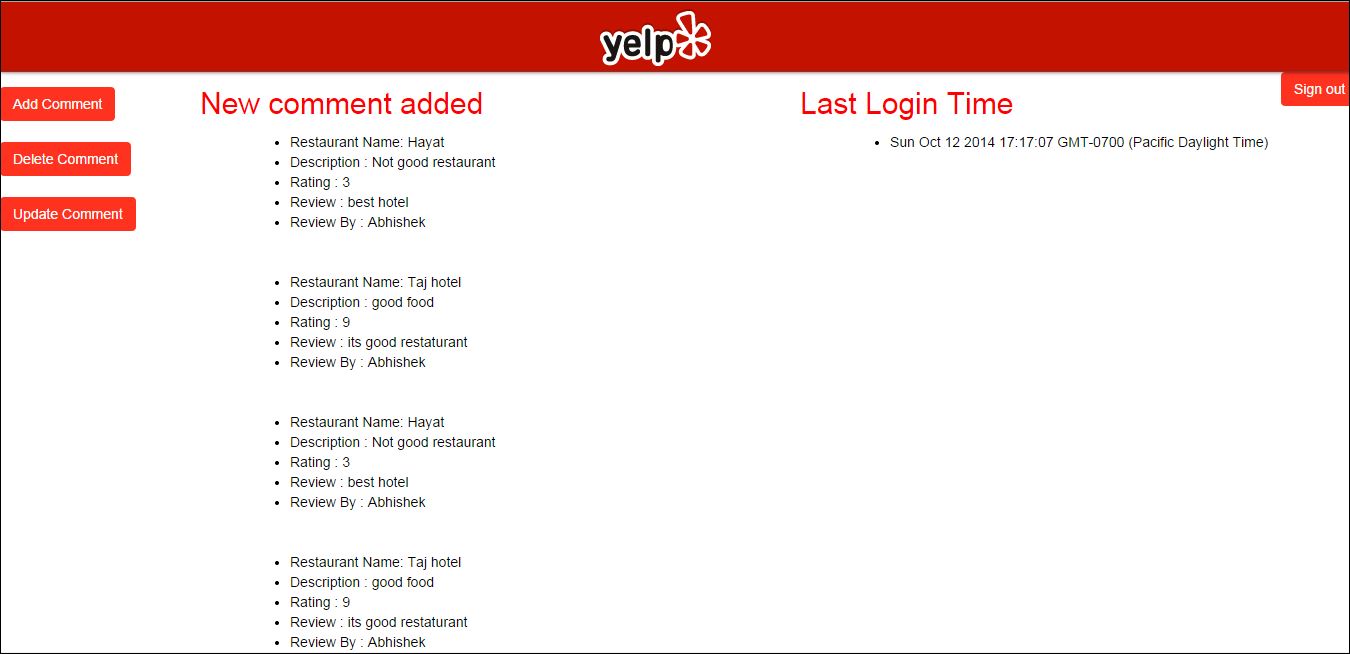
**Add comment form**

****

**Add comment - If comments rating is not number and reviewer does not give his name.**

****

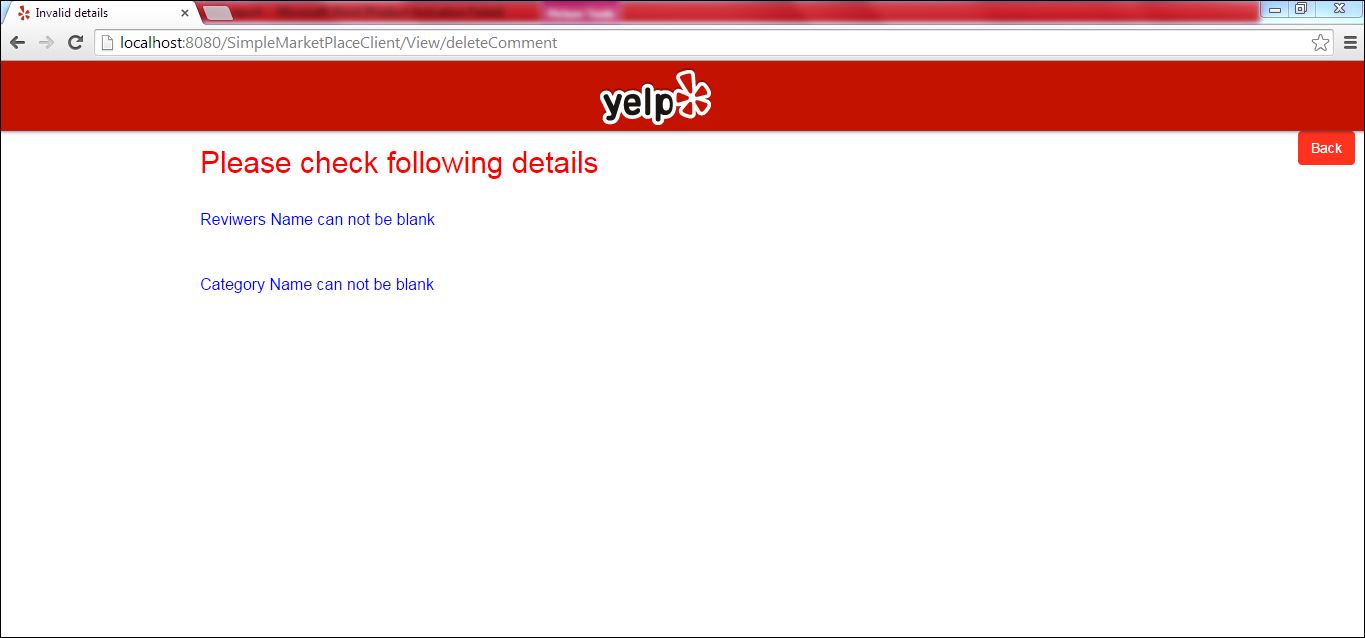
**Add comment - If correct comments are added.**

****

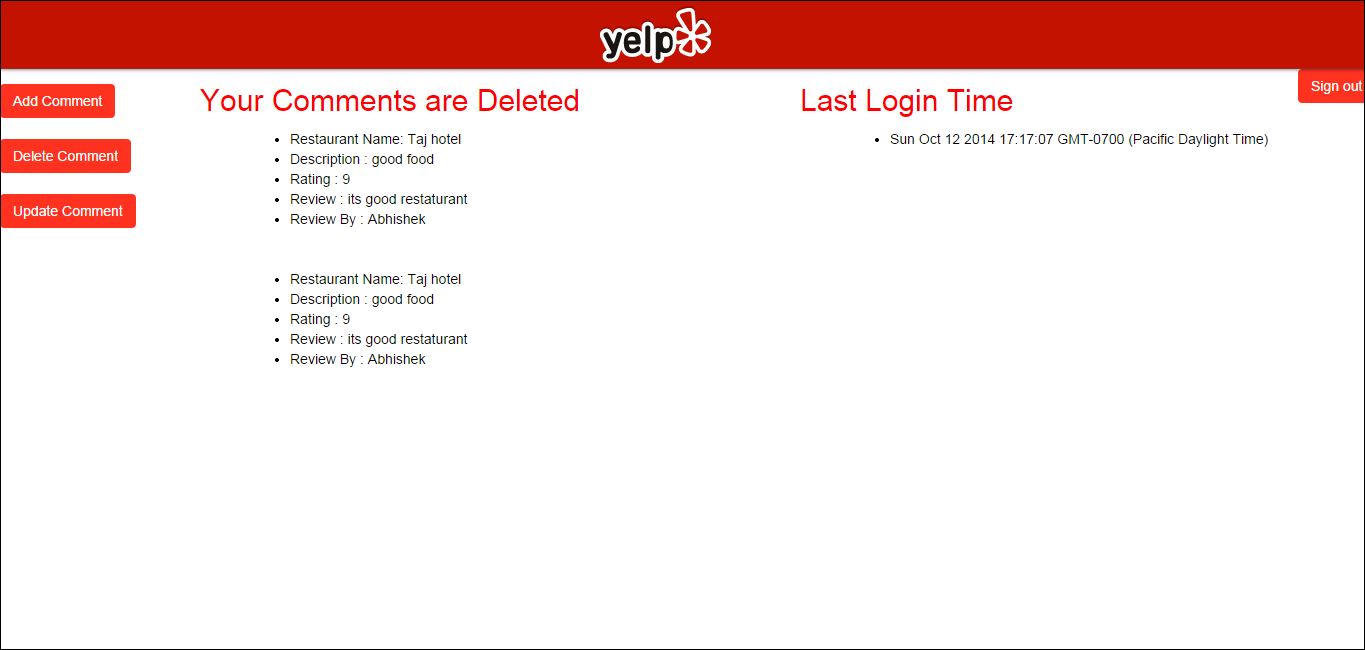
**Delete comment form**

****

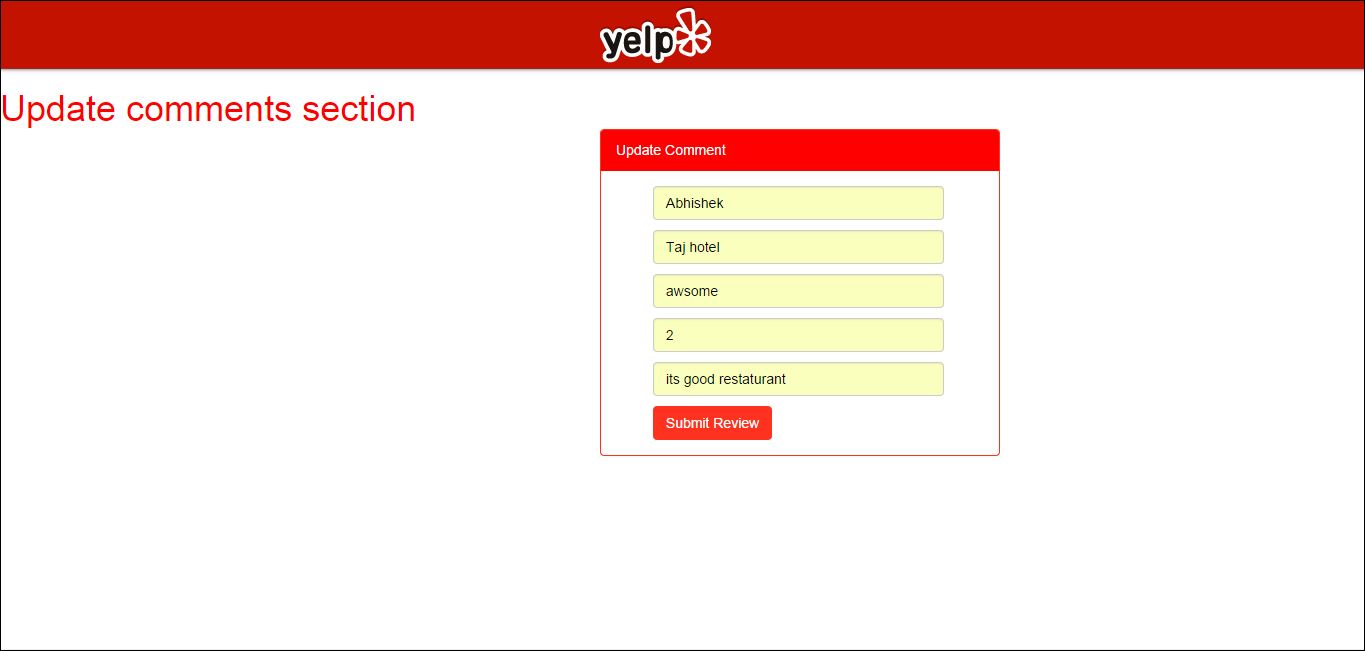
**Delete comment - If incorrect comment user tries to delete.**

****

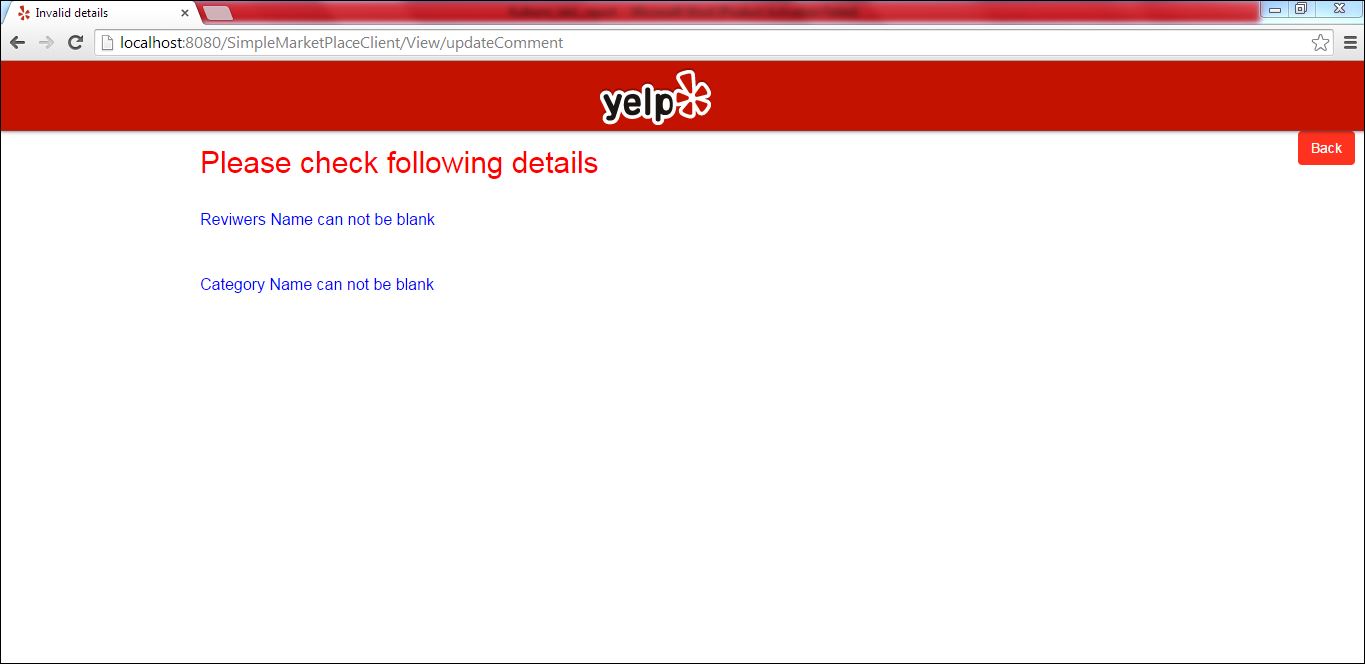
**Delete comment - If correct comments are given.**

****

**Update comment form**

****

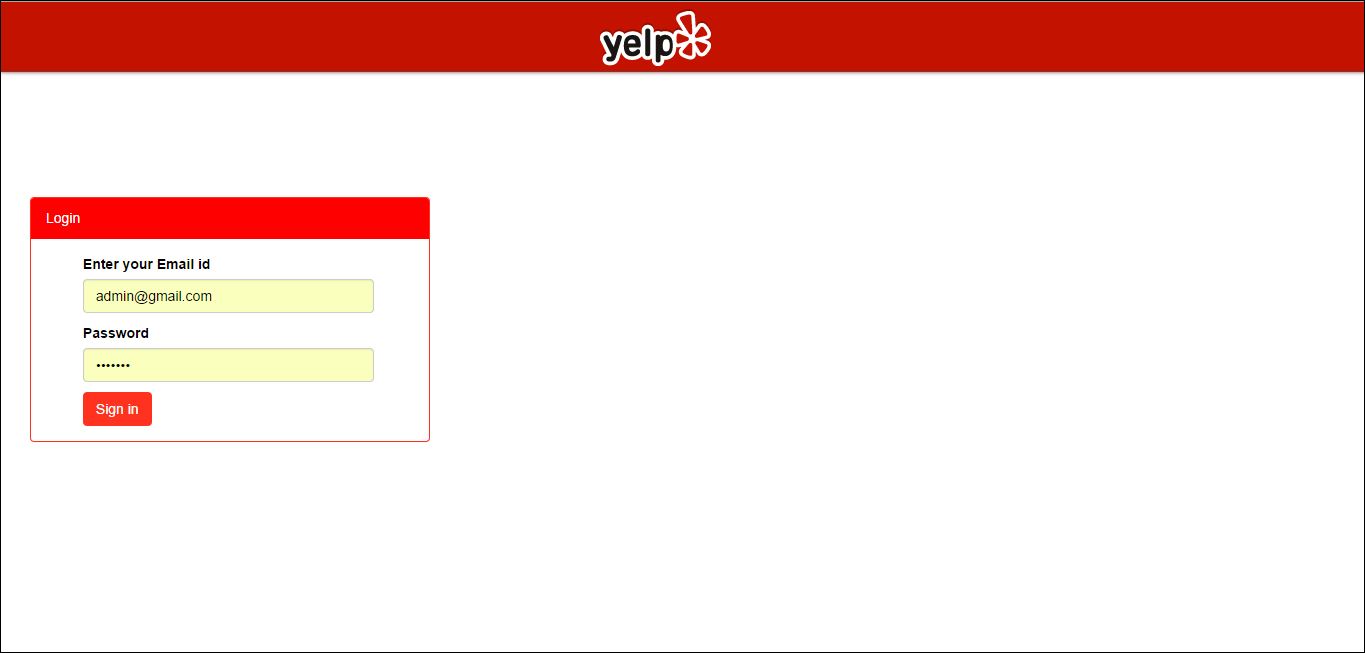
**Update comment - If incorrect comment are given by user.**

****

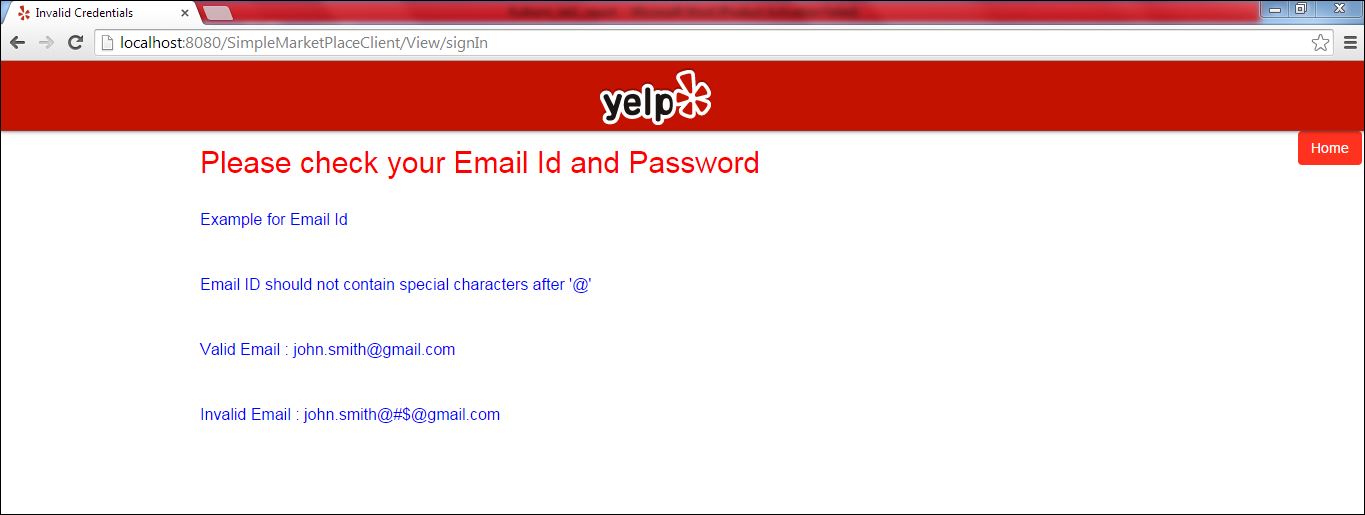
**Update comment - If correct comments are given.**

****

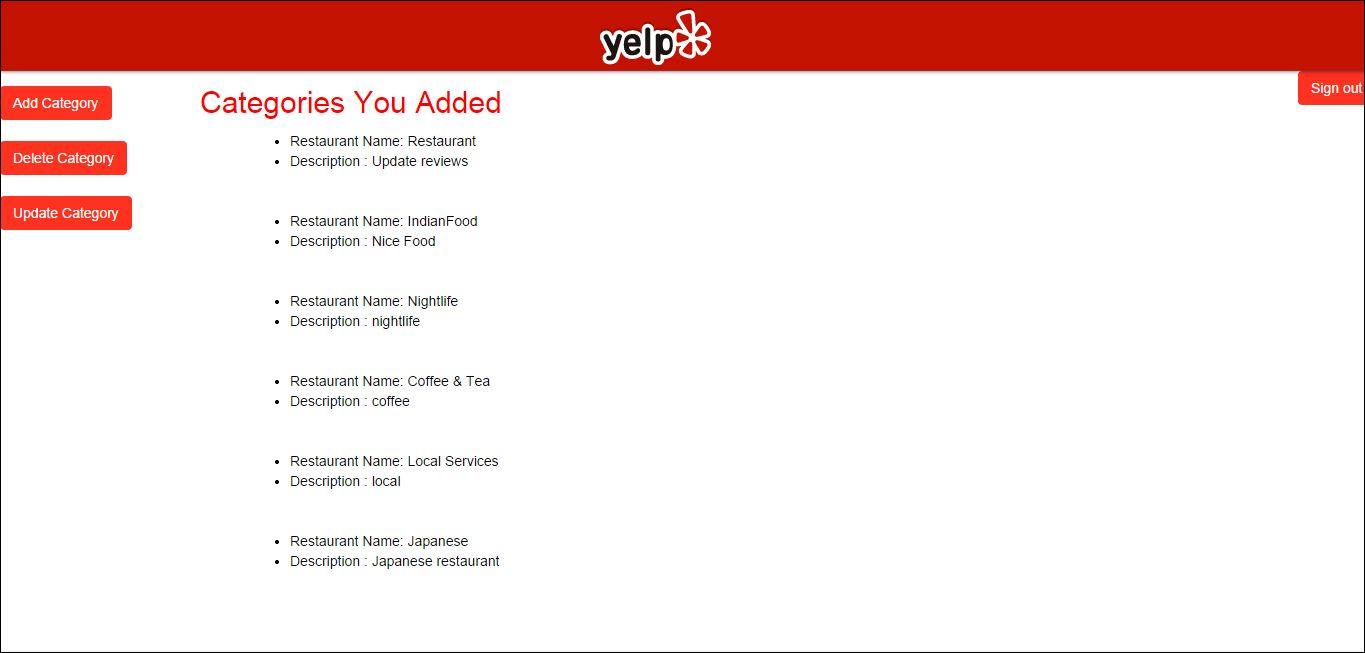
**Sign In – Admin**

****

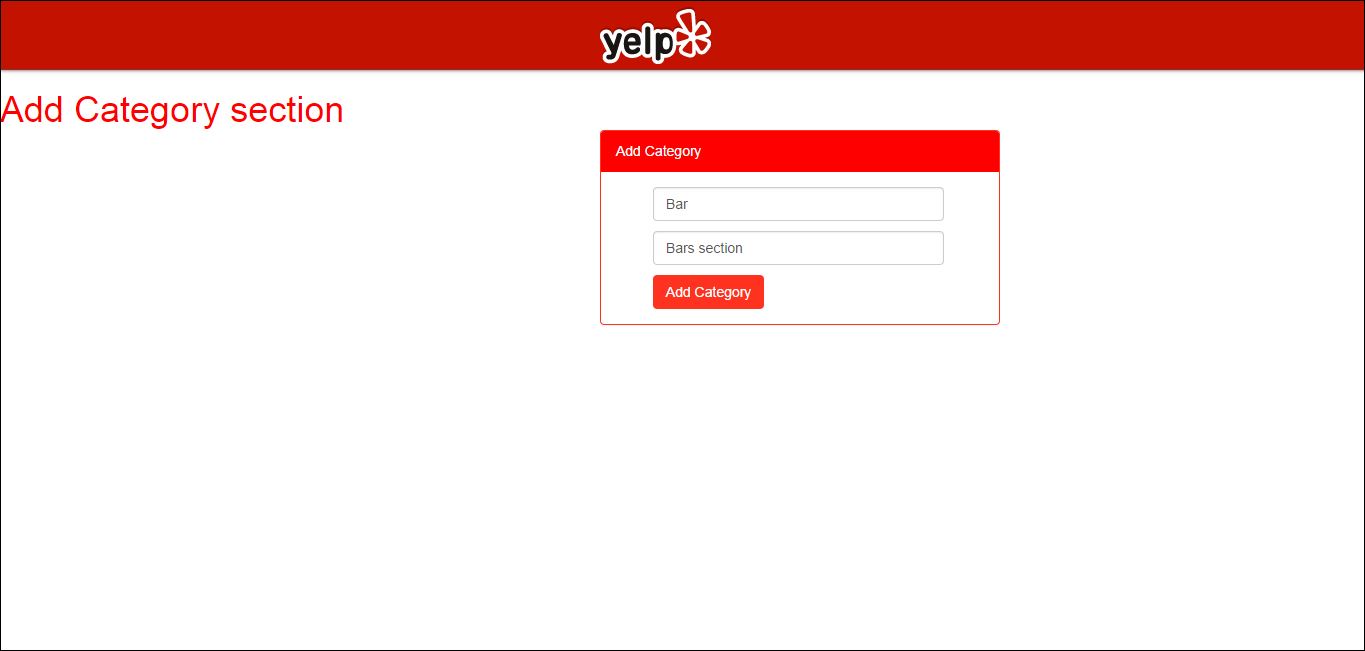
**Sign In – Admin - If credentials are not correct**

****

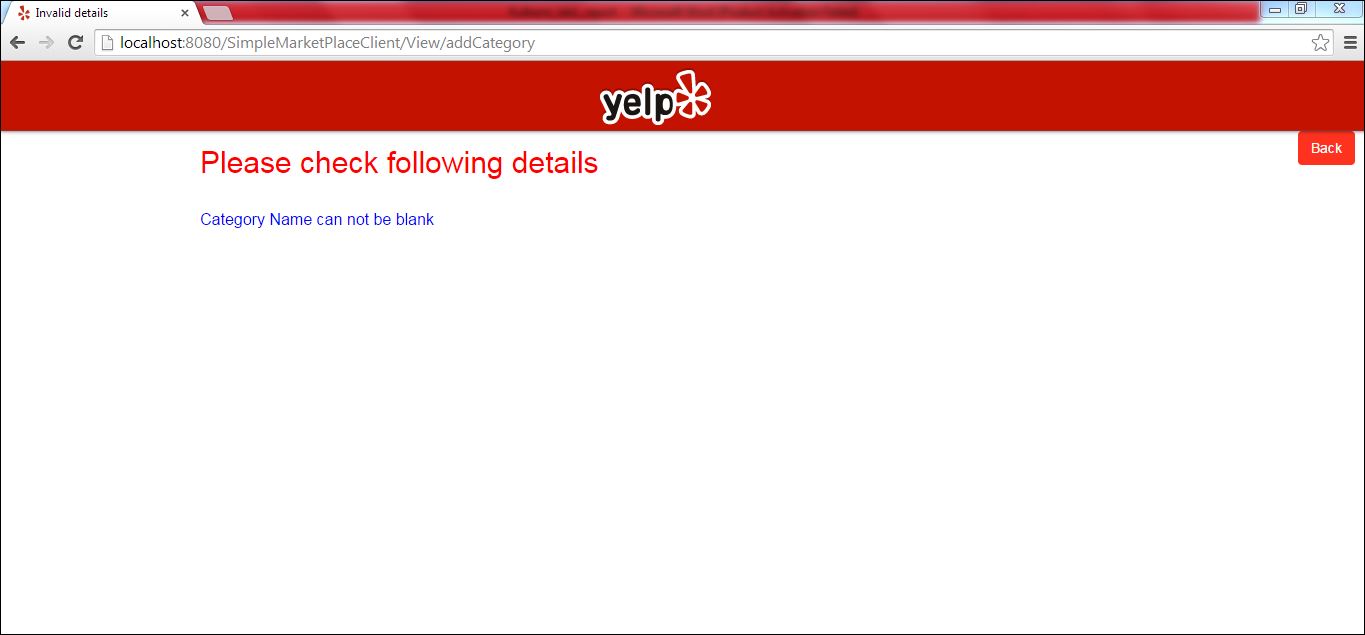
**Sign In – Admin - If credentials are correct**

****

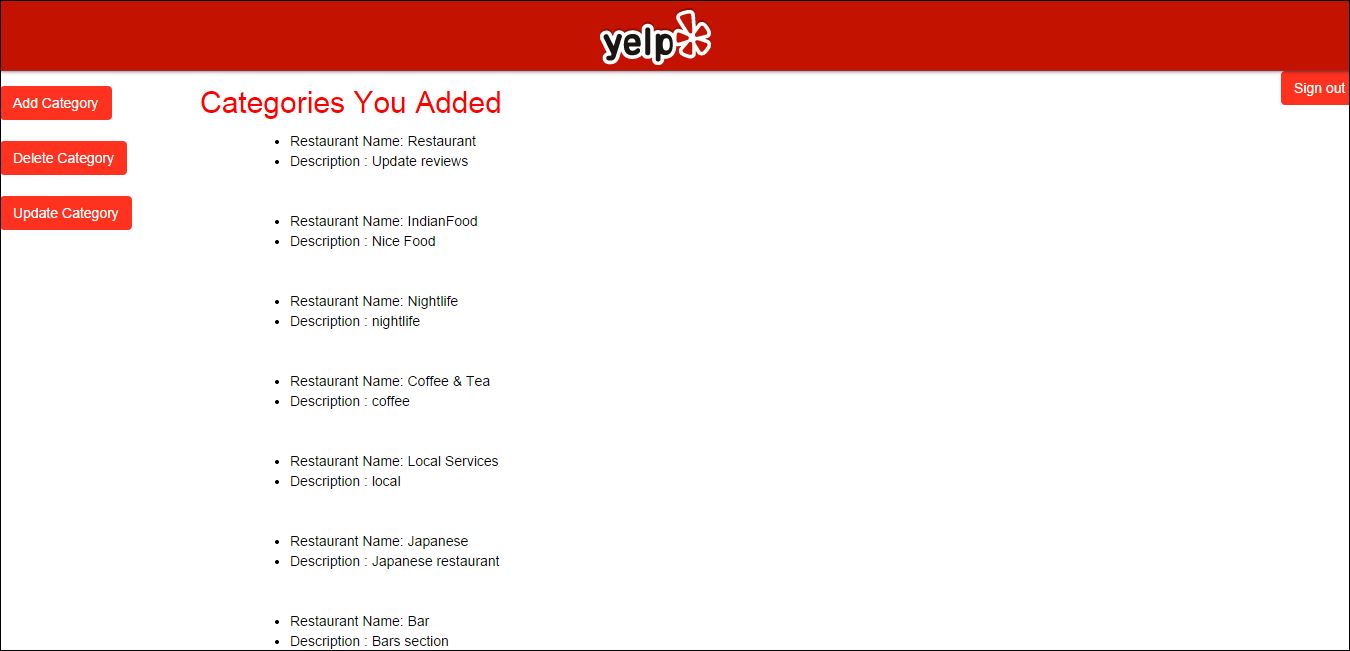
**Add category form**

****

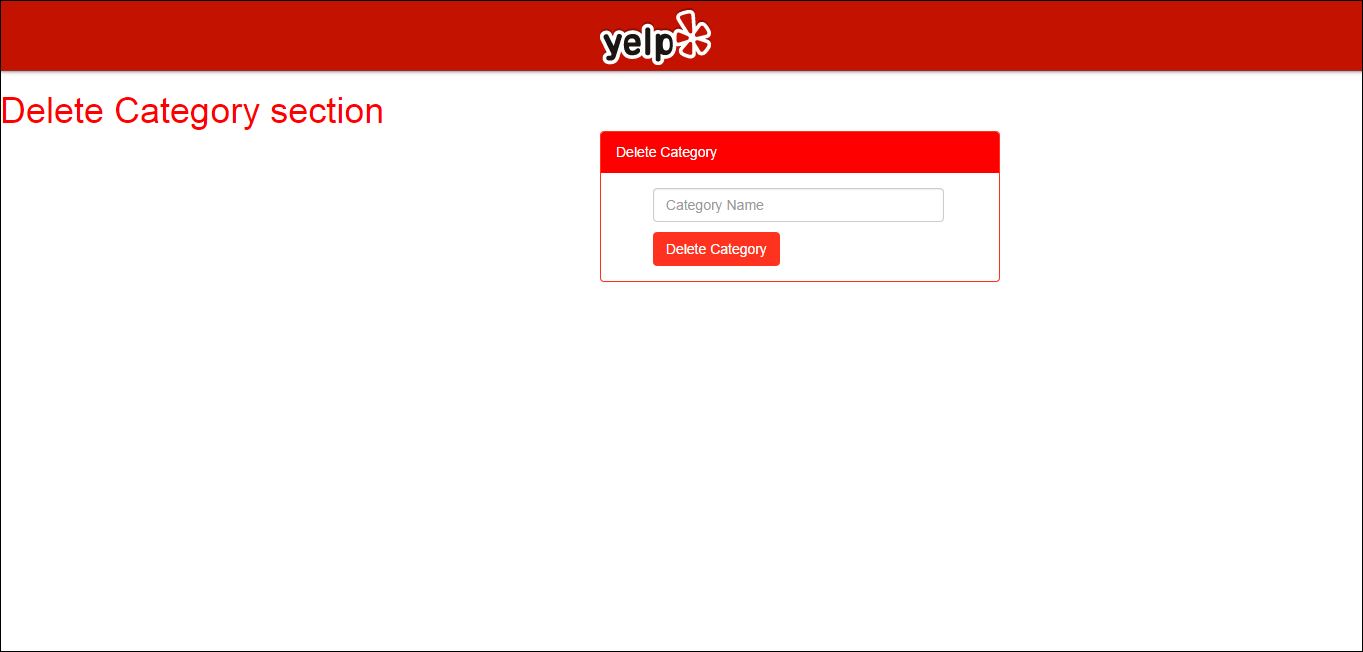
**Add category form : If category name is given as null.**

****

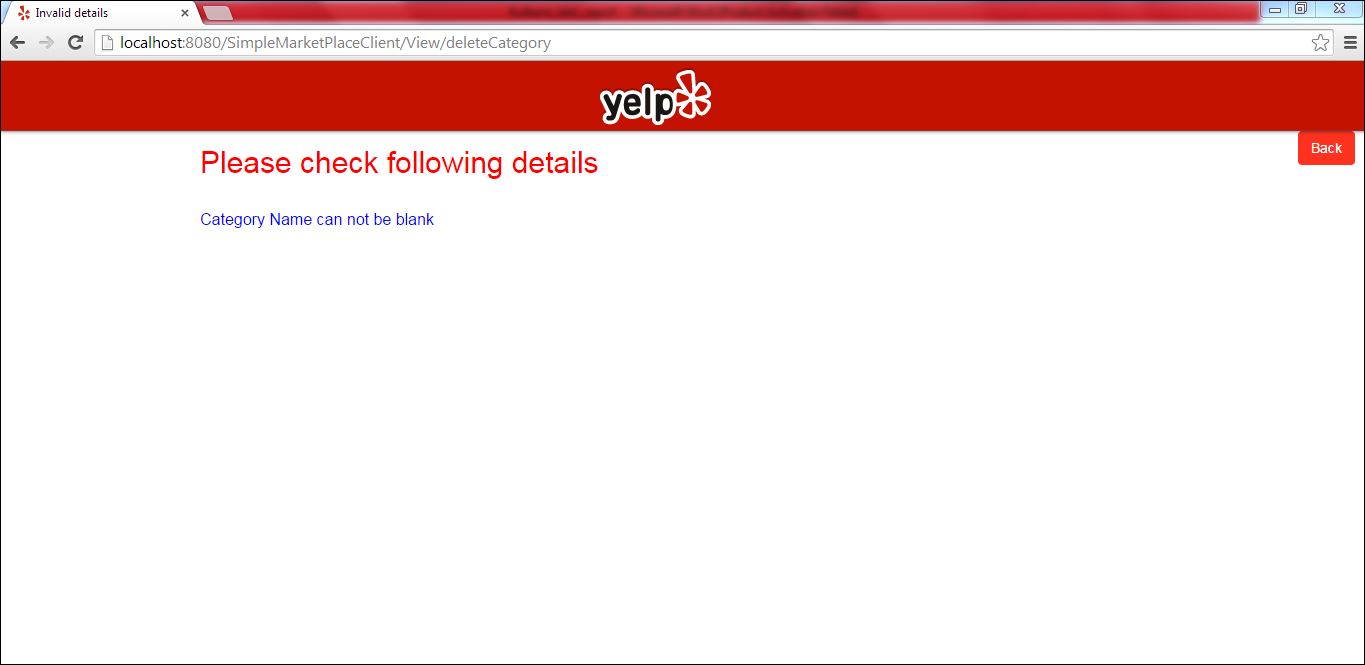
**Add category form : If category name is given correctly.**

****

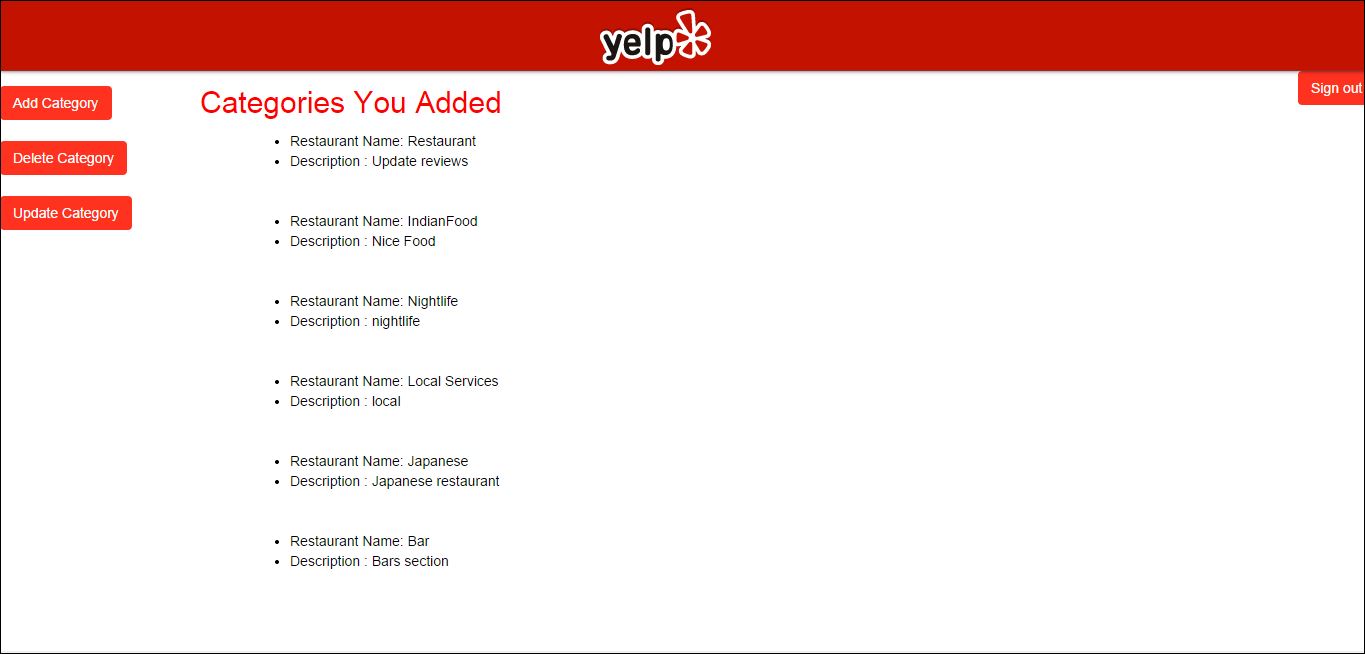
**Delete category form**

****

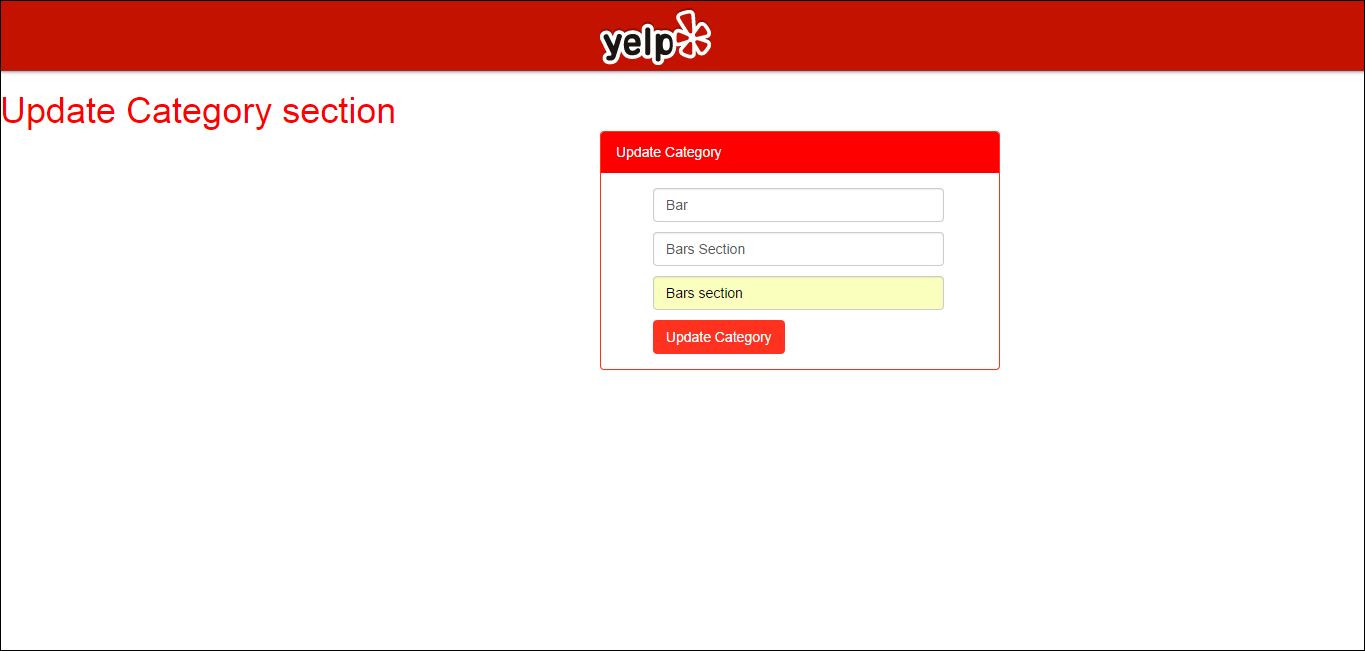
**Delete category form : If category name is given as null.**

****

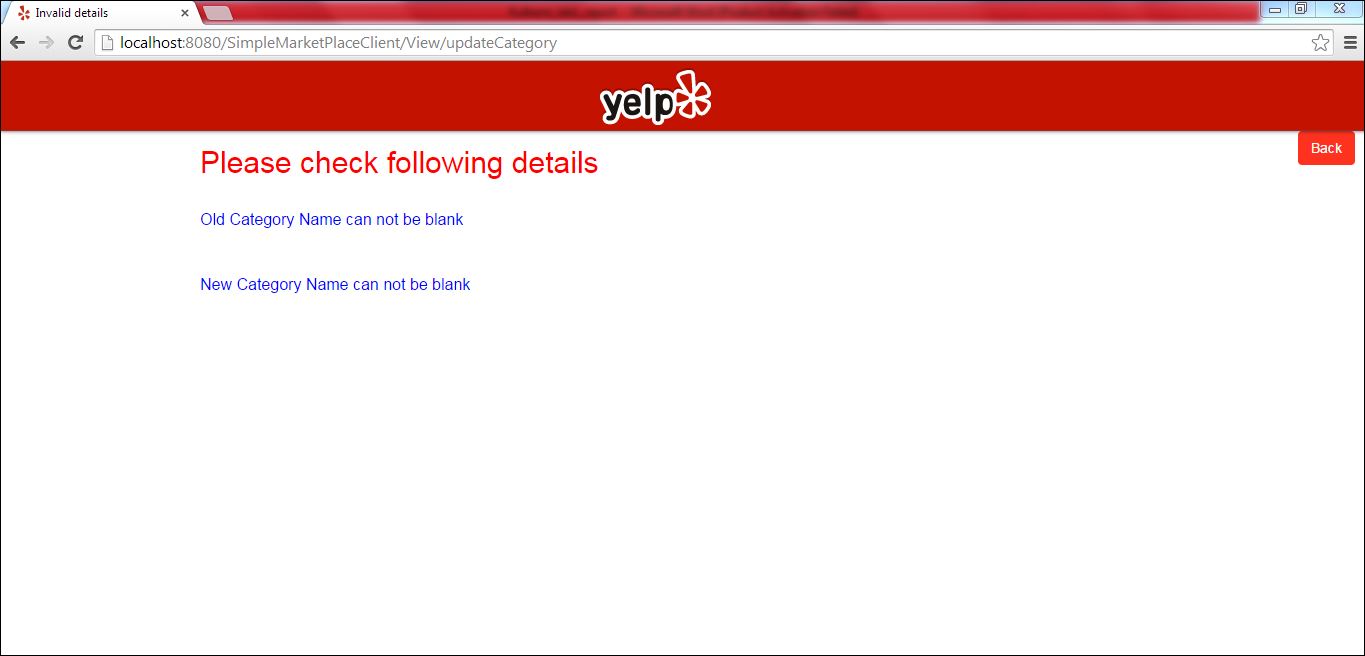
**Delete category form : If category name is given correctly.**

****

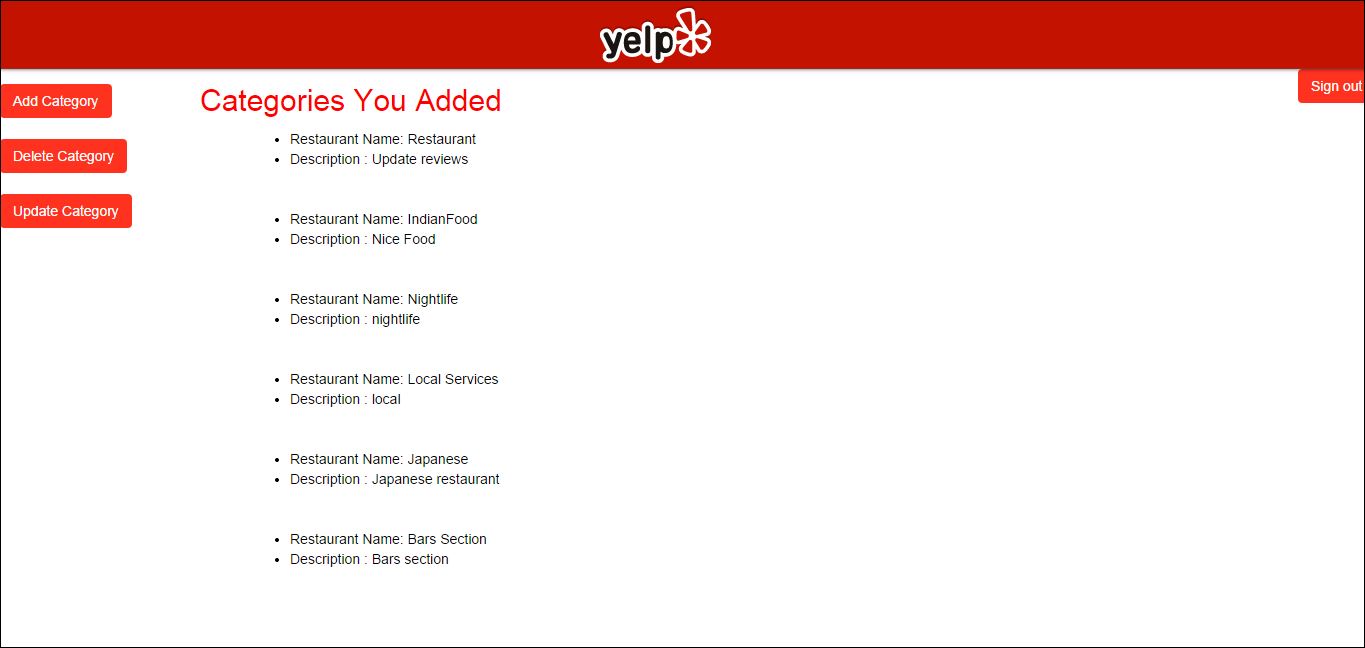
**Update category form**

****

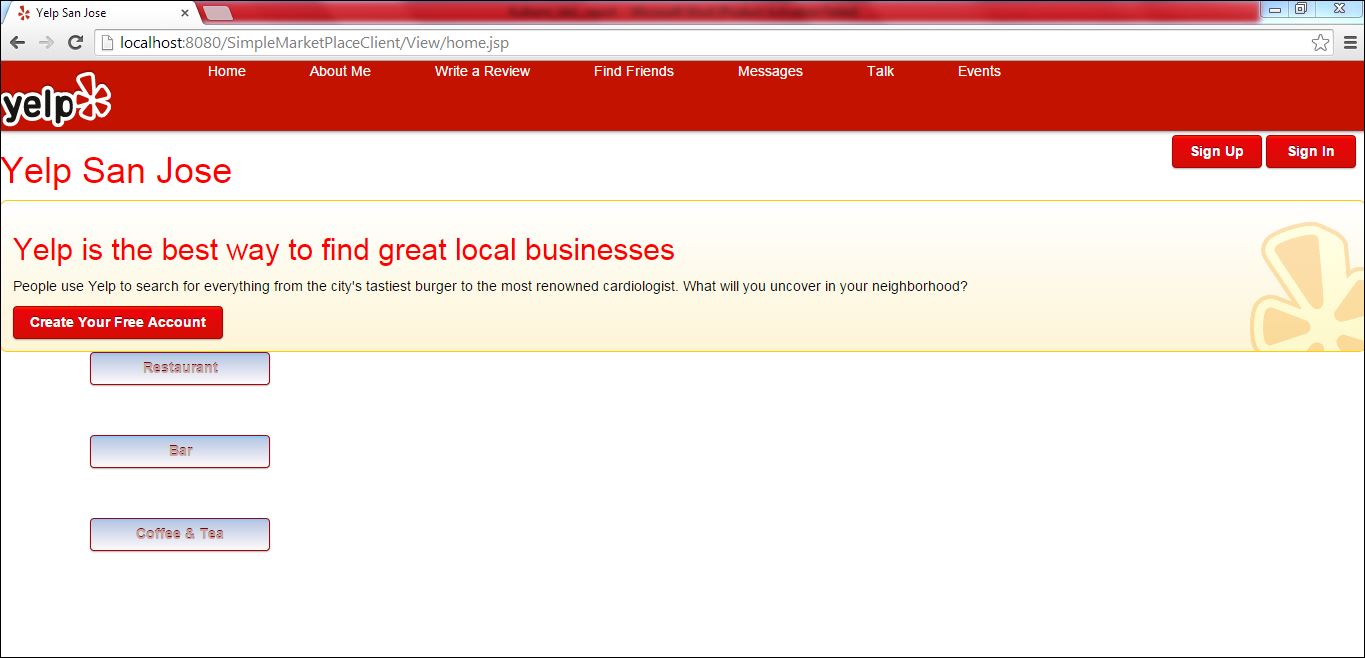
**Update category form : If category name is given as null.**

****

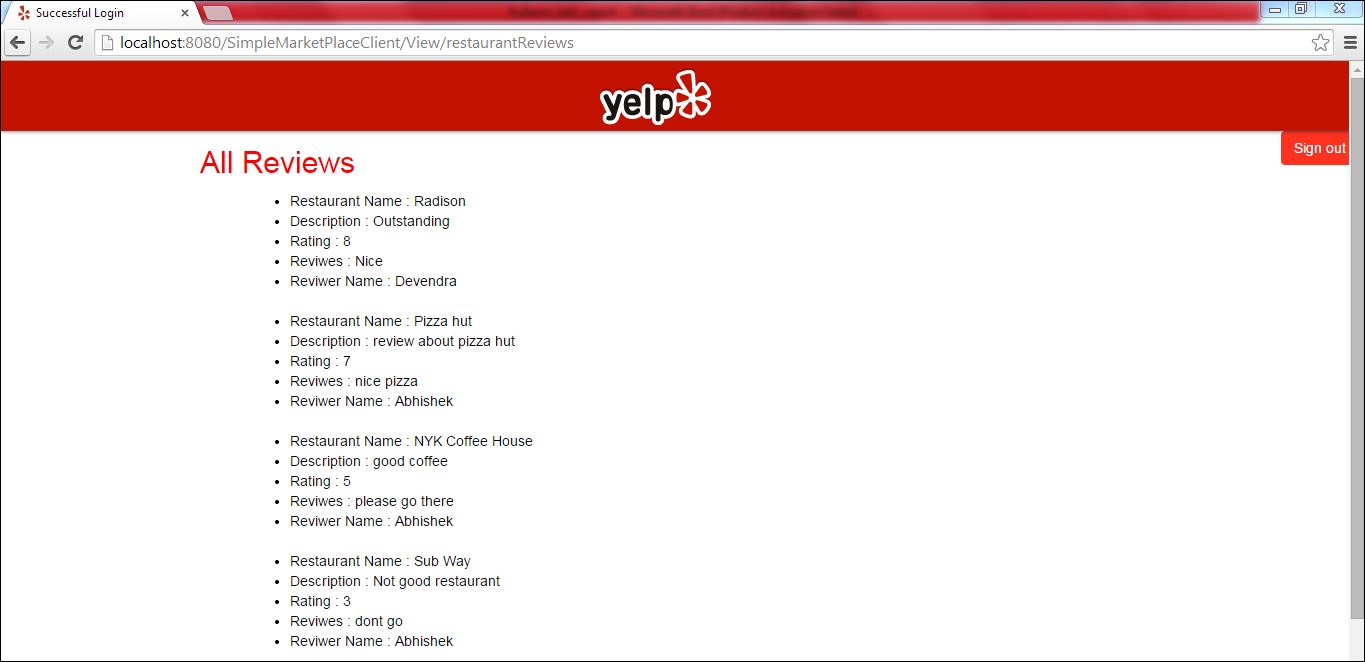
**Update category form : If category name is correctly given.**

****

**Sign out : Sign out will redirect user to home page.**

****

**Restaurant on home page : If user clicks on restaurants on home page he will see all the restaurants and its review wrote by other people**

****