Software Implementation and Testing Document

For

Group <8>

Version 1.0

Authors:
Alex J
John F
Justin N
Mason J

1. Programming Languages (5 points)

The majority of the code is written in Python, with heavy use of the flask/SQLite3 libraries to set up the database used within the application. While aware of the limits in Python's efficiency, the scope of the project has yet to reach a point where any sort of overhead has become a problem and we have found it would be a better decision to focus on writing higher quality code in an easily used language instead of needlessly tearing down functional structures when a clear idea of the final scope appears to show that careful coding is enough to avoid performance problems. The interface has been written in HTML, it continues to prove effective and easy to use and is currently only necessary for it's basic interpretation of the data.

2. Platforms, APIs, Databases, and other technologies used (5 points)

Flask has proven an effective library for the needs of increment two so we have decided to expand the structure created in increment one using it. Though it does stand as the most likely feature to run into conflict with some of the more advanced features planned within the scope of increment three.
SQLite3 is the database engine that the recipe and pantry databases are based upon.
Finally, we implemented bootstrap in order to overhaul the visual design of the html user interface.

3. Execution-based Functional Testing (10 points)

Functional Testing is still relatively simple, following the example of the demo video from the progress report. All pages are tested to ensure user navigation is smooth and without error. Each function is tested to input data, and taking that input data we can both see if it is presented properly in the list display and that recipes can be searched by name with the corresponding function. Testing was done to verify that when a recipe is added to the database that all related information is inserted into their respective tables and are able to be referenced through the primary key.

4. Execution-based Non-Functional Testing (10 points)

The current scope of the project does not require any large degree of non-functional testing. The systems at play do not incur the kind of overhead that we can expect to see performance issues. The main non-functional requirement we tested for was security, we thought it would be prudent to ensure the system was protected against SQL injection.

5. Non-Execution-based Testing (10 points)

Code review was done through github to check if the given implementation was ready for grading, and checked thoroughly on an individual basis. All page renders were visually inspected for improper outputs.