CMPE 273 Lab 1

Hongyuan Li (011838151)

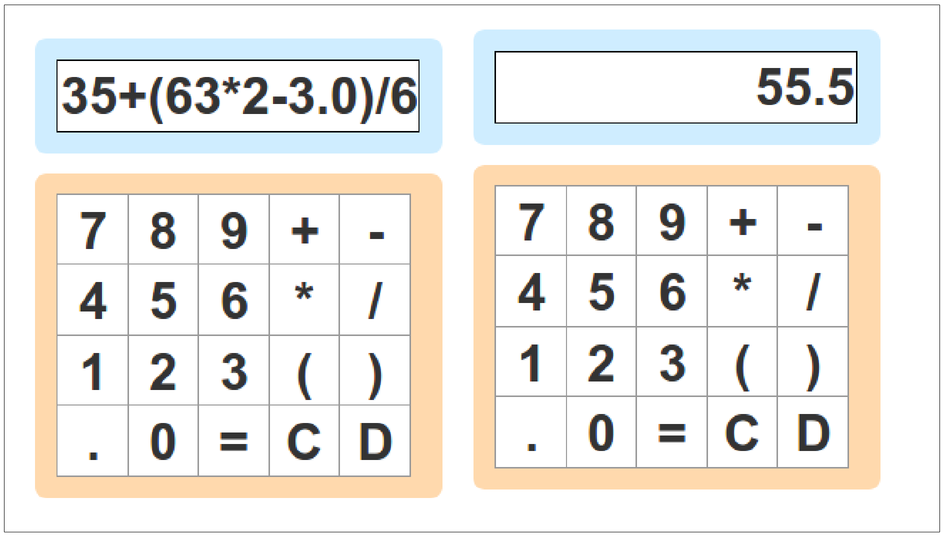
App1: Calculator

Introduction:

This is simple calculator that can perform: addition, subtraction, multiplication and division. The calculator server is a RESTful server that accepts input strings to parse and calculate the result.

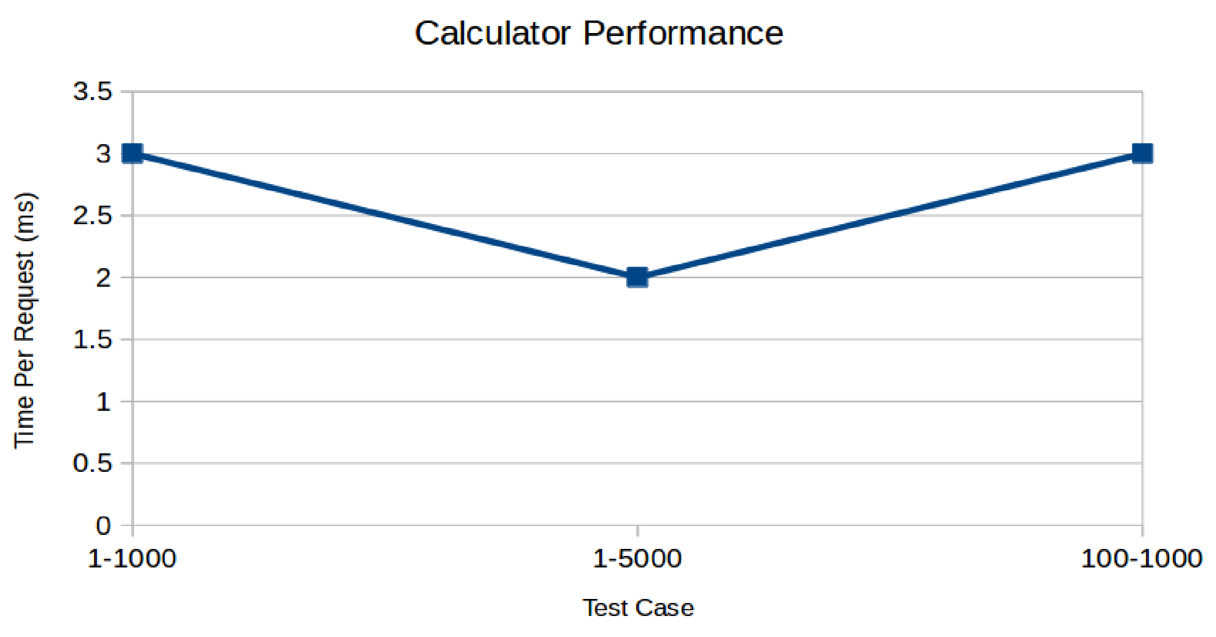
Results:

Here's the calculation example. After click the key pad with operations and number, we can click the equal sign "=". The calculator generates the results.



Performance:

As we can see the performance chart, the response time for this calculator is steady. There's no major different between 100 concurrent users and 1 user. Since there's no database component in this application and the server computation is very simple, the performance maintains at a good level when there are more concurrent users.



App 2: Freelancer

Introduction

The freelancer application is built to allow users post and bid for projects. Users can sign up, login and logout the website. Users can update their profiles with skills and images. Users can post projects with budget, skill set information. Users can bid on projects with price and number of days. Project owners can hire bidders.

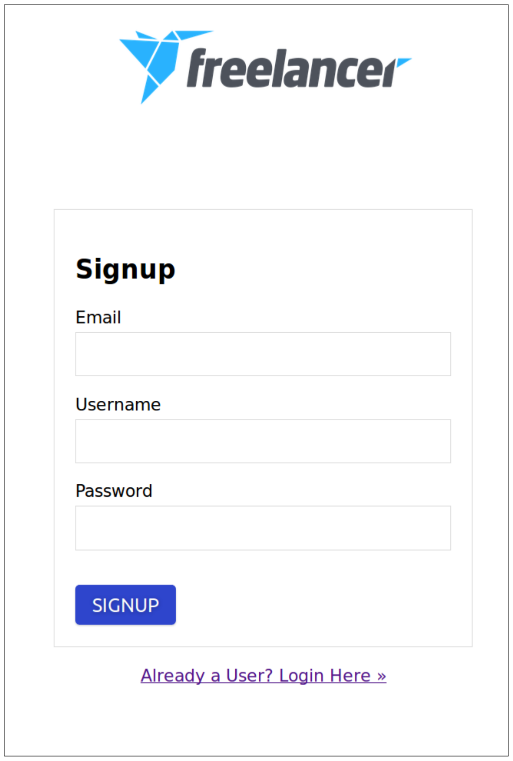
Backend server is built with Node.js and Express.js. In addition:
'cors' is used to hand cross origin requests.
'express-fileupload' is used to handle file uploads.

Frontent client is built with React, redux and Saga. In addition:
'react-router-dom' is used for routing
'admin-on-rest' and 'material-ui' are used to build the layout and widgets.

Results

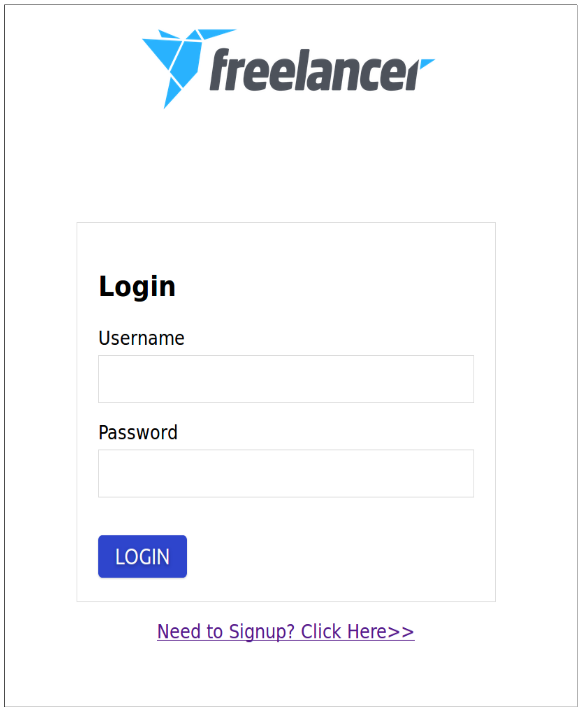
User functionalities:

1. Sign up
Users need to provide, email, username, and password to signup. Once signup is successful, page will be redirected to login. Bcrypt is used to encrypt passwords



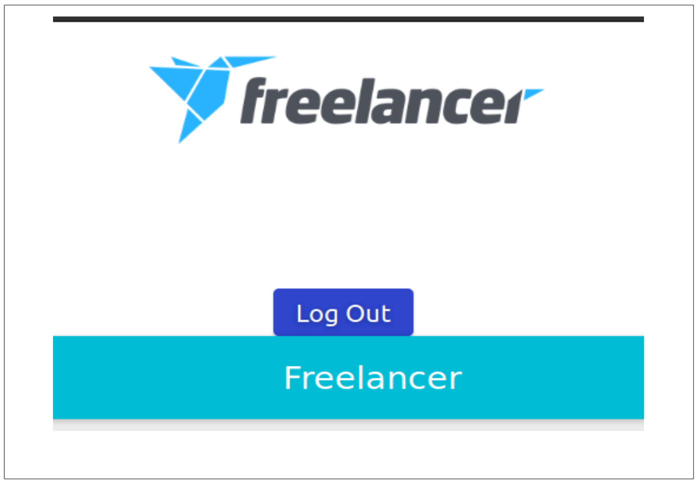
2. Sign in

Users need to use username and password to sign in. When login is successful, page will be redirected to home



3. Sign out

On top of the home page is the sign out button. When click, user is signed out and the page is redirected to login.



4. Menu:
On the left side of the home page, there's a responsive menu. Here's the direction.

Projects: list all the projects. Users can edit, create, and bid projects on this page, depends on ownership and availability.

Bids: list all the bids. Users can delete their own bids, if it's not chosen. Employers can click on "hire" to hire bidders.

All skills: list all skills available to be added to users and projects

Users: list all users.

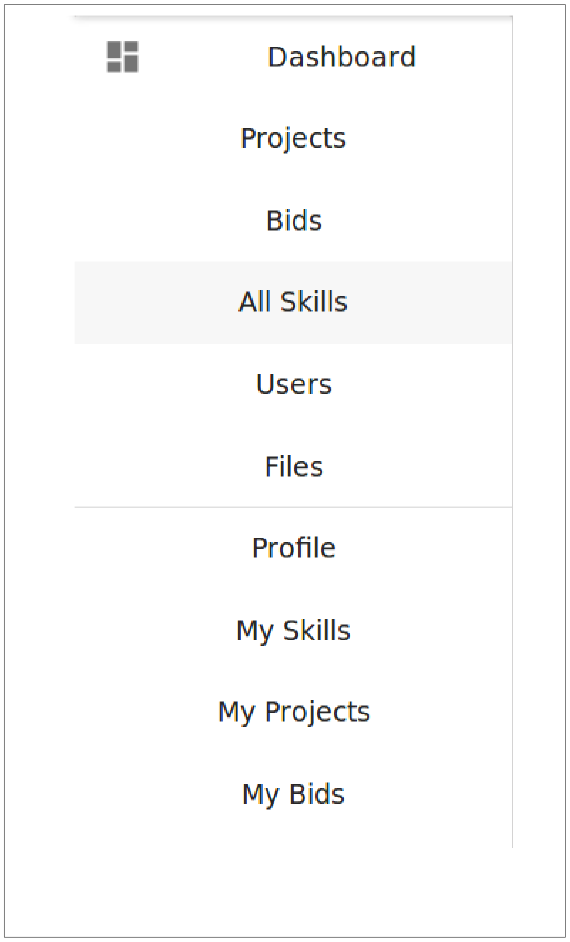
Files: list all files available for download.

Profile: the profile of the login user

My skill: list of my skills.

My Projects: list of all projects, current user posted.

My bids: list of all bids, the current user placed.

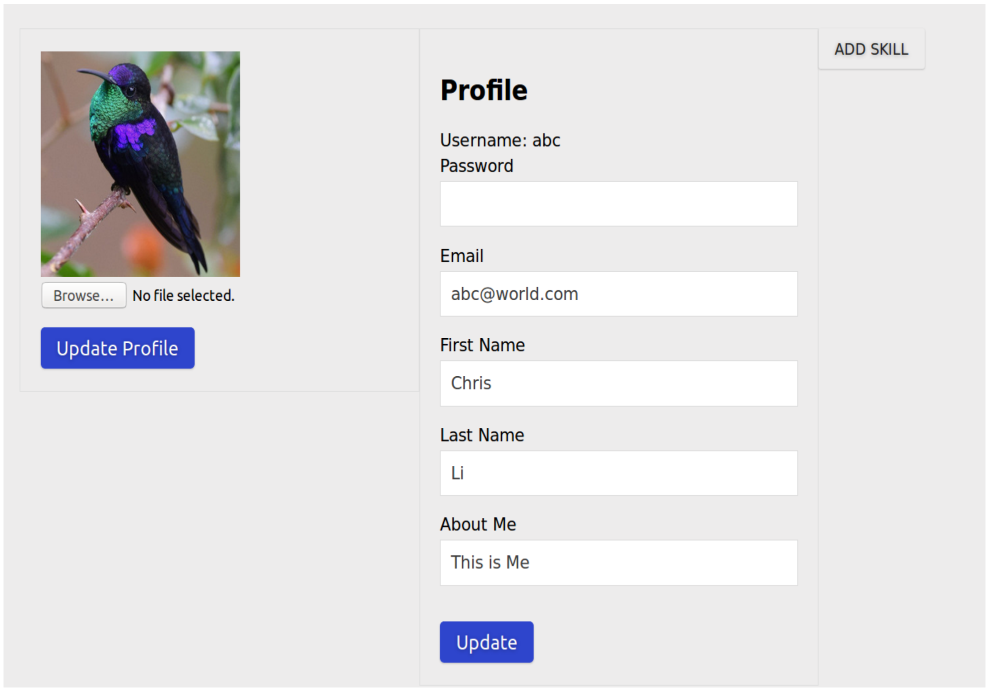


4. profile:

On the profile page, the user can first change his/her profile photo, on the left. Simply click at "browse", select a photo, and click at "Update Profile".

Users can also update other information on this page. Simply fill out the form and click at "Update".

Users can add skills, by clicking at the "Add Skills" on the right.



ADD SKILL

Profile

Username: abc

Password

Email: abc@world.com

First Name: Chris

Last Name: Li

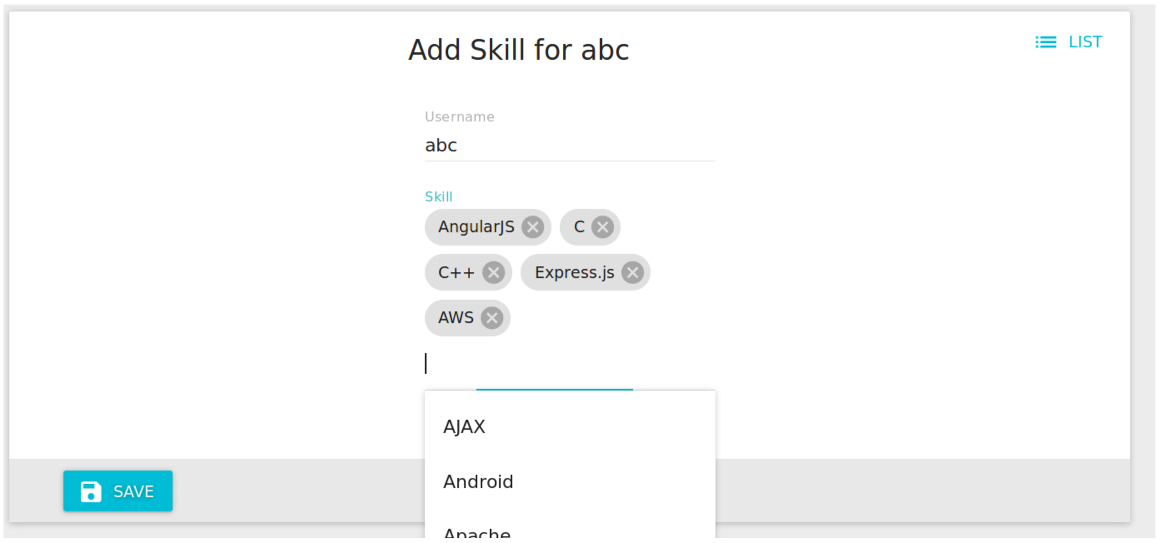
About Me: This is Me

Browse... No file selected.

Update Profile

Update

When clicked at the "Add Skill" button, users are redirected to add skill page. Username is pre-filled. Users can choose skills from a drop down list. Once click "save". The skills are saved and page is redirected to user skill list (or my skills):



Add Skill for abc

LIST

Username: abc

Skill: AngularJS, C, C++, Express.js, AWS

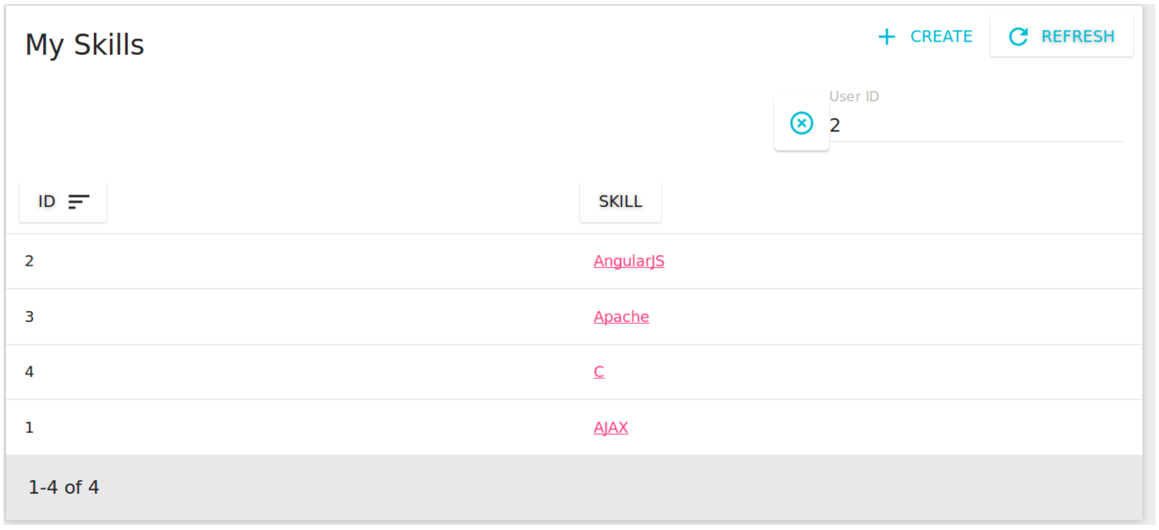
AJAX

Android

Apache

SAVE

This is a my skills page, that listed the skills a user has.



My Skills

User ID: 2

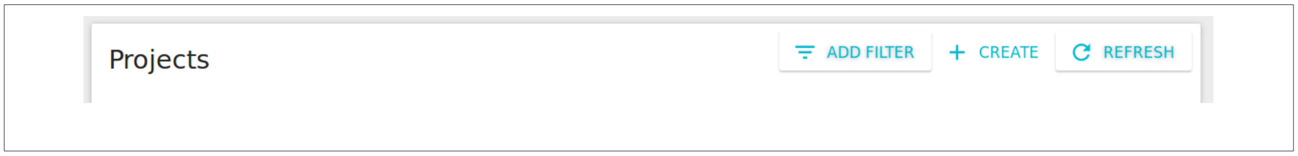
ID	SKILL
2	AngularJS
3	Apache
4	C
1	AJAX

1-4 of 4

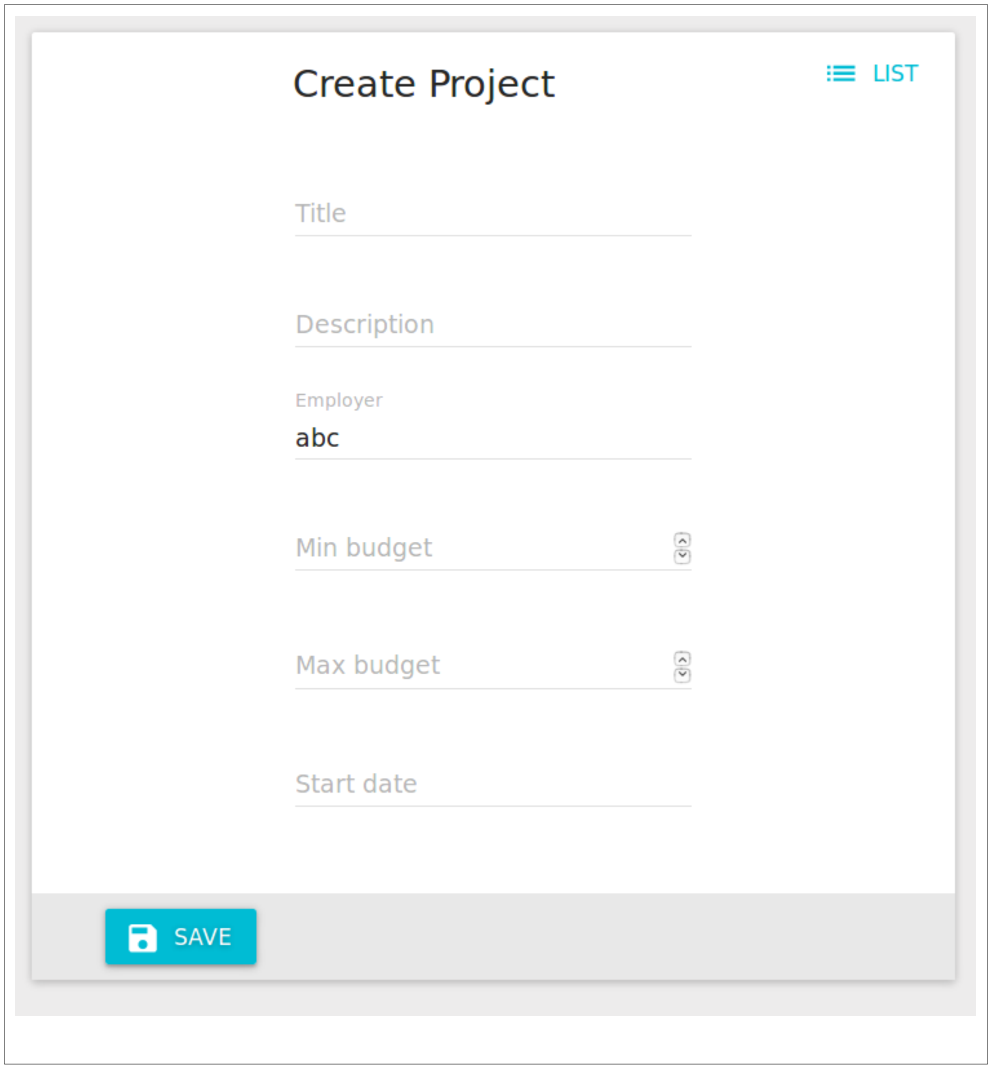
Project:

1. create a project

On the project page, users can click at the "create" button to create a project.

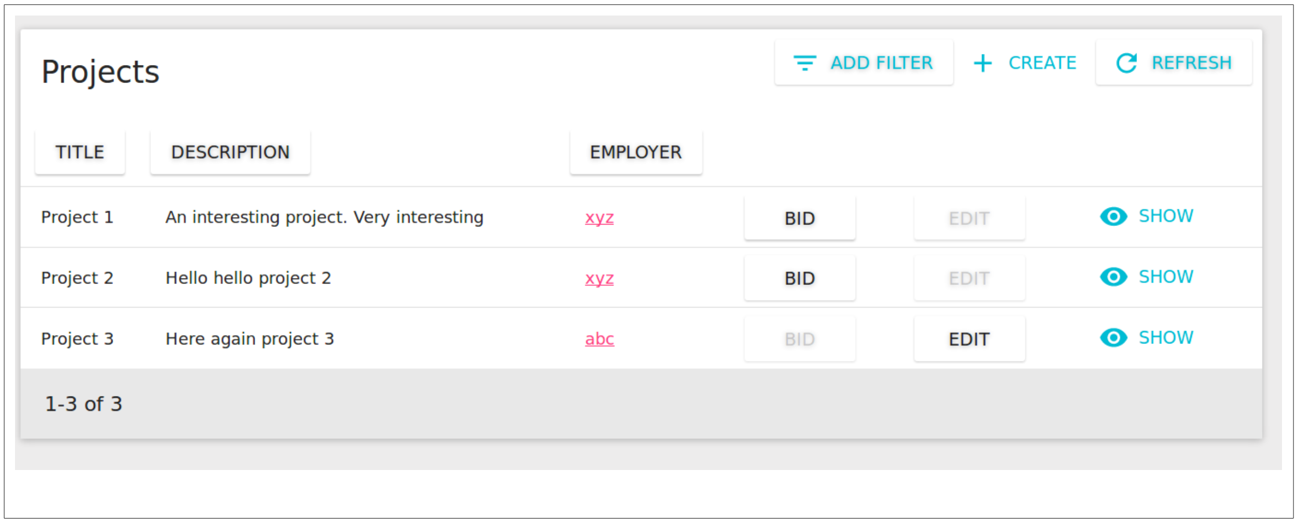


Users can fill out the create project form, to create a new project, with basic information. Once click "save", users are redirected to the projects page again.



2. Edit Project

Once a project is created, we can click at the "Edit" button to add skills and upload files. Only project owners can edit their own projects.

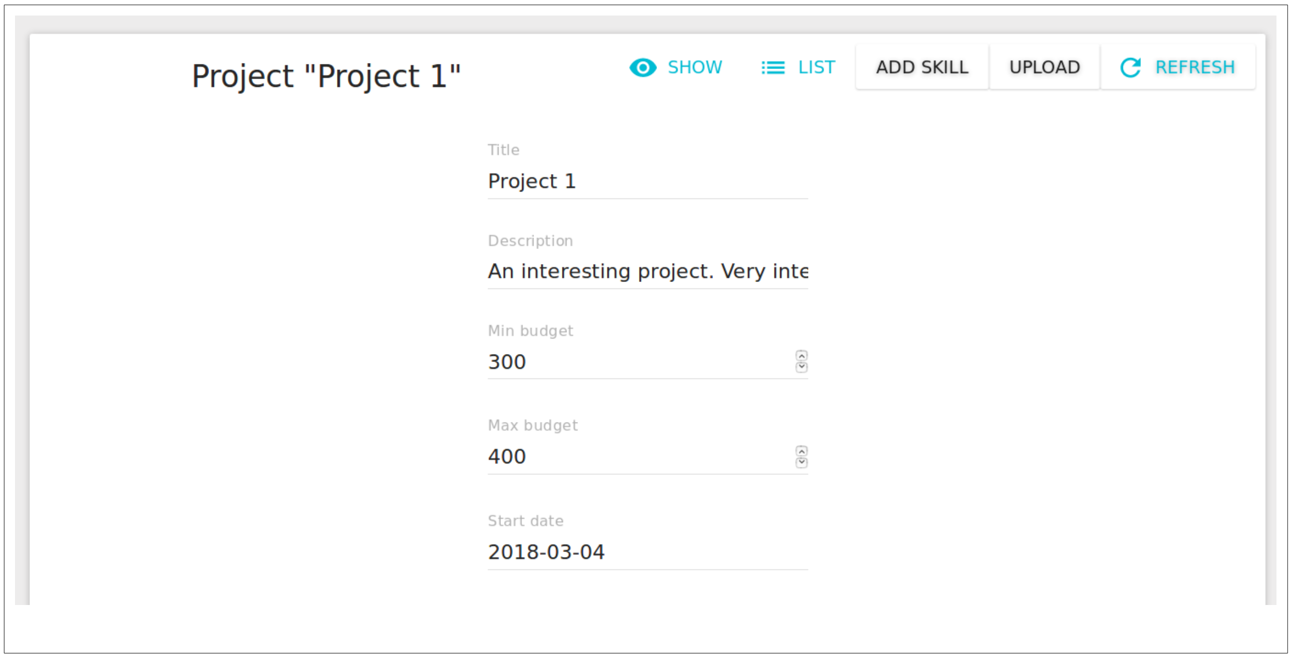

Projects

ADD FILTER | CREATE | REFRESH

TITLE	DESCRIPTION	EMPLOYER			
Project 1	An interesting project. Very interesting	xyz	BID	EDIT	SHOW
Project 2	Hello hello project 2	xyz	BID	EDIT	SHOW
Project 3	Here again project 3	abc	BID	EDIT	SHOW

1-3 of 3

On the edit page, we can click at "Add Skills" and "Upload", for adding skills and uploading files respectively.


Project "Project 1"

SHOW | LIST | ADD SKILL | UPLOAD | REFRESH

Title
Project 1

Description
An interesting project. Very inte

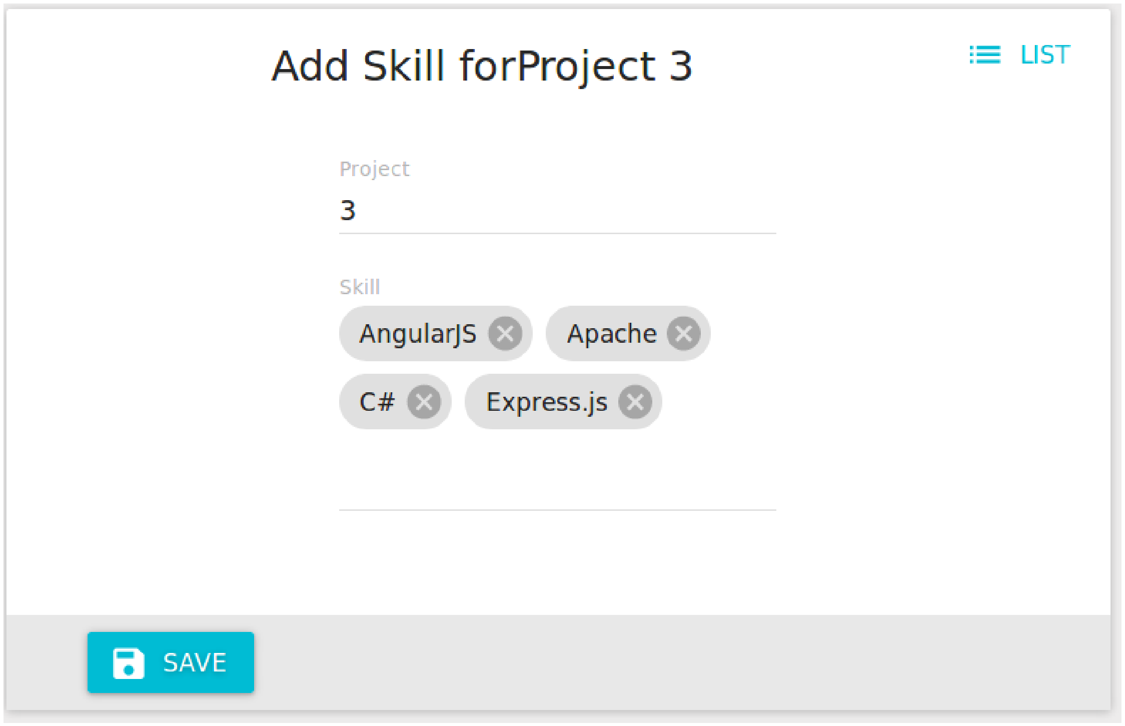
Min budget
300

Max budget
400

Start date
2018-03-04

3. Add Skills

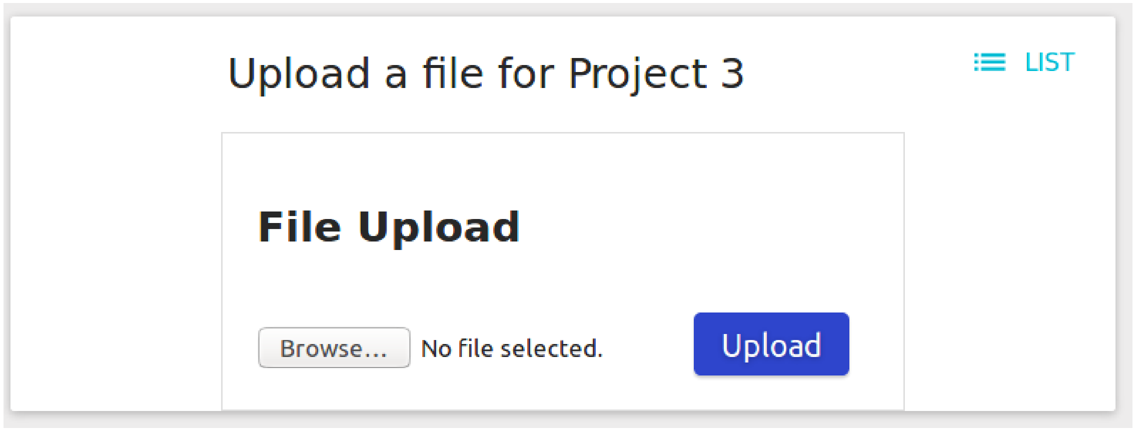
The add skill page allows the project owners to add skills required for this project.


Add Skill forProject 3

Project
3

Skill
AngularJS Apache C# Express.js

4. Upload files
We can also upload files for the project. Once the file is uploaded, page is back to the edit project page.

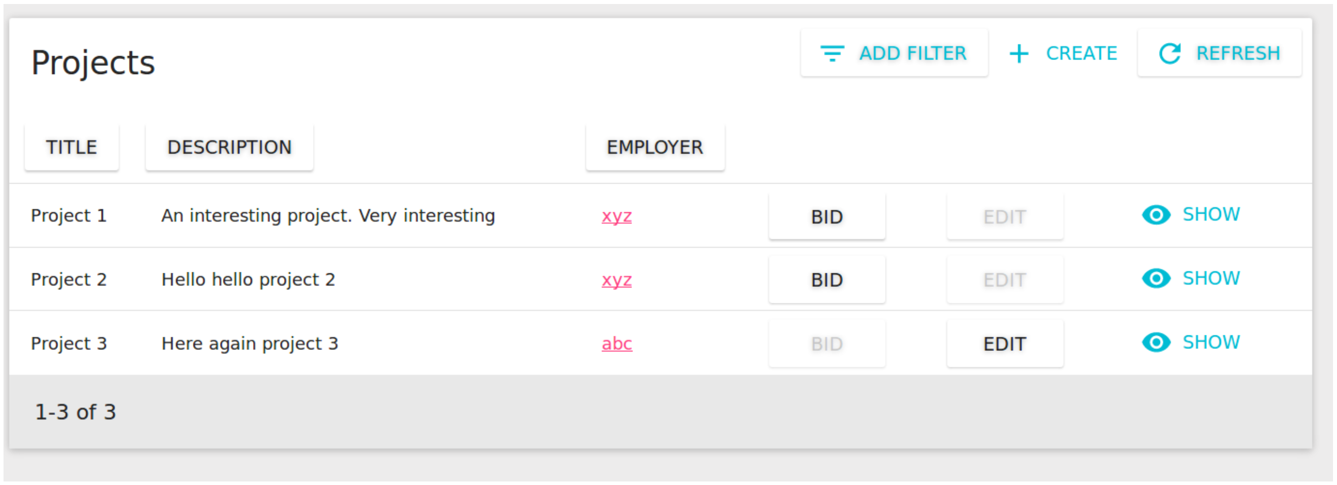

Upload a file for Project 3

File Upload

No file selected.

5. Projects

The projects page shows all the available projects, users can click at "BID" to bid on these projects. Employers are listed links to the user information page. The "Show" button, will open a detail view page for the project.



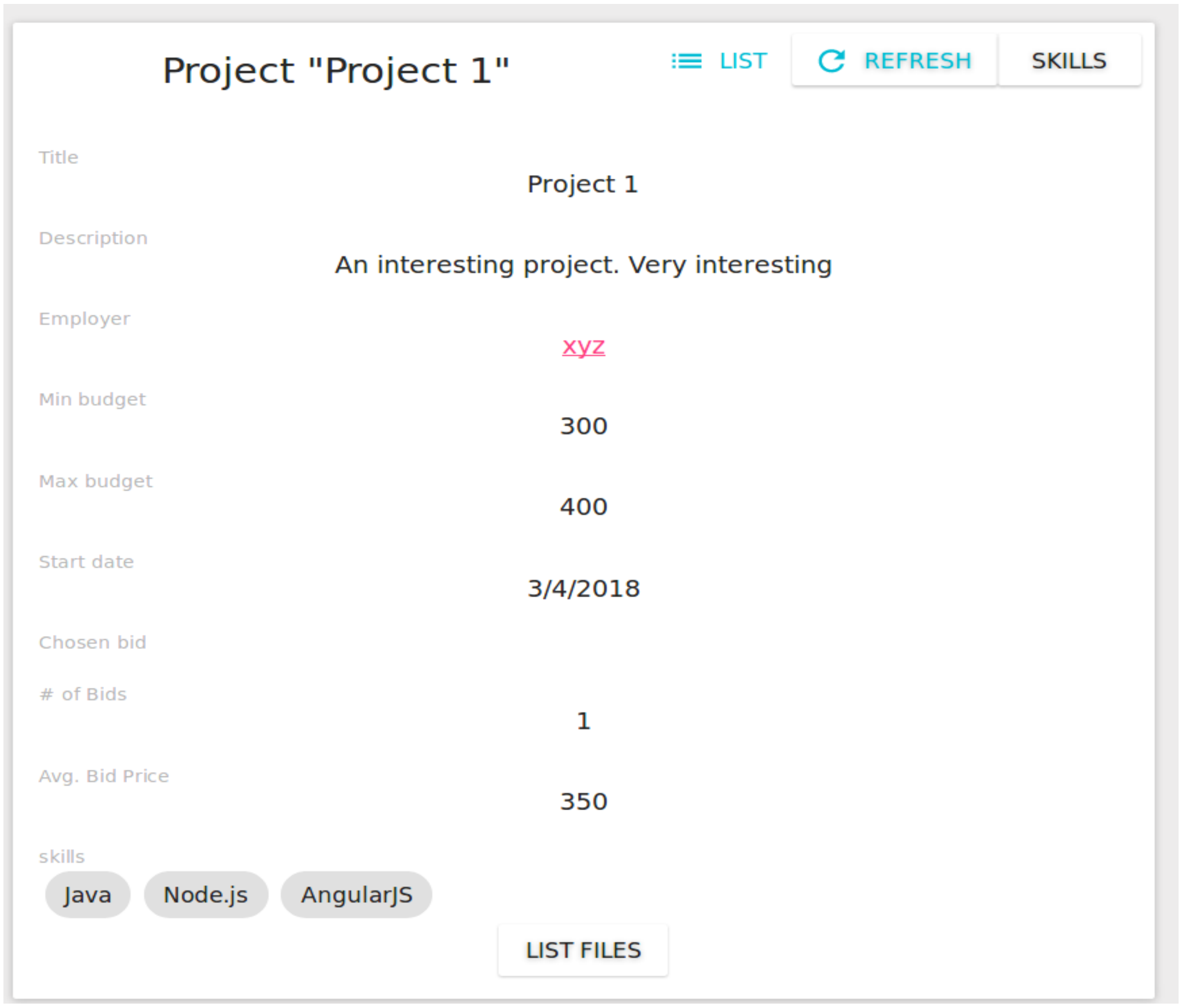
Projects

ADD FILTER CREATE REFRESH

TITLE	DESCRIPTION	EMPLOYER			
Project 1	An interesting project. Very interesting	xyz	BID	EDIT	SHOW
Project 2	Hello hello project 2	xyz	BID	EDIT	SHOW
Project 3	Here again project 3	abc	BID	EDIT	SHOW

1-3 of 3

6. Show Project



Project "Project 1"

LIST REFRESH SKILLS

Title	Project 1
Description	An interesting project. Very interesting
Employer	xyz
Min budget	300
Max budget	400
Start date	3/4/2018
Chosen bid	
# of Bids	1
Avg. Bid Price	350

skills

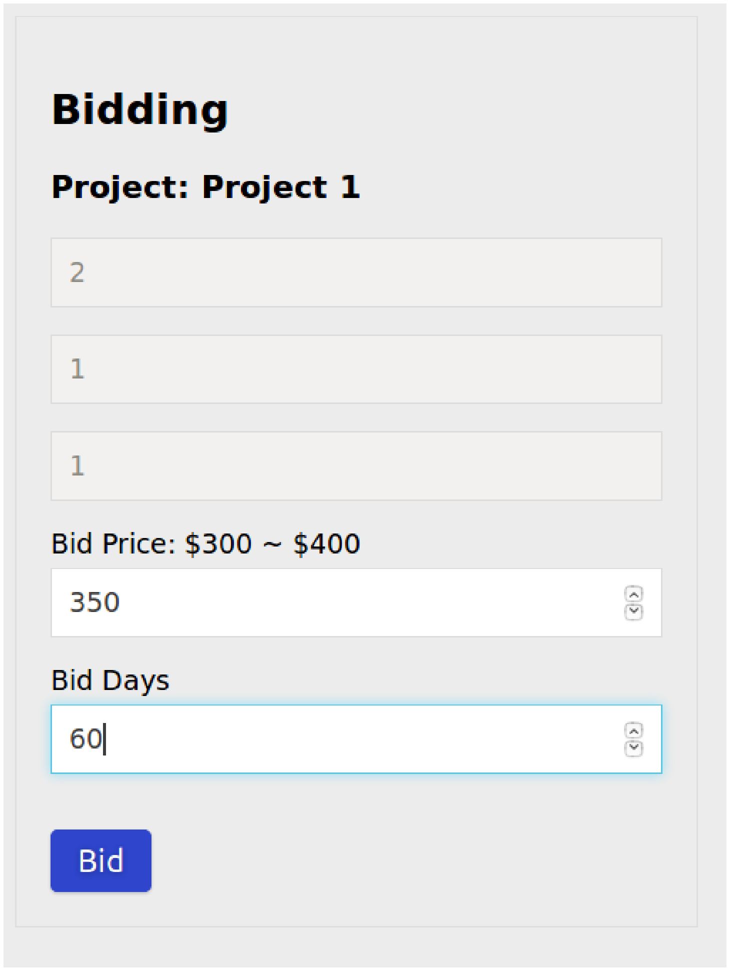
Java Node.js AngularJS

LIST FILES

On the show project page, we can see the regular projects details, as well as, number of bids, average bid prices, and whether a bid is chosen. It also shows the skills required for the project. If you click at the "LIST FILES" button, the page will be redirected to the file list page for this particular project.

7. Bid

By clicking on the "BID" button on a project, the bidding page is open. Project ID, User ID and employer ID are prefilled on disabled text filled. To bid the project, users only need to fil out the bid price and days. When clicking on Bid, the page is redirected to the bids list.



Bidding

Project: Project 1

2

1

1

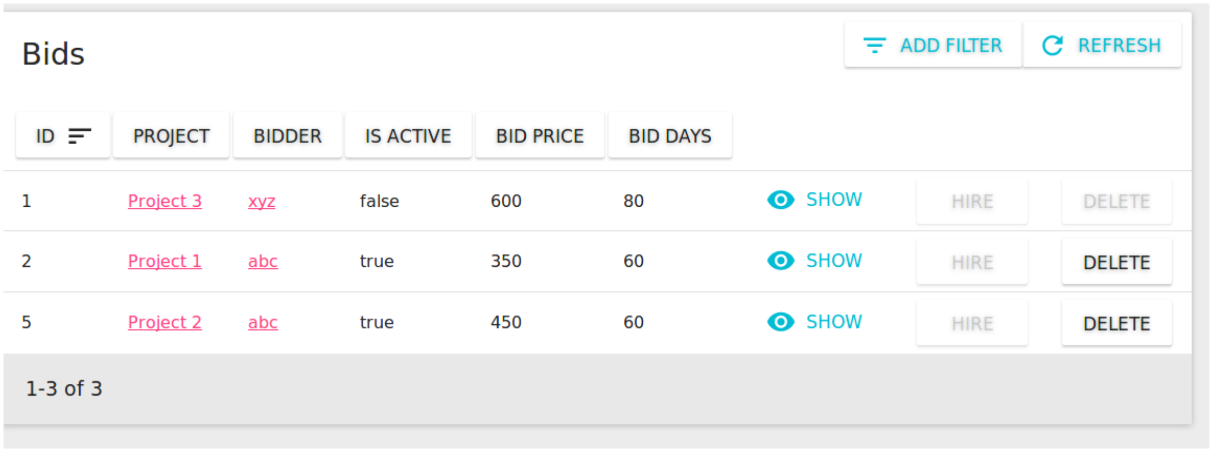
Bid Price: $300 ~ $400

350

Bid Days

60

8. List of bids:
The list of bids page shows all the bids placed. Project and bidders on the list are links to the project and user show pages. "HIRE" button is only available when the current user is the owner (employer)of the project and the project is active. "DELETE" button is only available when the current user is he bidder of the project and he project is active.


Bids

ADD FILTER | REFRESH

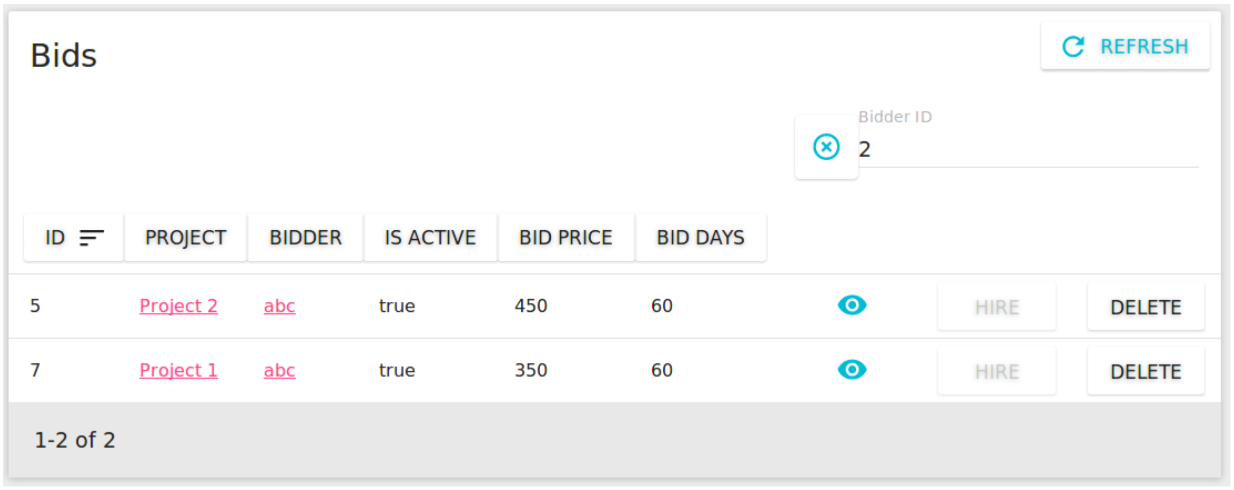
ID	PROJECT	BIDDER	IS ACTIVE	BID PRICE	BID DAYS			
1	Project 3	xyz	false	600	80	SHOW	HIRE	DELETE
2	Project 1	abc	true	350	60	SHOW	HIRE	DELETE
5	Project 2	abc	true	450	60	SHOW	HIRE	DELETE

1-3 of 3

My Dashboards:

1. My Bids:

Clicking on the "My Bids" item on menu (left), brings up the list page of bids that are placed by the current user.


Bids

REFRESH

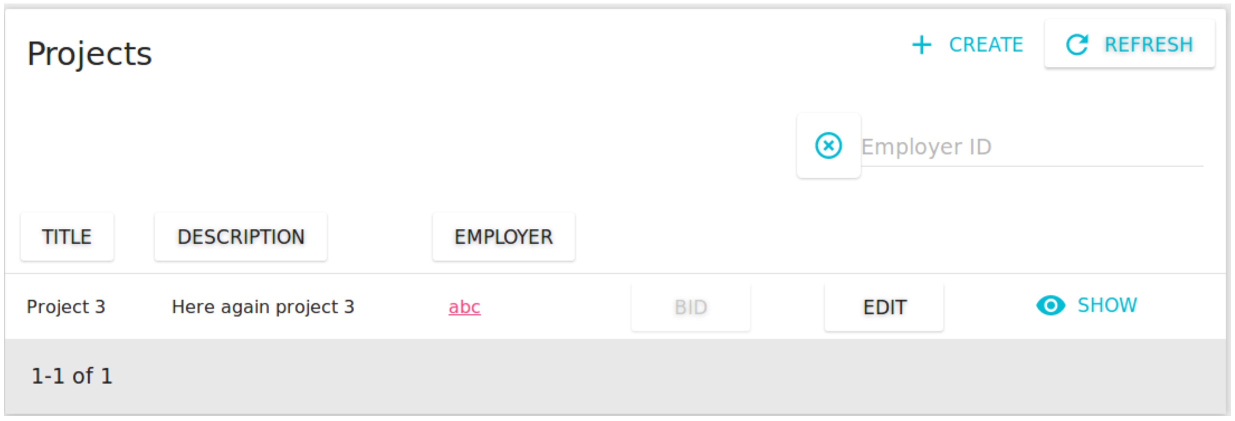
Bidder ID: 2

ID	PROJECT	BIDDER	IS ACTIVE	BID PRICE	BID DAYS			
5	Project 2	abc	true	450	60		HIRE	DELETE
7	Project 1	abc	true	350	60		HIRE	DELETE

1-2 of 2

2. My Projects:

Clicking on the "My Projects" item on memu(left), brings up the list page of projects that are created by the current user.



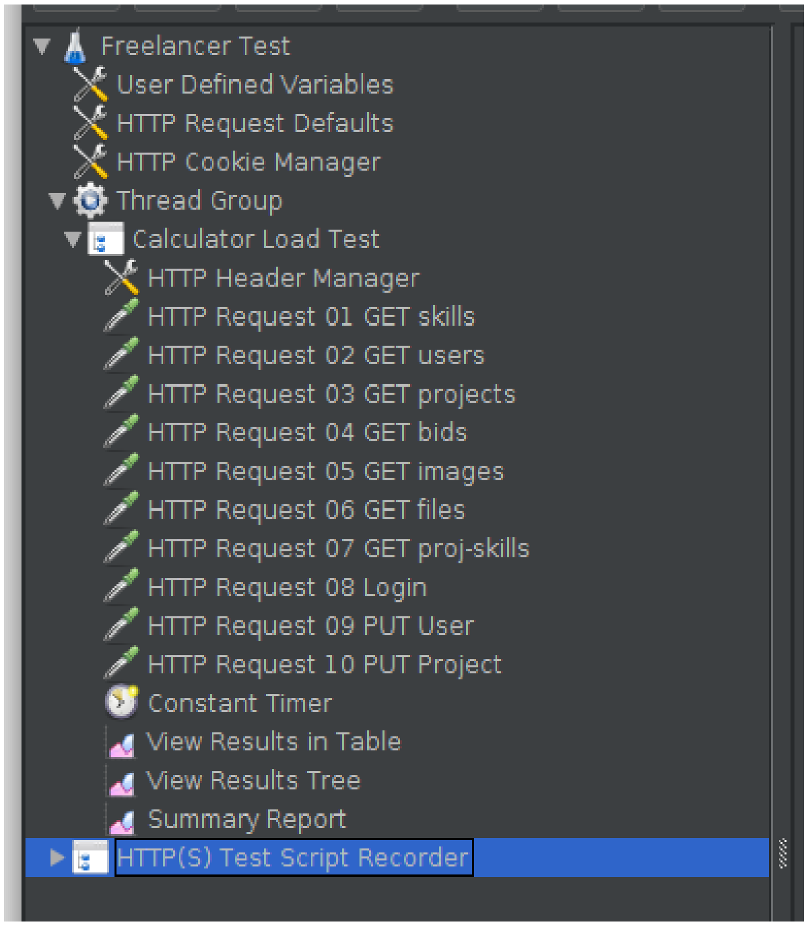
TITLE	DESCRIPTION	EMPLOYER			
Project 3	Here again project 3	abc	BID	EDIT	SHOW

1-1 of 1

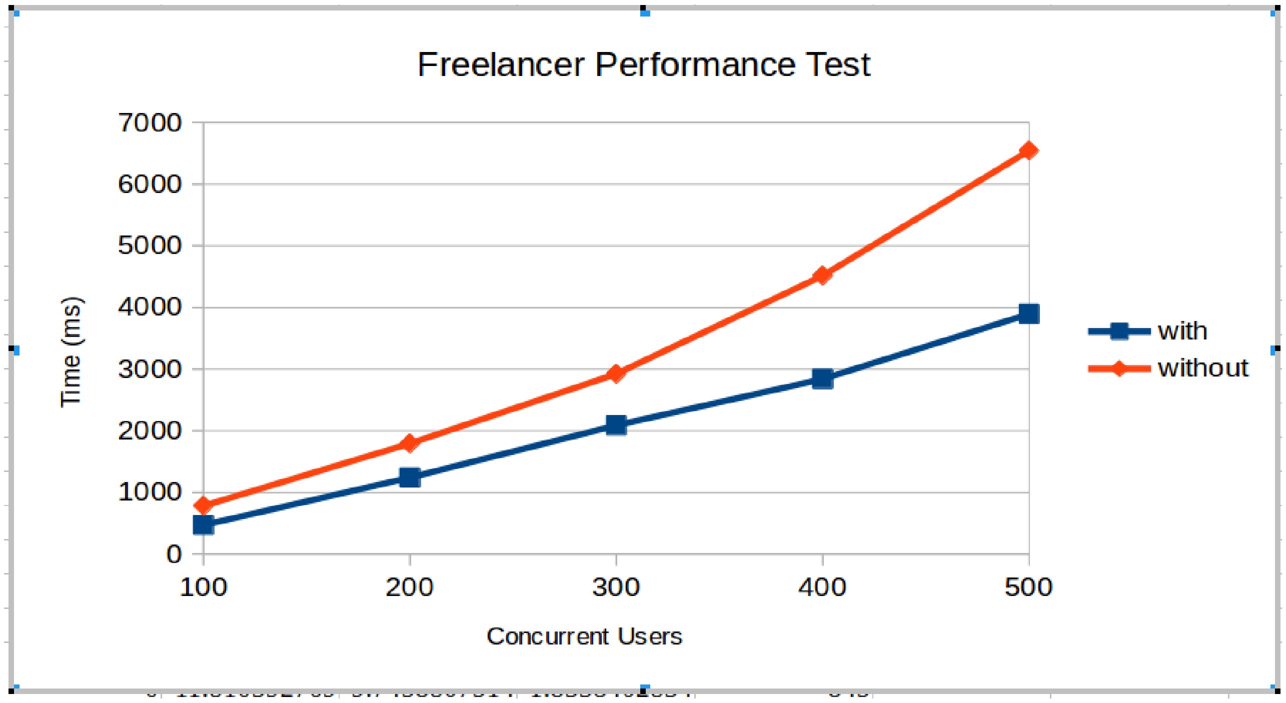
Performance:

The JMeter testing results is shown below for 100, 200, 300, 400 and 500 concurrent users with (blue) and without(orange) connection pooling. Each user sends out 50 random requests, chosen from the 10 options, as shown below. As we can see, with connection pooling, the response time increases with a higher rate than when using connection pooling.

The better performance of the connection pooling is attributed to the design of reusing connection. Web applications usually have a large number of small db requests, that the overhead for creating and destroying a connection for each request is just not worthwhile.



```
▼ Freelancer Test
    User Defined Variables
    HTTP Request Defaults
    HTTP Cookie Manager
  ▼ Thread Group
    ▼ Calculator Load Test
        HTTP Header Manager
        HTTP Request 01 GET skills
        HTTP Request 02 GET users
        HTTP Request 03 GET projects
        HTTP Request 04 GET bids
        HTTP Request 05 GET images
        HTTP Request 06 GET files
        HTTP Request 07 GET proj-skills
        HTTP Request 08 Login
        HTTP Request 09 PUT User
        HTTP Request 10 PUT Project
        Constant Timer
        View Results in Table
        View Results Tree
        Summary Report
  ▶ HTTP(S) Test Script Recorder
```



Q & A:

1. Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.

In this lab project, I used bcrypt for password encryption. Bcrypt is one of the most common password hashing functions used today. It's the default password hashing algorithm for linux systems. Like many other one-way hashing key stretching algorithms, bcrypt makes certain password cracking by hackers more difficult.

There a family of symmetric encryption algorithms. AES. the advanced encryption standard, requires a key to encrypt and decrypt information. The key needs be shared among users and servers, which is inconvenient for web applications.

Another famous asymmetric algorithm with key is RSA. To gain access, users use private keys to match with corresponding public keys. This is encryption method is rather secure. One reason is the large key sizes, which significantly increases the the difficulty for hackers. Yet, it is inconvenient to use the keys pairs for daily web browsing. User usually don't have the knowledge or patience to set up such systems.

Base64 can be use easily in two-way hash function. Because of this simplicity, it is vulnerable to password cracking. It's not recommended for password encryption.

2. Compare the results of graphs with and without in-built mysql connection pooling of database. Explain the result in detail and describe the connection pooling algorithm if you need to implement connection pooling on your own.

Connection pooling is a caching approach to maintain a group of database connections, thus the server can save resources for other activities and reduce the overhead of creating connections.

To implement a connection pool, the following functionalities are required to be covered.

(1) Initialize the pool of connections under the maximum pool size.
(2) A borrow connection function that allows other processes to borrow a connection from the pool. An available connection is chosen for borrow. The borrowed connection is popped out from the pool
(3) Once process finished the query, it returns the connection back to the pool

3.What is SQL caching? What all types of SQL caching are available and which suits your code the most. You don’t need to implement the caching, write pseudocode or explain in detail.

SQL caching, or more broadly speaking database caching, is a process to increase the performance on retrieving data. Generally speaking, web applications fires a large amount of requests to acquire data from databases, and a large proportion of these requests are accessing a subset of all the data in the database. Without caching, same sql queries need to be executed repeatedly, which wastes resources and delays the response. Instead, we can cache such information for fast read.

There are different types of caching, database caching, table caching, query plan caching and result set caching. For example. Redis is a distributed in-memory database system, which basically caches everything in its databsae. Memcached caches the result set of queries. This system is suitable for applications that frequently queries relatively static datasets. We can also cache tables, if a good variety of queries are performed on a small set of tables with reasonable data volume. We can also cache complex queries and query plans, thus the database doesn't have to recalculate and optimize how to efficiently execute the queries.

There aren't many complex queries in this application. Our database is MySQL, a RDMS that can have large data volume and don't have good scalability. Potentially, our user base can grow really fast, which leads to large increase in tables size. Thus caching the entire database, major tables or complex queries, are either impractical or unnecessary. What we could do is to frequently accessed data sets and small lookup tables.

Here's an example of caching result set in a Least Recently Used (LRU) cache.

```
class LRUCache:

    def __init__:
        # intialize the cache with Linked hash map
        lhm = new linked hash map;

    def get(query):
        move (query, result)  to the beginning of lhm
        return lhm.get(query)

    def put(query, result):
        put (query, result) to the beginning of lhm
        if lhm.size > threshold:
            remove the end of lhm.
```

<u>4. Is your session strategy horizontally scalable? If YES, explain your session handling strategy. If NO, then explain how you can achieve it.</u>

To certain degree, my session strategy is horizontally scalable. I built this application with RESTful API, designed to be stateless. The RESTful server (Express.js and Node.js) don't maintain use session information. To inquire the backend, the frontend application will send over all needed information. This means that the request can be routed to any server instances and get the same results back.

There are two problems here. First of all, the the database MySQL is not configured to be scalable. Even though, our backend server can be scaled to multiple, they will end up querying the same database instance which will certainly cause congestion.

Another problem, is that user session is maintained completely at the client side. When the user closes the browser or even refresh the web page could lead to the lose of states. To alleviate this issue, I persisted data on the local storage. However, the storage is limited and, sometimes, it could be out of the application's control.

The right way to maintain scalable sessions, is to let the backend server handle the session information and strategically store the session information in a scalable database.