Architecture / Platform Decision:

Based on each of our member's experience, we decided to use Python for our web application. We felt this would provide us a familiar playing field while learning new skills like hitting API's, connecting to a database, and using third party services like OAuth. Also, Python easily allows us to manage our dependencies using pip and virtual environments.

For the framework, we decided to use Django because there are many resources available and it is easy to prototype with. Also, creating a project using Django allows us to compartmentalize the project into several smaller apps, each of which serves its own purpose. Since our web app consists of a variety of components (logging in, hosting or joining, restaurant finding, and data management), we can make full use of Django to separate the parts cleanly and have them connected through Django's URL system. Additionally, Django has extensive support for Bootstrap that we will use to enhance our frontend without worrying too much about HTML and CSS.

To fulfill the database requirement, we chose to use MongoDB. Since our web app creates sessions for any number of users, we had to rely on a non-relational database that would allow us to expand or modify sessions as we needed, which we could not do easily with the strict schemas of a traditional SQL database.

An alternative platform we considered was to use Flask instead of Django. We ultimately thought Flask was not a good fit because we would need to use a lot of external libraries to achieve the same functionality provided natively by Django. Two such functionalities are built in database support and the admin panel that provides a clean interface in the browser to manage the application. We also considered using JavaScript for the web application but chose not to as we were not very familiar with it.