FUNCTIONAL SPECIFICATION

Password Manager with Peer-to-Peer Synchronisation

Dean Lynch
15359921

0. Table of contents

Contents

1. Introduction

1.1 Overview

I plan to develop a password manager that reads a locally stored database, which can be easily synced across devices using a peer-to-peer connection.

A password manager is a program used to store, manage and generate passwords. There are two common implementations of password managers:

- An online service offered through a web portal:
    - Users connect through a web page or browser extension
    - The database is stored online, i.e. not locally on the user's device.
    - User only must remember a username/password, they can access their database from most devices.
    - Examples: LastPass, Keeper

The problem with password managers that implement online/cloud database storage is that the user has no control over how or where the password database is stored.

- A locally installed application, with a locally stored database.
    - Users run an application locally on their device.
    - The application reads a locally stored database.
    - User must handle synchronization of their database themselves or host it on a server provided by the password manager.
    - Examples: Keepass2, Enpass

The problem with a locally installed application that the user is left responsible for keeping the database synced across multiple devices.

I intend to create a solution that will provide the locally stored database as well as offering a way of syncing the database between devices without the use of additional software or a third-party server or cloud storage.

1.2 Business Context

The application is primarily focused at privacy and security conscious users but may have use in enterprise with some changes to how the synchronisation works between devices.

This application will be used by users who are conscious about their personal privacy - people who want to have full control of their how and where their data is stored.

It could also be potentially used in businesses to share passwords companywide or across departments. A specific synchronisation implementation with enterprise in mind would have to be included in the application to make this viable. The scope of this feature will be covered in part 2.5 of this functional specification.

1.3 Glossary

- **Password manager:** A program used to store, manage and generate passwords.
- **Password database:** A file containing the users login information, but may also contain private information such as addresses, ID information etc.
- **Multi-factor authentication (MFA):** This is where a user must supply two or more factors to an authentication mechanism. An example of this would be a user supplying a password as well as a one-time-password to confirm a user's identity.
- **One-Time-Password (OTP):** Usually a one-use password generated by an authenticator app that is needed alongside a password to confirm the user's identity.
- **Authenticator app:** A mobile application such as Google Authenticator or Authy that supplies users with one-time-passwords.

2. General Description

2.1 Product / System Functions

Password management:

- **Generate a new password database:** The user will be able to generate new password databases to securely store their password information.
- **Create and Edit password entries:** The user will be able to easily create and edit password entries.
- **Organisation:** The entries created by the user can be split into different groups and categories to make it easier for the user to organise and find their password entries.
- **Generate passwords:** The user will be able to generate string and secure passwords when creating new or editing existing password entries.
- **Multi-factor authentication:** For users who are not satisfied with a single password to confirm their identity, the application will implement multi-factor authentication to add extra layers of security to the database.

Synchronisation:

- **Synchronise database between devices:** The user can synchronise their password database across multiple devices without the need for a third-party synchronisation program such as Syncthing or cloud storage providers such as Google Drive or Dropbox.
- **Pairing devices easily and securely:** The process of pairing two devices will be straight forward and intuitive. It will not require a vast knowledge of computer networking. The pairing process will also ensure that no unknown devices can pair with your own device without your explicit permission.

2.2 User Characteristics and Objectives

The perspective users of this application include people who are conscious about their personal privacy and want to have full control of their how and where their data is stored. It will mostly be aimed at people who already use a password manager regularly and are not completely satisfied with the implementation of the password manager that they currently use.

2.3 Operational Scenarios

2.3.1 Create a new database:

Use Case 1	**Open existing database**	
Goal in context	User wants to get access to the database	
Subject Area	Password Management	
Preconditions	The user has created/connected a database to the application	
Success End Condition	The user gets access to the database	
Failed End Condition	Invalid credentials or authentication will mean failure to gain access to the database.	
Primary Actors	User	
Secondary Actors	Multi-factor authentication provider	
Trigger	The database is found, and the user is presented with the authentication screen.	
Description	**Step**	**Action**
	1	User inputs credentials
	2	Database is decrypted
	3	User is brought to the list of password entries
Variations	**Step**	**Branching Action**
	2	Credentials must be valid. The database remains encrypted and an error message is shown
Extensions	**Step**	**Branching Action**
	2	Credentials consist of: 1. Password 2. Multi-factor identification 3. Key file

2.3.2 Open existing database:

Use Case 2	**Open existing database**	
Goal in context	User wants to get access to the database	
Subject Area	Password Management	
Preconditions	The user has created/connected a database to the application	
Success End Condition	The user gets access to the database	
Failed End Condition	Invalid credentials or authentication will mean failure to gain access to the database.	
Primary Actors	User	
Secondary Actors	Multi-factor authentication provider	
Trigger	The database is found, and the user is presented with the authentication screen.	
Description	**Step**	**Action**
	1	User inputs credentials
	2	Database is decrypted
	3	User is brought to the list of password entries
Variations	**Step**	**Branching Action**
	2	Credentials must be valid. The database remains encrypted and an error message is shown
Extensions	**Step**	**Branching Action**
	2	Credentials consist of: 4. Password 5. Multi-factor identification 6. Key file

2.3.3 Create an entry:

Use Case 3	**Create an entry**	
Goal in context	User creates an entry containing their login information	
Subject Area	Password Management	
Preconditions	User has access to the database	
Success End Condition	User creates a new entry	
Failed End Condition	A new entry is not created	
Primary Actors	User	
Secondary Actors		
Trigger	User clicks the “New Entry” button	
Description	**Step**	**Action**
	1	User is brought to the “New Entry” page
	2	User inputs their login information
	3	User selects the save option
	4	The new entry is added to the database
Extensions	**Step**	**Branching Action**
	2	Login credentials can consist of: 1. Username 2. Password 3. URL 4. Expiry 5. Notes

2.3.4 Edit an existing entry:

Use Case 4	**Edit an existing entry**	
Goal in context	User will edit an existing entry in the database with updated information.	
Subject Area	Password management	
Preconditions	User has access to the database There is one or more existing entries in the database	
Success End Condition	An existing entry is updated with new information	
Failed End Condition	The entry is not changed	
Primary Actors	The user	
Secondary Actors		
Trigger	User selects the “Edit Entry” button	
Description	**Step**	**Action**
	1	User is brought to the “Edit Entry” page
	2	User edits the information inside the entry
	3	User selects the save option
	4	The database is updated with the edited entry
Variations	**Step**	**Branching Action**
	3a	1. User selects the “cancel” button 2. The entry is not changed

2.3.5 Delete an entry:

Use Case 5	**Delete an entry**	
Goal in context	The user will remove an existing entry from the database	
Subject Area	Password Management	
Preconditions	User has access to the database There is one or more existing entries in the database	
Success End Condition	The selected entry is removed from the database	
Failed End Condition	The entry is not removed from the database	
Primary Actors	The user	
Secondary Actors		
Trigger	User selects the “Remove Entry” button	
Description	**Step**	**Action**
	1	The user will be prompted with a dialog box asking if they are sure that they want to remove the
	2	The user confirms that they want to remove the entry
	3	The entry is removed from the database
Variations	**Step**	**Branching Action**
	2	1. The user selects the cancel option 2. The entry is not removed from the database

2.3.6 Pair two devices:

Use Case 6	**Pair two devices**	
Goal in context	Pair two devices so that the database can be synchronised between the two devices.	
Subject Area	Synchronisation	
Preconditions	The two devices are connected to the same network and the application is running on both devices	
Success End Condition	The two devices are paired with each other for synchronisation	
Failed End Condition	The two devices are not connected with each other for synchronisation	
Primary Actors	The user	
Secondary Actors	Device A, Device B	
Trigger	User selects the "Pair New Device" button on device A	
Description	**Step**	**Action**
	1	User is shown a pop-up dialog on device A with the device ID and other connection information/options.
	2	User selects the "Pair New Device" button on device B
	3	User is shown a pop-up dialog on device B with the device ID and other connection information/options.
	4	User takes note of the unique device ID on both devices
	5	On device A, the user will find and select the device ID of device B from a list of devices available to pair
	6	The user will be asked to confirm that they want to pair with the device that they have selected
	7	On device B, a prompt will appear asking to confirm pairing with device A. Device A's device ID will be shown
	8	On device b, the user will confirm the pairing between the two devices.
	9	The two devices are now paired
Variations	**Step**	**Branching Action**

	5	1. The device ID of device B cannot be found. 2. The user must start again with step 1
	6	1. On device A, the user cancels pairing with the device 2. The devices remain unpaired
	7	1. On device B, the user cancels pairing with the device 2. The devices remain unpaired

2.4 Constraints:

Level of Security:
As this application will store and manage very important user information, a high level of security must be maintained. A lot of time will have to be invested into researching the levels of security that this application should adhere to.

Cancellation of CA4005 module:
I was very disappointed to learn that the CA4005 Cryptography and Security Protocols module has been cancelled this semester. Fortunately, I am still able to access the notes for this module. This means that I will have to learn about the topics covered in this module independently. This was frustrating as I had to take another module in its place, meaning that I will have to invest even more time covering this module on top of an already full schedule.

Time:
Time management is very important for this project. While we do only have 9 hours of allocated to module lectures and labs, the assignments given to us in the three modules are taking up a large amount of our time outside of the lectures. These assignments all carry high markings, so it is very important that time is also invented in them.
Over the Christmas period it is also important that time is invested into studying for the January exams. I also expect the second semester to carry similar time constraints, again with important assignments and exams to study for.

2.5 Scope

My goal is to create a product that will provide all the features necessary for simple password management and to provide peer-to-peer synchronization across devices. The password management features should include the creation/generation, editing and removal of password entries. The application should also include two-factor authentication as an extra level of security when accessing/securing the password database. This is the minimum viable product that I intend to create for this project.

Other ideas that I have considered to include in this application include:

- **OTP (One-time password) support and synchronisation across devices.** This is a feature I would love to include in the application, but it is unrelated to the password management features and would take a considerable amount of time to implement. I have also yet to research the feasibility of this feature - this is something I will investigate if I feel that I have the time to implement this feature.

- **Support for multiple database formats.** I would like to allow users to have the ability to use multiple database formats, but I feel that for this project and with the time constraints it is important that I focus on one format and get the implementation of the one format right before I attempt to allow to use of multiple formats.

- **Support for non-password entries.** I would like to include the support for entries such as addresses and identification card information.

Each of these features I would not consider to be part of the minimum viable product, which means that these features are not part of the core functionality of the application. The features outlined above will only be included as part of the application if there is the time available to implement them once the core features have been implemented successfully.

Enterprise-focused synchronisation is also an area I can see this project being used as outlined in the business context. However, due to the time constraints of this project I will not be able to ensure enterprise level security and testing. No company in their right mind should use a password manager that hasn’t gone through rigorous testing and auditing. Therefore, enterprise synchronisation features will not be in the scope of this project.

3. Functional Requirements

3.1 Password database creation

- **Description:** User must be able to create a new database with a password and two-factor authentication.
- **Criticality:** This is essential for users who don't have an existing database. The user will not be able to use the application without a password database.
- **Technical Issues:** Ensuring that the database is created in the correct format so that it is not at a risk of being broken or corrupted when in use.
- **Dependences with other requirements:** None.

3.2 Open existing password database

- **Description:** User must be able to open and access an existing database.
- **Criticality:** This is essential as users must be able to access their password database. If this does not function, users will not be able to access their stored passwords.
- **Technical Issues:** File must be unencrypted properly to ensure that the user has access to all his friends
- **Dependences with other requirements:** This is dependent on requirement 1, as a password database must be created first before it can be accessed.

3.3 Create Entry

- **Description:** User must be able to create a new password entry in the database
- **Criticality:** This is an important feature, without this requirement users will not be able to add to their database.
- **Technical Issues:** Saving a new entry to the database must be done correctly or the user may lose the entry indefinitely.
- **Dependences with other requirements:** This is dependent on requirement 2, as the user must have access to the database before a new entry can be added.

3.4 Edit entry

- **Description:** User must be able to edit an existing entry
- **Criticality:** This is an important requirement as users need to be able to update entries in cases where login information changes or passwords are compromised
- **Technical Issues:** It is important that the existing entry is replaced by the edited entry
- **Dependences with other requirements:** This is dependent on requirement 2, as the user must have access to the database before an entry can be edited.

3.5 Delete an entry

- **Description:** User must be able to remove an entry from the database
- **Criticality:** This is a not as critical as creating or editing entries, as the user can still use the application without this requirement. The removal of entries is important
- **Technical Issues:** It is important
- **Dependences with other requirements:** This is dependent on requirement 2, as the user must have access to the database before an entry can be deleted.

3.6 Synchronize database between devices

- **Description:** Pair multiple devices so that the user's password database is synchronised across devices
- **Criticality:** This is one of the major features of the application, so it is quite important that this is working correctly. Without this requirement the user will have to manually synchronise their database across devices.
- **Technical Issues:** This could take a lot of time to implement as it's important that the file being synchronised between devices is not exposed or corrupted in the process.
- **Dependences with other requirements:** This is dependent on requirement 2, as the user must have access to the database before the synchronisation process will be allowed to start.

3.7 Generate new password

- **Description:** User must be able to generate a new password automatically when in the process of creating or editing an entry.
- **Criticality:** This is not a critical requirement; however, it is important that the user has the option to use long generated passwords to ensure they are very secure.
- **Technical Issues:** This should be relatively easy to implement; however, it is important that each generated password is completely random.
- **Dependences with other requirements:** This is dependent on requirements 3 and 4 as the user must be able to create or edit an entry to use this feature.

3.8 Organise entries into groups:

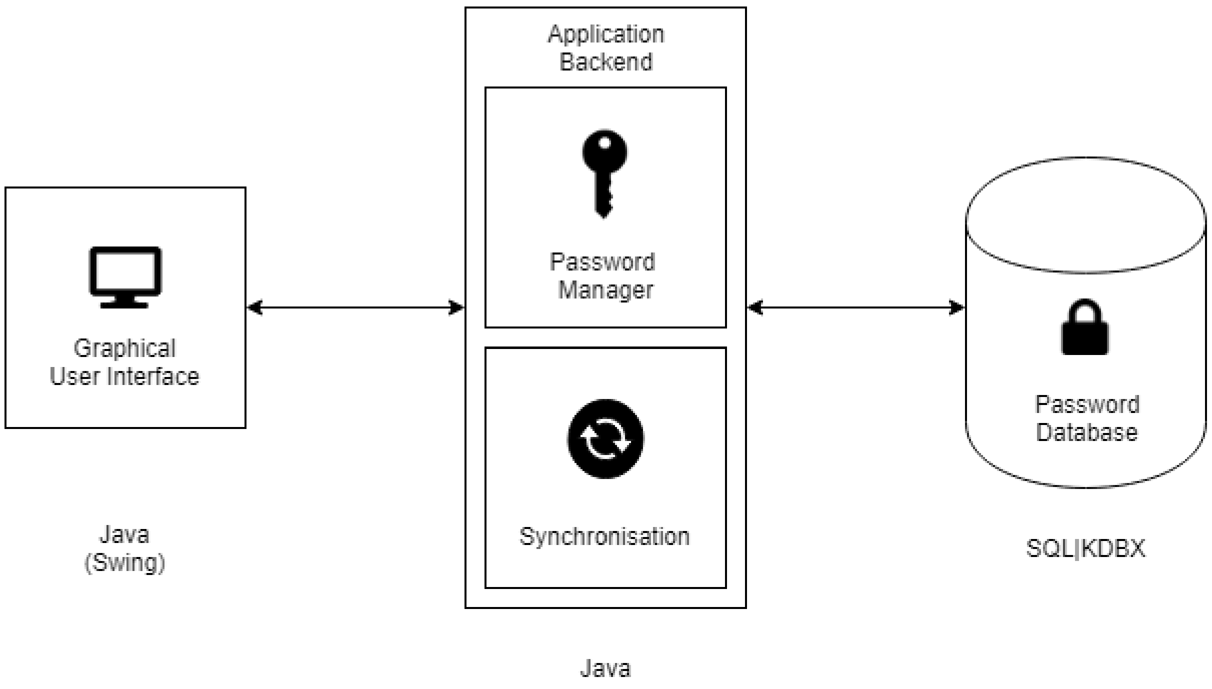
- **Description:** It should be possible for users to organise their password entries into groups.
- **Criticality:** This is not a critical feature; however, it will improve the organisation and usability of the application.
- **Technical Issues:** None
- **Dependences with other requirements:** This is dependent on requirement 3, as the user must first have existing entries before they can organise them into groups.

3.9 Password Expiry

- **Description:** User can decide if they want a password to expire after a specified amount of time.
- **Criticality:** This is not a critical feature as it is not commonly used, but it is a useful feature for users who regularly must change their passwords.
- **Technical Issues:** It will be important that passwords only expire after the specified time.
- **Dependences with other requirements:** This is dependent on requirement 3 and 4, as the expiry will be specified in the creating or editing of entries.

4. System Architecture

4.1 System architecture diagram



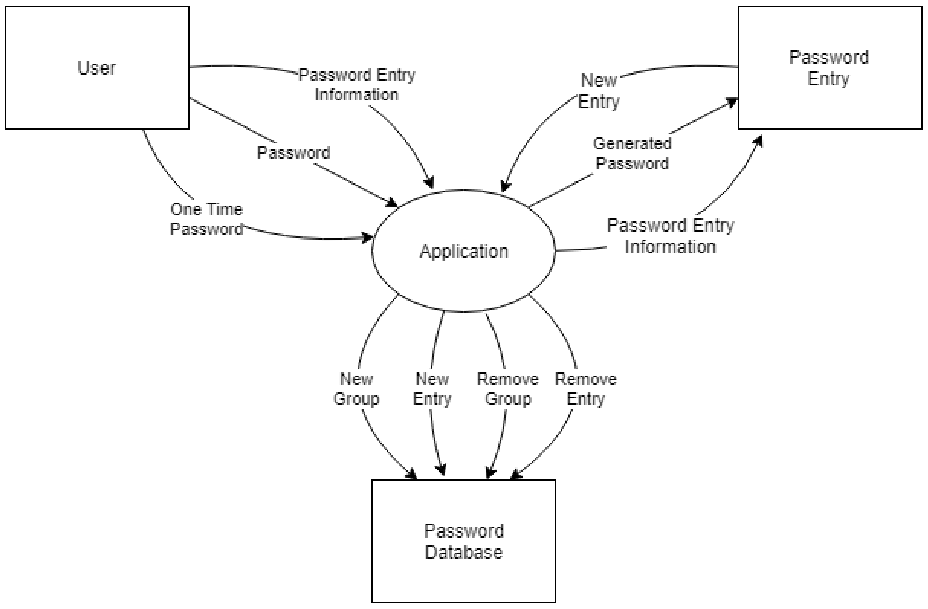
The above diagram shows a high-level structure for my application. The application backend will consist of two main components – Password Management and Synchronisation. The password management portion will handle the encryption and decryption of the password database, as well as reading and writing to the database. The Synchronisation component will handle the synchronisation of the password database between multiple devices. Both components will be implemented using Java.

The password database will consist of a single file. This could potentially be either an encrypted SQL database or use the open source KDBX format.

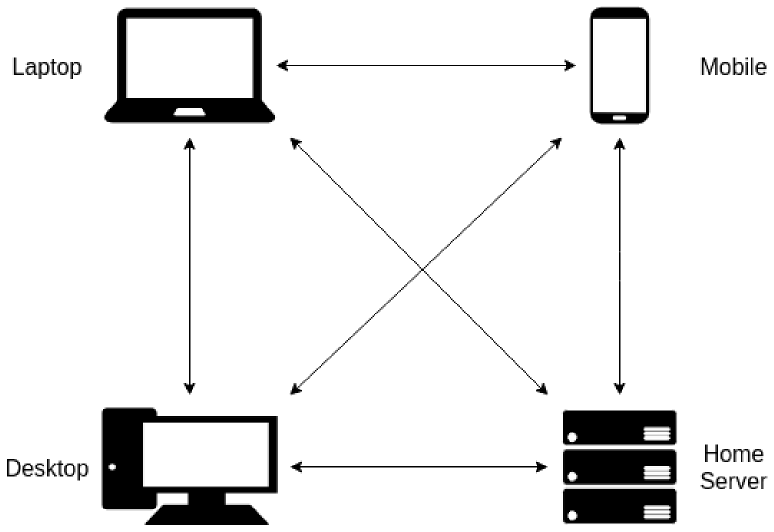
The graphical user interface will be implemented using Swing. Swing is a graphical user interface widget toolkit for Java. It is platform independent; it is written completely in Java. Originally this was planned to be implemented with JavaScript/TypeScript and connected to the application backend using a REST API. However, due to the possible performance and security drawbacks of this implementation I have decided to implement the user interface using Swing instead.

5. High-Level Design

5.1 Context Diagram

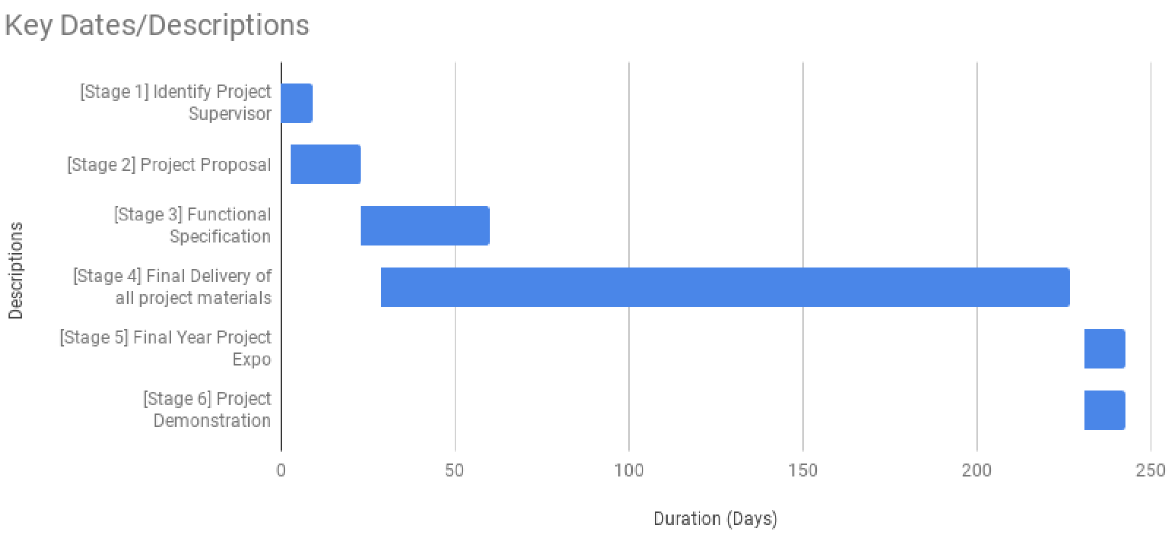


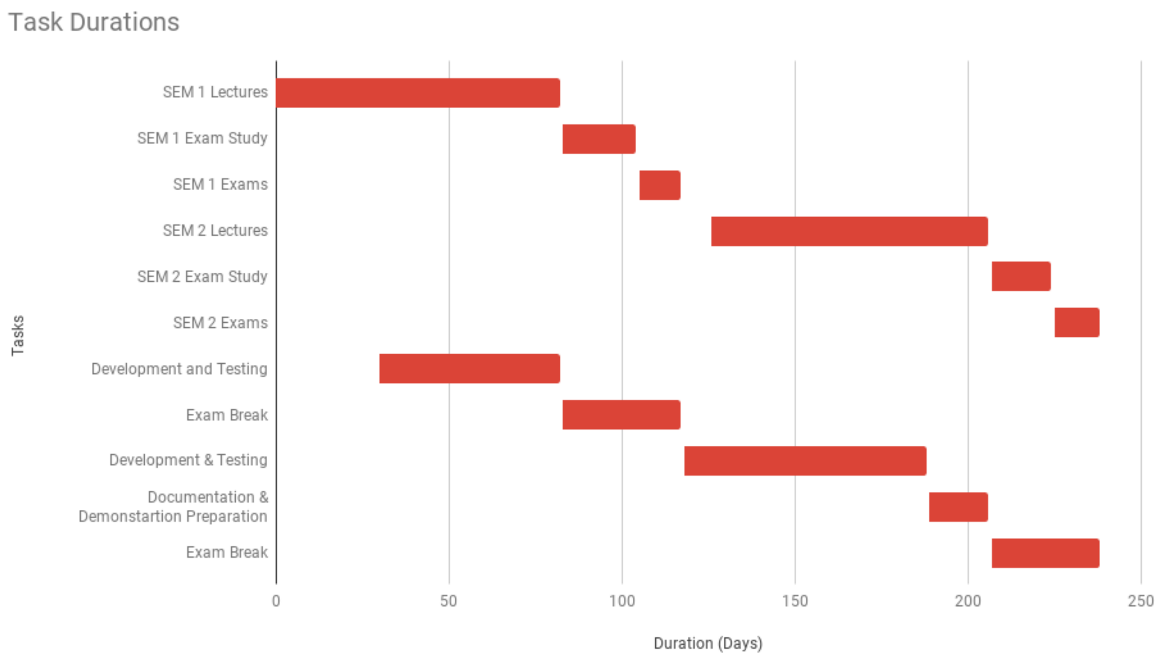
5.2 P2P Architecture Diagram



This is a simple diagram depicting the possible peer to peer connections that will be created between multiple devices in order to synchronise the password database between the devices. Ideally the user will always have more than one of these devices online when making alterations to the database. A reasonable example of this would be either a mobile or home server; these are devices that are kept online almost 24/7.

6. Preliminary Schedule





7. Appendices

Libraries:

- **Swing -** https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
- **KeePassJava2 -** https://github.com/jorabin/KeePassJava2/

Existing Password Managers:

- **KeePass Password Manager -** https://keepass.info/
- **Enpass Password Manager -** https://www.enpass.io/
- **Lastpass Password Manager -** https://www.lastpass.com/
- **Keeper Password Manager -** https://keepersecurity.com/