**Coursework 2 Report**

**1. Design Changes**

For coursework 1, A design choice was to allow the user to create contacts to send money in and out to, this was then changed to payment type since the user would be making multiple different payments and grouping them into categories will allow the user to view their expenses in groups.

Originally databases were going to be used to store the users’ expenses, but this then changed to using JSON to read and write to the file, two files are made one is a file that stores the user login details and another file is one that stores all users expenses. JSON was chosen other a database as it is more simple to implement and JSON is more compact and will take up less memory

Also, certain class names have changed this was to make sure that the program is readable and that each class has a sensible name so another engineer could easily understand the program.

A method that was not implemented was being able to delete a payment this was a very important method that was not able to be implemented due to a lack of time management and not understanding data grid tables. This has negatively affected the program since the person will not be able to remove a payment that was created by mistake, and those payment details will end up in their report as well.

**2. Evaluation: Implementation choices etc**

To make the code more readable, sensible names were created for the variables and methods, and to use indentations to show the process of a piece of code happening. And the program was split into multiple classes this was to allow testing for each class separately to help search for bugs easier, the use of comments in code helps to explain to the programmer what the purpose of a method is thus improving the readability of the program.

Scalability is important in software development as it allows to help ensure as the application is used more, the application can deal with the increase in users and the program can still run at an optimal standard. To help the scalability of the code asynchronous threading techniques were used to help with the software scalability since I was able to place certain tasks behind other tasks and let them be completed asynchronously while more important tasks were placed ahead to be completed, for example this was method was used to read and write data to a file since as the file grows. Also to help with the scalability of the program data was split into two files, one file would contain multiple users login details, such as their username and password, and another file contained the multiple user's finances and each file would share a unique ID to display the correct data to the specific user, this helps with the scalability since as the program grows all the data will be split into two smaller files as opposed to one large file and using a shared ID will allow the program to search for necessary data quickly. This could have been improved by creating multiple files for each user for example user 1 would have a file that contains their expenses, and user 2 would have a separate file that contains their expenses. Storing the users LoginID in a variable when the program was running was to allow the program to not have to keep on reading the file to access the LoginID when needed this allows the computer to be able to do other tasks, and to only have the read a file once.

The program was designed using the SOLID principles. The first principle (single responsibility principle) was implemented so each class had one specific function for example the login class will only deal with the user logging in and finding their login details. And another class was used to create a constructor, the purpose of this is to allow each class to have a specific responsibility, so if the program were to be extended it would be extended based on that responsibility, this helps with the readability of the project since if new programmers were to work with the code, they will be able to understand the system since every class is based off one specific purpose. Implementing the first principle of SOLID also allows the programmer to test each class separately to find bugs easier and the maintainability of the program is improved as they could develop the program based on the specific purpose of the class.

The second principle of SOLID (open-closed principle) was implemented when creating an object of the users’ finances. A class called Finances was created and contained a constructor, and in another class, an instance of this object was created and then extended by setting specific values to each piece of data inside the constructor and was then written to a file.

The Fourth principle (integration segregation principle) was also implemented this was so each method did not have a large amount of content instead they were split into smaller methods and within a class or across different classes and then those methods were called this was to help ensure the code maintains its readability, and that the methods maintainability could also be ensured by checking each method separately for errors.