**CMPE 275**

**Project Report**

**SurveyApe**

**Team Members(Group-14)**

**Anjana Pradeep (011950809)**

**Ujjval Soni (012475645)**

**Divya Thazhathethil (012458875)**

**Rajeshri Sonwane (012406134)**

1. **Introduction**

SurveyApe is a web-based application that allows Surveyor to compose, distributed and analyze stats of surveys. Surveyor can create three kinds of surveys and each survey can have various types of questions (text, checkbox, radio, dropdown, images, star rating). Surveyee can participate in a survey using invited link(general and closed survey) or by using the application(open survey). Surveyor can add and invite participants to give the survey. The participants will receive the link through email.

Implemented this application using Spring boot, MySQL and React. It is created using MVC framework, where Model is the MySQL database, View are the react pages and Controller is the Spring boot back end application. We used various other concepts like Object Relational Mapping, Dependency Injection and transcations.

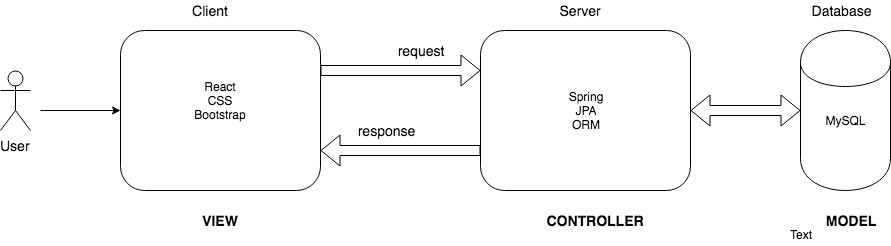
1. **Bonus Feature**

Export and import survey questions:

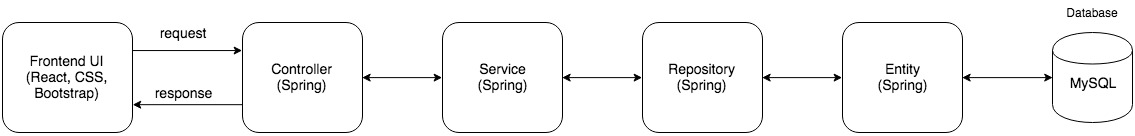
The surveyor can export any survey that he has created and download it as a text file. The text file has the JSON of the questions and its option.

The surveyor can then again import this file while creating a new survey to add the questions and options.

1. **High Level & Component Level Design**



***Figure 1: High Level Design***



***Figure 2: Component Level Design***

1. **Technology Choices**

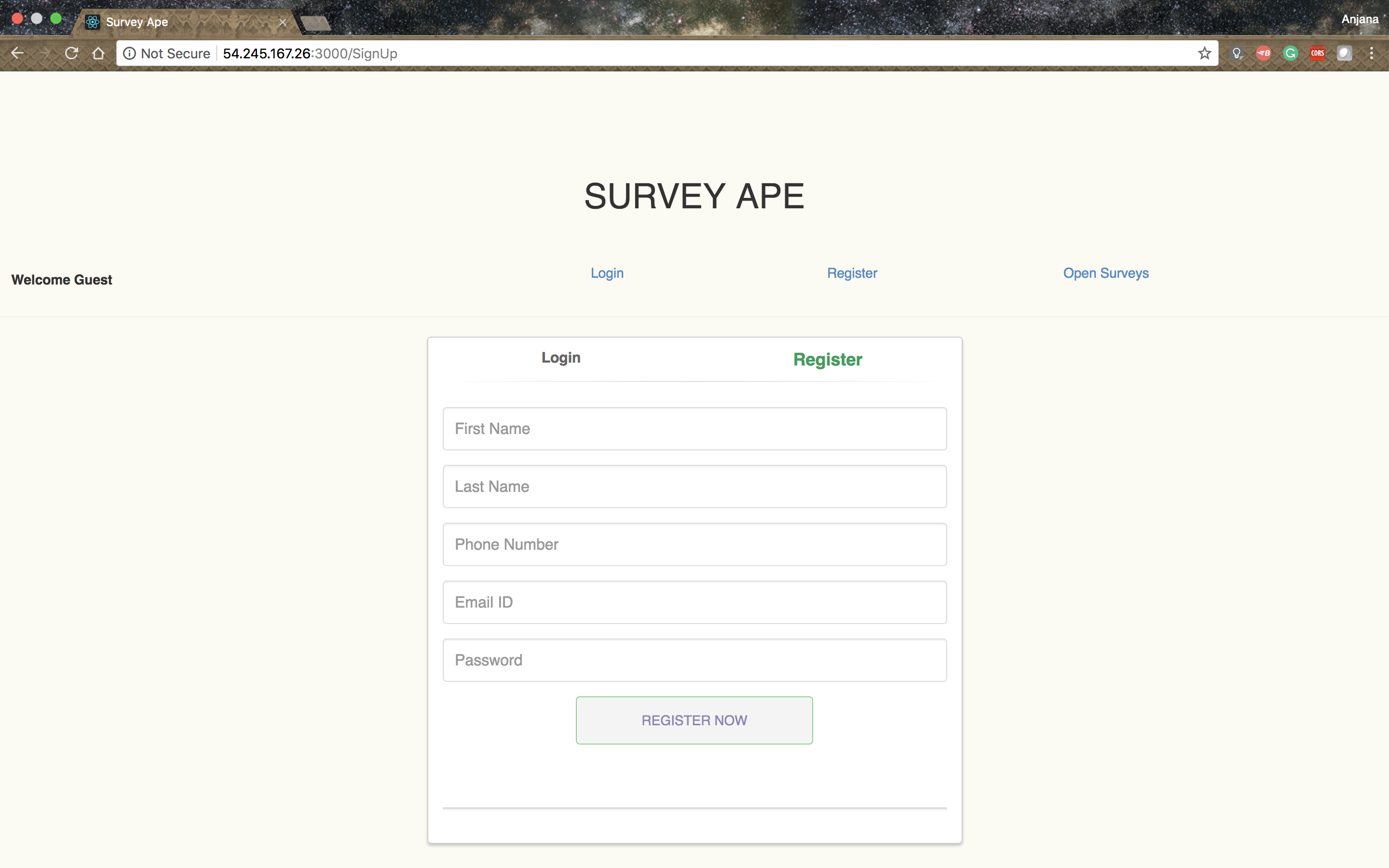
We used the following technologies to implement our project:

* Spring Boot for the backend
* MySQL as the database
* React to create interactive UI
* CSS/Bootstrap for styling and structing of web pages

1. **Description of Features**

**a. Register**

Allows the user to create an account in SurveyApe



***Figure 3: Register/Create an account***

**b. Signin**

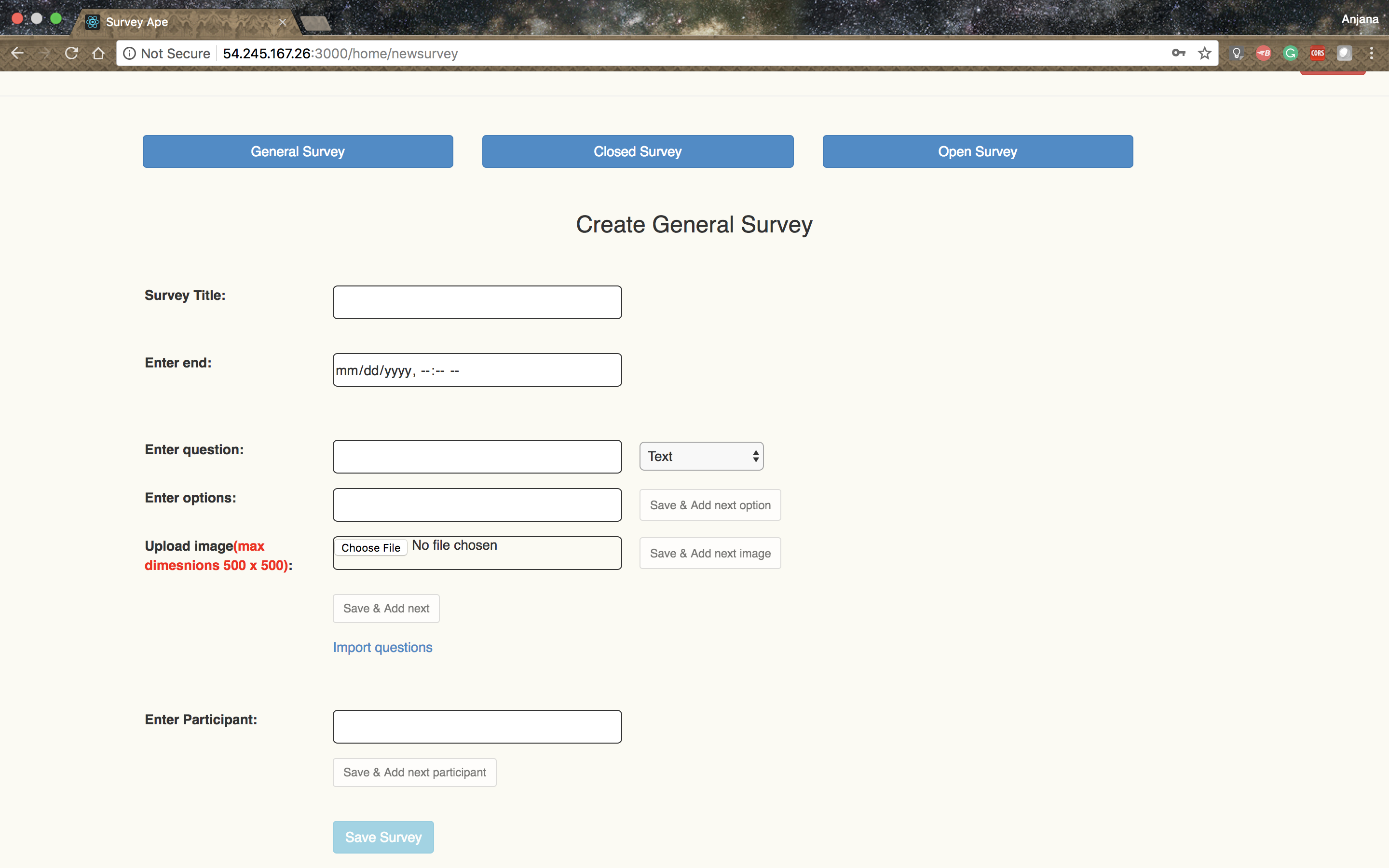
Allows user to login to the application



***Figure 4: Signin***

**c.** **Create General Survey**

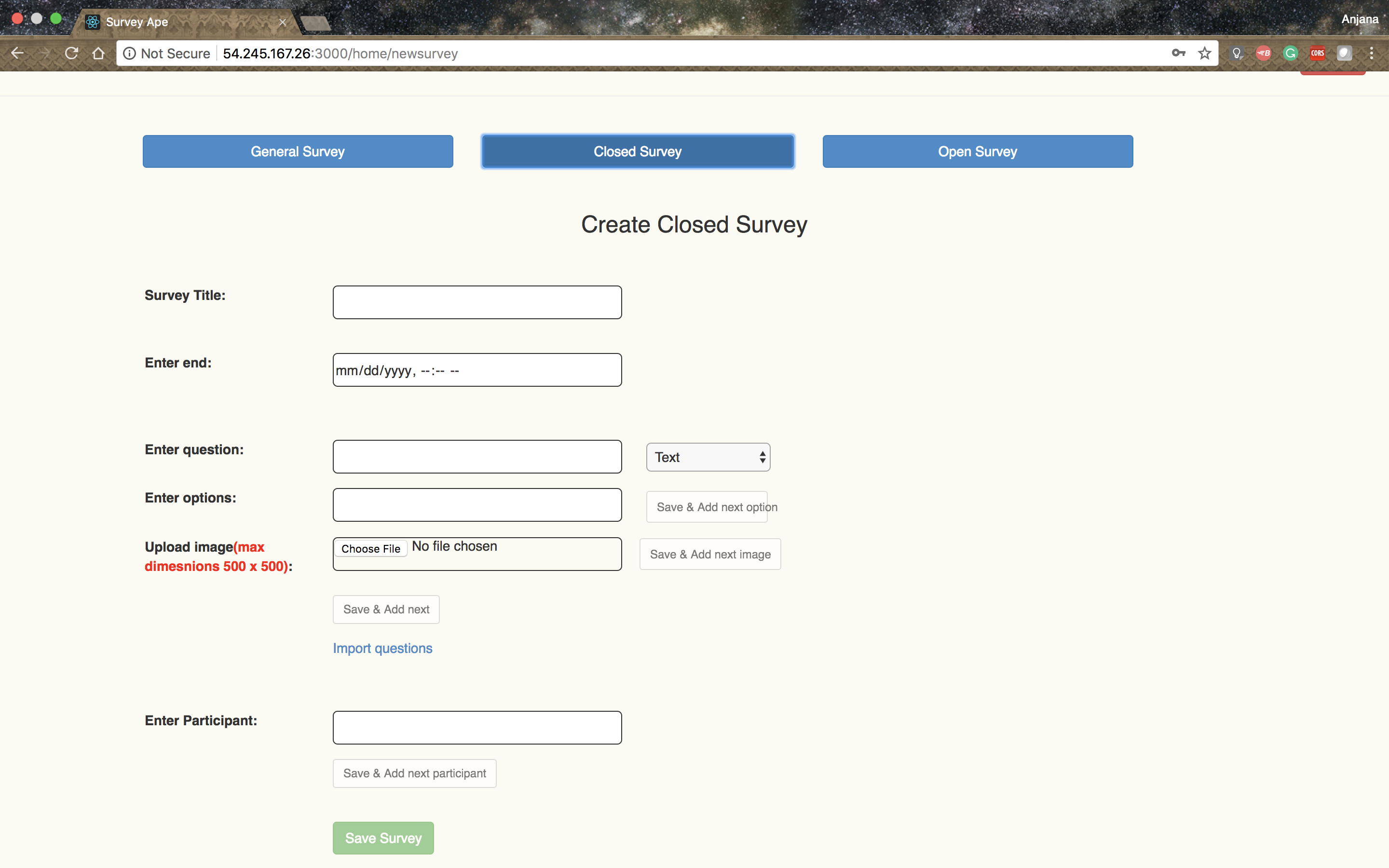
For general survey, user can add title, end time, question and its options, participants.



***Figure 5: Create General Survey***

**d. Create Closed Survey**

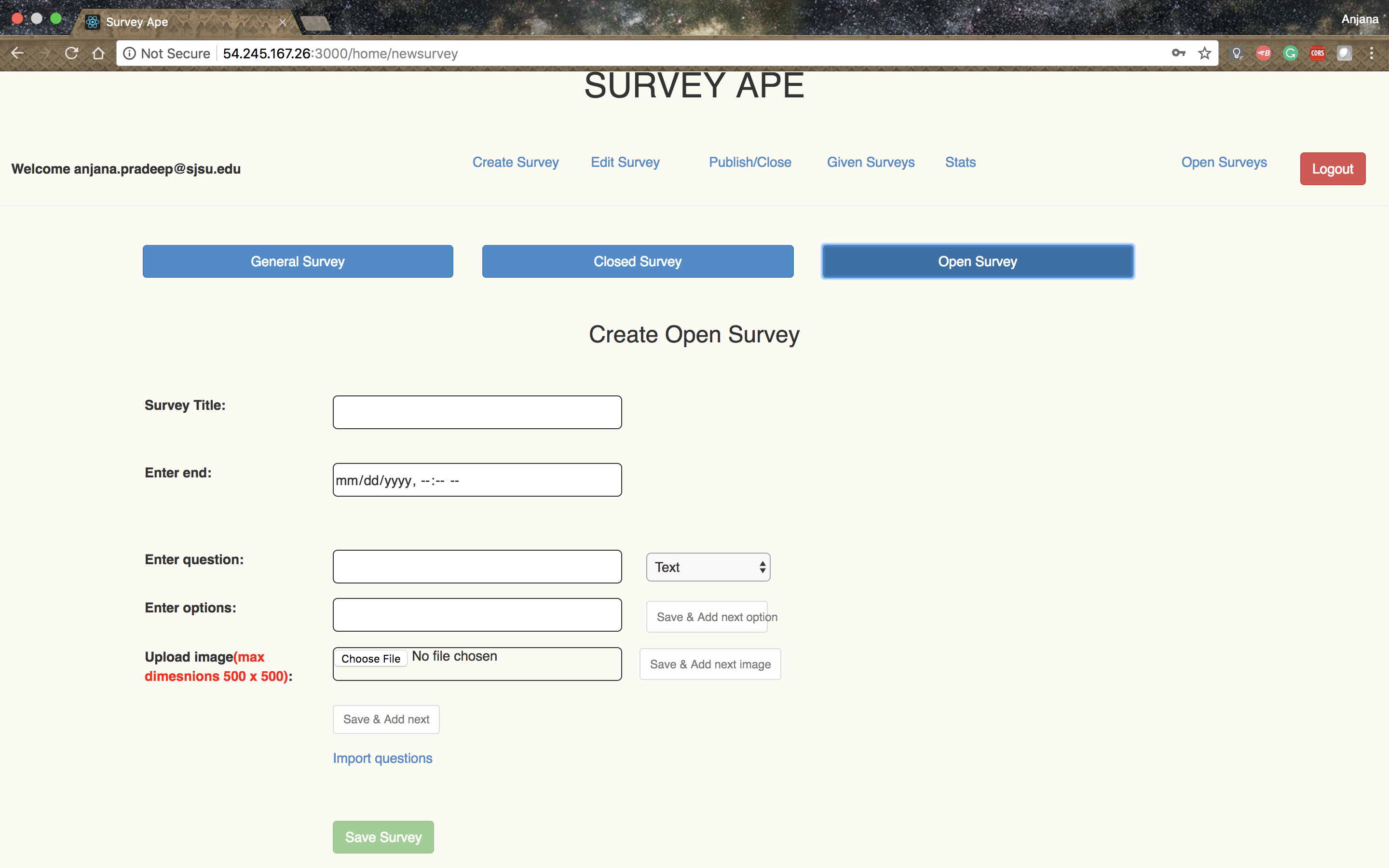
For closed survey, user can add title, end time, question and its options, and participants.



***Figure 6: Create Closed Survey***

**e. Create Open Survey**

For open survey, user can add title, end time, question and its options



***Figure 7: Create Open Survey***

**f.** **Edit Survey**

Allows user to edit previously created surveys and can add more question and its options, and new participants.

**g. Survey Status**

Surveyor can change the status of surveys created by him.

Publish: to publish the survey to make it available to the participants.

Unpublish: to unpublish the survey which makes it unavailable to the participants. It can be published again.

Close: to permanently closes the surveys and it cannot be published again

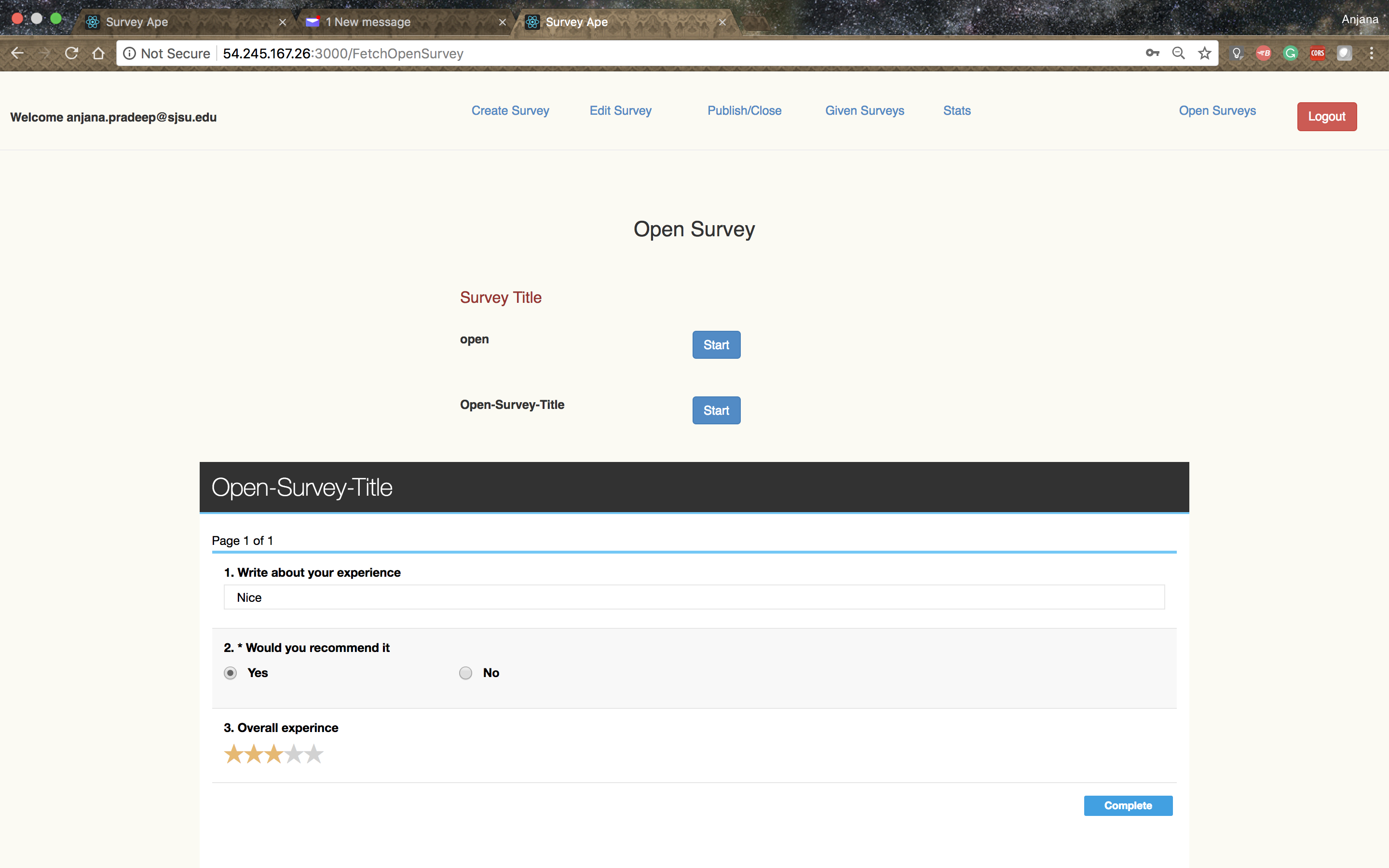
**h. Give Survey**

General survey is given by a common link which is sent by mail

Closed survey is given by a unique link which is sent by mail

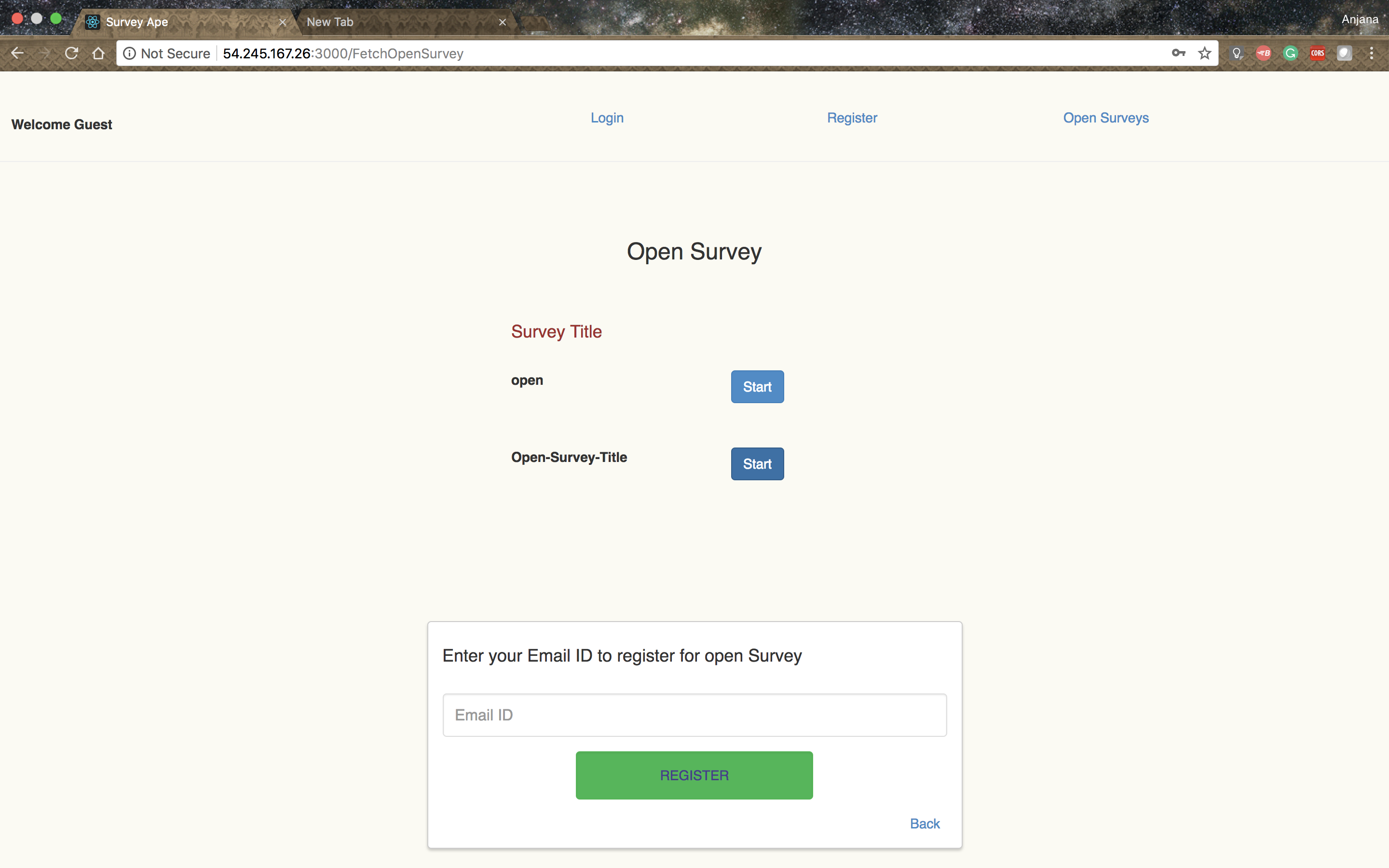
Open survey can be either given by logging in to the application (for users who have an account) or by requesting a link (for users with no account in SurveyApe).

For logged-in users



***Figure 7: Open Survey - for logged in user***

For guest users



***Figure 7: Open Survey – request for link***

****

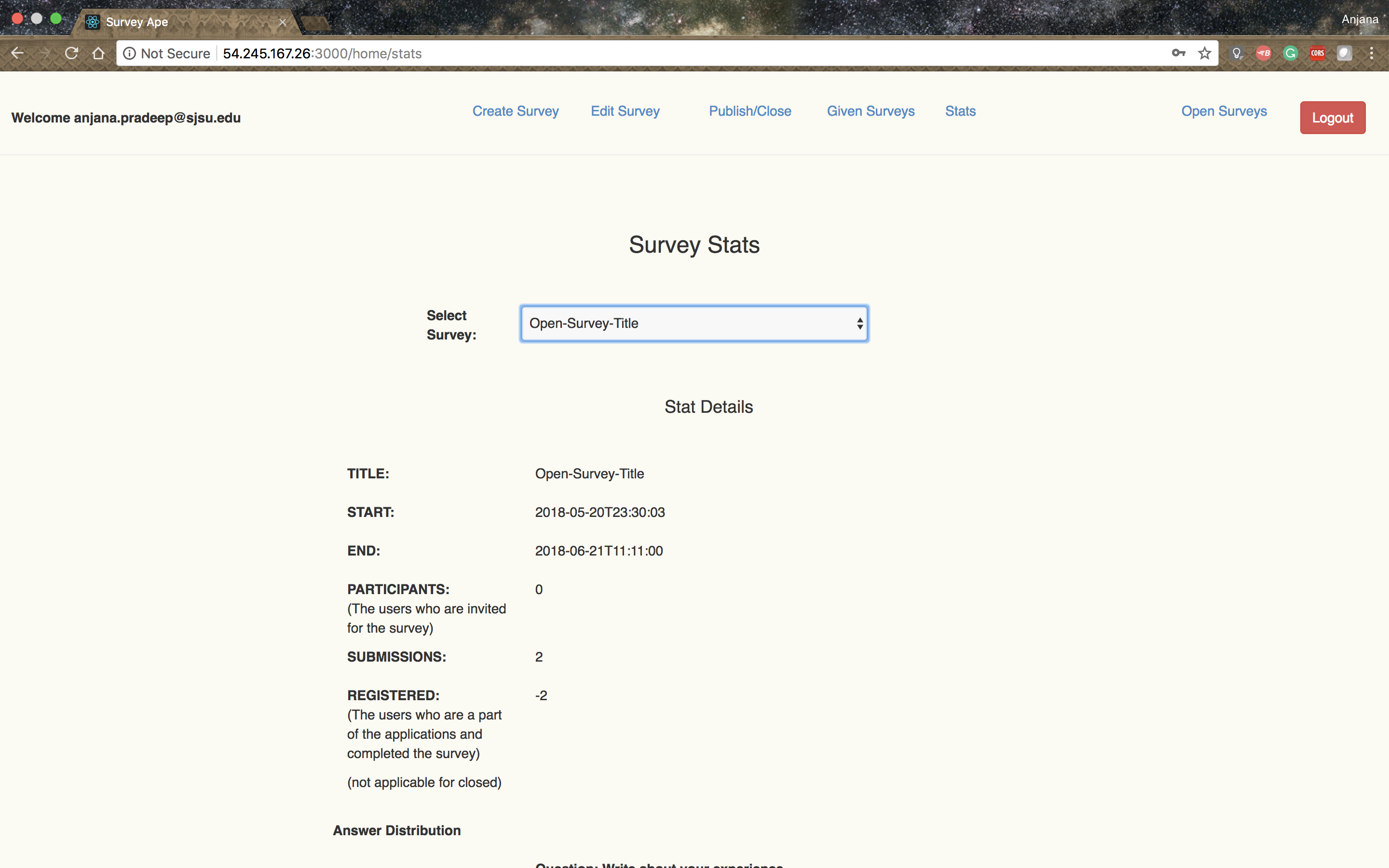
***Figure 7: Open Survey - for guest user***

**Given surveys**

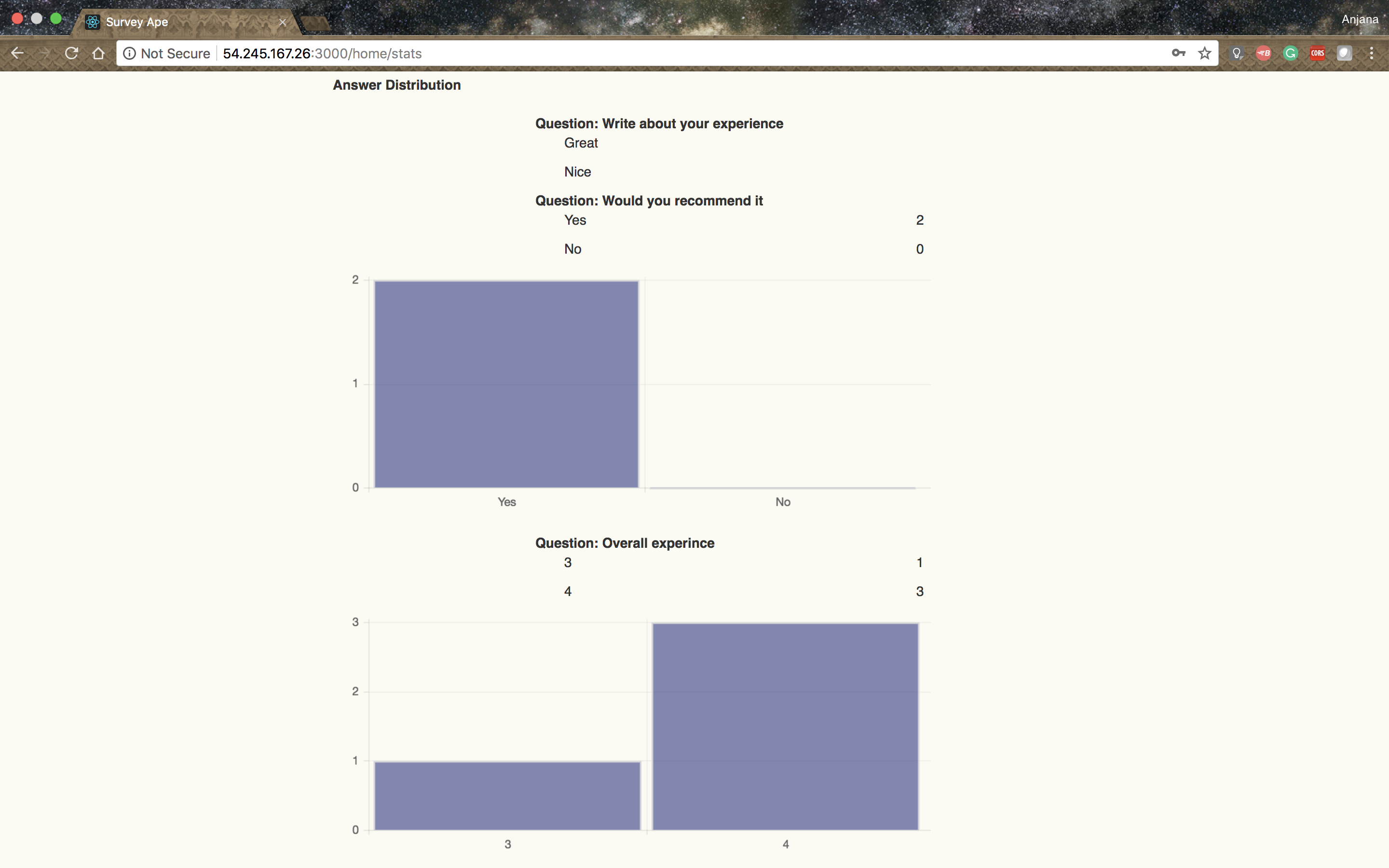
Surveyee can view the list of surveys given by him. If the survey is submitted, he can view the survey in read-only mode. If the survey if not yet submitted, he can continue with the survey.

**Survey Stats**

The surveyor can see the stats for all the survey created by him. Using the dropdown he can pick one survey and see the stats.



***Figure 7: Stats***

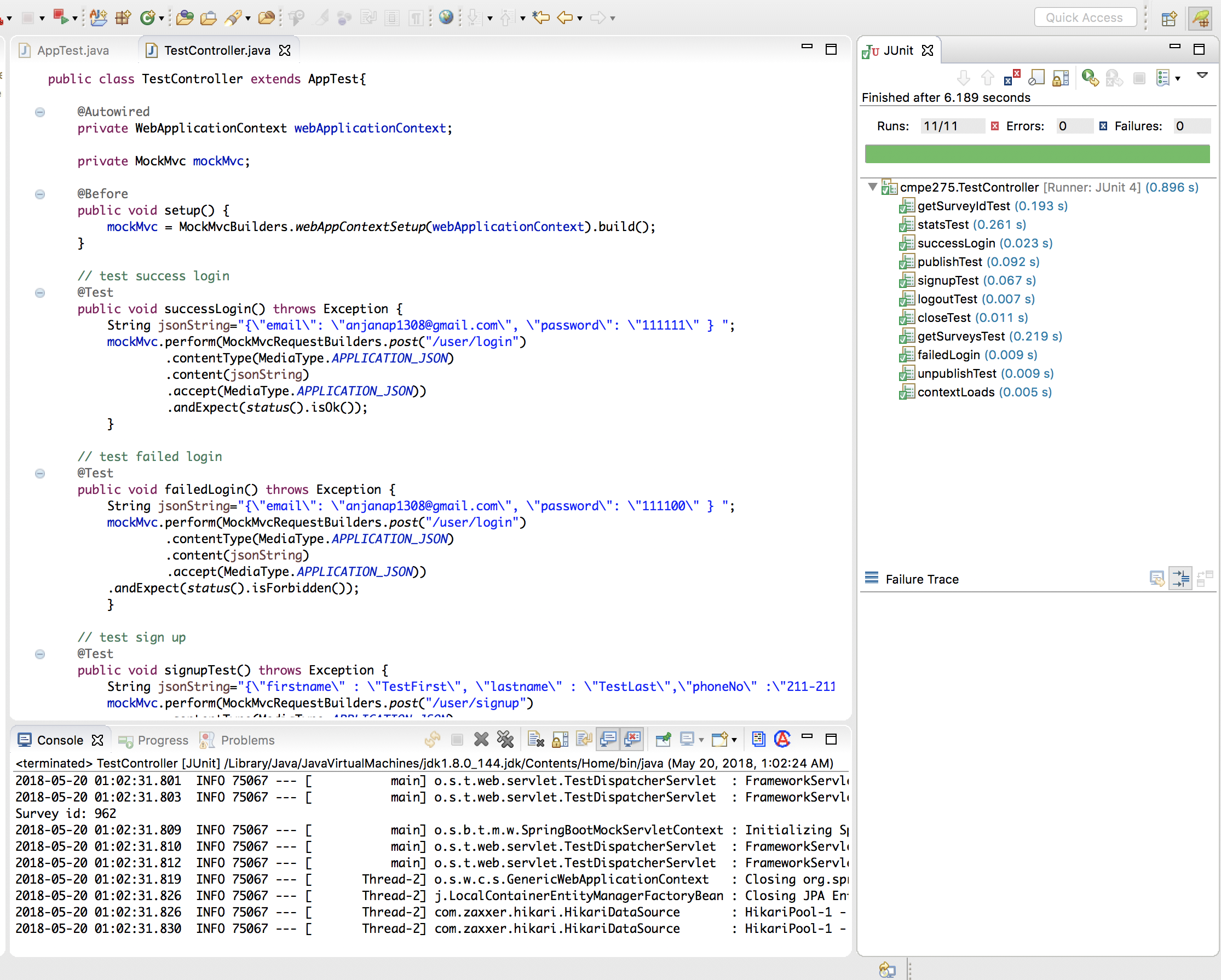


***Figure 7: Stats***

1. **Testing Plan**

Performed unit testing of the application using Mockito. Mockito is an opensource testing framework for Java.

Following are the results of the unit test cases:



***Figure 7: Unit test cases***

1. **Lessons Learned & Future Work**

Working on this project we got to learn and implement everything we learned many important aspects of working on a team project. We made a few mistakes along the way like hastily reading the project requirements and misunderstanding a few of them and thus in the end we had to change a lot of things just to fit those requirements. We also learned that while keeping all the code in a single file and not separating the code according to their functions created many errors, as when everyone was working on the same file there were many conflicts at the end of the day. We learned that separating files and creating a modular structure was very important. We learned about various technologies and frameworks such as spring, spring boot, dependency injection, transactions and various others. This project was challenging and fun at the same time.

For the future, we have decided we will be working on this project further and we will try to make the UI/UX better. We have also decided to check for more exceptional conditions where our application could fail and make sure that no matter how the user tries to interact with the system, he still won’t be able to crash the system. Moreover, add more functionalities, if time permits and make the application not just a prototype but an industry level application. But overall it was a great experience working in the project.