iTaste

1. Introduction
iTaste is an Android-based application that enables its users to share news about foods and drinks that they have tasted on the USC campus. It brings together food lovers and food seekers: those who want to critique various USC food venues, those who want to tell others about special events offering snacks, desserts, beverages, and even full-fledged meals, and those who just want a bite or something to quench their thirst. Each posting of a review or food announcement is called a “Taste”, and users can tell other users about what they have tasted by posting a Taste.

![iTaste Home Screen](image)

**Figure 1 iTaste Home Screen**

1.1 Basic functions
Client

*Add a Taste*

1. Allow users to add a Taste regarding food-related events on campus.
2. Users can specify event time, location, categories, rating, and content in a message. For free-text input, users can input via voice transcription or text-based input.
3. Categories will group Tastes into the following: non-alcoholic beverages, alcoholic beverages, meals, snacks, desserts, and other.
4. Users can attach an accompanying picture to the Taste.
5. Messages will be sent to the server with user-specified information.
6. Each mobile user will be identified with their Gmail account, which will show up next to the Tastes.

View/Delete/Edit Tastes
1. Users can access a list of all of their Tastes.
2. Users can edit/delete their Tastes.

Search Tastes
1. Users can search for Tastes within a user-specified radius centering on their location.
2. Users can search for proximity Tastes by specifying a point on a Google map. The proximity radius can be specified by the user.
3. Users can rate Tastes.
4. Users can search for Tastes by choosing a category.
5. Users can further refine the initial results from category or proximity search by using free-text keyword filter. Keywords can be provided via voice transcription or text-based input.

Server
1. Accept requests from a client to add a Taste into the database.
2. Accept requests from a client to delete a Taste from the database.
3. Delete requests will only be fulfilled if the user requesting is the owner of the Taste.
4. Accept requests from a client to query for Tastes from the database.
5. The server will store data on an Oracle DBMS.

1.2 New ideas/features
1.2.1 Searching for nearby Tastes
Users who are not sure of what they crave but want to find something to munch on nearby can use this feature to find Tastes whose location are nearby the user. The application performs a kNN query for the 25 nearest Tastes to the user’s current location. Additionally, the user can use iTaste’s dynamic filtering functionality to narrow down the search result.

1.2.2 Bookmarking interesting tastes
If a user finds a Taste that is further off in the future, they can use the bookmarking capability to save it to their personal bookmark list for future reference. This prevents the user from having to remember the taste that they are interested in and having to re-search for it prior to the Taste’s event time. Users also have the ability to remove items from their bookmarks to clean up their list or to remove bookmarks that they are no longer interested in.

1.2.3 New Taste Alerts
One of the hallmark features of iTaste, the New Taste Alerts feature, enables users to be constantly connected to other users. New Taste alerts allow users to be instantly alerted of any new Taste that are posted within a configurable distance from the user. By default, the application will alert users if there are new Tastes posted within 200 yards of their location. Once a new Taste is detected within the specified radius of the user’s location, the application will post an Android notification to signal to the user that there are new Tastes in their vicinity.
1.2.4 Walking Directions

USC campus is a big place, and sometimes it may be daunting to navigate the brick filled landscape, especially for new students looking for something nice to eat or refreshing to drink. iTastes provides turn-by-turn walking directions within the USC campus to lead the user from their current location to the Taste they are interested in so that they can take the fastest route and start munching on some goodies.
Figure 3 Walking Directions Screen
2. Application
The application is designed with strict separation of concerns following a 3-tier architecture model. Each tier is insulated from the others through a well-defined interface, resulting in a design where there are zero dependencies between the tiers. Figure 4 shows a diagram depicting the iTaste architecture.

![iTaste Architecture Diagram]

**Figure 4 - iTaste Architecture**

2.1 Server
The server is designed as a service model that produces data for the client to consume. The server has no knowledge of the existence of a client and will merely produce data as a response to queries or manipulate the persistent data as instructed by appropriate requests. All client requests are channeled through one of the three Servlets, which are described in detail below.
2.1.1 PostServlet
The PostServlet handles all data manipulation requests, such as, requests to post a Taste to the server and bookmark a Taste. The PostServlet calls appropriate methods provided by the database interface and these methods then query the database. For instance, to handle a request to post a Taste, the PostServlet retrieves all the request parameters, creates the appropriate Taste wrapper object, sends this Taste object to the database interface, and the database interface builds the appropriate insertion query. Depending on the success or failure of the processing of the request, the PostServlet sends an appropriate XML response to the client.

2.1.2 QueryServlet
The QueryServlet handles all data retrieval requests from the client, such as, requests to retrieve all of a user’s Taste. It also processes spatial queries, such as, retrieval of all Taste within a certain radius and all Taste that is nearest to the user’s current location. For example, to process a request to retrieve the nearest Taste to a user’s current location, the QueryServlet gets the necessary parameters from the request, such as, user’s latitude and longitude, and then passes these parameters to the database interface, which then queries the database and returns a list of matching Tastes. If retrieval of the Tastes is successful, the QueryServlet builds an XML of all the Tastes and sends this XML to the client. Otherwise, the QueryServlet sends an appropriate error message.

2.1.3 ImageServlet
The ImageServlet handles all image retrieval requests including subsampling logic for thumbnail requests. ImageServlet queries the database based on query parameters provided by the client and obtains the binary data of the image via the BLOB column of the Taste table. This binary data is then subsampled to create a thumbnail (if applicable) and then the data is written to the ServletOutputStream as a response to the client’s request.

2.2 DB
An entity-relationship diagram for the database component is shown below in Figure 5. There are three essential entities: Place, Taster, and Taste. Place represent a location on the USC campus, such as, the Lyon Center, Taster represents a user, and Taste maps directly to a Taste.

The relationship among the entities is described as follows: multiple Tastes are located at a Place and multiple Tasters posts, rates, bookmarks and is alerted of multiple Tastes.

The actual database schema is designed using the object-relational model. Place, Taster, and Taste are mapped to Oracle user-defined types, place_type, user_type, and taste_type, respectively. Taste’s multi-valued attribute, categories, is mapped to a separate type, category_array_type, which is an array of category_type. The database tables, which consist of place, taster, taste, alert, bookmark, and rating, are designed to store these user-defined types and map to the relationship among the given entities.
2.3 Android client
The android client interacts with the user via a set of activities. Below is a subset of the representative activities, in terms of user interaction. Included with each activity is a description of its basic purpose and screenshots.

2.3.1 Representative Activities
2.3.1.1 AddTasteActivity
This activity is used to add a Taste to the application. Users specify Taste information and it is the launching activity for LocationSelectionActivity and TakePictureActivity. This activity is extended and used by EditTasteActivity with almost no changes. The only difference is that EditTasteActivity pre-populates the form with the existing Taste information.
2.3.1.2 LocationSelectionActivity

This activity allows the user to select a location for the Taste they are adding or editing. Every building on the USC campus is presented as a possible choice for the user as pins on a Google Map.
The coordinates of the buildings on the USC campus are parsed from a KML file and stored into the database in the Location column of the Place table. To display the corresponding pin for each building, an XML file containing the ID, name, abbreviated name, and centroid of each building is parsed when the user starts the LocationSelectionActivity.

2.3.1.3 TakePictureActivity
This activity allows the user to take a picture for a new or existing Taste. It utilizes the Android phone’s camera to take a picture and temporarily saves it onto the phone’s local storage prior to sending it to the server for persistent storage. After taking a picture, the user is presented with an option to retake the picture.

2.3.1.4 ListTasteActivity
This is the basis of all activities that show a list of Tastes, such as BrowseAlertActivity, BrowseBookmarkActivity, BrowseMyTasteActivity, BrowseNearbyTasteActivity, SearchTasteResultActivity. It displays a list of tastes, each with its Title, Description, Location, Event Date/Time, Rating, Number of Raters, Taster, and a thumbnail image. Its categories are shown as icons, adhering to the following legend in Table 1. When a user selects a Taste, the ListTasteActivity initializes and launches ViewTasteActivity to display the Taste in a more detailed manner.
Figure 8 Tastes List Screen
2.3.1.5 SearchTasteActivity

This activity allows the user to search for Tastes by location and categories. On initial load of the activity, the pin is placed at the user’s current location. Users can view the USC campus map by clicking on the “Center on SC” button. They can then move this pin to specify a point-radius and select a list of categories within which to search. The server will return all Tastes that belong to at least one of the categories in the user’s list and whose location fall within the user specified point-radius.

<table>
<thead>
<tr>
<th>Table 1 - Categories Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Non-alcoholic Drinks" /></td>
</tr>
<tr>
<td><img src="image" alt="Alcoholic Drinks" /></td>
</tr>
<tr>
<td><img src="image" alt="Meals" /></td>
</tr>
<tr>
<td><img src="image" alt="Snacks" /></td>
</tr>
<tr>
<td><img src="image" alt="Dessert" /></td>
</tr>
</tbody>
</table>

![Figure 9 Search By Point-Radius and Categories Screen](image)
2.3.1.6 **ViewTasteActivity**

This activity shows the user a detailed view of a selected taste from a ListTasteActivity. Depending on the user, different features are available. If the Taste is a Taste that the user posted, then he/she has the option to Edit or Delete it. If it is a Taste that is posted by another user, then the user will be able to add/change the rating for this Taste and bookmark this Taste. Lastly, all users will be able to get walking directions from their current location to the Taste’s location and page through the list of result Tastes.

![Figure 10 View My Taste Screen](image-url)
2.3.1.7 **New Taste Alerts**

The application periodically contacts the server to check if any new Taste has been posted with a location within a configurable distance from the user. If the server responds with new Taste alerts, the user will be notified via Android’s notification application and clicking on the notification will start iTaste and bring the user to the BrowseAlertActivity.
2.3.1.8 **Client-Server Interface**

The client communicates to the server using the ServerConnection class as an interface to the server. All communication details are abstracted from the client and handled by the interface. Server responses are also collected by this interface, which parses it and returns to the client Java objects. Thus, the methods of communication with the server and the server response types are abstracted from the client, providing better maintainability and separation of concerns.

3. **Installation and configuration**

The application consists of 3 main components: client, server, and database. Below are instructions on how to install, configure, and deploy each component to get the application working.

### 3.1 Database Installation

1. In the `src/etc/db` directory, you should find an SQL script called `create_db.sql`. Run this script on an Oracle 10g/11g database.
2. Note the database host, port, name, username, and password (if applicable). You will need this to configure the server.

### 3.2 Server Installation

1. In the `src/webapp/iCampusProfiler_server/WEB-INF/classes` directory, you should find the source code for the server-side component of iTaste. In `src/webapp/iCampusProfiler_server/WEB-INF`, you should find a `web.xml` file. Open this file and modify the database_[host, port, name, username, and password] properties with the correct values you noted in step 2 of **3.1 Database Installation**.
2. Your server should now be configured to communicate with the database.
3. Copy the entire directory `iCampusProfiler_server` located in `src/webapp` into Tomcat’s `webapps` directory. This has been tested on Tomcat 6, so please use Tomcat 7 for best compatibility.

4. To test if your server component is installed and configured correctly, navigate to `http://[tomcat-host]:[tomcat-port]/iCampusProfiler_server/`. Fill in `[tomcat-host]` and `[tomcat-port]` with the appropriate values for your Tomcat instance. Note this URL as you will need to provide it to your Android client to configure the client to communicate with the server. You should see a success message “Welcome to iTaste! Database connection successful.”

### 3.3 Client Installation

1. In the `src/android/iCampusProfiler_client/src` directory, you should find the source code for the client-side component of iTaste.

2. In the `src/android` directory you should find the `iCampusProfiler_client.apk` file. To install this on your Android phone, go to your android SDK installation directory and use the `adb` in the `tools` directory to install. We used this command to install it onto our phones:

   ```
   ./adb install <path-to-apk-file>
   ```

3. This application is configured to interface with the server at `http://hamedaan.usc.edu:8080/team3/`.

If you wish to build your own apk file from the source code or use another server, follow the steps below:

1. You will need to go to the `src/android/iCampusProfiler_client/res/values` directory and modify `strings.xml` file. Open this file and modify the `serverBaseUrl` property with the URL that you noted in step 4 of 3.2 Server Installation.

2. You will need to get a Google Maps API Key otherwise the MapView that is used in the application will not display. For information on how to obtain a Maps API Key visit `http://code.google.com/android/add-ons/google-apis/mapkey.html`. Once you have obtained the Maps API Key, go to the `src/android/iCampusProfiler_client/res/values` directory, open `strings.xml` file and modify the `maps_api_key` property with your Maps API Key.

3. Your client should now be configured to communicate with the server. Build the client source code and package it into an apk file using ADT (Android Developer Tools).

4. To install this newly built apk file on your Android phone, go to your android SDK installation directory and use the `adb` in the `tools` directory to install. Use this command to install it:

   ```
   ./adb install <path-to-apk-file>
   ```

**Important Note for Client Installation**

- Ensure that Wi-Fi is enabled and Internet connection is available for your phone. e.g. if you are using USC wireless, be sure that your phone is registered with the system so that it can access the Internet via USC Wi-Fi.

- Ensure that your location can be determined by Wi-Fi. To do so, go to your phone’s `Settings -> Location & security` and check the “Use wireless networks” option. This is for ease of testing since Wi-Fi location is faster to obtain when indoors. If you wish you use GPS, set the `useWiFiLocation` property in `src/android/iCampusProfiler_client/res/values/strings.xml` to false and rebuild and reinstall the APK file. This will cause iTaste to use the best location provider with no requirements, so GPS should be used when available.