Lab – 2 Report

**Introduction**:

The goal is to learn to design and implement distributed service oriented application using Kafka. Prototype of freelancer application is made using node.js, kafka, react, and MongoDB. Node is separated into two parts. “Backend services” are designed as consumer and “frontend services” as consumers. Passwords are encrypted using Passport. The server performs the basic user functionality, post project, home, details view, dashboard, transactional manager.

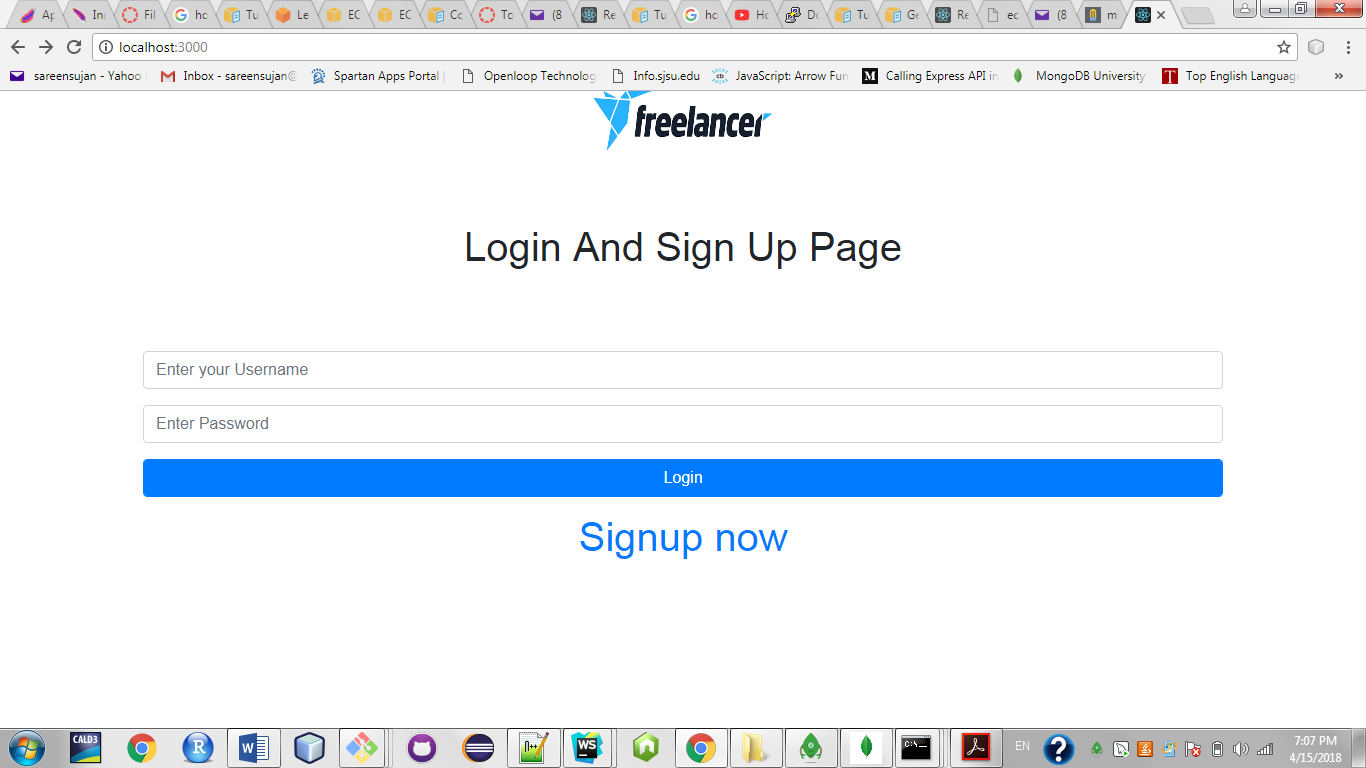
The whole application is then deployed on Amazon AWS and using mlab.

**System Design:**

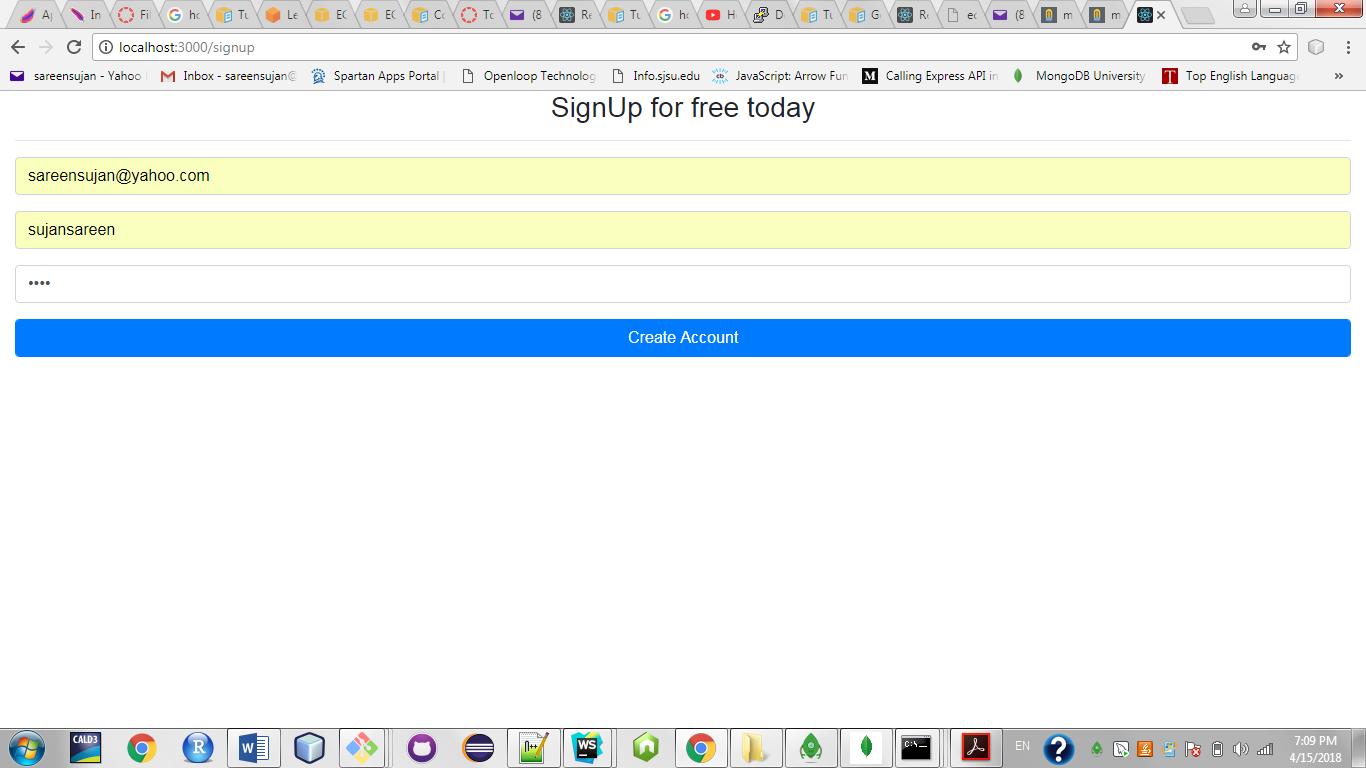
When a user requests something, the request goes to producer (Node.js) and from producer, it goes to consumer through zookeeper. ZooKeeper is used for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Then from consumer, it goes to database(MongoDB). Then the response goes from database(MongoDB) to consumer and then producer to consumer (using Zookeeper) and then consumer to the user. The whole application is then deployed on Amazon AWS and using mlab.

**Screenshots for running application on local machine:**

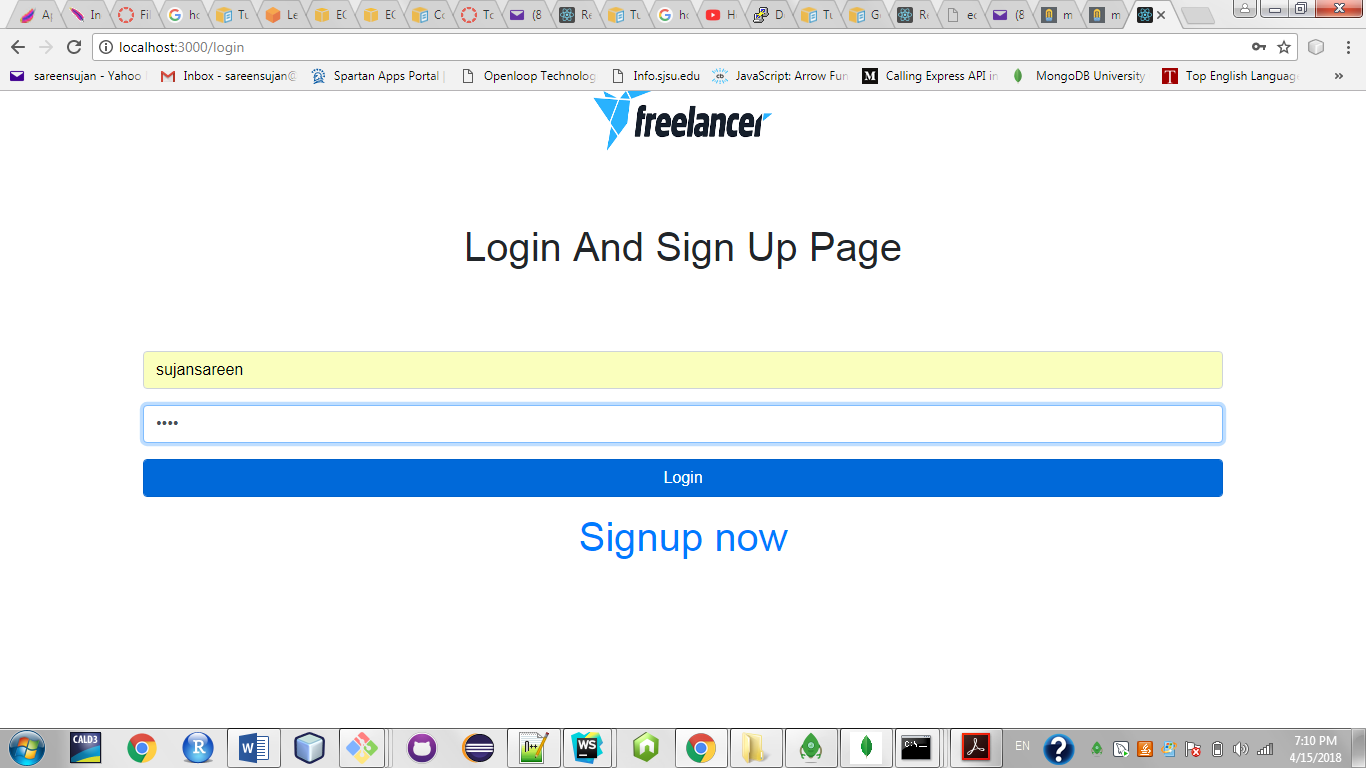
**1)** Login and Sign up page



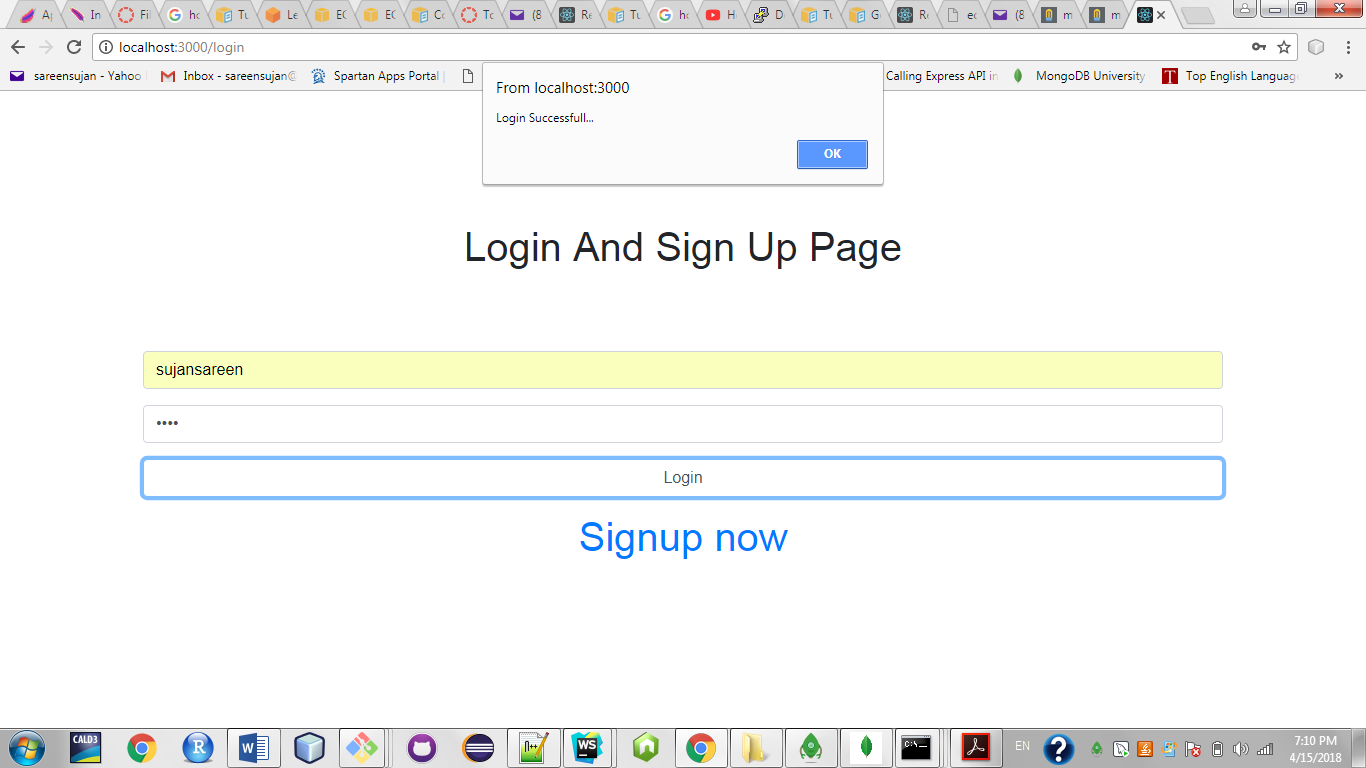
2) Login and SignUp page during signing up a new user:



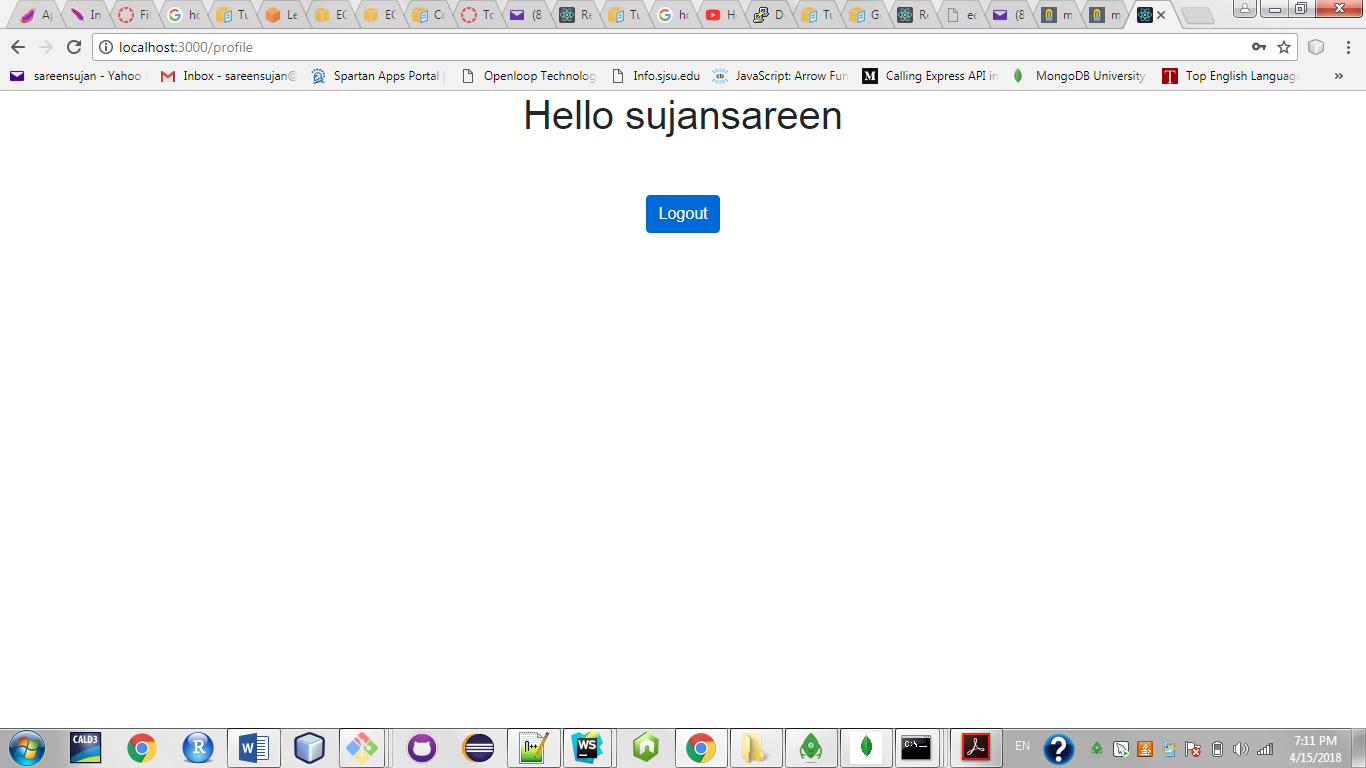
3) After signing a user and then loging him in:

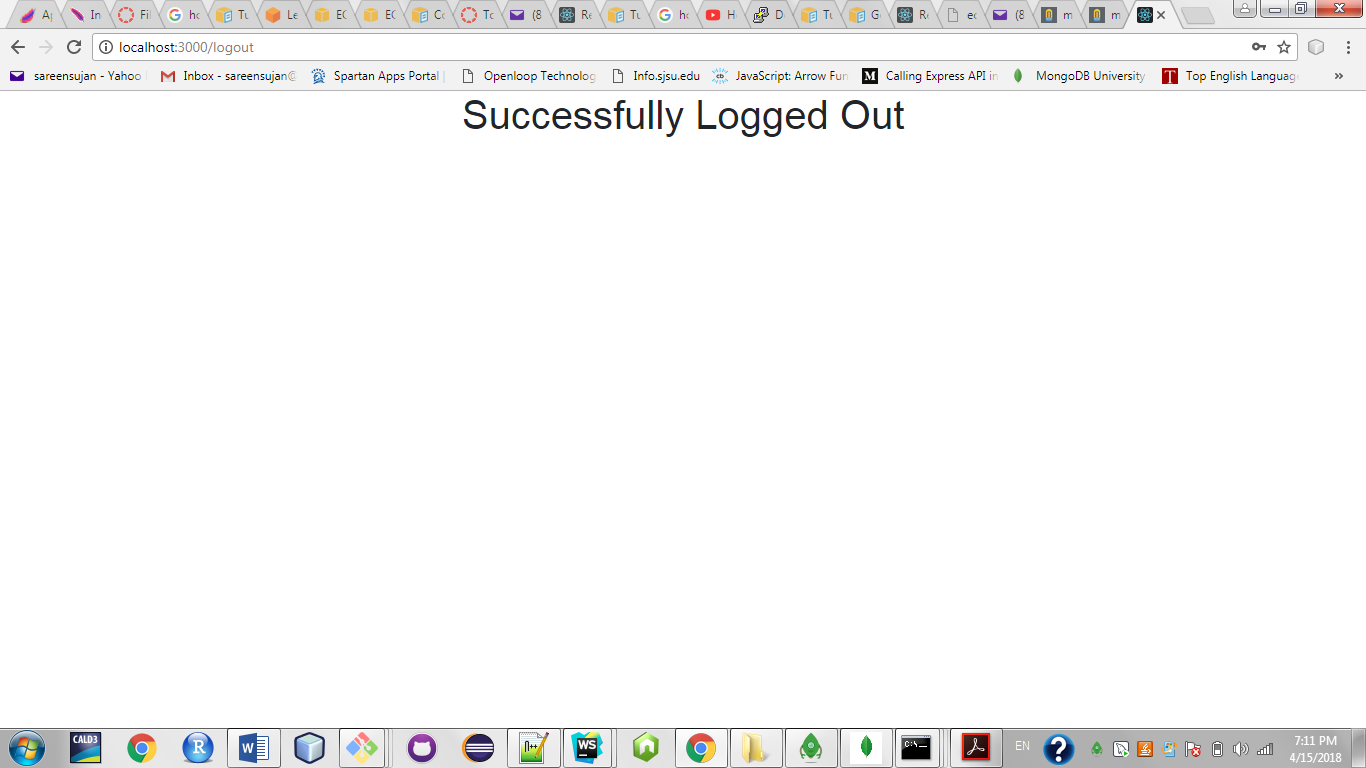


4) After a user is successfully logged in:

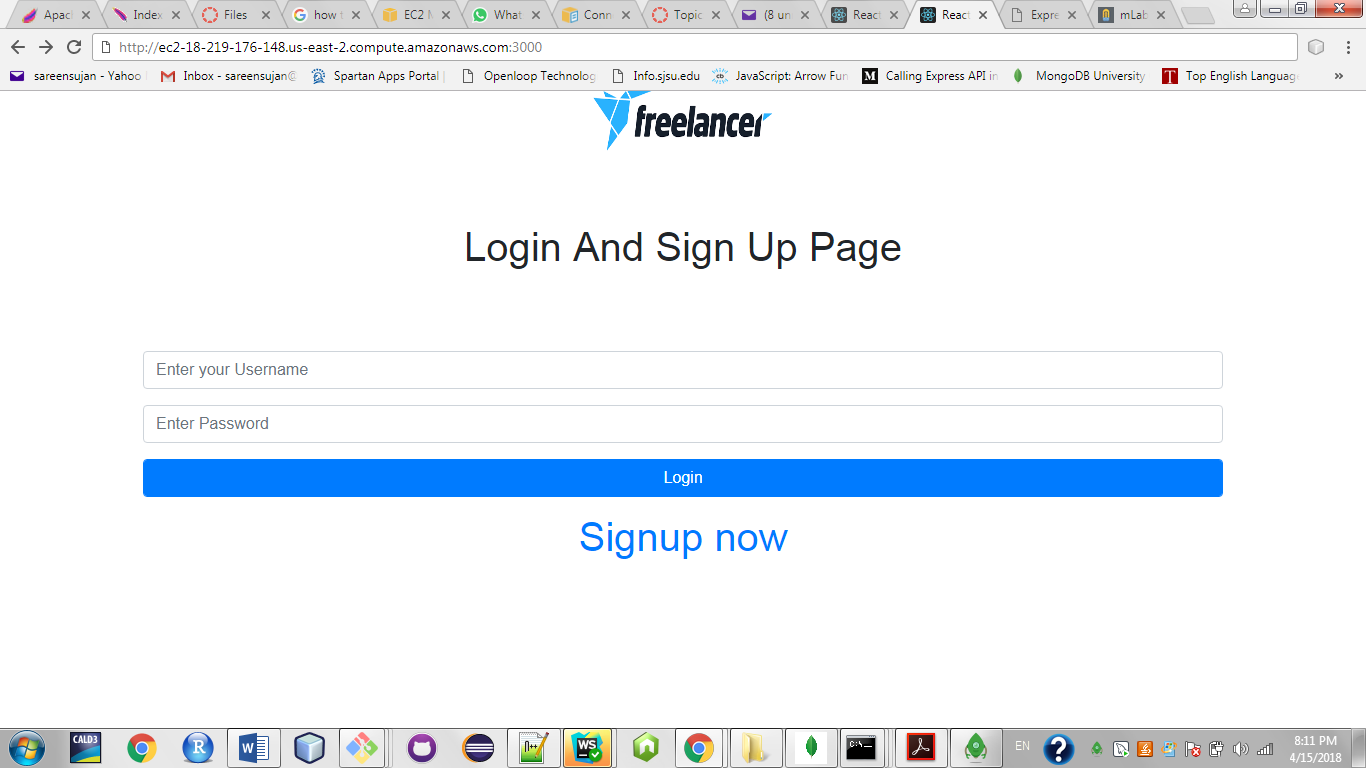


5) The next page which appears after a user logs in:



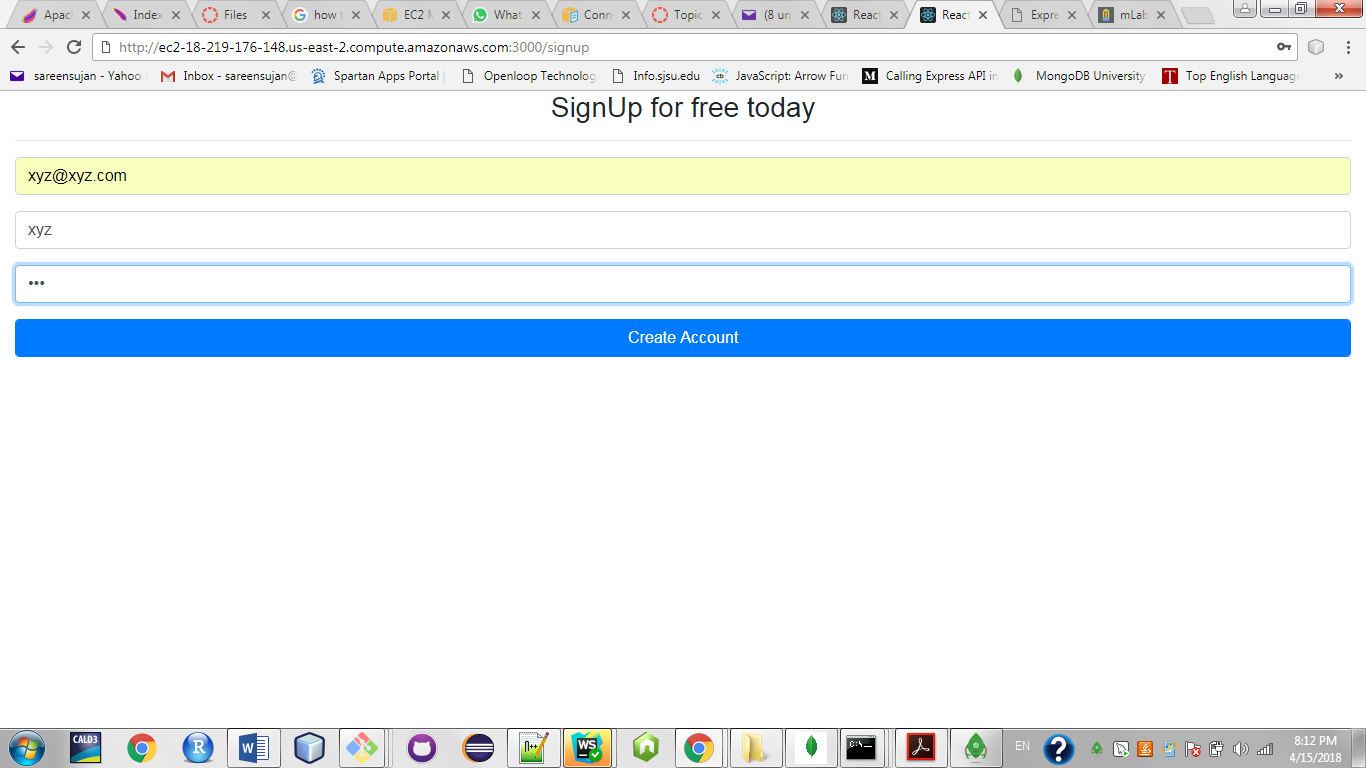


6) After a user logs out:

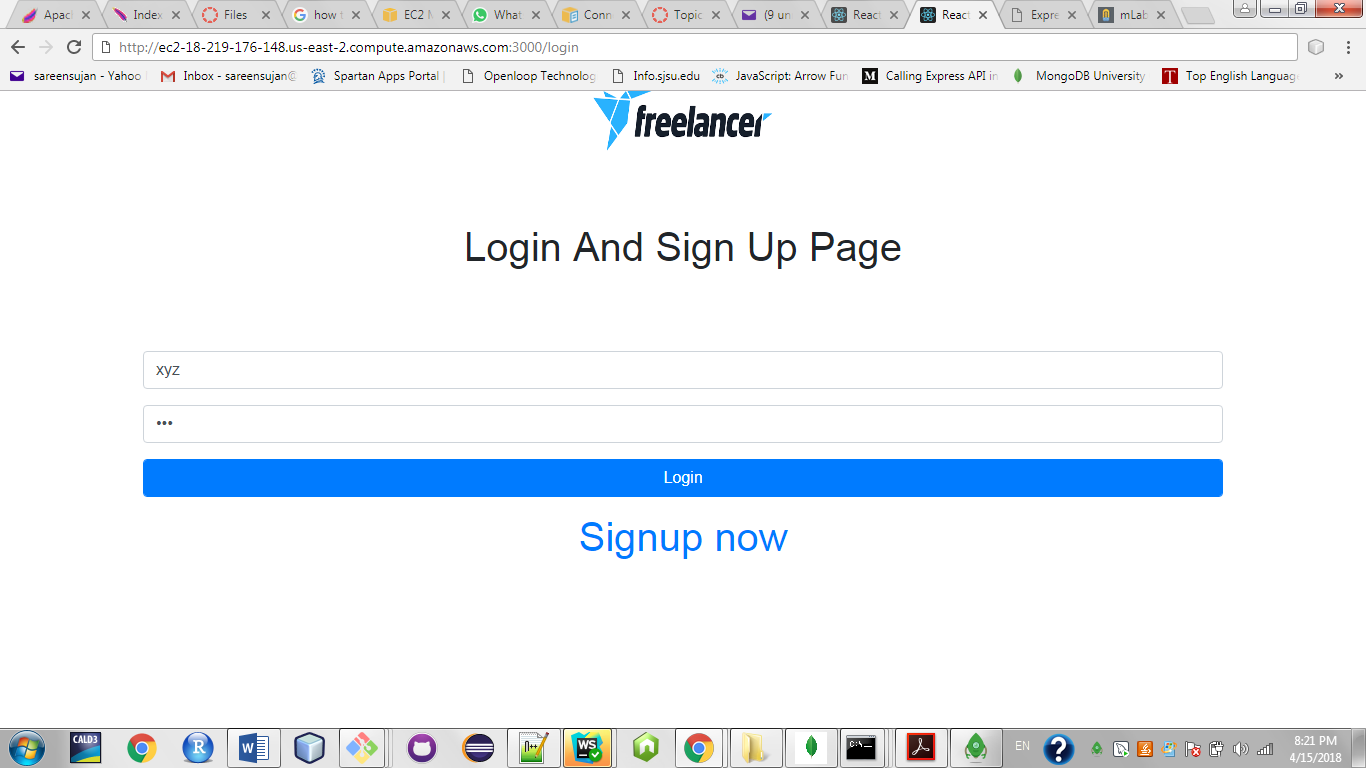


**Running application after deploying it over AWS cloud**

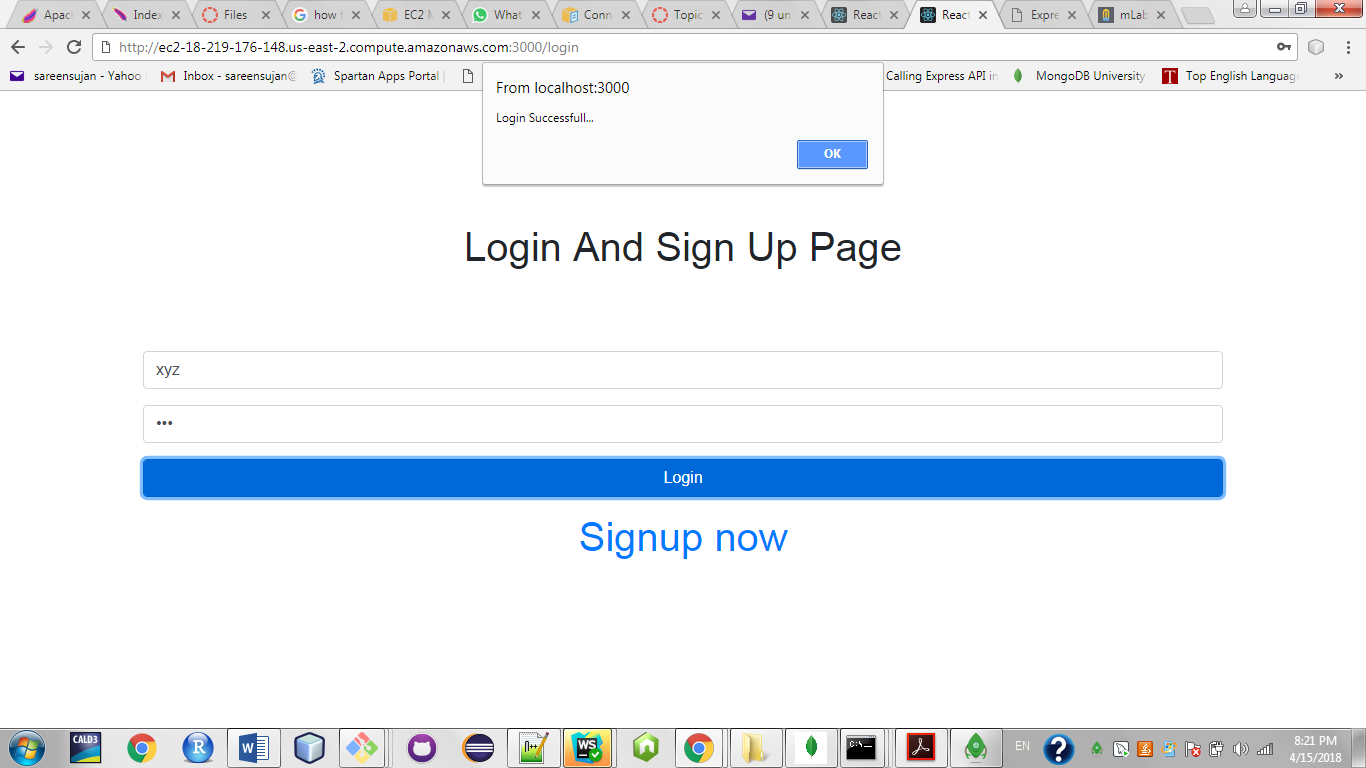
**1)** Login and Sign up page



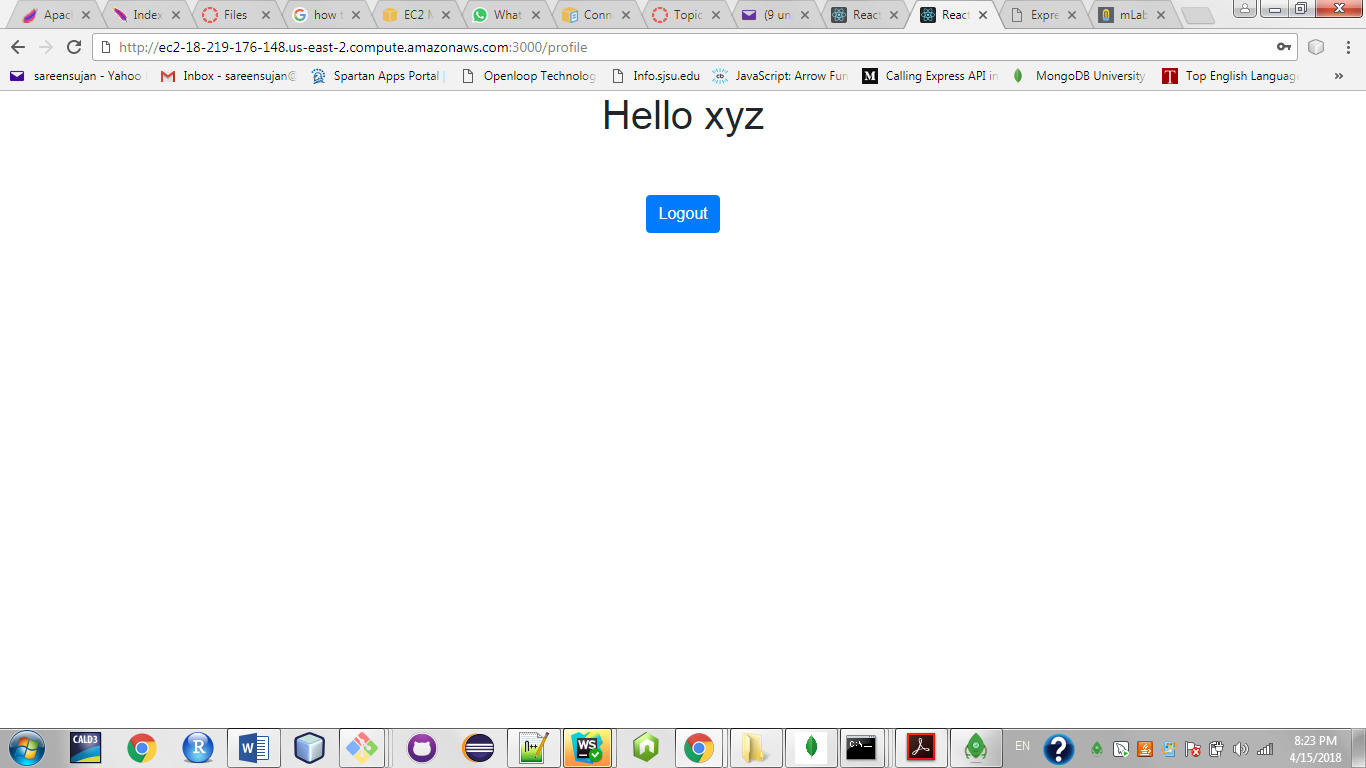
2) Login and SignUp page during signing up a new user:



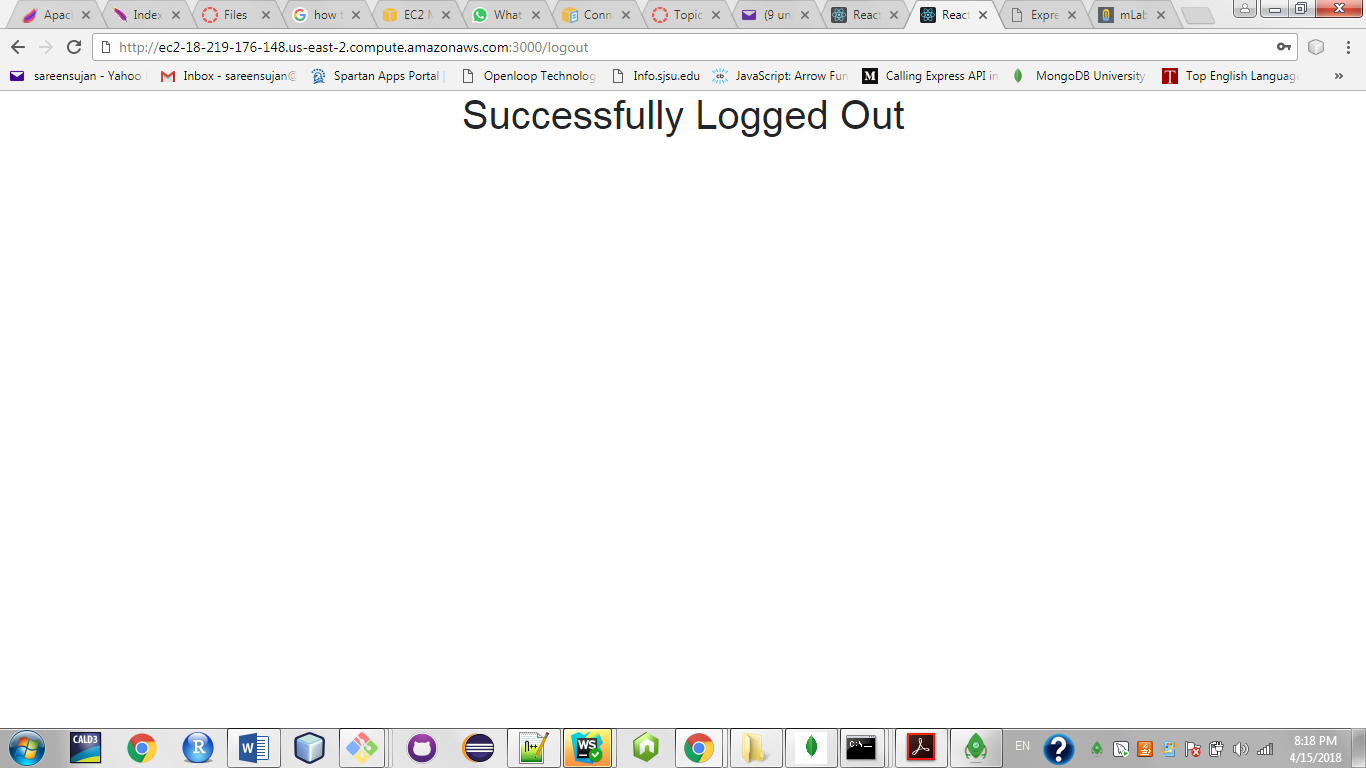
3) After a user is successfully logged in:



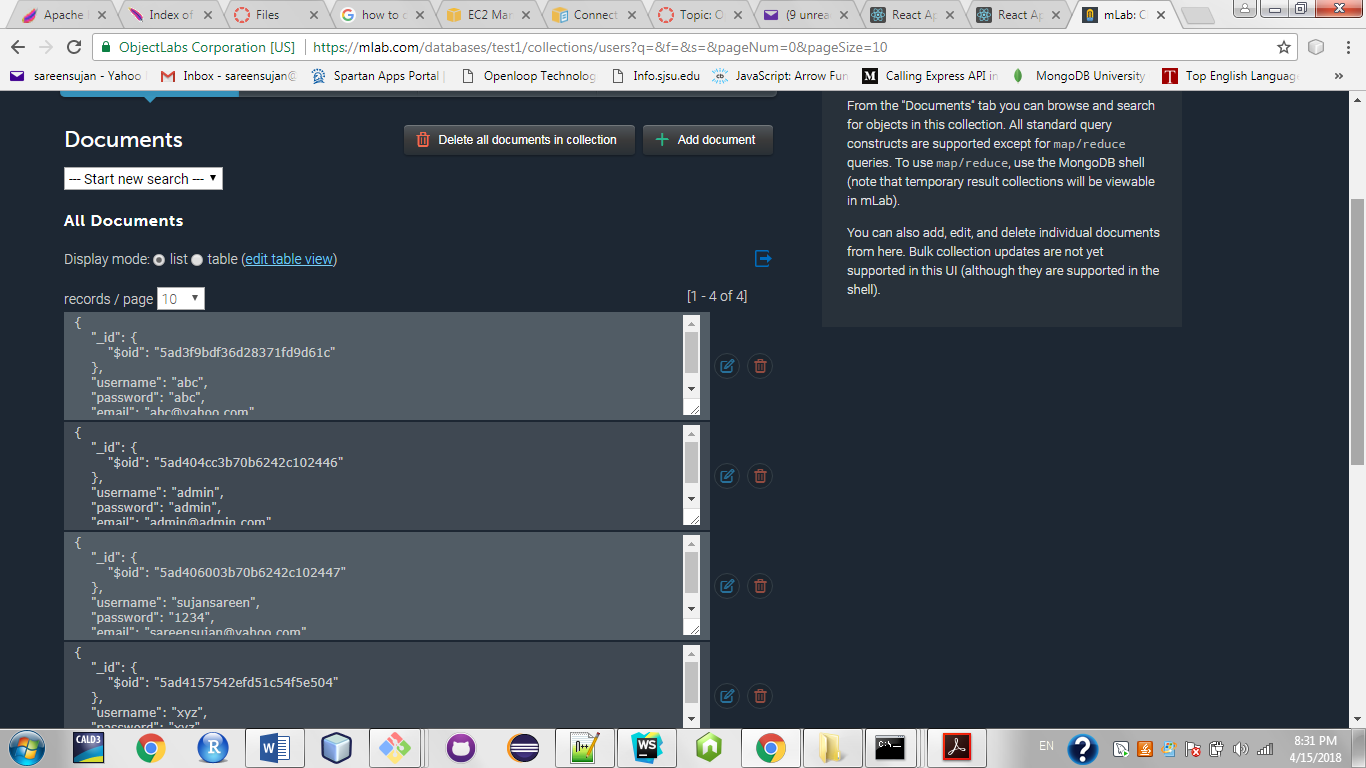
4) The next page which appears after a user logs in:



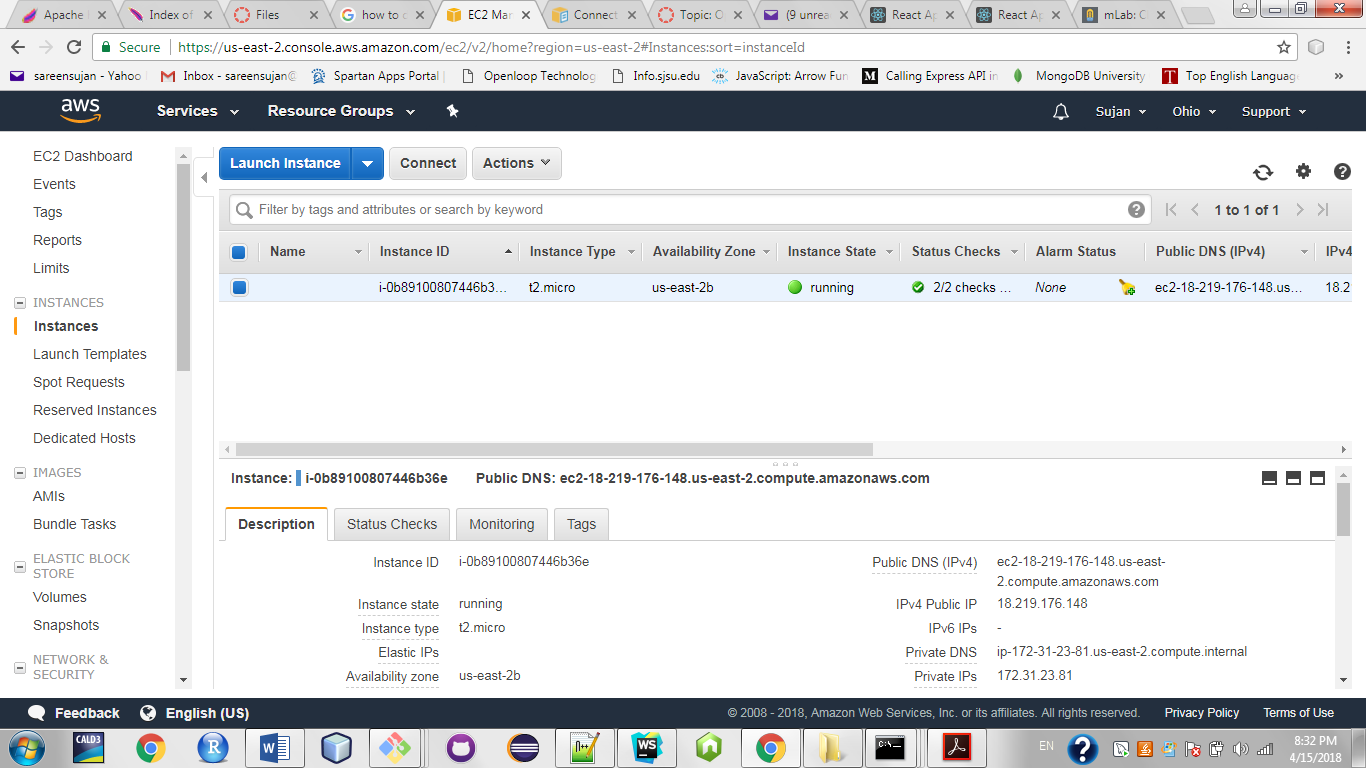
5) After a user logs out:



**Screen shot from MLab**



**Screenshot of Amazon instance:**



Ans 1) With passport authentication, any user can maintain his profile in a single location. In other words, he does not need to remember separate passwords and user names for different web sites. User’s sign-in and sign-out pages are hosted centrally rather than being specific to different web sites.

Whereas, with client-session strategy (used in lab-1), client has to enter his credentials each time when he moves to different web site. Client-session strategy is useful when client moves within a website from one page to another.

Ans 2)

With Kafka:

Kafka has advantages over traditional messaging systems. Kafka can scale the processes over the members of consumer group. Kafka is a multi-subscriber as it can send messages to multiple consumer groups. Kafka has a good storage system which is fault tolerance. It has low latency ,which help increase the response time. It can also handle high volumes of data.

Without Kafka:

Traditional messaging system can’t have both the properties of scaling the processes and being a multi-subscriber at the same time. Without kafka, the order of the messages can be lost or the notion of parallelism can be compromised.

Kafka is faster than traditional messaging system in performance. Kafka access the disk in a sequential manner which helps it to have a speed similar to the speed of memory. Kafka does not use memory to write anything because it doubles the size of the data stored.

Also, kafka uses Zero Copy which copied data directly from Kernel Context to the Kernel Context. It also batches the data into chunks which decreases the latency.

Ans 3) For MongoDB:

It can be used where data availability and size of data have higher priorities. For ex., storing project details (Project Name, Description, Files, Skills, Budget Range, Average

Bid); user name and its basic information (Name, Email, Phone Number, About Me, skills); information about bids (Bid, Period in days, Avg. Bid)

For MySQL: It can be used where security has the priority.

For ex., for storing credits cards information, for storing confidential transactions information, for storing financial account information, for storing monetary related information which should be confidential, for storing passwords.

AWS credentials:

Link to deployed website:

http://ec2-18-219-176-148.us-east-2.compute.amazonaws.com:3000/

Username: [sareensujan@yahoo.com](mailto:sareensujan@yahoo.com)

Password: Software&1

Mlab credentials:

Username: [sujansareen](mailto:sareensujan@yahoo.com)

Password: Software&1

GitHub credentials:

Email: [sujan.sareen@sjsu.edu](mailto:sujan.sareen@sjsu.edu)

Password: Software&1