**Software Implementation and Testing Document**

**For**

**Group <NoteWorthy>**

Version 3.0

**Authors**:

Lexie W

Humbert T

Daimeun P

Tyler G

David M

Orlando K

**1.**  **Programming Languages (5 points)**

*List the programming languages used in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).*

*1.     HTML: Used for frontend user interface.*

*2.     CSS: Used to add styling and flare to HTML components.*

*3.     JavaScript: Used for implementing functionality of our Web App’s note taking and storing interface.*

*4.     Python: Used for backend MongoDB database and communication with front end form submissions.*

**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).*

*1.     MongoDB: we use this for our backend in terms of hosting our database.*

*2.     Flask: Used to facilitate a connection between React and MongoDB.*

*3.     React: Used for real time rendering of our front-end interface. (In node\_modules folder)*

*4. Bcrypt: for password encryption*

5.

**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).*

*Overall*

*- Tested the most used path*

*User Accounts*

*- In order to test our login/register system, we created test users that we would use to communicate with our MongoDB hosted database. From MongoDB’s interface, we were able to see our test users’ emails and associated encrypted passwords. Afterwards, we had a test page on our backend named IsUserLoggedIn that would return the values of Boolean variables we were using to check if a user was successfully able to log in. As an example, we have a Boolean variable that tells us if a user attempted to register an account with an existing email address. During the testing for this case, the Boolean variable told us that the email was indeed a duplicate. Other test cases were approached in the same manner.*

*Proprietary Note taking interface*

*- Physically typed in the note box and pressed tab till things worked.*

*- Added a current topic indicator to test where position is when changing notes.*

*Storing and Viewing of Notes*

*- Using visual studio collaborative session, we all attempted to fix the issues that we had and then had two of our members loading and testing the front end to see what the output was until it displayed to our expectations.*

**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).*

*User Accounts   
1. Our passwords were initially stored as plain text in our MongoDB hosted database. Using the bcrypt library, we were able to take the passwords submitted by the user during the registration phase and encrypt them before storing them in our DB. During the log-in phase, the password is recovered from our DB, decrypted, and compared to the password the user submitted in order to grant them access to the webapp’s dashboard.*

*2.     Reliable connection to database through “middleware” = Done*

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

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

*All portions of code were worked on with two or more group members. This allowed us to review each other's code while actively working on it. It also kept all code written stay within our style guidelines. Once the code is integrated into the full project our team goes back over each section of newly added code and reviews the code for any issues with integration as well as general bugs.*