**Software Implementation and Testing Document**

**For**

**Group 27**

Version 3.0

**Authors**:

Ian Estevez

William Hudmon

Jack Throdahl

# Programming Languages (5 points)

The programming languages used include Python, HTML, CSS, SQL, and JavaScript. Python is the primary language used to create the database, the tables therein, the external MySQL users relevant to it, as well as the main web application, all its business logic, and the routing between the different front-end pages. HTML was used to create the user experience, while CSS was used to style and configure the appearance of each HTML page. SQL was used to run the queries and commands required to create the database and its tables, as well as insert any new user accounts into the user table. We chose Python for its versatility, ease of use, and well-suitedness towards the development of web applications, not to mention it is a common language for all of the team members. JavaScript was primarily used on some select pages to dynamically hide or show special forms depending on user input. HTML and CSS were used to sculpt the user experience, considering the scope of the application should not require much outside of the capabilities of these languages other than some dynamic aspects moving forward. Finally, SQL was used to make the database, as it is not only a common language among the team members, but also simply one of the most commonplace languages used for making queries.

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

The database component of the project uses a MySQL server. The primary Python module of the project, involving routing between front-end pages, creating the database, running queries on the database, and taking information from and presenting information to users, was created with Flask 2.2.2. The primary platform used to store the project source code is GitHub.

# Execution-based Functional Testing (10 points)

Functional testing for the project was performed iteratively as features

representing functional requirements were being implemented, and consisted mainly of performing walkthroughs of niche use cases to ensure no errors occurred, and the site functionality and output was as expected. Generally, as features were being implemented, the site was run and re-run from the command line, and a use case relating to the functional requirement in question was attempted to discern whether the feature worked correctly. Whenever errors were encountered, or a site crash occurred, the relating code (such as the appropriate function in the main Python module) was stepped through, or the peripheral dependency issue within the project was addressed, and the above-mentioned process was repeated.

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

Non-functional testing was generally not performed for the project, considering solutions for the plan to distribute the database are still being developed, and the scope of the project renders safety or security concerns a non-issue, as the project is still meant to run locally from a command line.

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

The majority of testing for the project consisted of general inspections of code quality, functionality, and conciseness, deeming code to be acceptable if it performed its intended function. Otherwise, this was accompanied by walkthroughs of use cases performed on the site, as mentioned in Section 3, ensuring each major code segment was linked to the implementation of some functional requirement.