Software Implementation and Testing Document

For

Group <13>

Version 1.0

Authors:
Jacob Siegel
Gam San
Dean Arriveiello
Braden Seidl
Declan Murphy

1. Programming Languages (5 points)
List the programming languages use in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).

- *NoSql (MongoDB) – Good understanding of how to use in application*
- *SQLite - Content Providers for local data storage*
- *Java – Android Apps are built in Java*
- *XML - elements are displayed from xml*

2. Platforms, APIs, Databases, and other technologies used (5 points)
List all the platforms, APIs, Databases, and any other technologies you use in your project and where you use them (in what components of your project).

- *Android Studio - Development environment for Android apps*
- *Fragment Controller to manage th13e different scenes - contained in layout and navigation folders*
- *SQLite with MainDatabaseHelper*
- *Firebase - remote SQL database*

3. Execution-based Functional Testing (10 points)
*Describe how/if you performed functional testing for your project (i.e., tested for the **functional requirements** listed in your RD).*

Main method of functional testing throughout our android app is through physical testing through the emulator provided in Android Studio by manually inputting the fields to test the functionality. The emulator gave us access to multiple different devices to test on.

Did extensive testing on input fields to make sure no errors would arise, and created many try/catch cases to deal with these issues.

The main menu, food page, recipe page and fitness pages were tested using a virtual emulator of a Google Pixel device. We tested all input fields for bugs when inputting large strings or non-alphanumeric characters. We also tested the limitations of our menus as we added a large amount of recipes/workouts/food items to their respective pages.

4. Execution-based Non-Functional Testing (10 points)
*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

Non-functional requirements are tested also through Android Studio's emulator by checking if the displays on the app correctly correspond to what it ideally should display.

We tested user account creation/authentication by creating and testing functionality within the app on multiple different accounts. We also tested our data persistence (MongoDB) by inputting test data into the recipes, exercise, and shopping list fields then relaunching the app. We also tested persistence after crashing the app and also while changing between user accounts.

5. Non-Execution-based Testing (10 points)
Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

Code was inspected by group members. Each member had to understand the committed code in order to be on the same page with issues/development. As a group, we inspected the code to make sure errors would not arise, and if they did we were able to fix them initially- but that is as far as our non-execution-based testing went. Other forms of Non-Execution-based testing were difficult or redundant to perform, as the majority of the features in the app rely on text input from the user.