**Name : Sravani Gunti**

**SJSU ID : 010090405**

**CMPE273: Enterprise Distributed Objects**

**Lab 2 Assignment: Using RabbitMQ**

Table of Contents

[Introduction 2](#_Toc415656316)

[Login Service: 4](#_Toc415656317)

[Sign Up 5](#_Toc415656318)

[Login Fail 6](#_Toc415656319)

[Sign In: 8](#_Toc415656320)

[Sign Out: 9](#_Toc415656321)

[Profile Service: 10](#_Toc415656322)

[Edit Profile 10](#_Toc415656323)

[View Profile: 12](#_Toc415656324)

[Member Service: 15](#_Toc415656325)

[Show Connections: 15](#_Toc415656326)

[Search Member 16](#_Toc415656327)

[Send Invitation to connect 16](#_Toc415656328)

# Introduction

LinkedIn is a business-oriented social networking service used for professional networking.

In this lab, I have developed a Message Oriented LinkedIn Application using a RabbitMQ Provider. RabbitMQ is open source message broker software that implements the Advanced Message Queuing Protocol (AMQP).

New node app is started with **package.json** file; I will be using express server and amqp module as my RabbitMQ Client. My package.json file looks like below:

{

"name": "Linkedin\_RabbitMQ\_Server",

"version": "0.0.1",

"private": **true**,

"scripts": {

"start": "node app.js"

},

"dependencies": {

"express": "3.2.6",

"ejs": "\*"

}

}

Then I have created a server.js file that will house the entirety of my app. I have initially declared my dependencies at the top and setting up my server.

**var** amqp = require('amqp')

, util = require('util');

http.createServer(app).listen(app.get('port'), **function**(){

console.log('Express server listening on port ' + app.get('port'));

});

I have then created a server connection for overall RabbitMQ server connection, the connection to my message exchange and the connection to my messaging queue. I have created **3 queues for Login Service, Profile Service and Member service** in my **server.js file.**

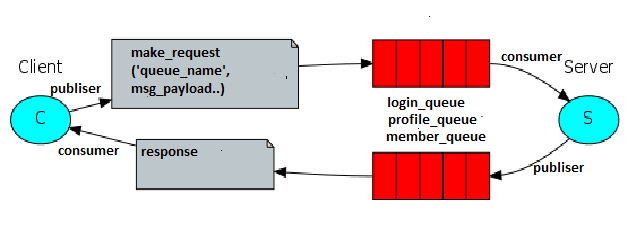
**var** connection = amqp.createConnection({host:'127.0.0.1'});

connection.on('ready', **function**(){

console.log("listening on login\_queue,profile\_queue,member\_queue");

connection.queue('login\_queue', **function**(q){

For each queue created, each task is delivered to multiple consumers known as “publish/subscribe” pattern. In the messaging model in RabbitMQ , the producer doesn’t send any messages directly to a queue. And the producer doesn’t even know if a message will be delivered to any queue at all.



Flow diagram of how messages are sent from producer and received by consumer

The messages are subscribed at the Server side and published at the client side.

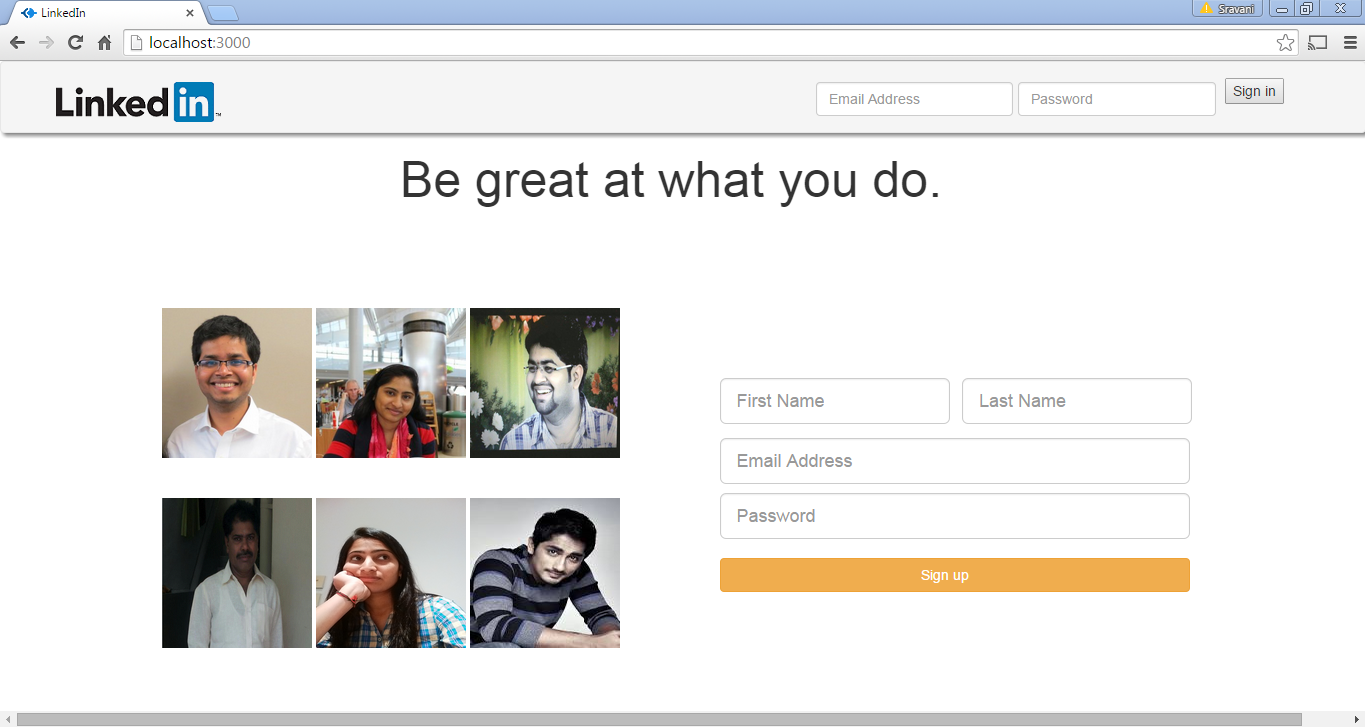
The Server performs the following tasks:

# Login Service:

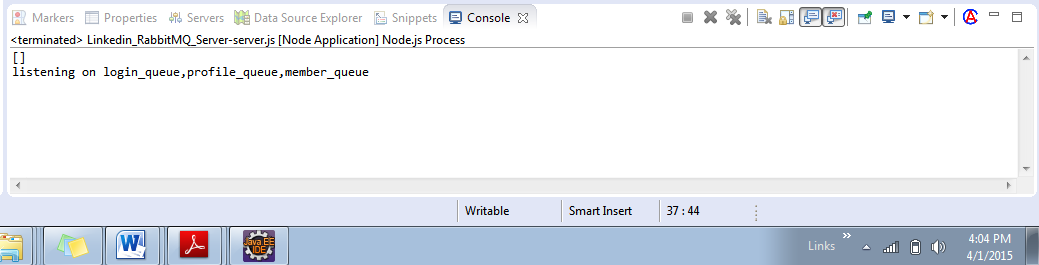
A message queue, ‘login\_queue’ is created so that the login service can access it. The ‘login\_queue’ carries the message payload and delivers it to the back end call back function.

The LinkedIn Client acts as the producer of the request and the back-end server acts as the consumer. When the request is processed, the back-end server provides the response onto the queues acting as the producer now and the client at the receiver end acts as a consumer.

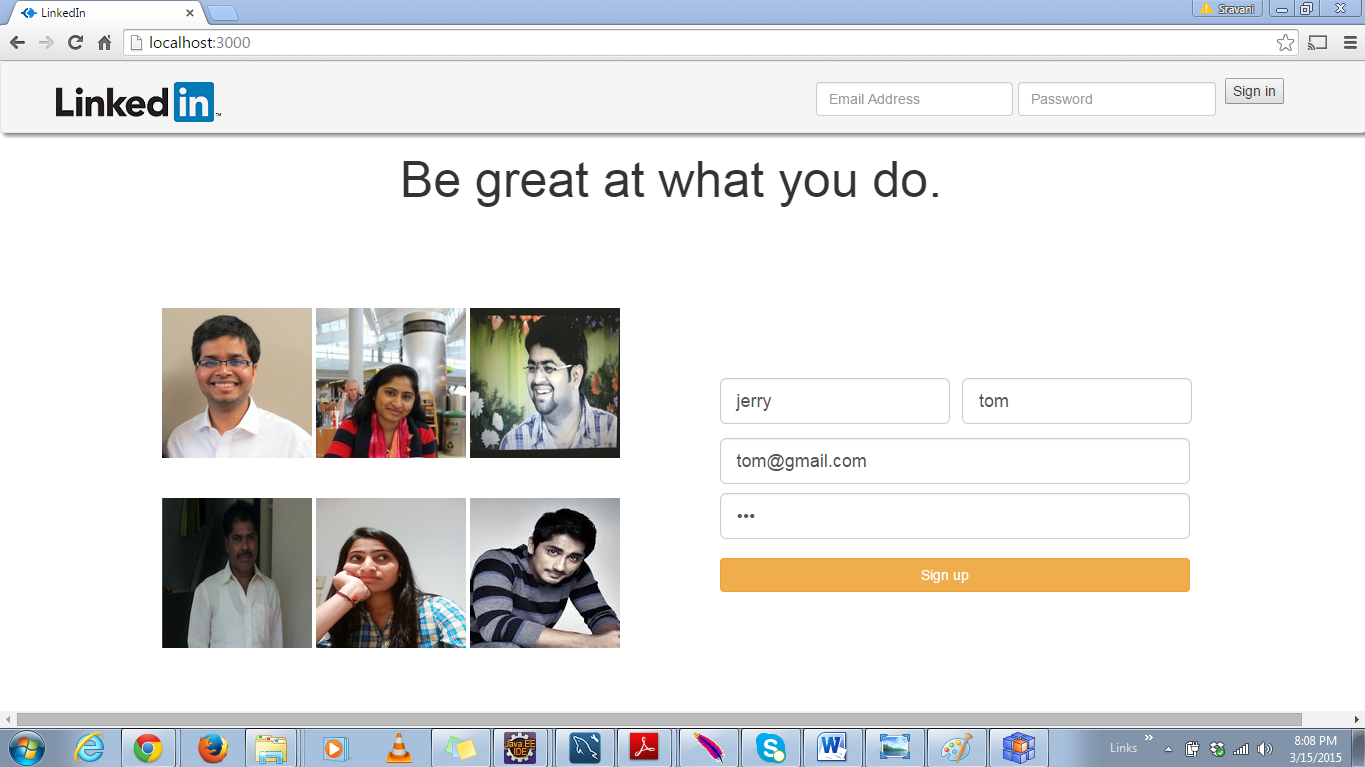
* The Sign In/Sign Up page of the Prototype of LinkedIn Application is as shown below.



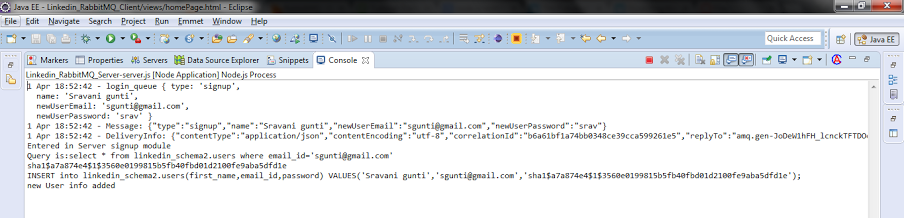
In the figure below we can see that the console of the server is in listening state with three different queues, that is, ‘login\_queue’, ‘member \_queue’ and ‘profile\_queue’ for 3 different services, that is, LogIn Service, Profile Service and Member Service respectively.



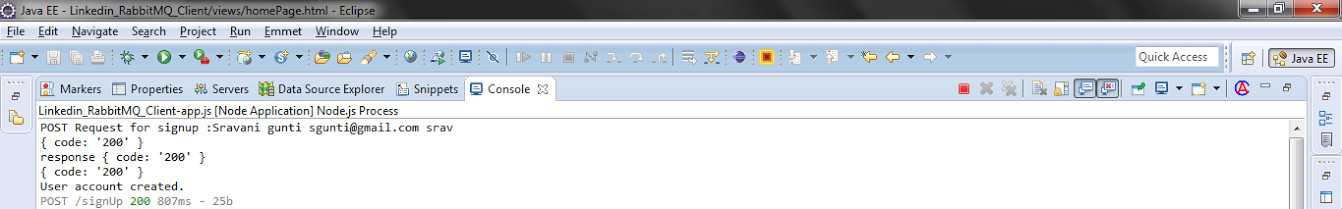
## Sign Up



* The figure below shows the console window of the server which contains the queue used i.e. ‘login\_queue’ along with the message payloads sent and queries executed.

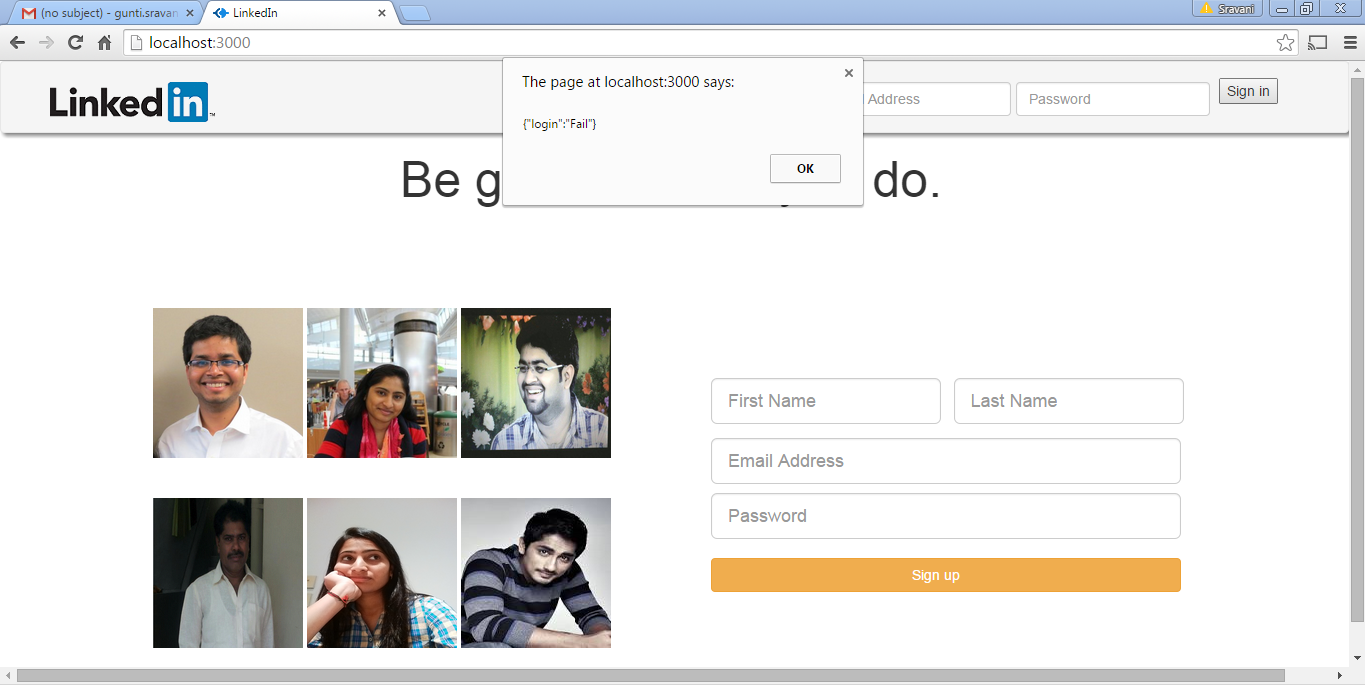


* The figure below shows the client console with the ‘signup’ request and response captured.

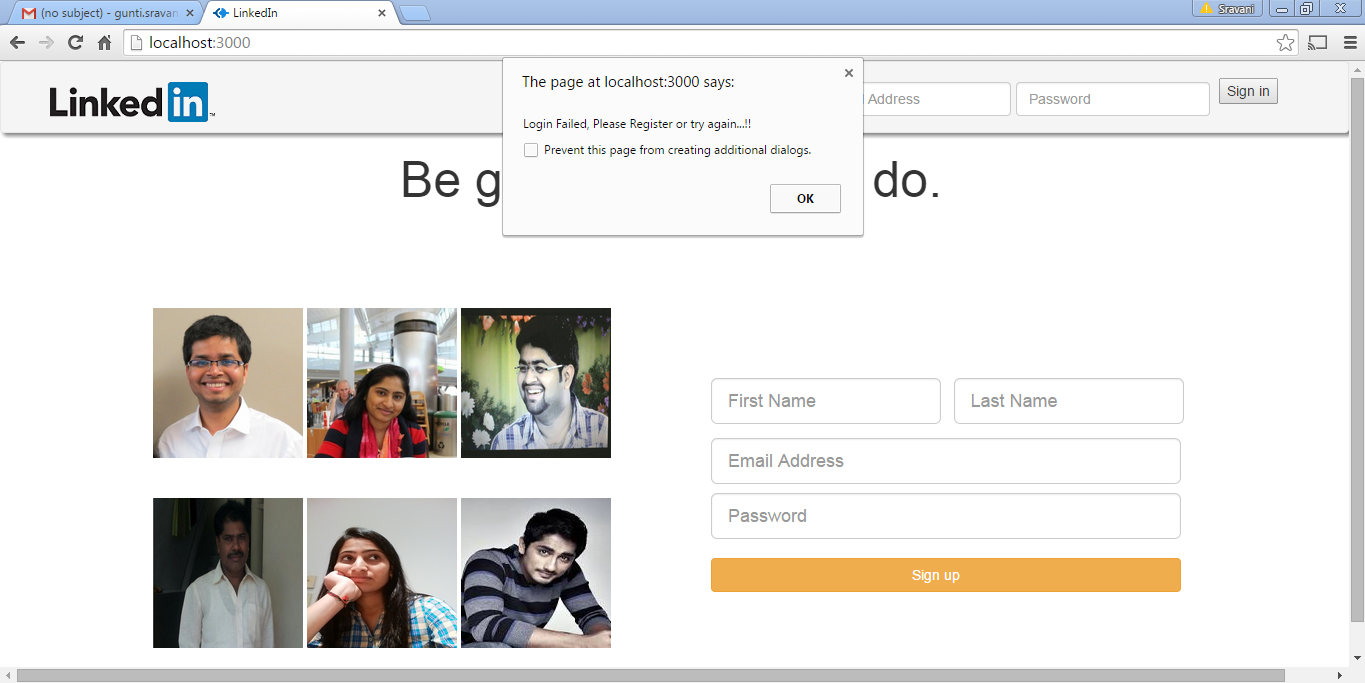


## Login Fail

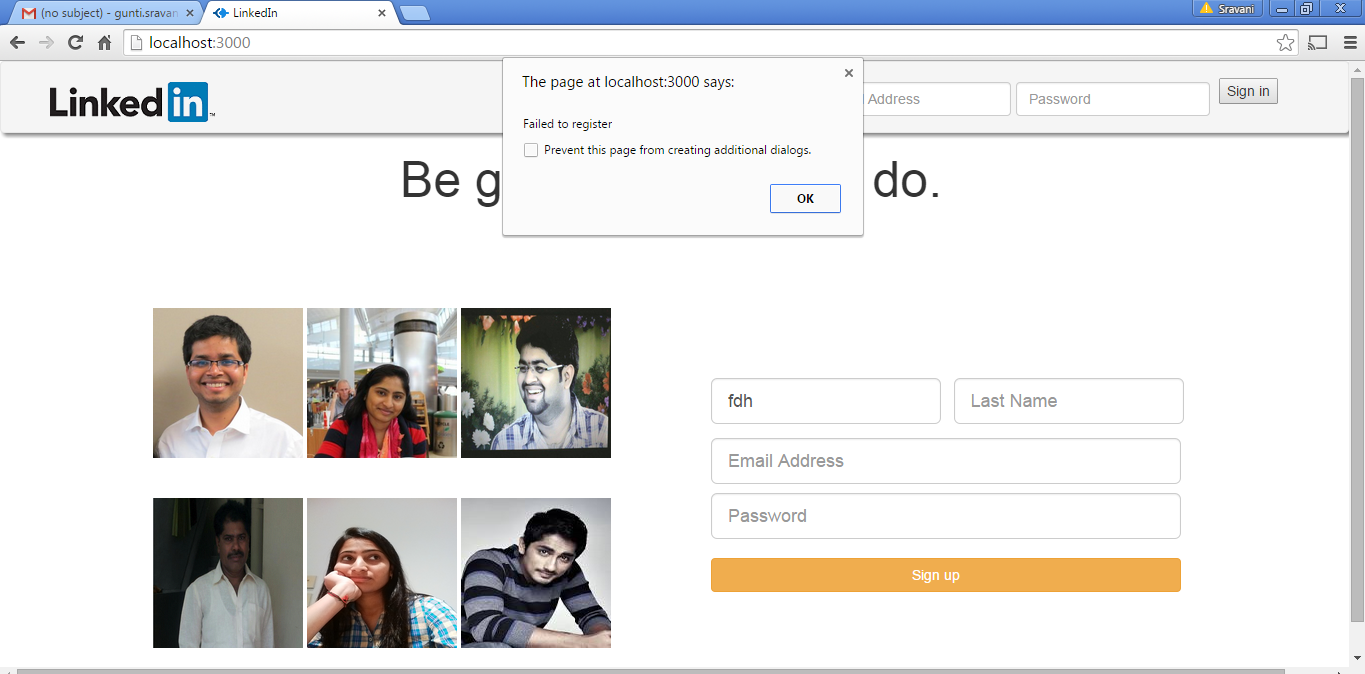
When no email address and password is specified, then an error alert is displayed on the screen as “Fail”.



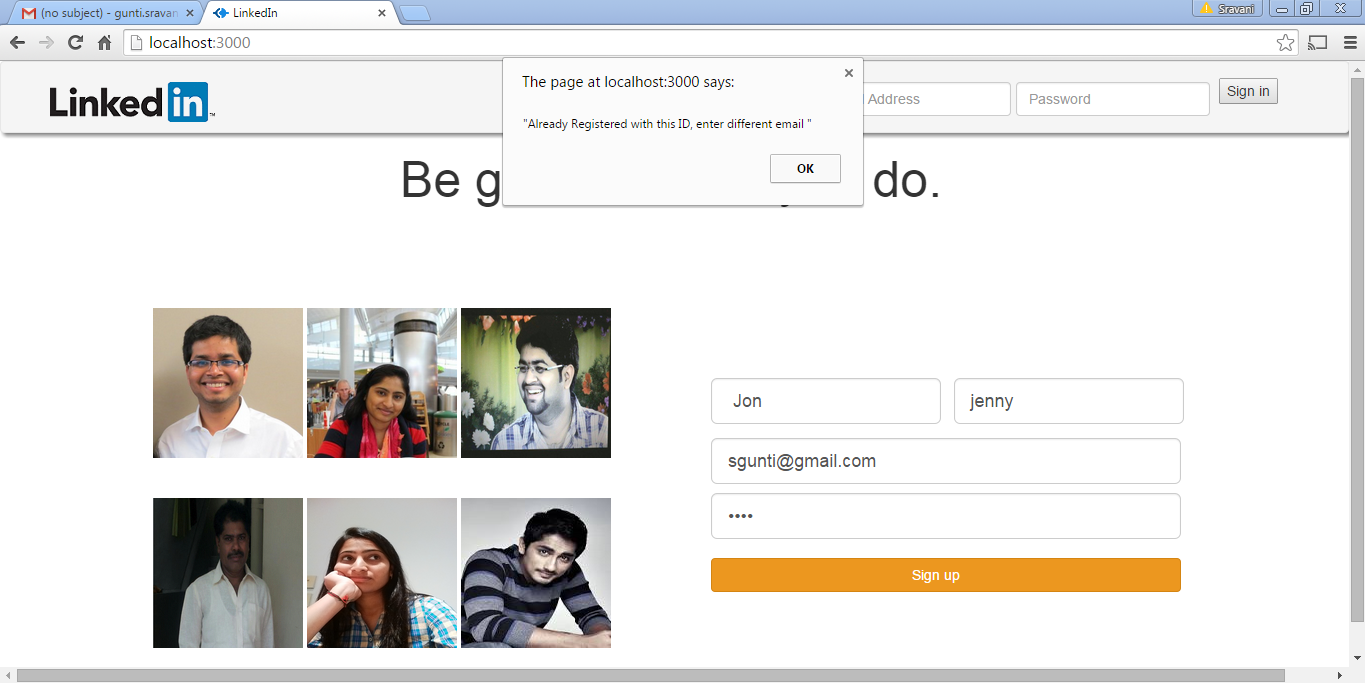
* When an invalid Id is mentioned, then an error message is displayed on the screen.



* While registering if no values are entered then an error message is displayed.

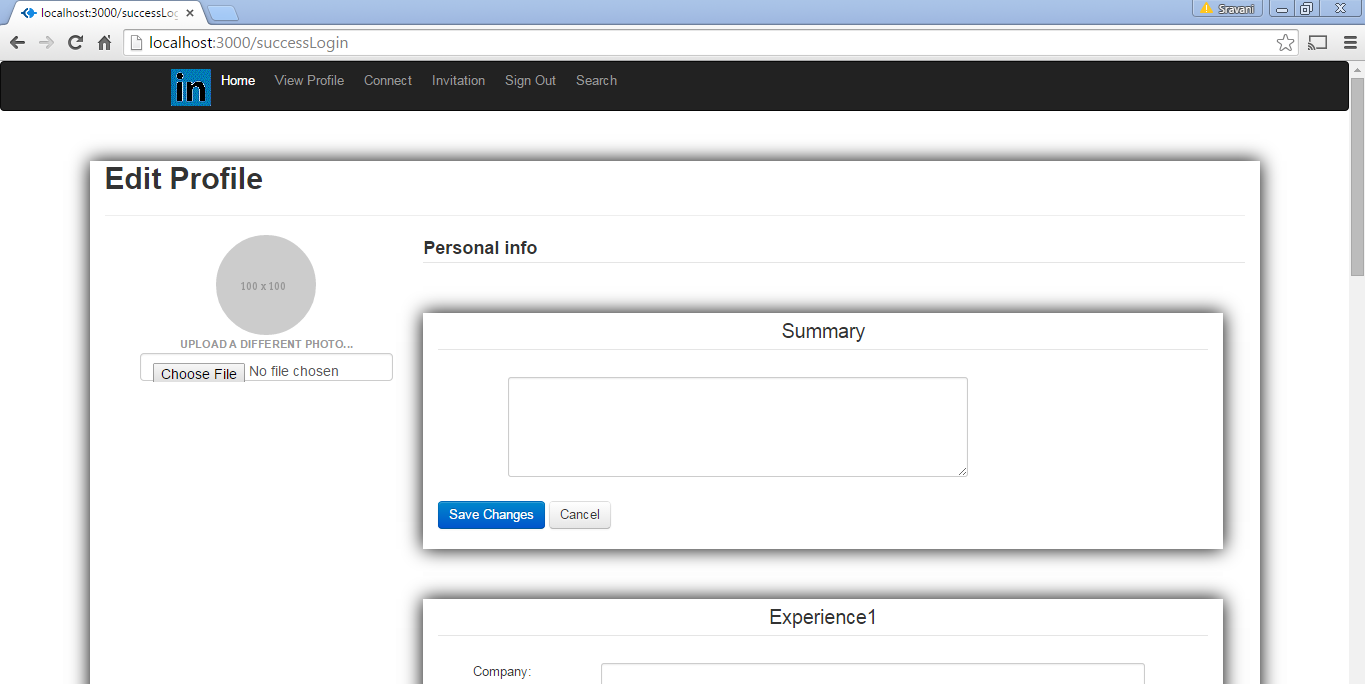
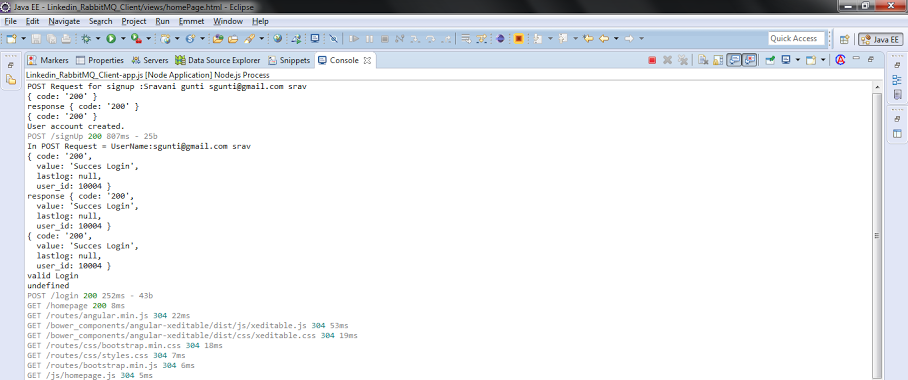


* If a user tries to sign up with the values of an existing user, then an error message pops up on the screen saying that “Already Registered with this ID, enter different mail”.

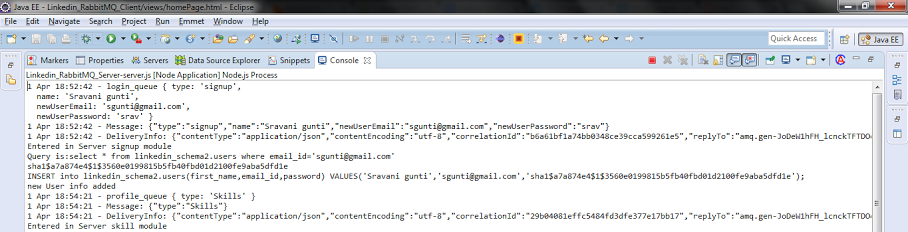


## Sign In:

When successfully logged in, then it navigates to the profile page.

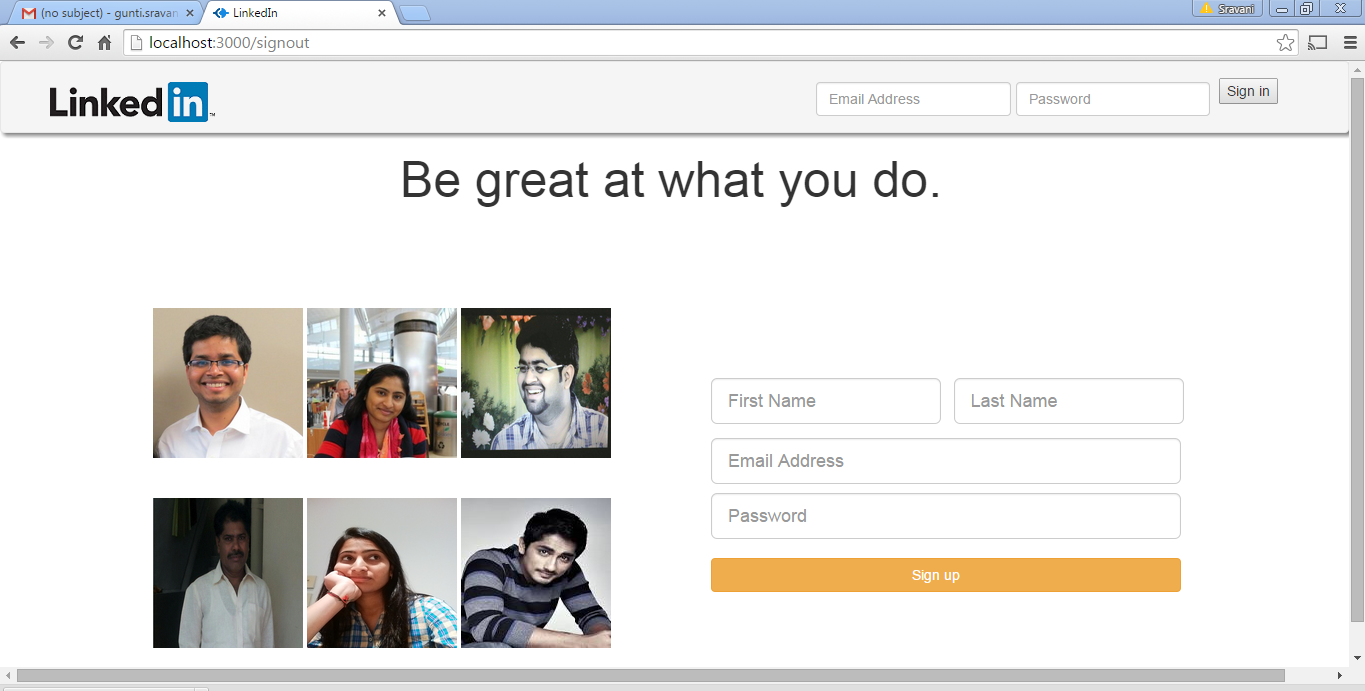


* The figure above shows the console of the client application displaying the request sent and the responses received.
* The figure below shows the server console stating the queue used, message payload sent in the ‘login\_queue’.



## Sign Out:

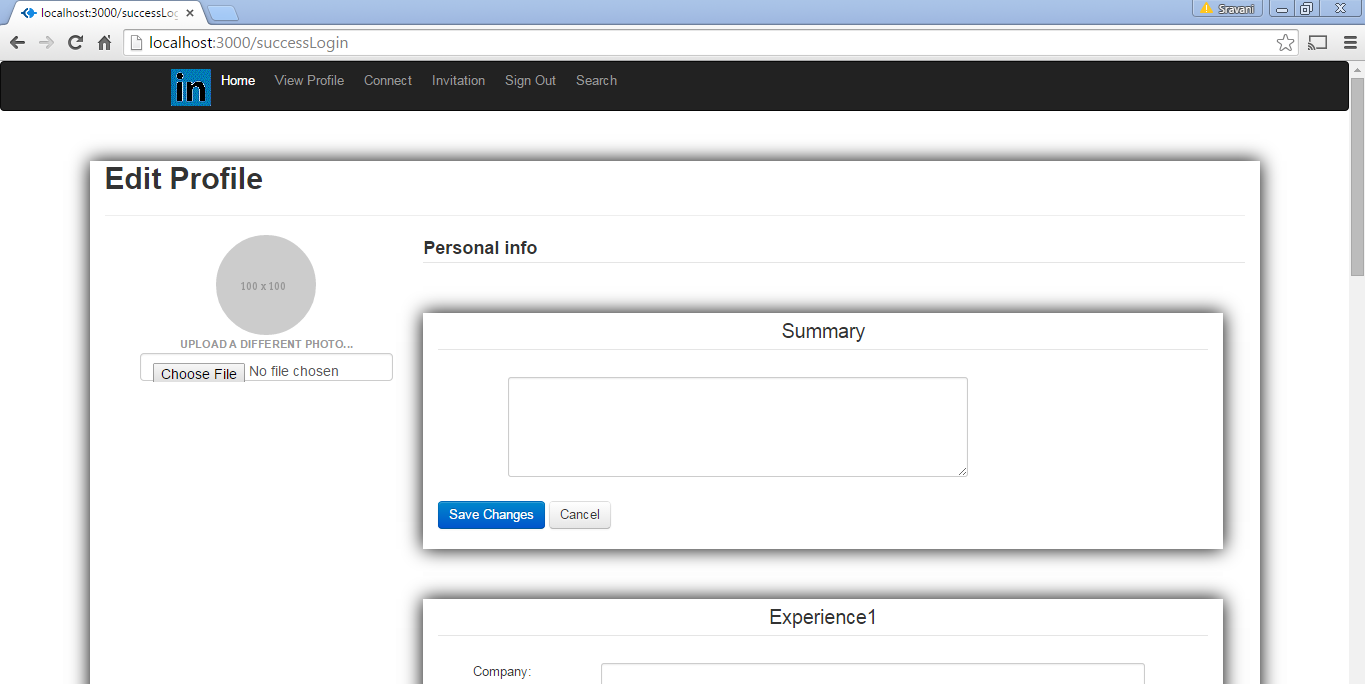
When the user signs out, then he is navigated back to the login page.

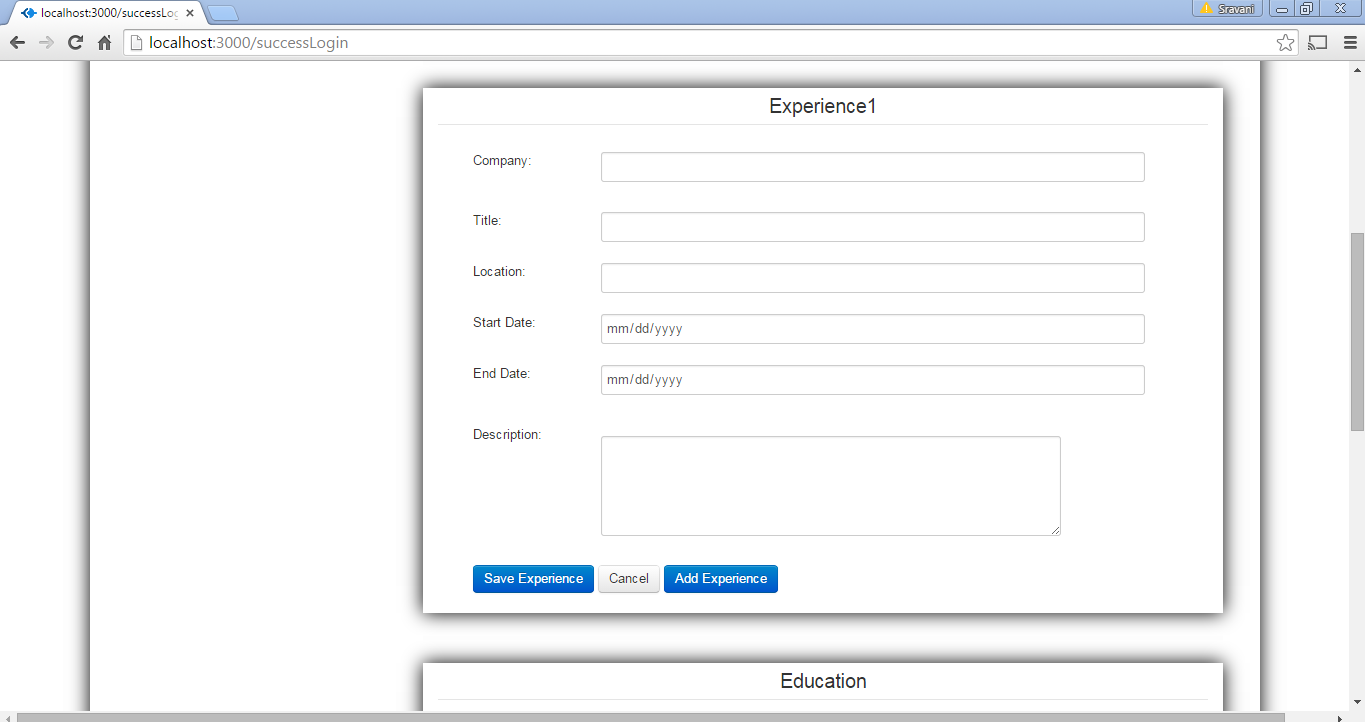


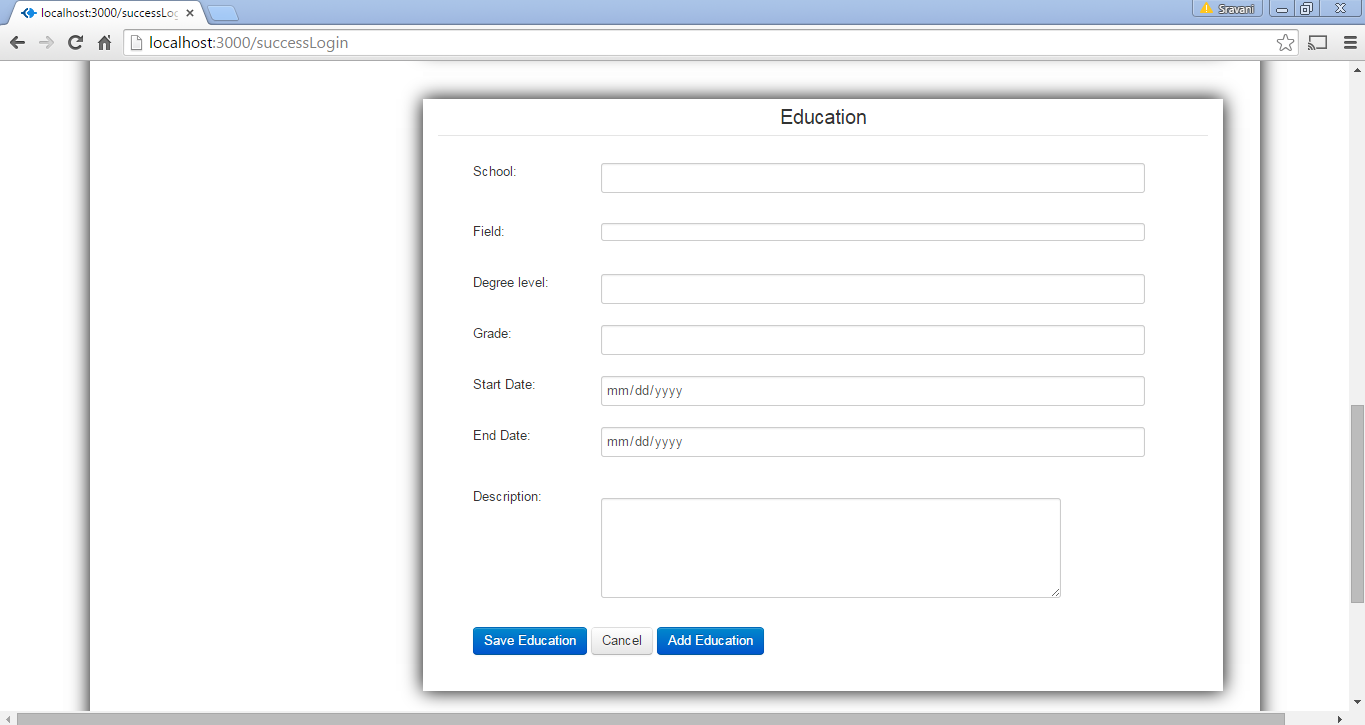
# Profile Service:

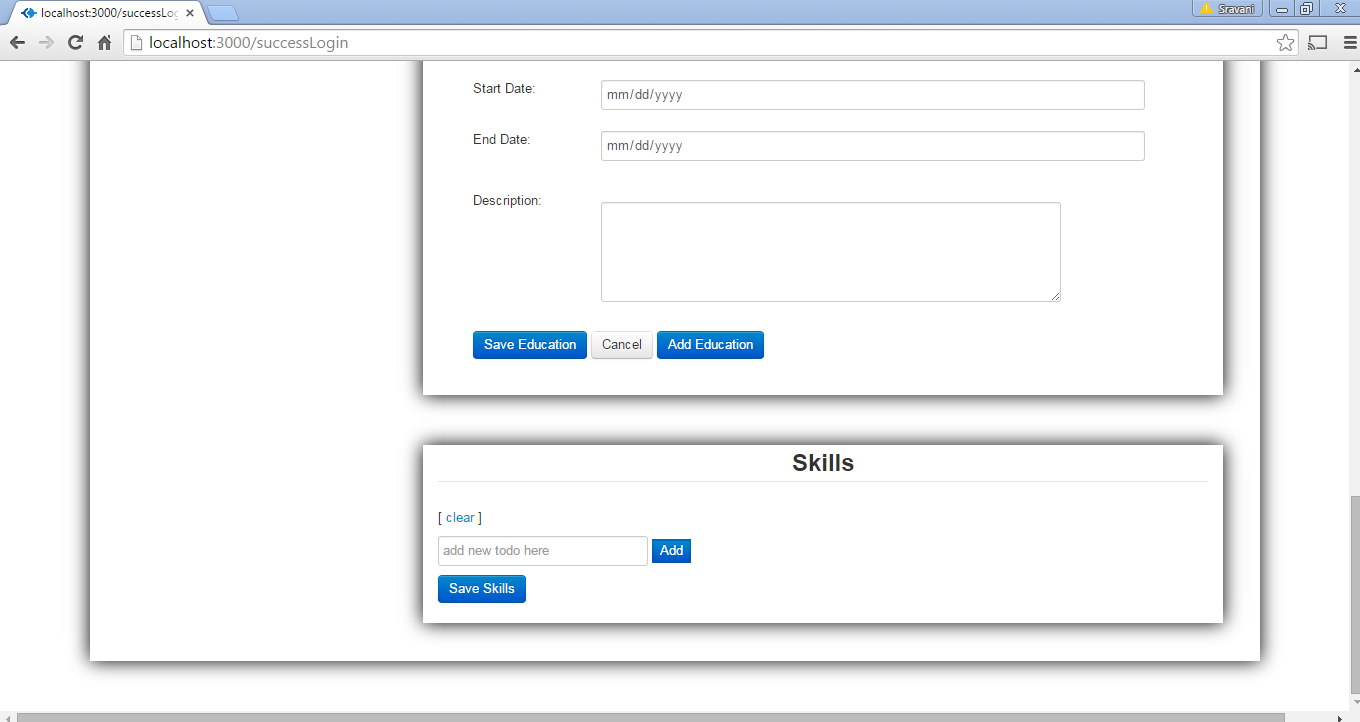
The message queue ‘profile\_queue’ is accessed by Profile service, which carries the message payloads and delivers it to the back end call back function. The Profile services are used to create user profile and edit user profile as shown below.

## Edit Profile

* After successful Sign Up, the user is navigated to the success Login page, where he can edit his profile like summary, education, experience and skills.
* 

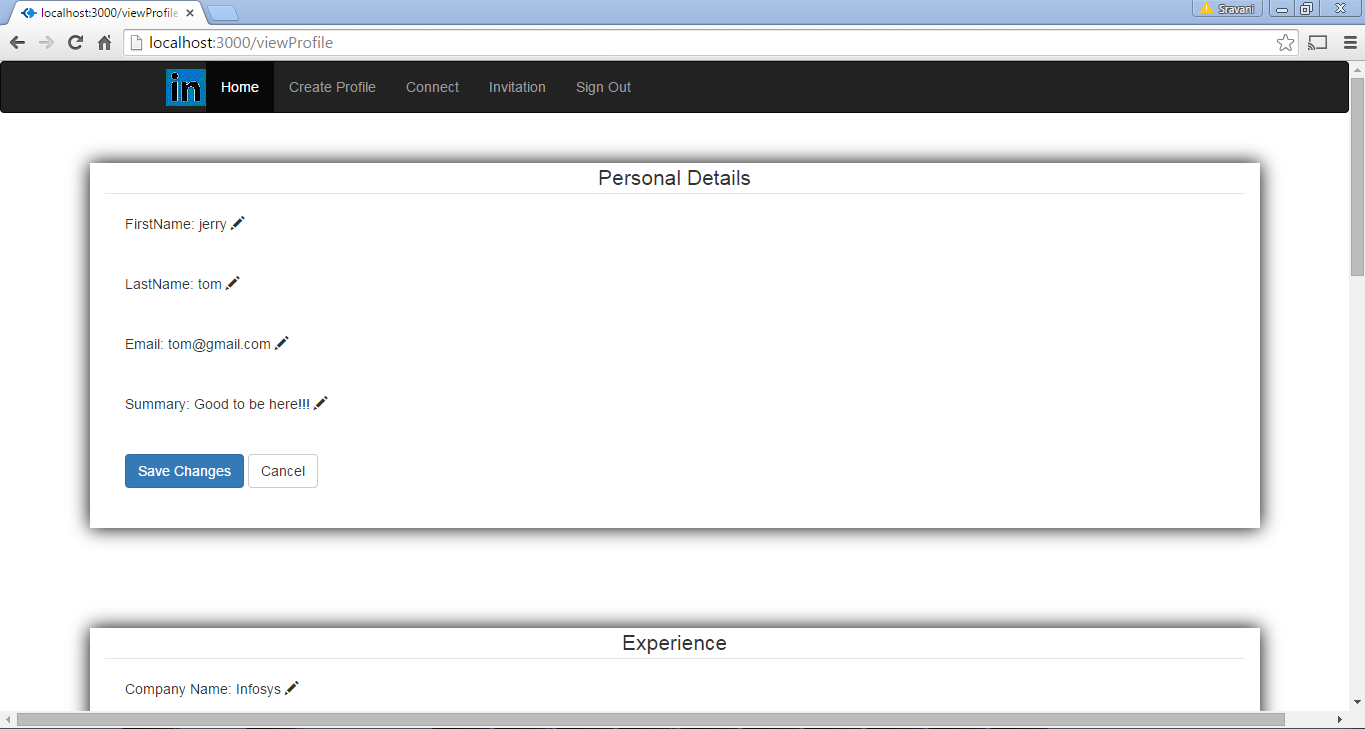


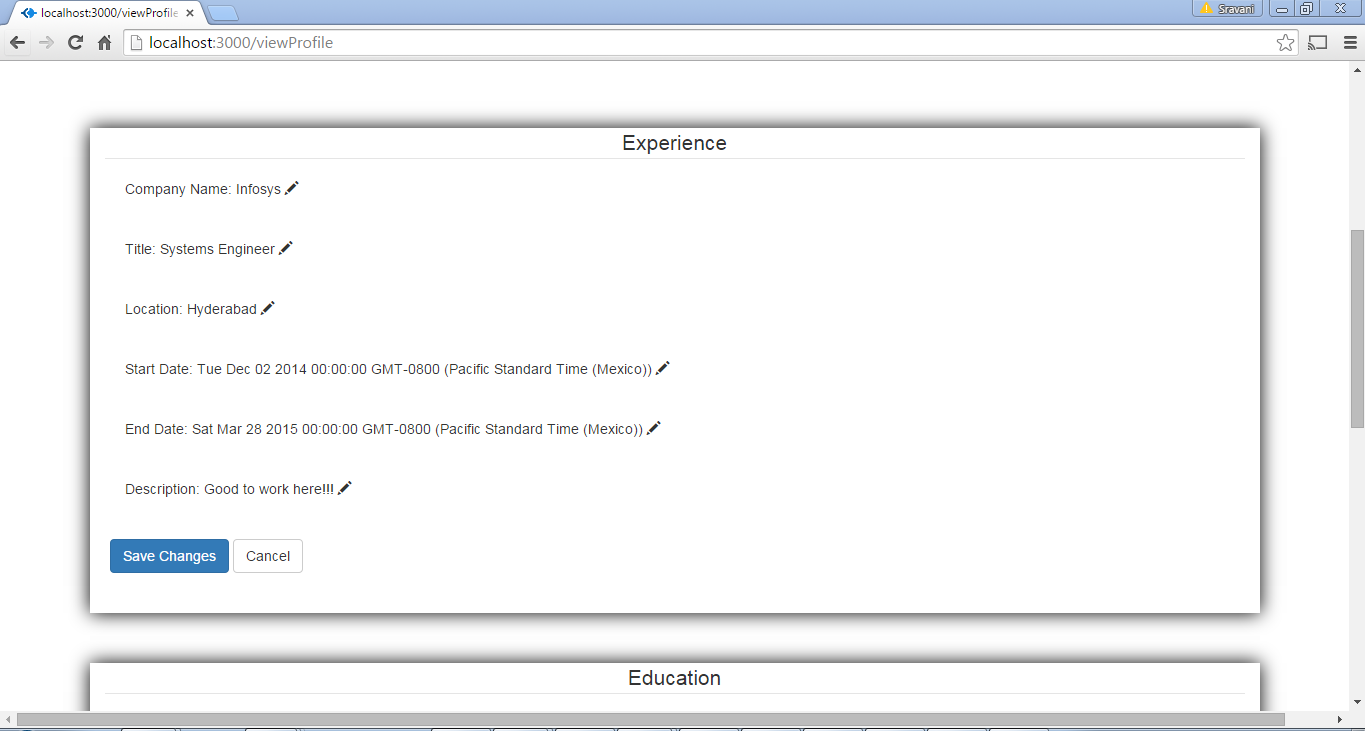


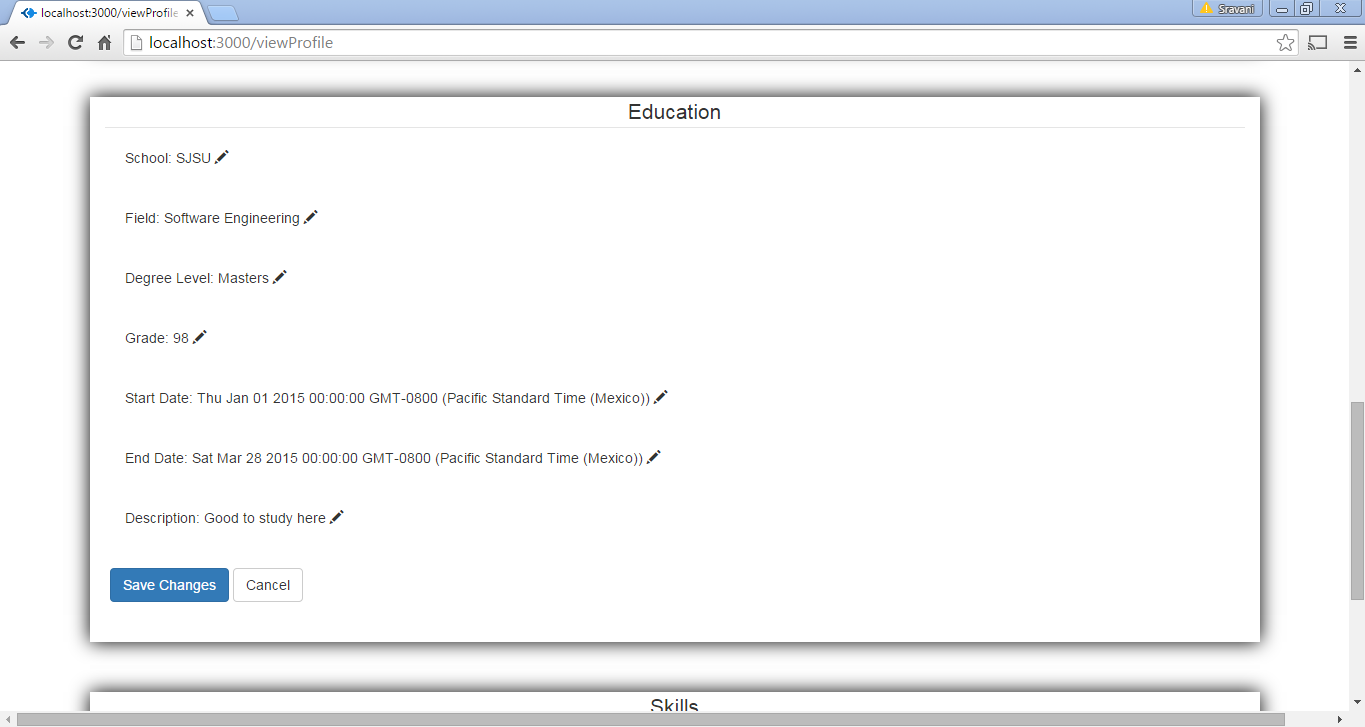


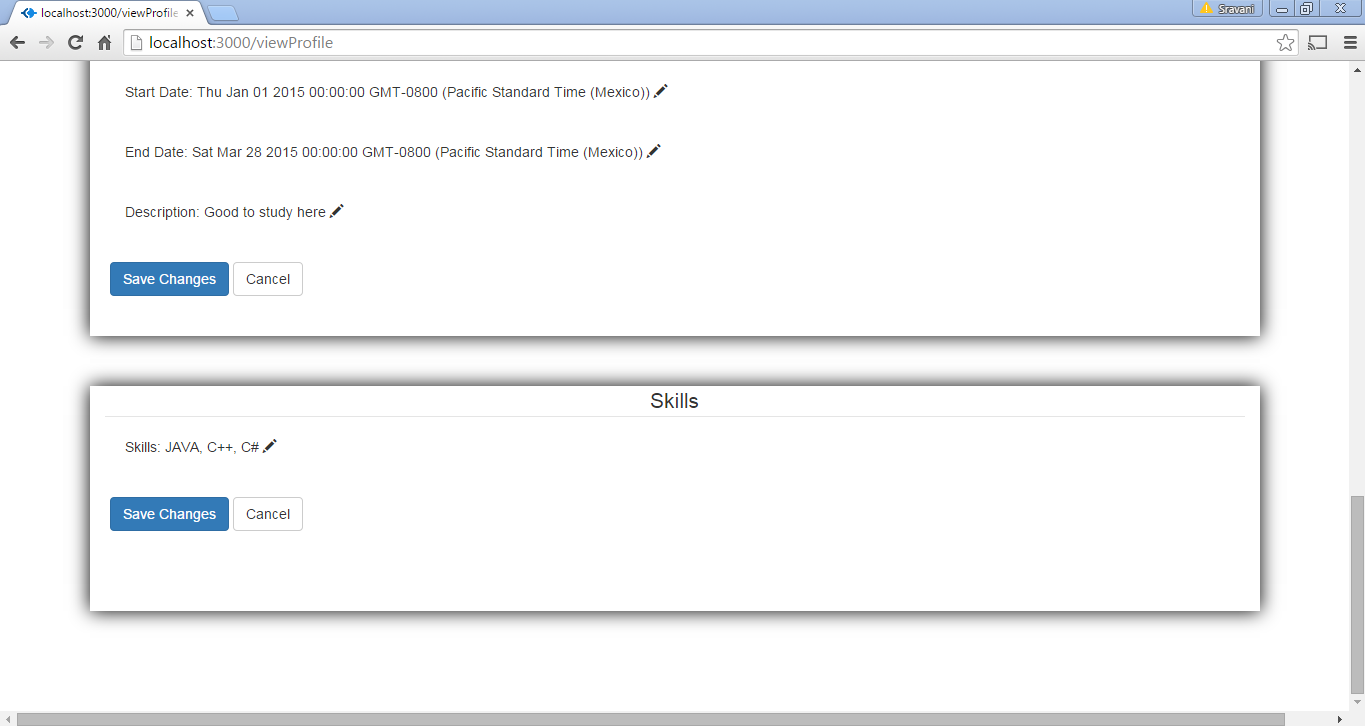
## View Profile:

* When successfully the data is saved into the database, then when Save Changes is clicked, it is navigated to View Profile Page.







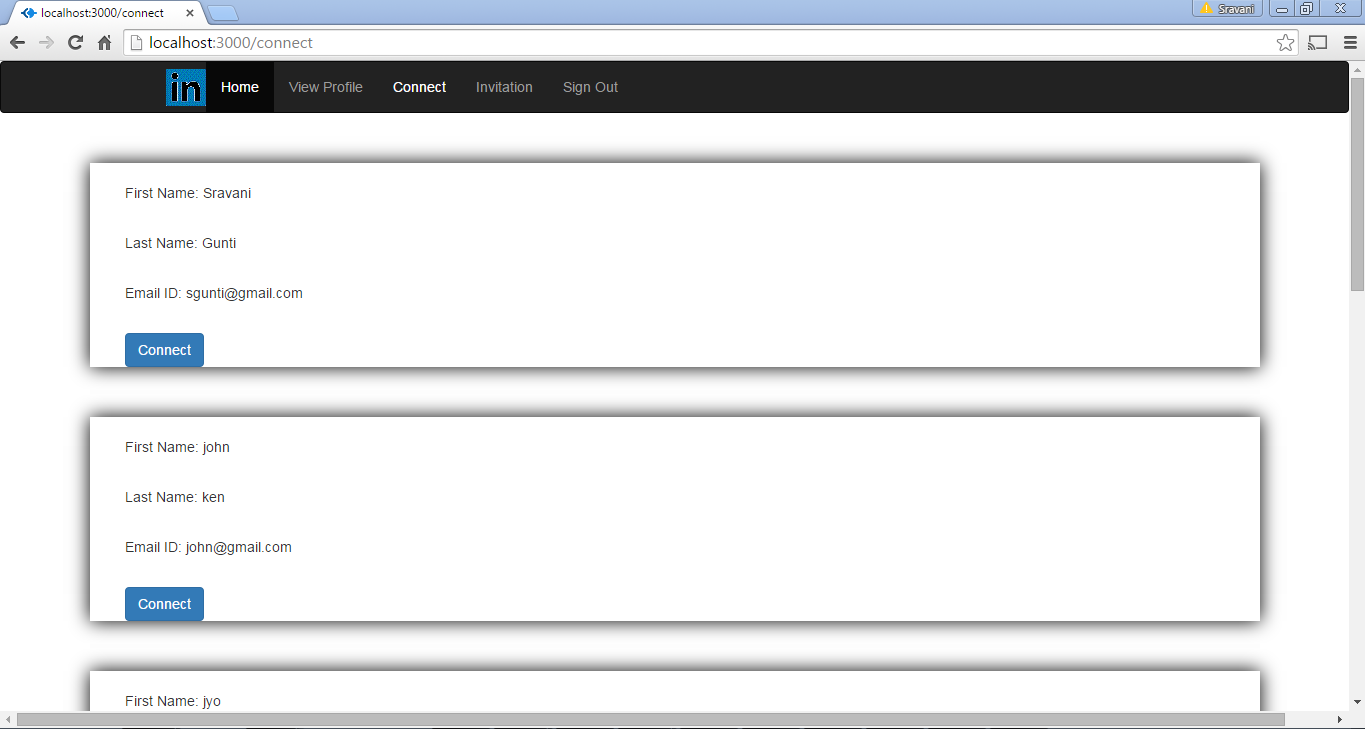


# Member Service:

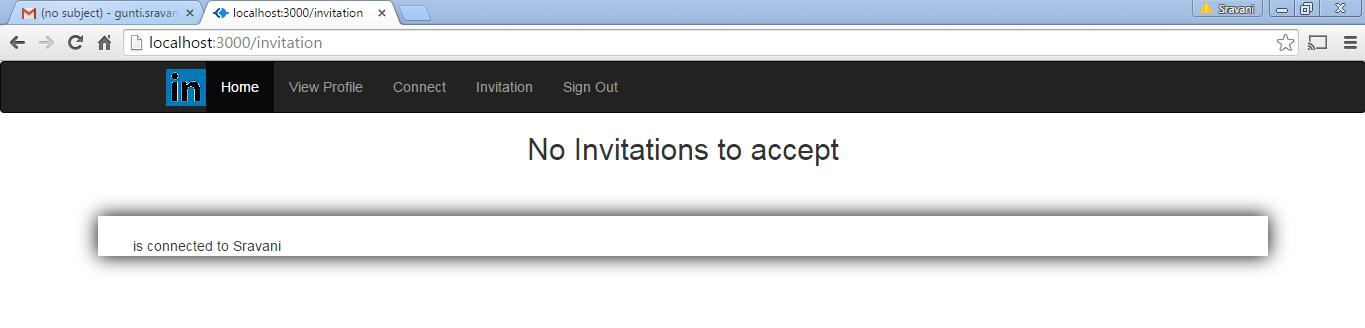
The message queue ‘member\_queue’ is accessed by Member service, which carries the message payloads and delivers it to the back end call back function. The Member services are used to search a member, send invitation for connections, accept the invitations and to show the connections.

## Show Connections:

* When Connect tab is clicked, the list of users of the linkedin application is displayed and the user can connect to the other users.

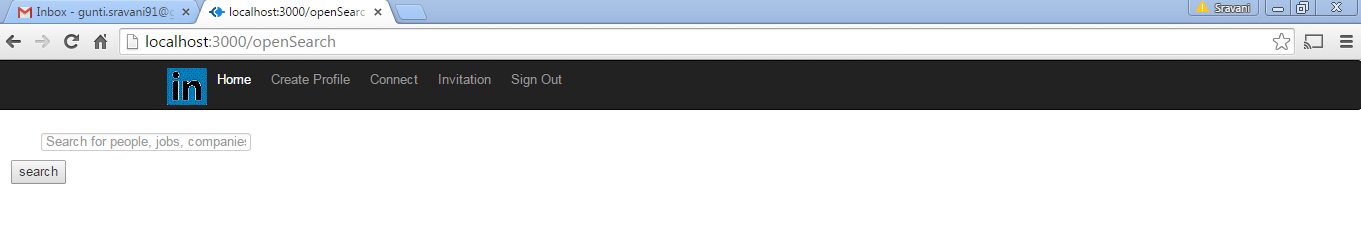


* In the invitations page, the users who have been connected to the current user is displayed.



## Search Member

* When the user wants to search a particular member from the list of linkedIn users, then Search tab is clicked.
* User can either enter first name, last name or email address of any user



## Send Invitation to connect

* List of users of LinkedIn application are displayed.

