Text

Description automatically generatedText

Description automatically generated

*Module: GUI and Database Connectivity (COMP4604)*

*Title: Java GUI + Database Application Project*

*Design Document*

by

STUDENT NAME

Kevin Flynn (C18502203@mytudublin.ie)

STUDENT NAME

Carl Herdman (C18703995@mytudublin.ie)

Date of submitting document – 29/04/2022

Table of Contents

Cover Page1

Table of Contents2

Introduction3

Requirements4

Database Schema (Conceptual)5

Database Schema (Logical/Relational)6

Flowchart for Login7

Low-Fidelity Prototypes8,9

Usability Heuristics10

Notes11

References12

# Introduction

The task for this project is to design and implement a GUI Database-aware application using Java and a selected database. The Java code will be written using the NetBeans IDE and the selected database will be Microsoft Access.

Project teams consist of two students. The two students who will create this project are Kevin Flynn and Carl Herdman.

The classic three-tier architecture will be employed, which consists of the following:

1. Presentation Layer – GUI in Java (NetBeans IDE, Java Swing)
2. Application Layer – Java
3. Database Layer – Microsoft Access

The image below demonstrates a graphical version of this architecture:

Diagram

Description automatically generated

The theme (context idea) of this application will be that of a Library Management System, where people can buy and rent books, etc.

# Requirements

The application will:

1. Allow inventory control to be maintained – As the context idea is a library management system, it will be possible to store and maintain information of books, such as their title, authour, publish date, genre, price and stock.
2. Allow management to print lists of users and lists of stock – Like the previous requirement, this information can be maintained through the MS Access database and displayed using the Java GUI. Lists of users can be displayed in a ‘Users’ table and stock can be displayed in the ‘Books’ table (book stock).
3. Allow the addition and removal of users. – The ‘users’ table will display the added users. A user can be added from the ‘Add a User’ page and removed using the ‘Remove a User’ page. A user can only be removed if they exist in the ‘users’ table.
4. Allow the management to check overdue accounts – If a user has not returned a book to the library within a week, the information of the book and the user who bought it will be displayed on a table in the ‘Overdue Accounts’ section
5. Allow users to rent and purchase items/service – This is relatively simple for a library, as people can rent or buy books. It is only possible to issue a book for rent or purchase if the book exists in the ‘Books’ table and the user exists in the ‘Users’ table. If one of these are not fulfilled, an error message will display, telling the user.
6. Allow users to return rented/purchased items. A book can only be returned to the library if it has been issued. When a book is returned, the stock figure in the ‘Books’ table will increase by one, or if they bought two of the same books, the stock will increase by two, etc.
7. Allow users to print table information out to a printer or a file format such as PDF - An example of additional functionality.

# Database Schema (Conceptual)

The conceptual schema for this project will consist of an Enhanced Entity-Relationship (EER) diagram, as can be seen from the image below:

Diagram, text, letter

Description automatically generated

# Database Schema (Logical/Relational)

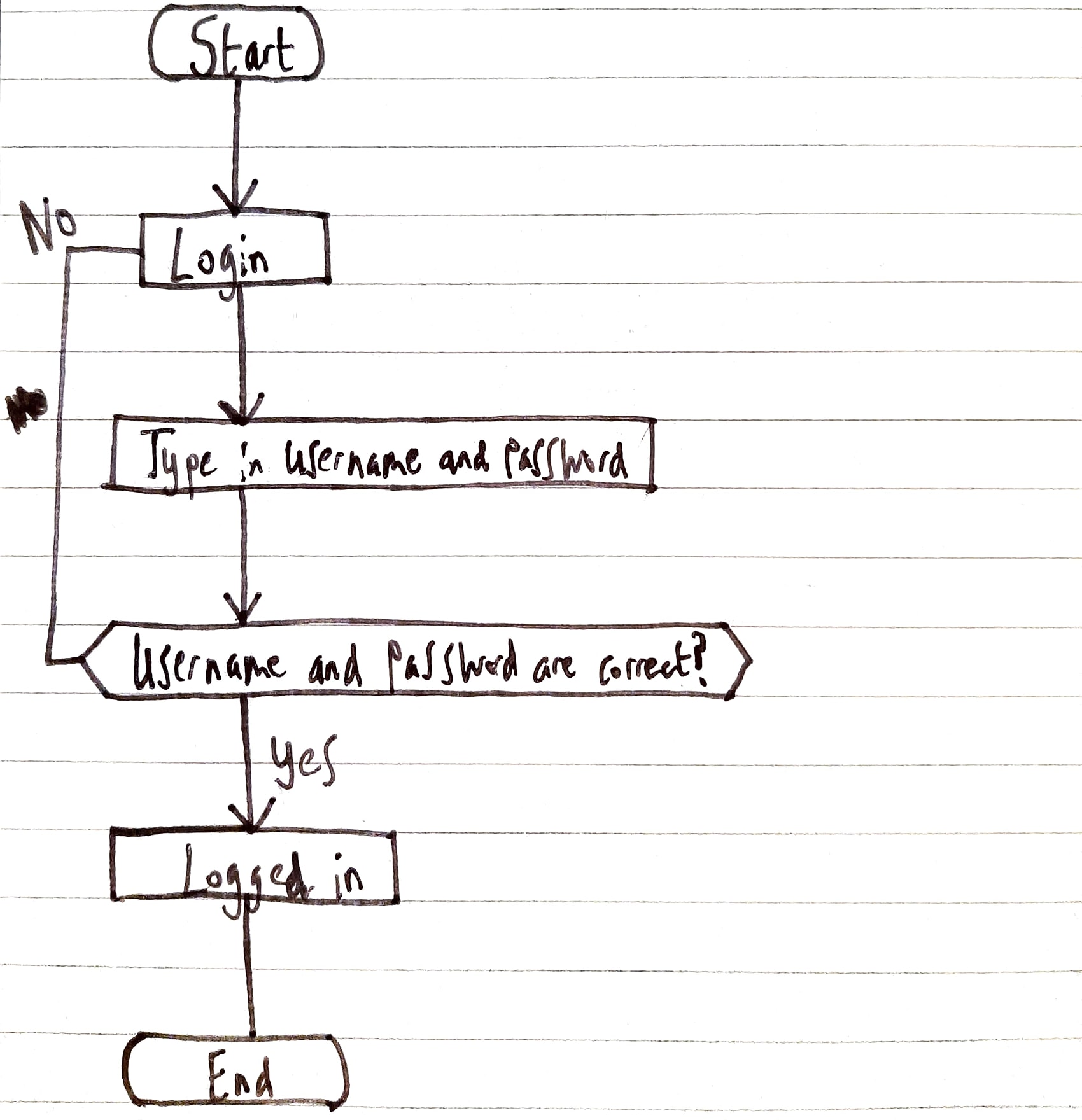
The logical/relational schema for this project will consist of the EER diagram mapped into the relational model, as can be seen in the image below:

A picture containing text, receipt

Description automatically generated

# Flowchart for Login

The image below shows the flowchart for the login process of the application:



# Low-Fidelity Prototypes

Two low-fidelity prototypes have been prepared with ten usability heuristics applied. The prototypes can be seen in the images below:

Diagram

Description automatically generated

Prototype 1 (this prototype will be used as it has a more technical layout compared to prototype 2)

Diagram, engineering drawing

Description automatically generated

Prototype 2

# Usability Heuristics

Usability heuristics are general principles to follow to ensure that there are no design issues with a user-interface. Jakob Nielson, a well-known web usability consultant, identified ten of the most important usability heuristics. The GUI for this application was created using these heuristics as a point of reference. The following information displays the ten usability heuristics and examples of how the application fulfils them:

1. Visibility of system status. The user knows exactly where they are, as they begin in the login page.
2. Match between the system and the real world. Words used in this application are universally understood and not specific to any arcane subject.
3. User control and freedom. Buttons such as ‘Logout’ and ‘Return to Home Page’ are clearly identifiable on each page of the application.
4. Consistency and Standards. The chosen prototype contains a consistent and accessible layout.
5. Error Prevention: The user will not be allowed to login without the proper credentials (username and password).
6. Recognition rather than recall. All of the necessary information is displayed within each page. The user will not have to worry about remembering certain things, as everything is easily available and identifiable within the application
7. Flexibility and Efficiency of use. Efficient application with tabs on how to proceed from the home page.
8. Aesthetic and Minimalist Design. A minimal design is implemented, with custom font and colours that are easy on the eyes.
9. Help users recognise, diagnose and recover from errors. Error messages are implemented throughout the application. For example, if the user issues a book that does not exist within the database, the error message ‘This book does not exist within the database’ will display.
10. Help and documentation. The necessary steps to follow throughout each page are easily identifiable to any user. If the user faces any problem throughout the application, they will be told.

# Notes

Notes followed for this design document:

Text, letter

Description automatically generated

# References

Below is a list of references used for the application:

<https://www.ibm.com/docs/en/db2/11.1?topic=sql-updating-data-in-tables-using-preparedstatementexecuteupdate-method>

<https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html>

<https://docs.oracle.com/javase/tutorial/uiswing/misc/printtable.html>

<https://docs.oracle.com/javase/tutorial/uiswing/layout/card.html>

<https://www.tabnine.com/code/java/methods/javax.swing.JPanel/revalidate>

<https://brightspace.tudublin.ie/d2l/home/179682>

<https://www.guru99.com/download-sample-test-case-template-with-explanation-of-important-fields.html>