CSE 546 - Project 2 Report

Darshan Dagly (ASU ID: 1215180174)
Smruti Berad (ASU ID: 1214907915)
Viken Shaumitra Parikh (ASU ID: 1215126783)

1. Problem Statement

How many times have you seen a lot of left-over food at restaurants at closing time? Alternatively, how many times have you heard of food kitchens, who provide food to the homeless, in need of help and support? One one hand, everyday 150,000 tons of food is thrown away and wasted in the US due to lack of consumption. This includes products that are near their expiry date and leftover food from restaurants, events and households. On the other hand, an approximate 50 million Americans go hungry everyday or struggle to put food on the table on a daily basis. Moreover, 66% homeless kitchens and pantries have to turn away people on a daily basis due to shortage of food to offer. These statistics clearly show a gap between the two ends of the food consumption spectrum that needs to be addressed.

What if there was a platform which could help these two types of places connect so as to curb food wastage?

Our application provides a portal which enables a connection between the people who are willing to donate food and the people who are in need of the food. The application provides donors with a functionality of adding items which they are willing to donate along with their own details to facilitate pickups. These donations are visible to the kitchens, who if interested, can get in touch with the donors. These donors can also function as volunteers to help transport food between the different donors and kitchens. The kitchens can look for donors around their area to see donations around them for easier accessibility to food. Concurrently, the kitchens can also make requests of items they are in dire need of with their own details. The donors can then view these requests and if in possession of these items or willingness to transport them can contact the kitchens. As in the case of the kitchens, the donors also can look for kitchens around them in need of donations to help with accessibility. The kitchens can also view volunteers and their details and search them by their location to help transport food to them.

The application is made distinct from the current solutions as it is a secure cross-platform application which allows different parties to come together to prevent food wastage which would have a major social impact and help people.

The application is targeted towards two types of audiences:

1. Places like restaurants, shops or even homes who have a food surplus or are willing to give away food to help those in need.
2. Places like food kitchens or pantries which are open to people who are in need of food.

The application was currently designed on a smaller scale with just a few basic functionalities in mind for this subject. It can be improved in various ways in the future. The application can be made more interactive by showing the locations of the kitchens/donors around the user's location on a map. The application can also provide real time location tracking on the pick up and drop off services making the user interactions smoother. Also, the application currently supports a Business to Business model. It can be expanded further to support a Business to Customer model by helping individuals to find food in the areas around them.

2. Design and Implementation

2.1 Architecture

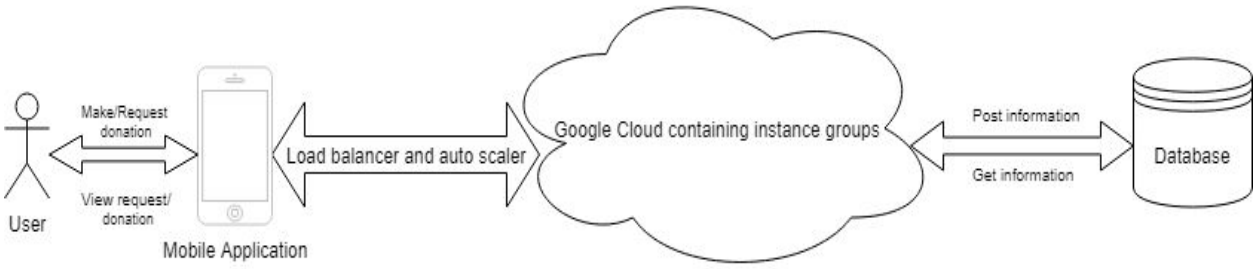

Fig 1. Architecture of the application

The system has the following main components:

1. Frontend -

   The system is divided by functionality into two main users - donors and kitchens. The application has two main activities for the donors:

   - Fill out a form with their details, the details of the items they are donating and the pick up details such that they are clear to the kitchen who would be in need of the items.

- View requests made by the kitchens to see the their requirements. On clicking on a request, the donor can view further details about the food kitchen and the food they need. The donors can search for kitchens in their area as connecting with people around them is far easier than a general overview of all the kitchens.
- Add in a request to function as a volunteer by entering their details, location and the time and date they are available to help with the transportation.

Similarly, the application has two main activities for the kitchens:

- Fill out a form with their details, the details of the items they are in need of and the drop off details such that they are clear to the donor who would be donating the items.
- View donations made by the donors to see what they are offering. On clicking on a request, the kitchen can view further details of the food donor and of the food they can provide. The kitchens can search for donors in their area as connecting with people around them is far easier than a general overview of all the donors.
- View donors acting as volunteers. On clicking on a request, the kitchen can view further details of the volunteer and the time and place they are available at. The kitchens can search for volunteers in their area and connect with them directly over phone or email.

2. Autoscaler -

   The application is designed to handle multiple users at once. To do this, an autoscaler is needed. Based on parameters like target_cpu_utilization, target_throughput_utilization, and max_concurrent_requests, the load balancer decides whether it needs to create a new instance to handle it. As we are dealing with several users which are further divided into two types - autoscaling is of utmost importance for the system.

3. Middleware -

   The Middleware is designed to handle multiple users and their sessions at once. It will provide api endpoint services to various all the front end functionalities. It will query the backend for viewing and inserted the requests and send the

responses back to the front end with various validations and error handling.

4. Database -

   The application uses a relational database (Cloud SQL) provided by Google Cloud Platform to store details of the applications users along with other information like donation requests, donations, volunteer requests etc. This data is stored in separate tables which are then used to populate the View Donations, View Donation Requests, and View Volunteer Requests pages that can be viewed by different users as required.

2.2 Autoscaling

Auto-Scaling is done by the Google App Engine (GAE) automatically. Scaling-Out is done by GAE when new instances are to be created for decreasing the response time provided by the application and to process large number of concurrent requests made by users. Scaling-In is done by the cloud when instances are not required due to low requests to save the resources used by the cloud. Google has the provision for automatic scaling which is the default type of scaling provided to any application. Automatic scaling of instances uses dynamic instances. Each instance has its own request queue. When the load balancer notices a large amount of requests in the queue of an instance, it automatically creates a new one. The cloud has a default set of parameters set to handle scaling of the application which you can adjust depending on the balance of performance and cost you need. The parameters are CPU utilization, maximum/minimum instances, throughput utilization, requests, idle instances etc. All such parameters have a default value which can be changed as per your requirement. We have used automatic scaling for our application as it offers multiple parameters and is convenient to change any time depending on the estimate number of users using the application.

3. Testing and evaluation

Testing is a very important part of creating a complete application which does not produce bugs, functions as expected to yield excellent customer service and is

resistant to malicious attacks. The different types of testing carried out on our application were:

1. Boundary Value Testing: Various boundary and unusual inputs were tested for Registration Page, Login Page, Donation Page, Donation Request Page, and Volunteer Page to ensure that only valid data was stored in the database and the application did not crash or fail to perform as expected in case of a bad input.

2. End-to-End Testing: The deployed application was tested from start to end for a number of scenarios on our mobile devices to ensure that all the different components were working perfectly and as expected.

3. Functional Testing: Each of the proposed functionalities were tested and it was ensured that the pages were working as expected and designed.

4. Integration Testing: The various components of the applications developed by different team members were integrated and tested to ensure that each of the components interacted with each other as expected and worked without any issues.

5. Unit Testing: Individual components of the application were tested on a stand-alone basis to ensure that the component had no bugs or defects.

Our application was tested for the following criterias:

1. **Proper validation of forms in application** -
   Forms in our application have been tested for their validation to make sure that only the correct information is entered by the user. This way a user cannot avoid giving out the right information. The fields have been validated for not leaving a field empty, entering a correct email id, entering a correct phone number, entering the correct password as per set criteria etc.

2. **Correct user access** -
   User access is a critical part of our application, as the users are clearly divided into two parts - donors and kitchens. The application was tested to reveal the correct information to a user as per the user type entered by them during registration. Only the donors could act as a volunteer and access the form, see the data entered by the kitchens and can enter data pertaining to donation of

items. Likewise, only kitchens can see the data entered by the donors for items being donated, view donors acting as volunteers and can enter data pertaining to request of donations.

3. **Working of functionalities** -
   The functionalities implemented in the application had to be tested for their working. We had to make sure the donations and volunteering requests added by the donors were added to the respective databases and available for the kitchens to see and the requests added by kitchens get added in the database and are concurrently visible for the donors to see. The sign up and sign in functionalities also worked as expected by creating a new user in the database and giving access to an existing user.

4. **Session Management** -
   The session, while in use, could unexpectedly close. In case of such a scenario the user must not be logged out and the session must resume from where it was left off. The application was tested for such a scenario where even after closing the application or after an interruption the same user was logged into the system on reopening the application.

5. **Response Time** - The application was tested for ping tests and it returned responses in 1 second and less from using a proxy server from California, USA.

6. **Testing of Deployed Code** - The deployed code was tested using automation testing for user inputs and user registration. It was also tested for multiple simultaneous users and their sessions. The application was also tested using boundary value analysis, and validation tests. The deployed code performed well on both frontend and backend validation scenarios.

4. Code

4.1 Functionality of every program in the system

- **src/pages/register -** This page is used to create and register a new user of the application.

- **src/pages/login -** This page is used for signing in into the application.

- **src/page/dashboard** - This is the main home page of the application. Depending on the type of user, different buttons to different pages are shown here. Users can use this to navigate to different pages.

- **src/page/donation** - This page is accessible only by the donors. This page is used to create donation requests by donors. Donors can list the items they are willing to donate along with their details, list of items, and date & time of pickup.

- **src/page/ViewDonationRequest** - This page is accessible only by the donors. Donors can view specific requests raised by different organizations and can search them on a city basis. Each list item has a view button which shows the request details and gives the user a call button to directly contact to organization which raised the request.

- **src/page/volunteer** - This page is accessed only by donors. Users who do not have food items to donate but however are willing to volunteer and help out can fill this form by entering various details like availability date, time and contact information.

- **src/page/DonationRequest** - This page is accessible only by organizations. Organizations who require specific items can raise a request through this form. They are given the option to enter item names, contact details etc.

- **src/page/ViewDonations** - This page is accessible by Organizations only. Organizations can view donations put up by donors, search and filter them by city. Each list item has a view button which shows the donation details and gives the organization a call button to directly contact to user which raised the donation.

- **src/page/ViewVolunteers**: This page is accessible by Organizations only. Organizations can view users who are willing to volunteer and contact them by email or mobile phone.

- **Main.py:** The python file is used to provide api endpoints to the front end for various functionalities in the front end. It uses a Flask framework to create a server. This server also queries the Cloud sql database via a proxy to get the data or insert the data as per the functionalities provided in the front end.

4.2 Installing programs and running them

1. Front End - We have used Ionic 3 framework to develop a cross platform application. We deployed it on a web url and generated a mobile application for android.
   The steps to do that are:
   A. “ionic cordova build --release android” command inside the Ionic code builds the apk in output folder in the platforms/android folder in the ionic source code folder.
   B. “keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000” command creates a keystore file to sign the application.
   C. “jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore Fooddonate-release-unsigned.apk alias_name” command signs the generated apk using our keystore file.
   D. “zipalign -v 4 Fooddonate-release-unsigned.apk Fooddonate.apk” command to optimize the apk.
   E. “ionic cordova build --release browser” command inside the ionic code build the code for deploying for website.
   F. Use the code in the www folder in platforms/browser and deploy the code on Google app engine using provided app.yaml file.
   G. The generated apk can be directly installed on an Android mobile device.
2. Middleware - We have used Python framework to develop middleware of the application. Along with its respective app.yaml file and requirements.txt upload it to a folder on google app engine. Use the command “gcloud app deploy” in this folder to deploy the rest services on app engine

3. Backend -
   The steps to do that are: Create a database on cloud sql using the sql file provided along with the code.

5. Individual Contribution

Darshan Dagly (ASU ID: 1215180174)

- **Introduction**:
  Everyday 150,000 tons of food is thrown away and wasted in the United States due to lack of consumption while an approximate 50 million Americans go hungry everyday or struggle to put food on the table on a daily basis. These statistics clearly show a gap between the two ends of the food consumption spectrum that needs to be addressed. Our application provides a multi-platform portal which connects the people who are willing to donate food and the organizations who are in need of these resources.

- **Design and Implementation:**

  The design of the entire architecture of the app was done by me and my teammates. We had discussed the pros and cons of different technologies and approaches before implementing the project using Ionic3. I was responsible for designing and implementing the front-end of the application like Donation Page, Registration Page, Donation Request Page, View Donation Page, Donation Details Page etc. I was also responsible to create endpoints for these pages using Python Flask through which the data entered by user was sent to the backend server and stored into the database. I was also responsible for designing some of the database tables which were used to store various details like user information, donation request information, donation information, volunteer information. I also added validations to the input fields of the entire application, ensuring that the data entered by the user was valid and in required format.

- **Testing:**

  Testing is an important part of any user ended application. It is one of the most crucial parts of the project as user facing applications are more prone to crashing and failing as the user inputs and interactions can never be predicted. For this part of the project, I was responsible to perform various sanity checks and tests on the applications including the final testing of the deployed application. I performed various testing like unit testing where I tested each standalone component, functional testing where I tested the overall functionality of the entire application, integration testing where I did sanity checks on the final integrated application and user input testing where I tried different unexpected inputs to ensure that all the different validation scenarios were handled. Once the project was deployed and the .apk file was generated, I installed the application on my mobile device and performed one final round of end-to-end testing before presenting it for the demo.

Viken Shaumitra Parikh (ASU ID: 1215126783)

- **Introduction:** Everyday 150000 tons of food is thrown away and wasted in the US due to lack of consumption while an approximate 50 million Americans go hungry everyday or struggle to put food on the table on a daily basis. Our idea of food donation system is to bridge this gap and provide a mobile and web portal accessible to everyone who would want to donate extra food, could be an individual or an organization and those organizations who would want to receive these food donations and distribute it to people in need. Additionally people who cannot donate food, could choose to volunteer to deliver these food requests.

- **Design and Implementation :**
  1. **Front End** - We choose a cross platform development framework - Ionic 3 which allows us to create a web application and a mobile application supported on android and ios. The web application would be hosted on Google App Engine and mobile application could installed using a generated signed apk. My role in this was to develop various pages and deploying the app. Routing for data across various pages and managing the session for users, along with creating various user donation requests and viewing them were the task accomplished. A search by city location was also a functionality provided to the users.
  2. **Middleware** - We choose Python and Python Flask Framework for it's simple and rapid development. To create a loosely coupled system, we created various api endpoints for each service provided in the front end. I worked on various api endpoints through the course of the project. Managing the session in the backend was one of the tasks of the application. The other major tasks were creating requests and displaying those requests based on the user access.
  3. **Backend** - We choose to have a Cloud SQL database as we required a relational database system for storing and querying various types of user and user requests. My role was designing the database system for various fields in users, donations, donation requests and volunteers tables.
  4. **Auto Scaling** - As Google providesThe cloud has a default set of parameters set to handle scaling of the application which you can adjust depending on the balance of performance and cost you need. The parameters are CPU utilization, maximum/minimum instances, throughput utilization, requests, idle instances etc. All such parameters have a default value which can be changed as per your requirement.

- **Testing:**
  Deployment and testing were one of the major tasks for our application. We deployed our web application and also the python flask serve on Google App Engine. Access control, validation testing for various data input and output, boundary value. My major tasks were testing the application in various test scenarios and deploying the application and database.I also did ping testing the application to check the response time. Also tested the application for multiple users and requests after integration of the application