CS/CE Senior Design Specification report

Cerebro Password Manager

Dylan Cenotto, James Dempski

Faculty Adviser: Xiang Fu

Fall 2021

12/02/2021

Contents

Chapter 1: Project Description and Goals

Section 1.1: Project Scope and Vision

Modern Internet users use websites and applications and must create accounts for the multitude of services that each user accesses. A recent study by NordPass [1] found that the average person is expected to remember 100 passwords. This number has increased 25% from the previous 2019 poll that found that the average person is expected to remember 70-80 passwords. User account setup which requires you to create and remember a login and a password. Logins are usually set as your email address or phone and are generally easy to remember, however they are insecure and require a password that adds security to the credentials. Because of the increasing number of services that the average user has accounts with, the simplest solution is to use one password for every account. CyberNews [2] reported that a hacker leaked a list containing 8.4 billion passwords. Massive data breaches and cyberattacks put individual account passwords at risk, but this risk is amplified across every service that uses the same password. The problem of duplicate passwords can be solved by our password manager which would only require the user to remember the one password to the password manager. By making it convenient and easy to remember passwords for users, users will not have to rely on one password anymore and could have a variety of passwords across their services.

Several other password managers have been created in the past by tech giants such as Google and Apple and by smaller companies such as LastPass and 1Password. These other managers, like our implementation, save a variety of passwords for each user. However, some are just limited to one browser. Chrome and Firefox both have password managers built into their browser. The browser-based managers do not allow the user to use their passwords in other browsers. This can be inconvenient when using a device that has a different browser from the user's normal browser. There have also been documented CSRF and XSS vulnerabilities [3] in browser-based managers that can allow hackers to access the user's passwords through exploiting the browser. Other companies like Microsoft and Apple have password managers built into their operating systems. This offers more mobility than browser password managers but still restricts scope of password access to devices that use the same operating system as them.

Our project intends to create an application that securely generates, stores, and fills in passwords for various websites. All data associated with our application will be stored on a server, which means that it will follow the client-server architecture model.

The application will have a login page that will authenticate the user and provide them with their saved login information list. This allows for users to view and use their stored information from any client that is connected to the internet. All login and user data will be stored securely on a web server, with which the local client will communicate. When the user starts working with the application, they will be able to register their account in the system, after which they will be able to login with the registered credentials. If for any reason they forget or lose their account credentials, they can recover them via an email confirmation and password recovery.

Once the users log in, they are redirected to their personalized list of website entries for which they've saved their password. They will be able to select a website, which would redirect them to the login page of that website with the credentials filled in. If resources permit, our application will have the feature of authenticating "behind the scenes". This entails that our application can start a secure ssl communication line and retrieve the session cookie in the background, bypassing the authentication and redirecting the user to the webpage already signed in. There will also be an option to generate a secure random password within the app, which they could use for their websites.

Passwords can be saved for a number of different websites. For each website the application will save two things: the credentials and the HTML objects responsible for the login process on the page. For example, if a website has an input field for the username, another one for the password and a button to sign-in, the names of the three HTML objects will be saved in the database. This means that the website entries must be added manually to the database, which means that only those websites will be supported. However, more advanced users who are familiar with HTML will be able to add their own custom entries, by providing the necessary object names themselves.

As stated before the application will be cloud-based, meaning that all the user credential data will be saved on a remote server. This is done so that users are not limited to just one device when using our application. If, for any reason, that device is corrupted or unavailable, all the passwords will be safely stored remotely, and can be accessed from any device that has the client installed and no loss of data occurs. It also relieves the host device from unnecessary data being stored locally. The client will be cross-platform, available on Windows, Java and Linux.

Since the project is cloud-based it will require some third party cloud services, such as AWS or GCP, for server maintenance. The logic within the server will be organized via PHP scripts. The data for our application will be organized using one of the Structures Query Language (MySql, PostgreSQL). The client will be programmed in Java, because it has a very good debugging interface and has well suited libraries for constructing a GUI. The client will communicate directly with the server and vice-versa.

Section 1.2: Project Goals

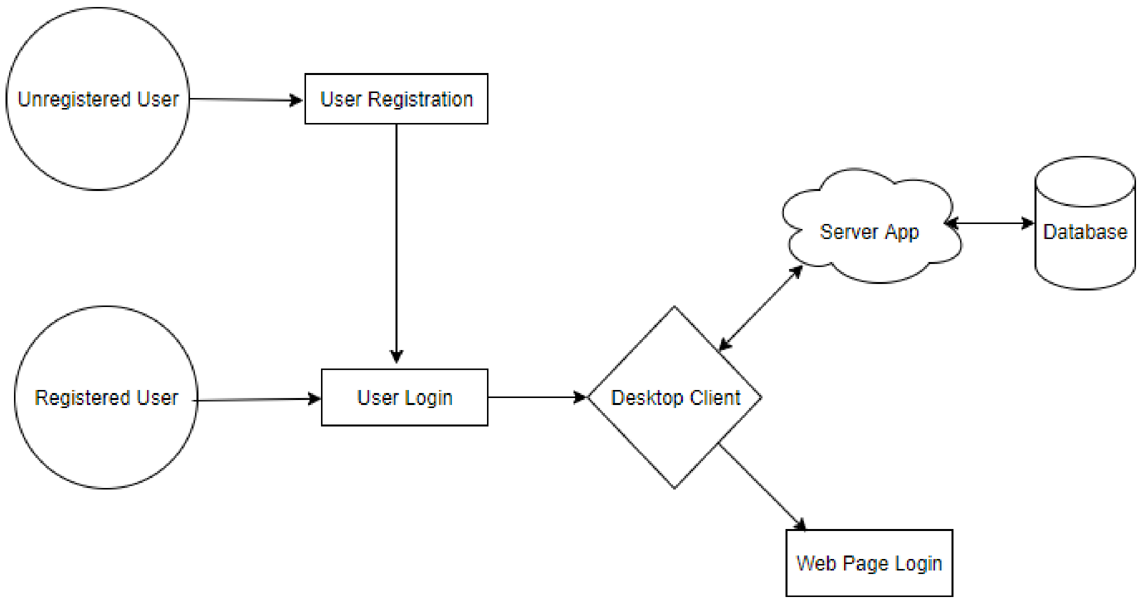
The following are the goals for our project:

- Create a password manager that securely stores passwords for multiple users.
- Create a local client application that can be run on Windows, MacOS, and Linux where users can access their passwords.
- Create a cloud server application that can process incoming requests, securely store necessary data, and respond to requests from the client.
- Make a client-server relationship that allows users to access their passwords from any computer with an internet connection.

Chapter 2: Requirements Specification

Section 2.1: System Perspective

Figure 1: System Overview



Section 2.2: User Stories

US-1 User registration

- User: Unregistered users
- Description: Describes process for user registration
- Reason: To make a password-protected account that only one user can access
- Step-by-step description:
    - Bob the unregistered user selects Register button on main page
    - Bob fills in Username, Password, Confirm Password, and Email fields
    - Bob selects Submit button to complete registration
    - Desktop Client sends email and password to cloud to add to user database

    - Web Server sends acknowledgement to Desktop Client
    - Desktop Client directs Bob to User login page

US-2 User login

- User: All users
- Description: Describes logging in to desktop client
- Reason: To authenticate client with the server
- Step-by-step description:
    - Bob the registered user will enter their username and password in designated fields
    - Bob will select Login button
    - Credentials sent to server for authentication
        - If credentials match an existing account, send positive acknowledgement to client as well as an authentication cookie for the remainder of the session
        - If credentials do not match an existing account, send negative acknowledgment to client
    - Client displays results of login from the server's acknowledgement
        - If acknowledgement successful, client displays "login successful" and directs user to main page
        - If acknowledgement unsuccessful, client displays "cannot find an existing account with those credentials, if you are a new user please follow the link below to register now"
    - Client displays link to User Registration page

US-3 Viewing the list of websites with saved passwords

- User: Registered Users
- Description: To describe main page of client

- Reason: Show what user can do on main page
- Step-by-step description:
    - Bob the registered user just logged in to the desktop client
    - Bob can read through their saved password entries
    - Bob can select Logout button to log out of the desktop client

US-4 Adding a password to the saved passwords list

- User: Registered Users
- Description: Describe process for adding password
- Reason: To add a password to the list of saved passwords
- Step-by-step description:
    - Bob finds a website they would like to save a password for
    - Bob clicks on Add New Websites button which opens Add New Website page
    - Client displays Add New Website page
    - Bob fills in Website Name, username, password, and Additional Field
    - Bob selects Save button
    - Client updates entry in Server database

US-5 Accessing a web-page with the credentials filled in

- User: Registered users
- Description: Describe process for opening web pages
- Reason: To open web-page and fill out and submit credentials
- Step-by-step description:
    - Bob finds the entry for the web-page that they want to visit
    - Bob selects Visit button

    - Client opens a browser instance and opens the user-requested web-page
    - Client fills out and submits credentials on the web-page
    - Bob will be directed by web-page to the home-page of the specified website

US-6 Generating a secure password

- User: Registered Users
- Description: Describe process of generating a secure password
- Reason: To make a secure password for users to enter for website passwords
- Step-by-step description:
    - Bob selects Generate Password button on main page
    - Bob enters the Password length
    - Bob enters the Min. Number of Uppercase and Lowercase characters
    - Bon enters the Min. Number of Special Chars
    - Bob selects Generate button
    - Client uses the specified parameters to generate a password
    - Client displays generated password in field adjacent to the Generate button

US-7 Editing a previously saved password entry

- User: Registered Users
- Description: Describe editing saved password entry
- Reason: Edit a saved password entry from main page
- Step-by-step description:
    - Bob finds the entry that they want to edit
    - Bob selects Edit button
    - Client displays Add New Website page

    - Bob fills in Website Name, username, password, and Additional Field
    - Bob selects Save button
    - Client updates entry in Server database

US-8 Recovering and resetting super-password

- User: Registered Users
- Description: To describe process for recovering a lost super-password
- Reason: Provide a way for users to access account without password
- Step-by-step description:
    - After User Login fails three times, display Forgot Password? Link
    - Bob selects Forgot Password? Link
    - Bob fills in their email address and selects Generate Email Token button
    - Client emails user a reset token
    - Bob finds reset token in their email and fills in Enter Email Token field
    - Bob selects Submit button
    - Client verifies token is correct and directs Bob to Make New Password page
    - Bob fills in Password and Confirm Password fields
    - Bob selects Submit button
    - Client update super-password in Server database

Section 2.3: External Interfaces

Figure 2.3.1 User Interfaces

UI name and number	Short Description	Reference to user stories
UI-1.1 Login Screen	Client page for user login	US-1, US-2

UI-1.2 Registration Screen	Client page for user registration	US-1, US-2
UI-1.3 Main Screen	Client main screen where they can see the list of the websites with saved passwords	US-3, US-5
UI-1.4 Password Generation Screen	Client screen for generating a secure password	US-6
UI-1.5 New/Edit Website Entry Screen	Client screen for adding a new website entry or editing an old entry in the system	US-4
UI-1.6 Advanced New Website Entry Screen	Client screen for adding a new website entry using the advanced mechanics	US-4
UI-1.7 Super-Password Recovery	Client screen for recovery password in event user loses it	US-8

Figure 2.3.11 - UI-1.1 Mockup

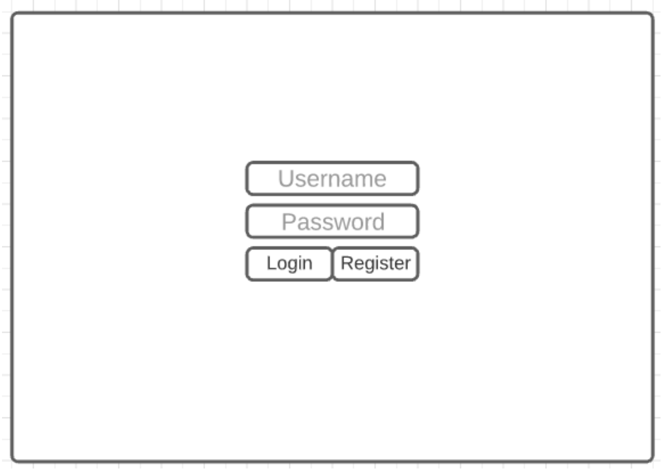


Figure 2.3.12 - UI-1.2 Mockup

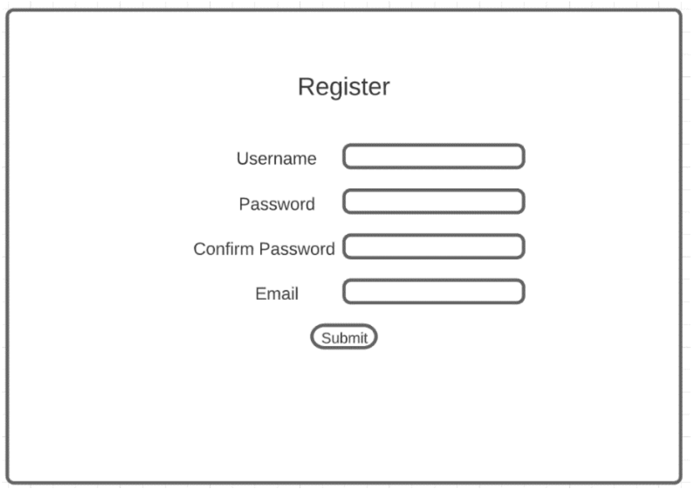


Figure 2.3.13 - UI-1.3 Mockup

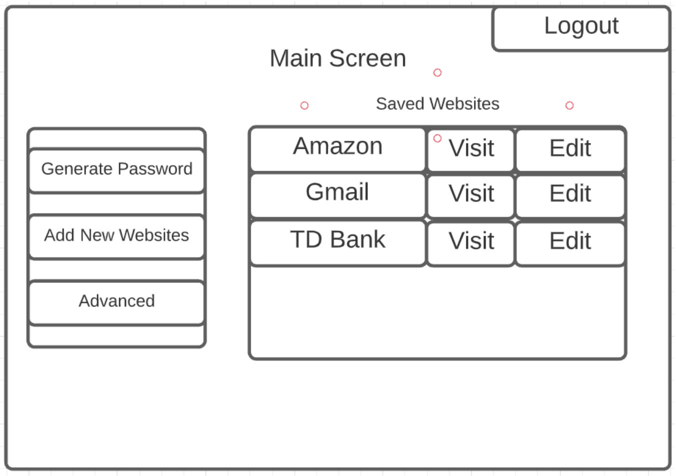


Figure 2.3.14 - UI-1.4 Mockup

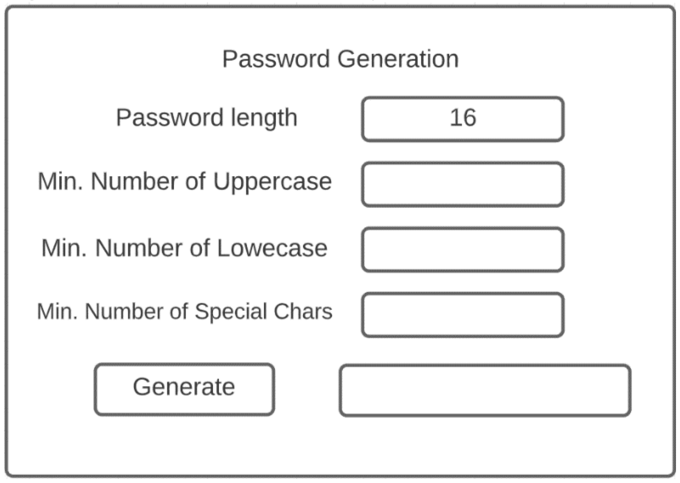


Figure 2.3.15 - UI-1.5 Mockup

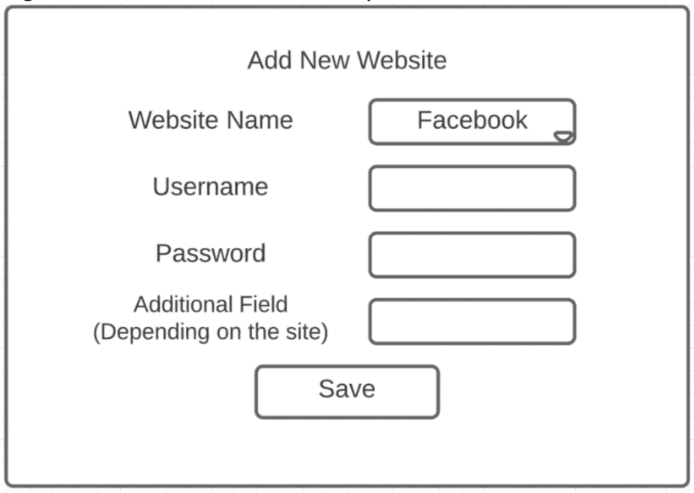


Figure 2.3.16 - UI-1.6 Mockup

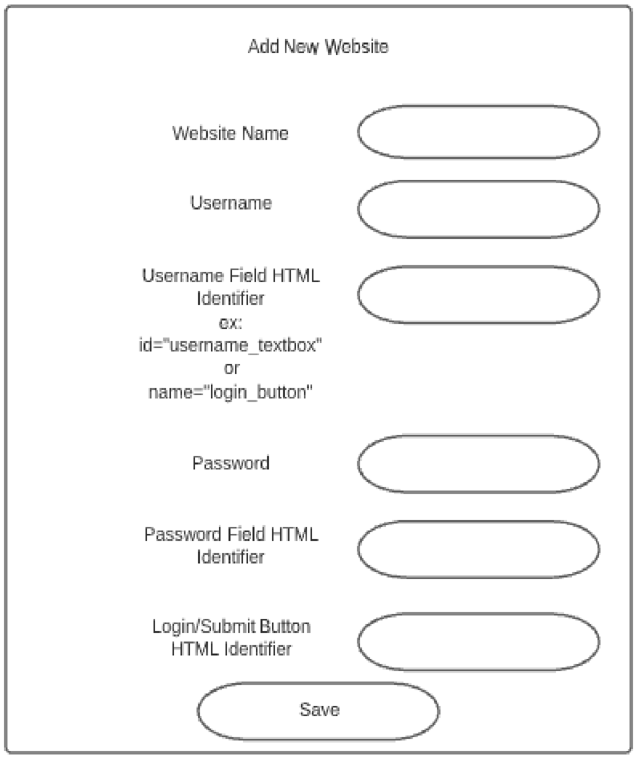

Add New Website
Website Name
Username
Username Field HTML Identifier ex: id="username_textbox" or name="login_button"
Password
Password Field HTML Identifier
Login/Submit Button HTML Identifier
Save

Figure 2.3.17 - UI-1.7 Mockup

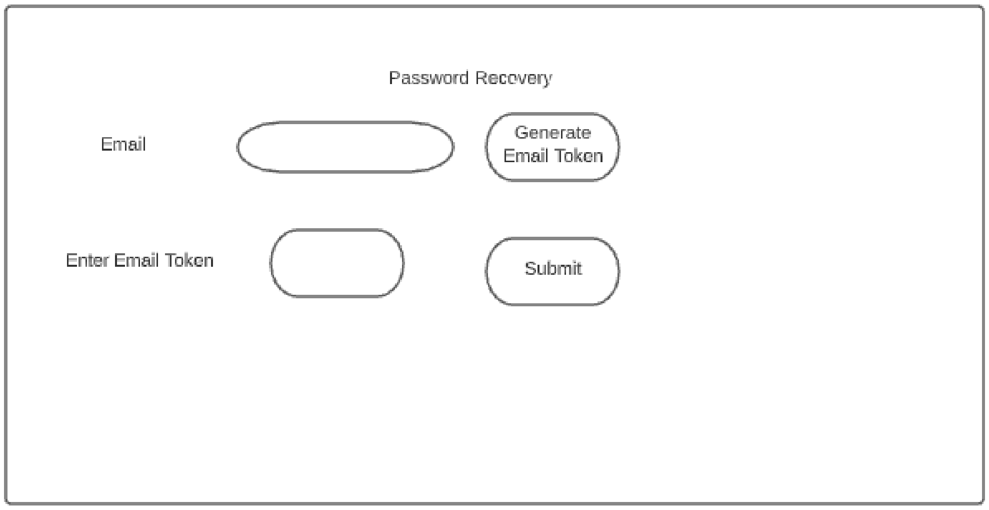


Figure 2.3.2 Inputs

Input # and name	Source	Description	Valid range	Data format	Reference

I-1.1 Username Box	Text Field	Field for the username	<32 chars	Text	UI-1.1
I-1.2 Password Box	Password field	Field for password	<32 chars, strong password*	Secure Text	UI-1.1
I-1.3 Login Submit Button	Button	Submit login information		Button	UI-1.1
I-1.4 Register Button	Button	Button to go to registration page		Button	UI-1.1
I-2.1 Username box	Text field	Field for entering username during registration	<32 chars	Text	UI-1.2
I-2.2 Password box	Password field	Field for entering users password during registration	<32 chars, strong password*	Password text	UI-1.2
I-2.3 Password confirm box	Password field	Field for confirming the password the user entered in I-6	<32 chars, strong password*	Password text	UI-1.2
I-2.4 Email box	Text field	Field for entering the users email for registration, is used for account recovery	Valid email address	Email	UI-1.2
I-2.5 Security question drop-down	Dropdown menu	Dropdown list of security questions, is used for account recovery		Dropdown selection	UI-1.2
I-2.6 Security question answer box	Text field	Answer to the security question selected in I-9	<32 chars	Text	UI-1.2
I-2.7 Registration submit button	Button	Button to verify and submit registration information		Button	UI-1.2
I-3.1 Website dropdown menu	Dropdown menu	Dropdown list to view all the available websites to save the		Dropdown selection	UI-1.5

		password for			
I-3.2 Website username input	Text field	Field for entering the username for the chosen website	<32 chars	Text	UI-1.5
I-3.3 Website password input	Password field	Field for entering the password for the chosen website	<32 chars, strong password*	Password text	UI-1.5
I-3.4 Additional fields	Text fields	Additional fields if the website requires more than just a password and username	<32 chars	Text	UI-1.5
I-3.5 Add website entry button	Button	Button to add the website entry with the credentials to the users account		Button	UI-1.5
I-4.1 Password length field	Number field	Filed for the number of characters to be generated in the password	<32	Number field	UI-1.4
I-4.2 Number of special characters field	Number field	Field for the least number of special characters to be generated in the new password	can’t be 100% of password, strong password*	Number field	UI-1.4
I-4.3 Number of uppercase letters field	Number field	Field for the least number of uppercase letter characters to be generated in the new password	can’t be 100% of password, strong password*	Number field	UI-1.4
I-4.4 Number of lowercase letters field	Number field	Field for the least number of lowercase letter characters to be generated in the new password	can’t be 100% of password, strong password*	Number field	UI-1.4
I-4.5 Number	Number	Field for the least	can’t be 100%	Number	UI-1.4

of digits field	field	number of digit characters to be generated in the new password	of password, strong password*	field	
I-4.6 Generate password button	Button	Button to verify password requirements and generate password		Button	UI-1.4
I-5.1 Open website button	Button	Button that will open a browser window on the web page with the pages credentials filled in		Button	UI-1.3
I-5.2 Edit website entry button	Button	Button that will redirect the user to UI-1.5 to edit a previously save website entry		Button	UI-1.3

* Password must be at least 8 chars long and contain at least 1 special character, 1 digit, 1 uppercase letter, 1 lowercase letter

Figure 2.3.3 Outputs

Output # and name	Source	Destination	Description	Data format	Reference
O-1.1	Client	Server, credentials database	Checking with the credentials database to allow/disallow entry	Text field and password field	UI-1.1, US-2
O-1.2	Client	Server, credentials database	Adding a new credentials entry in the database		UI-1.2, US-1
O-2.1	Server, passwords database	Client	Retrieving passwords entries saved by the user from the database	Website entries: username and website name	UI-1.3, US-3
O-2.2	Client	Server, passwords	Inserting a new entry into the servers	Website credentials	UI-1.5, US-4

		database	passwords database or updating an old entry		
O-3.1	Client	Browser	Sending the credentials to the browser for authentication	Website credentials	US-5
O-4.1	Client	Client, UI-1.4 output password box	Generating a secure password within the client	Text	UI-1.4, US-6,

Section 2.4: Functional Requirements

Figure 2.4.1

Name	FR-1 Verify User Credentials
Description	The system must be able to process the inputted data and verify it with the external database
Rationale	The system must verify the users credentials in order to provide them with their password information
System Behaviour	The user inputs the data into the login page fields, the client collects that data, connects to the server and sends the data to the server. After verifying the data, the server either sends the necessary data for the session authentication or sends an unsuccessful authentication response. Once the client receives the response, depending on its type, the user will either be allowed into the system or will receive an error message.
References	US-2, UI-1.1, FR-1, O-1.1

Figure 2.4.2

Name	FR-2 Register New User
Description	The system must be able to register a new user
Rationale	New users must be able to register a new account to be able to start using the application
System Behaviour	The user clicks on the registration button and is redirected to a registration page where they are prompted to enter the necessary information for registration. Upon completion, the client verifies the information and sends it to the server, which adds the information to the credentials database

References	US-1, UI-1.1, UI-1.2, O-1.2

Figure 2.4.3

Name	FR-3 Retrieving saved passwords
Description	The system must be able to retrieve the users saved password from the server
Rationale	The user must be able to access their saved passwords
System Behaviour	After successfully logging in, the client sends a request to the server to retrieve the saved website credential entries associated with the user. Once the client gets the response, it redirects the user to the main page of the application with an organized list of the received credentials
References	US-3, UI-1.3, O-2.1

Figure 2.4.4

Name	FR-4 Accessing web-page with saved credentials
Description	The client will be able to redirect the user to the web page with the saved credentials filled in
Rationale	The user must be able to access the website with the saved credentials already inputted for them
System Behaviour	Once the user is in the main page of the client with their saved websites listed, they will have the option to click on a button located on each entry to open the webpage with the credentials filled in.
References	US-5, UI-1.3, O-3.1, I-5.1

Figure 2.4.5

Name	FR-5 Adding a webpage entry
Description	The system will be able to add new website entries for users
Rationale	The user must be able to save their passwords for the website they choose
System Behaviour	The main page of the client will have a button, located on each entry, that will redirect the user to the screen where they can add a new website entry. Once on the page they will be required to choose a website from the dropdown menu for which they want to save the passwords, add the username and password and add any additional fields, if necessary. Once that information is submitted, it is

	sent to the server where it is added to the website entry table. After that the user will see that entry in the main screen of the application.
References	US-4, UI-1.5, O-2.2

Figure 2.4.6

Name	FR-6 Password Generation
Description	The application will be able to generate a customizable random password
Rationale	If the user will ever need to generate a random password, that feature will be present within the application
System Behaviour	To generate a password the user must click on the button that will redirect them to the password generation page. In that page the user will be able to customize the password structure, length and which characters must be included. Once all the parameters are set and submitted, the password will be generated and outputted to the user.
References	US-6, UI-1.4, O-4.1

Figure 2.4.7

Name	FR-7 Editing Website Login Information
Description	The application will be able to modify an existing entry
Rationale	In case the users credentials changed for a certain website, it is possible to change the saved information in the system
System Behaviour	Once the user clicks on the button to edit an entry, they will be redirected back to UI-1.5, to edit their old credential information. Once the new information has been submitted, the application will verify the entered information and will send it to the servers database for update. This can also be used to copy the password to clipboard for further use.
References	US-7, O-2.2, UI-1.5, I-5.2

Section 2.5: Non-Functional Requirements

2.5.1 System requirements

- NFR-1: The desktop application must be cross-platform. Meaning it should work on Windows, Linux and Mac operating systems.
- NFR-2: The desktop application must be able to work with the most common browsers (Google Chrome, Microsoft Edge, Safari, Firefox).

2.5.2 Security requirements

- NFR-3: Login data in the User Credential database will be encrypted to ensure security in case of a data breach
- NFR-4: Passwords in the Website Credentials database will be encrypted in a way that only the users account, whose password is stored, can successfully access it
- NFR-5: Communication with the systems server will only be done over a secure HTTPS connection
- NFR-6: Every incoming request to the server must pass authentication before being processed
- NFR-7: All sensitive information will be encrypted during communication between the client and the server
- NFR-8: The database can only be accessed through a desktop application

2.5.3 Performance requirements

- NFR-9: The system server must have an uptime of 99.9%
- NFR-10: The request from the client to the server to get all website entries associated with a given user have to be done in a span of no more than a second, since it will be done every time a user goes to the main screen of the application

2.5.4 Logical Database Requirements

- NFR-11: The server database will store the required HTML information to successfully fill in the necessary fields within the webpage

Chapter 3: Standards and Constraints

Section 3.1 Standards:

Standard	Description

HTML	Standard markup language for web pages that is used when displaying web pages to users
PHP	Widely used server scripting language which is used when sending requests to database and desktop client app
MySQL	Database management system that is used to manage relational database tables and to connect with the server
HTTPS	Protocol that uses TLS as well as certificates and handshakes to secure communication over the internet
TCP/IP	Standards that define how packets should be sent and how those packets should be routed, processed, and transmitted
RSA	Asymmetric cryptography standard that uses private and public keys to encrypt data
SHA-512	Hashing algorithm that is used to encrypt data
JSON	Open-standard file format used to send human-readable data objects

Section 3.2: Constraints

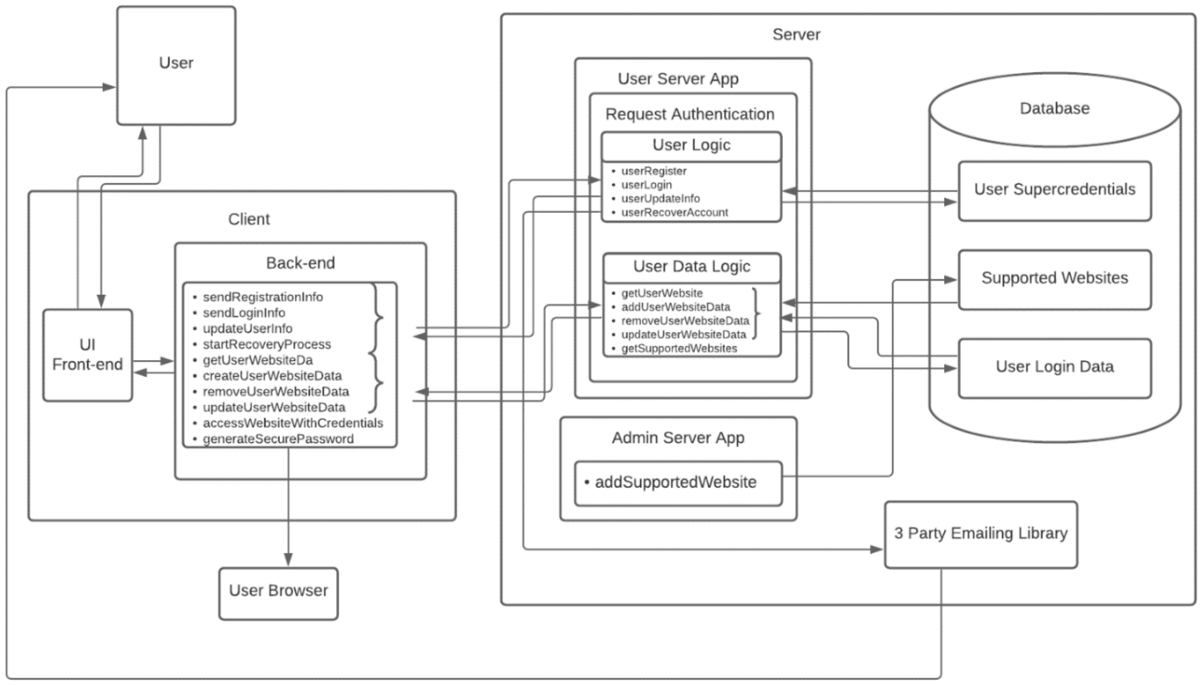
Type	**Name**	**Description**
Economical	RC-1	If number of hours exceeds free limit, then cloud-based virtual machine will accrue a balance of $0.01/hour
Environmental	N/A	No major environmental constraint or

		limitation
Health	N/A	No major health constraint or limitation
Safety	RC-2	If the user is subject to a keylogger attack or if the desktop client app is in any way compromised then their data's safety may be at risk
Welfare	N/A	No major welfare constraint or limitation
Ethical	N/A	No major ethical constraint or limitation
Legal	N/A	No major legal constraint or limitation
Social, global, and cultural	RC-3	Some features will be difficult for users that do not have an advanced understanding of how passwords are generated as well as what websites will be able to be auto-populated with credentials
Political	N/A	No major political constraint or limitation
Manufacturin g	N/A	No major manufacturing constraint or limitation
Sustainability	N/A	No major sustainability political constraint or limitation

Chapter 4: Project Design

Section 4.1: Overview of System Components

Figure 4.1.1



The system follows a typical Client-Server architecture pattern, with a centralized server and multiple clients that can access it. The client is comprised of a front-end user interface and a backend logical unit. The user can only interact with the UI, which in turn communicates with the backend. The backend of the client is its only component that communicates with the remote server. The server has two main units: the server application and the database. The server app authenticates and processes the incoming requests from the backend of the client and retrieves/modifies data from the database. The database consists of the tables: User Super-credentials, Supported Websites and User Login Data and it only communicates with the server app.

Section 4.2: Structure and Relationship

4.2.1 Client

The client is the program that the user will have to install onto their personal computer in order to use the services of our application. It will be compatible with multiple operating systems (Windows, Linux, Mac), which corresponds to the NFR-1 non-functional requirement.

4.2.1.1 User Interface

The client user interface, also known as the front-end, is the only component of the system with which the user interacts. The UI gathers all the data from the user and sends it to the backend of the client. It also shows the organized data received from the backend back to the user. The UI does not communicate with any other component other than the backend in order to protect the system from potential vulnerabilities.

4.2.1.2 Backend

The client backend oversees processing the data received from the UI and passing it to the remote server as well as receiving the data sent by the server, organizing it, and passing it back to the UI for the user. The backend only communicates with the UI and the server app. The backend is divided into two logical subcomponents: user super-credentials and user login data. The super-credentials subcomponent oversees handling user login, registration, account information updates and user account recovery. This corresponds to US-1, US-2 and US-8 user stories and FR-1 and FR-2 functional requirements. Similarly, the login data subcomponent handles saving, deleting and modifying login information for a particular user website as well as retrieving saved entries for that user. User stories US-3, US-4, US-7 and requirements FR-3, FR-5, FR-7 correspond to this subcomponent. Both of these subcomponents communicate with their corresponding server counterparts. The client will also be able to access the user's browser in order to fill in their login data on the login page as described in the FR-4 functional requirement.

4.2.2 Server

The server side manages the processing of all the requests from the clients and stores all the necessary information for the successful operation of the application.

4.2.2.1 Authentication

In order to avoid Denial of Service attacks and other potential vulnerabilities, an authentication protocol component is added to the system structure. All the requests from the clients must first pass through the authentication protocol before it is processed by the server application. The protocol will also prevent the processing of non-client requests by the server. This authentication component corresponds to the NFR-7 and NFR-8 requirements, the details of the protocol will be discussed further in more detail.

4.2.2.2 Server Application

As stated before, the server side is divided into two main units: the server application and the database. The server app first authenticates and then processes the incoming requests from the clients, and, if necessary, communicates with the database. Only the server app can communicate with the database to prevent potential data breaches and other attacks. Similar to the client, the server app is divided into two main logical units: user logic and user data logic. User logic supervises the process of authentication of the user within the system: login, registration, account information updates. It also is the only component that has access to the user super-credentials table. The other unit, user data logic, takes care of the user's login information that they decide to save. This component will have access to the User Login Data table, and it will be able to read from the Supported Websites table.

4.2.2.3 Emailing Service

The server app will also make use of 3d party libraries in order to send emails during the password recovery process, as described in the US-8 user story.

4.2.2.4 Database

The second component of the server side, the database, will be stored on the same physical server as the logic and will only communicate with the server app. It is divided into three tables: User Super-credentials, Supported Websites and User Login Data. Since the database and the server application will be on the same physical server, communication between them will not be encrypted in any way, as it is considered secure.

4.2.3 Admin Server Application

Since the supported websites are added by the developers, typical users should not have access to change data in the corresponding table in the database, which is why there will be an administrative logical component within the server structure. This component will be able to add and remove entries in the table, which will then be used by other users.

Section 4.3: Detailed Component Description

Figure 4.3.1

SC-1	Client
Type	Application
Purpose	A local application that the user has to install in order to successfully use the password manager
Inputs	All the input from the user
Outputs	Outputs the user interface to the user
Data	The client will store a public and private key embedded in the source code of the app to successfully authenticate with the server. Once the user logs into the app, the client will also store the identification number of the user to reference in further server calls. Most of the data will be stored in session memory, after the session ends, the data is deleted
Internal Structure	The client application is going to be created using the Java environment. It consists of a UI and a backend component, consult figure 4.1.1 for further details.
Processing	The client can be thought of as a gateway to the password manager

	application, as this is what the user interacts with during usage. The client is divided into two parts: the front end and the back end. The user only interacts with the front-end, the front end passes the data to the back end, which processes the data and sends it to the server. The user can accomplish all functional requirements when using the client. It is cross-platform, which means that it can be installed on any computer running a Windows, Mac or Linux operating system.
Dependencies	The user interacts with the front-end of the client, the front-end communicates with the back-end of the client, which sends requests to the server. The client will also be able to access the users browser in order to open login pages with the login data filled in.
Resources	Java SDK, Selenium Library, Cross-Platform UI Java library
Cross-reference	All Functional Requirements and User Stories, NFR-1, NFR-2, SC-2, SC-3, SC-4

SC-2	Client User Interface
Type	Module
Purpose	The interface with which the user interacts
Inputs	All physical input from the user
Outputs	Outputs necessary data to the user in the form of an interface
Data	The UI will not store any data
Internal Structure	The UI will be implemented using a Cross-Platform Java library
Processing	The main purpose of the UI will be to interact with the user. Consult section 2.3 for further details on the structure of the UI
Dependencies	The UI processes all input from the user and passes it to the back-end of the client, and vice versa.
Resources	Java SDK, Cross-Platform UI Java library
Cross-reference	NFR-1, NFR-2

SC-3	Client back-end
Type	Module
Purpose	Process data received from the UI and communication with the server side
Inputs	Data from the front-end
Outputs	Data from the server
Data	Public and private keys of the user (generated during sign in, for login data encryption/decryption), user identification (received from the server after sign-in), which will be stored in session memory. Public and private keys of the client (for authentication with the server) will be embedded in the source code.
Internal Structure	Backend functions are listed in Figure 4.1.1
Processing	The back-end of the client controls communication with the remote server and gathers/processes data necessary for the UI. It is also in charge of all the

	user browser procedures (s.a. opening a browser and filling in the user's data)
Dependencies	The back-end is the middleman between the UI and the server. It also requires browser drivers and the Selenium library in order to successfully operate on the user's preferred browser.
Resources	Java SDK, Selenium library for Java, Browser drivers
Cross-reference	FR-4, FR-6, SC-2, SC-4

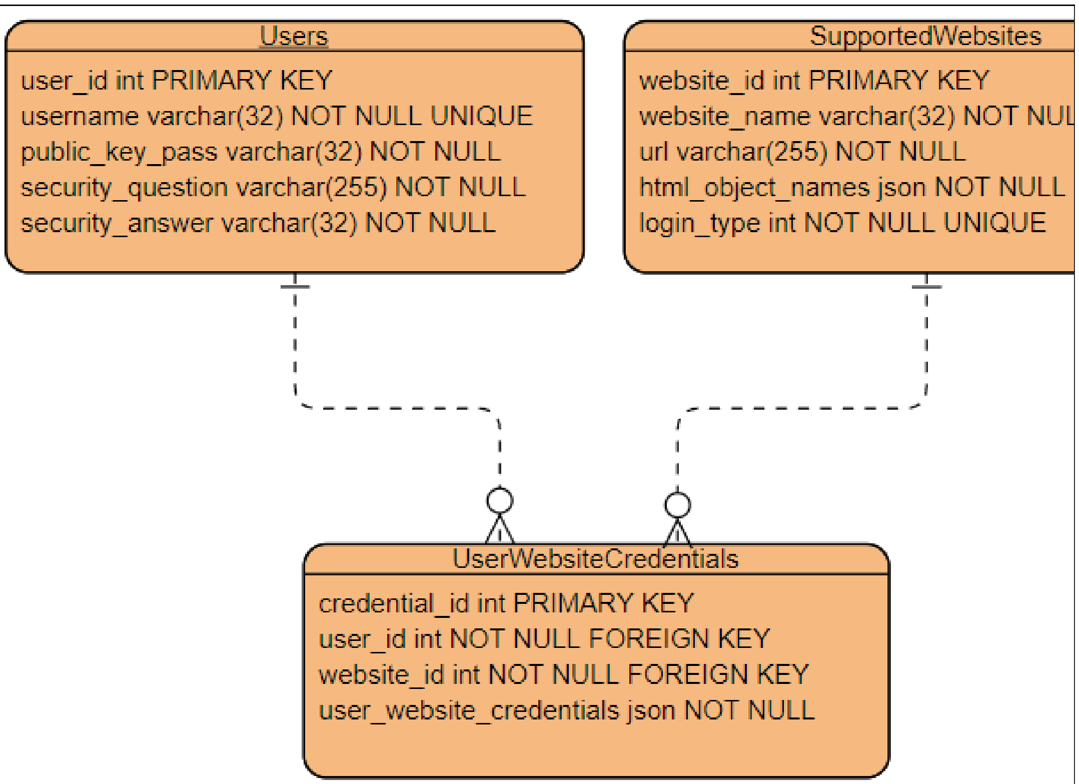
SC-4	Server
Type	Server
Purpose	To securely store user login data remotely, so that it can be accessed from anywhere at any time.
Inputs	Requests from the client back-end
Outputs	Organized data from the database in the form of a response, or a 200 response code if no data needs to be returned
Data	The server database will store all the necessary user data (user super-credentials, login data) and information for all the supported websites. The server will also store the public key of the client (for authentication) embedded in the source code.
Internal Structure	The server consists of three components: the main server application, the database, and the admin server application. The server only communicates with the client, if the requests pass authentication.
Processing	The main purpose of the server is to store data necessary for operation, change that data, and provide it to the client in a secure fashion. The server will also be used to send emails to users during account recovery.
Dependencies	The server responds to the requests sent by the client. The server only responds if the request passes the authentication protocol.
Resources	PHP, GCP server, 3d party emailing API for PHP, MySQL
Cross-reference	NFR-8, NFR-6, NFR-7

SC-5	Authentication
Type	Protocol
Purpose	To authenticate each request received by the server
Inputs	Requests from the client back-end
Outputs	Pass/No pass, depending on whether the request passes authentication
Data	The public key of the client is embedded into the source code of the protocol
Internal Structure	A PHP class that contains the authentication function
Processing	Each request to the server must contain an authorization token. Each token sent from the client will be a checksum of all the other parameters passed in the request encrypted using the private key of the client via RSA. The pseudocode for the protocol goes as follows:

	• The protocol inspects the request • Gets the authentication token passed in the request • Decrypts it using the embedded client public key • Gathers all the other passed parameters and checksums them • If the decrypted data matches the checksum of all the parameters, then the request passes authentication, otherwise it does not.
Dependencies	Each request to the client must first pass the authentication protocol, it is used by the server application.
Resources	RSA, MD5 or SHA512 checksums
Cross-reference	NFR-8

SC-6	Main server application
Type	PHP programs
Purpose	To process all possible incoming requests from the client and maintaining the database
Inputs	Requests from the client back-end
Outputs	Organized data from the database in the form of a response, or a 200 response code if no data needs to be returned
Data	No data is stored in the application itself
Internal Structure	Consists of two modules: user logic and user data logic, which are both wrapped in the authentication protocol.
Processing	The user logic processes super-credentials requests - registering a user, changing account information, account recovery and user logins. The user data logic processes requests associated with user website data. Functions within the modules can be found in Figure 4.1.1
Dependencies	The client back-end sends requests to the server application, the app communicates with the database if necessary, and responds to the request. The server app also uses the emailing API during account recovery.
Resources	PHP, 3d party emailing API for PHP
Cross-reference	NFR-5, NFR-6, NFR-7

SC-7	Database
Type	MySQL server database
Purpose	To permanently store the data necessary for the operation of the password manager.
Inputs	Queries from the server app
Outputs	Organized table data
Data	See the Internal Structure section
Internal Structure	Here is the Entity Relationship Diagram for the database

	 Users user_id int PRIMARY KEY username varchar(32) NOT NULL UNIQUE public_key_pass varchar(32) NOT NULL security_question varchar(255) NOT NULL security_answer varchar(32) NOT NULL SupportedWebsites website_id int PRIMARY KEY website_name varchar(32) NOT NUL url varchar(255) NOT NULL html_object_names json NOT NULL login_type int NOT NULL UNIQUE UserWebsiteCredentials credential_id int PRIMARY KEY user_id int NOT NULL FOREIGN KEY website_id int NOT NULL FOREIGN KEY user_website_credentials json NOT NULL
Processing	The server database permanently stores the data and responds to queries sent by the server app. Sensitive information in the "UserWebsiteCredentials" table will be encrypted using the public key associated with the user's account, so that in the case of a breach the sensitive information will remain secure. The "SupportedWebsites" table will store the login type and the necessary object names that need to be filled in, since each website can have a different authentication style.
Dependencies	The database only communicates with the server application, in order to avoid potential data leakage and vulnerabilities. The "SupportedWebsites" cannot be changed through the main server application, it can only be read. The table will only be updated through the admin server app.
Resources	MySQL
Cross-reference	NFR-3, NFR-4, NFR-11

SC-8	Admin server application
Type	PHP programs
Purpose	To process administrative requests
Inputs	Requests from the client back-end, which are authorized to use the admin app
Outputs	Organized data from the database in the form of a response, or a 200 response code if no data needs to be returned
Data	No data is stored in the application itself

Internal Structure	See Figure 4.1.1
Processing	The admin server application will provide additional functionality for the administration of the system, such as adding/removing entries in the "SupportedWebsites" table.
Dependencies	The authorized client back-end sends requests to the admin application, the app communicates with the database if necessary, and responds to the request.
Resources	PHP
Cross-reference	

Section 4.4: Methodologies

4.4.1 Research New Tools and Methods

4.4.1.1 Research Tools

- Open-Source Automated Testing Tools: Selenium, Katalon Studio, and Cucumber [4]. The following tools are all tools that can be used to automate testing; however, Selenium has extensive tools for driving web browsers and works well with a default plugin for the development environment that we are using.
- Programming Languages for Desktop Client App: Java, C++, and Python. Due to our need to make the project cross-operating system, we are using Java because of how portable the JVM environment is. Java also has plugins which allow the Selenium dependencies to be imported easily.
- Programming Languages for Server App: PHP and Ruby on Rails. Ruby on Rails is an extensive framework and the project members already have previous experience with PHP, so we elected to use PHP as the server. We were also advised to use PHP by our faculty advisor
- Cloud Platform for Hosting Linux Virtual Machine: Google Cloud Platform (GCP) and Amazon Web Services (AWS). GCP offers $100 more in free credit than AWS and has a better performing CPU on the free tier of the platform [5] [6].
- Database Management Service: MySQL, Microsoft SQL Server, PostgreSQL. Our faculty advisor recommendation and experience guided our decision in selecting MySQL to manage the database tables.

4.4.1.2 Research Methods

- Encrypting User Credentials: Initially, we were going to store a hash of the user's passphrase in a table. We switched to public key encryption scheme to add further protection of the user's super-credentials.
- Session Initiation: Initially, we wanted the server to generate a cookie that will be included in the information communicated with the server. Our advisor suggested we generate license keys to be included in the software downloads and use Shamir Heuristics to generate authentication tokens to initiate connection with the server.

Section 4.5: Design Alternatives

Our first design for the database had two tables as opposed to three. We knew that we wanted to have one table for user super-credentials which are the credentials that the user uses to log into our application and a table for the credentials for each of the user's individual websites. However, when we came up with the design of the second table it became cluttered with the list of supported websites and the user credentials for each website as well as the login data for each web page. This led us to separate the list of supported websites for each user from the credentials and login information for each site for a total of three tables.

We originally wanted the entire project to be a desktop application that stores the user credentials as hashes in a local file on the user's machine. To differentiate ourselves from other password managers, we wanted to expand our project to work on different operating systems and with different browsers. Because we wanted our app to work on multiple platforms, we decided to use Java to develop the app as Java executes code in the Java Virtual Machine (JVM) which has the same environment regardless of the operating system. We needed a browser driver that worked with Java, so we explored different options until we found Selenium. Selenium is a testing tool that can be added to Maven, which is a default plugin for Eclipse. Selenium allows us to open web pages in browsers and auto-fill their credentials directly into the HTML fields on the web page.

The decision to switch to cross-platform also led us to switch from a desktop app with a layered architecture to a client-server architecture. We wanted to store encrypted user data on a server in databases as opposed to just having the passwords stored in hashes on the user's machine. Because the server is hosted remotely, the users will be able to save passwords to the server on one machine and access them on another regardless of what operating system the machine is running.

Section 4.6: Reuse and Relationships with Other Products

We will not be acquiring any new hardware for this project and will be using our personal hardware to develop the project. We are using Java to implement the desktop client application. We are programming in the Eclipse Integrated Development Environment (IDE) and are programming in a Maven project. Maven is a project management tool that is a default plug-in Eclipse which also can include dependencies like Selenium. Selenium is a testing tool that is capable of both launching browsers and injecting user credentials into web page fields. The PHP-based server application will run on a Linux Virtual Machine that will be hosted on Google Cloud Platform. We will be using MySQL and creating tables that hold the user super-credentials, supported websites, and user login data.

Section 4.7: Resource List

Figure 4.7.1

Name	Description	Availability	Cost
Eclipse	Integrated Development	Always Available	Free

	Environment		
Maven/Selenium	Testing software with browser control capability	Always Available	Free
Google Cloud Platform	Virtual Machine on the cloud	Always Available	$0.03/hr
Dylan	Project Group Member	10hr/week	Free
James	Project Group Member	10hr/week	Free
MySQL	Database Service	Always Available	Free
Java/PHP	Programming Languages	Always Available	Free

Section 4.8: Resource Skill List

Figure 4.8.1

Resource name	**Skill description**	**Where the skill will be applied**	**Reference to design components user stories**
Software engineering	Project design and testing	Throughout the project especially during planning and specification phases	Does not map to a specific user story
Fundamentals of Cybersecurity	Basics of cryptography and Linux file system	When creating the encryption scheme and when navigating Linux Virtual Machine	US-1, US-2
Internship	Backend Server Development	When setting up secure communication logic between the client and the server	NFR-5, NFR-6, NFR-7
Secure Systems/Ethical Hacking	PHP and Web application security	When creating and securing the server application	US-3, US-5
Database Management	MySQL and other database management principles	When creating and updating database tables on the server application	US-4, US-5, US-7
Fundamentals of Computer Science III	Programming and debugging in Java	When programming in Java for the desktop client application	US-2, US-3, US-5, US-6, US-7

Chapter 5: Testing

Section 5.1: Test Plan

We are going to test every component that we implement. We will be testing web browser compatibility by testing credential injection on Chrome, Firefox, and Safari. Load and stress testing will be performed on the server to gauge the server's ability to process large amounts of requests. Due to the increased security concern with password management, we will be testing specific vulnerabilities such as SQL Injections, IDOR-based attacks, Cross Site Scripting attacks, and Server-Side Request Forgery. We will also need to do testing on the input that the user enters during user registration and credential validation.

We will test the functionality of each component after the implementation of that component. At the end of each sprint, we will test the integration of the components with the rest of the system. By the end of the last sprint, we will have a working prototype of the entire project and plan to test against all our user stories and use cases. We will also test the application with test users and record feedback on the design and functionality of the service.

Section 5.2: Test Cases

Table 5.2.1: *TC-1 Login*

Name	TC-1 Verify login works
Type	Integration test
Description	To check that a registered user can sign in
Preconditions	User must have a registered account
Basic course of events	1. Enter username and password 2. Click the login button
Expected results	If the login was successful, the user is redirected to their main page, otherwise an error message is shown
Acceptance criteria	Successful login with correct credentials, unsuccessful sign in with wrong credentials
Cross-references	US-2, FR-1
Scheduled Sprint	Sprint 1

Table 5.2.2: *TC-2 Registration*

Name	TC-2 Valid registration

Type	Integration test
Description	To check that a user can register with their information
Preconditions	Running client
Basic course of events	1. Enter email, password, password confirmation 2. Select a security question from dropdown menu 3. Answer security question 4. Click the register button
Expected results	If the user entered valid parameters, registration will be successful, otherwise an error message is thrown
Acceptance criteria	Successful registration with valid parameters, unsuccessful registration with invalid parameters
Cross-references	US-1, FR-2
Scheduled Sprint	Sprint 1

Table 5.2.3: *TC-3 Server Controller Testing*

Name	TC-3 Server Request Testing
Type	Unit test
Description	To check that the server correctly responds to a specific request
Preconditions	Valid private/public key of the client application
Basic course of events	1. Gather all necessary parameters 2. Generate authentication token for the request based on the parameters 3. Send the request to valid destination
Expected results	If all the required parameters are passed and the authentication token is valid the request will return a 200 response with the valid data, otherwise the server will return an error response with an error message
Acceptance criteria	Correct responses to valid requests, invalid requests return an error response

Cross-references	NFR-6
Scheduled Sprint	All sprints

Table 5.2.4: *TC-4 Cross-Platform Testing*

Name	TC-4 Cross-Platform Testing
Type	Unit test
Description	To check that the client can run on multiple operating systems (Mac, Windows, Linux)
Preconditions	N/A
Basic course of events	1. Download client application on a specific operating system 2. Verify that it launches and performs correctly 3. Repeat on a different operating system
Expected results	The application client correctly performs across multiple operating systems
Acceptance criteria	Client correctly works on Windows, Linux, Mac
Cross-references	NFR-1
Scheduled Sprint	Sprint 1

Table 5.2.5: *TC-5 Client Correctly Retrieves User Website Credentials*

Name	TC-5 Client Correctly Retrieves User Website Credentials
Type	Integration test
Description	To check if the system can retrieve all the stored user login information
Preconditions	User must be registered and must have login information stored on the server
Basic course of events	1. Once the user navigates to the main window of the client application, the client sends a request to the server to retrieve the necessary credentials 2. Once the server responds with the user's login

	information, it is decrypted in the client, and displayed on the main page of the application
Expected results	The main window of the application correctly displays the user's credentials
Acceptance criteria	User credentials are retrieved
Cross-references	US-3, FR-3
Scheduled Sprint	Sprint 2

Table 5.2.6: *TC-6 Saving Website Credentials*

Name	TC-6 System Correctly Saves the Users Login Information
Type	Integration test
Description	To check if the system can successfully save the users login information
Preconditions	User must be registered and have login credentials for supported a website
Basic course of events	1. User clicks on the "Add New Entry" button on the main page 2. User is redirected to the "Add New Entry UI" 3. User enters all the necessary data 4. User click "Add" 5. Login information is saved and displayed on the main screen
Expected results	The user can successfully save their credentials in the system
Acceptance criteria	User credentials are saved in the system
Cross-references	US-4, FR-5
Scheduled Sprint	Sprint 3

Table 5.2.7: *TC-7 Full system test*

Name	TC-7 Full system test
Type	System test
Description	To verify that the system behaves correctly according to requirements
Preconditions	User must have the client application installed
Basic course of events	Go through all user stories and verify that the system successfully completes all of them
Expected results	The system behaves correctly according to user stories and requirements
Acceptance criteria	System behaves according to user stories
Cross-references	
Scheduled Sprint	Sprint 4

Chapter 6: Project Plan

Section 6.1: Sprints

Section 6.1.1 Sprint 1

- **Goal**: To set up the server side of the system
- **Starting date**: 12/17/2021 **Duration**: 6 weeks
- **Sprint master**: Dylan Cenotto
- **Demo goal**: Initial communication between all system components (UI, app backend, server, database)
- **Backlog**: **Table 6.1.1**

Table 6.1.1: Backlog for Sprint 1

Task #	Task description	Hardware/Software resources	People(s)	Estimated hours	Points difficulty	Cross-reference user stories, require-elements, design

T-1	Initial Server Setup - setting up the server in the GCP cloud, creating a github page for the server side	GCP server, Laptop, WSL (for script testing), Github	Dylan, James	6	6	Project Goals 3 and 4
T-2	Request authentication	PHP, PHPSecureLib (Cryptography library for PHP), RSA	Dylan	5	8	NFR-4, NFR-5, NFR-6
T-3	Set up communication between server logic and database	GCP server, MySQL server, PHP	James	6	6	NFR-8, NFR-11
T-4	Develop a server script that registers a user in the database	PHP, MySQL, computer	Dylan	3	7	FR-2, US-1
T-5	Develop a server script that signs the user in	PHP, MySQL, computer	James	2	3	FR-1, US-2

T-6	Develop a server script that adds a website to a particular user	PHP, MySQL, computer	James	4	6	FR-5, US-4
T-7	Develop a server script that retrieves all credentials associated with the user	PHP, MySQL, computer	Dylan	4	4	FR-3, US-3
T-8	Set up JAVA environment and set up github page for client	Java SDK, Eclipse IDE, Gihub, computer	Dylan, James	5	7	
T-9	Develop the request authentication logic	Java, Eclipse, computer	Dylan	5	8	NFR-6
T-10	Develop a way to send requests over HTTPS and read response from client application	Java, Eclipse, Java library for request transmission, computer	James	4	6	NFR-5
T-11	Develop and implement	Java, Eclipse, computer	Dylan	8	9	NFR-4

	logic for public/ private key generation unique to the user in the client application					
T-12	Develop a function that sends a request to register the user from the client application	Java, Eclipse, computer	Dylan	5	7	FR-2, US-1
T-13	Find a cross-platform UI library for Java that works on Windows, Linux and MacOS	Java, Eclipse, Internet	Dylan, James	3	7	NFR-1
T-14	Design and create UI-1.1 (Login Screen)	Java, Eclipse, UI library	James	6	7	US-2, FR-1, UI-1.1
T-15	Integrate UI-1.1 with corresponding backend functions (T-17)	Java, Eclipse, computer	James	2	5	US-2, FR-1, FR-2
Total hours necessary:				68		

Section 6.1.2 Sprint 2

- **Goal**: To set up the backend of the client and have it correctly communicating with the server

- **Starting date**: 01/31/2022 **Duration**: 4 weeks
- **Sprint master**: James Dempski
- **Demo goal**: To have most of the server controllers ready, with additional functionality in the app backend
- **Backlog**: **Table 6.1.2**

Table 6.1.2: Backlog for Sprint 2

Task #	**Task description**	**Hardware/Software resources**	**People(s)**	**Estimated hours**	**Points difficulty**	**Cross-reference user stories, require-elements, design**
T-16	Develop a server script that updates the user supercredential information	PHP, MySQL, computer	Dylan	4	5	US-8
T-17	Develop a server script that updates a website entry associated with the user	PHP, MySQL, computer	Dylan	4	6	FR-7, US-7
T-18	Develop a server script that deletes a website associated with a user	PHP, MySQL, computer	James	2	3	FR-7, US-7

T-19	Develop a function that sends a request to add a website to a particular user from the client application	Java, Eclipse, computer	Dylan	3	5	FR-5, US-4
T-20	Develop a function that retrieves all credentials associated with the user	Java, Eclipse, computer	James	3	4	FR-3, US-3
T-21	Develop a function that sends a request to update the users supercredential information from the client application	Java, Eclipse, computer	Dylan	4	5	US-8
T-22	Develop a function that sends a request to update a website entry associated with the user from the client application	Java, Eclipse, computer	Dylan	4	5	FR-7, US-7
T-23	Develop a function that sends a request to delete a	Java, Eclipse, computer	James	2	4	FR-7, US-7

	website associated with a user					
T-24	Develop a function to generate a secure password given user params	Java, Eclipse, computer	Dylan	2	4	FR-6, US-6
T-25	Integrate Selenium library for Java to open web pages and fill in data with necessary information	Java, Selenium library, computer	Dylan	5	6	FR-4, US-5, NFR-2
T-26	Design and create UI-1.2 (Registration Screen)	Java, Eclipse, UI library	Dylan	5	6	US-1, FR-2, UI-1.2
T-27	Integrate UI-1.2 with corresponding backend functions (T-16)	Java, Eclipse, computer	Dylan	2	4	US-1, FR-2
Total hours necessary:				40		

Section 6.1.3 Sprint 3

- **Goal**: To set up the frontend of the client and integrate it with the corresponding backend function

- **Starting date**: 03/01/2022 **Duration**: 4 weeks
- **Sprint master**: Dylan Cenotto
- **Demo goal**: To have most of the application backend ready, new UI pages integrated into application and account recovery functionality
- **Backlog**: **Table 6.1.3**

Table 6.1.3: Backlog for Sprint 3

Task #	Task description	Hardware/Software resources	People(s)	Estimated hours	Points difficulty	Cross-reference user stories, require-elements, design
T-28	Develop a server script that sends an email to the user with a recovery code for account recovery	PHP, 3d party emailing library for PHP	Dylan	6	6	US-8
T-29	Develop a function that sends a request from the client application to send an email to the user with a recovery code for account recovery	Java, Eclipse, computer	James	4	5	US-8
T-30	Design and create UI-1.3 (Main Screen)	Java, Eclipse, UI library	Dylan, James	8	8	UI-1.3, FR-3, US-3

T-31	Integrate UI-1.3 with corresponding backend functions (T-19)	Java, Eclipse, computer	Dylan	4	6	FR-3, US-3
T-32	Design and create UI-1.4 (Password generation screen)	Java, Eclipse, UI library	James	4	6	FR-6, US-6, UI-1.4
T-33	Integrate UI-1.4 with corresponding backend functions (T-24)	Java, Eclipse, computer	James	1	4	FR-6, US-6
T-34	Design and create UI-1.5 (Adding a new website)	Java, Eclipse, UI library	Dylan	4	5	UI-1.5, US-4, FR-5

T-35	Integrate UI-1.5 with corresponding backend functions (T-18)	Java, Eclipse, computer	Dylan	2	5	US-4, FR-5
	Unit and Integration testing	Eclipse Debugger, JUnit, Wireshark	Dylan, James	10		
Total hours necessary:				43		

Section 6.1.4 Sprint 4

- **Goal**: To finish developing the front-end, add supported websites and final testing
- **Starting date**: 04/01/2022 **Duration**: 4 weeks
- **Sprint master**: James Dempski
- **Demo goal**: To have the application ready for presenting
- **Backlog**: **Table 6.1.4**

Table 6.1.4: Backlog for Sprint 4

Task #	**Task description**	**Hardware/Software resources**	**People(s)**	**Estimated hours**	**Points difficulty**	**Cross-reference user stories, require-elements, design**
T-36	Design and create UI-1.6 (Adding a new website advanced)	Java, Eclipse, UI library	Dylan	6	8	UI-1.6, US-4, FR-5

T-37	Integrate UI-1.6 with corresponding backend functions	Java, Eclipse, UI library	Dylan	2	4	US-4, FR-5
T-38	Design and create UI-1.7 (Password recovery)	Java, Eclipse, UI library	James	4	6	UI-1.7, FR-6, US-6
T-39	Integrate UI-1.7 with corresponding backend functions (T-23)	Java, Eclipse, UI library	James	2	4	FR-6, US-6

T-40	Add as many supported websites as possible to the database and backend of client	HTML, MySQL	Dylan, James	14	6	
	Integration testing, final bug fixes	JUnit, Wireshark	Dylan, James	14	5	
	Extra time, in case team falls behind schedule		Dylan, James	14		
Total hours necessary:				56		

Section 6.2: **Risk Plan**

1. R1: Public and Private Key Generation unique to each user
    a. Designing and implementing a custom public/private key algorithm based on user password may be too difficult and resource consuming
    b. Likelihood depends on the depth of functionality of the cryptographic libraries for Java
    c. 4
    d. Develop a new encryption strategy together with our project supervisor
2. R2: Request authentication may be too slow
    a. Authenticating each incoming request may be too time consuming, significantly slowing down the speed of the application

b. Depends on the speed of the server and client app
c. 3
d. Develop a new authentication strategy using session tokens and maintenance

Section 6.3: **Estimated Financial Budget**

Item	**Estimated Cost**
GCP Virtual Machine Rental	$50.00

Section 6.4: Teamwork Plan
We will have weekly meetings throughout the development of the project to check up on progress and review goals for the next week. During the sprints we will have these on a daily basis to stay on top of the development during those periods. The group will also meet with their mentor weekly throughout the development of the project. The team will utilize Jira, a project management software, to keep track of task assignments and task completions. The team will also utilize Git for source code and version control. The Git repositories will be hosted online via GitHub.

List of References

[1] Study Reveals Average Person Has 100 Passwords, available at https://tech.co/news/

[2] Billions of passwords leaked online from past data breaches, available at https://www.techrepublic.com/

[3] Lost Pass, available at https://www.seancassidy.me/lostpass.html

[4] Open-Source Testing Tools, available at https://www.softwaretestinghelp.com/open-source-testing-tools/

[5] Google Cloud Platform Free Tier, available at https://cloud.google.com/free

[6] Getting Started with AWS, available at https://aws.amazon.com/getting-started/