**UP**

By

Nkosinathi Madalane (s214212483)

Submitted in partial fulfilment of the requirements for the degree

BACCALAUREUS TECHNOLOGIAE

In

INFORMATION TECHNOLOGY

In the

FACULTY OF ENGINEERING, THE BUILT ENVIRONMENT AND  
INFORMATION TECHNOLOGY

Of the

NELSON MANDELA METROPOLITAN UNIVERSITY

Supervisor: Mr RG Leppan

November 2018

Table of Contents Page

[1. Chapter 1: Business Case 3](#_Toc529173506)

[1.1 Introduction 3](#_Toc529173507)

[1.2 Literature review 3](#_Toc529173508)

[1.3 Related system analysis 4](#_Toc529173509)

[1.4 Problem Statement / Opportunity / Directive 6](#_Toc529173510)

[1.5 Objectives / Proposed solution 8](#_Toc529173511)

[2 Chapter 2: Requirement Specification 9](#_Toc529173512)

[2.1 Project Scope 9](#_Toc529173513)

[2.1.1 Information Scope 9](#_Toc529173514)

[2.1.2 Functional Scope 9](#_Toc529173515)

[2.1.3 Communication Scope 9](#_Toc529173516)

[2.2 Project Specification (Requirements) 9](#_Toc529173517)

[2.2.1 Functional Requirements 9](#_Toc529173518)

[2.2.2 Non-functional requirements 11](#_Toc529173519)

[2.2.3 User Interface Requirements 11](#_Toc529173520)

[2.2.4 Database Requirements 11](#_Toc529173521)

[2.3 Design constraints (System Risks & Control Requirements) 14](#_Toc529173522)

[2.4 High level Use Case diagram 14](#_Toc529173523)

[3 Chapter 3: Detailed Technical Design 15](#_Toc529173524)

[3.1 Task Analysis 15](#_Toc529173525)

[3.2 User Interface Design 17](#_Toc529173526)

[3.3 DB Design 18](#_Toc529173527)

[3.4 Conclusion 20](#_Toc529173528)

[4 Chapter 4: Implementation 20](#_Toc529173529)

[4.1 Introduction 20](#_Toc529173530)

[4.2 Implementation Tools 20](#_Toc529173531)

[4.3 Implementation Techniques 21](#_Toc529173532)

[4.3.1 Database Implementation 21](#_Toc529173533)

[4.3.2 User Interface implementation 21](#_Toc529173534)

[4.3.3 Functional Specification 22](#_Toc529173535)

[4.3.4 Extracts of complex code 22](#_Toc529173536)

[4.3.5 Reference source of code 24](#_Toc529173537)

[4.4 Conclusion 25](#_Toc529173538)

[5 Chapter 5: Test Plan and Test Results 25](#_Toc529173539)

[5.1 Introduction 25](#_Toc529173540)

[5.2 Testing 26](#_Toc529173541)

[5.2.1 Test Cases 26](#_Toc529173542)

[5.2.2 Code Testing 31](#_Toc529173543)

[5.3 User Interface Evaluation 32](#_Toc529173544)

[5.3.1 Informal User Testing 32](#_Toc529173545)

[5.3.2 User Satisfaction 32](#_Toc529173546)

[5.4 Changes Implemented 33](#_Toc529173547)

[5.4.1 Implementation 33](#_Toc529173548)

[5.4.2 Design 33](#_Toc529173549)

[5.5 Conclusion 33](#_Toc529173550)

[6 Chapter 6: Deployment Documentation/Installation Guide 34](#_Toc529173551)

[6.1 Introduction 34](#_Toc529173552)

[6.2 Operating System used 34](#_Toc529173553)

[6.3 Development Environment 34](#_Toc529173554)

[6.4 Components Needed 34](#_Toc529173555)

[6.5 User Guide 34](#_Toc529173556)

[7 Chapter 7: Conclusion 46](#_Toc529173557)

[7.1 Introduction 46](#_Toc529173558)

[7.2 Project Goals 46](#_Toc529173559)

[7.3 Achievements 46](#_Toc529173560)

[7.4 Problems Encountered 46](#_Toc529173561)

[7.5 Future Research 47](#_Toc529173562)

[7.6 Conclusions 47](#_Toc529173563)

[8 Progress Report 48](#_Toc529173564)

[9 References 48](#_Toc529173565)

## Chapter 1: Business Case

## Introduction

This document is written to make people aware of the issue that is faced by students, but even beyond focusing on varsity students alone, this issue is faced by every person who owns a USB flash drive and stores valuable data on it. (Jeong et al., 2007) says “The various problems related security have been occurred as it is used widely. Nevertheless most USB flash drives do not include the security mechanism, an attacker can easily acquire the private information in the USB flash drive”.

On this document I will cover the issue that is faced by students (in particular NMU students) of losing their USB flash drives. I will provide my insight on USB flash drives, what is the purpose it, risks and solutions. After reading this document one should understand my goal which is: more than anything but to protect the information contained on a USB flash drive device.

## Literature review

USB flash drives are used by many people on daily basis. USB Flash Drives (UFD) or “USB sticks” are becoming popular means of storing and transporting data and applications. UFD sales worldwide reached $2 billion US Dollars in 2005, according to Gartner research (Al-zarouni, 2006). This popularity of this device makes it vulnerable to information security. As (Jeong et al., 2007) stated on his article, Nevertheless most USB flash drives do not include the security mechanism, an attacker can easily acquire the private information in the USB flash drive.

There are available drive encryption software we can find online. BitLocker is a computer hard drive encryption and security program released by Microsoft Corporation as a native application in its Windows 7 Enterprise and Ultimate editions, Windows Vista Enterprise and Ultimate, and Windows Server 2008, R2 and 2012 operating system versions. It is a drive security and encryption program that protects drive content and data from any offline attack (Techopedia, 2018). The aim of BitLocker is to provide for greater data security by encrypting the entirety of a hard drive, as opposed to the current systems available which only provide limited encryption (Woodward, 2006). We can use software encryption such as BitLocker to protect information in our USB flash drive too by password/pin protecting it. This will protect information in your drive from being easily accessed by intruders. Of cause this technology is very useful and I have nothing against it. Infect I believe USB flash drive uses should be aware of such benefit and use it to protect their information. But I want more than that alone, after a student loses his/her USB Flash Drive I want them to be able to retrieve their information as well, Infect I want them to be able to get their USB Flash Drive device back.

This project is focusing on allocating a unique Identifier (in a string format) as a USB ID on each USB of each student and retrieve the login details of that student and store this information on a database (provided that a student has agreed to a screen prompt that he/she is the original owner of the USB), then once the USB of that student gets lost and student reports it as missing, the program will flag the missing USB as “missing”, using its USB ID the program will compare every USB that has been inserted on NMU computer to see if the ID matches the ID of the flagged USB as missing, if these IDs match we can confirm the USB is found and also check which student is using the USB that has been found by retrieving his/her logged in details from the computer he/she is using. I also want to password protect the USB by encrypting them to protect the information contained in it.

## Related system analysis

Google find my phone is a system related to this system. It uses your phone Geographical Positioning System (GPS) to locate your lost phone.

Most Android phones now come with Find My Device (formerly Android Device Manager) built in. This service will automatically track your phone's location, so if it ever goes missing you can hop on your laptop or a friend's phone and find it is last known location, ring your phone if it's nearby and you need a hint, or lock and/or erase your phone if you fear it has been stolen (Lagace, 2018). Figure 1 show a picture of find my phone interface.

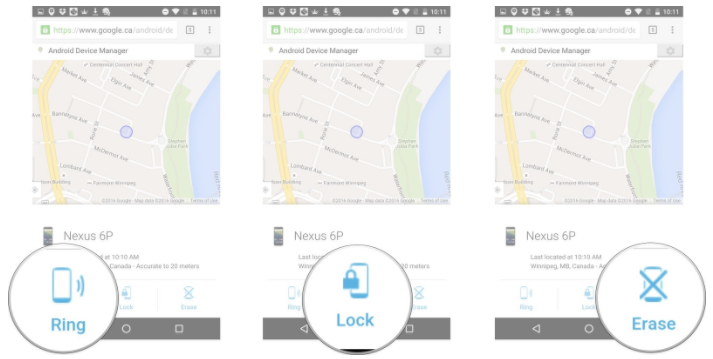


Figure 1

One of the related systems to my system is BitLocker.

Figure 2 shows how you can use BitLocker to unlock your drive, from (How to Securely Encrypt a USB Flash Drive, 2018)

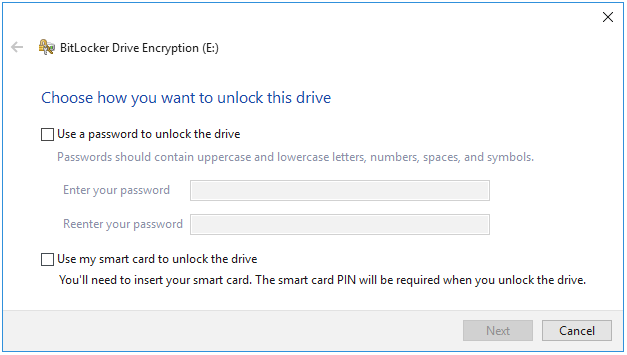


Figure 2

## Problem Statement / Opportunity / Directive

The idea of **up** originated from a frustration of losing a USB. I have lost many of my USBs over the previous years, other students lost even much more USBs then I did. Originally, the idea was to create an application that finds a USB when it is lost.

For this section I have decided not to use a Problem Statement Matrix Table, since I already have described the problems using it on my Business Case document. But this time around I want to have more words and figures that describe exactly what the problems are.

(Von Solms, n.d.) says “Business information and the systems processing such information has become a key asset in most organizations today”. But is information an asset only to an organization?

Can information be an important asset to students too?

Figure 3 is a picture I have taken at missionvale campus on 07 May 2018, it is a paper that was posted on the walls around the campus.

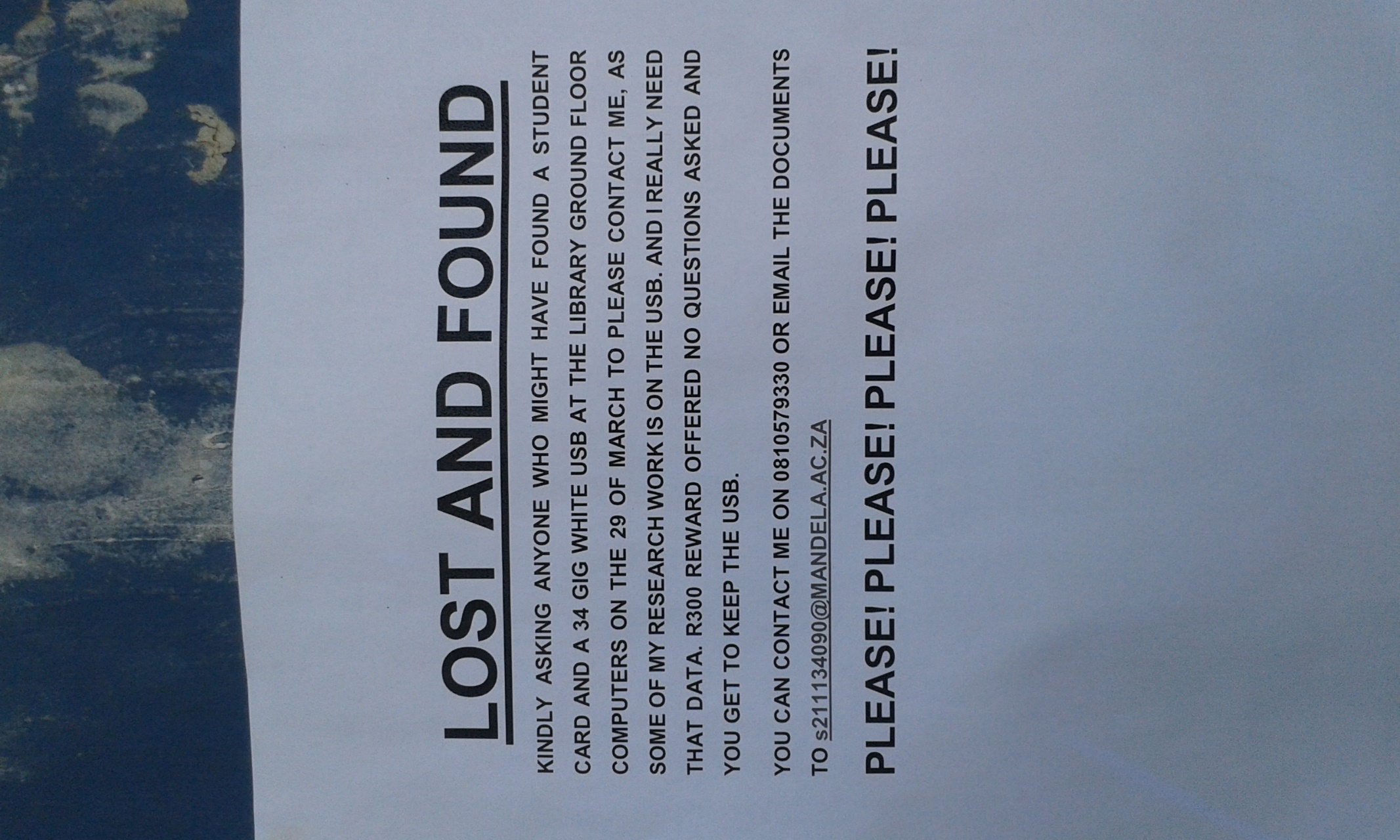


Figure 3

Figure 3 shows how important information can be to a student. Some students use a USB as their main resource of storing information (mostly school work). If they loss that information it may cost them on their academics. So, my point is information is important to an organization for their processes, but we should also look at how information is important to students or any other user as an individual.

Since USBs are small, they are easy to loss. At NMU North campus on the school of ICT, on the Lost Items records for 2017, USBs are the most reported items on the list. Infect they make it number one in each year. Figure 4 illustrates the reported lost items on the School of ICT at North campus on the year of 2017.

Figure 4

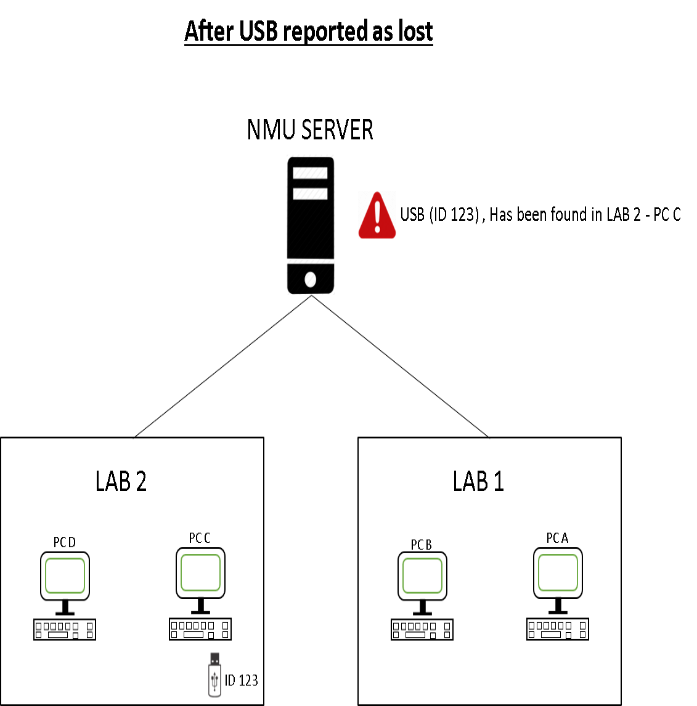
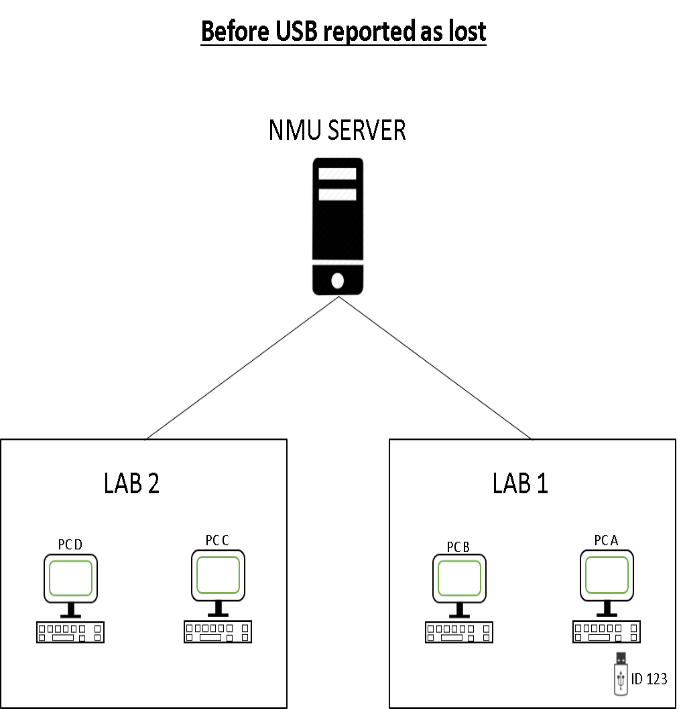
There is currently no system in place similar to “find my phone” (To locate your lost smartphone) that locates lost USBs. If there is a such a system in any place, we as students are not aware of such.

Which brings me to my other concern. **up** is also willing to make students aware of information protection. Some students are not aware that there are software applications that encrypt USBs to protect your information. Similar to Microsoft’s BitLocker (BitLocker is a full disk encryption solution, says (Jackson, 2015).) that has an option to encrypt your USB that comes for free on windows 10. But some students are not aware of this feature and are not using this benefit.

## Objectives / Proposed solution

My idea is to create a software system that locates a student USB after it was reported as a lost item within the campus. This would be achieved by using a unique string as an ID to identify a USB, link that USB to its rightful own on the database and if the USB has been reported as lost **up** would pick it up within the network and show details of where it is found and on who is using it. Figure 5 illustrates a structure of how **up** should work in the campus.

Figure



PC D

USB (ID 123), Found in LAB 2 – PC C

PC A

PC B

PC C

PC A

PC B

PC D

PC C

Figure 5

USB (ID 123)

USB (ID 123)

On Figure 5 (Before USB reported as lost): USB (ID 123) had been inserted by the rightful owner in LAB 1 on PC A. in this instance **up** will prompt the user if he/she is the owner of the USB, if that student says yes to the prompt, then **up** will retrieve to student number of the student that is logged in and store a unique ID that is used to identify a USB to the records of the logged in student. Figure 5 (After USB reported as lost) illustrates: if the USB (ID 123) has been reported as lost. If any student picked up the USB and insert it on any lab (in my example LAB 2 on PC C) **up** will detect using the USB unique ID; USB (ID 123) that was reported as lost has been found in LAB 2 – PC C. And send the student that is current using the lost USB an email like “Please return the USB to the security’s office, it does not belong to you.”.

**up** will also provide an option for students to password protect their USB so that anyone who picks it up cannot access the information on it unauthorized. And also encrypt the data so that it may not be readable to the intruder.

# Chapter 2: Requirement Specification

## Project Scope

### Information Scope

**up** will need to make use of a unique USB ID. This ID may be a concatenation of a USB vendor and product ID or a hidden txt file in the USB contain a generated unique string. **up** will protect a student’s information contained in the USB by password protecting the USB and encrypting it.

### Functional Scope

**up** will provide a client and an admin system. The client system will be working on the background to retrieve USB unique ID, student’s login details and store these details on the database. **up** will use these details to identify lost USBs within the campus network. **up** will also password protect students USBs and may also provide an option to encrypt data in the USB. On the admin side: the admin will use the system to flag the lost USBs after they have been reported as lost, as well as receive the notifications of found USBs, which labs the are located in, time they were located and which student used the USB that was reported.

### Communication Scope

An email will be sent to a student that the located USB that was lost is found in. A simple windows prompt will appear on the screen to communicate with the clients(students).

## Project Specification (Requirements)

### Functional Requirements

The tables below illustrate the functional requirements for **up.** They each focus on the purpose, input, the process and the output that the system will deal with, for each function.

Table 1 is a functional requirement for the ‘Locate a lost USB’ function.

Table

|  |  |
| --- | --- |
| Locate a lost USB | |
| Purpose | To locate a USB after it was reported as lost and bring it back to its rightful user |
| Input | When the user inserts a USB in a computer. **up** will automatically retrieve the USB unique ID, student login details and computer details such as (PC number, lab number and time). |
| Process | **up** first checks if there is any lost USB, if there is a lost USB, the system will compare the inserted USB unique ID to the lost USB unique ID. |
| Output | If the inserted USB ID matches the lost USB ID. A notification on the admin system will appear to show the lost USB has been located. And a display of a USB ID, USB friendly name, student number, PC number, lab number and time of log in. and an email sent to the student the USB is found on. |

Table 2 is a functional requirement for the ‘Password Protect USB’ function.

Table

|  |  |
| --- | --- |
| Password Protect USB | |
| Purpose | To protect USB from unauthorize access to a student’s information. |
| Input | A student confirmation to allow the password protection to be active on the USB. |
| Process | The system will use DiskPart command ‘REMOVE’ to unassign the drive letter to a USB so that it does not show when it is stolen until a correct password has been entered to make it show. |
| Output | While the correct password is not entered, the USB will not be listed as one of the drives. |

Table 3 is a functional requirement for the ‘Encrypt USB files’ function.

Table

|  |  |
| --- | --- |
| Encrypt USB files | |
| Purpose | To prevent unauthorize access to student’s personal/school information. |
| Input | A student confirmation to allow the USB file encryption to be active. |
| Process | The system runs a java CryptoUtils to encrypt the files. |
| Output | The file’s data will not be readable. |

### Non-functional requirements

**up** should not access the information contained on the USB of the student, but only protect it. Any actions of **up** should be confirmed by a student before they execute. Unique USB ID should be unique and not match with any other USB. A student that lost a USB should not be informed by the system who the USB was found on, but only the administrator of the system should know. **up** will be installed on each computer to work on them so **up** should not access any computer resources without authorized access to them. **up** should abide to the policies of an organization that it is implemented on.

### User Interface Requirements

**up** is not much of an application software where it would include a lot of user interaction, but most processes of **up** work on the background without user interaction: So, I will not use most of the Nielsen’s 10 Heuristics for user interface requirements. However, there is an admin system which will interact with a user and there are prompts that the client(student) will interact with, So I might use a few of the 10 Heuristics.

* It is very important that the prompts that the students interact with must give a very clear message that avoids misconceptions so that the student understand what they are agreeing on.
* There user should have a freedom to undo any option they accepted **up** to perform on their USB.
* There should not be any irritating pop-ups or messages to the student or the admin of the system.
* On the admin side the system should have a minimalistic design, it should look professional and not like a system that is interactive and enthusiastic.

### Database Requirements

Figure 6 is a Database structure of **up**. The Computer and the Student details will be retrieved from the NMU Active Directory. The relationship between the Student Table and the USB Table shows that a Student can have none, one or more than one USBs attached to his/her record on the database. The Log Table will retrieve the USB ID and the Student ID from the USB Table (The Student ID is the ID of the student that the USB belongs to). The dateTime attribute is to save the date and time the log is created, which is every time any USB is inserted on any computer and the Status attribute on the Log table is to save a “found” status if the lost USB has been found. But the Status attribute on the USB Table is to save a “lost” status if ever the USB has been reported as lost. The Admin Table is to store the Admin details for login in the system and is connected to the Active Directory to check if the Admin is an authorized stuff member in the institution/NMU. Students will use their Student Number and Student Password to login to the system.



Figure

## Design constraints (System Risks & Control Requirements)

The biggest limitation for **up’s** locating lost USB is: **up** is currently a desktop application, so it does not run on a computer that it is not installed on. Meaning that if a student picks up a lost USB and use it at home on his/her laptop; **up** cannot trace the USB unless the student inserts it on one of the computers that are in the network on campus. This feature might be fixed in future by making some programs run on the USB itself and post the GPS location of the computer the USB is inserted on or some other useful details of a computer that the USB is inserted on, to track who is using the USB.

**up** also needs ‘run as server’ administrative privilege to run the DiskPart commands. This may be fixed by including <requestedExecutionLevel ..> code on the manifest file of the application.

On password protecting and file encrypting the USB may have draw backs: A DVD of a car mp3 player does not provide the functions to unlock the USB, so this prevents the student from accessing their information. This can also be solved by giving an option for a student to encrypt certain files. Like encrypting school work files and leave the music and video files unencrypted.

## High level Use Case diagram

Figure 7 is a High-level Use Case diagram of up. On uc01 the system will have a screen that describes to a student what is up and what are the benefits of using up. This screen will let a student have a choice to let up assign a unique USB ID to his/her USB so that the unique ID can be used to locate a USB when it gets lost. uc09: the system compares the inserted USB ID with the USB ID of any lost USB ID, if the IDs match the Admin will be emailed that the lost USB has been found in a particular lab used by a particular student. The student using that lost will also receive an email warning to return the USB to the Admin’s office.



Figure

# Chapter 3: Detailed Technical Design

## Task Analysis

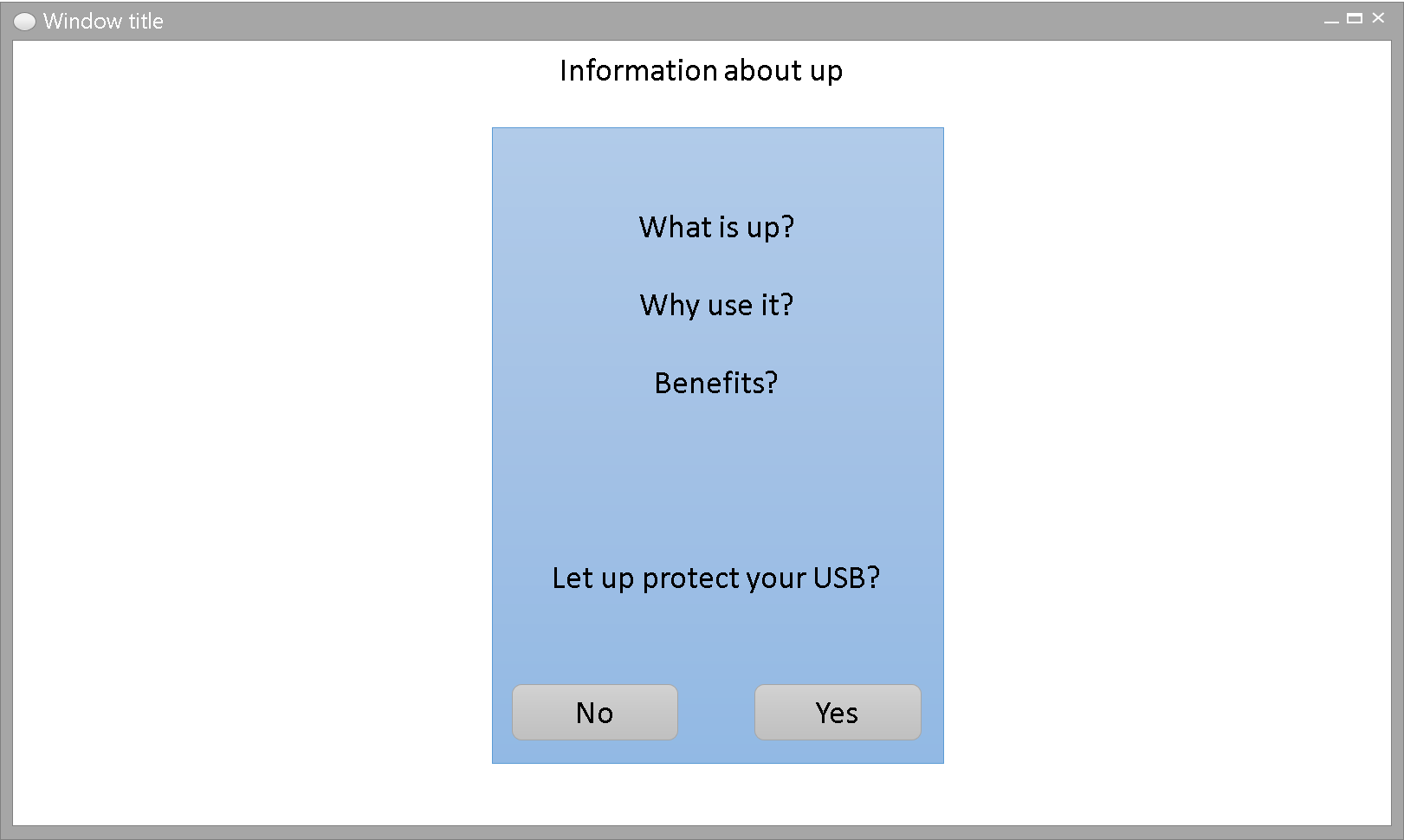
Since most of the system functions do not require a user interaction with an interface, this Task Analysis will show some of the tasks carried by the user, but some are carried by the system itself. The Task Analysis here will take a flow diagram structure. Figure 8 shows the Task Analysis for **up**.



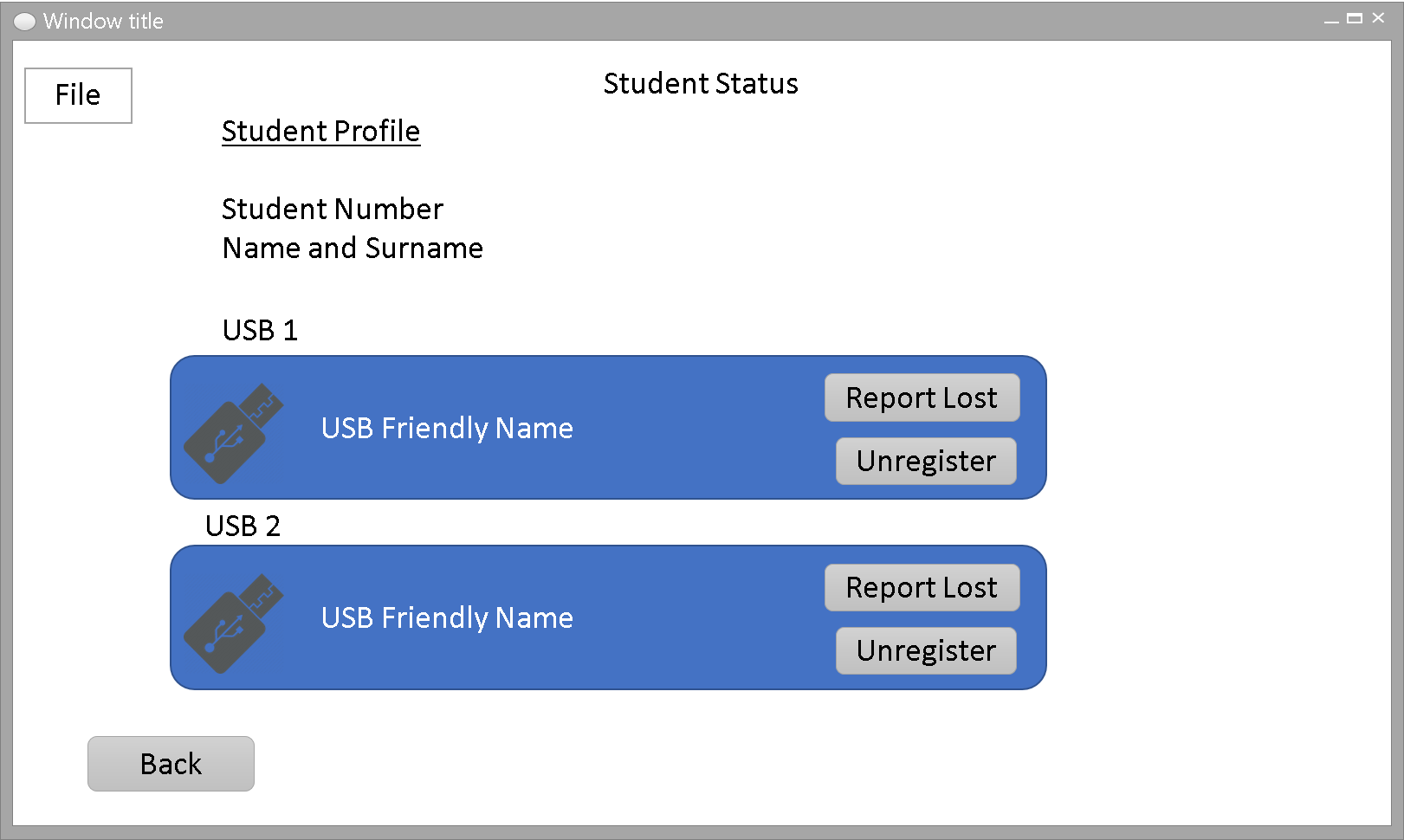
Figure

## User Interface Design

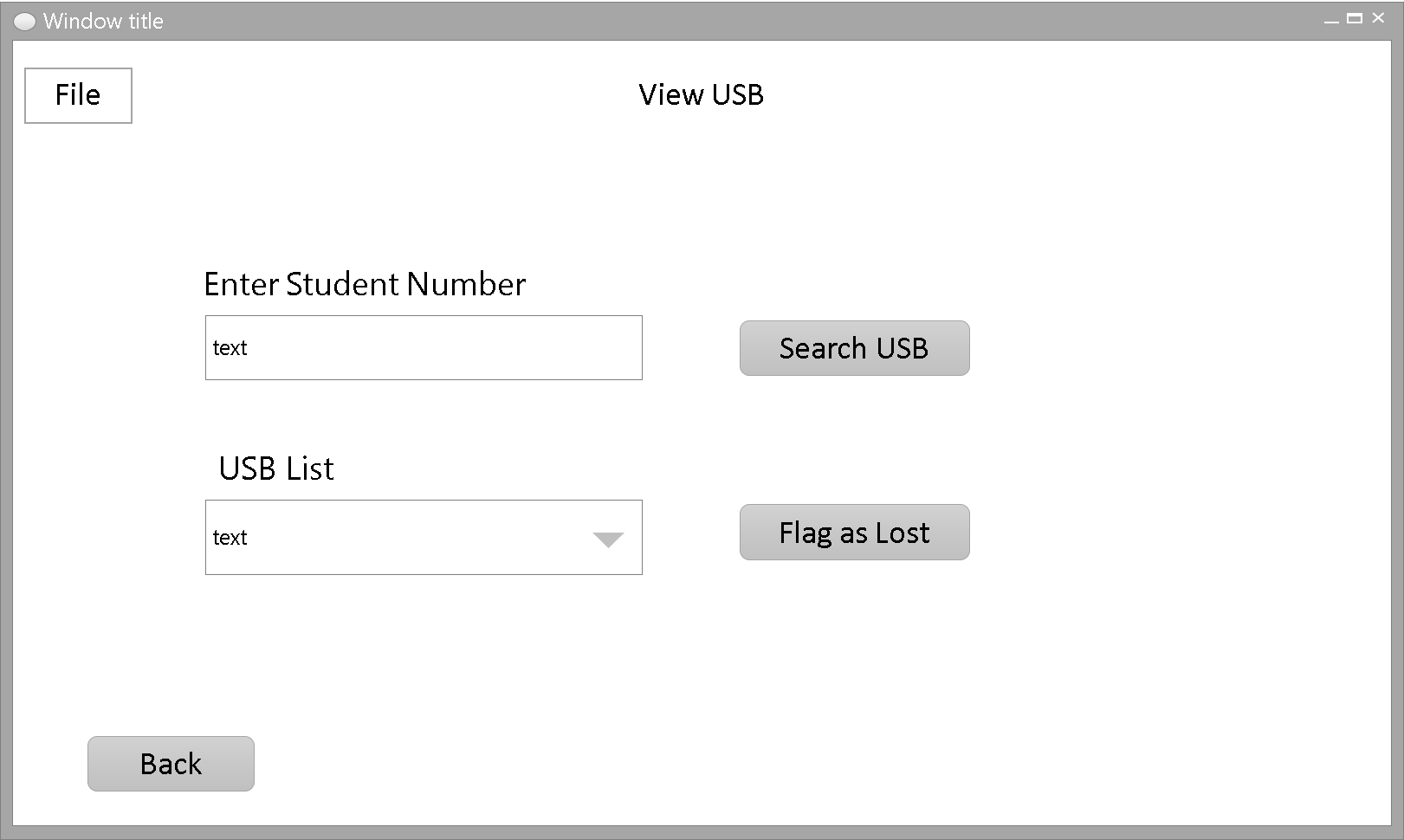
This section provides some user interfaces for **up.** Theinformation about **up** screen, Student Status screen and the Admin View USB screenfrom Figure 9 to Figure 11.



Figure



Figure



Figure

## DB Design

This section shows relationship between entities that take part on **up**. Figure 12 shows the Database Design for **up**.



Figure

## Conclusion

This project is focusing on finding lost items within the campus (in particular lost USBs). This will save many students from trouble of not retrieving their information back when they have lost their USB.

Information Security is a very important field, it focuses on protecting information. Which big systems today consume massive amount of data. Recently facebook has been charged for misusing people’s information. My point is in the IT world data is one of the most important factors to deal with, meaning it needs heavy protection as well. This facebook issue shows that it is not only corporate companies that need to protect their data, but even normal users.

We should ask ourselves, if we have policies to protect business information assets, shouldn’t we have ways to protect individual’s information asset? Why would we not consider protecting student’s information? or is information asset less important to students?

But if this is done in a small scale I believe it can be done in a bigger scale. I see this idea going beyond tracking USB within a Local Area Network but tracking USBs on a Metropolitan Network or even over the internet.

# Chapter 4: Implementation

## Introduction

This document is written to give more detail of the internal behaviour of **up**. Some of the code will be included in this document and the explanation of the algorithms.

This is a technical document and mostly includes technical terms to communicate the choices made to develop **up**. On this document the choice of tools chosen to develop **up** the software application will be listed and an explanation of why the tools were chosen.

## Implementation Tools

**up** is a desktop application made to locate and protect students USB Flash Drives (USB). The main functionality of the application is to locate lost students USBs on campus. But the application also provides USB protection features namely: USB password protection, File Backup and File Encryption.

There might be questions like: why up is a desktop application instead of a web application, since web applications are a popular choice? The choice of **up** being a desktop application instead of being a web application is because **up** needs to retrieve some information on each computer it is running on, this information includes: computer name and ip address, logged in user and access to some computer drives. Of cause a language to achieve these functions is a programming language: like C#, Java or C++ etc. But not JavaScript as a scripting language. A web application would only run one of this programming languages in the web server and not the client computer. The client computer would then run JavaScript on the browser for some logic. So, then a conclusion can be made: that if **up** was a web application the JavaScript code would not be able to retrieve the information needed by the application processes on each client computer, but would only retrieve this information on the web server computer that the programming language is running on. A desktop application is a solution to this problem. **up** would then be installed on each computer it needs to run on and run the programming language of the application on that specific computer to retrieve the information of that computer needed by the application processes and send it to a hosted database.

## Implementation Techniques

### Database Implementation

The **up** application is using MySQL as a database server and Workbench. Workbench provides an interface that allows an easy way to interact with the database server (which is MySQL or MariaDB). MySQL is a better choice to use with JAVA since there are provisions of database connector like JDBC: A high-level, application programmer view of JDBC is an API, called the JDBC API, that provides methods allowing an application to connect, query and manipulate multiple databases (Burton & Marek, 1998). JDBC uses MySQL Connector/J as a driver to communicate with JAVA.

### User Interface implementation

The interface is simple and have a minimalistic approach by only displaying the functional controls of the system. Since **up** is a desktop application it is not JFrames (JAVA Frames) do not use html so they are not CSS friendly in order to style them. So, in order to style JFrames default styling properties were used.

### Functional Specification

The **up** application main function is to provide a unique identity to a USB Flash Drive. According to research currently there is not unique enough why to identify a USB Flash Drive based on the information. There are couple identifies provided in the USB Flash Drive, one example is a device ID (a combination of a vendor ID, product ID and a revision code). Based on the information retrieved from the device descriptor, Windows builds a Device

Instance Identifier (device ID) of the format: USB\VID\_*v(4)*&PID\_*d(4)*&REV\_*r(4)* (Carvey & Altheide, 2018). These identifiers are not unique enough to identify a USB Flash Drive, because a USB that is same product from the same vendor will have the same vendor ID and product ID. Other identifies such as a Container ID and a Class GUID are assigned by an operating system to a USB Flash Drive to identify the USB Flash Drive drivers. However, the Container ID and the Class GUID cannot be relied on to uniquely identify a USB Flash Drive, as the same sequence may be applied to another USB Flash Drive that needs to same drivers. A temporally solution to this problem is to store a hidden text file in a USB Flash Drive and generate a unique key using a JAVA UUID then store the unique UUID on the text file. A UUID is 128 bits long, and can guarantee uniqueness across space and time. UUIDs were originally used in the Apollo Network Computing System and later in the Open Software Foundation’s (OSF) Distributed Computing Environment (DCE), and then in Microsoft Windows platforms (Leach, Mealling, & Salz, 2005).

### Extracts of complex code

On this section few screen shots of some of the complex codes that were used on **up**. This is to demonstrate the achievements that were made and how they were made through the code snippets.

Figure 13 is a code used to identify a newly inserted drive, e.g. when a USB Flash Drive is plugged in the program will recognise its drive and use its letter along the program.

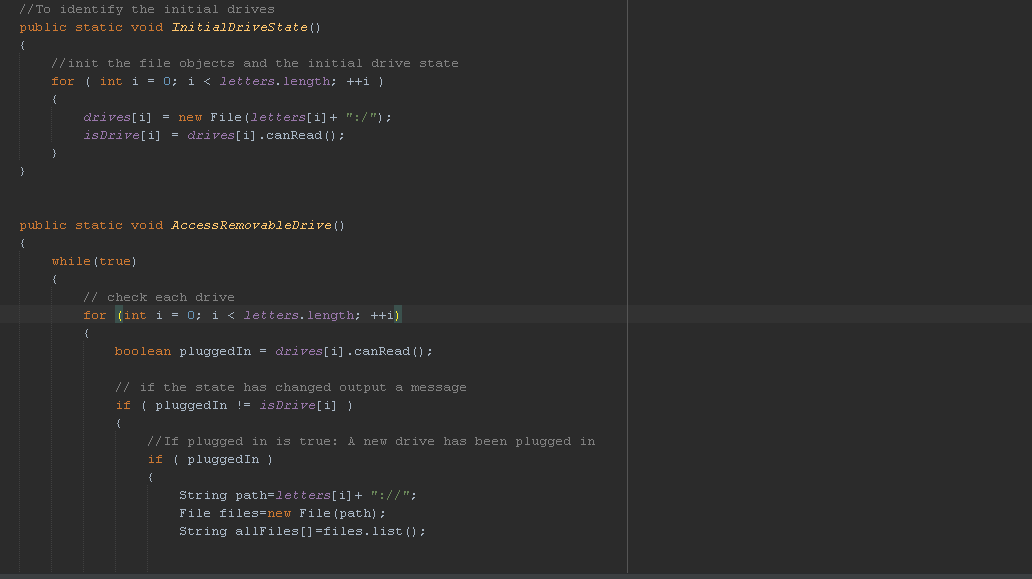


Figure 13

Figure 14 is a code used to retrieve some of the computer properties like a computer host name, computer ip address and the logged in user.

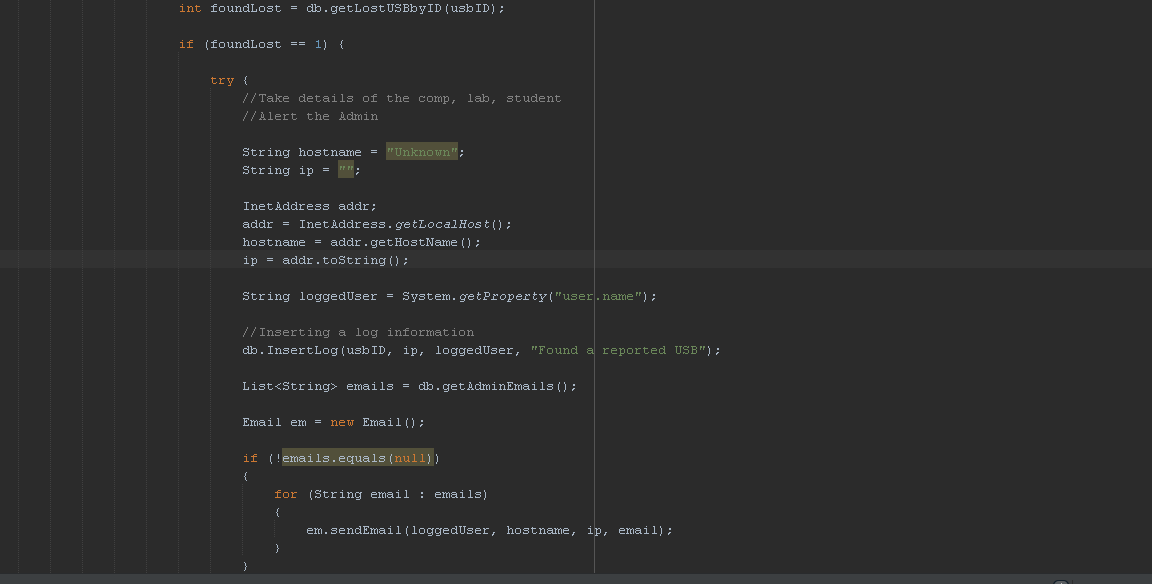


Figure 14

Figure 15 is a code used to store a text file (USBID.txt) that contains a unique UUID to the USB Flash Drive. After that a Runtime.getRuntime.exec() command is used to super hide the text file on the USB Flash Drive using a Windows BATCH command (attrib +s +h “path”).

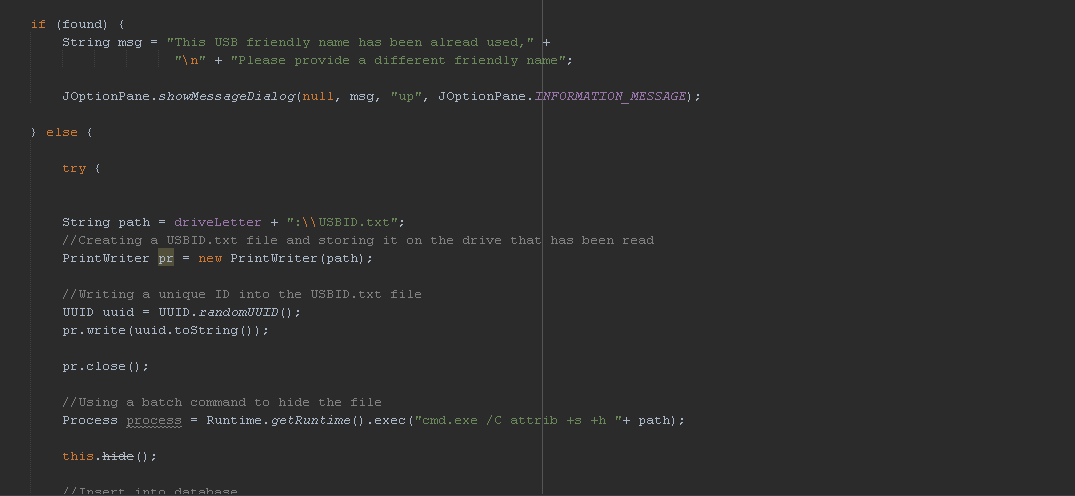


Figure 15

Figure 16 is a code used to encrypt and decrypt a file using AES encryption algorithm.

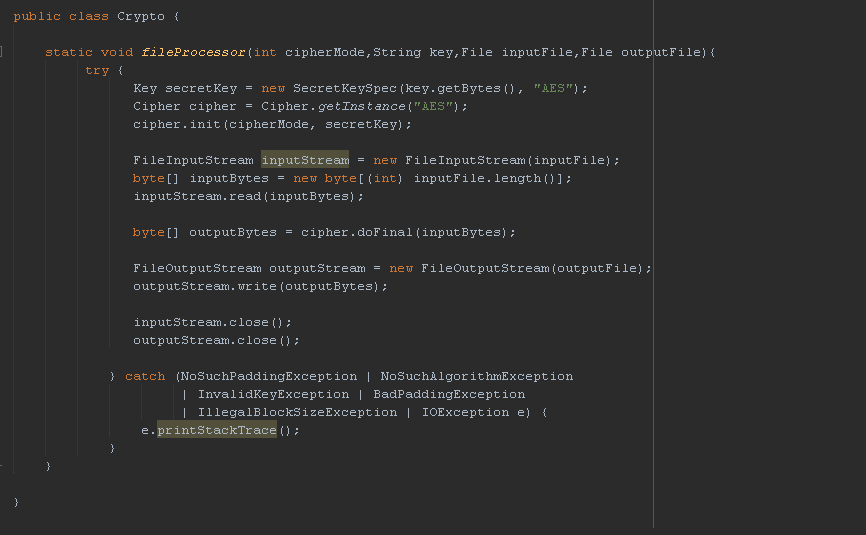


Figure 16

### Reference source of code

A code for identifying a newly inserted drive is retrieve from stackoverflow web site

Link: <https://stackoverflow.com/questions/44033740/variable-declaration-from-one-class-to-another>

A code for retrieving some of the computer’s properties is retrieved for the oracle web site

Link: <https://docs.oracle.com/javase/7/docs/api/java/net/InetAddress.html>

A code to get the Windows username is retrieved from stackoverflow web site

Link: <https://stackoverflow.com/questions/19990038/how-to-get-windows-username-in-java>

A code to run Windows BATCH commands via JAVA is retrieved from tutorial point web site

Link: <https://www.tutorialspoint.com/java/lang/runtime_exec.htm>

A code to Encrypt and Decrypt Files and Folders is retrieved from the oodles technologies web site

Link: <https://www.oodlestechnologies.com/blogs/Encrypt-And-Decrypt-A-File-In-Java>

A code to copy files using a Windows BATCH command is retrieved from stackoverflow web site

Link: <https://stackoverflow.com/questions/5587643/runtime-getruntime-exec-problem-with-copying-files>

## Conclusion

This document is meant to provide an overview of the technologies that were used to develop the **up** application. There should be an understanding of why these technologies were used and where were they used. Some of these technologies have their advantages and disadvantages, but they achieved their purpose in the development of the **up** application.

# Chapter 5: Test Plan and Test Results

## Introduction

The purpose of this chapter is to provide test cases of the **up** application. The chapter will describe the tests that were carried on and the results from the tests. These tests will be carried as a software would test the code, as a black box tester (normal user would test the functionality of the application without touching the code, but using the controls). And also provide a satisfaction of a user on using the application.

## Testing

### Test Cases

Test cases are a black box tests that are carried by the developer and a user. The user will enter input into the controls of the system and compare the expected results against the actual results. Table 4 is the first test case that was carried out.

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT** | **Up** | | | |
| **PREPARED BY** | Nkosinathi Madalane | | **DATE CREATED** | 21/09/2018 |
| **FUNCTIONAL SPECIFICATION** | Register Student | | | |
| **TEST OBJECTIVE** | To test if the system validates the student details correctly. | | | |
| **USE CASE ID** | UC03 | | **TEST CASE ID** | TC01 |
| **TEST DATA (INPUT)** | Enter student details | | | |
| **Test No** | **Steps** | **Input Data** | **Expected Results** | **Actual Results** |
| 1 | Enter an invalid student number | Student number = s21000 | An error indication on the Student Number text box should show | The student number text box is highlighted in red indicating incorrect student number |
| 2 | Enter a correct student number but without an ‘s’ at the beginning | Student number = 214212483 | The system should accept the student number an prepend an ‘s’ | The system successfully registered the student |
| 3 | Enter a correct student number with an ‘s’ at the beginning | Student number = s214212483 | The system should accept the student number as it is | The system successfully registered the student |
| **TEST DATE** | **21/09/2018** | **TEST TAKEN BY** | **Nolubabalo Madalane** | PASS |

The second test was testing the student logging into the system. Shown on Table 5

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT** | **Up** | | | |
| **PREPARED BY** | Nkosinathi Madalane | | **DATE CREATED** | 21/09/2018 |
| **FUNCTIONAL SPECIFICATION** | Student Login | | | |
| **TEST OBJECTIVE** | To test if the system validates if the student should use correct details to login to the system. | | | |
| **USE CASE ID** | UC04 | | **TEST CASE ID** | TC02 |
| **TEST DATA (INPUT)** | Enter student login details | | | |
| **Test No** | **Steps** | **Input Data** | **Expected Results** | **Actual Results** |
| 1 | Try to login without entering any details | Student username = ‘” and Student password = “” | An error message show “you have entered incorrect login details” | The system shows an error message “you have entered incorrect login details” |
| 2 | Only enter the Student username and try to login | Student username = “Nkosinathi” and Student password = “” | An error message show “you have entered incorrect login details” | The system shows an error message “you have entered incorrect login details” |
| 3 | Enter incorrect Student username and incorrect Student password | Student username = “Nkosi” and Student password = “123” | An error message show “you have entered incorrect login details” | The system shows an error message “you have entered incorrect login details” |
| 4 | Enter correct Student username and correct Student password | Student username = “Nkosinathi” and Student password = “pass123” | The system should successfully login the user | The student successfully login the system and the student dashboard is displayed |
| **TEST DATE** | **21/09/2018** | **TEST TAKEN BY** | **Nolubabalo Madalane** | PASS |

Table 6 shows a test case for a student backing up the files on the system

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT** | **Up** | | | |
| **PREPARED BY** | Nkosinathi Madalane | | **DATE CREATED** | 21/09/2018 |
| **FUNCTIONAL SPECIFICATION** | Student backup files and folder | | | |
| **TEST OBJECTIVE** | To test if the student can back up the file and folders on the H: | | | |
| **USE CASE ID** | UC08 | | **TEST CASE ID** | TC03 |
| **TEST DATA (INPUT)** | The student selects a file and a folder to backup | | | |
| **Test No** | **Steps** | **Input Data** | **Expected Results** | **Actual Results** |
| 1 | Select a file and try to back it up | An up.sql file has been selected | A folder named backups containing the up.sql file should be created on the H: | A folder named backup has been created with the file up.sql on the H: |
| 2 | Select a folder and try to back it up | A folder cPanel that has cPanel videos has been selected | A folder named backups containing the cPanel folder with all the videos should be created on the H: | A folder named backup has been created with the folder cPanel on the H: but the cPanel folder was empty |
| **TEST DATE** | **21/09/2018** | **TEST TAKEN BY** | **Nolubabalo Madalane** | FAIL |

Table 7 shows a test case for file and folder encryption

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT** | **Up** | | | |
| **PREPARED BY** | Nkosinathi Madalane | | **DATE CREATED** | 21/09/2018 |
| **FUNCTIONAL SPECIFICATION** | Student encrypt files and folder | | | |
| **TEST OBJECTIVE** | To test if the student can encrypt file and folders | | | |
| **USE CASE ID** | UC06 | | **TEST CASE ID** | TC04 |
| **TEST DATA (INPUT)** | The student selects a file and a folder to encrypt | | | |
| **Test No** | **Steps** | **Input Data** | **Expected Results** | **Actual Results** |
| 1 | Select a file and try to encrypt it | An up.sql file has been selected | The up.sql file should be encrypted and unreadable | The up.sql file is encrypted and is unreadable: |
| 2 | Select a folder and try to encrypt it | A folder up that has two-word documents has been selected | All the files on the folder up should be encrypted | Both files on the up folder has been encrypted |
| **TEST DATE** | **21/09/2018** | **TEST TAKEN BY** | **Nolubabalo Madalane** | PASS |

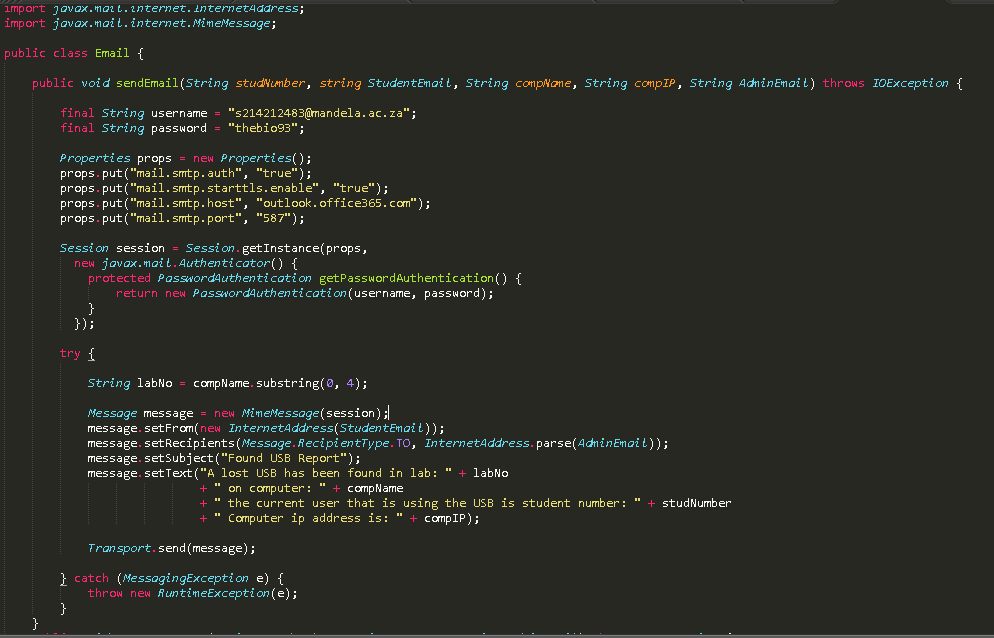
Table 8 shows a test case for decrypting files

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT** | **Up** | | | |
| **PREPARED BY** | Nkosinathi Madalane | | **DATE CREATED** | 21/09/2018 |
| **FUNCTIONAL SPECIFICATION** | Student decrypt files and folder | | | |
| **TEST OBJECTIVE** | To test if the student can decrypt file and folders | | | |
| **USE CASE ID** | UC07 | | **TEST CASE ID** | TC05 |
| **TEST DATA (INPUT)** | The student selects a file and a folder to decrypt | | | |
| **Test No** | **Steps** | **Input Data** | **Expected Results** | **Actual Results** |
| 1 | Select a file and try to decrypt it | An up.sql.encrypted file has been selected | The up.sql.encrypted file should be decrypted and be readable | The up.sql.encrypted file is decrypted and is readable: |
| 2 | Select a folder and try to decrypt it | A folder up that has two-word documents has been selected | All the files on the folder up should be decrypted | Both files on the up folder has been decrypted |
| 3 | Select a decrypted file and try to decrypt it | An up.sql file has been selected | The up.sql file should be still readable | The up.sql file is readable |
| **TEST DATE** | **21/09/2018** | **TEST TAKEN BY** | **Nolubabalo Madalane** | PASS |

### Code Testing

A code to send an email to the student emails using a Gmail account was tested and it took a long time for the email to go through. Then it was tested the second time using a student email account to a student email account and it successfully sent the email in less time. Figure 17 is a code example of the email code.



Figure

## User Interface Evaluation

### Informal User Testing

The way the interface was tested was by providing a user to use the system for the first time. The user could successfully register and login without asking a question. But when the user reached the student dashboard screen, the user did not know exactly what to do. So, a help function was created to encounter this problem. after the help was included the user used the help and got an idea of where to go for each function, but the interface kept it simplistic design.

### User Satisfaction

Table 9 show the user satisfaction based on the interface aspect of the system. The numbers 1 – 5 indicate the level of satisfaction with 1 meaning very unsatisfied and 5 meaning very satisfied. The number that the user selected is highlighted green and the comments were made by the user on each section on the Table 9.

Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | The system was easy to learn. | 1 | 2 | 3 | 4 | 5 |
| Comment | After using help, I found the system very easy to learn and use later on. | | | | | |
| 2 | The system was easy to navigate. | 1 | 2 | 3 | 4 | 5 |
| Comment | The simple design shows me where to go for each function. | | | | | |
| 3 | The system prevented errors by limiting text input. | 1 | 2 | 3 | 4 | 5 |
| Comment | I never saw and limiting text input function. | | | | | |
| 4 | Feedback clearly communicates if an action succeeded or not. | 1 | 2 | 3 | 4 | 5 |
| Comment | For all the actions I made I got a feedback on my success. | | | | | |
| 5 | The interface makes good use of screen real estate (not too cluttered and not too much wasted space) | 1 | 2 | 3 | 4 | 5 |
| Comment | The system has a simple design and uses the space pretty well. | | | | | |

## Changes Implemented

During the implementation of the system, some of the planned designs and implementations may change or be excluded. There were few changes made during the implementation of **up** from the design to the code implementation.

### Implementation

A function to backup files online to a google drive was changed due to the required plugins and configuration that could take time. So, instead of backing up the files on the google drive the application backs up the files on the student H:.

### Design

The ERD changed by excluding the Active Directory table. When the function of retrieving the computer properties with a JAVA java.net.InetAddress library, a need for an Active Directory was eliminated. And some of the use cases also changed.

## Conclusion

Testing a system shows unexpected results that the developer did not notice. This helped to identify some of the unnoticed errors in the code and also on the design of the application. The test is an eye opener that a system may not react the same way when it is used be a different user instead of the developer of the system. This insures the system is in good quality before it goes into deployment.

# Chapter 6: Deployment Documentation/Installation Guide

## Introduction

This section will provide guide on what is needed to run the **up** application on any computer. The operating system that is required will be listed, the Hardware and Software requirements and the programs that are needed.

## Operating System used

During the development of **up** the Operating System that was used is Windows 10 Home Single Language, 64-bit

## Development Environment

NetBeans 8.2 is the IDE that was used to develop an **up**-JAVA application. A JDBC library was used to connect the application to the MySQL database and the java-mail library is used to send email.

## Components Needed

1. To run **up** you will also need a version 8 JRE (JAVA Runtime Environment) and a JDK (JAVA Development Kit).
2. Once the JRE and the JDK are installed you will only need a copy of the executable **up .**jar files that can be easily ran by double clicking on them.

## User Guide

The up.jar will be running on the background, the screens will only appear once the user has inserted a USB Flash Drive on the computer. Figure 18 is a pop-up that will show once the user inserts a USB Flash Drive on a computer.

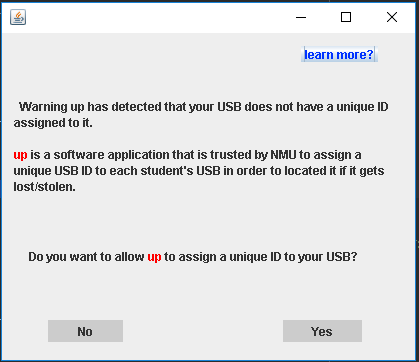
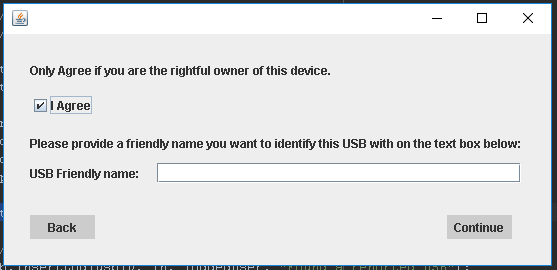


Figure 18

A user can click the “No” button if they do not want the application to assign a unique ID to a USB Flash Drive or click the “Yes” button to assign a unique ID to a USB Flash Drive. Once the “Yes” button has been clicked Figure 19 screen will show.



A user should only click this check box if they are the actual owner of the USB Flash Drive

Figure 19

The user should click the check box to agree as the rightful owner of the USB Flash Drive and type a friendly name they want the USB Flash Drive to be identified with. Then click continue when they are done.

The upClientApplication.jar is the application that is meant for the student to use in order to manage their USB Flash Drive. The first screen the user will see on the upClientApplication.jar is a login screen shown on Figure 20. Where the student can either register of login to the application.

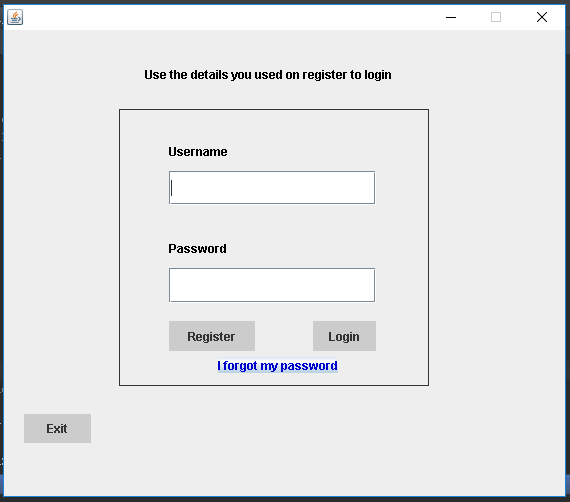


Figure 20

Figure 21 is a screen to register

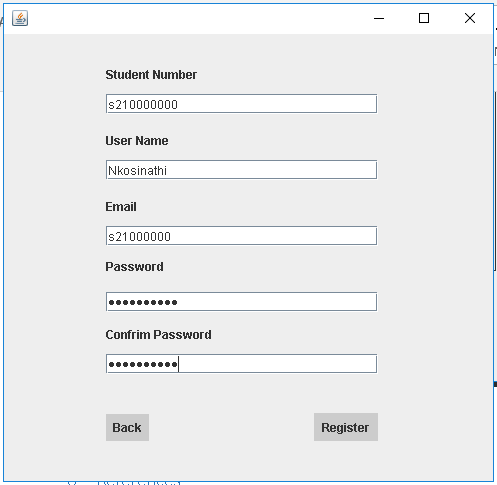
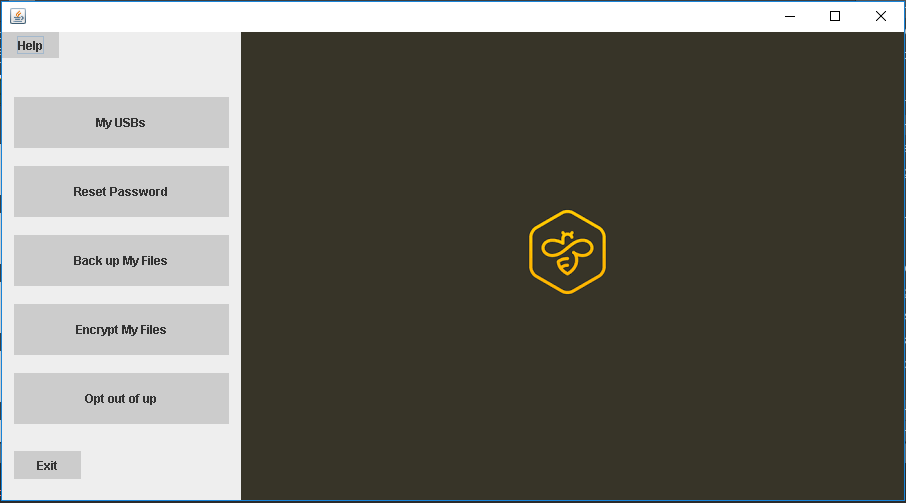


Figure 21

The details that are used on the register screen can be used to login on the screen show on Figure 20. Once the student has successfully logged in, the first screen of the application they will see is a student dashboard shown on Figure 22. Figure 22 shows and gives a little description of what each button does on the application.



To deregister from the application

To Encrypt or Decrypt Files

To Copy Files to a student H: (H Drive)

To Change a Password

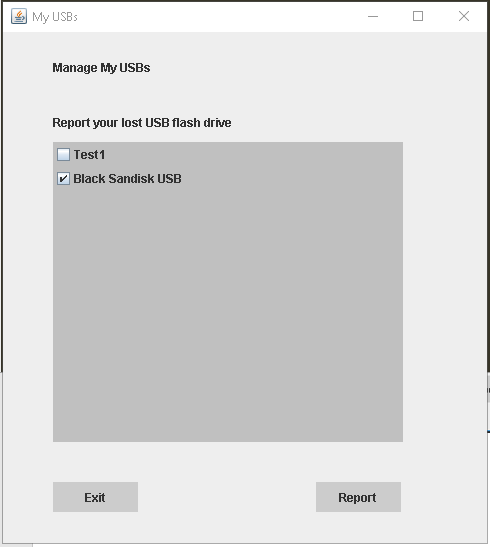
To list all your USBs and report them as lost

Help Button

Figure 22

To Exit the application

A help button will show a screen similar to Figure 9 were there is a short description of each button on the dashboard. Figure 23 will show a list of USB Flash Drive that has been registered and belongs to the logged in student.



Click the Report button

Click on the Check box of the USB you want to report

Figure 23

A student can click on the check box of the USB Flash Drive they want to report as missing and then click the report button. The Administrator will receive an email reporting the that particular USB Flash Drive (“Black SanDisk USB”) of this student has been lost. The lost status will be assigned to the USB and any time after this time, when this USB has been inserted on any computer at campus, an email containing the computer’s ip address, host name, Lab Number and the logged in user (that would be the student number of the student that is using the USB that was reported as lost) the lost USB has been inserted in.

A student can also Reset a password by clicking on the Reset button shown on Figure 22. A student is required to type in the old password first on the screen shown on Figure 24 before resetting the password. Figure 25 shows the screen to type in the new password and confirm the new password to avoid typing errors that might happen during resetting of a password.

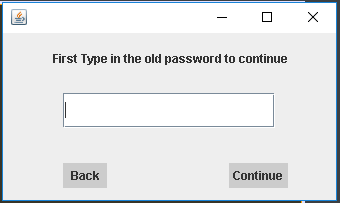


Figure 24

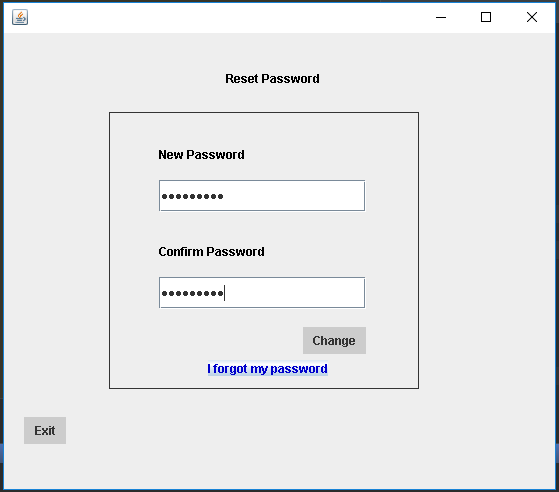


Figure 25

If the student wants to back up files on the USB Flash Drive. They may click on the “Back up My Files” button which will open a JAVA File Chooser shown on Figure 26 to locate a file or folder they want to back up. After choosing a file or a folder and clicking open on the JAVA File Chooser the file/folder chosen will be copied to the student’s H:.

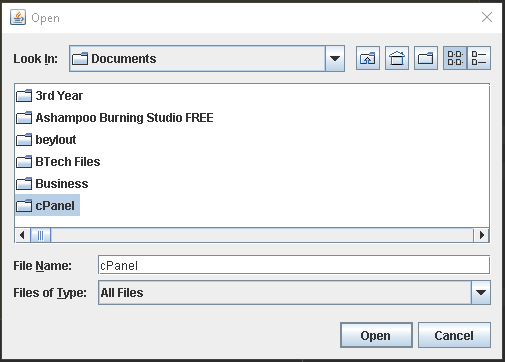


Figure 26

To protect the files of the student from unauthorized access **up** also provides a file/folder encrypt or decrypt function. After clicking “Encrypt My Files” button shown on Figure 22, a student can either encrypt or decrypt a file/folder by choosing on the options on the screen shown on Figure 27 and JAVA File Chooser like the one on Figure 26 will be shown to choose a file/folder to encrypt or decrypt. After encrypting a file, it will be in a format shown on Figure 28. The encrypted file is encrypted using the AES file encryption algorithm with a secret key provided by the developer on the application. Therefore, the file cannot be used since only the developer knows the secret key to provide for decrypting the file. When you decrypt it back it will be back to its original form shown on Figure 29.

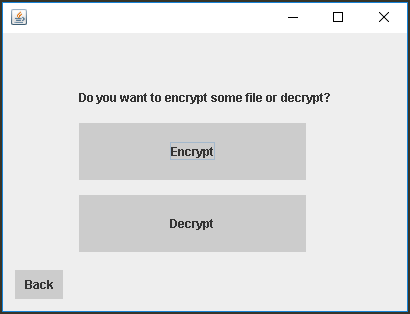
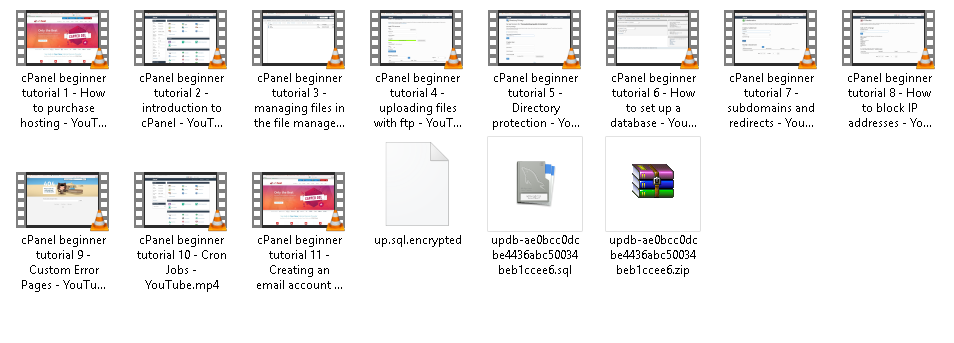
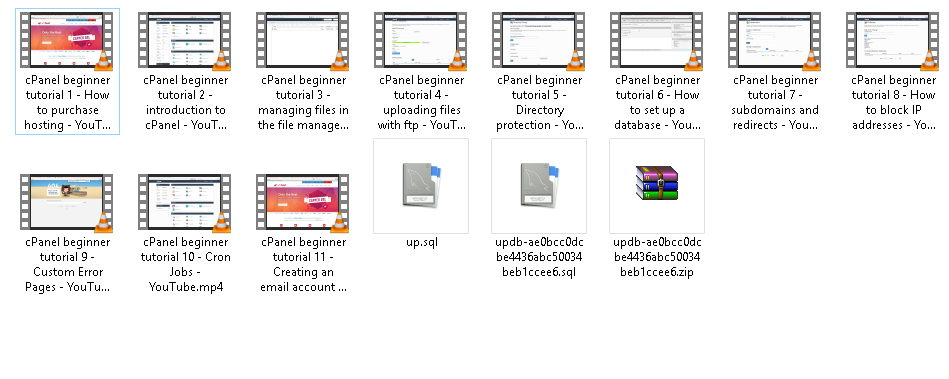


Figure 27



Encrypted File

Figure 28



Decrypted File

Figure 29

To deregister from the application a student can simply click the “Opt out of up” button.

The first screen the admin will see after logging in is an admin dashboard shown on Figure 30.

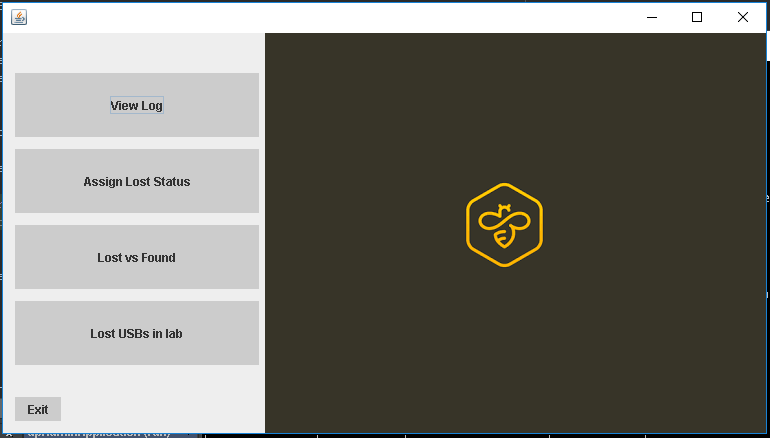


Figure 30

To view a log created by the application click the “View Log” button, which will lead to Figure 31 screen. When a student goes directly to the admin to report a lost USB Flash Drive, an admin may click the “Assign Lost Status” button and search the USB using a student details then report the USB as lost. Figure 32 is a screen shot of the “Assign Lost Status” screen.

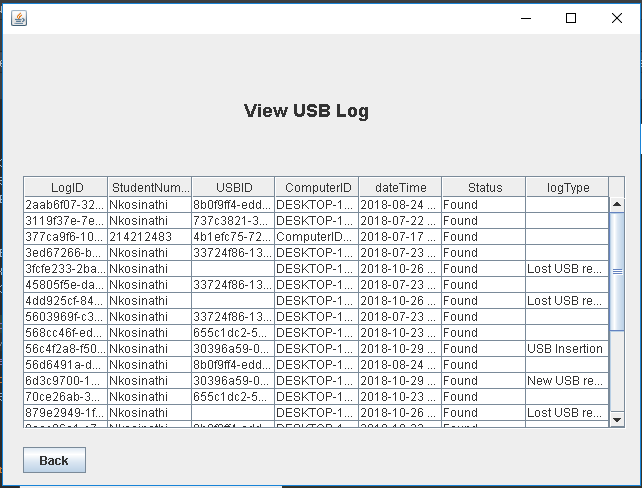


Figure 31

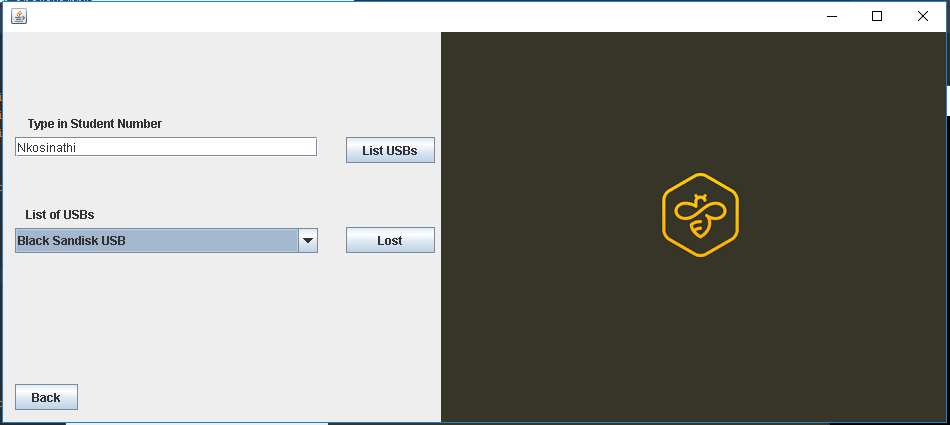


Figure 32

Somethings an admin may be interested in is to know how may USB Flash Drives had been lost versus the number of found USB Flash Drive each year. Figure 33 shows that report and Figure 34 shows how many USB Flash Drives that were reported as lost had been found on each lab.

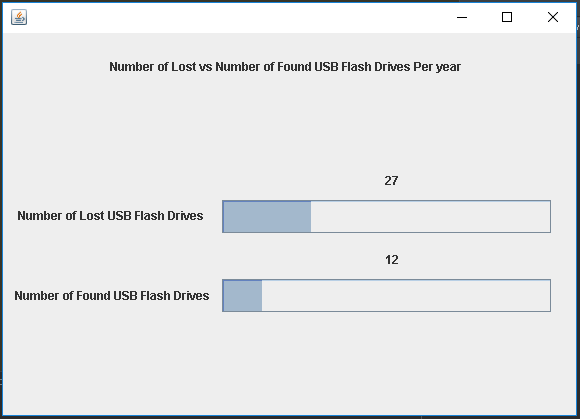


Figure 33

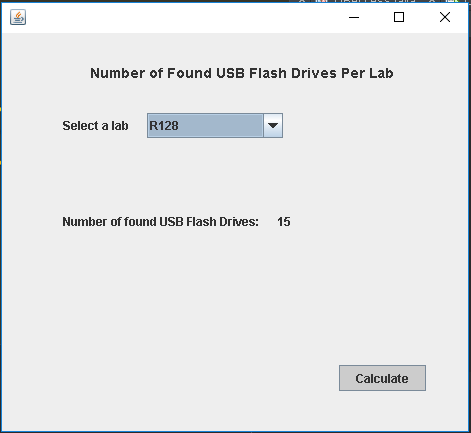


Figure 34

# Chapter 7: Conclusion

## Introduction

This Chapter will cover the conclusion of the project. It will be talking about the goals of the project, the problems that were encountered and the achievements made. This chapter will also include future research and plans looking forward to make **up** better and work in the real world. After reading this chapter a user faced with a similar problem should at least know some of the challenges and how to encounter them.

## Project Goals

The main goal of this project was to assist on an existing problem on campus were students lose their USB Flash Drives on daily bases. After losing a USB Flash Drive myself, I realised what counted the most was the information that was contained in it and that I might never get it back. The project was then implemented to locate students lost USB Flash Drives in order for the student to find back what belongs to them. An also to protect student’s information contained on those USB Flash Drives by providing a function for backing up their data and the function to encrypt their data.

## Achievements

The biggest achievement of this project was to actually prove that the application can trace a lost USB Flash Drive by immediately sending an email containing a computer ip address, lab number, logged in student’s student number and the USB Flash Drive that was found. This information can then help the admin to know exactly at what time the USB Flash Drive was found, on which computer on which lab was it inserted and which student was using that USB Flash Drive. Other achievements of the application were being able to back up the files on the H: and to encrypt and decrypt the files successfully.

## Problems Encountered

There were difficulties encountered through the development of the **up** application. Some of them was allowing the application to create a google drive back up, the issue was installing a gradle plugin and use it on the NetBeans IDE.

Another problem was assigning the USB ID on the Flash memory to make it inerasable. The plan was to write my code into the USB Flash Drive using STM32-F4 or Arduino to add an attribute named “USB ID” with the value of the unique ID for each student USB Flash Drive. But the manufacturers of the USBs make the Flash Drives one time writable, meaning it is protected from writing any code on its microcontroller.

At the beginning **up** was also going to have a function to use Windows BitLocker to automatically lock the USB Flash Drive and use the original owner password to automatically open the USB Flash Drive to prevent the owner of the USB Flash Drive having to type in the password every time they use their own USB Flash Drive. How this was going to work: since BitLocker provides a terminal interface named manage-bde that allows a user to enter their password to open the USB Flash Drive via a Windows Command Prompt, the up application was going to use the “getRuntime.exec” method to call the manage-bde command and enter the actual student password from the database on the background. This would only happen when **up** has detected the logged in student the same user of the inserted USB Flash Drive. The problem with this function was that BitLocker protects commands coming from any other source than the actual user input on the Windows Command Prompt.

## Future Research

From here the future plans for **up** is to use a device like STM32-F4 or Arduino to demonstrate inserting a unique ID on the Flash memory to make it inerasable. This demonstration can be taken to the USB Flash Drive manufacturers to suggest a solution to provide a unique ID attribute to each USB Flash Drive for USB Flash Drive tracking purposes and other purposes like computer forensics (hacked companies may be able to trace a USB Flash Drive that plugged the virus).

Another research is to find an open source USB encryption software that can work well with **up**.

## Conclusions

My final conclusions are that up can be a helpful application. Of cause, there are still improvements that should be made. But this should be an eye opener to Information Security. Information Security is mostly Concerned about protecting information of an organisation, but protecting a USB Flash Drive may also mean protecting an organisation’s information assets. Since USB Flash Drives are widely used even in organisations, they may pose a threat to organisations. (Galea, 2015) says “Although USB Flash drives are extremely useful devices for transferring data, they do come with security risks. Employees using USB drives at home and then plugging them back into the corporate network is a security concern for any IT Administrator.”. My wishes are for **up** to be the awareness to protect all the USB Flash Drives in the world.

# Progress Report

The current problem at the university is that many students lose the USB Flash Drives and the data in it. It is a fact that many students do not backup their school work, but use their USB Flash Drive as a main storage. Some students lose their final year projects or research work they had on their USB Flash Drive.

The objective of this project is to find those lost USB Flash Drives, to protect student’s data and to make an awareness of Information Security. The challenges that were faced through this project was technical problems like backing up files online, embedding BitLocker on **up** and inserting the USB ID on a Flash Memory to make it inerasable, but among those problems the major challenge was actually learning a different programming language which is JAVA. There were many unexpected challenges with JAVA, like connecting to the database, syntax for the if statement conditions and the foreach loop (which is a for loop in JAVA) and knowing how to dynamically generate controls to a JFrame form. The challenge of not being able to insert the USB ID on a Flash Memory made me look for alternative way to achieve my goal.

When I successfully located the USB Flash Drive on another computer, this made me learn: it is easy to achieve such goals than we think, but we just have to give it a try before we think it is difficult and give up.

# References

(2018). Retrieved from Techopedia: https://www.techopedia.com/definition/13576/bitlocker

*About tv rename*. (n.d.). Retrieved from tvrename.com: http://www.tvrename.com/about.html

Adam, P. (2010, 02 13). *Automatically Organize Your Downloaded, \*Ahem\*, TV Shows*. Retrieved from Life Hacker: http://lifehacker.com/5471112/automatically-organize-your-downloaded-ahem-tv-shows

Dillet, R. (2015, 07 15). *TVShow Time Releases Sleek New App For Its Community Of TV Addicts*. Retrieved from Techcrunch: http://techcrunch.com/2015/07/15/tvshow-time-releases-sleek-new-app-for-its-community-of-tv-addicts/

*How to Securely Encrypt a USB Flash Drive*. (2018). Retrieved from online-tech-tips: https://www.online-tech-tips.com/computer-tips/encrypt-usb-flash-drive/

Khoury, R. E. (2013, 09 27). *Stay on Top of the New Fall TV Season with SeriesGuide*. Retrieved from App Storm: http://android.appstorm.net/reviews/entertainment-reviews/stay-on-top-of-the-new-fall-tv-season-with-seriesguide/

Lagace, M. (2018, March 7). *How to find your lost Android phone | Android Central*. Retrieved from androidcentral: https://www.androidcentral.com/how-track-android-phone

Matei, A. (2014, 09 05). *theRenamer*. Retrieved from Softpedia: http://www.softpedia.com/get/System/File-Management/theRenamer.shtml

Oxford University Press. (n.d.). *Definition of disorganized in English*. Retrieved from Oxford Dictionaries: Language matters: http://www.oxforddictionaries.com/definition/english/disorganized

*theRenamer*. (n.d.). Retrieved from The Renamer: http://www.therenamer.com/

*TV SHow Time*. (n.d.). Retrieved from TV SHowTime get notified when new episodes are out: https://www.tvshowtime.com/en

Uwe Trottmann. (n.d.). *Series guide*. Retrieved from Series guide: https://seriesgui.de/

Williams, M. (2014, 09 23). *theRenamer 7.69*. Retrieved from PC Advisor: http://www.pcadvisor.co.uk/download/audio-video-photo/therenamer-769-3214212/