

**Project Report – Lab 3: Using**

**GraphQL and Apollo Client**

**CMPE 273 | Enterprise Distributed Systems | 13th May ‘19**

Submitted by : Shivang Mistry - 013823108

**Introduction**

This lab aims to implement several functionalities of Canvas which a Learning Management System for San Jose State University using GraphQL and Apollo Client. Canvas allows student and faculties collaboration on a single uniform platform and exchange resources with ease and efficiency. GraphQL is a data querying and manipulation language for APIs. Apollo client is a client-side library for leveraging data fetching and manipulation from GraphQL API. This application makes use of them to perform basic functionalities.

**System Design**

**Assumptions: -**

* User registers with a unique SJSU ID.
* Database used is non-relational MongoDB.

**Functional requirements: -**

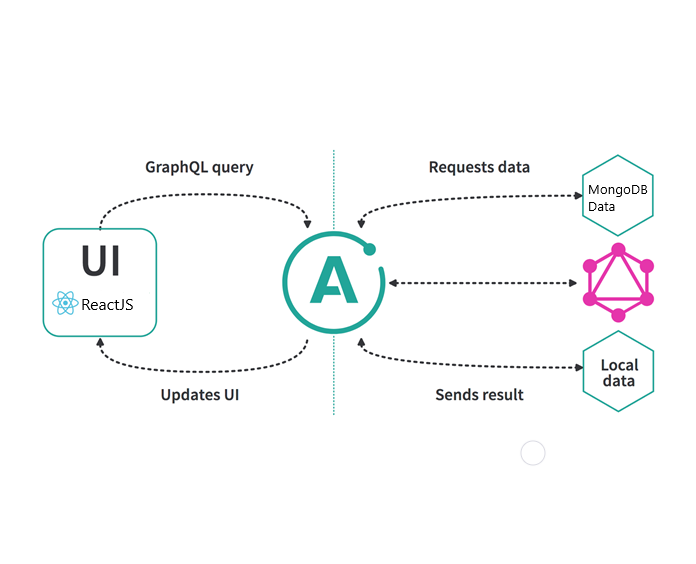
* User can create and login to their account.
* User can maintain and update their profiles.
* Faculty can create courses. Student can enroll to available courses.
* View courses created/enrolled. View course info.

**Technologies: -**

**Software:** ReactJS, NodeJS, MongoDB, GraphQL, Apollo Client, React-Bootstrap.

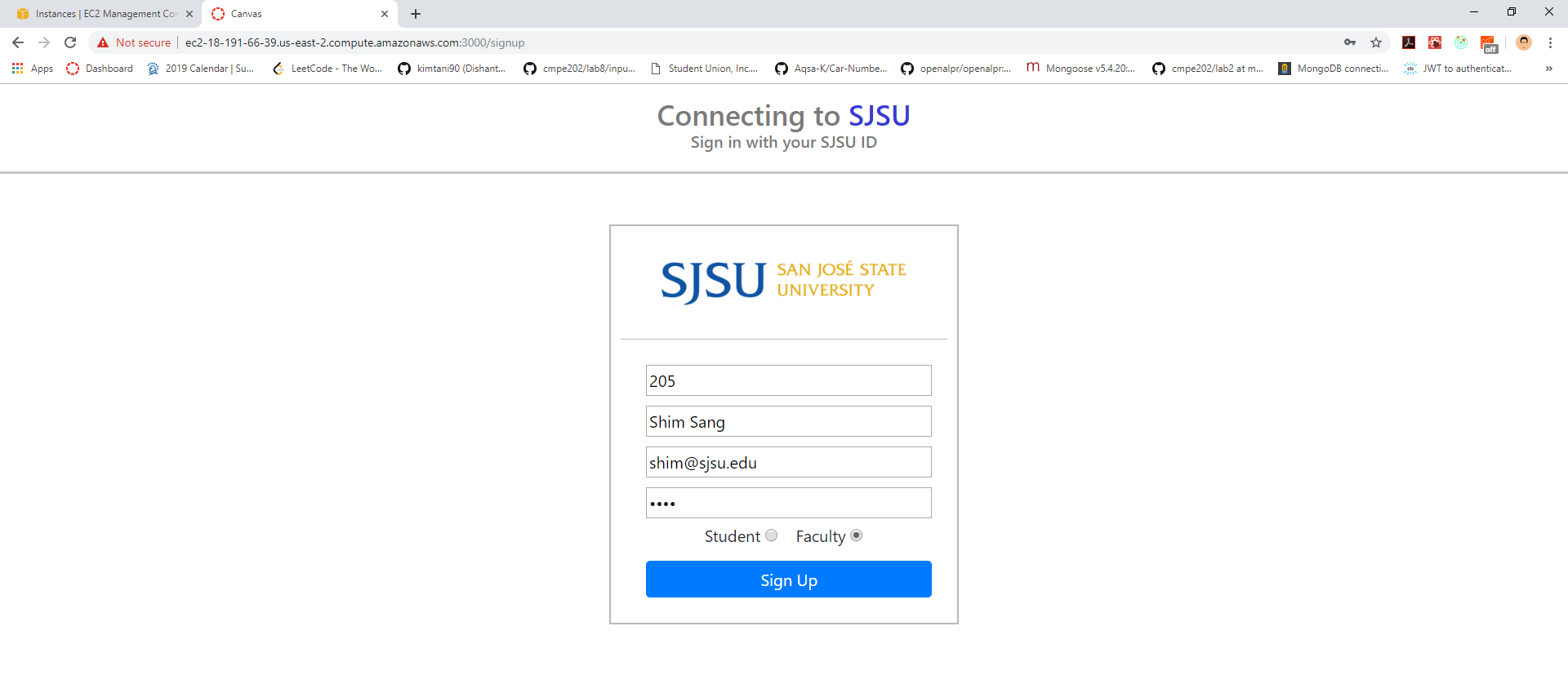
**Hardware:** Google Chrome v75 or greater preferred.

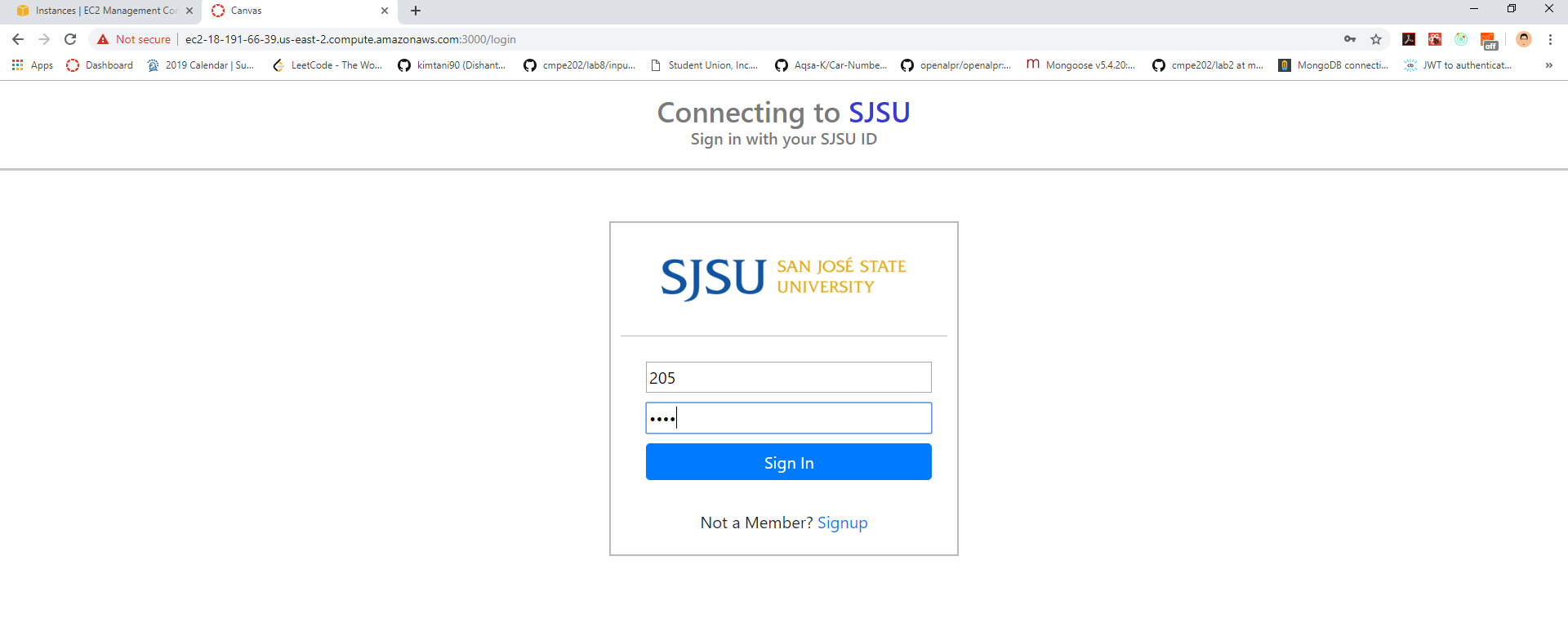
**Architecture Diagram: -**

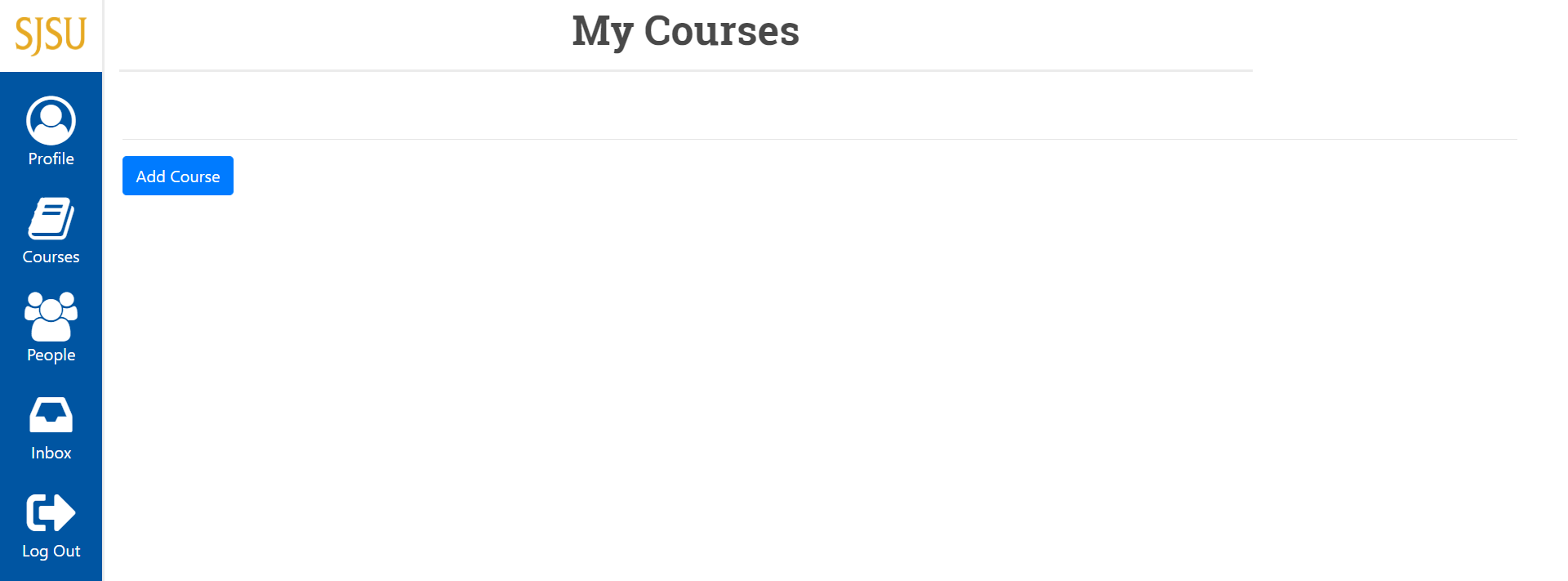


**Results**

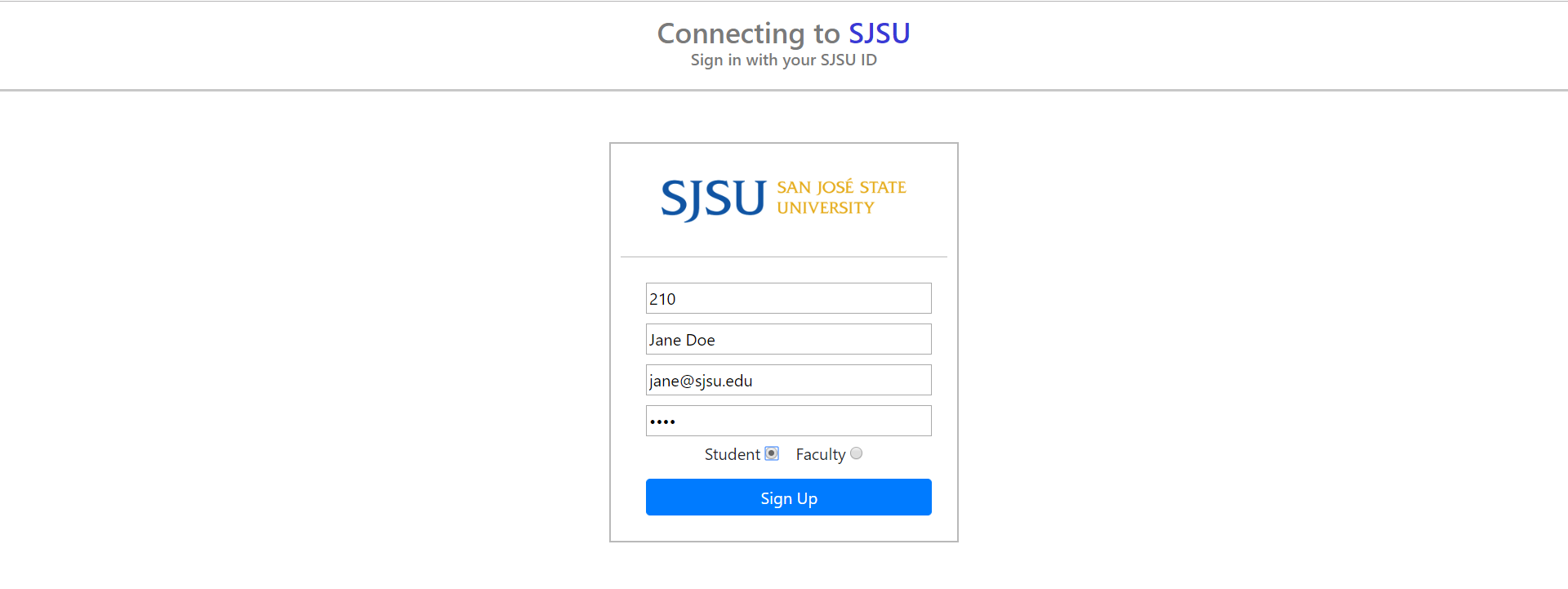
Faculty Signup and Login

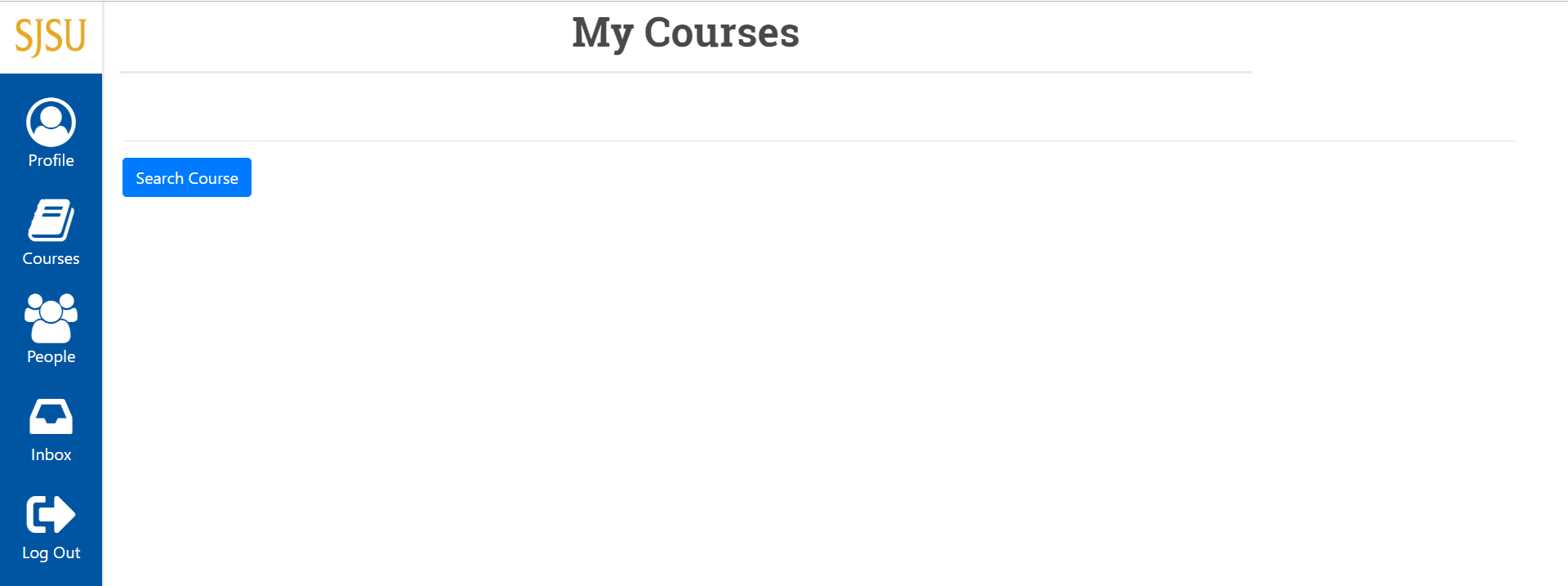


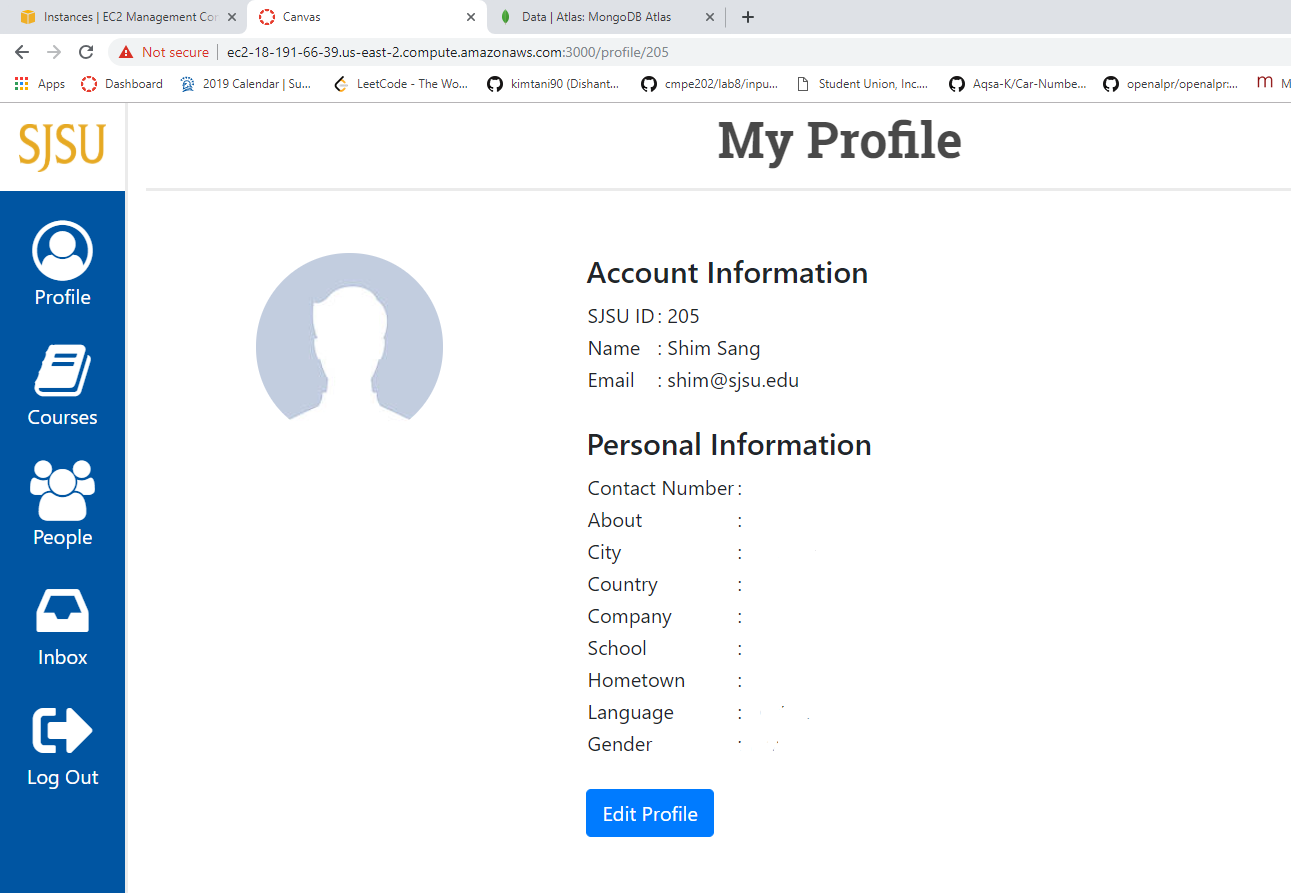




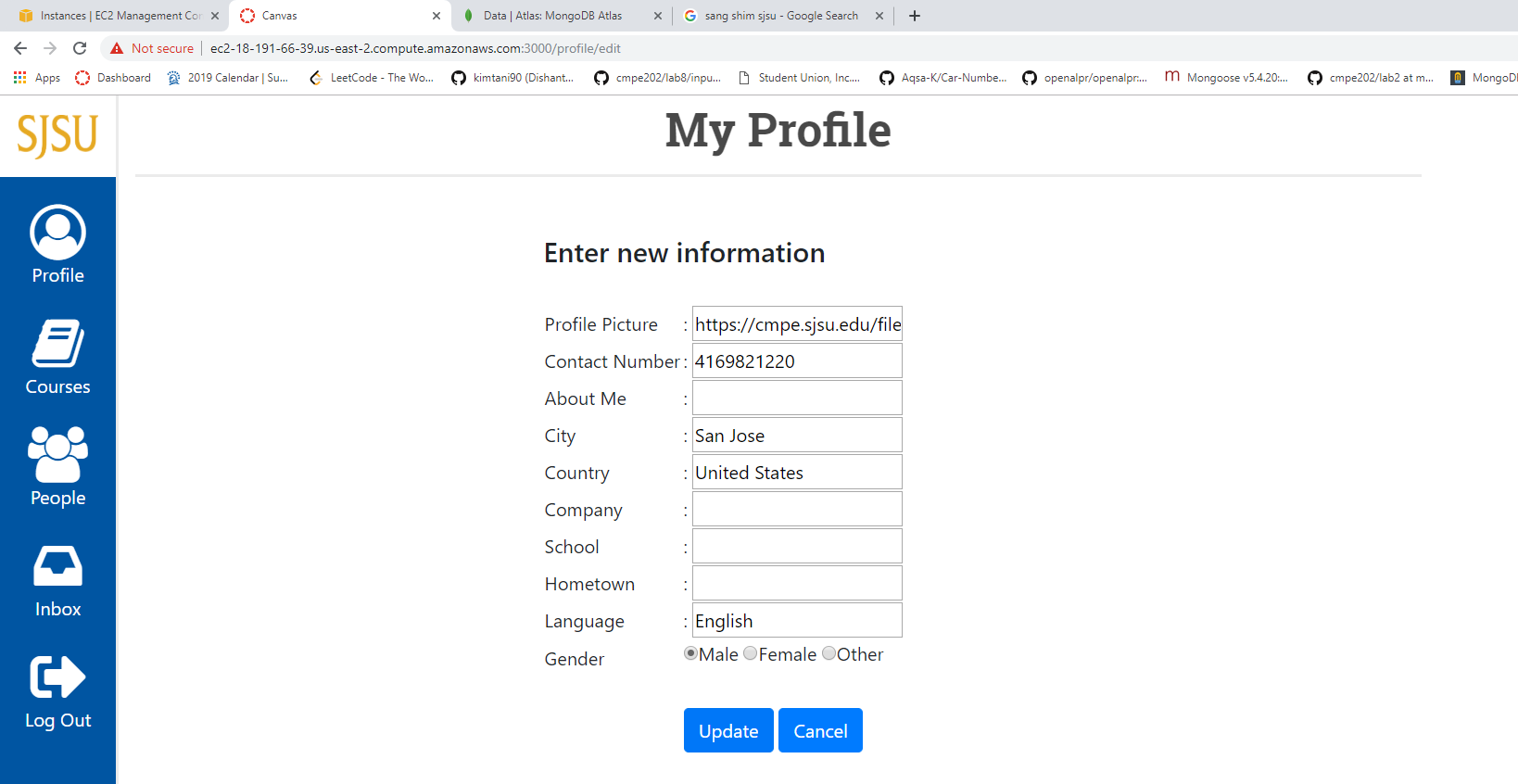
Student Signup and Login

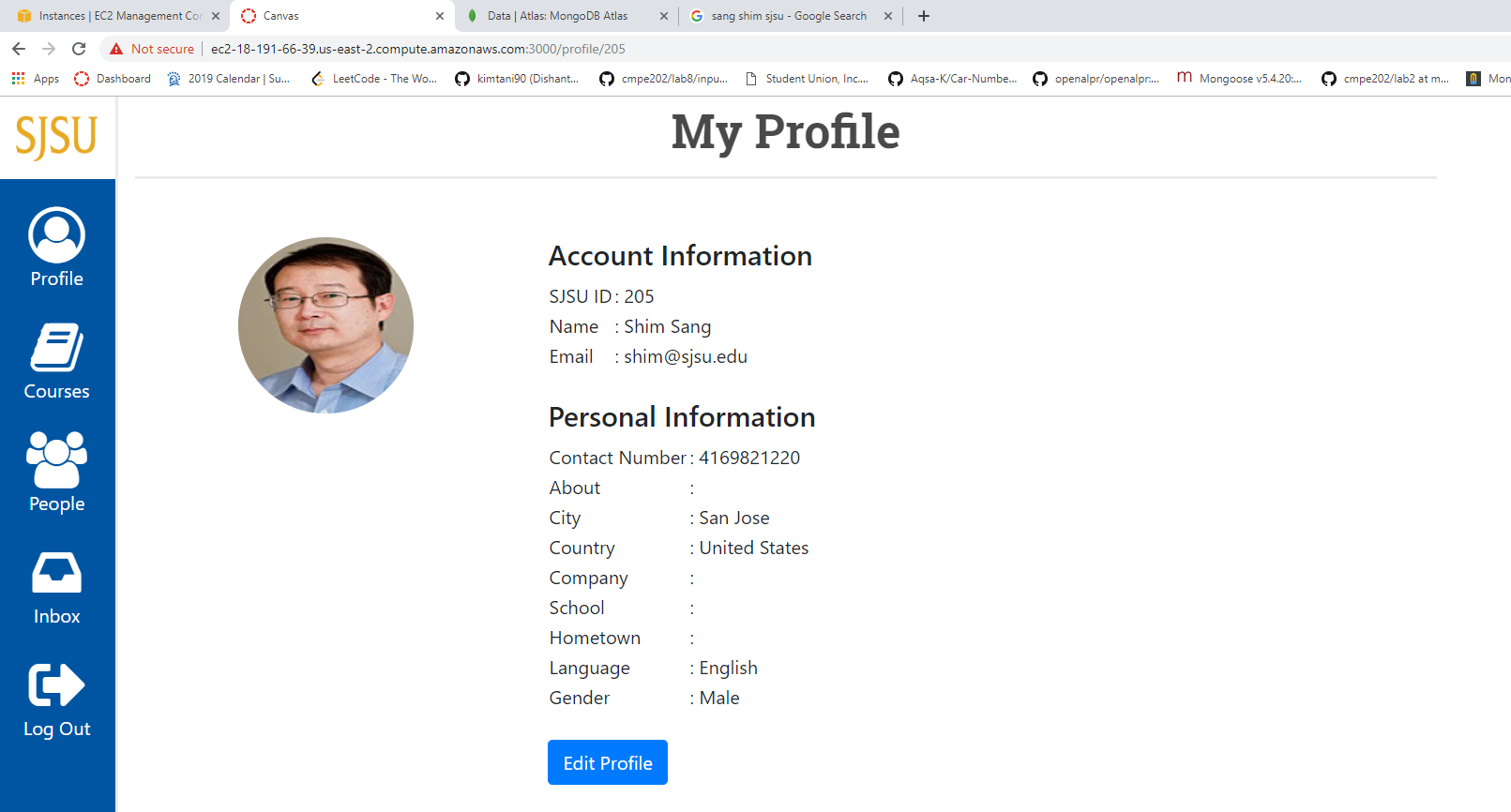




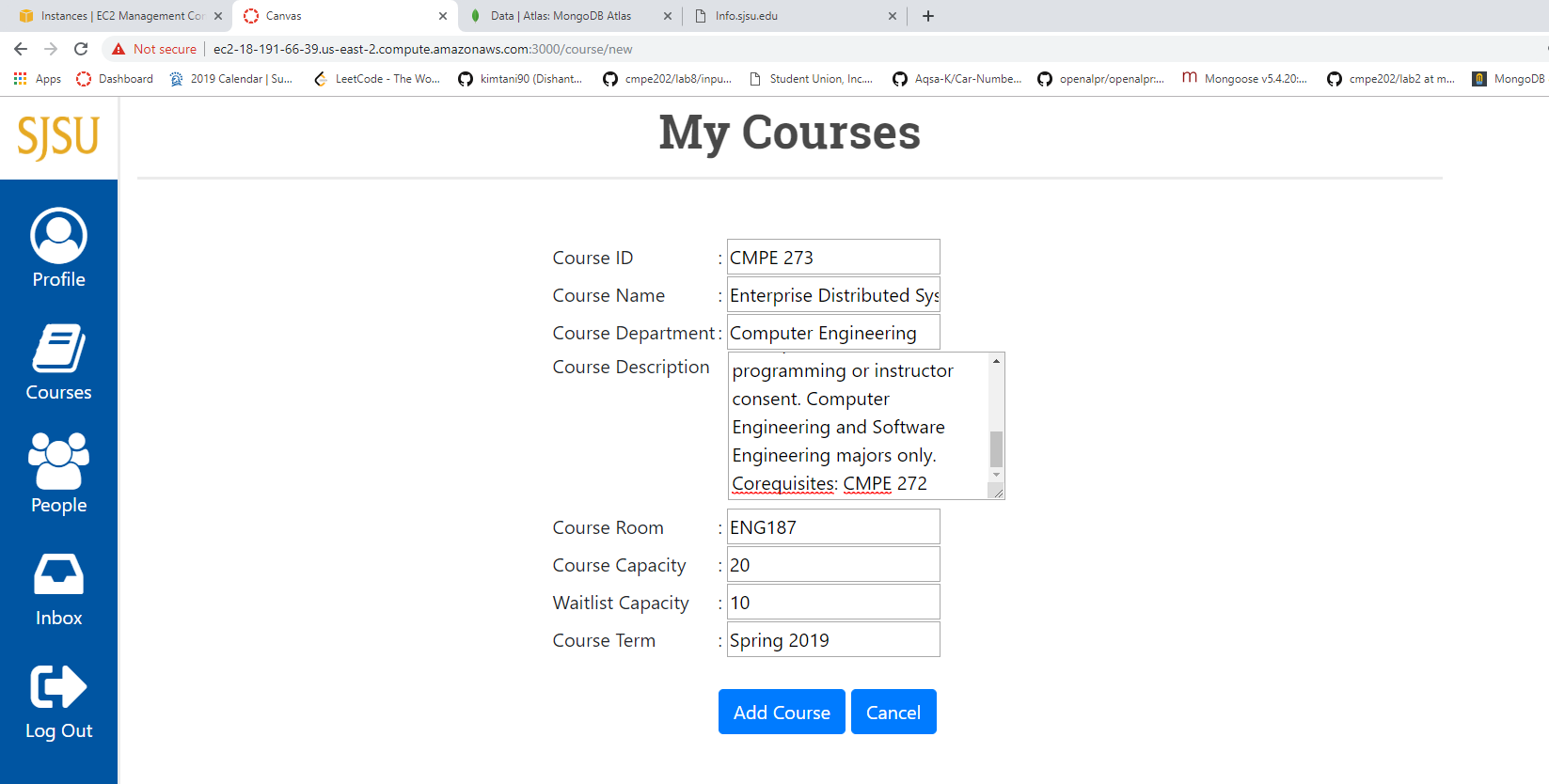
User Profile

Edit Profile

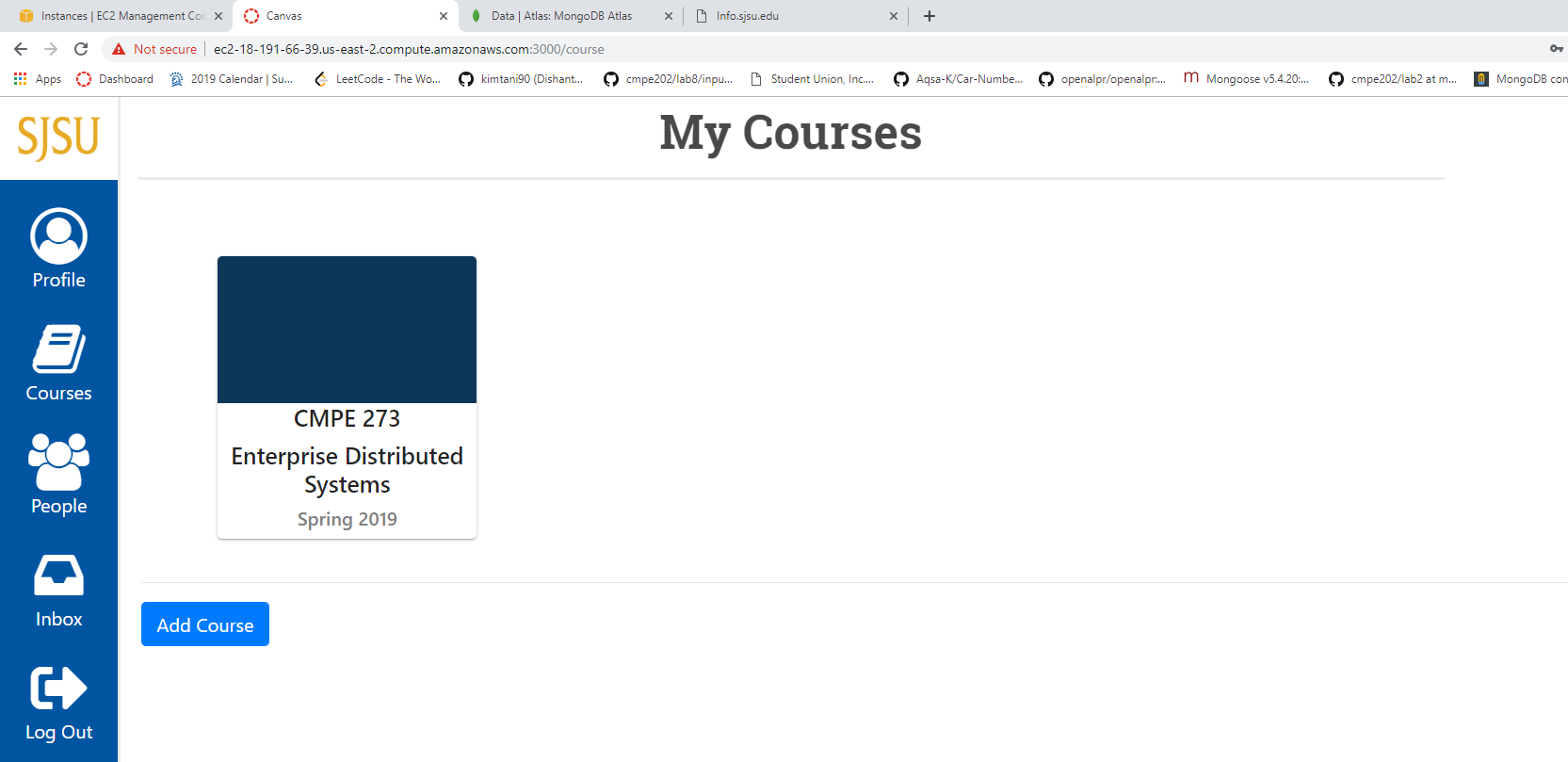




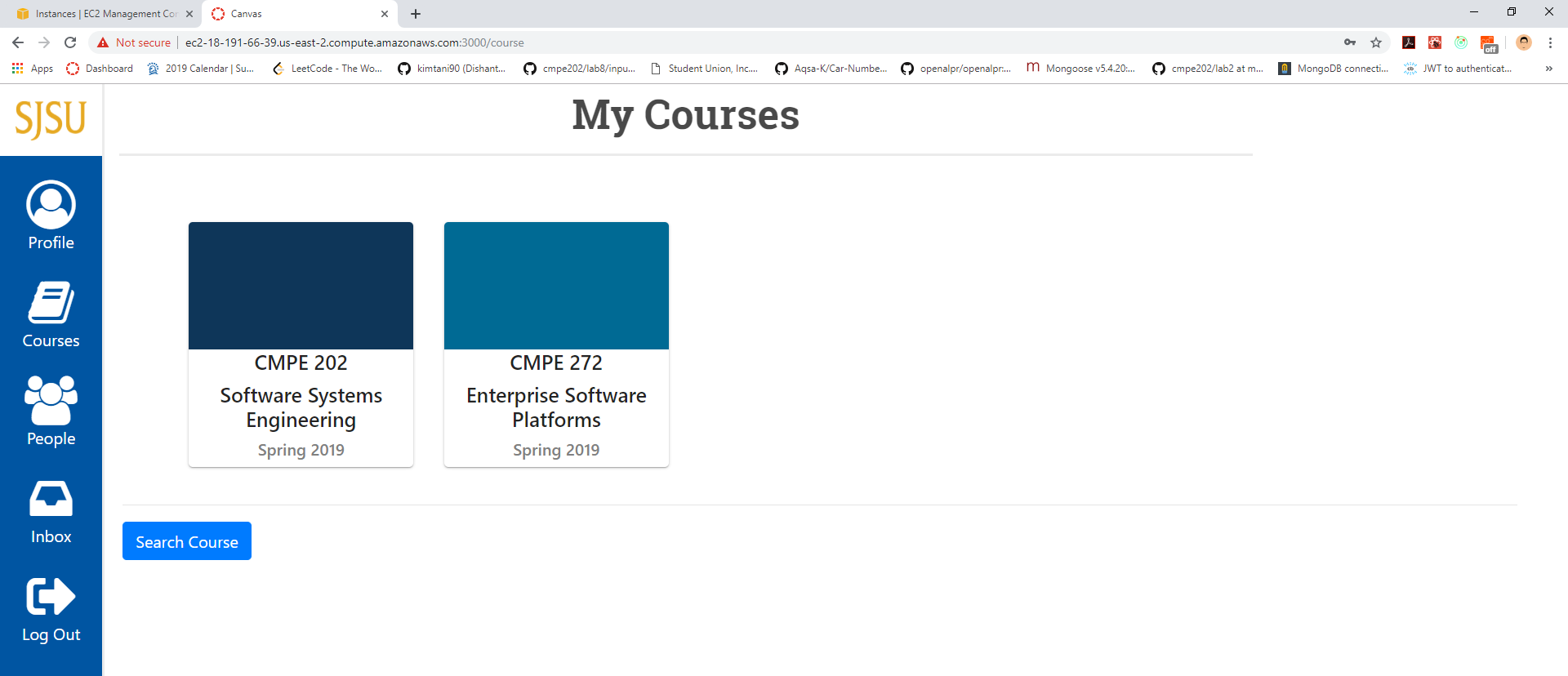
Course creation



Faculty Courses View



Student Courses View



**Answers**

1. GraphQL handles only serializable data, there is no way to upload files directly as part of mutations. But there are several ways to transfer multipart data without using external library, such as:

* Base64 Encoding: Image can be encoded in base64 format and send as string. This method has several disadvantages like
  + Encoded image being larger than original file.
  + Encoding and decoding operation being expensive operation.
* Separating upload requests: To have two servers, one for storing files and second to store image storage URL passed as mutation. It adds a layer of complexity to handle multiple file uploads. Also, this method is not asynchronous, and it is complex to manage the upload server.

1. To enable multipart data transfer using an open source library, I would suggest using ‘ graphql-multipart-request-spec ‘. It allows to nest files anywhere within GraphQL mutations as follows:

{  
 query: `  
 mutation($file: Upload!) {  
 uploadFile(file: $file) {  
 id  
 }  
 }  
 `,  
 variables: {  
 file: File // somefile.jpg  
 }  
}

This method is quite simple because adding a file is as simple as adding another mutation parameter. We implement this, we run a client and a server specification. The client specification defines how to map any file objects in a mutation into a key that locates where they are in a multipart request whereas the server specification defines how to parse that map and make the files re-accessible based on the key provided in the map.

Hence, in apollo-client, a mutation runs as follows:

this.props.mutate ( { variables: { file: yourFile } } )