The Mess Matrix

Design Document

Final-Design-Document-TheMess.docx

Version 1.1

March 14th, 2019

TheMess

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Mike Jeromin	Initial Rough Design Document Draft	03/03/2019
1.1	Mike Jeromin Sam Jones Ebraheem Aldubis Skyler Knecht	Final Design Document	03/14/2019

Contents

1. Introduction

Overview:

The Mess Matrix is a tool that will allow users to protect themselves against an observer trying to steal a password being typed into the keyboard during login. A 6x6 matrix filled with random characters from a-z and 0-9 will be presented on the screen, and the user can refresh the display or confirm that his or her current password letter is on screen until the complete password has been entered. Encryption is also used to store the passwords.

The Mess Matrix project will be written in Python and embedded in a webpage using HTML, CSS, and JavaScript. The application will have a file containing all the possible usernames and passwords along with an encryption file.

Deliverables/ Delivery Dates:

March

3/4 Design Document #1 Rough Draft
3/18 Design Document #1 Final Draft
3/27 Design Document #2 Rough Draft

April

4/3 Design Document #2 Final Draft
4/8 Status Meeting/ Presentation **Prototypes DUE**
4/15 Press Release #2 Rough Draft
4/17 Press Release #2 Final Draft
4/22 User's Manual Rough Draft
4/29 User's Manual Final Draft

May

5/1 PROJECTS DUE!!!
5/1 Final Presentations
5/6 Final Report Due

Review of Plan and Project Schedule:

TheMess will meet on Wednesday every week and review the current status of the project plan and schedule. The team will then determine if they are on task for the next deliverable that needs to be developed. If the team finds that they are in fact behind they will take the necessary steps to get back on schedule.

Terms and Acronyms:

SLCM: Software Life Cycle Model, a software development methodology used by software engineers.
Encryption: The process of encoding messages or information in such a way that only authorized parties can read it.

GUI: Graphical User Interface, allows user to interact with electronic devices through graphical icons and visual indicators.
Python: A high-level, general-purpose programming language that emphasizes readability.
Shoulder Surfing: The practice of spying on the user of an ATM, computer, or other electronic device in order to obtain their personal access information.
Kanban: A visualized scheduling system used to improve efficiency.
Matrix: An array of numbers, symbols, or expressions arranged in rows and columns.

2. Project Organization

2.1 SLCM:

For the development of the program, the Kanban Life Cycle Model will be implemented. As a team, we decided that the Kanban life cycle model would be the best method to use for our program. We believe this because Kanban was originally an Agile life cycle model that focused primarily on continuously delivery of a project without overworking the development team. Kanban would allow the members of the team to know exactly where they were in the project development cycle by being able to visualize the work through a well-planned-out task board. To implement this feature, we will be digitalizing our board by using an app called Trello. We will also be able to release multiple different versions of the application by prioritizing features to guarantee that all must-have functionalities make it into the application.

2.2 Responsible parties for project functions and activities:

Team Leader Skyler Knecht is in charge of establishing meeting times and assigning responsibilities to the team, along with having a hand in some of the Python programming. Project Architect Sam Jones is in charge of the majority of the Python application programming. Lead Tester Ebraheem Aldubis is working closely with Skyler and Sam to ensure that our prototype and final project gets released with no bugs and works logically. Configuration Management Specialist Mike Jeromin is in charge of documentation within the blog, and version control on GitHub.

3. Managerial Process

3.1 Team Reporting/Monitoring Mechanisms:

Each week, the team will judge their progress at their weekly meetings. As stated above, the scheduled meetings are on Wednesdays each week. Throughout the meetings, the team will first decide if the previous weeks goals were met to determine if they are in fact on schedule. If those goals are in fact accomplished a new set of goals will be set. These goals would be determined by the Work Breakdown and Scheduling function along with the list of Team Deliverables. Communication among the team is a major key of how much progress we make. By following this procedure, TheMess will ensure that they will meet every deadline.

4. Technical Process

List of Tools and Practices:

- **Atom**
- **GitHub**
- **Git**
- **Visual Studio**
- **HTML**
- **CSS**
- **JavaScript**

Documentation Strategy:

TheMess as a team, will make sure that all code will be documented using GitHub and will notify other teammates via text or Discord. Using GitHub as our version control will allow other teammates to easily access files. On top of this function, we also have a website where the Configuration Management Specialist will post a weekly blog of what has been happening inside and outside of the project. With this approach, the team will be able to maintain the code as private while being able to inform the public what to expect from the project and any new features that may be coming out.

5. Risk Analysis

Risk Rating Scale:

Probability:

1. **Low**
2. **Medium**
3. **High**

Severity:

1. **Moderate**
2. **Critical**
3. **Catastrophic**

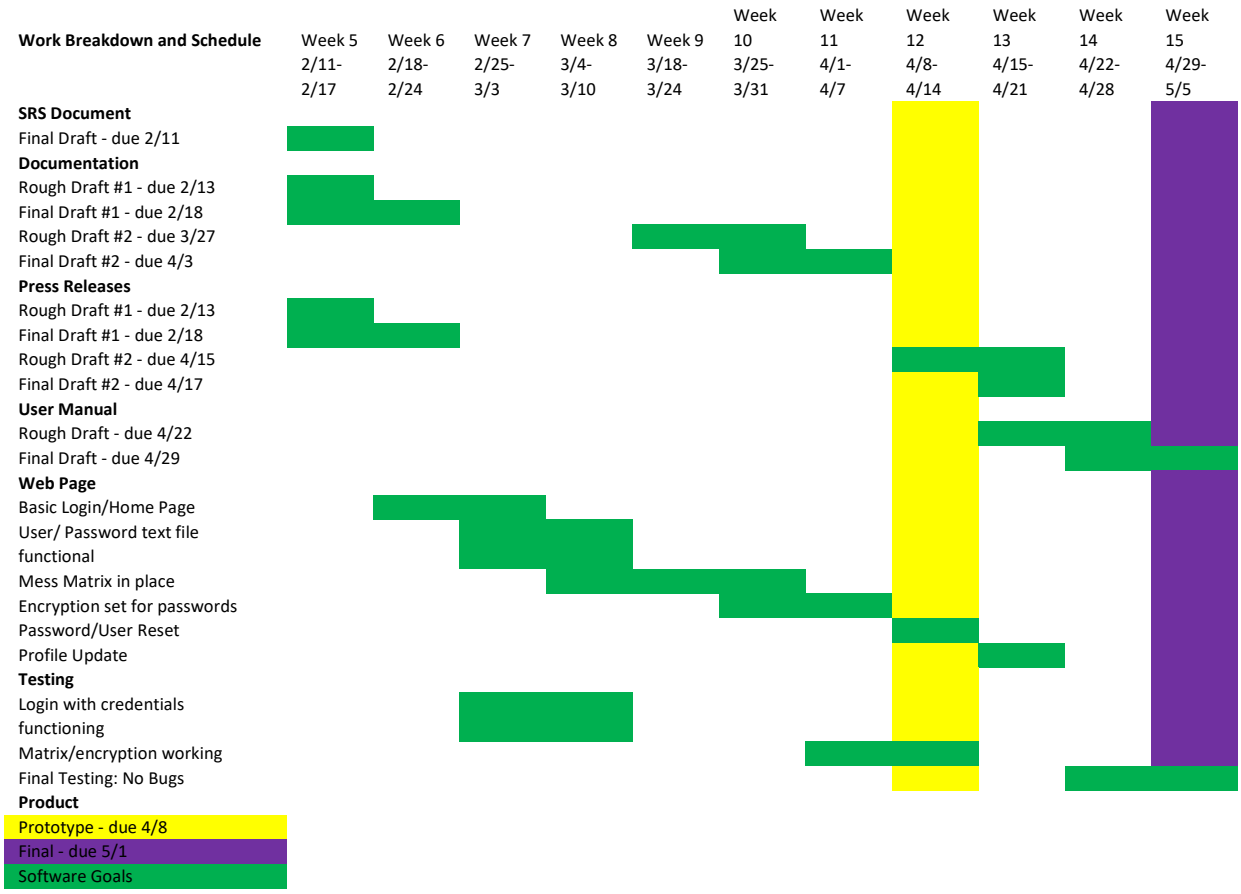
Hazards:

1. **Team member illness**
    a. **Probability:** Medium
    b. **Severity:** Moderate
    c. **Solution:** Train all team members on subject in case someone else needs to take over.
2. **Lost data**
    a. **Probability:** Low
    b. **Severity:** Catastrophic
    c. **Solution:** Use GitHub to its fullest extent to minimize risk of losing data.
3. **Feature creep**
    a. **Probability:** Medium
    b. **Severity:** Moderate
    c. **Solution:** Make sure everything is in working order with no bugs before we add more features.
4. **User Input Error**
    a. **Probability:** Medium
    b. **Severity:** Moderate
    c. **Solution:** Have lots of validation and error checking.
5. **GitHub crashing**
    a. **Probability:** Low
    b. **Severity:** Catastrophic
    c. **Solution:** Have back up files on our computers.
6. **Communication with team members**
    a. **Probability:** Low
    b. **Severity:** Catastrophic
    c. **Solution:** Have weekly meetings and a group text in case of emergency.
7. **Not enough time to finish working version of Prototype/Final project**
    a. **Probability:** Low
    b. **Severity:** Catastrophic
    c. **Solution:** Stay on task with our Work Breakdown Schedule and Project Deadlines.
8. **Product not what client wants**
    a. **Probability:** Low
    b. **Severity:** Catastrophic

    c. **Solution:** Have consistent meetings with client to make sure expectations are met.

9. **Documentation Issues**
    a. **Probability:** Medium
    b. **Severity:** Moderate
    c. **Solution:** Using version control we will be able to access an older working file to fix our mistakes.
10. **Test Cases Missing**
    a. **Probability:** Low
    b. **Severity:** Moderate
    c. **Solution:** Debug and test the application many times before moving onto the next part of the project.

6. Work Breakdown and Schedule



7. Architectural Design

Data Flow Diagram: