**CMPE 273 – Lab 3**

**Sricheta Ruj (SJSU ID- 012527606)**

**Git URL (**[**https://github.com/srichetaruj/Freelancer-Prototype-Lab3-SpringBoot.git**](https://github.com/srichetaruj/Freelancer-Prototype-Lab3-SpringBoot.git)**)**

**Goal:** To Develop Freelancer like scalable web application developed in React JS, SQL and Spring Boot.

**Purpose of the system**: The purpose of the system is to develop a real-like Freelancer web application. The application helps professionals to get their work done by engineers simply by sitting at home. It has following features:

1. Signup as a Freelancer and as a Employer
2. Login
3. Set up profile
4. Dashboard (for Employer a well as Freelancer)
5. Post a Project (upload file)
6. Bid for a project
7. Home page View all open Projects

**Technology Used**:

**Client side:**

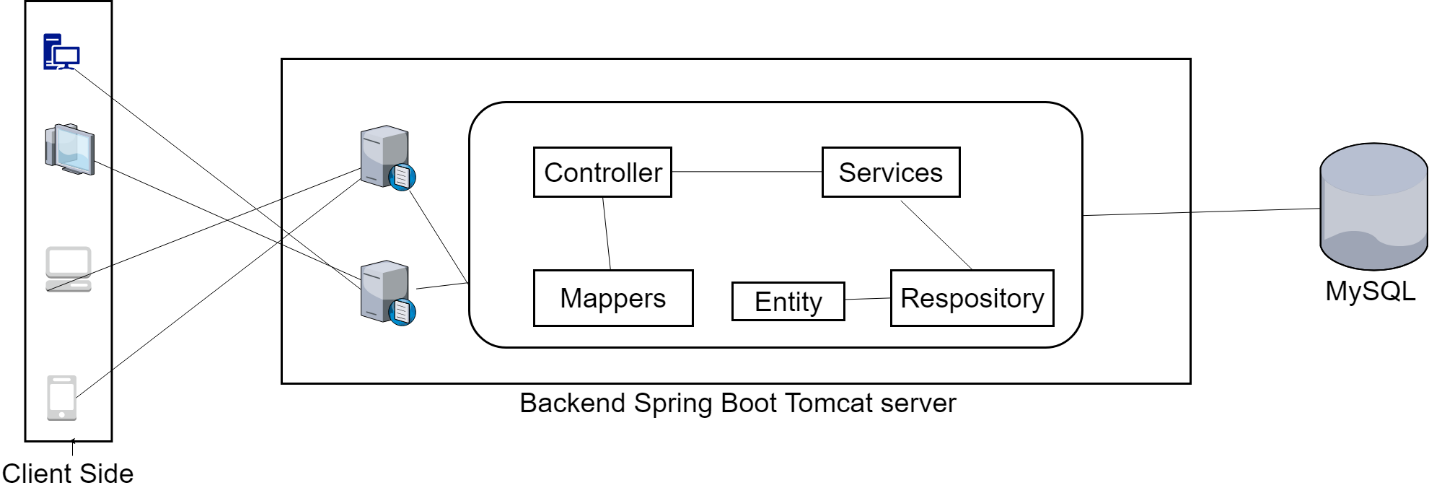
1. **React** as View in MVC architecture used reusable Components
2. **React-Bootstrap-Typeahead** to choose multiple skills
3. **Redux** for state maintenance.
4. Used **react-file-download** to interpret data stream.
5. **axios** to call the REST service.
6. Used **react-router-dom** to handle routing between different pages.
7. **React-StepZilla** to show progress view of complete profile page.
8. Used “**redux-thunk**” as middleware to support data flow between redux store and react components.

**Server side: -**

1. Used spring boot to create APIs
2. Used SQL as database
3. Used “Junit” ,“RestTemplate” and “SpringBootTest” for testing
4. “HttpSession” for maintaining sessions.
5. “MD5” for password encryption.

**Architecture of distributed system**: -

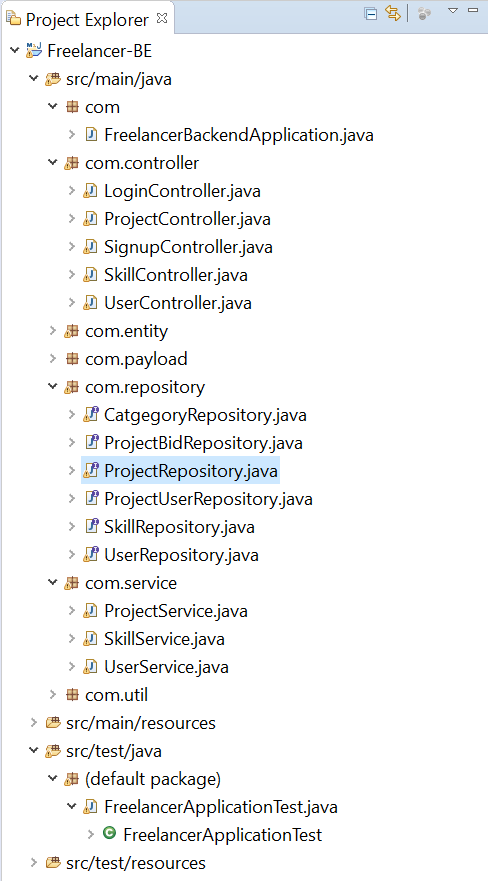
Controllers has all the the Uri’s mapped and if any URI gets hit the respective method gets called. This service fires the SQL JPA query using repository. I have used Spring data JPA.



**Code Flow:**

1. A Rest service is hit by the client which is React JS in our case.
2. The dispatcher servlet scans through the all mappings and maps to the one that gets called.
3. The Controller calls the service method.
4. The Service object calls the repository who is responsible to communicate with the MySQL database. I have used JPA for that.

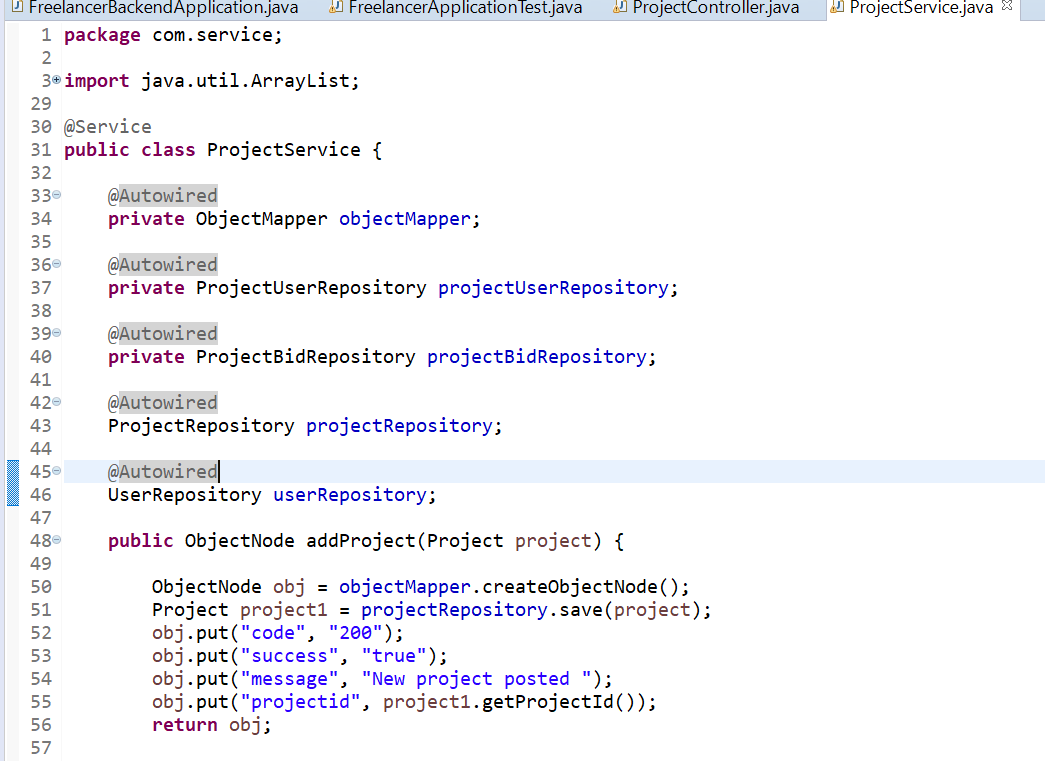
**Code Structure :**



**Controller :**



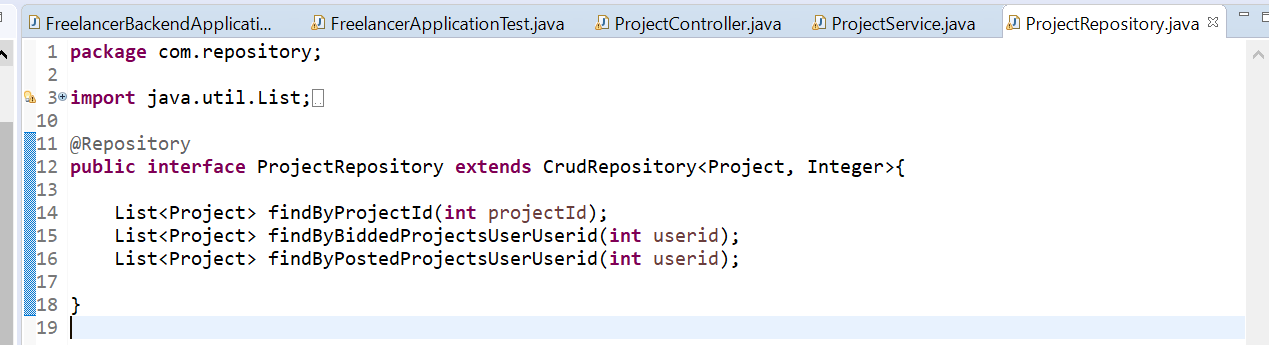
**Service :**



**File Upload Handling**



**Repository :**



**Entity class :**



**Password Encryption**

I used MessageDigest for password encryption. The *Java MessageDigest* class represents a cryptographic hash function which can calculate a message digest from binary data. When you receive some encrypted data you cannot see from the data itself whether it was modified during transportation.



**Database Design** : -

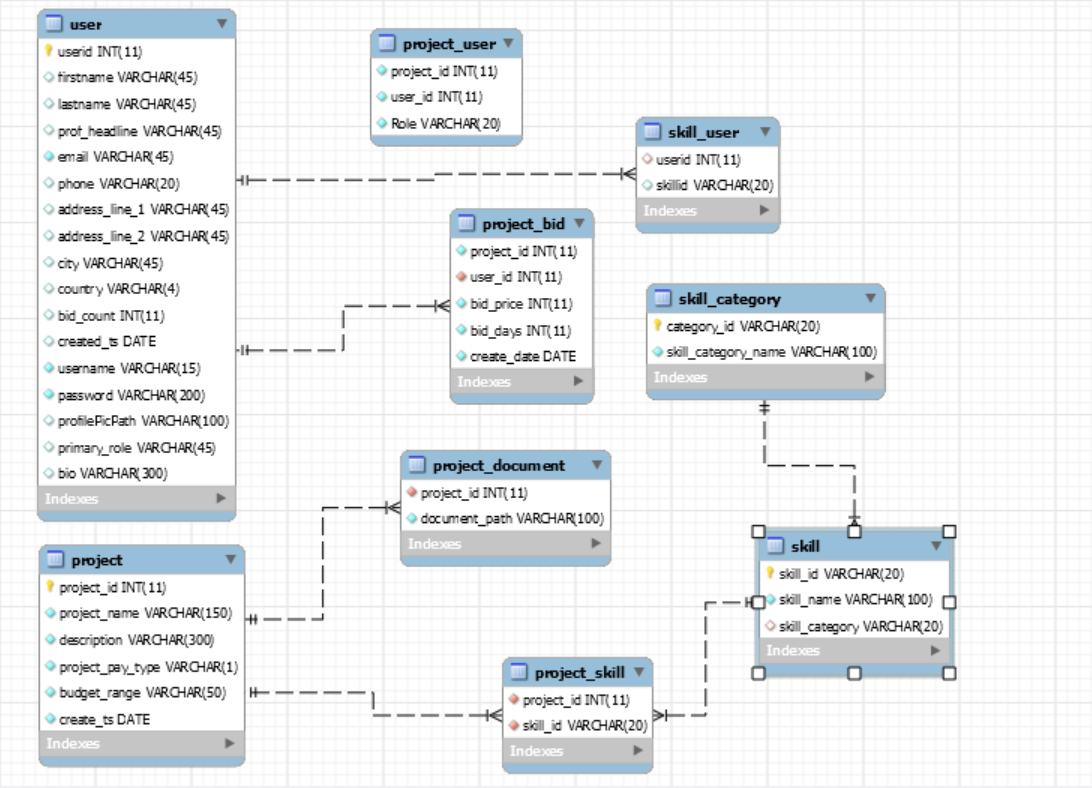
**Database used :** MySQL

**Tools used :** MySQL Workbench (Reverse Engineering to draw Database design diagram)

**Tables used:**

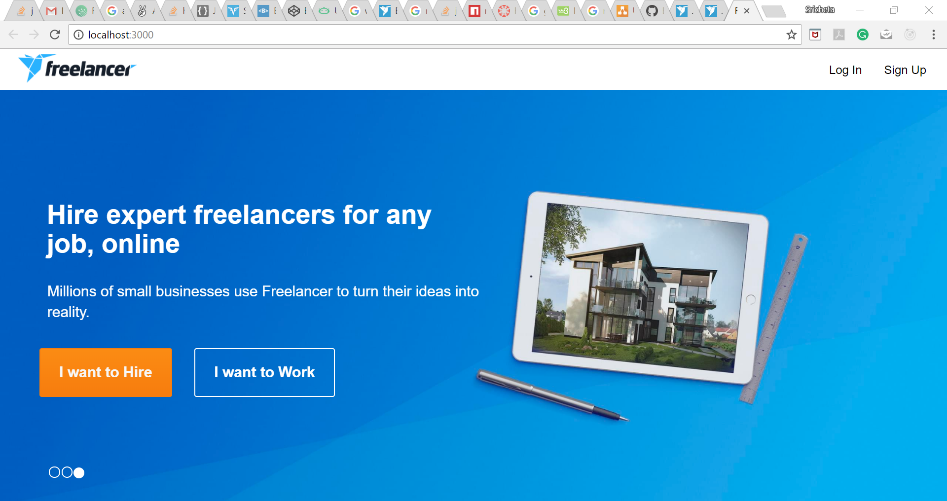
1. **User :** has all the information of a person to registers in the website
2. **project\_user :** has information of users associated with a project by what role (freelancer /Employer)
3. **skill\_user :** has information of users associated with a skill.
4. **project\_bid :** has information of users associated with a project on which they are interested.
5. **skill\_category :** has information of skills associated with a particular category.
6. **project\_document :** has information of files associated with project.
7. **skill :** has all the skills
8. **project\_skill :** has information of project associated with a skill.
9. **project :** has information of all the projects.

**The database design has been normalized up to 3NF.**

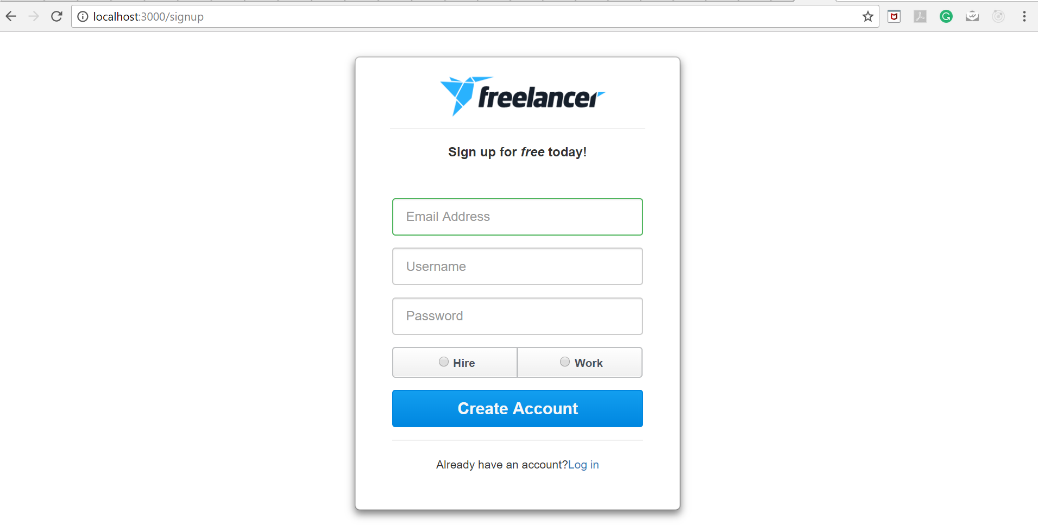


**Screenshots**

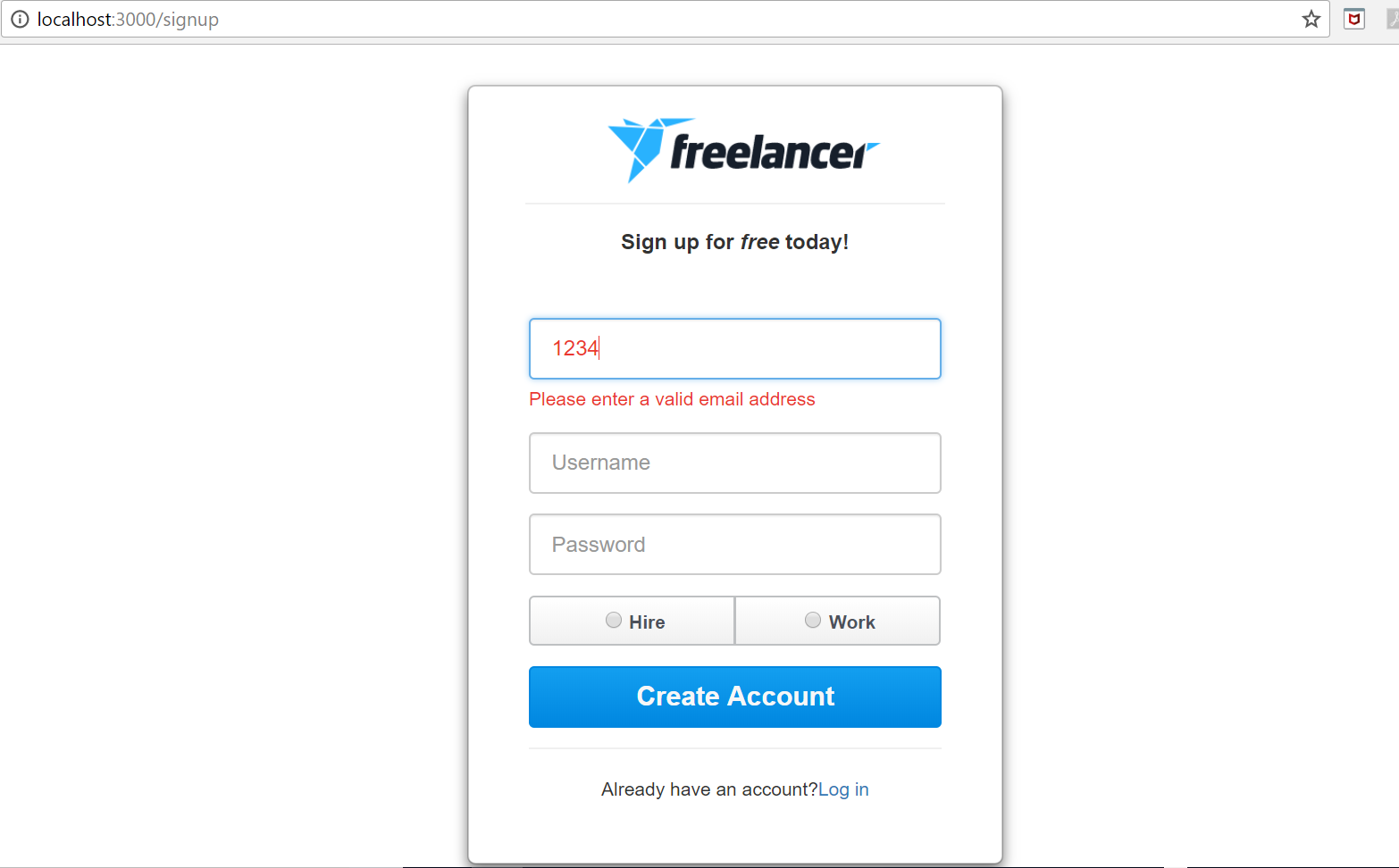
1. **Landing Page** has 2 options to login and signup



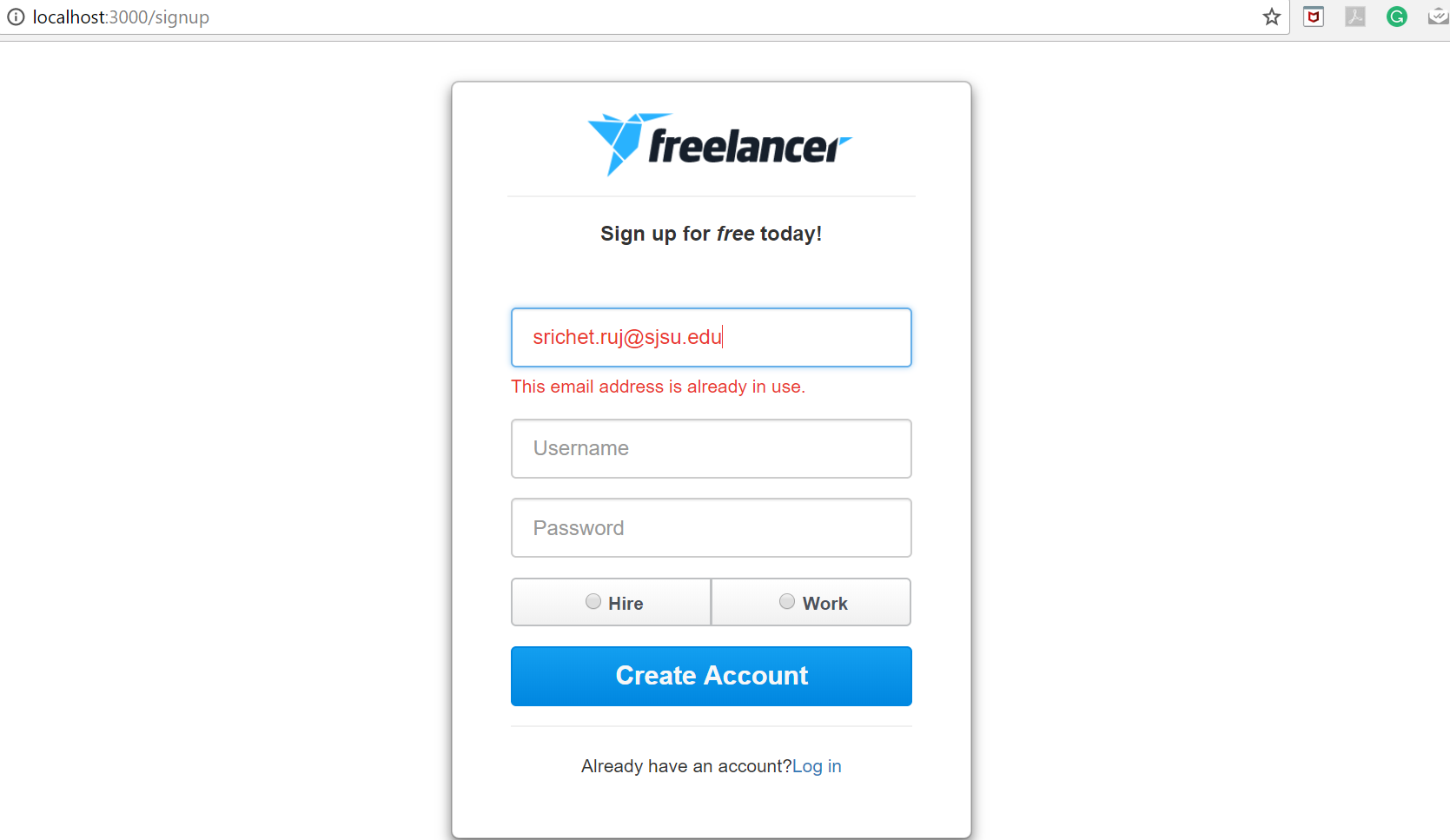
1. Signup page landing



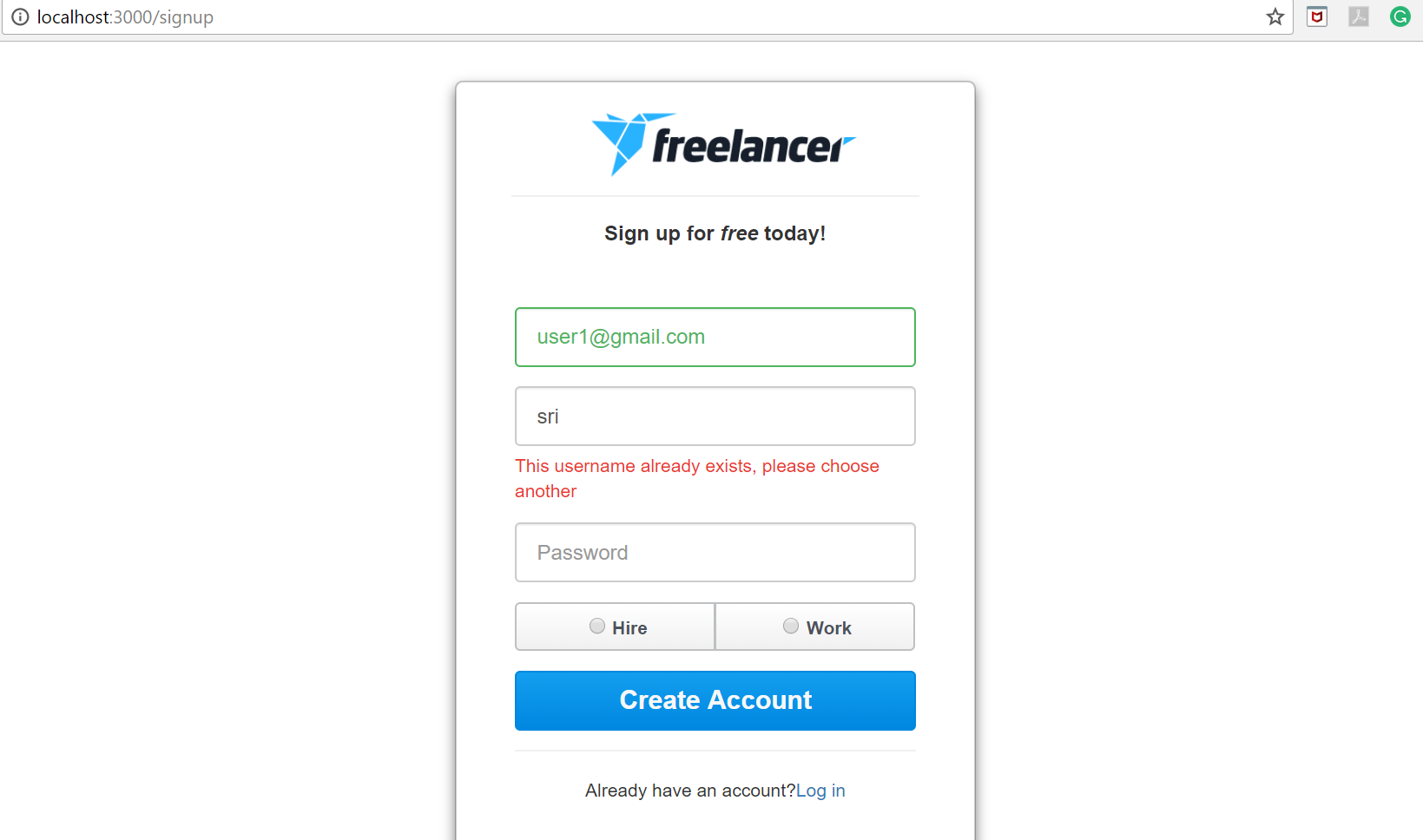
1. Signup page validation **(Invalid Email)**



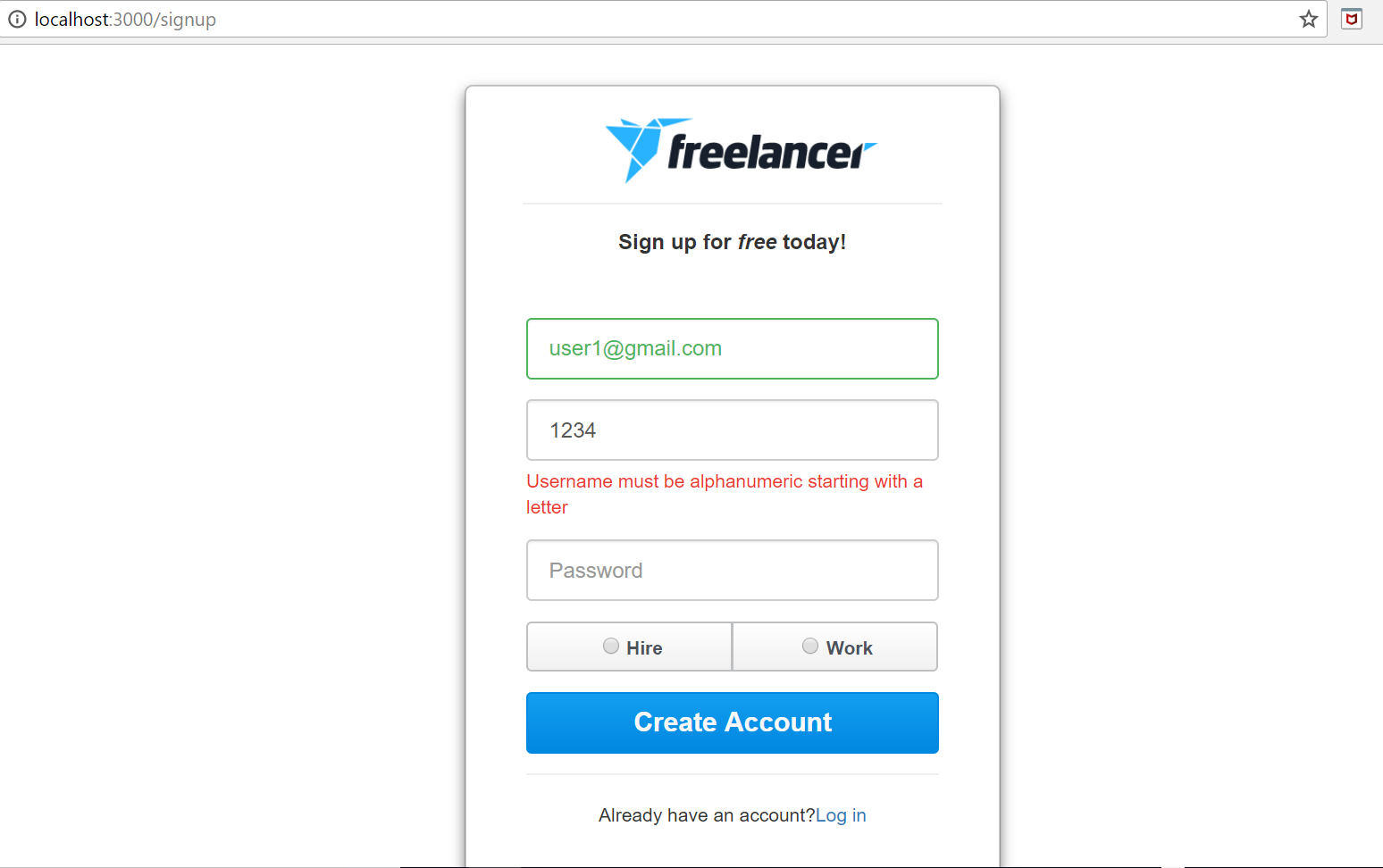
1. Signup page validation **(Email already in use)**



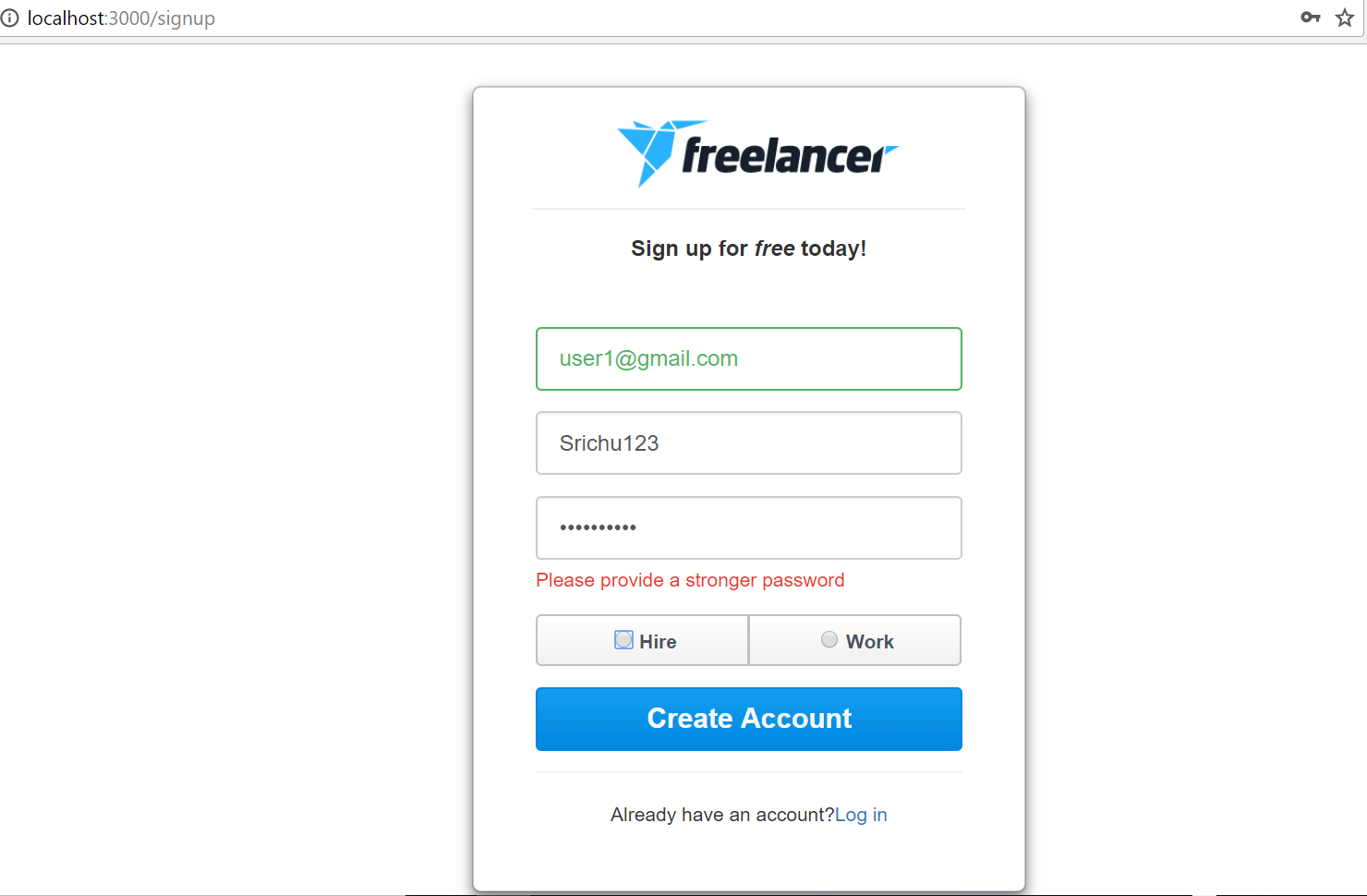
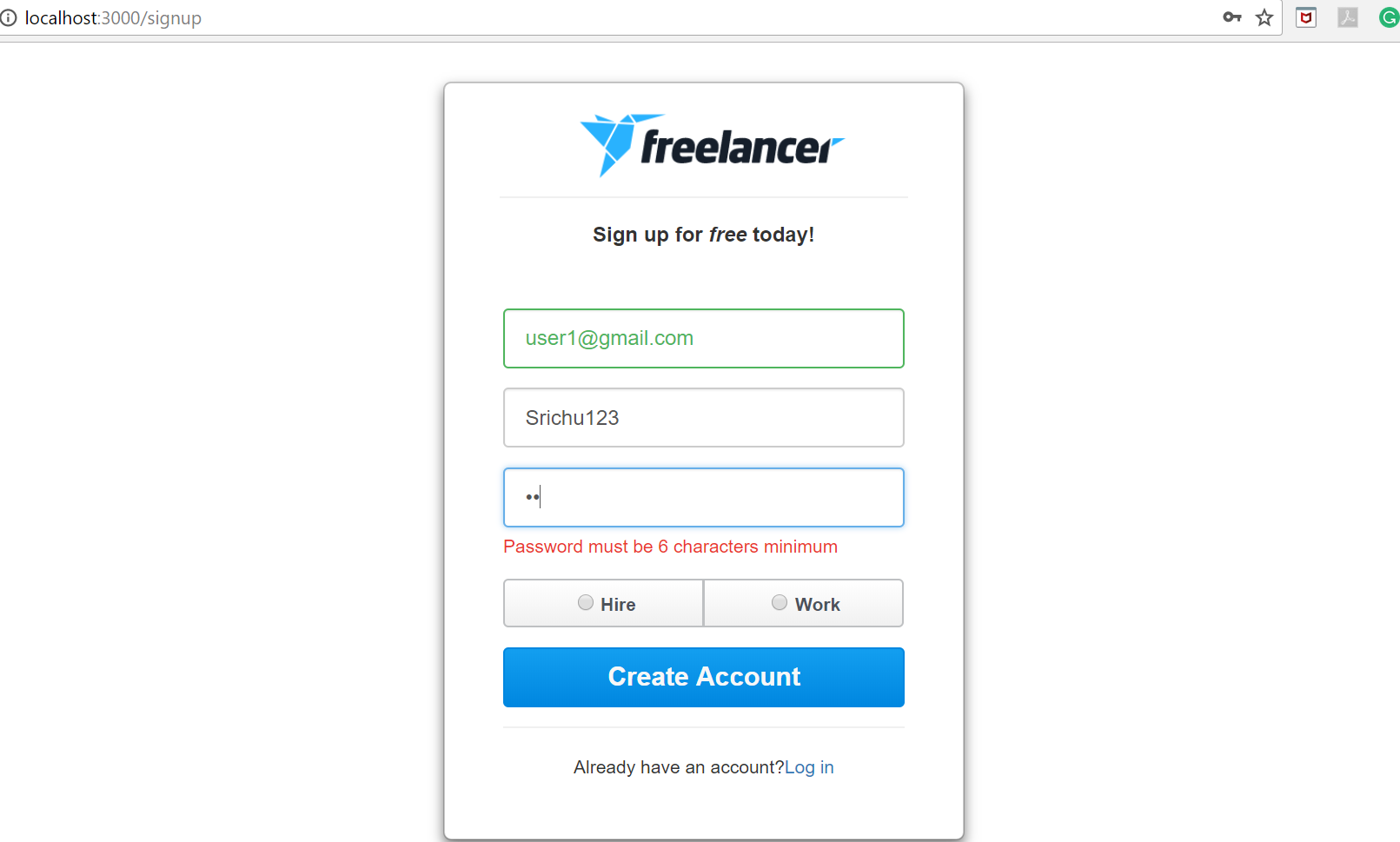
1. Signup page validation **(Username already in use)**



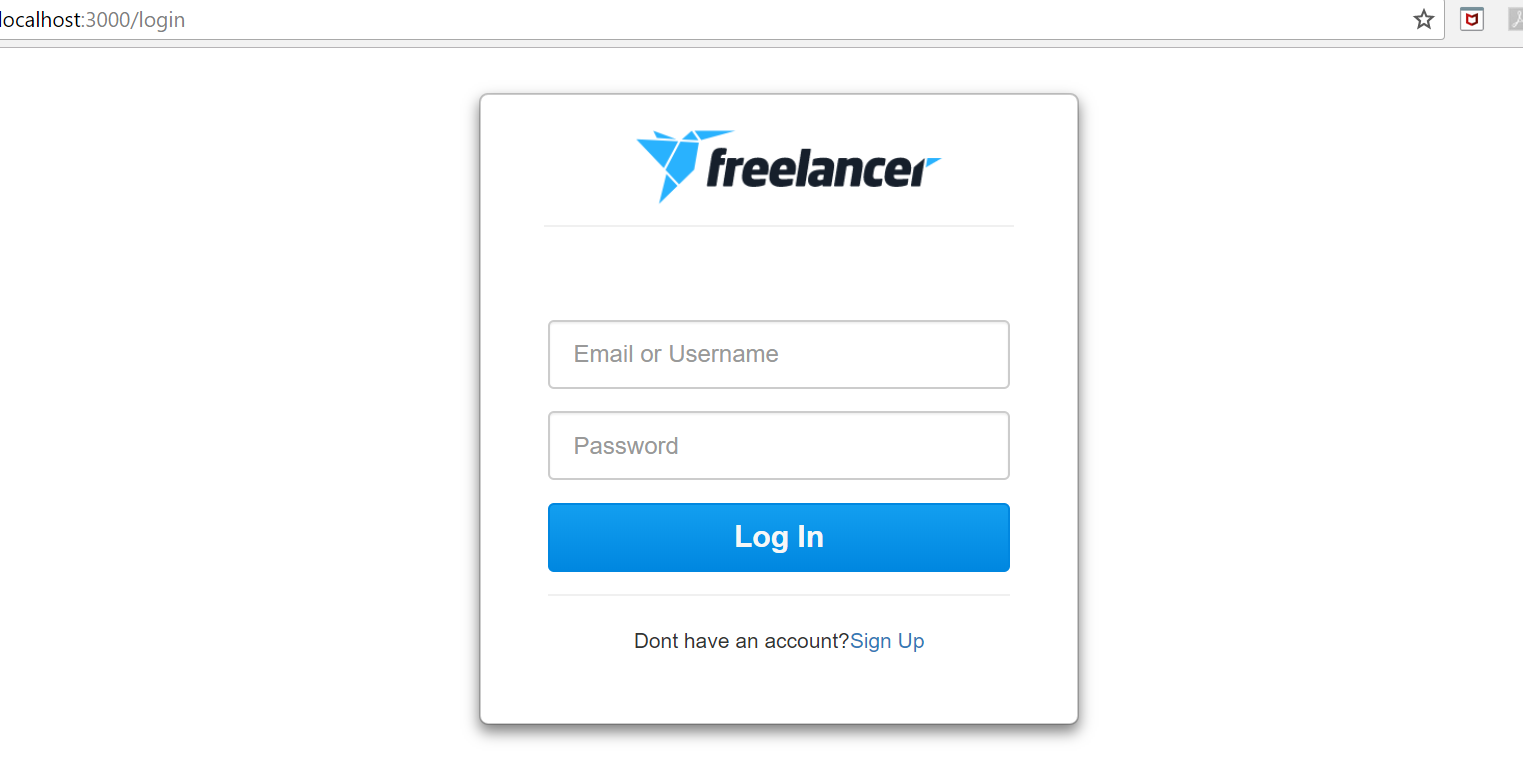
1. Signup page **(Username criteria validation)**



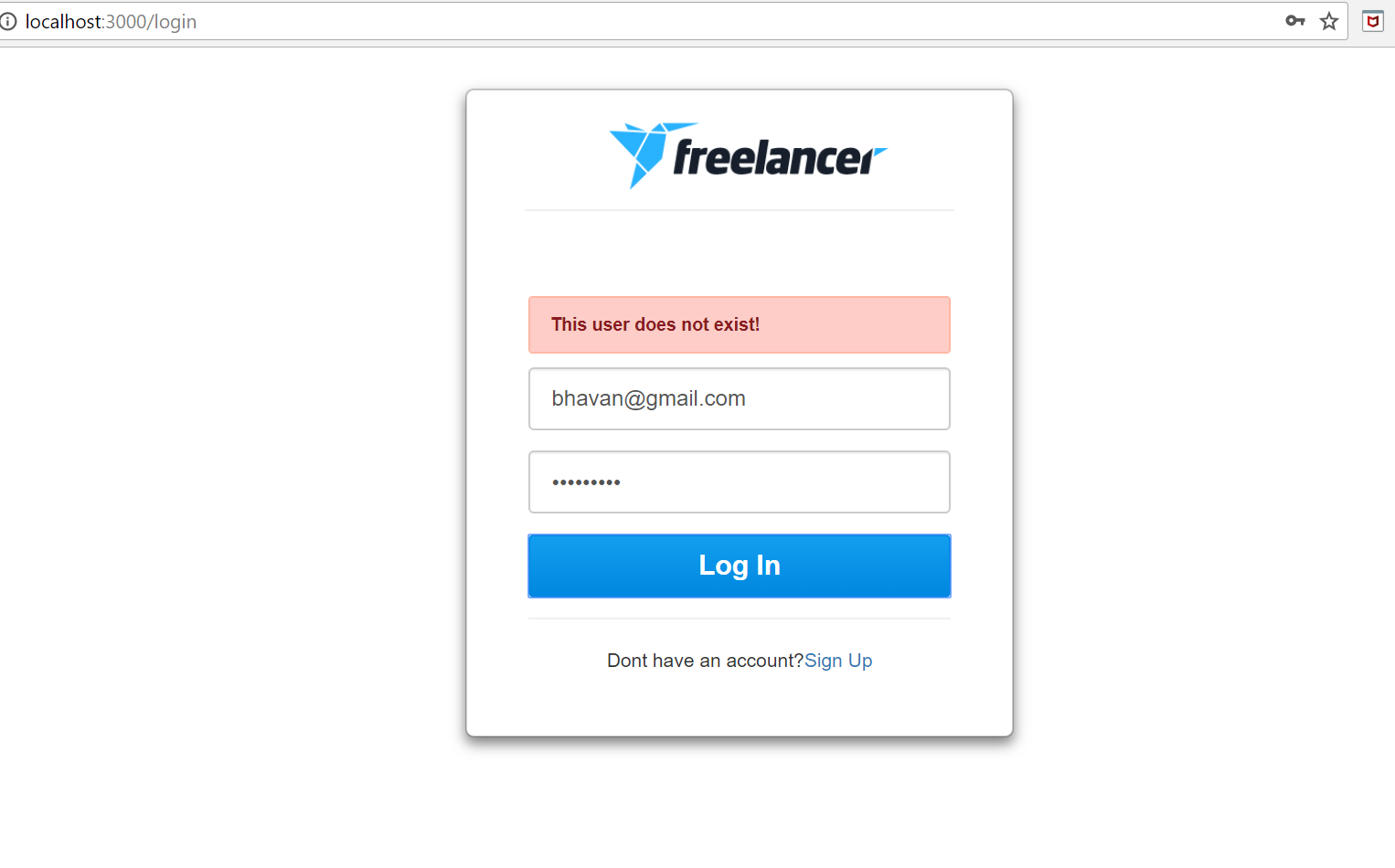
1. Signup **( strong password validation )**



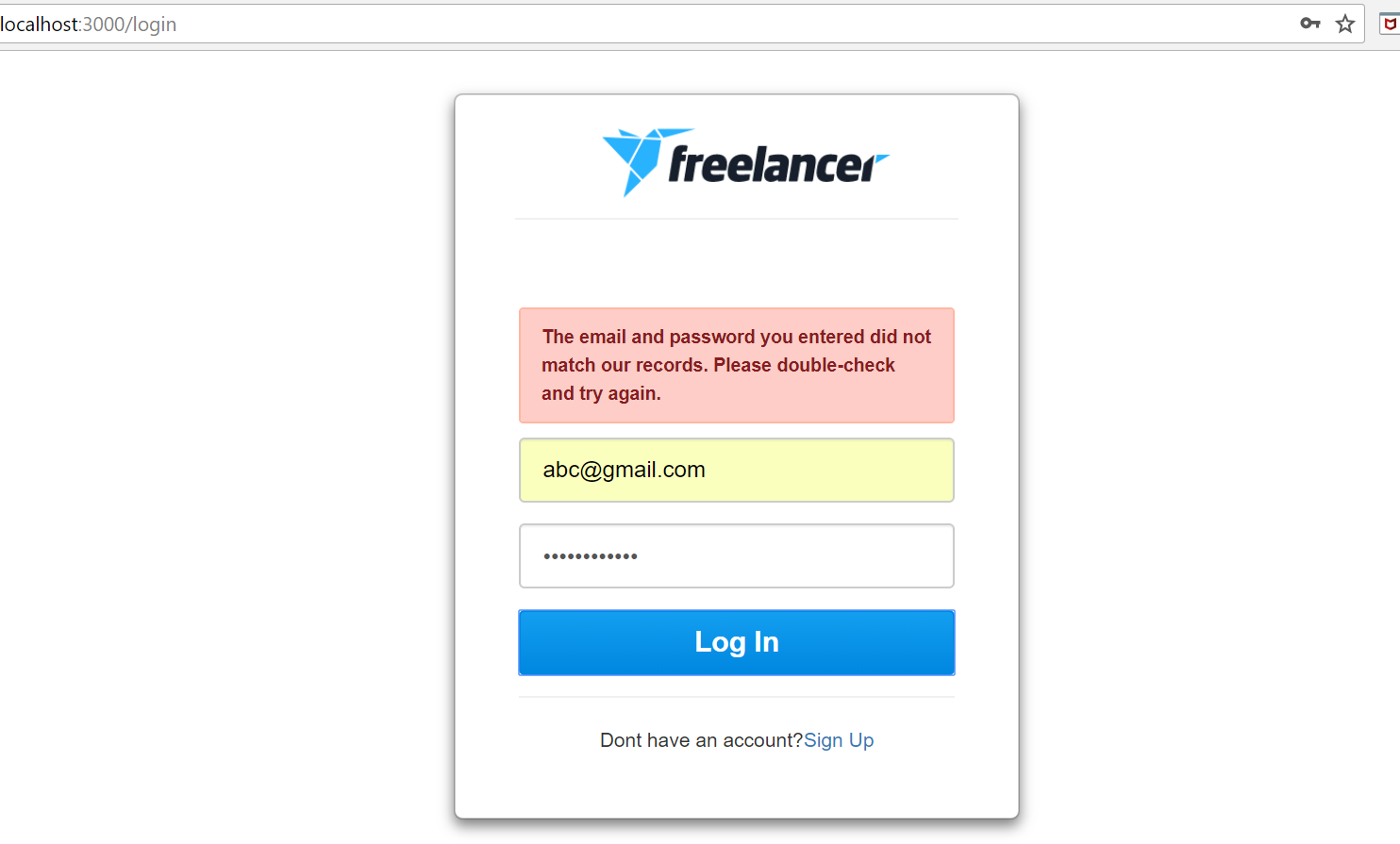
1. **Login page (login either with username or email )**



1. **Login Page validation (when user does not exist)**

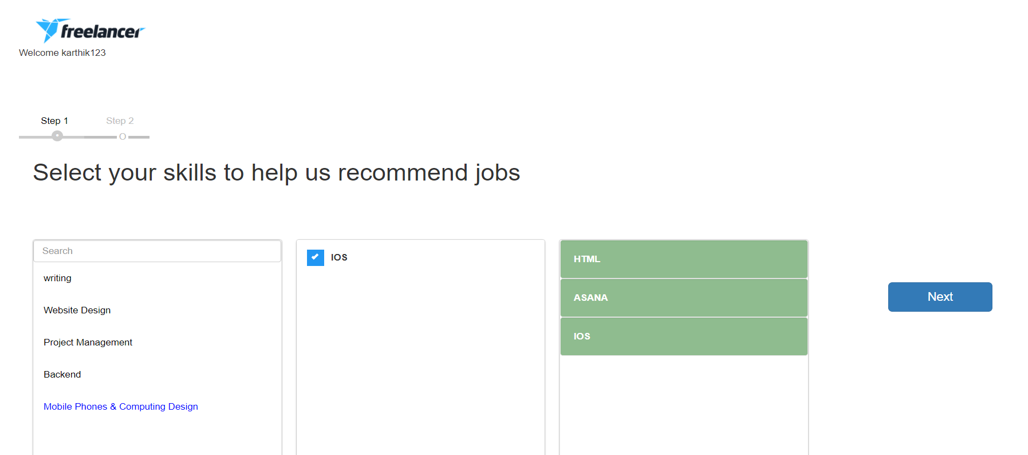


1. **Login page validation (When password entered is incorrect)**

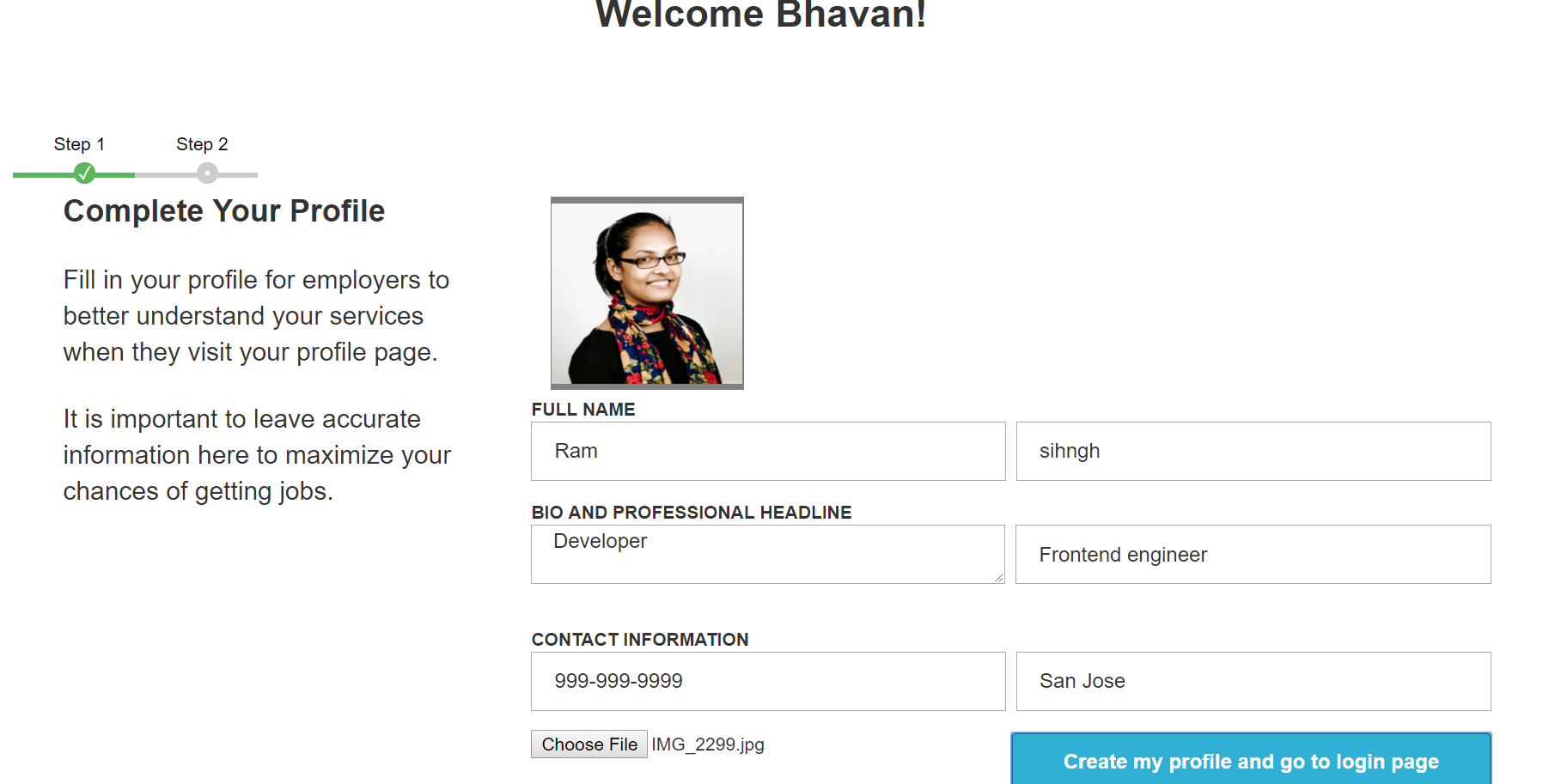


1. **After Sign up if clicked on “Work” radio button the sign up form navigates to complete form which asks to submit skills as per category:**

**There is a progress par to show the steps completed**

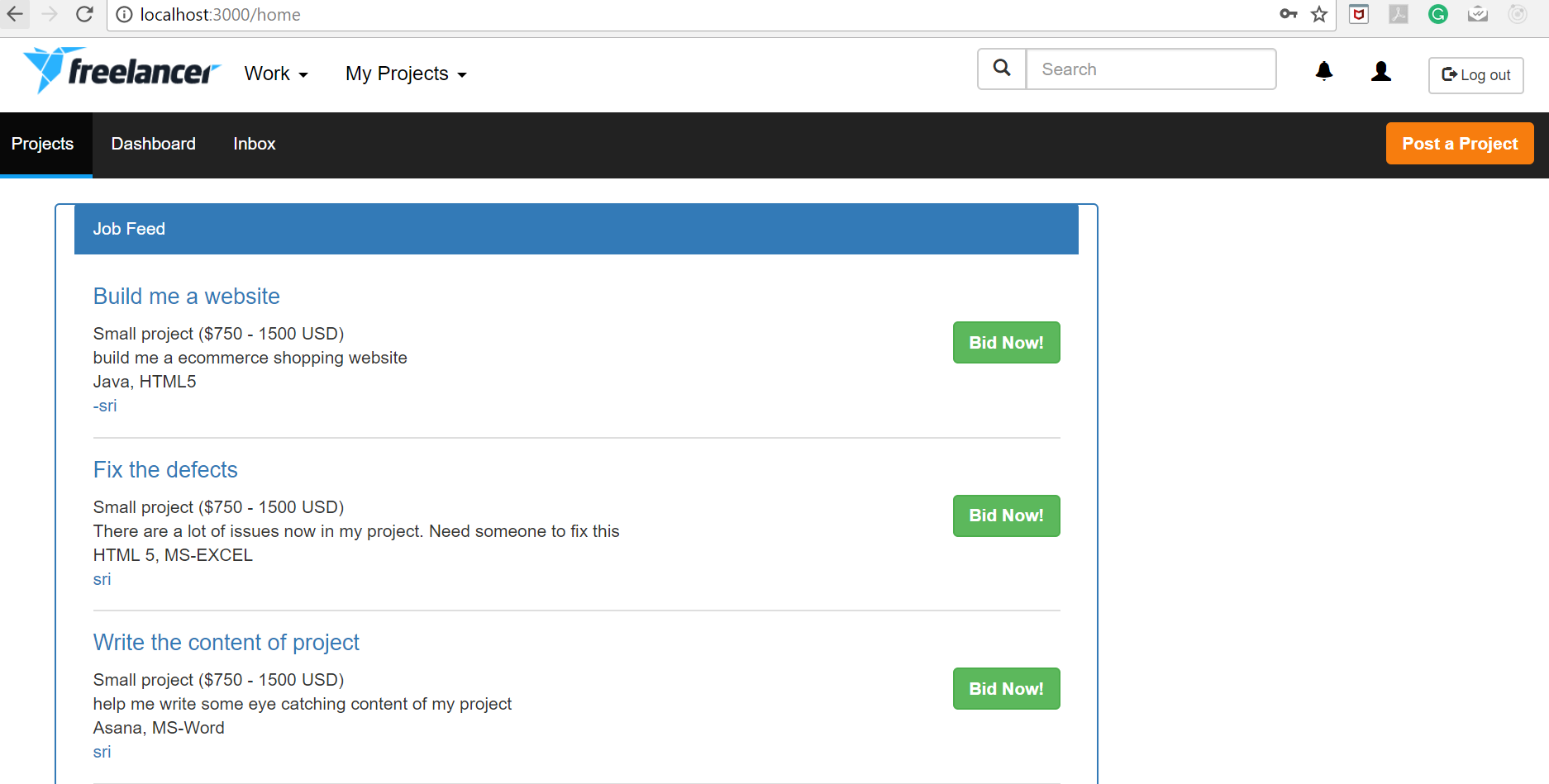


1. **Step 2 of complete profile**

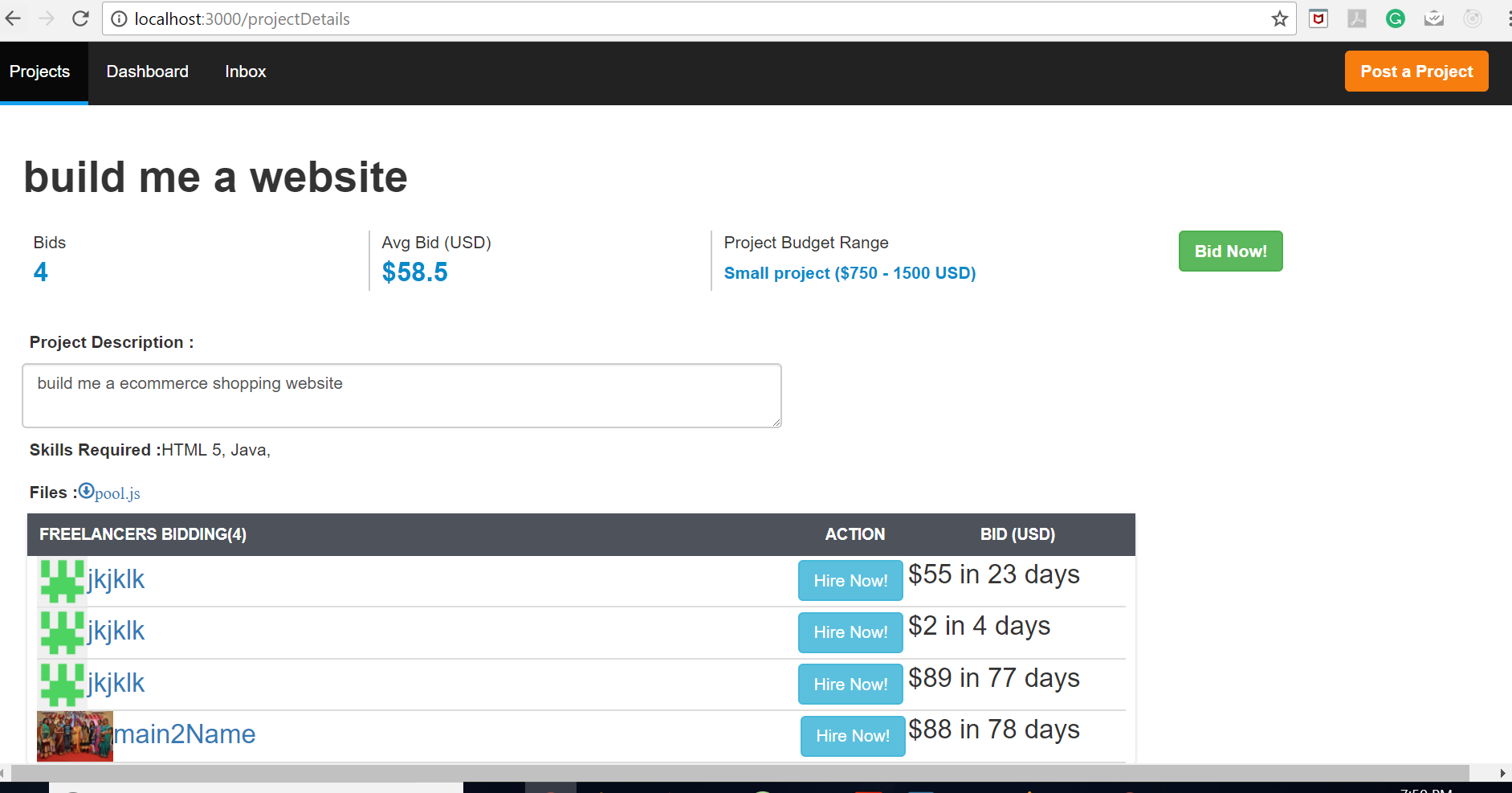


1. **After complete profile login and see the home page which has list of open projects**

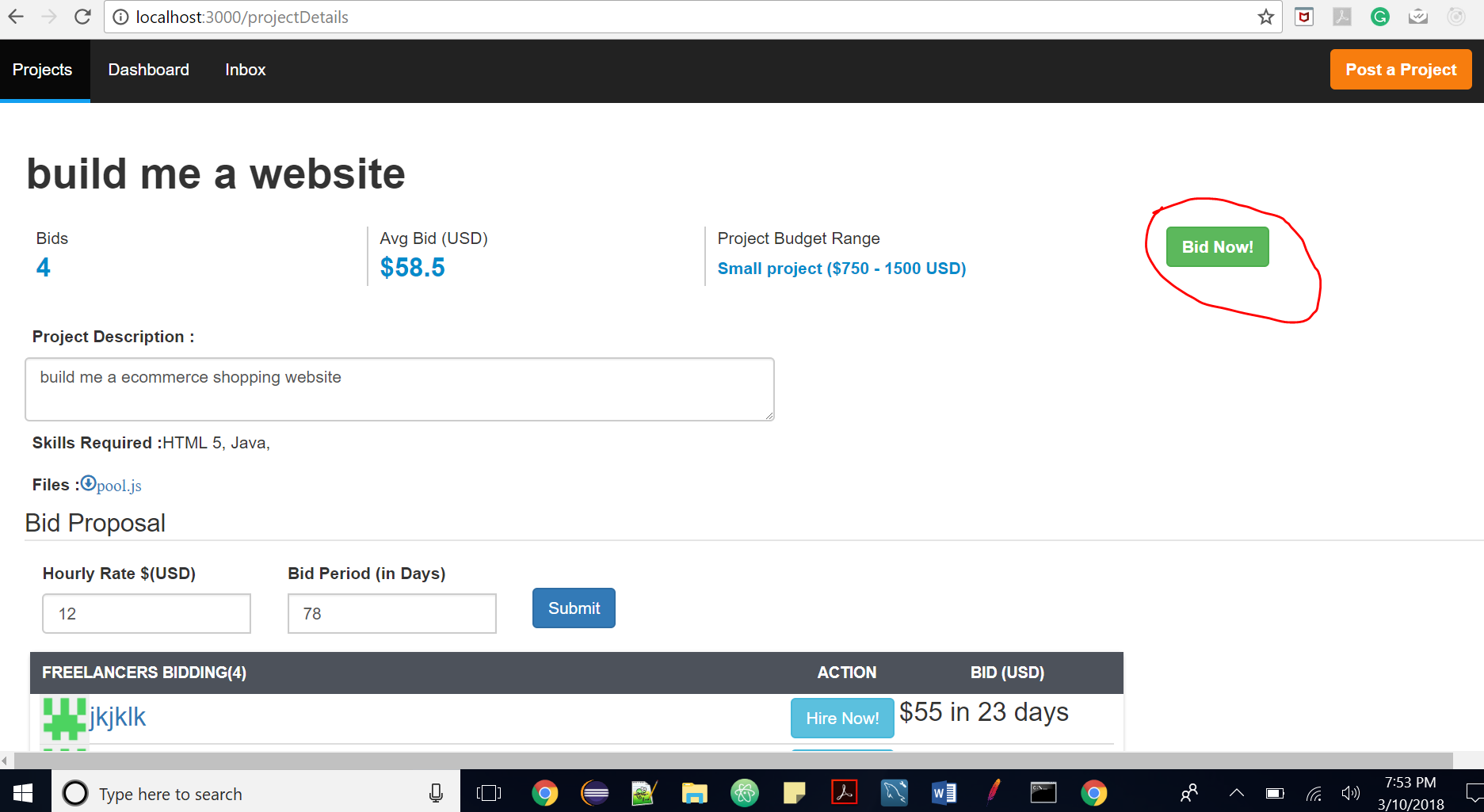
**(username**



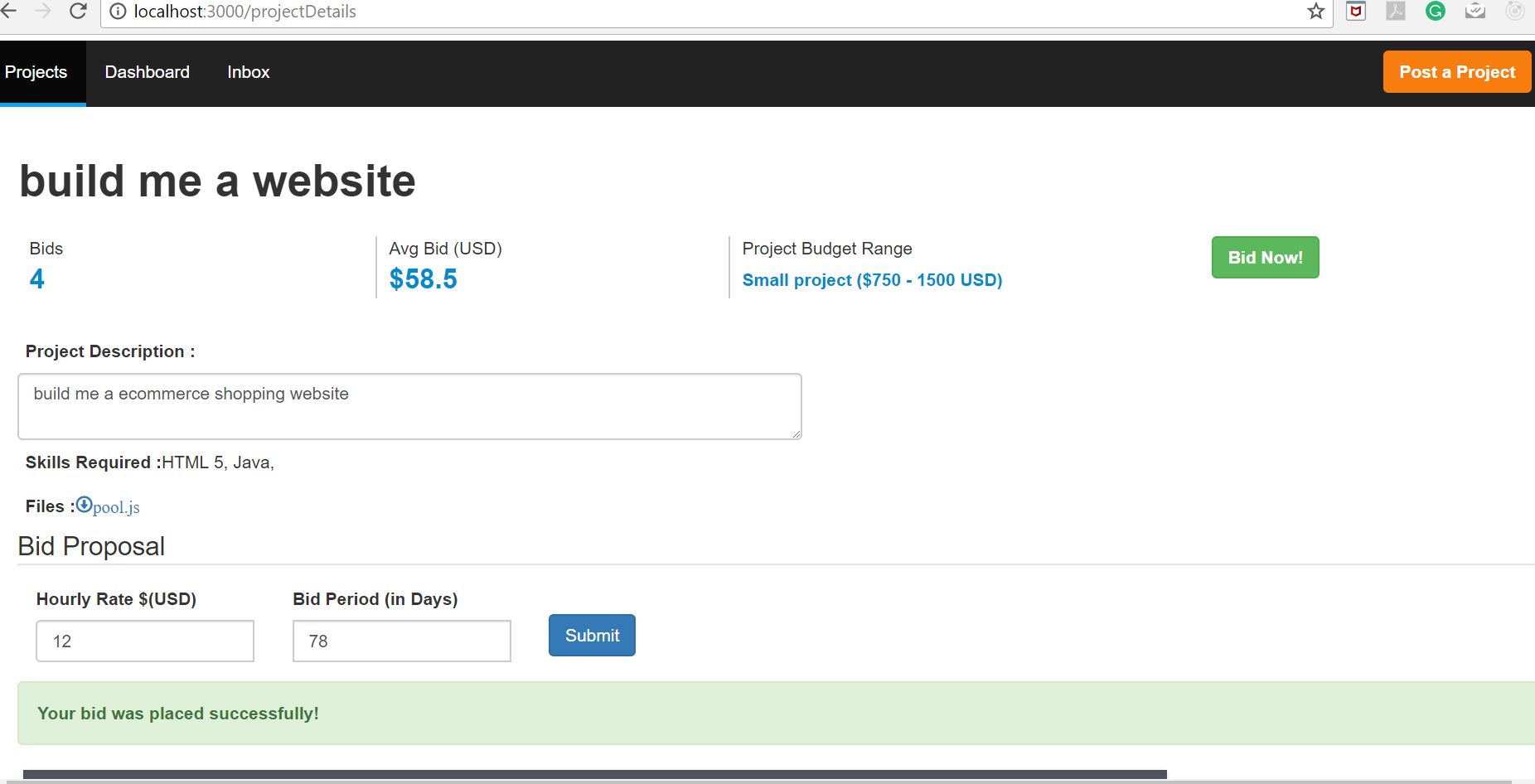
1. **Click on the project name link to view the Project details screen ( count of bids, average bids, description, file download, number of users bidding)**



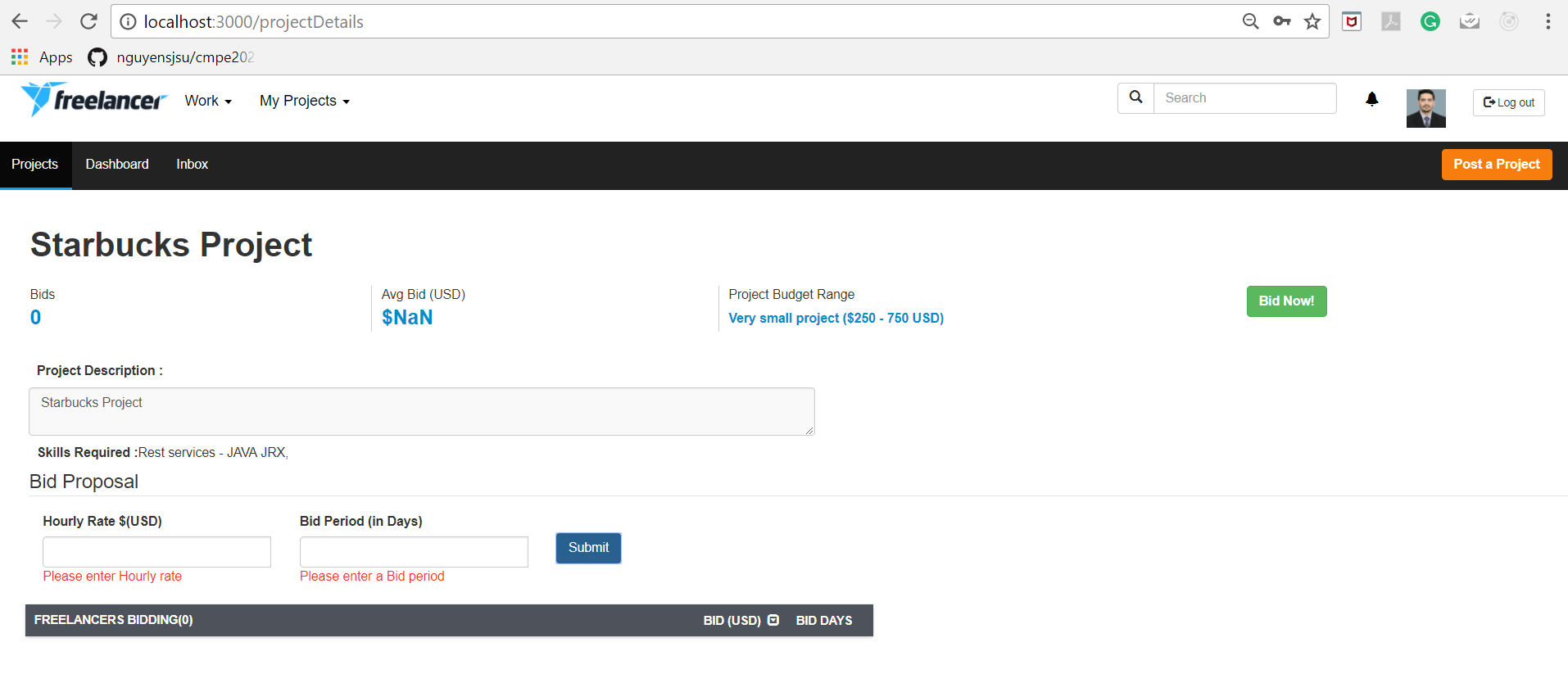
1. **Bid Proposal (toggle on Bid Now button )**



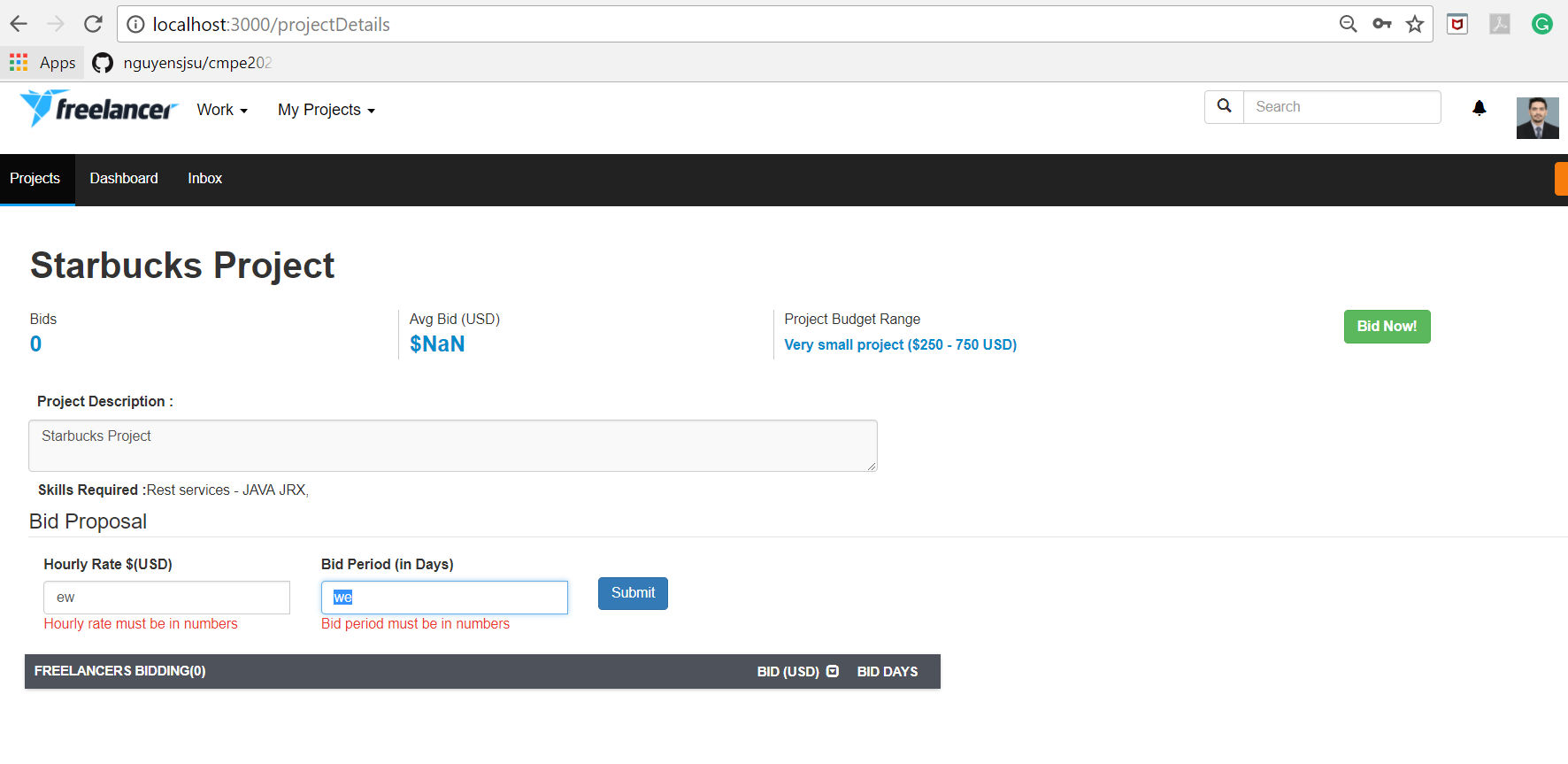
1. **Bid Proposal submitted successfully**



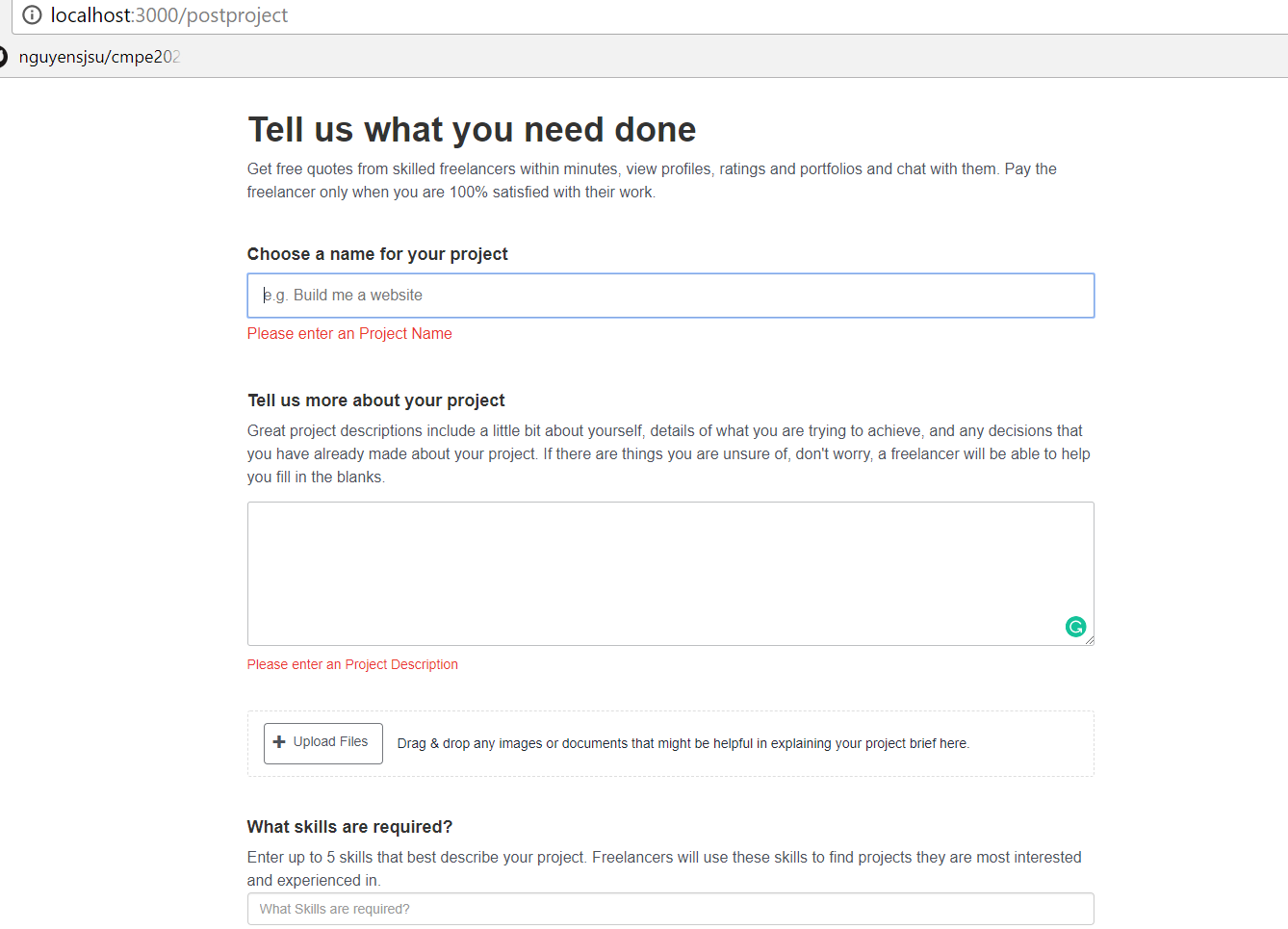
1. **Bid Form Validations ( check whether value has been entered)**

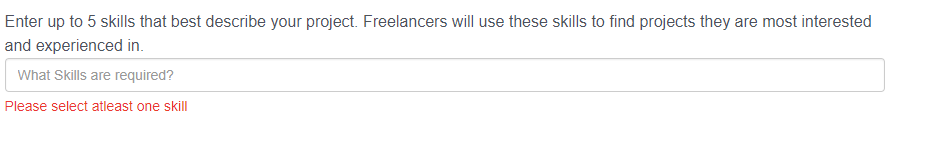


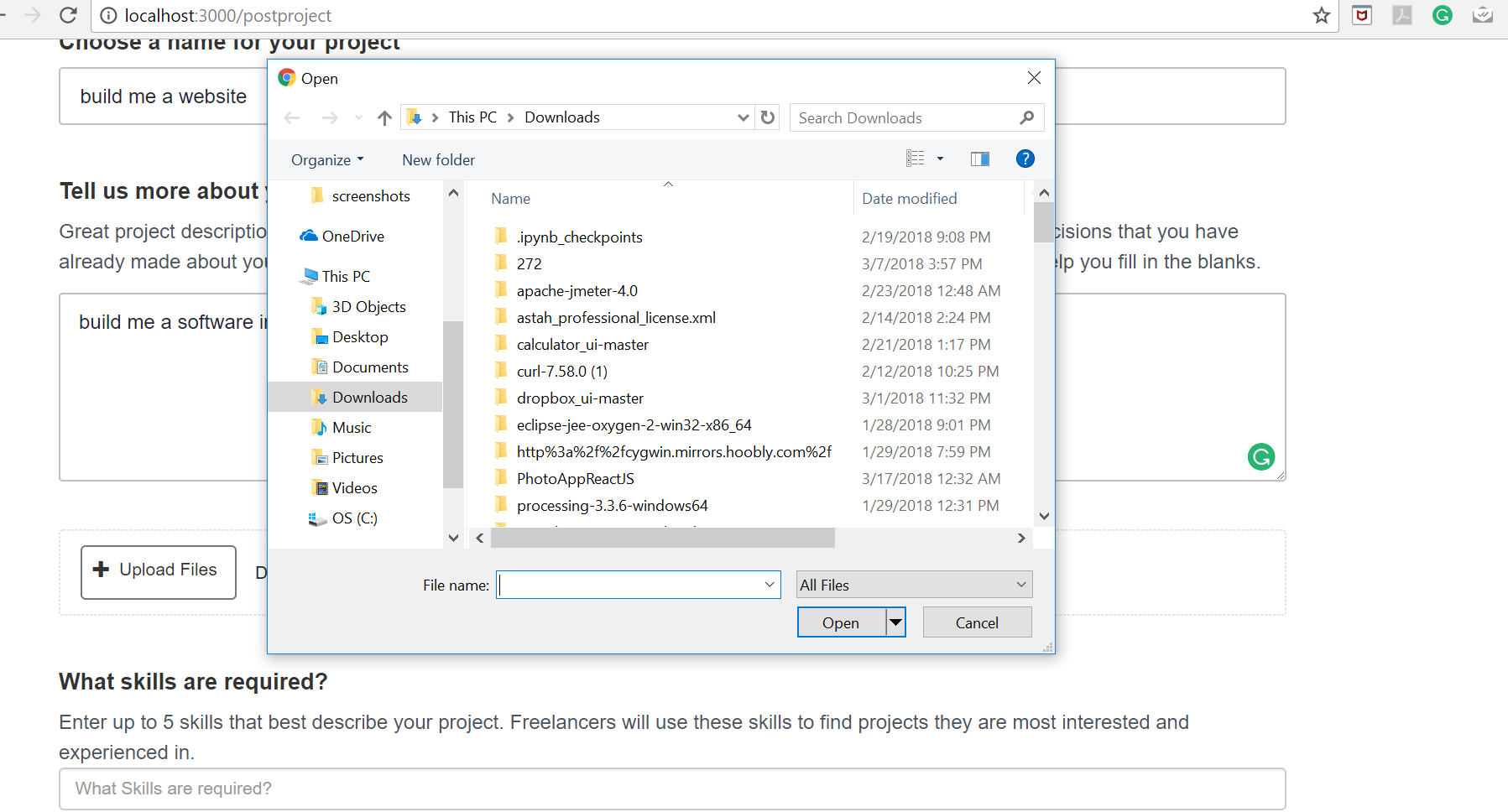
1. **Bid Form Validations ( check whether value is numeric or not)**

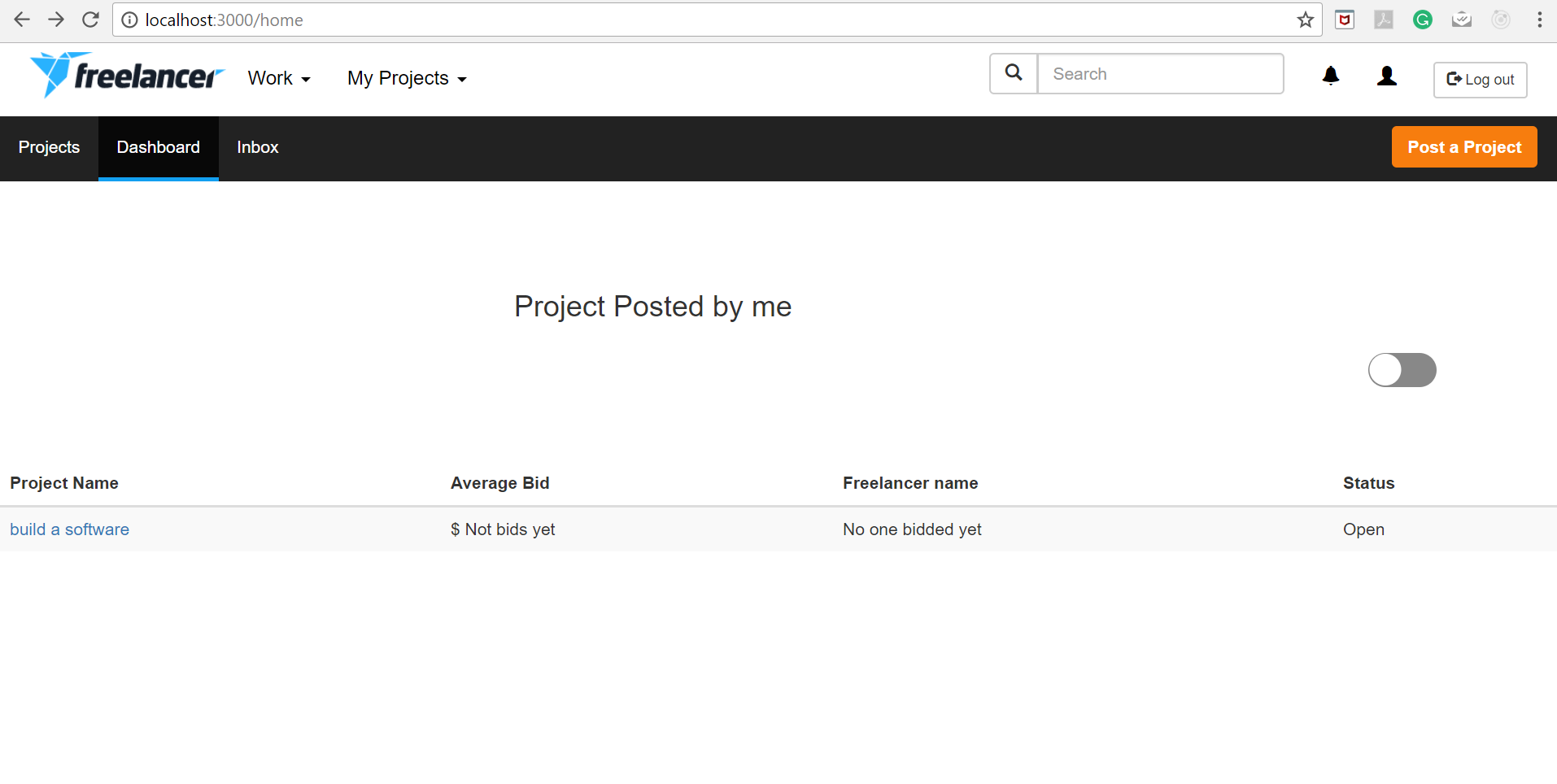


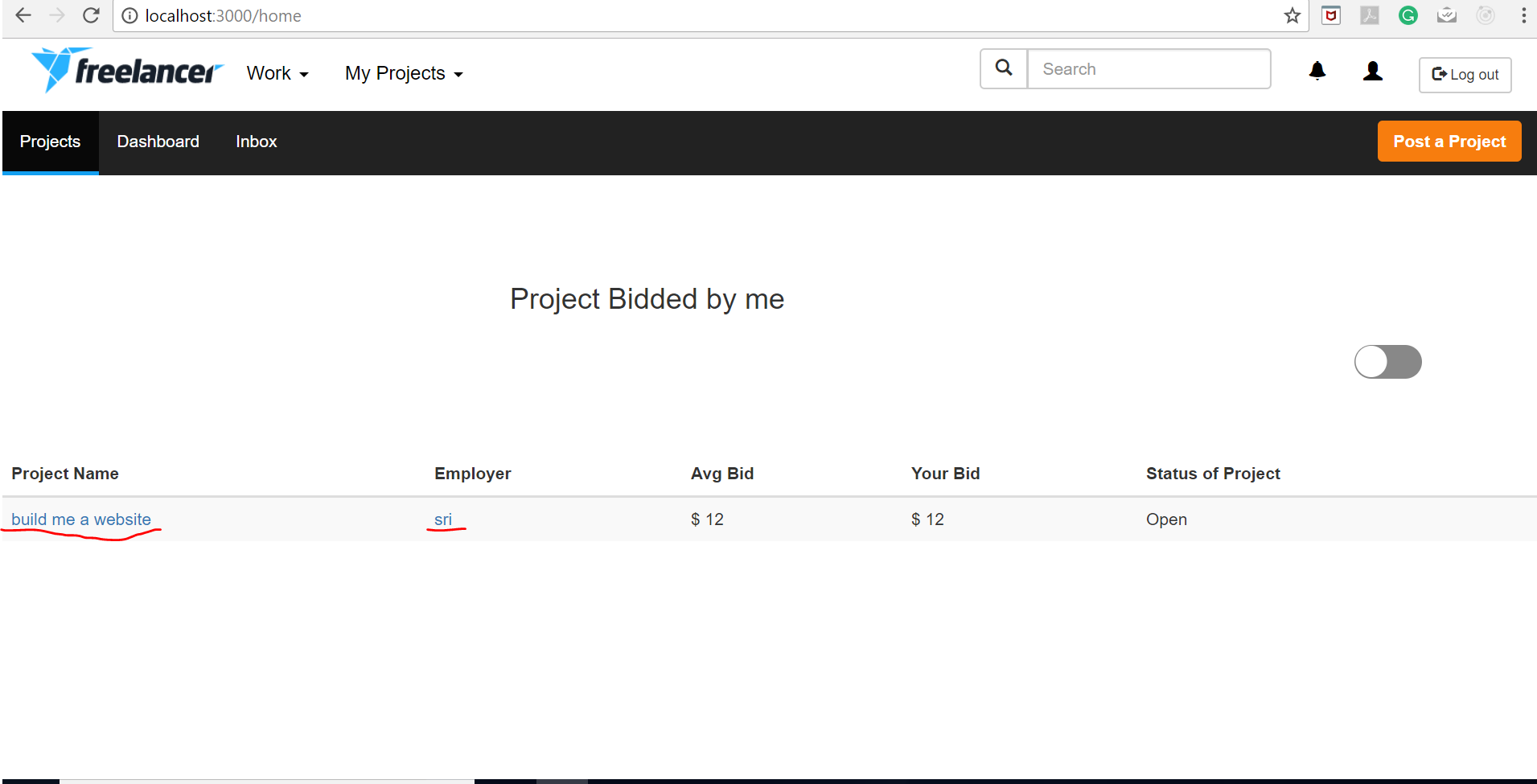
1. **Post Project Page ( including upload files )**



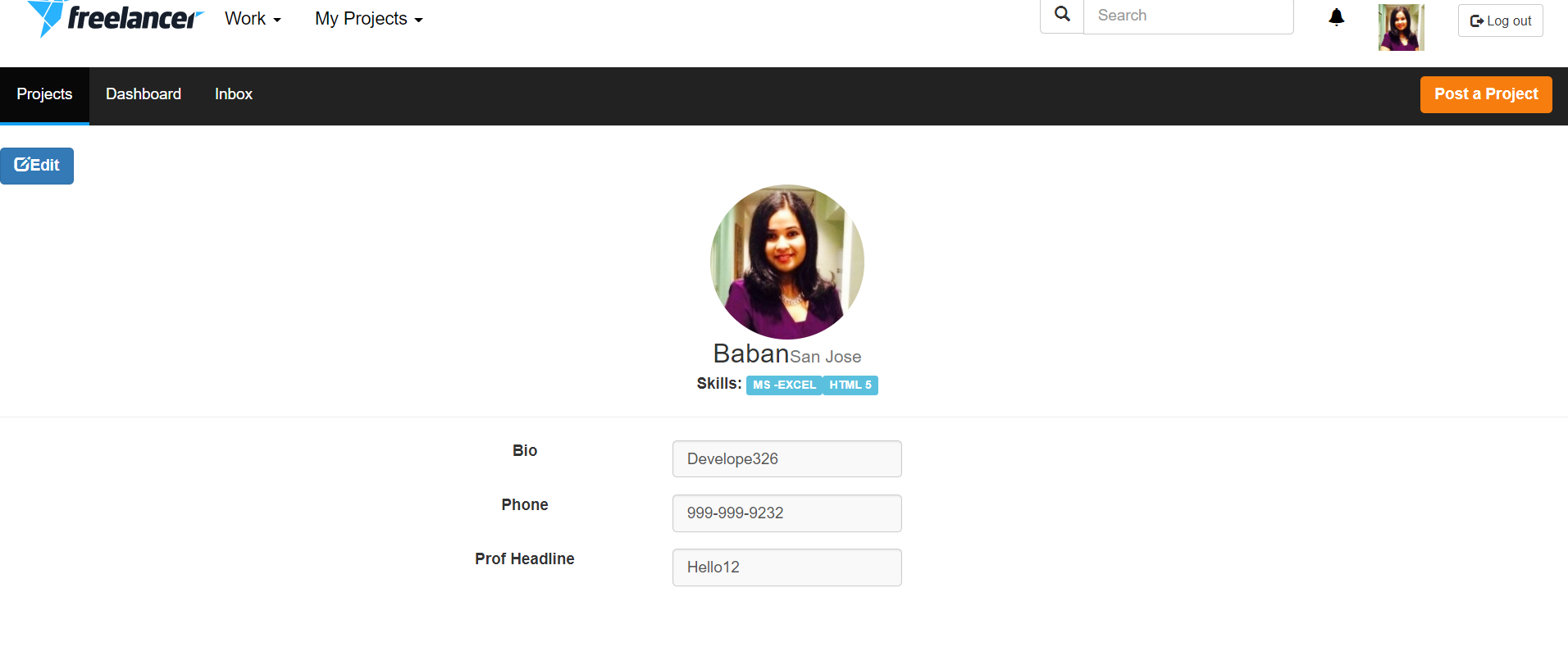


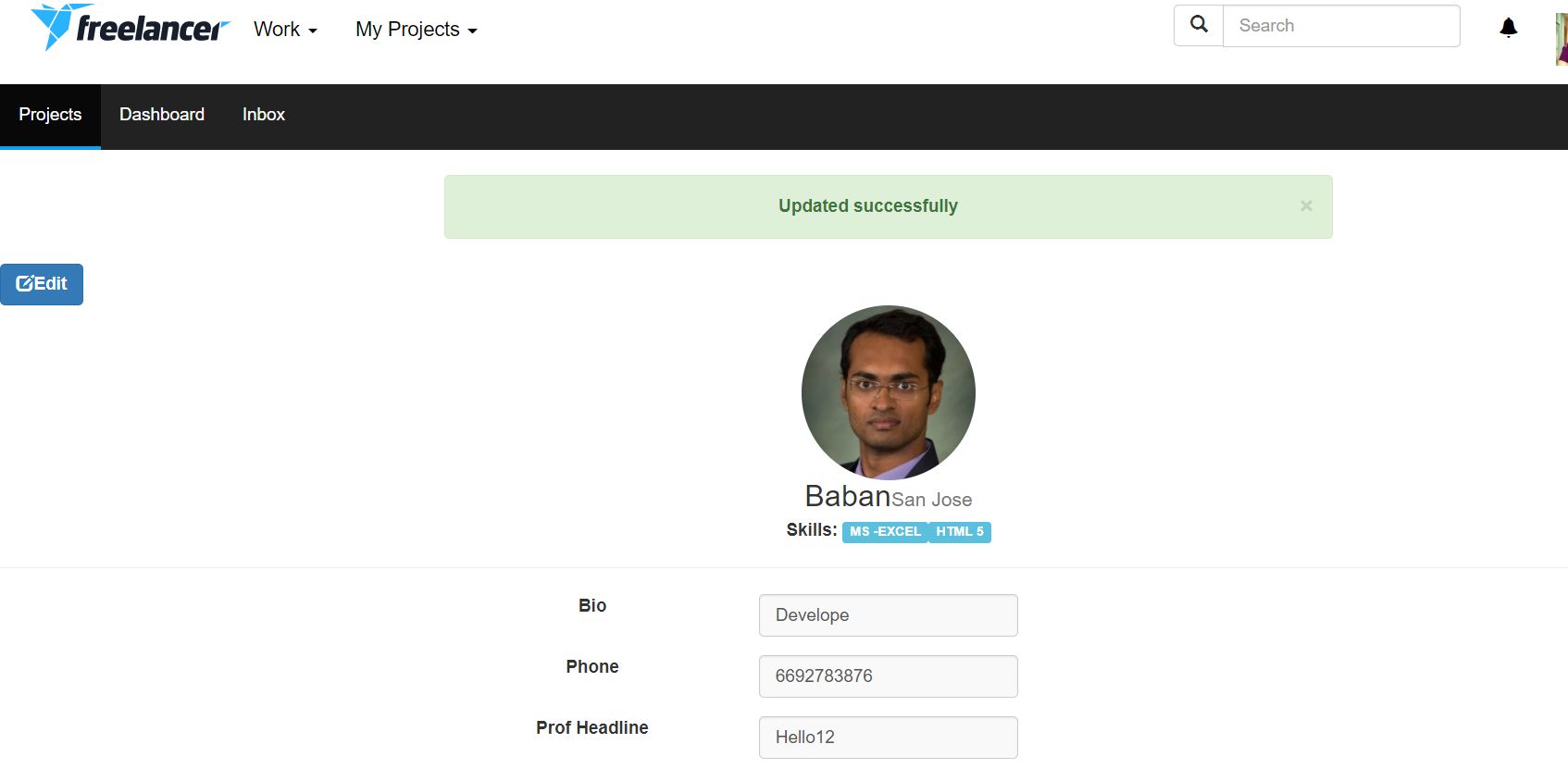


1. **Projects posted by me dashboard** **(both project name and username have hyperlink**
2. **Projects bidded by me dashboard (both project name and username have hyperlinks)**

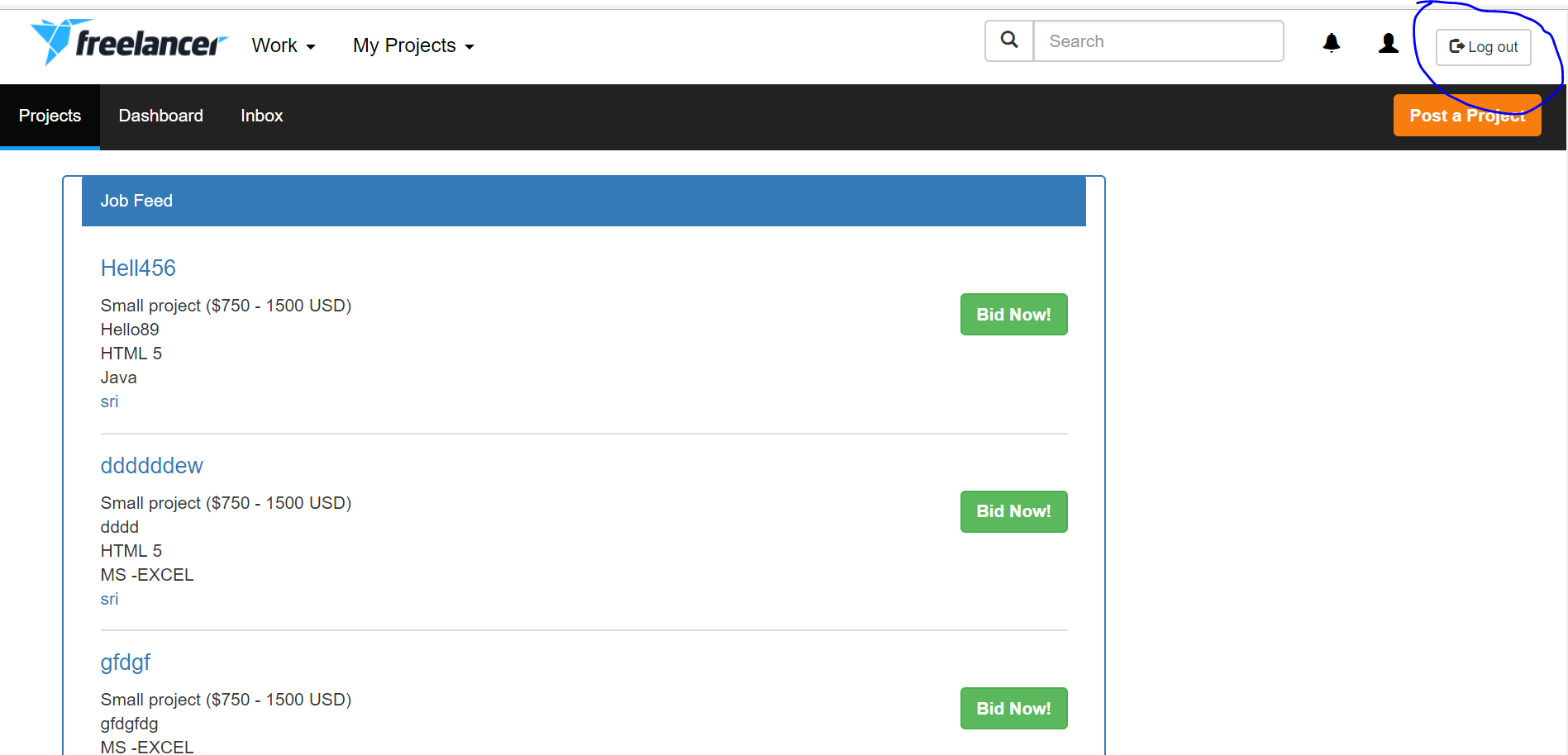


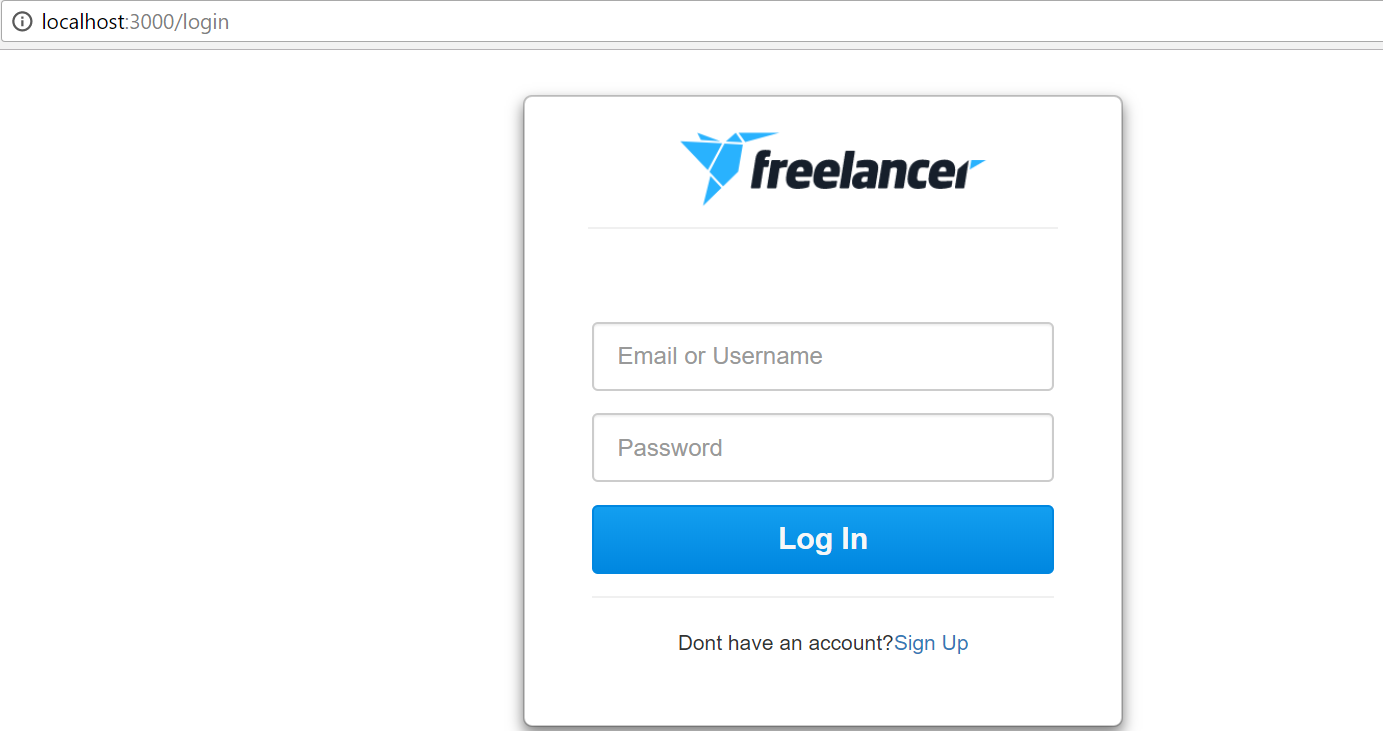
1. **View/Edit Profile**





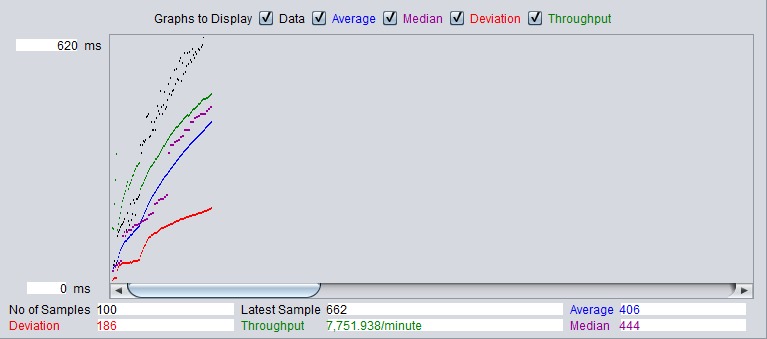
1. **Log out and return to login page**



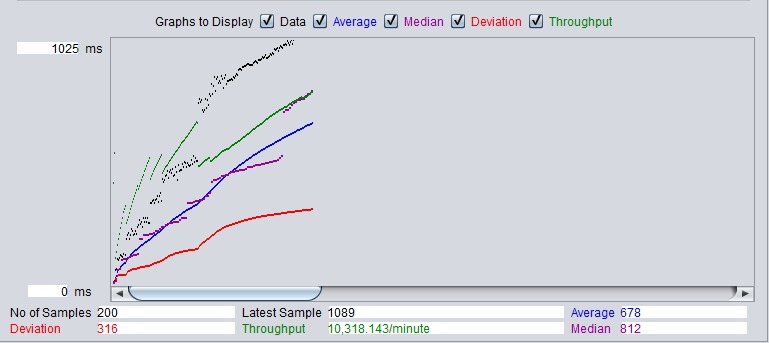


**JMeter**

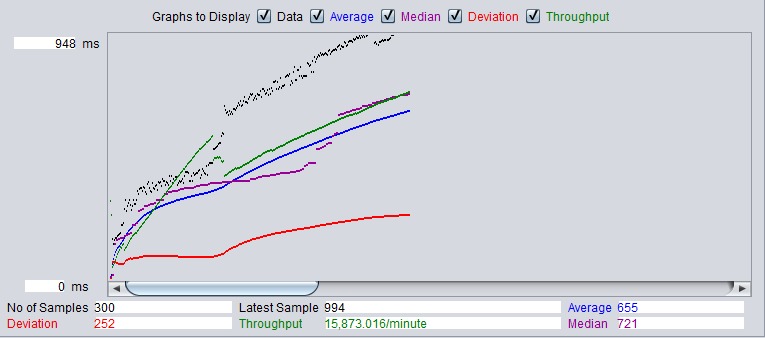
1. **100 Users**



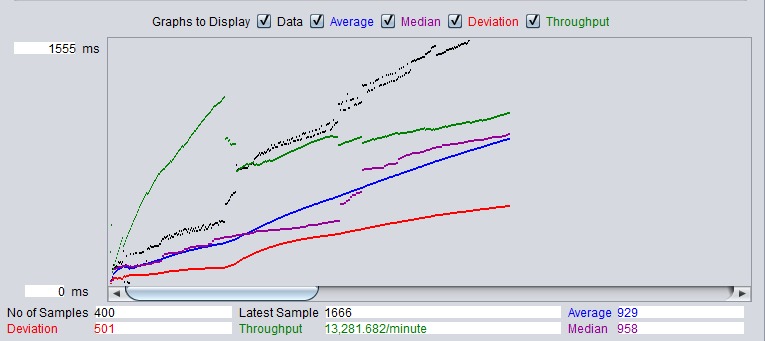
1. **200 Users**



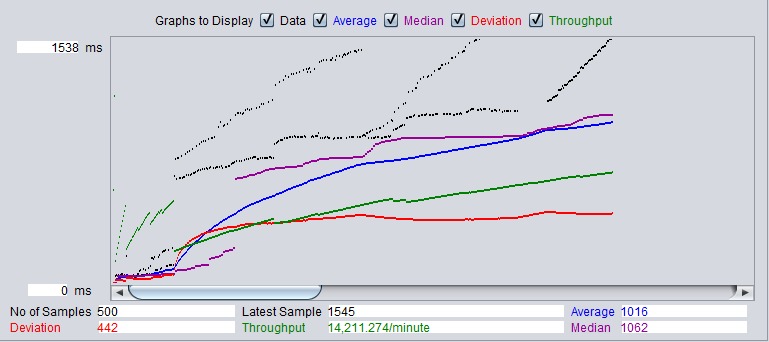
1. **300 Users**



1. **400 Users**



1. **500 Users**

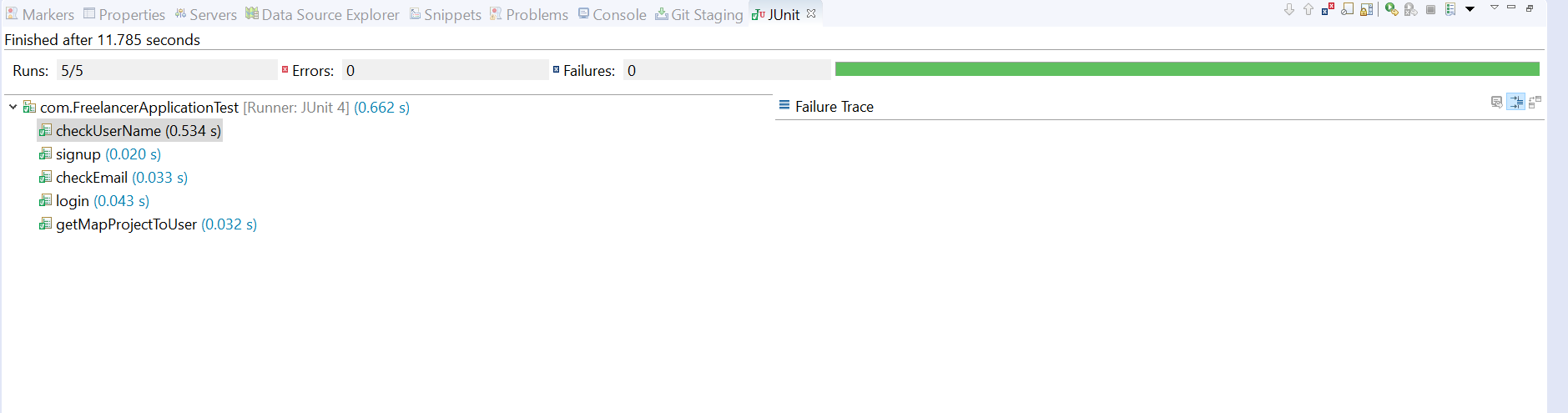


|  |  |
| --- | --- |
| Jmeter Load Testing | |
| # users | average time in (ms) |
| 100 | 620 |
| 200 | 1025 |
| 300 | 948 |
| 400 | 1555 |
| 500 | 1538 |

**Junit**

I have tested 5 API’s

**Result**



**Code**

