National College of Ireland

Computing

Cyber Security

2020/2021

William Redmond Lawlor

x18106170@student.ncirl.ie

The Ace Manager

Technical Report

Contents

Executive Summary

This technical report is going to discuss how a password manager will be created using web languages, with the electron framework to create a desktop application. It will be targeting the current weaknesses of other password managers on the market.

Using the web languages, and framework, as discussed above it, is taking advantage of the strongest features to create a desktop application that has its information stored in the cloud because the application is on a desktop it is more secure than being stored simply in the web, and since the passwords are not stored in the cloud, and encrypted it is harder to access them. As with many other passwords, managers keep passwords in their system, or just in the cloud.

This application will enforce proper creation and storage of passwords and login methods with multi-ways to login with two-step authentication. Then how the passwords will be encrypted with the JavaScript libraries to create a vault on a MongoDB Atlas Database separating two key decoder factors from each other.

As well, it can only decrypt one password at a time and will use to key unique data, to be a session and tokens, to monitor the person's interactions on the application to continuously verify them as an authentic user of the application, like you would with a social media token or any other type of website. This also includes the removal of a master password to be able to recover access to all passwords at once, such as using the same codes as you would log into your computer to gain access to your passwords on google chrome.

1.0 Introduction

1.1. Background

This project has been chosen to be undertaken because it would pose a challenge to develop a password manager application. By integrating both frontend and backend coding, using frameworks together to create the application. This would make a clearly defined easy to use interface that would be able to create, edit and delete a password from the manager easily.

The motivation for this project started with the dissatisfaction with the current offers for password storage, such as google chromes inbuilt one, and others on the market. As with research, most people do not see the use of a password manager - ('Study: 27% of Users "Don't See the Point" of Password Managers' 2022). Examples of some serious databases included many of the popular ones today. –"In

2017, LastPass reported a serious vulnerability in its browser add-ons and asked subscribers to refrain from using it." ('Are Password Managers Safe to Use in 2022?' 2022) and again, in 2019 – "2019, serious vulnerabilities were found in the code of Dashlane, LastPass, 1Password, and KeePass. This applied to Windows 10 users and only if the right malware was installed. Once again, the users didn't suffer any reported casualties."
('Are Password Managers Safe to Use in 2022?' 2022).

Therefore, it is important to try an attempt to create a password manager by using the best qualities of each of the market values at the moment such as taking advantage of a cloud-based storage system and separating the decoders at the same time.

Primarily most of the key information is stored together in a vault where it remembers you, meaning that if a breach were to happen everything could be decoded at once.

While using the highest, and latest standards out there and available on the market right now. There are also many concerns which needed to be addressed, with the current level of password managers and how they are known/previously been hacked.

Identity is controlled through to process. The token will be stored within the application while the session is stored in the process, and only when both are present in the system can you bring down any important pieces of information.

So, the idea will be to create a desktop application that acts as the interface for being able to access everything stored within a vault on MongoDB linked to the user. Make sure that nothing is decrypted until requested by the user, and that a master password cannot decrypt all passwords at once. That everything is validated each step of the way. To use the latest web technologies to back up and strengthen the desktop application.

The goal to be achieved.

1. Make improvements to what already is on the market now.
2. Address problems with the storage of information on the market.
3. Make use of other products/frameworks to assist in the creation of a modern easily accessible application.
4. It poses a real challenge to create as to how you would securely store such password information for a user, and it will stretch the limits of previously attempted projects through

1.2. Aims

The aims of this project are the following:

1. At all times, it is a challenge to create, and it involves thinking about each step

clearly on how the project is approached. This will, create and improve all forms of coding, problem-solving, research, and evaluation.

2. A real learning experience, while it needs to be a challenge, should not be so challenging that it harms the learning that is needed.

3. Make a clear and precise graphically user interface, that is easy to use and understand for any user.

4. That it is storing the user passwords securely and separately from the main system, with a way to easily decode them.

5. The ability to be able to add, edit and delete passwords that are in the system directly from being able to view the passwords.

6. To make use of different languages and features together smoothly but don't realm the project so that it is concise.

7. Make an excellent login system and make ways for future-proofing the project.

8. Validation of each input into the system and restricting the way it interacts with other features in the project.

9. Adding Sessions and Tokens throughout the project to act as increased buffers for making commands within the system approved, and a needed requirement to gain access to all areas.

10. To have a completed project by the end of the timeframe.

1.3. Technology

These are the following technologies that will be used within the project:

Framework:

1. Electron.JS: is an open-source framework that allows for the development of a desktop application while using the following programming languages to create it.

- HTML
- CSS
- JavaScript

It is incorporated into the project using the Node.JS framework to be able to apply the packages together. It will be used to create the graphical user interface to be able to display to the user what is going on in the application. This is assisted by the chromium rendering engine.

This application allows for cross-platform compatibilities with minor changes that can be made to code that make it work across multiply operating systems. Acting as both a front-end and back-end side of things with the ability to process this different information through its render and main process.

2. Bootstrap: Is an open-source CSS framework, and will be applied to the pages to help

with the overall GUI of the application

Runtimes:

1. Node.JS: This is an event-driven application that is often used with web development, as the application's main bases are using web-based programming languages, it will help include the packages. Acting as the runtime environment for the application that can assist with cross-platform compatibilities of the application itself.

This will allow for many packages to be installed into the system that will be used within the application:

Package Name and Version:	Purpose:	NPM link:
@hapi/joi: 17.1.1	Validation for Input	https://www.npmjs.com/package/@hapi/joi
Axios: 0.26.0	Restful API interaction provides access to the internal API links	https://www.npmjs.com/package/axios
Bcryptjs: 2.4.3	Hashing login passwords and comparison of entered passwords	https://www.npmjs.com/package/bcryptjs
Boxicons: 2.0.9	Icons	https://www.npmjs.com/package/boxicons
Connect-Mongodb-session: 3.1.1	Database connection: Session's checking	https://www.npmjs.com/package/connect-mongodb-session
DOTENV: 16.0.0	Storage of key API details and other important key information, which does not go to production	https://www.npmjs.com/package/dotenv
Electron.js: 17.1.2	The main framework for GUI etc	https://www.npmjs.com/package/electron
Express: 4.17.2	Backend Server	https://www.npmjs.com/package/express
Express-Session: 1.17.2	Detects session and uploads to MongoDB for checking for the session.	https://www.npmjs.com/package/express-session
jQuery: 3.6.0	Extra JavaScript library for password display	https://www.npmjs.com/package/jquery
Jsonwebtoken: 8.5.1	Web-Tokens for authentication	https://www.npmjs.com/package/jsonwebtoken
MongoDB: 4.2.2	Access MongoDB through Node.JS and Express server	https://www.npmjs.com/package/mongodb
Mongoose:6.3.2	Creating the connection for the application	https://www.npmjs.com/package/mongoose

2. Chromium: This is the open-source web browser that electron uses in the runtime. It is developed and maintained by Google, so any issues that the chromium has, the electron.js will also have the issues. Making it important to always keep it up to date.

Database:

1. MongoDB Atlas: This is a NoSQL database program meaning it is document-oriented, that hosts its service that offers free clusters and paid clusters, which this application is using the free model subscription.

This allows for any storage of data, IE passwords, login details, and session storage, which can be recorded, encrypted in the application, sent encrypted through to the database and returned in the same functions to be decrypted while maintaining the encryption.

Programming Languages:

HTML: This will be used to design the structure of the web pages and help organise and separate everything equally into its sections.

CSS: The overall layout and structure of the pages, this acts as the user interface.

JavaScript: This will be the main controller for most of the applications, it is used to create any necessary functions within the system.

This will also include the jQuery library from JavaScript

Python: This will be used to create a script that can be called on by JavaScript from the application to be able to run. This script provides easier access to some features that could be more difficult to create with JavaScript files in files such as password generation.

Restful API/Server:

Express: Is the Node.JS module that can create a local server within the background that can be used to gain access to the database and offer token and session storage checking by using JavaScript to control it. This also counts as its framework within the application as well.

This will control all the routes of the applications, for login, creating the password, editing, and deleting passwords in connection with the database. This is the main controller for all system actions. It comminates with an electron to deliver results to the user.

Miscellaneous:

1. Visual Studio Code: The IDE that the application will be developed in.

2. GitHub: Storage of the code and keep track of the progress.

3. Postman: API testing and making sure everything works as intended, can be used to

check for the URL addresses within the application on the Express server that runs in the background.

1.4. Structure

In this following section the project's system will discuss the following:

1. The requirements section: This will focus on the explanation of the system and what would be required of the system to create and use it.

2. The functional requirements of the project: What will explain what is required to complete the project

3. The non-functional requirements of the project: What is not completely needed to be included in the project at this stage.

4. Use Case Descriptions and Use Case Diagrams: This will describe the requirements and functions through the descriptions and use cases in much more detail.

2.0 System

2.1. Requirements

There is a requirement for a Restful API to be built to act as the monitor for the following functions you see below.

It includes both a server and authentication routes, schema and other functions to produce the following as it is the connection from the application to the database as well as the organiser of the information.

1. The user will be required to Register and Login: The user when doing so needs an internet connection to do this because it will connect their account name to their passwords so that they can be pulled down and accessed in the environment.

2. They need to be able to do one of the following, use a USB or NFC, as part of their login requirements.

3. They need to be able to understand how you access the USB file or the NFC function on their device.

4. They will be able to see the application open into a new section/screen where they will be able to view their passwords.

5. They will then be able to see that they can select multiply options such as Password and Settings.

6. Password page will allow the user to be able to make a new password and add it to their account be it generated or created by them.

7. Profile page will allow the user to be able to view the settings of the application and make necessary changes to the account.

2.1.1. Functional Requirements

1. Create the application: Create the baseline for the application so that it can connect itself to the database using the express server so that it is possible to implement the other features of the application. Through the Electron.js framework.

2. Create the database: Create the NoSQL MongoDB Atlas, so that the data can be stored. Connection is established with the mongoose features within the Restful API.

3. Restful API: A server is created to act as the access point between the application and the MongoDB atlas access point, and an authentication file will control the routers for the application that will allow for all further functionality requirements to be able to work.

   E.G: Generate Password will be called in electron, via the button press, the API will pick up on the access route delivered to it via the electron function, and call on the file to do the requirements to generate the password and send it back to the user at the time of the press. Equal exchange of information.

4. Connection to the database: Add the connection to the database so that the information can be stored within the system. Via the Restful API

5. Login system: This system is used using express routes, in the background of the application and Axios to use the routes. The user will access the MongoDB database with their login details and gain access to their account upon successful checking of their key login in details and this will be used to open the other half of the application. This includes a username, password, USB or NFC to gain access. Via Restful API

6. Password view and storage: Once logged in the server will send the application the stored passwords in its system, while they are still

encrypted when the user goes to view them. They have unlocked indivisibly not altogether. Via Restful API

7. Password generation and adding: The user will be able to add a new password to the storage, by using the python file to create a generated password, and then add that password to the storage on the data server. Via Restful API

8. Password Editing: The ability for the user to be able to look at certain passwords and be able to edit or delete them from the database. Via Restful API

9. Hashing: Passwords are hashed within the system using Bcryptjs functions. Via Restful API with bcryptjs

10. Password Vault Encryption: create an encrypted vault with the application, that will send the passwords that are saved from the system to the database, in an encrypted way so that when they come back to the application, they are safe and secured by both the database and the application. Via Restful API

11. USB-Security Key: One of the login methods required to gain access to the system, which is a part of the login functionality is a separate sub-function.

2.1.2. Non-Functional Requirements

1. Change the Login methods: Change the password and Pin Code regularly and prompt the user.

2. Master Password: The ability to be able to set a master password to log into the system, for the user as a back system.

3. Forgot Password Feature: The ability for the user to be able to say they forgot their password and be able to change it within the login screen to be able to gain access to the passwords again.

4. Setting System: Modify the way the person logins, views the application and or uses the application.

5. Switching Screen: The ability for the system to be able to use separate window frames in the application. Login method appearance is different to the main window appearance and size.

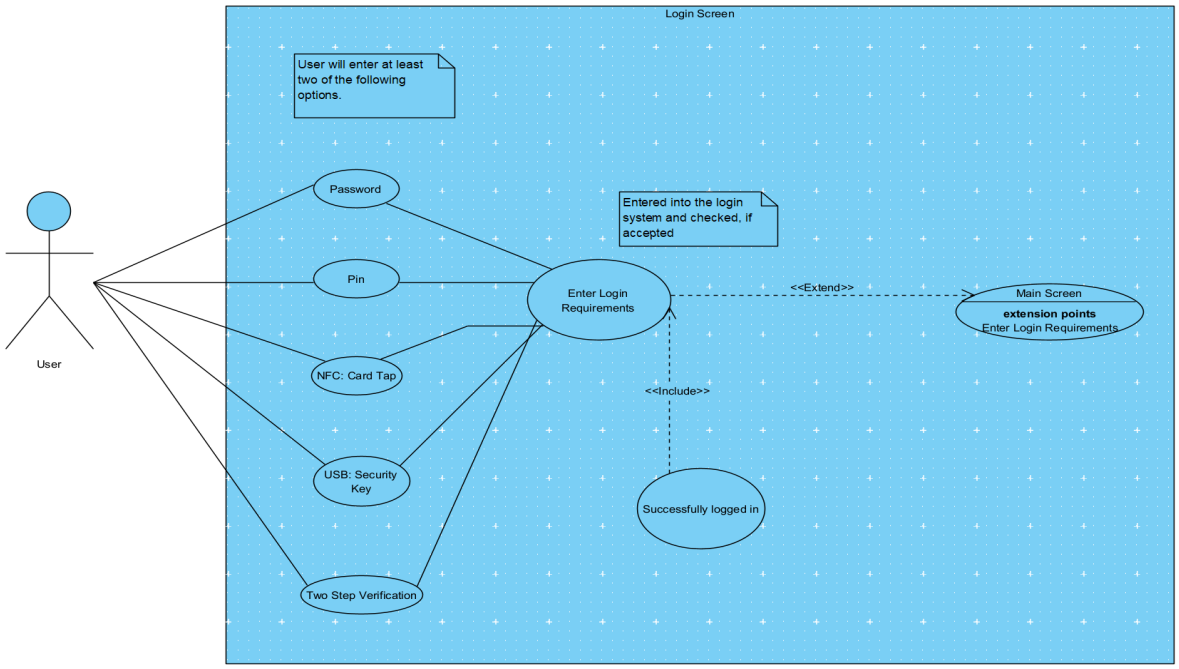
6. NFC Read/Write: This allows the user to create, read and write an NFC tag to be used in conjunction with the login system.

2.1.2.1. Use Case Diagram

Use Case Descriptions:

Requirement Number:	1
Requirement:	How a user Logins
Requirement Description:	Allows for the user to be able to log into the system and view their passwords
Description and Priority	How the user will log into the system, and what is there to protect the content of the details in the system.
Scope:	To display how the login system works as it connects to the server and displays the main screen to the user
Description:	What is required to log in to the application, and how the system checks that it is successfully login into the application.

Use Case Diagram:

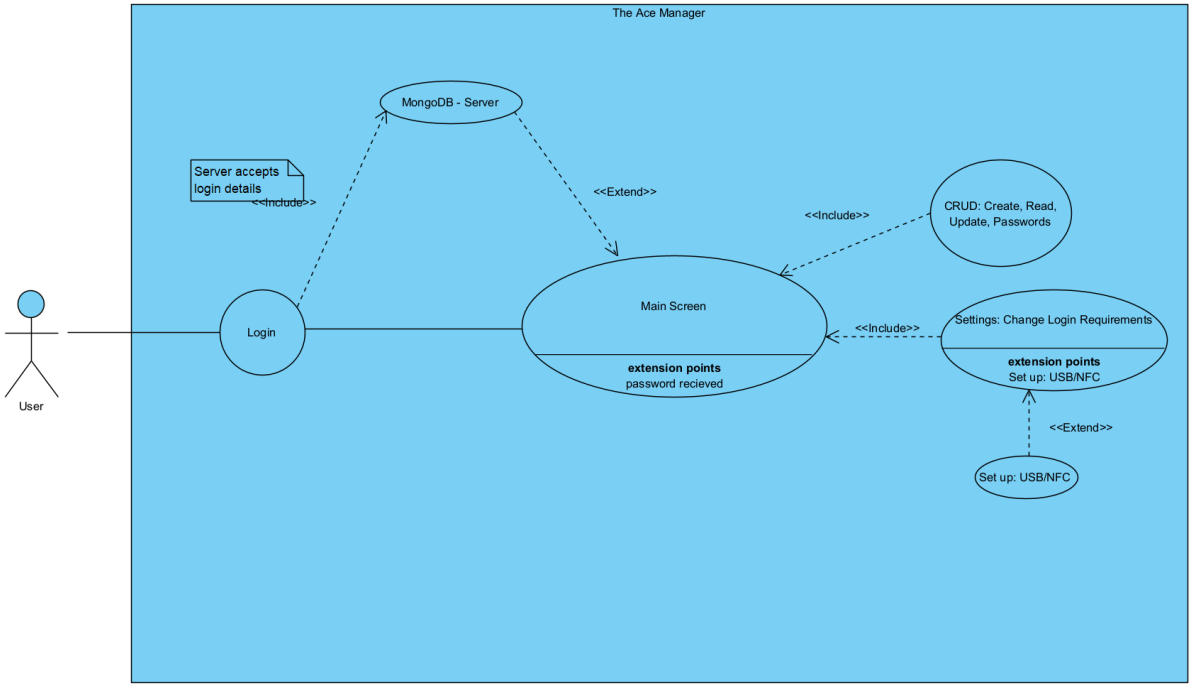


Flow Description	
Precondition:	1. The application is started up 2. Connected to the internet successfully. 3. The user has set up the application before to be able to log in to the system
Activation	When the user goes to enter their login details into the system.
Main Flow	1. The user enters two of the required passwords details to login the system, such as the password and pin code

	NFC – Card tag, with a password and Pin USB, with an NFC tag and a pin or Another combination 2. The System will then verify these details by comparing the information to the server information 3. The system will return to the user that they have successfully logged into the system and bring them to the main screen 4. The System will then pull down the passwords for that user, and the user will be able to view them 5. The User will now be able to copy their password to be used or view their passwords in the system
Alternative Flow	1. The user enters the wrong login details into the system. 2. The system checks the wrong login details against the user 3. The system will return that the credentials are wrong and ask the user to enter them again. 4. The user attempts to enter them again, and successfully enters the right one 5. The system informs the user that they have entered the correct details and will now be brought to the main menu
Exceptional Flow	1. The user enters the wrong details into the system 2. The system checks that the wrong login details 3. The user enters in the details again and fails to enter them correctly 4. The system checks again, and fails again as the details are wrong informing the user 5. The system counts up each failed entry and at three failed attempts, informs the user to try again in five minutes
Termination	The user is now able to see the main screen and gain access to their passwords to use at will until they log out
Post Condition	The system is now waiting for what the user wants to do next

Requirement Number:	2
Requirement:	Login into the system's main function
Description and Priority	The user login details are correct, and the server is producing their password.
Scope:	To display what is happening in the background of the login process to the main screen to view the passwords
Description:	The user has logged into the system and is now looking at their passwords on the main screen.

Use Case Diagram:

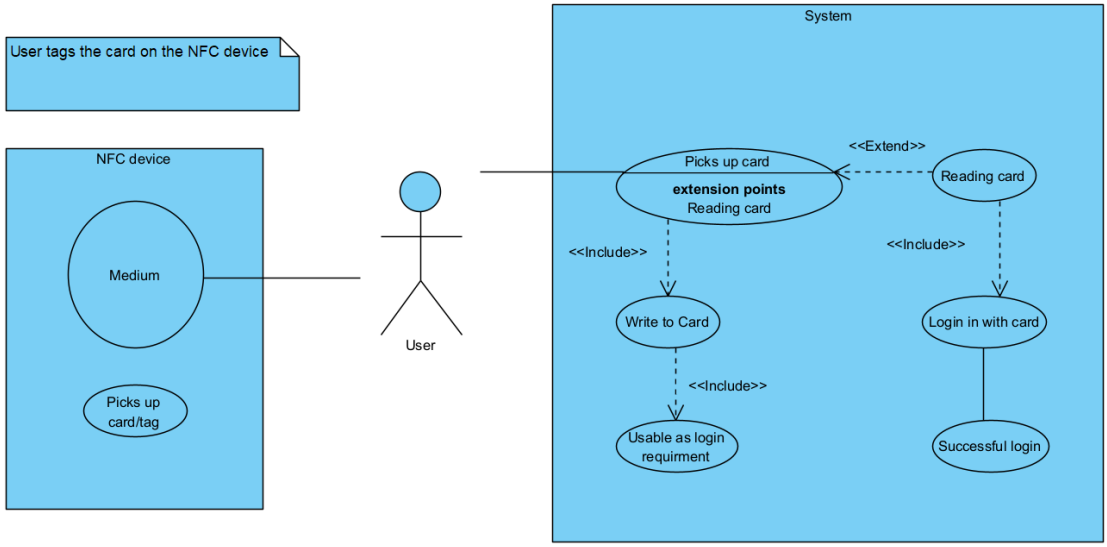


Flow Description	
Precondition:	1. The user has been able to connect to the internet successfully 2. The user logs in successful into the system with their details
Activation	The user has entered their details correctly into the system and would now like to view a password or make certain changes to a password.
Main Flow	1. The user has successfully logged into the system. 2. The system has successfully connected to the server and given the okay to proceed, requests the password from the server that is linked to the account 3. The user is now on the main screen viewing their passwords 4. The user now has access to the setting menu where they can make further changes to the login details or passwords

Alternative Flow	1. The user has failed to log in, so it is attempted again 2. The system stops them from login and doesn't call down the passwords at all 3. The user successfully logs in to the system, and the passwords are then called down for the user to see 4. They can access the settings menu and reset anything they wish too
Exceptional Flow	1. The user has failed to log into the system several times. 2. The system doesn't call down any passwords for the account 3. The system attempts to ask the user to wait before offering to reset the password
Termination	The user is now on the main screen looking at the entire application
Post Condition	The system is waiting for the user to proceed with an input

Requirement Number:	3
Requirement:	NFC – Setup and Login
Description and Priority	Set up the ability for the electron.js system and node.js to be able to pick up the NFC device on the computer and be able to interact with it for the user to be able to read and write to a tag on the system.
Scope:	Working tag in the system for the user with NFC
Description:	The user is set up and able to use the tag

Use Case Diagram:

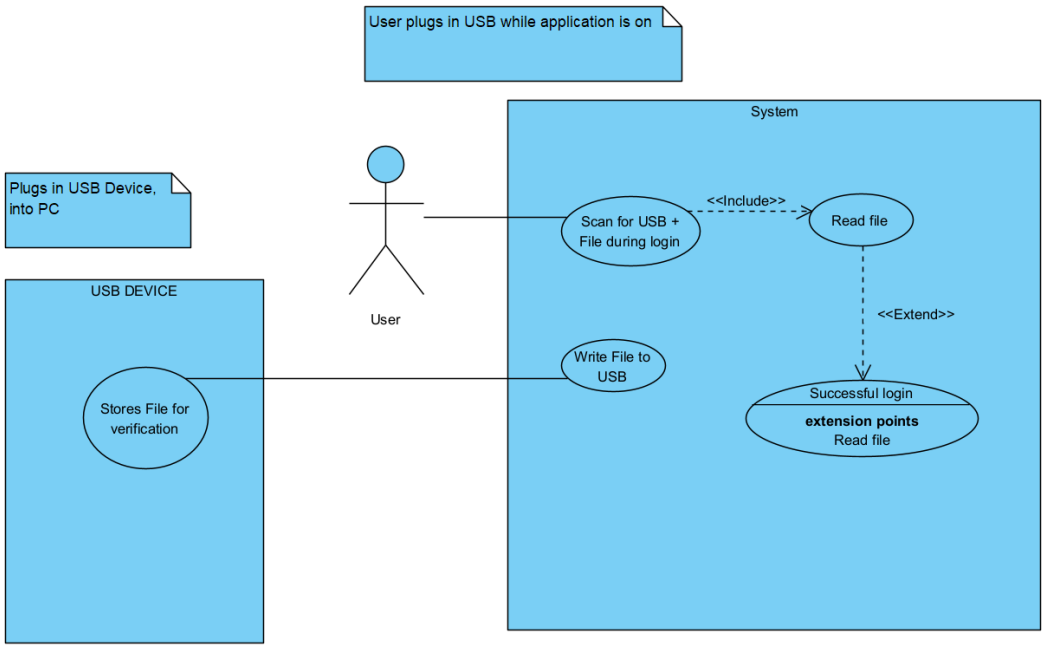


Flow Description	
Precondition:	1. The User is setting up the NFC card and wants to use it as a login detail that is connected to the system. 2. They have successfully logged into the system and have entered the setting menu
Activation	The user is attempting to set up the NFC on the system
Main Flow	1. The user is tapping the card into the NFC device and gets a response back 2. The system has picked up a new card on the system and is now writing a login detail to the NFC device 3. The system informs the user that they are now able to use it as a login requirement and are asked if they want to test it out 4. The user logs out of the system and enters login details one at a time, and taps the NFC tag when prompted by the system

	5. The system has now logged them into the system with the NFC tag
Alternative Flow	1. The user is attempting to set up the NFC tag, and is informed that it is not getting picked up and to attempt it again 2. The user attempts to tap the card again against the NFC device 3. The system picks up the card properly and informs the user that it was successful this time 4. The user can log in with the NFC card
Exceptional Flow	1. The user is unable to use the NFC function on the device and will have to troubleshoot what is going on 2. The system is informing them that they cannot use that function at the moment and to try again 3. The system does not allow the user to use the NFC function as a required login as it is not properly set up in the system just yet for that account user
Termination	The user is now able to use the NFC system within the application itself
Post Condition	The system is now in the main menu waiting for a response or logged out and ready to be logged into

Requirement Number:	4
Requirement:	USB – Setup and Login
Description and Priority	Set up the ability for the electron.js system and node.js to be able to pick up the USB device on the computer and be able to interact with it for the user to be able to read and write to a tag on the system.
Scope:	Work with a USB device to act as a login method
Description:	The user sets up a USB device to login into the system

Use Case Diagram:



Flow Description	
Precondition:	1. The User is setting up the USB and wants to use it as a login detail that is connected to the system. 2. They have successfully logged into the system and have entered the setting menu
Activation	The User has plugged a USB into the system, and now wants it to be used as the security key to log into the system
Main Flow	1. The user has placed the USB into the system for the first time in the setting menu when setting it up 2. The system asks the user to select which USB device they want to use and informs them of the risks 3. The user reads the warning and selects the correct USB they wish to use 4. The system now writes the key file/s to the USB and encrypts them, it is now set up

	5. The system asks if the user wants to go to the main menu or test out the login 6. The user picks one of the options
Alternative Flow	1. The user places a USB into the system for the first time, and attempts to make it a security key 2. The system asks for the user to select the key and inform them of the risks 3. The user selects the USB and reads the risks 4. The system attempts to write to the system and informs them it has failed 5. The user is asked to reselect the USB and attempt again 6. The system attempts again, and this time is successful 7. The system asks the user its options again, the main menu or log out and test it
Exceptional Flow	1. The user attempts to use the USB to create the files on the USB to act as a security key 2. The system attempts to write the files to the system and fails to write to it and informs the user to use a different USB after another attempt. 3. The user places a new USB key and selects it 4. The System writes to that USB successfully and informs the user
Termination	The user is now able to use that USB as a login method
Post Condition	The system is now in the main menu waiting for a response or logged out and ready to be logged into

2.1.3. Data Requirements

2.1.4. User Requirements

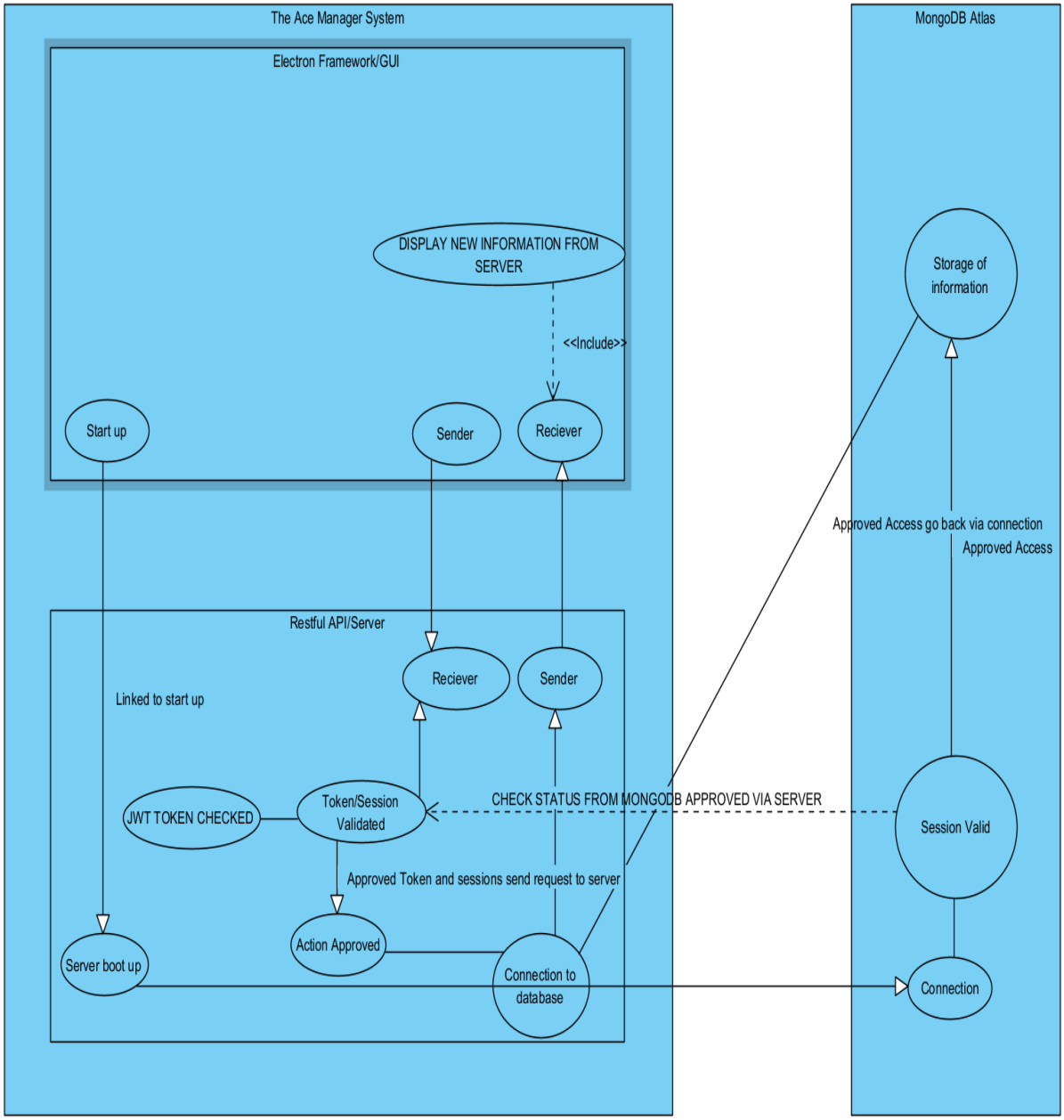
2.1.5. Environmental Requirements

Usability Requirements

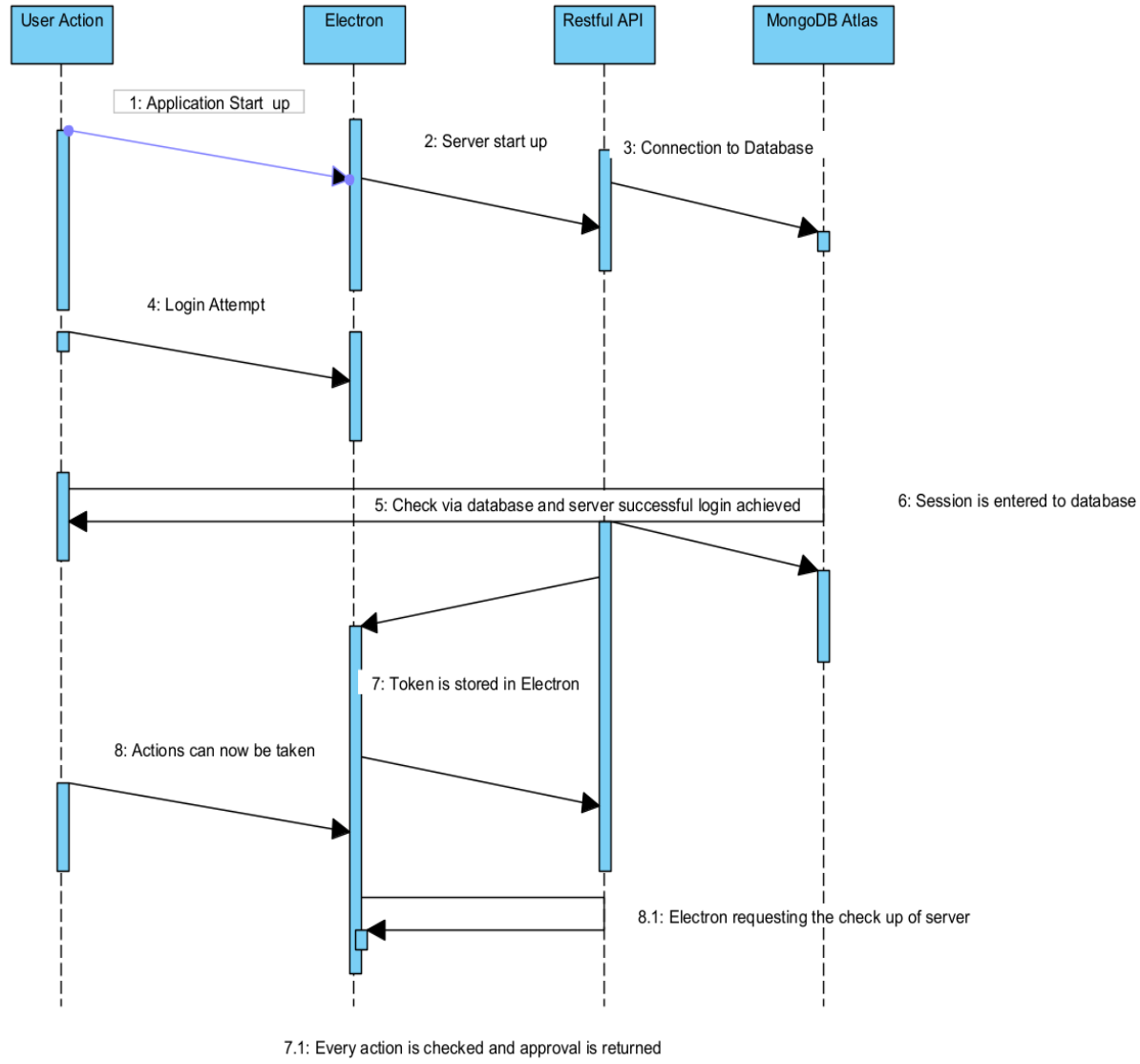
2.2. Design & Architecture

Design:

User case design



Sequence Diagram.



1. Electron.js: The job of this framework is to act as the interface and controller for the user, meaning the HTML, and CSS pages are displayed using the frame to create a window frame for them to be used in.

While using JavaScript to act as the middleware for the application, essentially acting as the middleman for the functions within the application.

So, when a button or action is being taken within the applications like you would on a website, it checks for what is to happen for the course of action to be taken and responds accordingly.

Architecture:

1. Electron: When it is turned, it displays the information calling the server upon

start-up to connect to the database. Then the two windows that are used, the main service and the login window, act as listeners for the actions on the application.

2. Express/Restful API: When the server is started upon a successful a session will be created and logged on the server to be checked at each point, as the server prevents access to other parts of the application until doing so.

It controls the login method, the ability to get at passwords, and any access around the site through its routing methods. It always checks for the token and authentication in the server, when accessing a point and will block the action completely.

It will encrypt the passwords using two methods, for login details it will use Bcryptjs, and for passwords, it will create the vault system to encrypt the passwords that will be sent to the database. It will then ensure that the application has the only method to decrypt the password at the time.

These actions are also waited to be detected by the electron.js server before beginning, meaning there is a tiny delay in some of the mostly unnoticeable actions.

It also makes sure to end all route actions upon completion of the task.

3. MongoDB Atlas: It stands alone from everything else other than verifying login details with the encryption and storing the passwords outside the application at the time.

2.3. Implementation

Password Generation: Python Script that takes one of a letter, upper case, and numbers as well as special characters and generates a code out of it. Sends it into a temporary list.

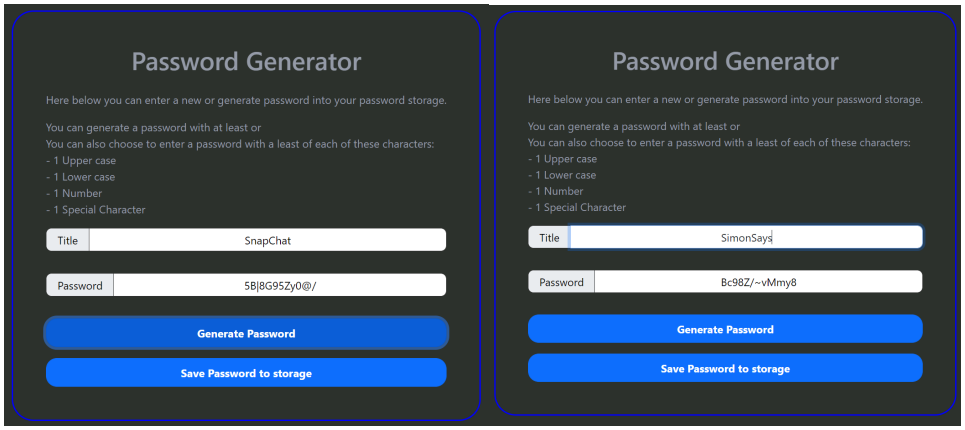
Code Snippet:

```
Max_Len = 12

Digits = ['0','1','2','3','4','5','6','7','8','9']
Lower_Case_Characters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'm', 'n', 'o', 'p', 'q','r', 's', 't', 'u', 'v', 'w', 'x', 'y','z']
Uppercase_Characters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H','I', 'J', 'K', 'M', 'N', 'O', 'p', 'Q','R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y','Z']
Symbols = ['@', '#', '$', '%', '=', ':', '?', '.', '/', '|', '~', '>', '*', '(', ')', '<']


Combined_List = Digits + Uppercase_Characters + Lower_Case_Characters + Symbols

Random_Digits = random.choice(Digits)
Random_Upper = random.choice(Uppercase_Characters)
Random_Lower = random.choice(Lower_Case_Characters)
Random_Symbols = random.choice(Symbols)

temp_pass = Random_Digits + Random_Upper + Random_Lower + Random_Symbols

for x in range(Max_Len - 4):
    temp_pass = temp_pass + random.choice(Combined_List)

    temp_pass_list = array.array("u", temp_pass)
    random.shuffle(temp_pass_list)

    password = ""

    for x in temp_pass_list:
        password = password + x

print(password)
```

Being called by the express server from the button calls on the electron.

```
router.get('/password', callPassword)

function callPassword(req, res) {

    var spawn = require("child_process").spawn;

    var process = spawn('python', ["src/PythonScript/password.py", req.query.password]);

    process.stdout.on('data',function(data){
        res.send(data.toString());
        res.end()
    })
}
```

Output:

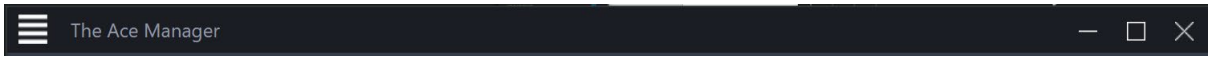


Frameless Window to create a custom title bar in the application:

By setting the frame to false, it removes the normal look of the application, this occurs when creating the Window class for the code:

```
//required variables to access file requirements.
const { app, BrowserWindow, ipcMain } = require('electron')
const ipc = ipcMain
const {dialog} = require('electron');
var fs = require('fs');
require('./RestfulAPI/server')
//Function declaration for the Parent Window IE the login screen
function createParentWindow(){

  const ParentWindow = new BrowserWindow({

      width: 420,
      height: 720,
      frame: false,
      webPreferences: {

        nodeIntegration: true,
        contextIsolation: false,
        devTools: true,

      },
  });

//calls the HTML file from the source folder for the Login
ParentWindow.loadFile('./src/HTML/Login.html')
```

Upon successful login, it will use a bar as seen below.



Here is the HTML code:

```
<body>

    <div class = "MainPart">
        <div class ="TopBar">

            <div class = "TitleBar">

                <button id="showHideMenus" class ="toggleButton"></button>

                <div class = "title">

                    The Ace Manager

                </div>

            </div>

            <div class="TitleBarBtns">

                <button id ="minimizeBtn" class = "TopBtn minimizeBtn" title = "Minimize"></button>
                <button id ="maxResBtn" class = "TopBtn maximizeBtn" title = "Maximize"></button>
                <button id ="closeBtn" class = "TopBtn closeBtn" title = "Close"></button>

            </div>

        </div>
```

JavaScript Code that provides functionality to the buttons:

These are the types of listening that the electron.js part of the application is out there listening for to take actions on the server.

```
ipc.on('closeApp', () =>{
  win.close()
})
ipc.on('minimizeApp', () =>{
  win.minimize()
})

ipc.on('maximizeApp', () => {
  if(win.isMaximized()){
    win.restore()
  }

  else{
    win.maximize();
  }
})

win.on('maximize', () => {
  win.webContents.send('isMaximized')
})

win.on('restored', () => {
  win.webContents.send('isRestored')
})
```

Express Server:

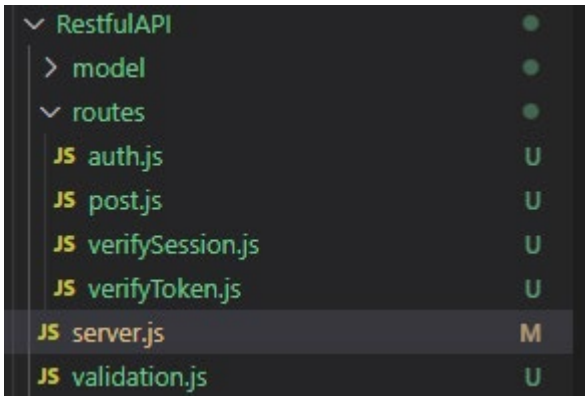
Here below we can see the requirements for each server. All the variables that have gone into creating the server.

```
//REQUIREMENTS
const express = require('express');
const session = require('express-session');
const app = express();
const dotenv = require('dotenv');
const mongoose = require('mongoose');
const MongoDBSession = require('connect-mongodb-session')(session);

//Import Routes
const authRoute = require('./routes/auth');
const postRoute = require('./routes/post');

// ENV file imports for application.
dotenv.config();
PORT = process.env.PORT

//Connection to DB
mongoose.connect(process.env.DB_CONNECT,
()=>console.log('Connected to MongoDB'));

//Start Session
//Creating Storage Session
const store = new MongoDBSession({
    uri:process.env.DB_CONNECT,
    collection: 'MySessions',

});
```

Next is the start-up:

```
//Session middleware
app.use(session({

    secret: 'HiddenCodeForCookie',
    resave: false,
    saveUninitialized: false,
    store: store

}));

//middleware
app.use(express.json());


//Router Middleware
app.use('/api/user', authRoute);
app.use('/api/post', postRoute);

app.listen(PORT, () => console.log('Server is up and running'));
```

API Folder Path structure:



Session Verify code:

```
async function verifySession(req, res, next) {{

    const CheckSession = await MySessions.findOne({SessionID: NewSessionID});
    if(CheckSession.Authenticated){

        next();
    }

}};
```

When this function is called from the system it targets Sessions ID that is created upon successful login of the system and then tells the system to move on from it. Except the session is logged to the server and upon exiting the app, the session is deleted from the database.

This is the same with the token.

Token Verify Code:

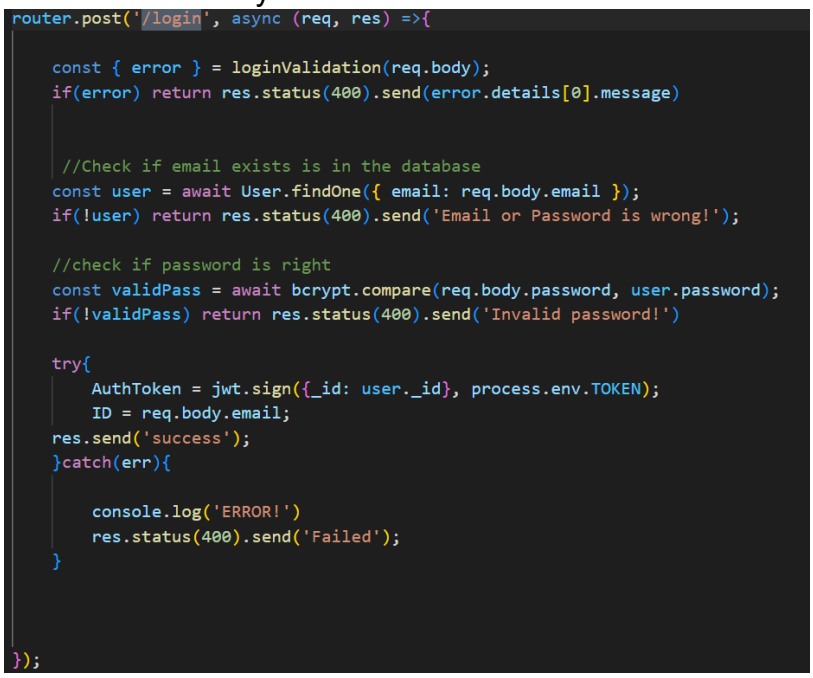
```
async function verifyToken(req, res, next)
{{
    if(!AuthToken) return res.status(401).send('Access Denied');

    try{
        jwt.verify(AuthToken, process.env.TOKEN);
        next();
    }catch(err){
        res.status(400).send('Invalid Token')
    }

}}
```

Checks for the JWT web token in the application with the Electron system.
An example of what these functions look like within the system.

```
router.post('/addPassword', verifySession, verifyToken, async(req, res) =>{
    const{passwordTitle, password} = req.body;
    const hashedPassword = encrypt(password);
```

the route in which is created for /add Password is followed by the two functions to be checked out first then they will be accepted into successfully adding the password.

Here is where they are created.



```
router.post('/login', async (req, res) =>{

    const { error } = loginValidation(req.body);
    if(error) return res.status(400).send(error.details[0].message)


     //Check if email exists is in the database
    const user = await User.findOne({ email: req.body.email });
    if(!user) return res.status(400).send('Email or Password is wrong!');

    //check if password is right
    const validPass = await bcrypt.compare(req.body.password, user.password);
    if(!validPass) return res.status(400).send('Invalid password!')

    try{
        AuthToken = jwt.sign({_id: user._id}, process.env.TOKEN);
        ID = req.body.email;
    res.send('success');
    }catch(err){

        console.log('ERROR!')
        res.status(400).send('Failed');
    }


});
```

Login methods, the information is first validated and validated at the point of the HTML entry, using email input field and password input fields, checks that the email

exists in the database, then if the password is right followed up by the token creation. A different process oversees checking the sessions.

```
async function SendPostLogin(){
    var Email = document.getElementById('EmailID').value;
    var Password = document.getElementById('EnteredPassword').value;

    var body = {email:Email, password:Password}

    axios.post('http://localhost:3000/api/user/login', body, axiosConfig).then(response =>{
        if(response ="success"){
            ipc.send("SwitchWindow");
            StartSession();
        }
    })
    .catch((error)=>{
        document.getElementById('FailedLogin').innerHTML ="You have failed to login!"
    })
}

async function StartSession(){
    const SESSION = {
        method: 'GET',
        url: 'http://localhost:3000/api/user/SessionStart',
    }

    await axios(SESSION)

}
```

Upon the acceptance of the details, session start() is called to access that point and create the sessions; and the route for /Login creates the JWT Token.

```
router.delete('/logout', verifySession, verifyToken, async (req, res) =>{

    try{

        req.session.destroy();
        await MySessions.deleteOne({SessionID: NewSessionID});

    }catch(err){
        res.send(err)
    }

});
```

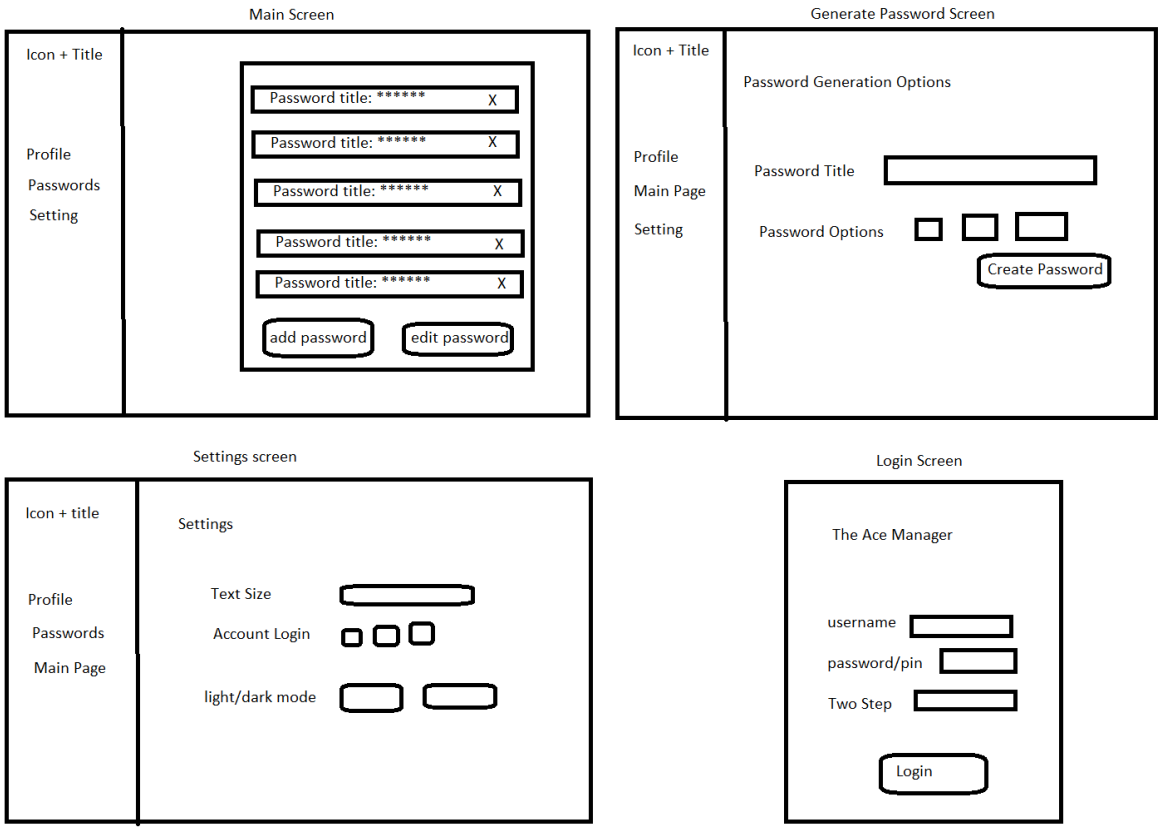
Upon logout, any session stored with the electron is deleted, and any sessions on the database are found and deleted.

Example of one of the schema

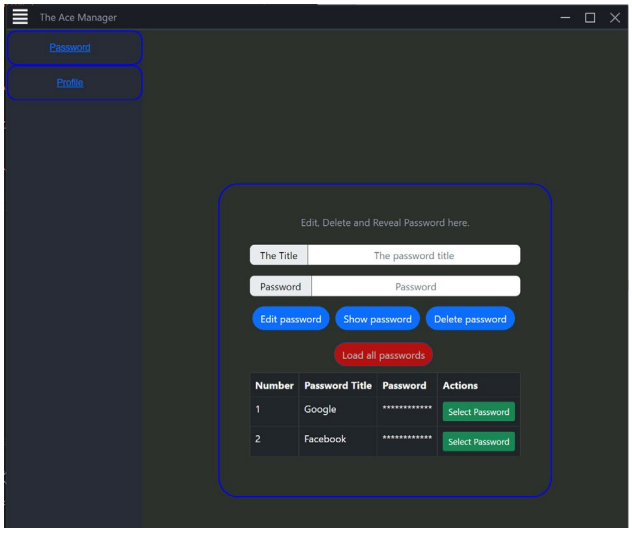
```
const userSchema = new mongoose.Schema({
    name:{
        type: String,
        required: true,
        min: 3
    },
    email:{
        type: String,
        required: true,
        unique: true,
        max: 255,
        min: 6
    },
    password:{
        type: String,
        required: true,
        max: 1024,
        min: 8

    },
    USB:{
        type: Boolean
    },
    USBSAVE:{
        type: String,

    },
    NFC:{
        type: Boolean
    },
    date: {
        type: Date,
        default: Date.now
    },
});
```

2.4. Graphical User Interface (GUI)

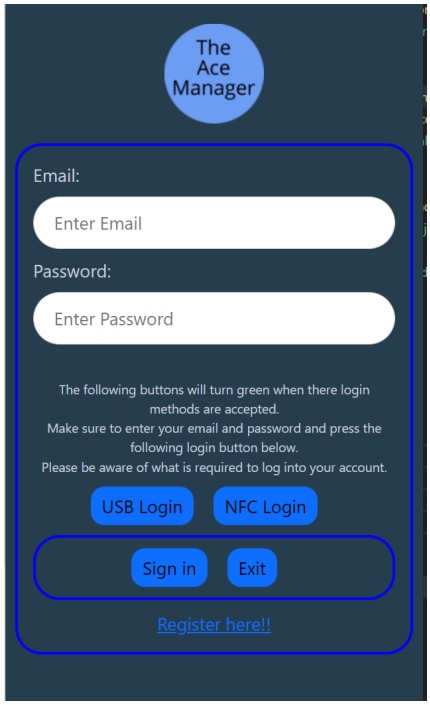
Initial Mock Up Design:



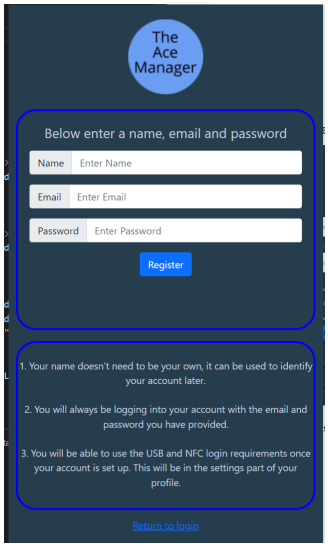
Final Version: Main Screen



Final Version: Login Screen



Final Version: Register Screen


The Ace Manager
Below enter a name, email and password
Name
Enter Name
Email
Enter Email
Password
Enter Password
Register
1. Your name doesn't need to be your own, it can be used to identify your account later.
2. You will always be logging into your account with the email and password you have provided.
3. You will be able to use the USB and NFC login requirements once your account is set up. This will be in the settings part of your profile.
Return to login

2.5. Testing

1. Writing Test case scenarios and verifying them.

Test Case ID	Test Scenario	Test Case	Pre-Conditions	Steps	Test Data	Expected Result	Post Conditions	Actual Result	PASS/ FAIL
TC_Login 01	Verify Login	Enter a Valid email and password	Registered Account	1. Enter Email 2. Enter Password 3. Click Login	Valid Email and Password	Successful Login, the screen switched.	The server is still running in the background. Token and Session are created and validated	Successfully logged in with token and session in the database.	Pass
TC_Login 02	Verify Login	Enter one invalid condition	Registered Account	1. Enter a valid email, 2. Enter an invalid password 3. Click login	Valid Email, Invalid Password	The prompted text informs of failure to log in	The server is still running in the background waiting for the correct entry	Rejected access to application	Pass
TC_Login 03	Verify Login with additional login	Enter valid email and password, and USB or NFC details	The account is required to have USB And NFC set up after registration.	1. Click USB/NFC, and let them go green. 2. Enter a Valid email or password 3. Click login	Valid Email and Password Approved USB/NFC login	Successful Logged into the application	The server is still running in the background, Token and Session are created and validated	Successfully logged in with token and session in the database.	Pass

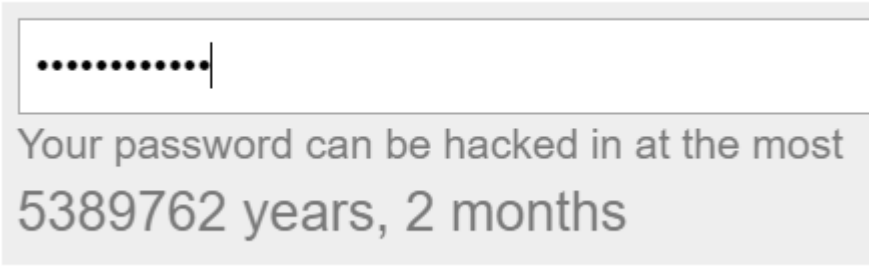
Test Case ID	Test Scenario	Test Case	Pre-Conditions	Steps	Test Data	Expected Result	Post Conditions	Actual Result	PASS/ FAIL
TC_Login 04	Verify Login with additional login	Enter an account with USB and NFC requirements without them	Registered Account and USB and NFC set up	1. Enter Email 2. Enter Password 3. Click Login	Valid Email and Password	Failure	Attempt again	Failure	Pass
TC_ Registrati on 01	Register Account	Enter a valid Name, Email, and password	No previous account with the same email	1. Enter a valid name 2. Enter a Valid email 3. Enter a valid password 4. Click the register button	Valid Name, email and password	Registered to database	Can now log into the system	Registered to database	Pass
TC_ Registrati on 02	Register an account with the same email	Enter valid name, used email and password	An account with the same email	1. Enter a valid name, 2. enter invalid/used email 3. enter a valid password 4 clicked the register button	Valid Name, invalid email, and valid password	Cannot use this email, please try again	Not registered	Not registered	Pass

TC_AddPassword_01	Logged in users add a password to the system	Add a password to their account	Logged in with a valid account, token and session approved	1. Login successful, access the Password page	Enter valid title and password	Added to database	Added to database	Successfully added and viewable	Pass
Test Case ID	Test Scenario	Test Case	Pre-Conditions	Steps	Test Data	Expected Result	Post Conditions	Actual Result	PASS/ FAIL
TC_ANY ACTION ON SITE SESSION TOKEN STATE	Logged in user without valid token or session	User attempts to access a password/create a password or delete a password etc any action	Logged in, but session and token are now invalid	1. logged. 2. Remove sessions from the server 3. Token has been refreshed.	Any Action	Rejected result	Forced to close the application	Rejected and forced to log out	PASS
TC_ANY ACTION ON SITE SESSION TOKEN STATE	Logged in user with valid token or session	User attempts to access a password/create a password or delete a password etc any action	Logged in, with valid session and token are valid	1. Logged in and look around for what they want	Any Action	Allowed Action	Gets intended results	Got intended action results	PASS
TC_Load Password	Logged in user goes to search for a password	Password is viewed successfully.	Logged in, and has registered passwords in the database	1. Logged in successfully 2. Has valid session and token ID	Valid Tokens and Sessions.	Requests titles from the database and loads them onto the table	Retrieved all password titles.	Gotten passwords	Pass

	and view its details			3. Clicks load passwords					
Test Case ID	Test Scenario	Test Case	Pre-Conditions	Steps	Test Data	Expected Result	Post Conditions	Actual Result	PASS/ FAIL
TC_Edit Password	The selected password or title is edited in the system.	Changing the password via inputs into the system	Logged in and has load password titles and selected one. The button has been enabled Has passwords in the system	1. logged in and has a valid session and log 2. selected and loaded passwords in the system 3. clicked edit password	Valid Token and log in, Valid new password and title	Updated in the system.	Updated in the system	Updated in the system	Pass
TC_Delete Password	The selected Password title is being deleted.	Removal of password from the system	Logged in and has load password titles and selected one. The button has been enabled Has passwords in the system	1. Logged in 2. Valid Sessions and Tokens 3. selected password to be deleted 4. clicked the delete button	Valid token and session alongside valid logged in credentials	Found, and delete the password, and removed it from the table	No longer found in the MongoDB database	Deleted from the system and row	Pass

Tried out the python script in the terminal of the application as Node.js can add a python shell into the dependency's files.

Then check out the results vs how long it takes to hack this password.



('How Long Would it Take to Crack Your Password? Find Out! - Randomize' 2022)
The password however is hashed out on the website, but the above score would be good for the password as a result. For the basis of that testing.

1.1. Evaluation

The end-user will be provided with a product that is easy for them to use, as upon opening the application it will deliver all the functionalities that have been set out in the above document sections. From initial testing, the application is already quick and responsive at the present, and with further development is likely to keep up with this.

2.0 Conclusions

The Advantages:

1. The systems architecture allows for the exchange of data IE: Passwords to happen only when it has been requested by the user. For example, when you open google chrome and store passwords there, you can only see that within the app data of the system and back them up to a new chrome browser if so desired.

When you open the ace manager, it brings you to the main page after login but does not load the passwords until it has been requested by the user. This means that if the user just wishes to store new passwords, they never load their passwords even as temporary data, or even see the passwords in the system.

2. As the main is a programming language used is JavaScript – with the use of HTML and CSS, you can store the token in a browser however because Electron JS will store those tokens in the local app data, it would be possible to be able to create forgery tokens, however, this is avoided with three features.

Creating unique data, one JWT token, and a session, at the point of login, each button click, has a verification function which checks the status of the session and Token stored within the MongoDB database, which will also delete them upon logout, meaning that they are separated and need active data on the server linked to their account to tor account.

Acting as a barrier for the user and an extra layer of protection.

Acting as a barrier for the user and an extra layer of protection.

3. Each password is only decoded in the system, upon being requested by the user to do so. As well, when selecting and loading the password to be revealed the system does not decrypt the password or actively store it, a list of encoded stars is put in their places for visual effect. Upon request to be decrypted, it finds the matching Password Title, and Username in the system and then decrypts using the secret code placed in the .env file.

4. Electron.JS Framework to be worked with multiply JavaScript and NPM libraries that there are many ways to be able to develop and use the project with careful planning and understanding. Which can also help improve

5. It is also possible to completely take this project and plug it into an Azure Microsoft server, or Amazon service and use them as a host alongside their database with very minimal changes to the overall JavaScript system and have a fully-fleshed-out website, and cloud storage again at the same time.

Meaning that if you wished to integrate into a web service as well, you can have both at the same time and any maintenances required for them, can be done at the same time, mostly the same development team if that was a route you went down.

6. With the NFC and USB systems, a master password is not as necessary to be able to recover certain pieces of information

7. Won’t allow for automatic sign-on into your (Changes, 2021) Won’t allow for automatic sign-on into your chosen activity, as well the application will be free to use, and unlock other password managers which cost money such as 1Password.

8. Node.js: Highly Scalable - (Kopachovets, 2020), as it is event-based, it increases speed and performance in many areas, opening the market for developers to draw on a pool of knowledge from other developers. (pokusew, 2021), integrates well with Electron.js

<u>The Disadvantages</u>:

1. Electron has a high demand for computer resources meaning that lower specs devices will increase the processing time for the application.

2. any issues involved with any of the services, will also be the same weakness of the application meaning that it, itself is avoidably a singular failure point, which can be the biggest and hardest issue to tackle when building any type of electron application.

Node.js: Limited by CPU, (Kopachovets, 2020) and call back can place the application into a loop, by mistake causing it to crash. With an overwhelming amount of access to modules, it can be hard to vet each of these modules and determine if they are what you need to be able to determine their effectiveness in your application.

With both combined, it shows that there is heavy ram and CPU usage of the application, which means low-end pcs will struggle to be able to use them.

3. If the user does not change their passwords regularly, they will not cause issues if any one of their passwords is compromised.

4. Electron requires a lot of in-depth knowledge of both web-based languages, and how the framework itself works to take full advantage of it.

5. Requires consistent updates of every package regularly.

The Limitations:

1. Electron.js and Node.js are updated regularly, so it is required that the application also be updated when this is updated so, this can present problems if any major changes are implemented into these features.

The application itself is limited by what it is coded with, while multiple languages can be used with it, you have to be careful with how it will perform.

2. Not everyone will have access to the same tools to advantage of the login system. Especially with the high cost of any NFC system to read the cards.

3. any issues as subject to what is available on the market right now from the Node.js Modules/dependencies; a lot of applications, but there is a big community behind it. Electron.js also works well with a fully experienced development team behind it as people being able to work on multiply-problems at once can get the system working faster.

Limited by how your approach to JavaScript, and the use of other languages.

There are many ways to build the electron.js application such as with React, and TypeScript. Narrowing down how you want to be the application can require certain things to be able to move forward.

Also limited by how the person intends to use the application, will they be using it effectively?

3.0 Further Development or Research

1. Bio-metric Login System

The best approach to this type of application with more time and resources is to build a fully integrated biometric login system. One of the better ways to do this will be to mimic the styles other banking applications have to allow for logging in. They offer the normal protection applied to the phone and then act as approval methods for login into the system.

Face and fingerprint identification would help boost the security of the system overall. Acting as an additional safeguard to the user/application. You could use using your phone to push for one of these options and then apply the push to approve the login.

However, because of the security requirements for a biometric system, you have to create a singular database for being able to store that information and add even more additional safeguards to be able to protect that data as it is extremely personal data that will be stored in comparison to a username, password, etc. So, it will require much more attention and precautions than a regular database. This main point to add to this is that this is not a one-person job as well and requires a lot of foresight in the development of the system.

2. Mobile Phone Application

This will allow for the users to also be able to access their passwords on the go as they go about the place, and as it is a personal device, following the correct development path will allow for increased security, as said above, you will be able to link it together for security approval, and include other key features. Such as key access to bio-metric add-on modules.

3. USB security key improvements

More research and development into fully functional keys and a secure key as a product for the application, there are several similar USB keys on the market now for this such as the Yubico Security Key, they are built-in with their security features rather than just uploading a single key file onto the USB to be selected by the user.

4. Redevelopment of the application

Given the level of development that is needed to be put into the project, it would be wise to employ ass team of developers to be able to build this as a product.

4.0 References

Bibliography

Akbarieh, E., 2021. *Introducing the Electron.js framework, its advantages and disadvantages.* [Online]
Available at: https://ded9.com/introducing-the-electron-js-framework-its-advantages-and-disadvantages/
[Accessed 20 12 2021].

Baocang, 2021. *node usb detection.* [Online]
Available at: https://www.npmjs.com/package/@baocang/node-usb-detection
[Accessed 20 12 2021].

Changes, E. B., 2021. *Breaking Changes.* [Online]
Available at: https://www.electronjs.org/docs/latest/breaking-changes
[Accessed 20 12 2021].

Driver, M. -., 2021. *MongoDB NodeJs Driver.* [Online]
Available at: https://www.npmjs.com/package/mongodb
[Accessed 20 12 2021].

Expert, 2021. *The Pros and Cons of Password Managers.* [Online]
Available at: https://expert.services/blog/managing-your-website/security/password-managers
[Accessed 20 12 2021].

Expert, 2021. *Top 5 most common passwords.* [Online]
Available at: https://expert.services/blog/managing-your-website/security/weak-passwords
[Accessed 20 12 2021].

Foundation, O., n.d. *Electronjs.org.* [Online]
Available at: https://www.electronjs.org/
[Accessed 20 12 2021].

Foundation, O., n.d. *Node.js.* [Online]
Available at: https://nodejs.org/en/about/
[Accessed 20 12 2021].

Kopachovets, O., 2020. *How to Benefit From Using Node.js for Your Next Project?.* [Online]
Available at: https://procoders.tech/blog/advantages-of-using-node-js/
[Accessed 20 12 2021].

L., S., 2021. *Electron.js: Create Cross-Platform Desktop Apps Using Web Technologies.* [Online]
Available at: https://www.cleveroad.com/blog/electronjs-framework
[Accessed 20 12 2021].

MongoDB, 2021. *MongoDB Data Encryption.* [Online]
Available at: https://www.mongodb.com/basics/mongodb-encryption
[Accessed 20 12 2021].

pokusew, 2021. *nfc-pcsc.* [Online]
Available at: https://www.npmjs.com/package/nfc-pcsc
[Accessed 20 12 2021].

5.0 Appendices

5.1. Project Proposal

6.1.1 Objectives

This project will set out to achieve an easy-to-use desktop application, that focuses on the generation and storage of a password for the user. The application will be created using an open-source software framework called Electron.JS, a service that allows the emulation of a desktop application using web-based technologies.

It will provide the user with a way to log into the system using multi-factor authentication. One of the authentications must include a bio-metric form of logging in. Such as facial recognition or fingerprint. The second will be a password or pin code, with the user being able to choose which one they wish to use. This will also include the development of a USB, that can be plugged into the device to act as a password as well.

When the user wishes to store and or generate a password, they will be able to label it or add a link to it, for the password. If the password is generated it will follow the guidelines of using passwords made set for the requirements of the website as well as following the advice of NIST. While an entered password will be checked for a meeting of requirements (L., 2021)and inform the user of its strength, as well as if it came as another password, it would recommend changing it. These passwords will be stored in an SQL database that is encrypted and offers features to stop hackings from accessing this data. Highly encrypted password storage

6.1.2 Background

I choose to undertake this project because I wanted to put my skills to the test using different technologies in conjunction with each other. As well, I wasn't completely happy with the range of password managers that are out there currently. I want to be able to make one that I would be happy to use daily as well as anyone else. That the application also wouldn't be related to google or an anti-virus program. As well the best forms of password managers come in the form of a desktop application. Which I feel confident I can program; this type of application will be able to include bio-metric login which I am very interested in programming and adding to this project.

I will meet the objectives I have set out in section 1.0, by taking it one set at a time. Evaluating and researching what needs to be done to achieve each step of the project. Regularly evaluate each stage at which I am at in the project and what features should be adapted and changed or even completely removed from the project. I believe using an Agile methodology will be the best approach to achieve it, as this project will be

needed to be in stages to be completed. As well regularly research what methods are needed for encrypting data and what are the current tools at the time for dealing with data security, as this data will always need to be protected for the user. I will also need to try slowly approach this project and not be quick about it, as I do not want to overlook anything in the objectives of this project.

6.1.3 State of the Art

Similar applications that exist to my project idea are:

Google Password Manager: This stores and also generates a password for you when you are filling out forms/login into a website with its auto-complete feature. This however stores your password in your browser and can be accessed by using the same password you would use to log in to your desktop or laptop. This particular feature I want to avoid as it is just a single password or pin to get access to these features.

LastPass: This is a recommended application password manager that uses multi-factor authentication, and is free to use, however, it is an outdated desktop application with its website being known to be hacked in 2015.

My project will differ from the above by using newer technologies and the current practices out there for the protection of data, including using multi-factor authentication that includes bio-metric forms of login in and exclude a singular method of logging as well as not many other applications out there are using Bio-metric login methods. So, this is how I believe my project will stand out from others; as well I don't want any access to the web currently in this project so while many of the applications offer form filling, I will not be using them currently; they are not completely secure as it is possible to look at them with a script on a website.

6.1.4 Technical Approach:

The approach I will take for this project in its development will be the scrum methodology; as I am very familiar with it, and it will help be able to break the project down into approachable steps that I want to be able to evaluate where I am daily in the project, what needs to be done that day, what is the objective I will be at by the end of the week then what the project will look like in a one or two week period. Then be able to realise during this sprint, where I will hopefully be during the end of the month and see where I need to make changes for the next sprint, and or the changes I need to make to the scope of the project so that I am happy with what I am going to be doing throughout the project.

Then hopefully will be able to identify my requirements through this development approach as I will need to adapt as I go on through the project. I will be able to identify the requirements by breaking down each step by the features I wish to include in the project, and then figure out what each of these needs to be coded/created in the project as a whole. Then be able to identify the steps required to be able to protect them from being accessed by outside sources. As the project includes a database and uses an open-source framework, I will need to be able careful when using these features so that I implement the protection features as I go along to improve my abilities to be able to test the project every month.

Breaking down the requirements into tasks will require me to objectively look at the project requirements after I have created them. So that I can break them down into timeslots and figure out which is going to require me the most amount of time to be able to develop. What I will need to research for them to be completed. Identify if they have a relationship with the database or another feature so that I can know if they require additional steps to protect them, and then be able to tick off the completion of said requirements in the sprint and overall evaluation of the project as a whole.

By charting out a product backlog with due dates on everything and organisation on them. This will allow me to objectively look at the project always and keep me on track with all the activities in the project as well as allow me to be able to research project features and define them more during each step of the project so that I am happy with what I am doing with the identified requirements of the project.

By meeting the end of the product sprint, I will be able to look at the project and what activities have been completed and then adapt the project for the next cycle. The sprints allow me to look at each milestone of the project and be satisfied with them. It will allow me to be able to discuss with another person where I might be lacking in the project or where I am wasting time, or plan for further stuff in the future of its development.

This will hopefully always give me a good overall approach to the project and know exactly where I am at all times because I feel like this is one of the key necessities to the project.

6.1.5 Technical Details

I will be implementing the open source-frame Electron.JS which is developed and maintained by GitHub, it uses a Node.js runtime environment, to allow for the creation of a desktop application.

This allows for HTML, CSS, and JavaScript to be used in conjugation with each other for the creation of the application allowing me not to have to use Java, or C# to develop the project completely.

The JavaScript will give me access to its library and available resources. I will be able to use APIs in the project that should be able to help me pull words into the system to create a functional passwords system, based on phases – “TheElephantInTheRoom”

PASSWORD API, which can compare passwords with a known list of breached passwords, will help with one of the features of informing the user if a password they have entered into the storage part of the database has been breached and recommending that they change it.

Hashing API will allow for the passwords to be hashed in the database and protected further, it is not recommended that you create your hashing but looking at the options available for the project will be good. This is an important approach to the protection of the data within the project. It is also an algorithm at the same time.

It also gives me access to python scripts, which offers me excellent access to creating protection for my application, as well as access to the database features in other

implementations that may come in handy for the project. It has many advantages and benefits to the project.

SQL will be used to create the local database and be able to access data within the application itself.

Special Resources Required:

6.1.6 Special Resources Required

1. USB: As a feature of the security key will be a USB stick, it is a special resource that is required to be able to use the application itself, that will create the encrypted file which is decrypted by the application to gain access to that account.

2. NFC Scanner: This would allow for a key card to act as a password as well for the user allowing them to tap the card and log in with a bio-metric password, for them.

6.1.7 Project Plan

For my First Step, I will be implementing create a GUI diagram, to be implemented into the project start away. This will take me less than two days as I want to be happy with the GUI before proceeding with anything, as I don't want to throw stuff together and hope it works; I need to carefully do everything with a clear plan in mind.

This will allow me to research and plan out the other features in a more detailed manner and figure out what is connected to what; and how everything will work together with each other. Continuous research, in particular for bio-metric implementations into the application, and how I will store that data in the database.

The next step while researching will be to create and test out a basic database and make sure I edit information from the application itself. Following that, I will then be able to make the first stage of the application, the login system, with a password and pin.

This will allow me to test out and figure out the security features that I will need to add and how I will add them.

From here I can begin to test out the facial recognition as I have webcams on hand to test them out while I acquire additional parts for the fingerprint scanner and bio-metric forms of logging into the system.

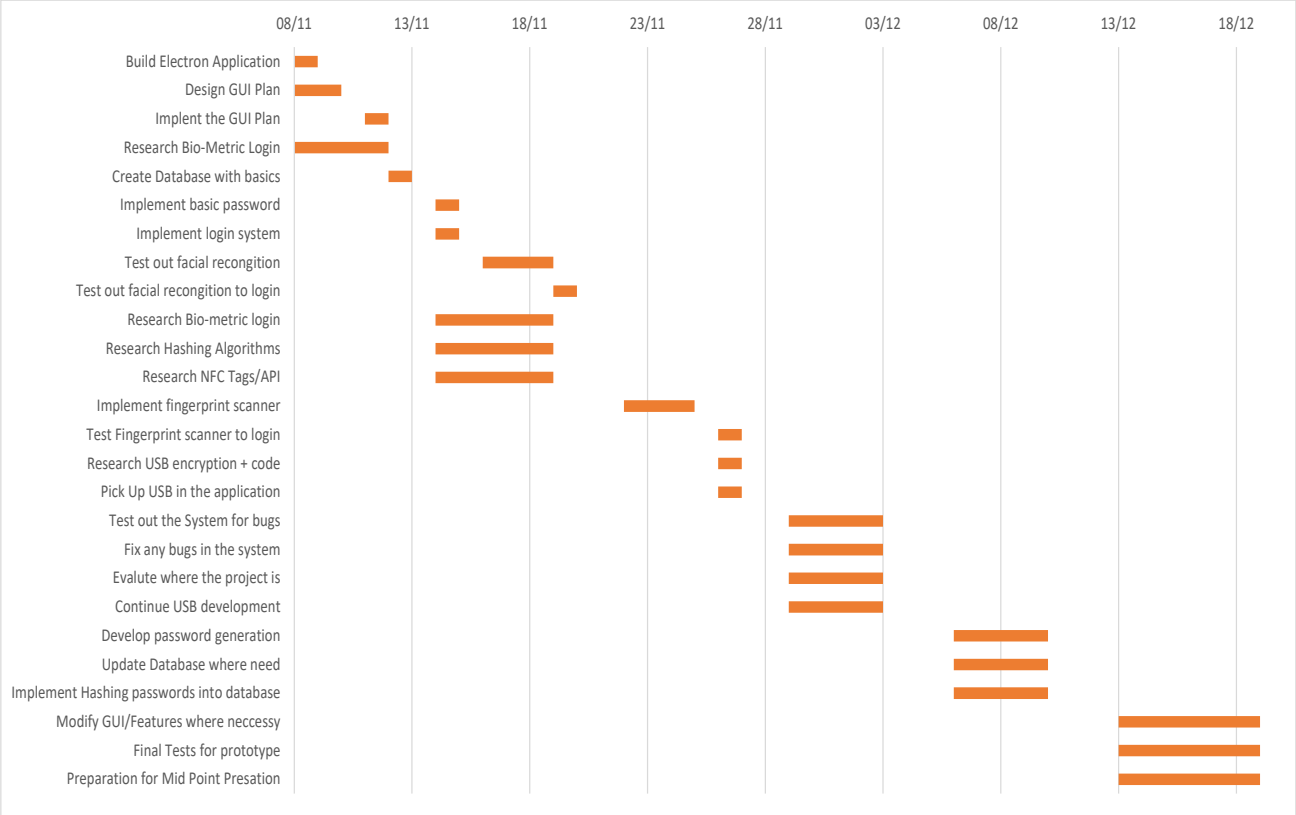
Once I get that running, I research those features again with my new knowledge and

test using the facial recognition with the application itself. From here, I'll have a good working prototype and have met one of the requirements I have set out. I will still be far from done.

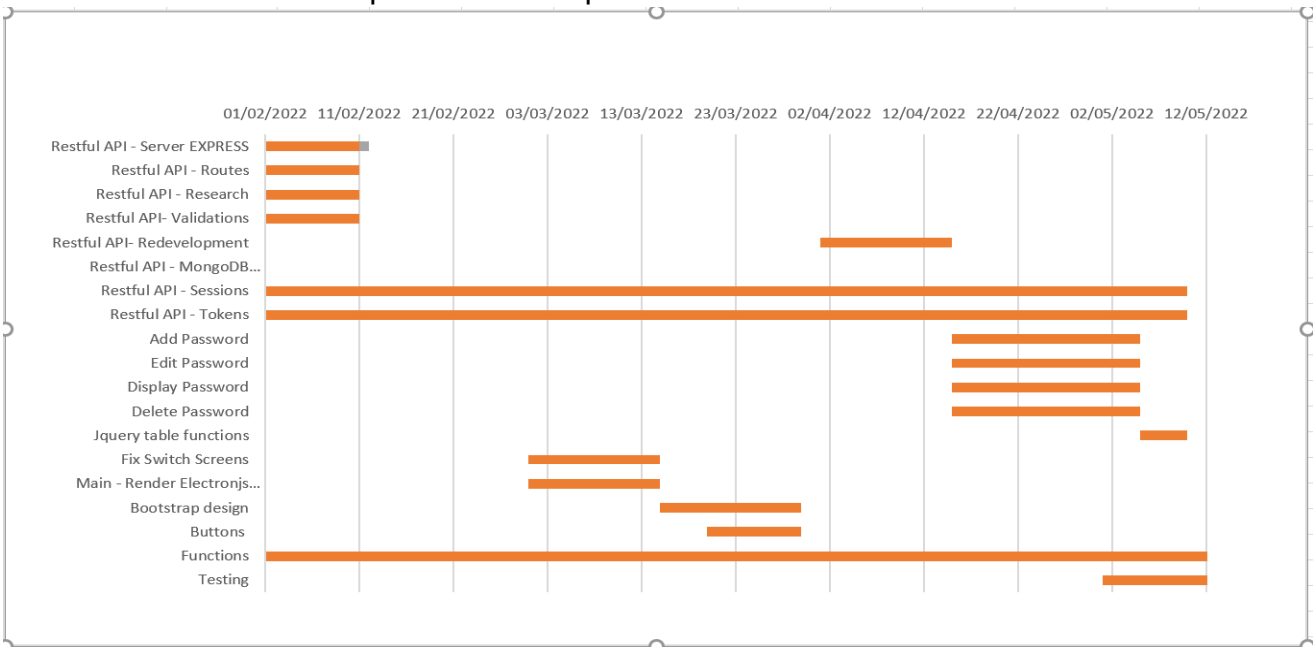
I'll be able to have a clearer retrospect of the project and refine anything if I need to.

Below I have included a Gantt Chart with how I think the project will currently progress up until the mid-point presentation.

Task	Start Date	End Date	Duration
Build Electron Application	08/11/2021	09/11/2021	1
Design GUI Plan	08/11/2021	10/11/2021	2
Implent the GUI Plan	11/11/2021	12/11/2021	1
Research Bio-Metric Login	08/11/2021	12/11/2021	4
Create Database with basics	12/11/2021	13/11/2021	1
Implement basic password	14/11/2021	15/11/2021	1
Implement login system	14/11/2021	15/11/2021	1
Test out facial recongition	16/11/2021	19/11/2021	3
Test out facial recongition to login	19/11/2021	20/11/2021	1
Research Bio-metric login	14/11/2021	19/11/2021	5
Research Hashing Algorithms	14/11/2021	19/11/2021	5
Research NFC Tags/API	14/11/2021	19/11/2021	5
Implement fingerprint scanner	22/11/2021	25/11/2021	3
Test Fingerprint scanner to login	26/11/2021	27/11/2021	1
Research USB encryption + code	26/11/2021	27/11/2021	1
Pick Up USB in the application	26/11/2021	27/11/2021	1
Test out the System for bugs	29/11/2021	03/12/2021	4
Fix any bugs in the system	29/11/2021	03/12/2021	4
Evalute where the project is	29/11/2021	03/12/2021	4
Continue USB development	29/11/2021	03/12/2021	4
Develop password generation	06/12/2021	10/12/2021	4
Update Database where need	06/12/2021	10/12/2021	4
Implement Hashing passwords into databa	06/12/2021	10/12/2021	4
Modify GUI/Features where neccessy	13/12/2021	19/12/2021	6
Final Tests for prototype	13/12/2021	19/12/2021	6
Preparation for Mid Point Presation	13/12/2021	19/12/2021	6
			0
Restful API - Server EXPRESS	01/02/2022	11/02/2022	10
Restful API - Routes	01/02/2022	11/02/2022	10
Restful API - Research	01/02/2022	11/02/2022	10
Restful API- Validations	01/02/2022	11/02/2022	10
Restful API- Redevelopment	01/04/2022	15/04/2022	14
Restful API - MongoDB Connection	01/02/2022	01/02/2022	0
Restful API - Sessions	01/02/2022	10/05/2022	98
Restful API - Tokens	01/02/2022	10/05/2022	98
Add Password	15/04/2022	05/05/2022	20
Edit Password	15/04/2022	05/05/2022	20
Display Password	15/04/2022	05/05/2022	20
Delete Password	15/04/2022	05/05/2022	20
Jquery table functions	05/05/2022	10/05/2022	5
Fix Switch Screens	01/03/2022	15/03/2022	14
Main - Render Electronjs responses	01/03/2022	15/03/2022	14
Bootstrap design	15/03/2022	30/03/2022	15
Buttons	20/03/2022	30/03/2022	10
Functions	01/02/2022	15/05/2022	103
Testing	01/05/2022	15/05/2022	14



Gantt Chart of timetable up until the final presentation:



6.1.8 Testing

One of the main methods I will be using for evaluating my system will be manually testing all the inputs. I will need to first make sure everything is working as I intend them to, for example. That when I click on adding the password to the system, it is all not letting me access information I shouldn't be able to.

Another way to do, this will be able to use Unit Testing on the application as I progress on the project to make sure everything is working as indeed. Each step of the project will also require me to test it out for bugs, which I can make notes of and fix as I progress through. This can be achieved by being able to use Selenium with it as it works with JavaScript. It supports web-based automation in the project.

Testing the database to make sure it follows proper procedures such as ACID and validates how data is accessed at all stages of the project.

Then one of the final tests I will need to perform will be acting in two roles, one as a user creating and making an account for the first time, and then acting like a hacker trying to get into the system.

5.2. Ethics Approval Application (only if required)

5.3. Reflective Journals

Supervision & Reflection Template

Student Name	William Redmond Lawlor
Student Number	18106170
Course	Computing: Cyber Security

Month: October

What? Reflect on what has happened in your project this month? For October I set about finishing up the project idea. I did this by evaluating what ideas I already had, and what changes I would possibly need to make for them. My main idea was to create a password manager, I adapted this idea as much as possible to meet the requirements for the software project. I set out to make the necessary changes such as additional security features, and how they would be implemented. How I would generate a password for the user; How much time I would need to implant each feature and what should I get done and over with first to increase productivity on the project as a whole. As well as what technologies I would be using.
So What? Consider what that meant for your project progress. What were your successes? What challenges remain? I feel this meant that I went from having an idea, to a solid plan for my software project overall. It also meant that with the new ideas and new angles. I could look at the project more objectively to progress on it. The challenge however that remained was how I was going to develop the application at hand and how I was going to do it, this was met with looking at new technologies and programming languages to look at. I was able to choose which technologies I wanted to use, and from the additional security features, how I could more structurally implement the ideas in betters forms/ways. Such is one of the key features of finalising the language/technologies I would be using: Electron has been a good option. This allows me to use HTML, CSS, JavaScript, Python, APIs, and several other languages in conjunction with each other to achieve my project goal. Multi-Factor Authentication was added to the idea of the project, this creates a big challenge, as the user of the application should be able to choose which authentication factors, they want to use to log into the application itself. This helped flesh out the project progress for this month.
Now What? What can you do to address outstanding challenges? To address the current challenges I have started by learning how to set up an electron environment, not related to my project and begin programming different things into it to become more familiar with it. I have also looked into how to integrate multi-factor authentication into the project as well. Another is creating an ideal timeline of what will be needed to be done each month, so I have set up a rough estimation for myself at the moment until I can grasp more on electron and what other details may be needed to be added to the project on time.
Student Signature | William Redmond Lawlor

Supervision & Reflection Template

Student Name	William Redmond Lawlor
Student Number	18106170
Course	Computing – Cyber Security

Month: November

What?
Reflect on what has happened in your project this month?

- Created the graphical user interface for the project
- Created the basis for the project and organised the file structure; understanding how to access the files through electron.js
- Read and learn about python coding to be able to create a password generator and implement it into the project.
- Implemented an SQLite3 database into the application
- Got up to speed on calling and installing applications required with node.js.

So What?
Consider what that meant for your project progress. What were your successes? What challenges remain?
I am now hopefully past most of the major blocks for the project and now that I have more time to organise my thoughts on the project. I will be able to progress much more quickly on the project.
I have had to realise what ideas may or may not work on the application itself. So, I have had to evaluate much more how I am going to introduce biometric data into the application
Compare SQLite against MongoDB and other databases to be able to consider what to use for the local database or if I should just create a single file for the system altogether.

Now What?
What can you do to address outstanding challenges?
To address the outstanding challenges in the project, I need to re-organise what I have left to do and evaluate what is going on. See what can and cannot be done, such as for the biometric data. See where I am not happy with the target and understand how long it is taking me to be able to implement each stage of the project into the system itself; I feel like some parts I may have completely underestimated and need more time to see how they function. This will come with time, but hopefully, after this month I have a stronger understanding.
Electron.js is an interesting open-source project to be able to use for the project given that it is using Node.js with it to be able to function.
That while using python with the project is hard but not impossible and take it one step at a time when trying to use the project. I believe I have been able to clearly define which parts are not going to be joined together and which parts are not going to be together.

Student Signature	William Redmond Lawlor

Supervision & Reflection Template

Student Name	William Redmond Lawlor
Student Number	18106170
Course	Computing – Cyber Security

Month: December

What?
Reflect on what has happened in your project this month?
Preparing a prototype for the mid-point presentation meant reading through the documentation for myself and figuring out which functional requirements I wanted to be able to show off for the presentation. This meant for I needed to evaluate what was completable for this stage of the project, I believe that having the ability to switch between the main three pages and how they would look for the user in theory would be the best thing to do.
The write up for the project also took a lot longer than expected so, in the end, I ended up putting more of an effect into the write up to be able to complete it first. Then try to add as much to the project first as I possibly could.
Finished creating a frameless window for the main design of the application, as an electron feature you can make your title bar using HTML, CSS and JavaScript to create functionality for the buttons, and give access to this function on all pages that require it.

Created a Parent and Child Windows for the application to be able to switch between the pages seamlessly especially for the login function of the application, separating it from the main part of the application.
The applications folder organisation: making it easier to view files for me and link to the files directly. It also allows for an easier flow between the pages and accessing the JavaScript files to provide the coded need for most functionalities.

So What?
Consider what that meant for your project progress. What were your successes? What challenges remain?

That the functional and non-functional requirements could nearly be fully set in stone from the document requirements and what I was now seeing was more possible for me to be able to do this month without further research and what would be required moving forward for myself.
This included that was the python script going to be completely necessary for the project, as, I could write the same script into the JavaScript section of the project and still get the same results.
Would dragging in an API to check the passwords also pose a risk to the project that I would have to deal with for the moment, I think it would be best to exclude it from the project and instead implement a list of the common passwords over ten years and disable from being entered into the database to encourage the user not to be able to use them within the database.
The MongoDB database connection, should it be used using the node.js functionality or just purely be used using the python connection method, for the moment, I believe using the node.js is the best way to interact with the project database for the moment seeing as node.js is built for that type of work.
This progress has allowed me to move forward fast now on the project seeing as I am coming to a further understanding of what is needed to be able to perform the functions of the project.

Now What?
What can you do to address outstanding challenges?

One problem I can address is the write up by trying to do it alongside writing up the features of the project and hopefully it will allow me to be able to improve the documentation overall as well as how I approach the use case diagrams etc, I feel that I could be clearer with them. With a list of functional and non-functional requirements, I now have a clearer vision of the inner workings of the project for myself rather than mental plans, allowing me to take further steps ahead each time I add a feature to the project. The challenge now is finishing off the project and making a clean cut application that I like and being able to test thoroughly for vulnerabilities.

Student Signature	William Redmond Lawlor

Supervision & Reflection Template

Student Name	William Redmond Lawlor
Student Number	18106170
Course	Computing – Cyber Security

Month: January

What? Reflect on what has happened in your project this month? This month I worked on learning about adding different authentication features such as token-based and certification based to the project with the focus being on creating an express server in the local environment of the Electron.js system upon start-up of the application and then incorporating this into the login system, from here, it also means I will be possibly able to run the pythons scripts much easier through that local server to be able to access them using an available child-spawn process. Then can chain them together to hash/encrypt them on the database through the various python and JavaScript. With token-based authentication, I can leave it being checked in the background for five minutes before it expires, and a new one is needed to be requested. This can be done through JavaScript. This is the same with certification, but more research is still required before adding it to the project. Limit testing the server as much as possible, learning how it interacts with the server at the time, when within the framework vs, when I am testing it outside the framework by itself so that I know I have it working. Working on the connection between the database and the server with the mongo DB atlas database, as some of the uploads were not working out correctly, as well as getting dummy data down onto the testing environment so that I know I am actually connected to the server and working with me properly. So that I can develop the crud functionality. Finishing up the information on the password creation HTML, CSS, and JavaScript pages/files. Getting the problem that was occurring such as with the express serving shutting down between parent and window switches. Looking into what was needed to be able to understand exactly what was happening to the express as there were problems occurring
So What? Consider what that meant for your project progress. What were your successes? What challenges remain? Set up a separate environment for the express server to work outside of the electron.js system, so that I could make sure it was working first possible problems that could occur were first happening only with the express server itself and then moving it onto the actual application with the electron

framework so that I could then tackle those problems from then on. One for testing it out and another for being there.
I was then able to implement tokens into the express server using JWT, a service on the node.js to be able to use it. Now it meant getting the application up and running with electron.js, so this meant checking out what the best approaches were, which was creating a rendering file at the start of the application and then using postman to check if it was working.
There was a lot to learn about the express and how it would function within the application as well, so learning how to swift through the problems and find the answers to what was going on was important and helpful in gaining a much further grasp of what was going to be needed to be done.

Now What?
What can you do to address outstanding challenges?
Will the express server always be able to work with the project? Or will it cause further problems down the line? Routing functions of the express server also allow for testing in postman, do I need to separate the implementations for the project to secure it.
So, I have to move more carefully than before and get on track with the other login methods so that they can be done all at once.

Student Signature	William Redmond Lawlor

Supervision & Reflection Template

Student Name	William Redmond Lawlor
Student Number	18106170
Course	Computing – Cyber Security

Month: March

What?
Reflect on what has happened in your project this month?
This month, I created the session for the server and application so now that when the user logs on to the database, we can see that in the database and compare them to each other to act like a real person is there.

Made a lot of changes to API and how it works and put in the preparation work for how the application was going to run with the system in place.

Found out exactly what I was going to need to do to be able to test the system and make preparation for it.

Made changes to how the technical report by reading it and working out what needed to be done with it.

Make a working login system.

So What?
Consider what that meant for your project progress. What were your successes? What challenges remain?

Getting the tokens working on the system at the moment as they are required to be placed within the header, might take several attempts to do. Integrate the login system into the application, so that it is easier to be able to use it fully tested outside the electron application. Re-reading the technical report more so that it is clear in my head Make sure to identify all the technologies with the application for the report and make sure I will be to discuss them. Create a vault for the application so that passwords are hashed and secure within the database	
Now What? What can you do to address outstanding challenges? Re-writing the technical report so that it is easier to read, and better overall as some points were not communicated well within it during the midpoint submission. Still working on what is going on within the electron project. Moving important file information onto the .env fly to act as a point of protection for the application for URLs etc. Preparing for the testing cycle.	
Student Signature	William Redmond Lawlor

5.4. Other materials used

Any other reference material used in the project for example evaluation surveys etc.