CSE 546 — Project 2 Report

Title - ASU Short Term Accommodator

Akash Kadam - 1215180915
Aditi Panigrahi - 1215180863
Piyush Waykole - 1215202716

1. Problem statement

Title: ASU Short term accommodation

Motivation: Incoming international students turn to ASU's student organizations to look for temporary accommodations during the initial days of their 1st semester until they get the possession of their rented house.

Currently, all communication takes place without any organized platform. We intend to create an application that provides students with such a platform that will allow them to look at the available options and then let the students and the volunteer hosts settle on an agreement mutually.

Functionalities of our application:

1. An organized platform for finding temporary accommodation.
2. The host can register on the application to list the specifics of the accommodation(voluntarily) they can offer.
3. Students can enter their requirements and the app will show the Top-k matches from the available accommodations.
4. Students will also be notified when a new accommodation is made available on the application.
5. Bulk Accommodation Volunteering - If any host or organization can offer multiple accommodations then it will be cumbersome for them to enter information for each accommodation, thus we provide a feature to add multiple accommodations in one go.

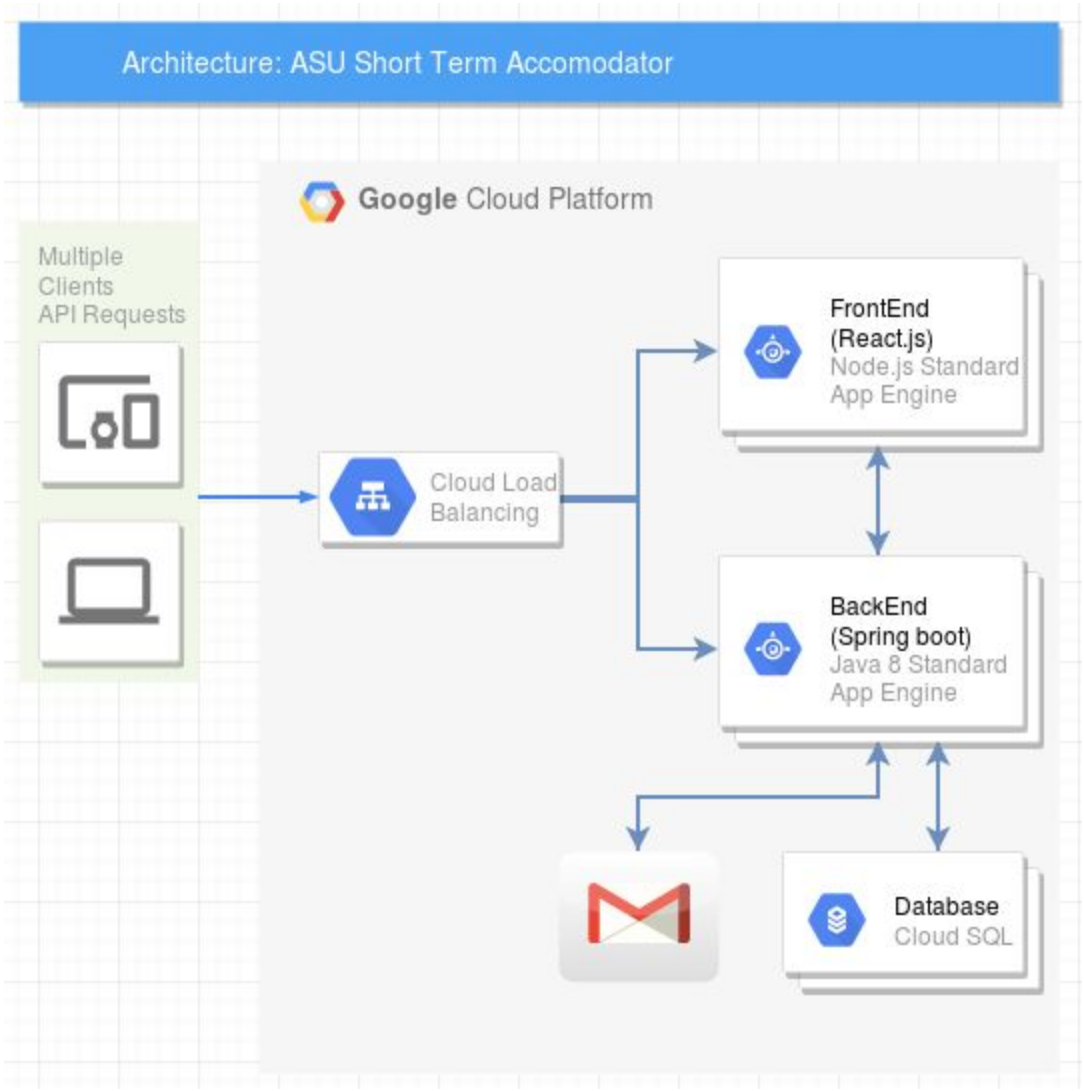
Future Goals:

It can be extended at times of disaster relief to provide shelter for those who need temporary accommodation due to unforeseen circumstances and the good Samaritans volunteer to provide shelter.

Technologies: Google App Engine, Google Cloud SQL, Google Gmail API, and any other required cloud technology.

2. Design and implementation

2.1 Architecture



The application consists of two major components - FrontEnd & BackEnd :

FrontEnd component generates and shows GUI to the users while the BackEnd component is responsible for computations, business logic and database interactions. A typical request from user would be processed like this:

First, the user sends an HTTP request to the application, it is first served by FrontEnd where user can see web pages created using react.js, then whenever user performs any action on web pages (say, login action), request is then served by BackEnd which connects to Database and other external API's to fulfill the request and returns the result to the user.

Both the component have exposed web API's to perform necessary actions.

1. FrontEnd:

    a. Design - The frontend is created using React.js library. The frontend is a single page application which renders different react component dynamically based on events generated by the user. The frontEnd consists of Registration Component, Login Component, Accommodation Component, Student Listing Component.
    b. GAE Environment and Deployment - Since FrontEnd is developed and designed using React.js framework, it is deployed onto the NodeJs environment. Google App Engine's NodeJs 10, Standard environment which is fully managed by GAE is used to deploy the service onto GCloud.

        Command used: gcloud app deploy (run from root directory).

2. BackEnd:

    a. Design - The backend is created using spring boot initializer and the packaging is war. The backend contains multiple controllers that have isolated responsibilities. The Entities are created keeping in mind the User Information and Accommodation information that is required for matching and are persisted in the Cloud SQL database.
    b. GAE Environment and Deployment - BackEnd is developed using Spring Boot framework, it is deployed onto the Java 8 environment. Google App Engine's Java 8, Standard environment which is fully managed by GAE is used to deploy the service onto GCloud.

        Command used: mvn appengine:deploy (run from maven root directory).

Google Cloud SQL is used to persist the data as we need the available accommodation information in order to find matching listings for a student.

2.2 Autoscaling

Google App Engine's Standard environments are used which are fully managed by Google Cloud and thus it takes care of autoscaling for both the components. We only need to configure *automatic_scaling* inside app.yaml or appengine-web.xml and Google performs scaling-up and scaling-down of the applications between zero and planet-scale.

Some of the parameters of autoscaling provided by Google AppEngine that we have used in app.yaml and appengine-web.xml are: *target-cpu-utilization, target-throughput-utilization, max-concurrent-requests.*

3. Testing and evaluation

a. Manual testing:

    We created users by going to the web application and performing the actions (Clicking buttons) on GUI.

    To evaluate, We double-checked this with the entries in the databases and of course, GUI had correct data shown.

b. Automation testing:

    We created an automated test plan using Apache JMeter for each service endpoint and run this test plan for 100,200,300,500 and 1000 times for each web service.

To Evaluate Results, We checked the response of each request using a script inside Apache JMeter and checked the instance count on Google App Engine's Dashboard. We could see the HTTP response OK (200) for each request, Database entry in Cloud SQL DB table. We also could verify scaling out/scaling by checking instance count against each service in GAE's Dashboard. The instance count was increasing when requests were more and decreased accordingly.

4. Code

Our Project consists of two parts FrontEnd and BackEnd.For Every functionality, there is an API call from frontend to backend to get the proper information needed to display to the user. Backend get the request from the frontend with specific request parameters,based on those parameters backend

the frontEnd basically consists of Registration Page, Login Page, Host Accommodation Page,Student Listing Page, Student Accommodation listing Page.

Registration Page: - when any new user(either student or host) comes to our website, she/he has to register first.If the user is a student she/he must have to register with personal details and desired accommodation details. If the user is a host then she/he must have to register with personal details and available accommodation details.Once the user submits the details then frontend calls the backend's "register" API which verifies the minimum required details and creates the user in the User table of Cloud SQL database.After successful creation of a user, backend returns the success response to frontend which then redirects the user to the Login page for login.

Login Page:- As the name suggests the Login Page is for a user to log into the website using username and password. There is an option for a user to log in as a student or host.When a user login as a student frontend calls backend's "studentLogin" API with the parameters and user type as a student.Backend API returns the list of accommodations which are suitable for that user based on the details that particular user has provided while registration along with the status of accommodation request for each accommodation.On the successful response from backend, frontend renders **Student Accommodation listing Page.**When a user login as a host frontend calls backend's "hostLogin" API with the parameters and user type as a host.Backend API returns the list of accommodation that particular host has posted.On the successful response from backend, frontend renders **Host Accommodation Page** showing these accommodation listing.

Student Accommodation listing Page:- This page shows a listing of accommodations to the user to choose from.There is an option to request an accommodation for every accommodation.When the student finds any suitable accommodation she/he can request the host for approval which calls the backend's "approveAccomodationFromStudent" which updates the status of accommodation for that particular user and notifies the host using Google Gmail API that someone has requested to stay at the accommodation listed by her/him.On the successful response from backend, frontend redirects the user to Login Page.

Host Profile Page:- This page shows the listing of all the accommodations posted by the particular host.Every listing has an option to see the list of users who have requested to stay at that location, on selecting that option for that accommodation frontend calls backend's "getStudentRequestsAgainstAccomodation" API which returns that list.On successful response, frontend renders **Student Listing Page** showing the list of Users.

Directions to run the program:

1. FrontEnd:

   You must need to have node.js installed on your computer, if not then please install from https://nodejs.org/en/.

   Go to the root directory (where app.yaml is), open a command prompt and run

   i. *npm install* - This command builds the project

   ii. *npm run build* - This command creates production ready files

   iii. *gcloud app deploy* - This command deploys on app engine.

2. BackEnd:

   You must need to have JDK installed on your computer.

   Go to the root directory (where pom.xml is), open a command prompt and run

   i. *mvn clean package* - This command builds the project.

   ii. *mvn appengine:deploy* - This command deploys on app engine