Problem Statement
Today, mobile applications for finding restaurants and food venues enable people to search for restaurants via keywords and filters. However, this is still a cumbersome task - there is no fine-grained way to filter specifically what someone wants, for example, a specific price range, type of restaurant, times for happy hours, reservations. This app enables users to upload profiles for what they look for in a restaurant, and enables personalized searches according to those preferences. This allows users to choose a profile they want to search with, and find the most relevant results according to that profile, which allows the user to find the restaurants for whatever occasion faster. The eventual goal for this application is to allow a user create a profile through a web interface, and allow that user to use the profile through a mobile interface to filter search results for restaurants.

Artifacts
The deliverable components for this project are detailed below:
1. Source code
   a. Android app
   b. HTML5 mobile website
   c. Web app
2. Documentation for using the project
   a. Documentation for using the app
   b. Developer documentation exposing web APIs from the web app
3. .apk file for the Android app

Methods and Techniques

Diagram of the system architecture:

- RevMiner
- Web App and HTML5 Site (web service)
- Web Browser
- RPC Client
- Android App

Android App
This application will use the standard Android SDK (programming language in Java). In essence, this is just a client RPC stub that communicates with the service. The user can input profile information and perform searches via this application, but must require an Internet connection to communicate with the service. A bonus feature for this application is to enable result caching, which allows the user to search some restaurants without an Internet connection.

**Web app and HTML5 mobile website**
This service will use the Django framework (programming language in Python), with MySQL as a database management system, and HTML5/JavaScript as the view layer. This service will be hosted on the UW CSE student hosting servers (abstract.cs.washington.edu). The service enables users to create a profile, which asks the user to fill out a form, either through a standard web browser or through the mobile HTML5 site, containing restaurant preferences. Such preferences include, but are not limited to: popularity by review, cost, type of food, location, happy hour, reservations, and open hours. This information is stored, per user, in the database, and can be retrieved via an authenticated web API.

Yielding personalized search results relies on revminer for providing the initial data. The search query is first made to revminer via our service to yield all possible search results. Then, the meta data of each of these search results are collected by our service, which then filters out irrelevant results and then displays the first ten relevant results to the user.

The HTML5 website exists as a graphical web interface for both modern browsers and mobile devices, and as a web API for RPC calls between revminer and the Android application.

**Division of Work**

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Eric</th>
<th>Siavash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android App/HTML5</td>
<td>Android App - build pages, and RPC stubs for communicating with the service.</td>
<td>HTML5 - build all templates required for Django, for both communicating with revminer and mobile devices.</td>
</tr>
<tr>
<td>Service - revminer</td>
<td>Connect with revminer API, implement search result retrieval and output to views</td>
<td>Implement filtering via profile information and build profile controller</td>
</tr>
<tr>
<td>Service - Android</td>
<td>Enable authenticated logins to service via Android device</td>
<td>Implement results display and GPS for Android device.</td>
</tr>
</tbody>
</table>

Success will be evaluated by using test profiles and search terms, then figuring out how relevant the results were according to the results we wanted. We will visualize this with a diff of our results, as well as percent accuracy comparing user expected value versus experimental value.