Software Implementation and Testing Document

For

TripleR by WORM

Version 1.0

Authors:
Melissa Ma
Orlando Lewis
Rachael Scott
William Tsaur

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).

- Python: along with python we are using flask for the API because of its lightweight structure and python is a language that is easier to understand amongst all. Pairing python database development is also easy to navigate.
- Swift: XCode on MacOS. We use this for the front-end development of the TripleR app. We chose this language because a large portion of us were familiar with XCode and mobile app development on iOS. We all also have iOS devices to implement testing in later iterations.

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).

- Azure Devops: We use Azure for its server to host the API. It can be accessed from Microsoft Azure site.
- MongoDB: NoSql Database located on the MongoDb site.
- Flask: back-end development web framework for API.
- XCode: front-end development and deployment in simulated iPhones.
- iPhone: testing platform.
- Cocoapods:
    a. EMTNeumorphicView is used on the Report, Rights, and Record button to apply a neumorphic design to the buttons
    b. RealmSwift/Realm is used to persist local data on the phone so that filled out forms can remain filled out if a user exits the application or exits the page
    c. CollapsibleTableSectionViewController is used on the Rights page, subpages Rights, Tips, and Resources to produce a collapsible table view where questions are headers and clicking on the questions would display the answers. This maintains readability and cleanliness of the pages.
- Postman: to help test the API

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).*

To test the application, we installed the application on our phones and tried out all the features ensuring that they worked as intended and that everything else continued to work as it did before. Each use case we implemented in the sprint was tested on a variety of iPhones to ensure functionality.
For the API, Postman was used to generate reports, delete reports and “single out”/ edit reports using the GET and DELETE functions on the platform.

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).*

To test the application, we installed the application on our phones and tried out all the features ensuring that they worked as intended. Not much testing has been done regarding battery life/cpu consumption as of yet.
For the API, no testing was completed for non-functional testing.

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

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

No code inspections/reviews/walkthroughs have been done yet. Most GUI testing done is execution-based on live devices. A member brought up a concern about a file and another verified that the code was extraneous and deleted for readability.