**Executive Summary**

**Group Highlights and Individual Contributions**

**Group Highlights (author: Andy Huang)**

Our design team has made significant progress since the Design Review meeting and the Progress Report. Firstly, we have developed the interface and functionalities of the three individual modules: the client interface, the administrator interface, and the backend server. Secondly, we have interconnected these modules with application programming interface (API) calls implemented using HTTP request and JSON response. Thirdly, we have incorporated various third-party libraries and frameworks into our project in order to enhance its functionalities.

For the client application, we have designed and developed the user interface and the underlying program services. In addition, we have successfully integrated Google Maps, Google OAuth login and Google Cloud Messaging (PUSH notifications) into our frontend client software. As a result, the users are able to login with their Google accounts, to check for dining information such as location, menu item, price, hours of operations, to submit restaurant reviews or feedbacks and to receive any promotional events through the use of our Android mobile application.

For the administrator application, we have designed and developed the user interface and the underlying program services. As a result, the restaurant vendors can create their own profile page, upload images of the restaurant or food items, and to communicate with the customers using a real-time multicast messaging system.

For the server program, we have developed a python backend program that utilizes the bottle framework as well as several third-party libraries such as "google-api-python-client" and "python-gcm" to assist us with the integration with Google services. Furthermore, we have developed a database in SQLite and the table schemas which enable efficient data transactions. Finally, we have deployed all the server components onto Amazon Web Services (AWS). As a result, the client or administrator applications can send data requests such as read, write, update or delete from anywhere in the public internet domain and receive a response according to the requests.

**Individual Contributions: Andy Huang (author: Andy Huang)**

**Individual Contributions: Muhammad Azhar (author: Muhammad Azhar)**

**Individual Contributions: Guiqi Wang (author: Guiqi Wang)**

**1. The U of T Food Place Portal**

**1.1 Introduction**

This report summarizes the motivation, design, implementation and testing of a food place portal developed on the Android mobile platform as part of our final year design project course ECE496. The report concludes with suggestions of improvements and future work.

**1.2 Background and Motivation (author: Andy Huang)**

According to statistics from the University of Toronto, there are around 58,000 students and 12,000 faculty staff members in the St. George campus. [1] Due to their busy schedules, most of them dine on-campus. Although several bring homemade food to eat, most still buy their lunch from various dining locations in and around the University of Toronto St. George (UTSG) campus. According to the UTSG Food and Meal Services directory, there are more than 100 dining places on campus that offer a wide range of meals [2]. However, due to their busy schedule, most people miss out on many of these options.

Nowadays, there are a lot of useful portals where people can search for adequate dining locations. For example, Yelp provides information on various dining locations. Yelp users can search for their favourite food and restaurants by simply entering keywords or using the filter options. However, it only provides information on 15 dining places on campus [3] compared to the U of T Food and Services website, which covers 35 dining places [4]. Also, Yelp’s search engine is not comprehensive as inputting one keyword into the search engine cannot provide users with a complete list of relevant dining places [5]. On the other hand, the university provides menu and hours of operations of many dining locations through websites like UeaT and CampusDish [6]. However, it disregards places such as food trucks and hotdog stands. In addition, all these food portals are missing a two-way communication channel for the customers and the businesses to connect with each other. While customers can often leave reviews or feedbacks for the businesses, the businesses often cannot deliver real-time information such as promotional events and address multiple users effectively using an one-to-many messaging model.

In order to provide better assistance to students and staff members in searching for suitable dining places as well as to enable efficient communication, it is necessary for our U of T Dining Portal to facilitate a service whereby students and staff members can check for dining information through the portal and to provide a multicast messaging system. These services will not only enable users to determine where to find their favourite food and meals on campus, but also promote the less popular dining places to a wider range of students and staff on the UTSG campus.

**1.3 Project Goal (author: Andy Huang)**

The goal of our project is to provide a mobile platform for a sustainable food portal that exposes the dining options on campus and helps the users to find their favourite food and meals as well as to receive real-time communication of certain events from their subscribed food locations.

**1.4 Project Requirements (author: Guiqi Wang)**

**1.4.1 Functions**

1. The system shall store the information such as the name, description, address, working hours, and menu of the food places inside and around U of T, St. George campus.
2. The system shall be able to add new or update existing information in the food portal
3. The system shall display information about the dining locations upon users' requests
4. The system shall allow for an administrative entity to create a profile page, upload initial restaurant information and subsequently update these information.
5. The system shall allow the administrators to send real-time information to the users who have subscribed to that restaurant
6. The system shall integrate with social media and allow the users to login with their Google or Facebook accounts

**1.4.2 Constraints**

1. The design shall be developed for mobile devices running the Android operating system
2. The design shall require internet connection either 3G or Wi-Fi network
3. The design shall be hosted on the public internet domain to be accessible by the users

**1.4.3 Objectives**

Performance

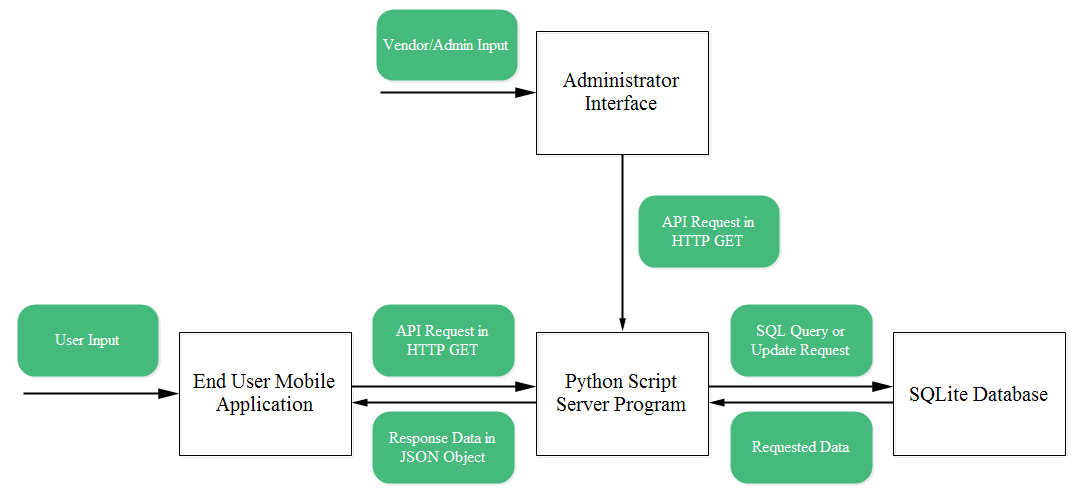
1. The response time of the software should be within 1 second [7]
2. The design should be backwards compatible up to Android 3.1 [8]
3. The design should allow for at least 100 requests to be handled at the same time
4. The design should cache some data locally to allow user access without internet

User Experience

1. The number of pages to navigate through to obtain the desired information not be more than 4 steps
2. The design should have a uniform font type that is readable for everyone
3. The design should provide security data encryption for administrators login

**2. Final Design**

**2.1 System-level Overview (author: Andy Huang)**Figure 1: System Block Diagram

****

End users (students and staff members) will first provide input using our mobile application‘s interactive user interface. Upon receiving an input, the application program will generate an application programming interface (API) request. These requests will be constructed in a HTTP URL format and sent to the server. Our server program will parse the client requests and construct query commands using the structured query language (SQL). The connection between a server and database is established via a connection object. Once a connection is established, our server program will query the database, which will return with the requested data that can be navigated via a cursor object. The server program will bundle the raw data into a structured object using JavaScript Object Notation (JSON). The JSON object will be passed back to the mobile application where it will be parsed and displayed on a user interface to the end users. In addition, we also have an administrator interface for the dining place vendors. This system will take data update or creation requests from an administrator as input and generate an API request to the server. The server program will parse the request and issue a SQL command to update the data in the database accordingly.

**2.2 Module-level Descriptions and Design**

**2.2.1 End User Mobile Application (author: Andy Huang)**

Components:

* User interface
* Google Map
* Google Cloud Messaging Receiver
* Logic Process Center
* Network Socket

Input:

* User requests to retrieve data through interactive interface
* Server response of requested data in JSON format

Output:

* API request to server to get the corresponding data
* Requested data shown on the graphical user interface

Functionality:

The client interface application handles the request from the user and generates the corresponding API to server to ask the desired data. While the client also accepts the corresponding data from the server and transform the data into readable format on the application screen for users. In addition, the client also receives PUSH notifications sent by the restaurant vendors and initiates the OAuth authentication protocol to enable users to login with their Google accounts.

**2.2.2 Administrator Interface (author: Guiqi Wang)**

Components:

* User interface
* Logic Process Center
* Network Socket

Input:

* User request to update data through interactive interface
* Server acknowledgment message

Output:

* API request to server to update the corresponding data
* Transaction status (success/fail)

Functionality:

The administrator interface handles updating and adding request and generates the corresponding API to the server to update existing or add new data into the database. The client will accept the corresponding transaction status message from the server and display it on the interface.

**2.2.3 Server Program (author: Muhammad Azhar)**

Components:

* Bottle framework: server framework for Python servers
* SQLite: access the database and transfer SQL messages and resulting tables between the server and database
* JSON: Organizes and encapsulates the requested data to be sent to the client
* TCP connection with the client for communication with the client

Input:

* Client request for some data or to change data

Output:

* Acknowledgement message accompanied by the requested data (if required)
* Error message if the server fails to execute the client request

Functionality:

When the server receives a client’s message, the parser parses the message to determine the client’s request. If it cannot realise the request, the server API sends an error message to the client. Otherwise, the API creates the SQL message depending on the client’s request. This SQL message is transmitted through SQLite to the database. When the server’s request is completed, SQLite fetches the output message and table (if needed) from the database and sends it to the API. The table’s data (if it exists) is then constructed by and encapsulated into a JSON object. Once the server creates the acknowledgement message, it sends the message with the JSON object (if it exists) to the client.

**2.2.4 SQLite Database (author: Andy Huang)**

Components:

* A SQLite db file with the table schemas and the data type designed and developed prior to application launch

Input:

* SQL queries to get the food information that is of interest
* SQL queries to manipulate or add food information data into the database

Output:

* The food information data that satisfy the query conditions
* The updated database due to the SQL update/add requests

Functionality:

The database stores and provides information such as a dining location's name, description, images, hours of operation, address, menu items and price. The database handles both the query and the data manipulation requests from the server. In the case of querying, the database will get the data that satisfy the query and return this information to the server. In the case of data manipulation, the database will add or update information depending on the type of the SQL manipulation request.

**2.3 Assessment of Final Design**

**3. Testing and Verification**

**4. Summary and Conclusions**