# Password manager with P2P Synchronisation

---

# Testing Report

---

**Student Name:** Dean Lynch

**Student Number:** 15359921

**Supervisor:** Brian Stone

18/05/19

**Abstract:** This is a unique application that combines password database management and synchronisation into a unified application. The application reads from a locally stored password database file, which can then be synchronised across a users devices using a peer-to-peer connection. This eliminates the need to expose the password database to third-party servers or cloud storage services, as well as saving the user the need to manually synchronise the database or manage their own self-hosted solution.

# Table of Contents

1. Unit Testing
2. Continuous Integration
3. Heuristic Testing
4. User Testing
5. Known Issues

# 1. Unit Testing

As this is a password manager, it is expected to be a secure application. For that reason it is very important to thoroughly test every component of the application. For any secure application I would expect the maintainer to be as transparent and open about the flaws in the application as possible. For this reason I want to say that the level of testing completed so far for this application is not up to a high enough standard for anyone to consider using this application with their own personal information. Good and thorough testing takes time and experience. While sufficient testing was completed for the password management component, and other utility classes, the test coverage for both the server/client components and JavaFx GUI isn't high enough for an application of this type.

JavaFx testing was a lot more difficult than I expected it to be. Unfortunately when I discovered this I was already too far into development to change technologies. When testing with JavaFx you need to have an instance of JavaFx running in order to access the different classes and objects for testing. I was not able to get this to run successfully, even for a simple tree builder class that isn't part of the GUI, but used one of the classes from the JavaFx library. While the interface testing is not overly important from a security point of view (as they are mostly concerning certain use cases of the application), I'm still disappointed not to have included them.

While I have tested some aspects of the server and clients to best of my ability within a the time available, I don't think the testing for these components is sufficient enough for a secure application. I had a lot of difficulties trying to unit test the server and client components of the application. I have no experience testing java sockets, let alone SSL sockets. I spent lots of valuable time researching and trying to learn how to do this correctly, but I couldn't find any consistent answers for my specific case.

Before I had implemented SSL sockets, and was only using standard Java sockets, I asked for help on StackOverflow. After a waiting a while, I eventually received some useful information on how to carry this out. Unfortunately by this time I had already started implementing SSL sockets in place of the regular sockets, which rendered a lot of the feedback I had gotten from StackOverflow useless. To add to this, the server/client components use logging to report any exceptions caught in the process, making it even harder for me to test as I could not simply look for the exceptions thrown in these cases.

# 2. Continuous Integration

Continuous Integration (CI) works to integrate code sent to a repository. New code is sent in a push request, which triggers a pipeline to build, test, and validate the new code before applying the changes in the repository.

I created a CI pipeline for my project using the GitLab CI platform available to us through DCU GitLab. I have used this throughout my project starting in the early stages of development.

The pipeline is ran every time a push is made to the project  repository. The CI pipeline has 3 stages, as configured in the .gitlab-ci.yml file in the root of the project directory. They are as follows:

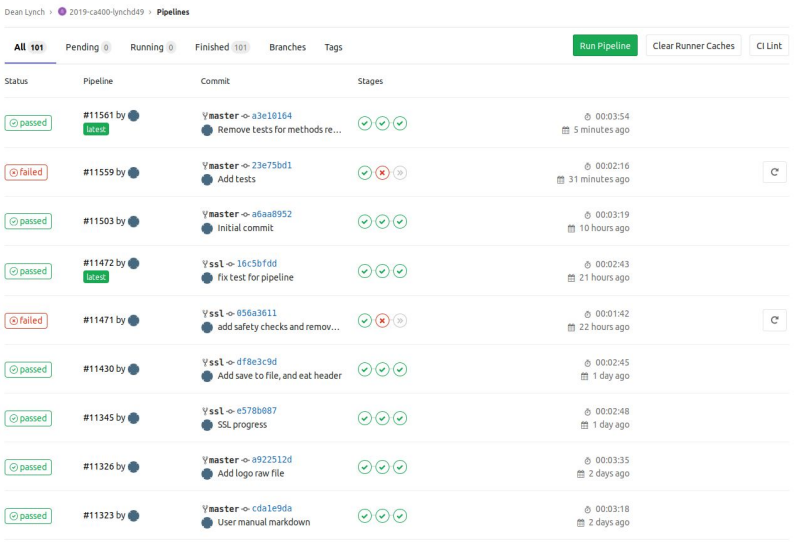
1. **Build:** The project was compiled using the ‘mvn clean compile’ command
2. **Test:** The full suite of tests written for the project is run using the ‘mvn clean test’ command.
3. **Run:** The project is run and packaged using the ‘mvn clean install command.

| .gitlab-ci.yml snippet | Screenshot of CI pipeline timeline |
| --- | --- |

```yaml
build:
  stage: build
  script:
    - mvn clean compile

test:
  stage: test
  script:
    - mvn clean test

run:
  stage: run
  script:
    - mvn clean install
```



# 3. Heuristic Testing

## Nielsen's Heuristics

**Simple & Natural Dialog:**

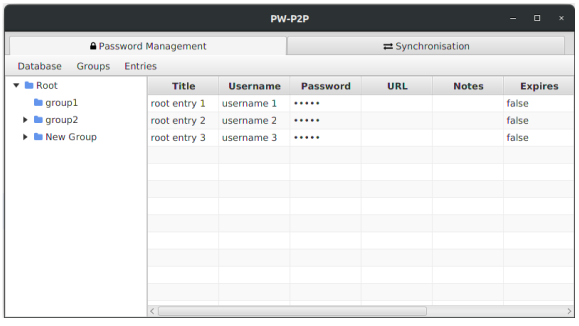
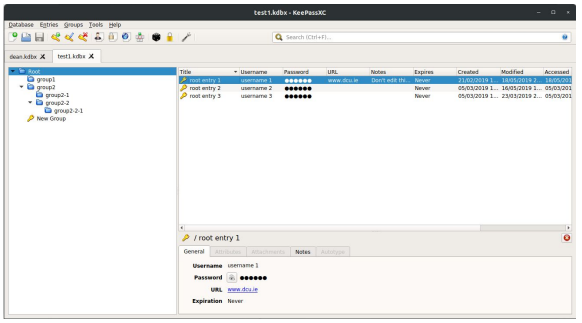
All dialogue with the user on the site is kept to a minimum. Dialog in the form of popups on the site is written in simple language.

**Speak the User's Language:**

No complex language is used for the different dialogues in the application. However, some technical language/knowledge is needed to understand the database layout  and composition as well as knowledge of basic networking terms. This means that some knowledge of the computer networking is necessary when using the application.

**Minimise the user's memory load:**

The application functions and is laid out in a very similar way to other password managers, so any users of these programs would not find it difficult to begin using this application. The pages are kept to a minimum and is simply split into two main pages password management, and synchronisation.

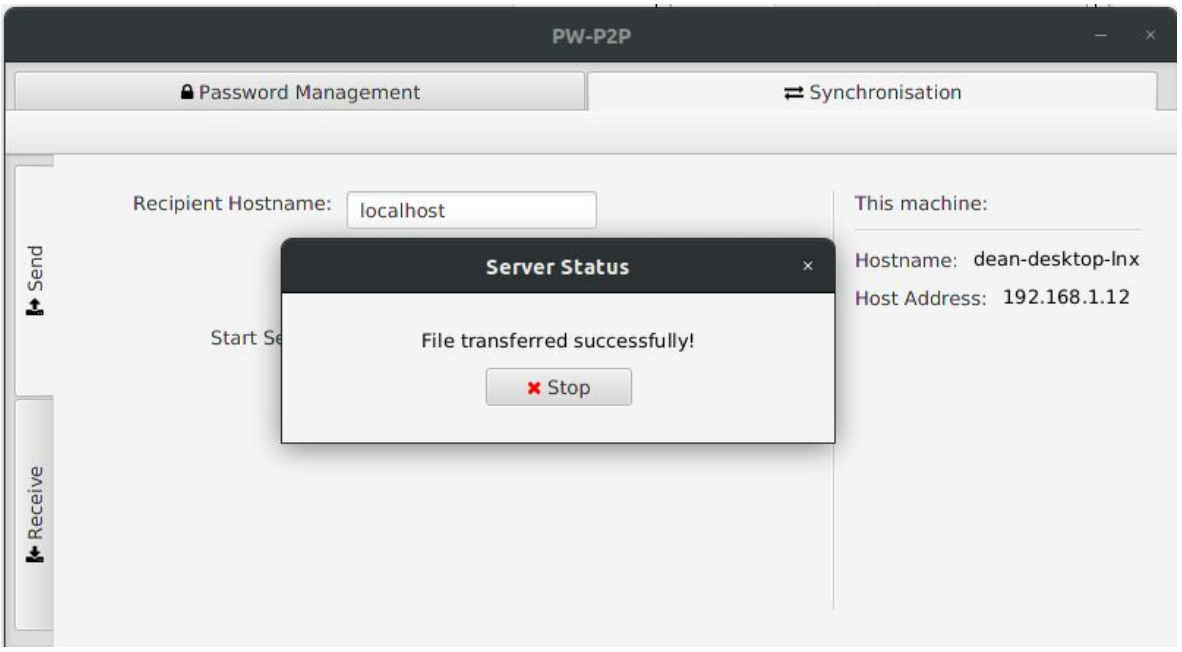
| This application | KeepassXC |
| --- | --- |
|  |  |

**Consistency:**

All language in the application is consistent in that it is kept simple as mentioned previously. The same buttons are reused from the same source in the code to ensure this. The style of the application is also kept consistent with the style of the operating system, as it is written using JavaFx and will conform same style as other native programs for the operating system it is running on.

**Feedback:**

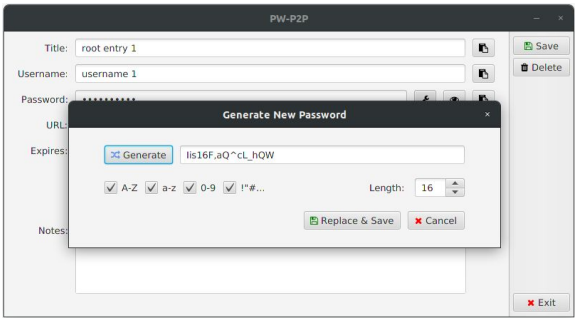
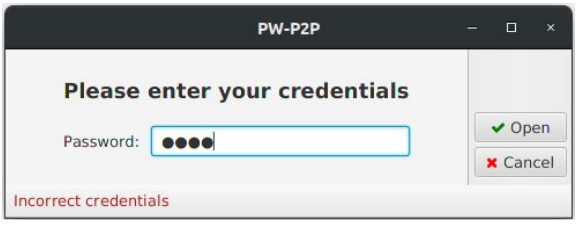
When a file is successfully transferred a popup dialog will be displayed on both devices indicating the success.

| Feedback shown when a successful file transfer is made |
| --- |
|  |



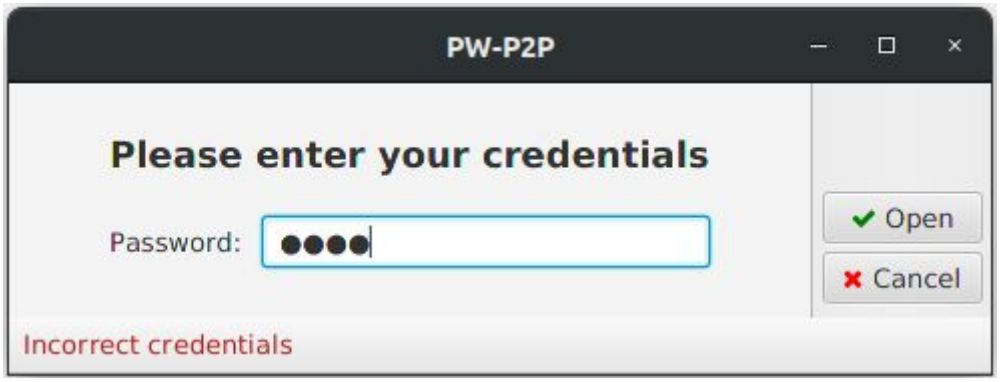
**Clearly Marked Exits:**

For all pop up dialogs and in the two entry pages of the application (file selection & credential entry) there are clearly marked exit buttons.

| Marked exits in Entry view<br>& password generation | Marked exit in credential entry |
| --- | --- |
|  |  |

**Good Error Messages:**

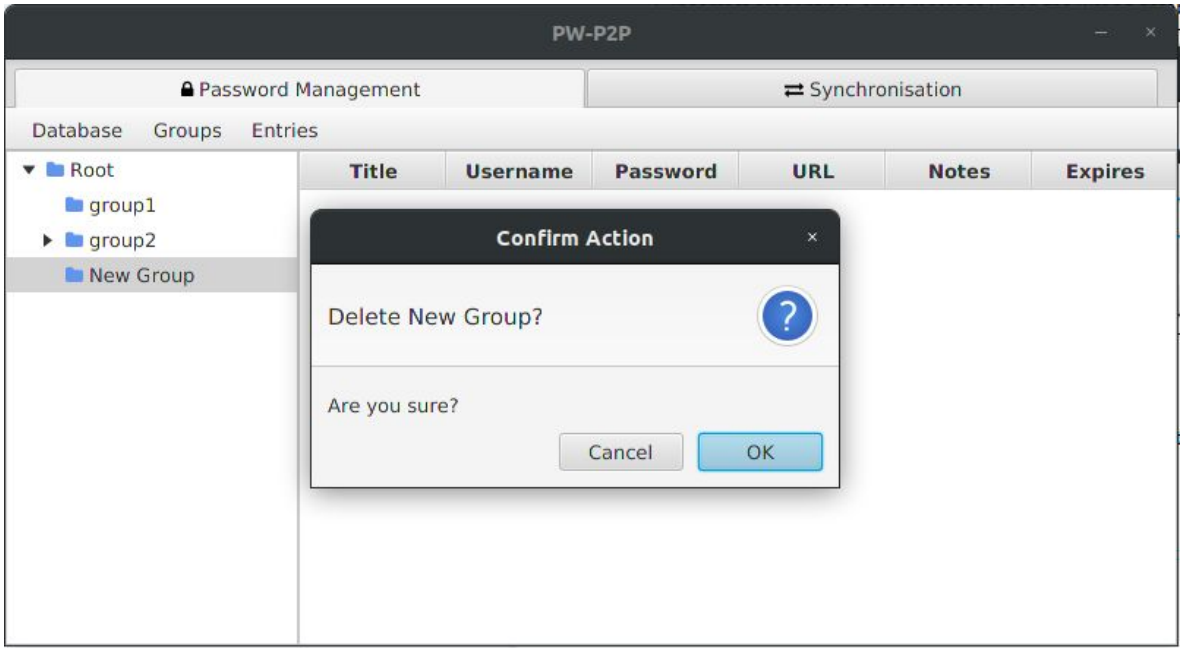
When there is an error accessing a database, or there is an error when transferring files between devices, it is clearly displayed in the application, or in a popup dialog.

| 'Incorrect credentials' error message when the wrong database credentials are entered |
| --- |
|  |



**Prevent Errors:**

When doing operations such as deleting an Entry or Group, the user is first asked to confirm their action before the operation is carried out.

| Confirmation dialog asking the user to confirm the deletion of a group |
| --- |
|  |



**Help and Documentation:**

Users have access to the user manual for the application in the project repository.

# 4. User Testing

## Section 1: Password Managers

The first section of user testing contained questions about the participants knowledge of password managers to get an idea of their current understanding of them.

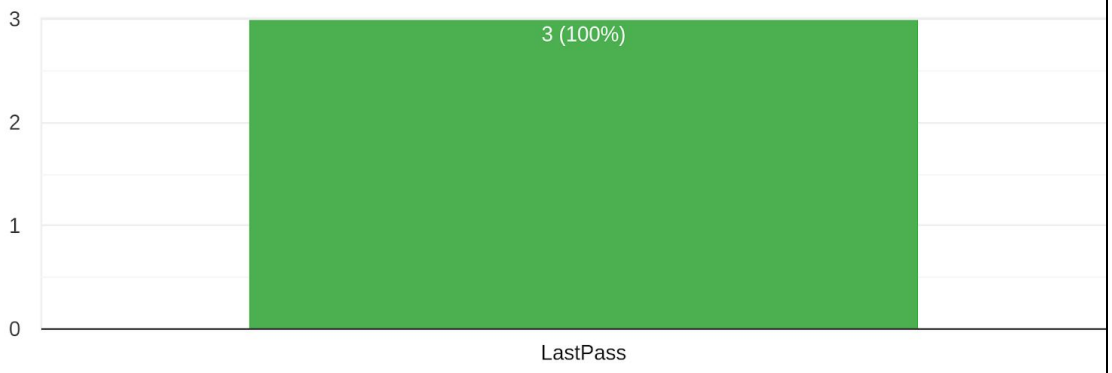
While these answers didn't help directly in the development of the application, they helped me to get a better understanding of the how people perceive password managers. This is helpful for any future questions I might get as I have a better understanding of the other side. It was also very interesting to see how many people changed their mind about their online privacy/security after asking them to visit [haveibeenpwned.com](haveibeenpwned.com).

| Question 1 |
| --- |
|  |

## Question 2

If you answered 'Yes' above, what password manager do you currently use?

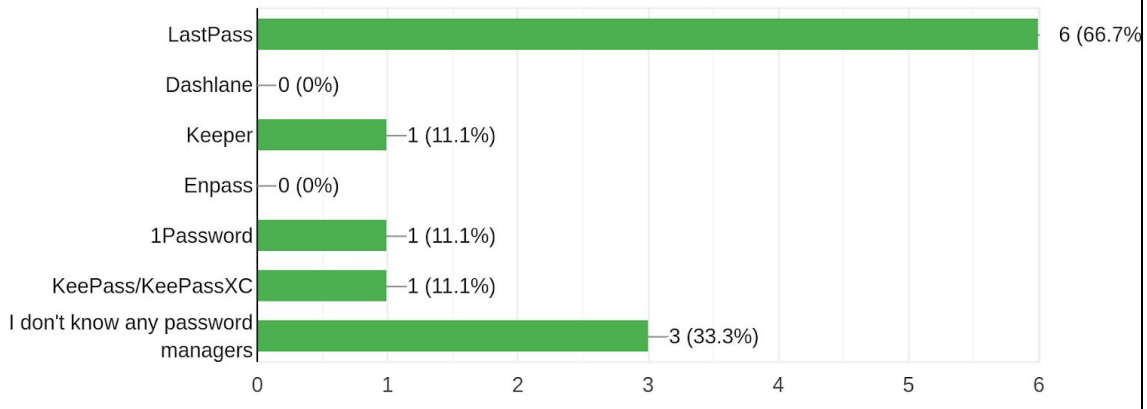
3 responses




## Question 3

Which of these password managers have you heard of?

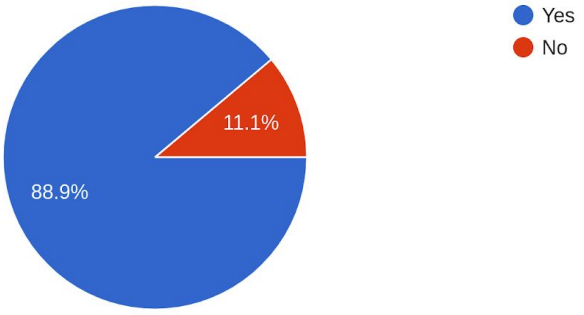
9 responses



## Question 4

Are you cencerned about your online security?

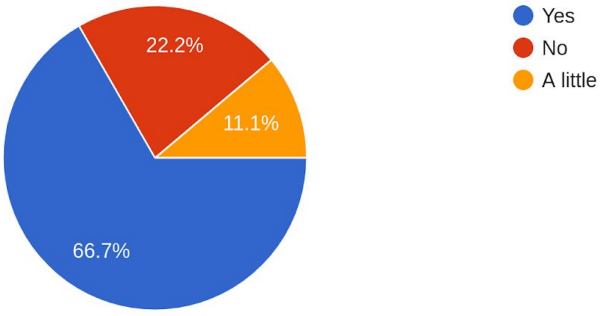
9 responses




## Question 5

Are you concerned about your online privacy?

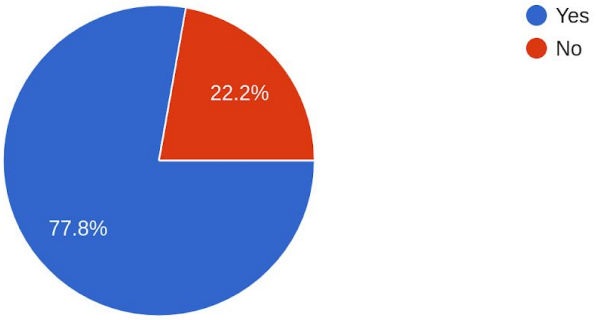
9 responses



## Question 6

Have you had an online account compromised before?

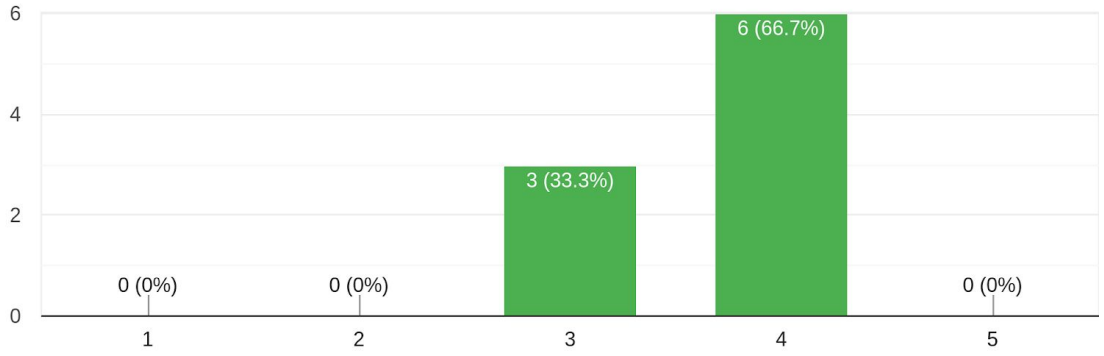
9 responses

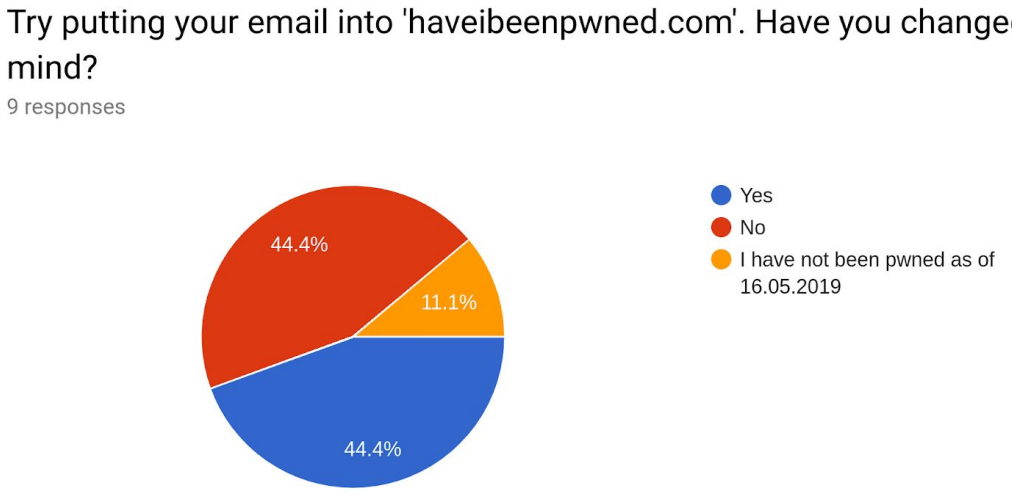



## Question 7

How concerned are you of having any of your online accounts compromised?

9 responses



| Question 8 |
| --- |
|  |

*Note:* In hindsight this last question should not have been mandatory and should only have asked users to complete it if they had answered negatively in the previous question. Regardless, there was still a very high number of people who said they changed their mind.

## Section 2: Functionality Feedback

This section was filled out by the participant after they had the opportunity to use the application. They were asked to complete some use cases in the application and then give their feedback.

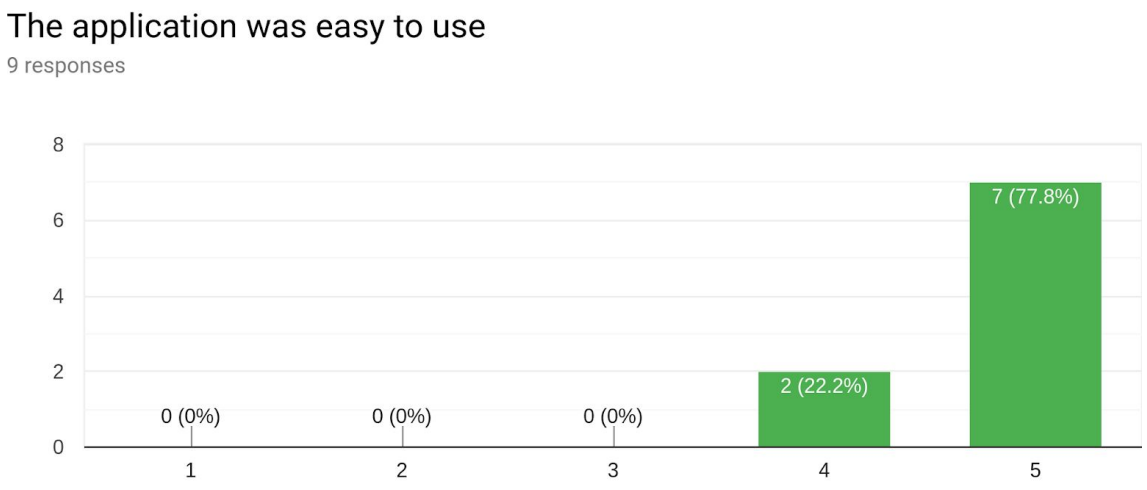
Unfortunately there was very little feedback given for any use cases and the little feedback given was not of any aid in improving the application. This was unfortunate as I don't think the application is perfect and certainly has usability issues that it needs to be improved. If more time was available between having a working application and the deadline, perhaps I could have been able to get more participants and get more valuable feedback.

## Section 3: General Feedback

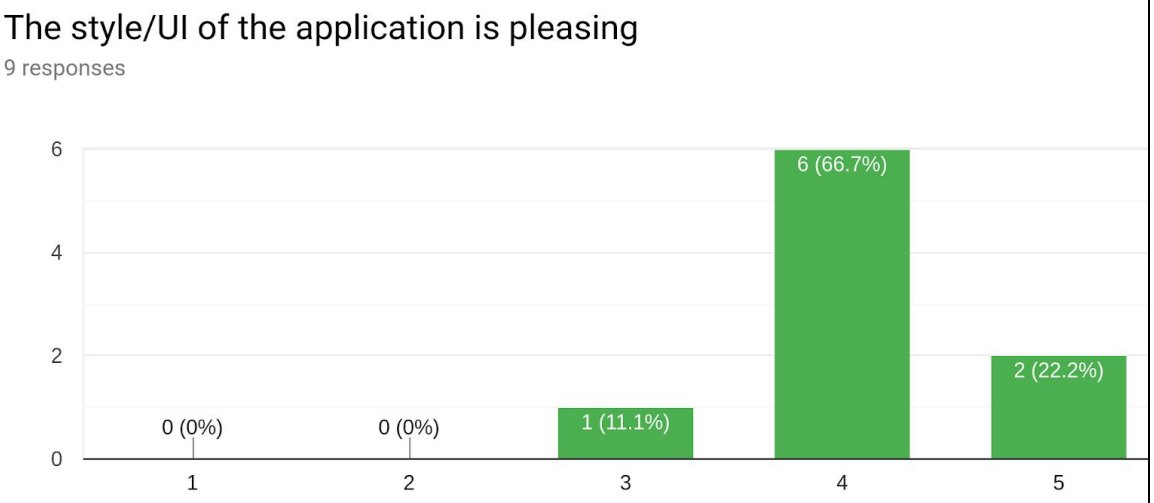
This section asked the participant some basic questions about their general thoughts on the application.

The positive feedback was nice to see and I'm glad that the participants enjoyed using the application. I do feel however there was a slight bias towards the application due to the fact that most of the testing was done with family and friends. Again, if more time was available I would have liked to do more user testing outside friends and family to hopefully get less bias and more valuable feedback.

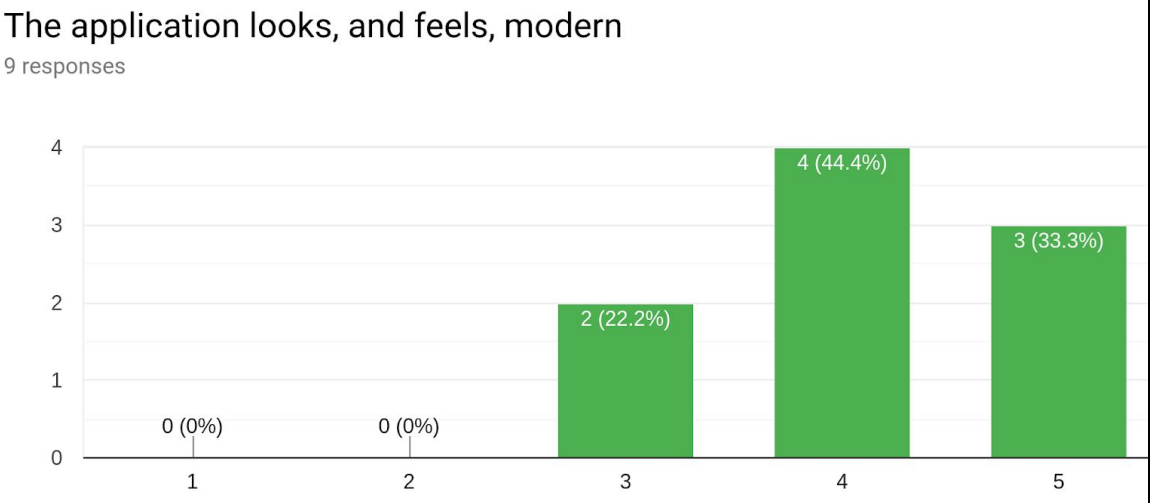
While it could be down to the previously mentioned bias, it was also quite concerning to see the number of people who would be happy to use, or recommend to other people, the application in its current state. I think most people are very naive about security in general, and for such an important security tool, people should be more aware of the background research that they should be doing into an application that they entrust all of their passwords and login information with.

| Question 1 |
| --- |
|  |

Question 2



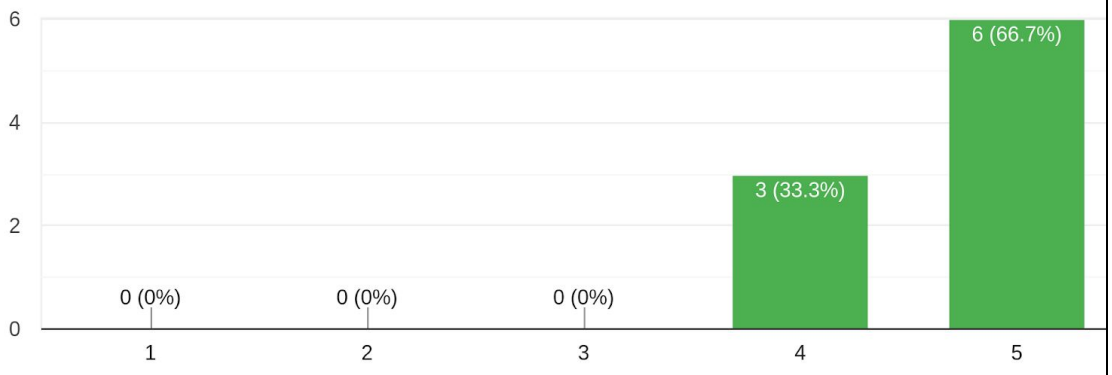

Question 3



## Question 4

The application does what it is intended to do

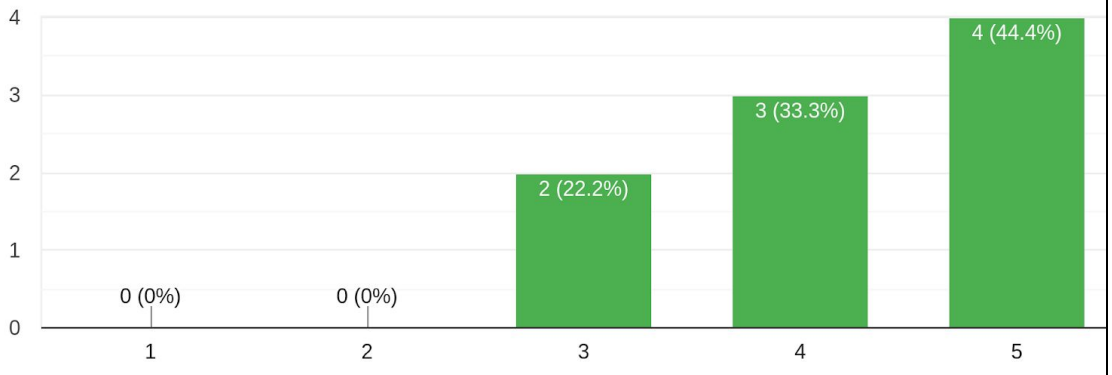
9 responses

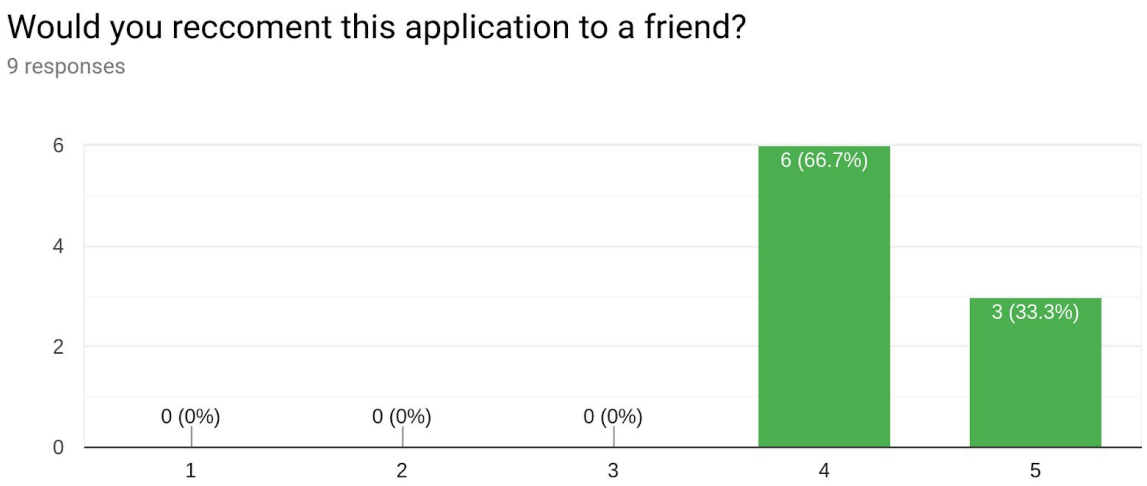



## Question 5

Would you consider using this application to mange your passwords?

9 responses



| Question 5 |
| --- |
|  |



# 5. Known Issues

| **Issue** | **Information** |
| --- | --- |
| JavaFx testing | JavaFx testing was a lot more difficult than I expected it to be. Unfortunately when I discovered this I was already too far into development to change technologies. When testing with JavaFx you need to have an instance of JavaFx running in order to access the different classes and objects for testing. I was not able to successfully get this to work, and that's before I'd even considered how I would get these tests to run successfully inside the CI pipeline. |
| Packaging | While I was able to package the application normally as a .jar file, this was of no use as it would not run successfully due to the differences with a JavaFx application. I spent a lot of valuable time trying to successfully package the file as a JavaFx jar, but I kept getting the same generic 'MojoExecutionException'. |
| Thorough server and client testing | I have no experience testing java sockets, let alone SSL sockets. I spent days researching and trying to learn how to do this correctly, but I couldn't find any consistent answers for my specific case. While I have tested some aspects of the server and clients to best of my ability within a short time, I don't think the testing for these components is close to sufficient for a secure application. |
| MacOS Testing | While the application may work on MacOS, there has been no testing done on the platform. |