CS673 Software Engineering

Team 2 - PassMan

Project Proposal and Planning

Team Member	Role(s)	Signature	Date
Kayla Bayusik	Security	*Kayla B*	05-17-21
Zhaowei Gu	Quality Assurance		05-16-21
Alexander Dewhirst	Design / implementation	*ABD*	05-16-21
Andrew Klimentyev	Configuration	*AKlim*	05-17-21
Francis Pulikotil	Team leader	*fxp*	05-16-21

Revision history

Version	**Author**	**Date**	**Change**
0	Group	05/17	Initial Version
1	Group	05/31	Iteration 1: Updated risk, timeline, non-functional and security requirements.
2	Group	06/14	Iteration 2: Added security and other sections to document outline. No other major changes.
3	Group	06/22	Iteration 3: Updated timeline, and iteration 3 metrics in QA plan.

1. Overview

(Please give an overview of your project. It should include the motivation, the purpose and the potential users of the proposed software system.)

Our project, PassMan, is a web application that allows users to store their passwords.

A lot of people use the same passwords on multiple websites. This is a problem because of the many leaks that occur each year. If hackers are able to recover a password for one website, they can simply try it with other websites and thus gain access to multiple accounts of the user.

To reduce such damage, it is advisable to use unique passwords on every website. These unique passwords should also be sufficiently strong to maximize security. A stronger password will be longer, and contain a mix of letters, digits, and special characters to make it more unpredictable.

Even the average person will have tens of accounts with different websites, and it becomes humanly difficult to keep track of so many strong passwords. This is where a password manager is useful - you keep all your strong passwords secure in the password manager application, and the password manager itself is secured with a single strong “master”

password. Our project PassMan is a password manager application whose goal is to store passwords securely, while being easy to use.

2. Related Work

(Please describe any similar software systems that you have found through the online research, and the differences between your software and those software systems.)

Google Password Manager:
Google password manager is a very powerful tool built with google chrome, it allows you to save passwords to different websites, even different accounts for the same website. It also Automatically sign in to websites using stored credentials, And finally it automatically asks you to update and save new passwords whenever you enter some new password.

1Password:
1Password is a password management service with both web and app products that allows you to store passwords securely. 1Password uses a master password or a fingerprint scan to unlock and view the secure credentials.

LastPass:
LastPass is a free password manager, with premium platforms available for corporate use, that stores encrypted passwords online. The standard version of LastPass comes with a web interface, but also includes plugins for various web browsers and apps for many smartphones, but all three interfaces involve the user logging in with a master password once to enter their password vault. The plugins allow LastPass to directly fill in username and password input boxes using the data stored in the vault for that site.

3. Proposed High level Requirements

a. Functional Requirements

i. Essential Features:

- As a user, I want to be able to register for PassMan, so that I can utilize the excellent features of PassMan.
- As a user, I want to be able to login to PassMan, so that I can be confident that my passwords are stored securely.
  As a user, I want to be able to logout of PassMan, so that I can keep my passwords safe when using any computer.
- As a user, I want to know my passwords are encrypted, so that I can be confident that my passwords are stored securely.
- As a user, I want to access PassMan on the internet, so that I can have access to my passwords in a user interface.
- As a user, I want to be able to generate passwords, so that I can have passwords that are very difficult to brute force.

ii. Desirable Features:

- As a user, I want to be able to import passwords, so that I can transfer passwords from another product to PassMan.
- As a user, I want to be able to export passwords, so that I can keep my passwords offline in case of emergencies.
- As a user, I want to be able to copy my passwords, so that I can quickly use them in a browser.

iii. Optional Features:

- As a user, I want to be able to create a business account, so that my employees can manage their passwords with PassMan.

iv. Nonfunctional Requirements

- As a user, I want the PassMan application to respond within a maximum of 1 second for every action I take, so that I am not inconvenienced/frustrated when I use it.
- As a user, I want the PassMan application to not corrupt my data under any circumstances, so that I can rely on my data being available to me when I need it..
- As a user, I would like to access the PassMan application as a web application, so that I can connect to it easily from any computer without having to install special software.

v. Security requirements

- As a user, I want the PassMan application to store my master password appropriately and safely in the PassMan database, so that my account and the information within remain secure.
- As a user, I want the PassMan application to provide me with master password requirements, so that the password I create to enter the password vault is not easily guessed or brute forced by a potential hacker.
- As a user, I want the PassMan application to store my password vault items, namely the item passwords, appropriately and safely in the PassMan database while also keeping them accessible, so that I can view and edit my stored passwords when I choose.
- As a user, I want the PassMan application to only display my stored data when I am logged in to the application and to stop displaying my stored data when I log out, so that my stored passwords are available and visible only to me.

4. Management Plan

a. Process Model

(Please describe your software process model, e.g. ?)

For PassMan, we're using the Agile process model. Agile is defined by continuous iterations of the development process. The team will develop the product over

several incremental parts, with each part designed to be small enough to be completed in a week. In each iteration, a small set of features will be implemented and released.

b. Objectives and Priorities

(Please describe your project objectives with highest priority first. Project Goals can include but not limited to complete all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality, etc)

- Allow users to store passwords securely, and retrieve them easily.
- Provide a clean and simple interface for users to interact with.
- Provide a strong password generation function
- Complete all other essential features.
- Keep the quality of the product high.
- Try to eliminate all known bugs and issues.
- Software will be deployed locally at first for personal use.
- (Optional) Deploy the software to the cloud (hosting) for multiple users to use.

c. Risk Management (need to be updated constantly)

(Please write a summary paragraph about the main risks your group identified and how you plan to manage these risks. Then use the separate google sheet for detailed risk management. The template is provided in the same folder with this file. Please provide the link to the sheet.)

We lost our team leader, Luke Gehron early on in the project planning phase. We are now down to 5 team members. The backup team lead was assigned to be the new team lead, so we don't have a backup team lead anymore, and some work will be redistributed among team members.

There is a very slight risk that one or more of our team members will be unable to work due to becoming infected with the global pandemic (COVID-19). This hasn't happened yet, but if it does, we will have to adjust the project scope and deliverables as necessary, and possibly assign extra work to the remaining team members.

Risk Management Sheet Link:
https://docs.google.com/spreadsheets/d/1RsJj_nPmRtgdhDiXhbwmpMzhXNNRd5ft

d. Monitoring and Controlling Tools and Mechanisms

We will use the following tools to facilitate group communication and monitor the project progress.

i. Pivotaltracker Link: https://www.pivotaltracker.com/n/projects/2498530
ii. Slack Link: https://bumetcs673ols-0g57838.slack.com/archives/C021UMF9RMX
iii. Github Link: https://github.com/BUMETCS673/BUMETCS673OLSum21P2
iv. Zoom meeting Link: (multiple)
v. Weekly meeting time: (flexible)

e. Timeline (need to be updated at the end of each iteration)

Iteration	Functional Requirements(E/D/O)	Tasks	Estimated/real person hours	Presentation Recording Link (5-10 minutes)
0	n/a	Define project requirements, planning, design, tech stack.	/37.6	https://drive.google.com/file/d/16YRWHGXo_CQIG3qfCwVxM88TwiixIDjJ
1	Implement application login and signup pages. Implement security features: password hashing, sessions.	Meet to discuss requirements, features, and security. Implement the features discussed. Create a presentation.	/73.7	https://drive.google.com/file/d/1nd5xL5jWE0nWiMSJBFzLxFBzmnwJktj3
2	Implement adding, editing, and deleting items from the user's vault; strong password generation; automated testing.	Meet to discuss and decide features to be implemented. Implement features discussed. Create a presentation.	/78.5	https://drive.google.com/file/d/1tSpokQ14BD5_K5J7G3oC6a77H7STRnbm
3	Implement CSRF protection. Add more automated tests. Implement blueprints for app pages. Implement encryption of the user's vault. Setup/add code coverage reporting. Final project presentation.	Meetings to decide which features to implement, and which tasks to assign to which team members. Prepare and present the final presentation.	/27.5	https://drive.google.com/file/d/1esFtyYK1d-M69GT1YHT4YWGU8bdhs21P

5. Quality Assurance Plan

a. Metrics

i. Definition (e.g. define what metrics will be used, how to keep track of metrics, and how to analyze the metrics for process improvement. Two types of metrics should be included: product metrics and process metrics. Particularly include product complexity (LOC, # of files, # of classes, # of methods, cyclomatic complexity, etc.) cost (in terms of man hours), defect and defect fix rate, user story points, etc.

Product metric

Metric ID	Metric Description	Process Improvement
pd-1	Open bug count	Analyze the bug and communicate with developer for improvement
pd-2	Closed bug count	Use as an indication how much bug have closed
pd-3	Product complexity	product complexity (LOC, # of files, # of classes, # of methods, cyclomatic complexity, etc.) from 1- 10

Process metric

Metric ID	Metric Description	Process Improvement
pc-1	user story accepted	Reflect team productivity, if decreases additional time need to be spend to compete the before new stories stars
pc-2	Testing coverage	Check if we have tested all of the code from 1- 10

ii. Results (to be completed at the end of each iteration),

Metric ID	Iteration 0	Iteration 1	Iteration 2	Iteration 3
pd-1	0	0	0	0
pd-2	0	0	0	0
pd-3	1	6	8	10
pc-1	0	2	8	11
pc-2	0	1	10	10

b. Standard

(e.g. documentation standard, coding standards etc.)

Design Documentation:

1. SPPP: To continually refine the scope of the project
2. Progress Reports: Tracks individual contributions to the project and informs management plan
3. Risk Management: Continually tracks and addresses risks
4. Meeting Minutes: Reference for team members responsibilities and focusses in a given week
5. SDD: Software Design Document, focus on technical scope of the project.
6. STD: Software Test Description, provided by QA leader, tests' scope that cover main functionality of the project.
7. Use linting to check your source code for programmatic and stylistic errors.

Coding standards:

1. Classes and methods should be thoroughly documented with their desired functionality and purpose.
2. Coding guidelines please follow the one established by Pylint and PEP8
3. JavaScript please follow the Airbnb's style guide on GitHub
4. CSS please follow the Airbnb's style guide on GitHub

c. Inspection/Review Process

(e.g. describe what is subject to review, when to conduct review, who do the reviews and how ?)

Code Review:

Our team will use CI/CD pipeline that does not allow code to be merged to the main branch without going through a pull request process. During the pull request code will be reviewed by QA leader, team leader, and any other team members who is also a stakeholder in the work. after the pull request is viewed ideally in a group setting then the code could be pushed in the main branch.

Documentation Review:

Documentation review is to be conducted at the end of each iteration by everyone in the group to make sure everything looks good.

d. Testing:

(e.g. who, when and what type of testing to be performed? How to keep track of testing results?)

A separate document about testing results should be linked here.

Testing Results Doc Link:

Unit testing:

Every developer will write unit testing code, failure cases will be reported at the end of iteration by the QA leader.

UI testing:

Done by any team member at the end of each iteration, failure cases will be reported at the end of iteration by the QA leader.

Other types of testing (integration, functional, non-functional):
These testing should be implemented at the end of each iteration by the QA leader, to make sure that product meets the requirements.
Unit Tests for the API:
Configuration leader akliment@bu.edu setting up unit tests
E2E tests with selenium for the full flow:
QA leader @zwgu@bu.edu setting up E2E tests

e. Defect Management

(e.g. describe the criteria of defect, also in terms of severity, extent, priority, etc. The tool used to management defect, actions or personnel for defect management)
Management tool: PivotalTracker
Severity:

1. Critical: Issue causes immediate crashes and prevents an app from running. Fixing this type of defect will be given the highest priority.
2. High: Intermittent crashes and/or major issues with the usability of the main features of the app. This level of severity will need to be addressed prior to the completion of the iteration.
3. Medium: App continues to run but feature testing yields unexpected or incorrect results. This level of defect will not prevent the completion of the iteration but will instead be moved to the backlog for the next iteration (if it is not fixed in the current iteration).
4. Low: Minor impact to usability, core app features unaffected. Users can work around the issue. Like medium severity defects, these defects will not prevent the completion of the iteration.

Priority:

1. High: Critical defects. Need to be fixed as soon as possible, ideally within a day and before the end of the iteration.
2. Medium: High/Medium severity. Depending on severity, this may need to be addressed before the end of the iteration.
3. Low: Low severity. Can be added to the backlog.

6. Configuration Management Plan

(For more details, please refer to SCMP document for encounter example)

a. Configuration items and tools

i. Items:
SPPP, Meeting Minutes, Progress Report, Iteration 0 presentation
ii. Tools:
Git, Github

b. Change management and branch management

Our team will use the following branch structure:
Main

Username-feature
Lab#

c. Code commit guidelines

Code will be committed to a new branch with established naming conventions.
When the code is ready to be merged into master, a pull request will be created and the project leader or configuration leader will sign off on it.
The pull request author is then responsible for making the merge.

d. Integration and deployment plan

The program will be deployed locally on the user's machine

7. References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard.)

1. Pylint https://pylint.org/
2. PEP8 https://www.python.org/dev/peps/pep-0008/
3. Testing Flask Applications with Pytest https://testdriven.io/blog/flask-pytest/
4. Testing Flask Applications https://flask.palletsprojects.com/en/1.1.x/testing/