# Analysis Section

## Feasibility Study (background to - and identification of - the problem)

Name of Client: Matthew Wilson

Working Title of System: Saviour Backup System

**What is the purpose of the new system?**

To allow users to backup removable media easily, quickly and automatically, so that in the event of something happening to the media, the backup would still be intact, and the software can restore it to another device. This can also be done through the software.

**What are the anticipated benefits of the new system?**

Reduces the chance of data loss on removable media, because the media can be backed up in the background automatically with little affect to the user.

**What useful outputs / information will the new system produce?**

Backup files of the removable media and config files on host computer.

**Who will enter data into the new system?**

The data will be entered by no one. There is no data to enter. The configuration for the USB manager for backups is generated automatically, from a template. The settings from that are set up by the user or the owner of the media.

**Who will make use of the outputs / information?**

Any business, big or small, even home users that want to keep their data backed up seamlessly. If someone has important data that they carry around on a USB drive, the software will stop them losing the data in the event of something happening to the drive.

**Will the new system need to permanently store data?**

Yes, it will store the backed up data on the client computer, and maybe on an FTP server they own. There will also be configuration data stored on the USB device.

**Is there an existing system that will be replaced?**

No, after research there is no other piece of software like this in existence on the global market, however there may be bespoke software solutions in place. This system isn’t designed to replace any other system, therefore there is no consideration needed for this product to meet any prior requirements or interfaces.

**Is there an existing system that the new system will need to interface with?**

Other than the windows operating system, no. Everything will be done by the program with no need for any other systems, other than the software needed for the program to run (.Net framework etc.)

**Does the new system need to run on any particular hardware?**

It must run on a windows based machine, in the desktop environment. It will be compiled for 32-bit to maintain support for different computers, and can be compiled for any different version of windows (8, 7 etc.).

**Outline a possible solution**

Programming Language: The program will be written in C#, with XML for the config files. Small pieces of other languages may be used (possible SQL and Batch, and **MAYBE** small amounts of C++ for file management)

File System: The file system will be windows based, and will be installed onto the client’s machine. Visual studio will make the installer, to make sure there are no problems with it there.

User Interface: The user interface will be using windows forms mixed with DotNetBar, which allows it to look nicer than stock. There will be some sections where console readout will be used for status.

## Formal Investigation

### **Interview**

***What is needed***

1. Can you tell me more about what you want the system to do?
   * Backup on ejection (ejection through program that is)
   * Simple user interface
   * Resource friendly
   * Backup restore in case of loss or replacement of drive.
   * Only backup drives that have been set up.
2. Does it need to run on any particular hardware?
   * No
3. Does it need to use any particular software?
   * No, all run inside windows
4. Should it be able to interface with any other systems?
   * No, run silently in the background

***Problems***

1. What are the benefits of the new system to companies and home users?
   * Piece of mind that if I lost my important data, then there would be a copy of the data still. However, I would still question the need for it because I use cloud, but loads of people still use USB devices
2. What would be an indication that this system is beneficial over other products?
   * That I have piece of mind that it does it itself.

***The users***

1. Who will use the system?
   * Small networks and home users
2. Are they likely to be competent IT users?
   * No
3. What level of user documentation and/or training will they need?
   * Basic overview of how to turn it on and make sure its running

***Security/Compliance***

1. Should the system store user information for analytics etc?
   * No, just make it work, simple as.
2. How secure should the software be (Password protected etc)?
   * No, it doesn’t worry me. The computers are likely to be password protected.
3. How should the data be stored (settings for backup, drive identity etc)?
   * Just as a basic folder, no compression.

### **OBSERVATION RECORD**

There is no custom system in place, or in existence that allows automatic backup of USB drives. The current method that people would use to back up a device would be to copy the files off manually through their OS. This method is not perfect, as the user would have to remember to back up the system every time they use it, which can be forgotten about rather easily. Unfortunately there are problems with this, the copy process on the OS doesn’t allow files to be compressed once they are copied, or at least not automatically.

After conducting some research into if people back up their devices at all, very few people did. This can be problematic because if they lose their data, then there is no alternative copy. After this conversation, all users not backup their devices.

### **DOCUMENT ANALYSIS**

There are no documents from any past system that needs to be analyzed. All I have to go on is what the client wants from the system at the moment.

## Description of the Current System

There is currently no system that needs to be updated or replaced. And there are no other systems that do anything like what I am designing the system to do. There are other systems that do roughly the same thing (backup data), but they aren’t appropriate really to this because they aren’t designed for removable devices.

## System Requirements

The new system will need to backup USB drives when they are inserted into the computer. The system will be able to back up many drives simultaneously, and still not affect system performance too much. The system will need to back up the files securely and properly, and override the backup when the drive is inserted and files have been changed on the drive.

The system is designed to remove any risk of work loss when the user uses a portable storage device.

## Identification of the Prospective User(s) and their skills

The system will be used by anyone that uses USB drives, whether it’s for small businesses, or home users. Any user will set up the program and use the program.

Chances are, the people that are going to use the program are going to have a very basic IT knowledge, however there may be people that also very computer savvy. I want to try and cater for both extremes, so the system will be very easy to use, but can be customized if needed.

The system will be very simple and easy to use so that there is very little training needed, so they can just start using the program instantly.

## Constraints/Limitations

The main limitation for the program will be storage space. If the program runs and there isn’t enough storage space, then the backup will fail, and a user may lose their drive, and have no backup of it. One idea would be to compress the backups as they come onto the computer, but this would need to be cleared with the client first. In order for this program to work on a computer, it will need certain things installed on it before this software can work. This software is only going to work on windows, due to the programming restraints, although most users are on windows anyway.

## Data Source(s) and Destinations(s)

Data will be coming from user input, and the device the user wants to back up. The destination of the data will be the directory that the backup will be stored, and the database file.

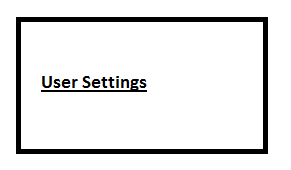
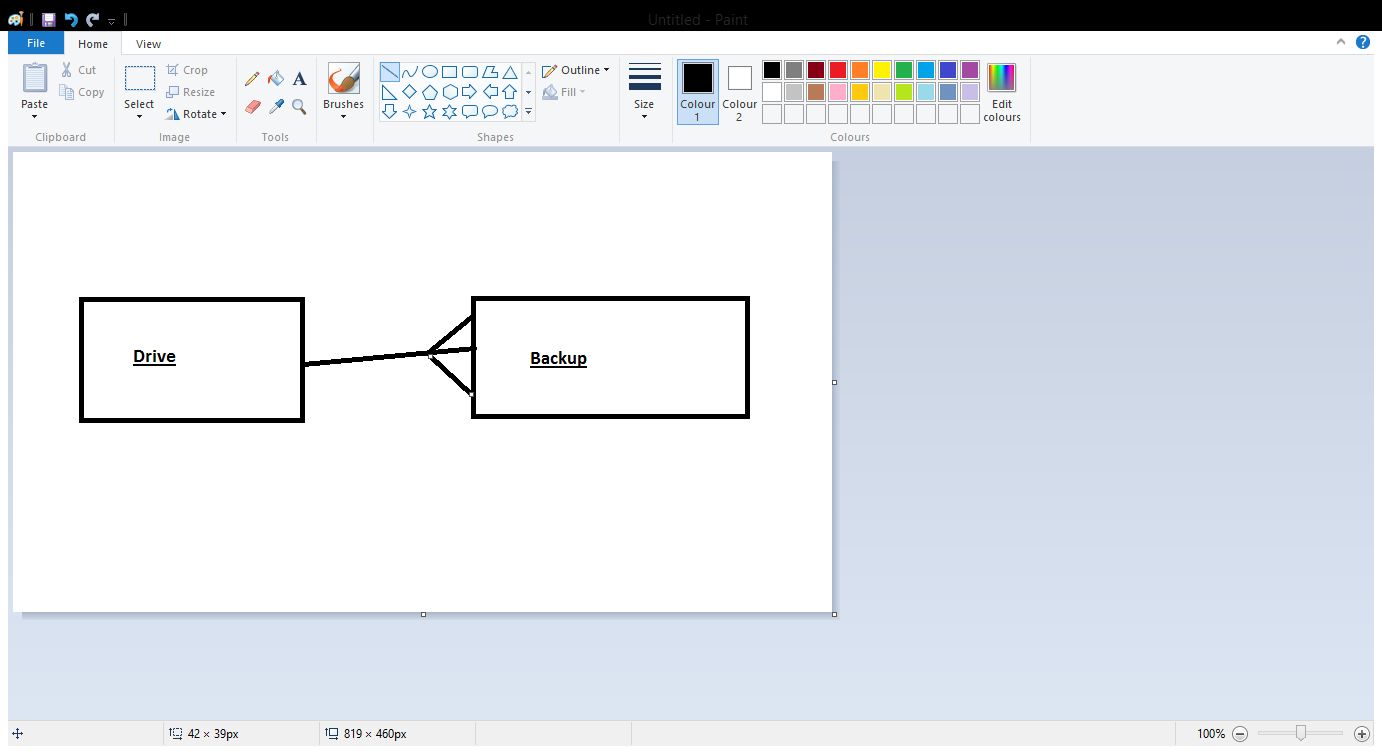
## Data Volumes

The system will be run every time the computer is started, and used heavily whenever the back. The customer base and user base is anyone, it will be released on the internet, I just have a client that wants to sell the software and needs someone to make it. Events will be triggered when the user starts up the computer, or when the user inserts a removable device into the computer. Data will be kept for all the time that the computer keeps a backup record of the drive.

## Analysis Data Dictionary (from the perspective of the end user)

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Description | Type (and length) | Validation |
| Drive ID | Allows global reference to the drive | Text (hash) | Must be a valid hash! |
| Drive Name | Stores the name of the removable drive | Text (11 char max (set by windows)) | Computer Generated |
| Drive Capacity | Stores the capacity of the removable drive | Number (no max length) | Computer Generated |
| Drive Format | Stores the format of the drive (type of file system) | Text (7 char max) | Computer Generated |
| Device Type | Stores what type of drive it is (removable, CD etc) | Text (15 char max) | Computer Generated |
| Date last backed up | Date of the last backup | Number (20 char max) | Computer Generated |
| Date Backup Set | Date that the backup rule was created | Number (20 char max) | Computer Generated |
| Latest Hash | Hash of all the files, done to see if they have changed since last backup | Text (hash) | Computer Generated. Must be a valid hash! |

## Entity Relationship Model

The user settings table stores all the customizations and settings that the user makes to the program. This includes any theme changes, and default directories.

## Object Analysis Diagrams

The system doesn’t use any extra objects, other than ones that are built into the operating system and framework.

## Objectives for the proposed system

1. Allow users to see what drives are connected in the main window of the application.
2. Allow users to add drives for backup through the software for future use in backup processes
3. Allow the user to backup content from the drive to their computer using details from the relevant backup record.
4. User can see per-file and overall progress of backup process through a dialog from the software to monitor the progress of their backup.
5. Allow users to modify details of backup records on their computer in case of a directory change or requirement changes.
6. Allow users to restore backups onto another drive in case of data loss on the drive, the primary use of the software.
7. Allow different users to have different backup rules on the same computer, for the same drive
8. Allow users to delete backup records from their computer if a drive is lost or not needed.
9. Use file compression to allow users to roll back through backups (compression needed to save file space)
10. Allow backups to run at the same time to different drives, to reduce the amount of time it takes to backup multiple drives.
11. Use (some kind of) hash to calculate of drive has changed since last backup, save time backing up redundant data.
12. The system should be self-documenting. It should be self-explanatory and need little in the way of extra documentation, meaning that anyone can use it easily and quickly.
13. The system must be able to run on windows 7, 8.1, and XP (both 32 and 64-bit versions), to make sure as many people as possible can use the software.
14. The user must be able to change its theme to match the users OS (in case the dark theme isn’t liked), allowing them to add a semi-personal touch to the software.
15. In order to be resource light, the system must use no more than 15kb of system memory when completely idle (in tray), meaning that the user can do more on their computer without the software impacting performance.
16. In order to be resource light, the system must use no more 1% CPU when completely idle (in tray), meaning that the user can do more on their computer without the software impacting performance.
17. The system must use no more than 5% CPU when GUI in use (adding backups etc), meaning that the user can do more on their computer without the software impacting performance
18. The system must use no more than 30kb system memory when GUI in use (adding backups etc), meaning that the user can do more on their computer without the software impacting performance
19. System must be protected so that no more than 1 backup is stored on the system per drive, per user. To remove redundant backups and save file space on their computers.

## Realistic appraisal of the feasibility of potential solutions

|  |  |  |  |
| --- | --- | --- | --- |
|  | Option 1 | Option 2 | Option 3 |
| Programming Language | **C#**  Compact code, and cross platform framework, however contains confusing advanced features. Simple data types with control. | **Python**  Very simple to use, however is interpreter so reveals source, and is slightly slower and resource heavy. No control over data types. | **C++**  Allows very custom and efficient code, and is cross platform. However requires a lot of code to be written. High control over data types and control over memory. |
| File Storage/database management system | **SQLITE**  Incredibly lightweight and highly supported. However lacks important data types. | **MySql**  Very simple and lightweight, however lacks some more advanced, and necessary data types. | **SQLCE**  Lightweight and easy to use, however is limited to 4GB per database file. |
| GUI | **C# (Windows Forms)**  Lightweight, native interfaces, with extensive controls, and libraries. However requires a lot of code to lay out. | **Tkinter**  Create very simple, lightweight interfaces, but lacks some more advanced controls that would be needed | **OpenGL**  Would create a very unique looking interface, at the cost of system resources. |

## Justification of chosen solution

I have chosen to do my coursework in C# for the back end code, Windows Forms with .Net framework for the front end, and SQLCE for the database.

I chose C# for the programming language because it is the one I am most confident with, and also it can do everything I want without needing anything else on the client computer. If I had chosen python, then the user would have to have the python runtime installed, and the source code would be available in the open. Python can be compiled, but it can be buggy sometimes. Alternatively, I could have used C++ for the back end, but its so complicated, and takes a lot of code to write something simple compared to C#.

I chose Windows form for the front end because it is by far the easiest thing to do in C#. Tkinter isn’t available in C#, its python only, and OpenGL is so complicated to make things simply appear on the screen, and it takes a lot of computer power to display things, unlike windows form which takes very little power to display.

For the database, I decided to use SQLCE. Mainly because is it really easy to implement into c#, and very lightweight. To keep it compact and lightweight, the databases have a limit of 4GB, but this shouldn’t be a problem because of how little data is being stored, and if its stored efficiently, then this wont be a problem.