IT721

Software Engineering

Final Project

Please sign the following statement: "I declare that this assignment submission will be my own work and I will not collude with anyone else on the preparation of this Assignment."

Name / Student ID: Benjamin King - 2014006529
Henry Sly - 2014001988
Sharmaine Rufford - 800987

Content	Mark	Result
System	35	
Debugging and Testing	10	
Implementation	5	
Presentation	10	
Sum	**60**	

Library Catalogue Application

SIT Paper

ITC721 Software Engineering

HSB Software Solutions Limited

Ben King:

Project Manager / Database Designer / Backend Developer / Tester

Sharmaine Rufford:

Administrative Assistant / UI Designer / Database Assistant

Henry Sly:

Backend Developer / UI Developer / UX Developer/ Tester

Weekly Meetings:

Wednesday 12pm – 1pm

Table of Contents

1.0 Project Introduction

The company HSB has been asked to create an application catalogue for a new library being built as part of the new development in Invercargill. The library (which is called “The Library”) will have a new application to use called **Rent a Leaf.** It has the ability to read barcodes, loan books and create different levels of access per membership. There will be a separate application called **Snatch and Go** for the self-service check out option.

HSB has been given flexibility with the application with small requests from the client. Their main request is for the application to be easy to use, reporting on books, user levels and not a large amount of clicking to get to their destination.

1.1 Objectives

The project’s aim is to create an application catalogue for “The Library” as they do not have one. This will be required before the opening of the new library and ready for the staff to use. This application will need to have a database set up with security for its users and members. There are several library catalogue systems available, so HSB is required to make this stand out from the rest.

With the design for this project, the members who have been chosen for the team have skills, and experience, in the areas of UI Design, and UX Design, allowing for a much more in-depth, and user-focused approach to the design. This will help with efficiency, usability, less training required and overall user comfort.

The new application will aim to meet the needs of the users by creating a reliable and efficient user experience. This will be done through developing a new code base, that shifts from messy legacy code to an OOP (Object-Orientated Programming) approach. It will also provide a user friendly, and stable platform.

The project will be considered successful when The Library has a complete system in use on its opening day with no issues.

The main requirements for success are as follows:

- The application will allow users to log-in with their The Library account only. This will provide a single identity for each user that uses the application.
- The application will be stable.

- Completely user and member friendly which is easy to use without a complicated manual.
- Minimal clicks to get to the destination.
- Reporting options.

The intended audience of this application is users of The Library. As part of this system, we will have three separate applications, **Snatch and Go** which is the Self Checkout Application, **Rent a Leaf** which Librarians and Customers will use, and **AutomaticEmails** which will be a small console based application that will send reminder emails at 6am in the morning once deployed .

Librarians will use **Rent a Leaf** to add/edit/remove books and many different objects within the system, while customers will be able to use it to look up books and update personal details while allowing them to place holds, see current issues, etc.

The main audience for the **Snatch and Go** Self-Checkout Application are the customers; this will allow them to use a self-checkout system to issue books to themselves without a Librarian.

1.2 Project Scope

As mentioned in the Objectives; HSB is required to make an application catalogue for “The Library” which needs to be user friendly, stable and ready before the opening day.

The new application will make the process of loaning books, adding new books, advertising certain books and events easier for the new library.

As this can turn into a large project, with new features being requested regularly, we will be reasonably pushed for time, so this project will require plenty of focus from the team members. The project is due to be completed around early-to-mid June. Although, the sooner we are able to deploy the new application for the users, the better, as this will give the new user plenty of time to organise the catalogue and understand the new system.

One of the features we will be adding is a self-checkout system (Snatch and Go) for the books; allowing customers to issue books to themselves. This system will ease the traffic flow at the librarian’s desk so they will be available for questions, book requests and returning books.

Another feature that will be included, is to request to hold a book for pick up. We will arrange a function which will have the ability to automatically display an alert to the

Librarian when returning a book to see if it is held, and then to automatically email the customer that the book is available to pick up.

One of the noteworthy components we will be featuring within our application is for book and customer identities to be verified using Radio Frequency Identification. This means that the customer can tap their ID card to the reader to verify their identity and login, and this will allow them to also tap the book to the reader enabling them to issue books. RFIDs (or using Book Barcodes) can also be done by a Librarian when issuing and returning books.

1.3 Project Constraints

1.3.1 Potential Technical Skills Barriers

With all the features we would like to add, we will have to limit it to the skill of each member. If extra research and training is required, we will have to make sure we allow time for this, whilst also confirming if the new feature is worth the extra work.

1.3.2 Limited Time

As we have made a strict schedule for this project we have to be careful as to not stray from it. We have allowed time for personal restraints that may occur, eg illness, exams, training.

1.3.3 Not adding too many features

With our time restriction, we have to make sure that we do not add too many features and make it so we cannot complete the project within the timeframe. We will have to monitor this throughout the project.

2.0 Project Requirements

2.1 General Requirements

2.1.1 Application Requirements

- The application will be developed as a 64 Bit Windows .NET Framework Application
- The application will be written in C# using Microsoft Visual Studio 2019
- The applications will be form-based, allowing the user to navigate through the Main Menu opening up forms.

2.1.2 Authentication and Security Requirements

- The program shall only allow users to log in and out with their unique 'The Library' account.
- The program will recognise each book copy via a unique identity (Book_ID) or RFID Tag Identity.
- For testing and demonstration purposes a database hosted on a web server will be used. Once deployed there should be an option within the application to specify server, user, password and database parameters, with an attached SQL file that will allow the administrator to create the database with the necessary tables.
- The information must be stored safely and securely so as to prevent breaking the New Zealand Privacy Act (1993). This means the inclusion of secure login portals, and the employment of Information Security techniques such as one-way password encryption and preventative security measures.

2.1.3 Operating System & Server Requirements

- The server shall be a Raspbarian Linux Server on a Raspberry Pi 4B
- The server will be running a File Transfer (FTP) Server, allowing client devices to save a copy of book cover images onto the server, and allowing the potential expansion of a catalogue showing these book covers by opening the file on the server.
- The MYSQL Server (MariaDB) will also be running on this Linux server.
- The Rent a Leaf application will run on Windows 10 64bit computers.

- The Snatch and Go application will also run on Windows 10 64bit computers.
- The Automatic Emails application will be run on the Raspbian Server via a CronTab Job.

2.1.4 User Interface / Front-End

- There will be a clean interface allowing the user to easily access options through the main menu.
- All error messages displayed to users will be clear and will offer advice or instruction.
- The user interface of the application will allow the user to enter information via plain text into the application (not having to directly type in SQL Commands).
- The Windows front end of the program shall follow Microsoft Windows programming standards and conventions.
- Validation of the Data's Integrity will be performed on the client-side before sending it to the SQL Server.

2.1.5 Code Quality

- All code shall be checked by all members of the team, before being implemented for quality control.
- Unit testing will be performed for any objects that are created within this application.
- An object shall only exist within the application for the lifetime of its intended purpose.

2.2 Application-Specific Requirements

2.2.1 Rent a Leaf Application

2.2.1.1 Functional Requirements

- The ability for users to log in to the application and access specific features within the application. This will be performed by using native logins, credentials are stored in a database table that the application checks when the user clicks the login button. This means that there is only 1 MYSQL Data Manipulation User used, which the

application uses to login to the database and perform these SQL Manipulation functions.

- Showing and hiding features of the application depending on the users access-level.
- The ability to easily logout of the application regardless of which screen/function the user is in.
- The ability for customers to place holds and to renew books.
- Looking up book and customer entries in the database via RFID Tags (and assigning these on the add/edit screens for the relevant objects).

The ability to:

Object	Functions	User Access
Book	Add New/View All/Edit/Remove	**Librarians:** Add/View/Edit/Remove
		Customers: View
Book Copies	Add New/View All/Edit/Remove	**Librarians:** Add/View/Edit/Remove
		Customers: View (Part of Book View)
Publisher	Add New/View All/Edit/Remove	**Librarians:** Add/View/Edit/Remove
		Customers: View
Retailer	Add New/View All/Edit/Remove	**Librarians:** Add/View/Edit/Remove
		Customers: No Access
Genre	Add New/View All/Edit/Remove	**Librarians:** Add/View/Edit/Remove
		Customers: View
User	Add New/View All/Edit/Remove	**Librarians:** Add/View/Edit/Remove

		(Only Customer Entries)
		Customers: View & Edit (only their details)
		Admin: All of Librarian's Permissions + Adding & Removal of Librarians
Books Issued	Add New/View All/Remove *Please Note: This will only be used through an Interface, such as the Issuing or Returning Self Checkout System, or through the current Issued Books Form*.	**Customers:** Issuing Books / View Current Books Issued
		Librarians: Issuing / View Current Books Issued / Returning Books
Connection Settings	View/Edit	**Admin Only**
The Admin User Access will have all of the Librarians Permission, plus any other extra permissions specified		

2.2.1.2 Non-Functional Requirements:

- The Rent a Leaf application will need to be fast, retrieving results in less than 5 seconds, ideally under 1 second.
- The Rent a Leaf application will need to cope with changing data, by ensuring that information is continuously updated to the database.
- The Rent a Leaf application needs to be able to be run on more than 1 computer at a single point of time.
- The Rent a Leaf application needs to be able to run on a variety of 64 bit Windows 10 Computers due to *'The Library'* having a variety of Dell, HP and Microsoft Surface Computers.

- The RFID Readers will be Phidget RFID Readers to ensure that they are compatible with the drivers installed on the computer and our application.

- The Rent a Leaf application will be set by default to the English language as The Library is located in New Zealand.

- The Rent a Leaf application needs to be easy to use, with a minimalistic interface with clear navigation options.

2.2.2 Snatch and Go (Self Checkout) Application

2.2.2.1 Functional Requirements:

- The ability for Users to login in and issue books to themselves,
    - Check whether the book currently has a hold on it, if so **Do Not** Issue it.
    - Check whether the user is allowed to issue the book or not (are they blocked).
    - Check whether the user is allowed to issue the Book based on the Age Rating.
- The ability for users to see what books that have issued on the screen during the current session,
    - Show the name, author, issue date, and due date.
- Error checking has been performed,
    - Eg. What happens when the same book is presented on the screen?
    - E.g. What happens if a book hasn't been returned and then is issued.
- The ability for Users to logout and then have a new user Login,
    - The ability for a receipt to be sent to the user's email address.
    - Ability to not have a receipt sent to the user's email address.
- The ability to return books through an option in the application.

2.2.2.2 Non-Functional Requirements:

- The Snatch and Go application will need to be fast, retrieving results in less than 3 seconds, ideally under 1 second.

- The Snatch and Go application needs to be able to be deployed onto more than 1 computer at a single point of time.

- Because all of the computers running the Snatch and Go application will be a Dell XPS computer, the application will need to support this hardware (with a Phidget RFID Reader attached) with a touch screen User Interface (such a Touch Screen Display will be used to navigate).

- The RFID Readers will be Phidget RFID Readers to ensure that they are compatible with the drivers installed on the computer and our application.

- The Snatch and Go application will be set by default to the English language as The Library is located in New Zealand.

- The Snatch and Go application needs to be easy to use, with a minimalistic interface with clear navigation options. Most of the navigation within the application will be automatic (such as the user using their ID Card to log in, which then loads the Issue book screen automatically), however, there will be some User-Driven navigation within the application (such as log-off button when finished issuing books).

2.2.3 Project Changes

Since our original Requirements Document, we have had a couple of Project Non-Functional Requirement Changes, which have been discussed in this document. These are:

- Changing from a Windows Server to a Raspberry Pi Linux Server.

- Changing from a shared SMB Folder to an FTP Server running on the Server.

- Creating a third small application which sends out email notifications about books that are due back, or overdue.

2.3 User Interface (UI)

2.3.1 Clean User Interface:

The application shall be composed of a clean, general consistent user interface. We will be using the less is more approach as this application can add many features and we do not want to make it too difficult or complicated.

2.3.2 Easy to understand:

The application shall be easy to use; The Library will be providing detailed training to staff members, once installed.

Confirmation boxes will be displayed before a User removes an entry from the database (excluding returning books).

2.3.3 Constant Colour Scheming & Interface Design:

The application shall have a consistent colour scheme, and interface design which will flow between the pages.

2.3.4 Clean & Concise information:

All the information, contained in this application, shall be displayed in a clean and concise way. The information added will be informative, necessary and worthwhile.

2.3.5 Good Content Balance:

Good use of white space shall be considered and accomplished, balancing content on the forms, in a way that is comfortable, and pleasing for the users.

2.3.6 Front-End UI Code:

The front-end UI code shall be written, and developed, using C#.

2.3.7 Back-End Code:

The database shall be coded using Structured Query Language (SQL).

2.4 User Experience (UX)

2.4.1 Load Times:

The application should load in under 3 seconds for the average user, with an optimal load time of approximately 1 second. This means that assets used must be fully optimized, such as images and process efficiency.

2.4.2 Learning for New Users:

The application shall be designed in a comfortable way allowing the users to understand and use it, with no training necessary. The application will be rather straightforward; this means that when developing the UI, and UX for the application, extra care will be taken to ensure the clearness and cleanness of the application.

2.5 Security & Privacy Protection Requirements

2.5.1 SQL Database Security

Due to how this C# Application will be connected to an SQL Database, the database will be set up with the main database user**, Username:** Library, **Password:** Library. Within the database, there will be a users table which will contain the username and passwords of the librarian and customer login details, which will allow us to ensure that the users only have access to specific options by filtering the options on the application based on their user level. This database will be connected through the internet to the application via a domain

2.5.2 User Security & Privacy Protection

Delicate personal information handled by this application, such as password information, must be securely stored. Passwords to be stored in the database will, therefore, be encrypted with a one-way hashing algorithm such as **PBKDF 2** or **SHA-256**. The one-way nature of these hashing algorithms means that the encryption is extremely difficult for potential hackers/attackers to decipher, and will ensure the original plain-text password is thoroughly obfuscated. To provide an additional layer of security, the password will be encrypted alongside a **semi-randomly generated Salt** of an undefined length. This will add further variety to the encryption process, and ensure the final hash is unique to that of the one generated by the hashing algorithm by default. This has the advantage of eliminating potential Collision and Dictionary attacks, forcing potential attackers to rely on social engineering, personality, and brute force attacks.

User passwords will not be processed as plain-text by the application. Instead, any necessary password processing such as user logins will be dealt with by comparing two Hashes generated with the same Salt. The salt will only be extracted from the stored hash for the instance of processing and will be discarded as soon as processing is completed. This is to further ensure that potential attackers cannot use the extracted salt to facilitate the use of Hash Collision techniques to gain unauthorized access to personal information.

2.5.3 SQL Data Injection

Within the application, we will be using the **MySQL.Data** C# External Library available via NuGet Application Manager in Visual Studio, using functions and Objects of this DLL library to perform SQL Data statements (SELECT, INSERT, REMOVE). To prevent SQL Data Injection which could be done by users entering SQL Code inside our Application we will be using the insert Parameters Values feature of the SQL Execute Reader Command which then stores these values as String, preventing SQL Injection.

2.6 Business Rules

- If a book is overdue by 14 days the user will be blocked from getting future books and will get an automatic email sent with the purchase cost of the book to be paid to the library.
- Books are issued for a period of 28 Days.
- Books can only be renewed once, and only if there are not any current holds on that Book. The due date for a renewed book is for another 28 days.
- If books are not returned within the allotted days the user will receive a fine.
- Only one person per user ID card.
- Only 1 Book per Book RFID Tag
- The maximum number of books a person can have out at one period of time is 20 Books, and 5 holds.
- A Book can only have 1 Publisher or Genre.
- A Book Copy can only have 1 Retailer
- There can only be 1 Record for a single Customer (not multiple)
- There can only be 1 Record for a single Book (not multiple)
- There can only be 1 Record for a single Retailer (not multiple)
- There can only be 1 Record for a single Publisher (not multiple)

3.0 Data Design

3.1 Internal software data structure

For our main application (Snatch and Go) we have got several different forms which use internal software data structure to add/alter/delete entries from our MySQL Database.

Most Database Tables (Books, BookCopies, Publishers, Retailers, Genres, Customers, Users) have supplementary C# Objects that are used to interact with, and retrieve information from the database. This information, is stored locally in C# Objects, allowing client side validation before they are sent to the MySQL Database which does Server Side Validation and updates/adds/removes the entries into the tables.

To facilitate this most of these objects have their own creation form which is repurposed to become a View Form within the application (preventing duplication of work and code integrity by reusing code). When the user goes to view the objects there is a checkbox which will enable and disable the editing feature of that object. For User Levels which do not have permission to add/edit/delete entries these features are disabled.

To ensure that there is no duplication of entries we have created an Add Customers form which is used to insert entries into the Customers and Users Table. By using this one form to insert SQL Commands into the two database tables this ensures that it is easy for users to use, while also ensuring that duplicate entries are not easily able to be performed. We do have an Add Users form which allows manual insertion of values into the Users Table.

Within our application we have reused the one main form which features a ListView. This ListView shows the entries within a table, allowing the user to filter the results by a chosen search term.

All of these creation forms accept entries through a range of Visual Components including Text Boxes and Rich Text Formatting Boxes. This allows the user to enter in custom content (such as a Book Title), limiting the length of the input string to the size set of the attribute in the table. These values are stored as strings in the application which are then used to create a C# object (such as Book), performing Client side Validation. If the object is successfully created then the object's properties are inserted as parameters in a SQL Command, allowing the entry to be inserted or altered in the appropriate table.

Radio Groups and Combo Boxes (drop down boxes) are featured within our application that restrict the users selection to a small selection of custom options (such as selecting an Access Level on the Users form). By using Combo Boxes we have created a method which for several of these (such as the Select Genre on the Add Book Form) selects the entries from the Genre Table and stores the Genre_ID as the selected value (e.g. 1), while the displayed field is the Name of the entry (e.g. Action). These values are stored as an Integer,

representing the ID of the appropriate entry's relationship within the database. This value is then used in the creation of an appropriate C# object performing client side validation. If the object is successfully created then the object's properties are inserted as parameters in a SQL Command, allowing the entry to be inserted or altered in the appropriate table.

3.2 Global data structure

Within our applications we have several different classes and properties which are transferred between the different forms with our application.

Our Applications contain a Settings File which stores most of these global data files. These include the string values of the connection string - IP Address/DNS Name, Database, Username and Password for the MySQL Database. By storing these in the Settings file this allows these connection parameters to be consistent and easily modified if the Database connection parameters need to be changed. Within the **Snatch and Go** and **Rent a Leaf** applications we have created a form which allows the user and admin to input these details if the application is unable to connect to the database. The Settings File also contains the connection details for the SMTP Server, Username & Password as well as the Twilio Username Parameters. These parameters are all stored as strings, allowing the user to modify them in a **Database Settings** form.

The Settings File also contains the UserID and AccessLevel of the current customer logged in (this is retrieved from the database when the user first logs into the application)**.** These settings are saved and used within the application to ensure forms that can be accessed by the users are relevant to their user level, and the UserID is used to ensure that the information retrieved is relevant to the user or customer (the Customer ID is retrieved by searching with the User ID).

We also have a Global Static Class called Connection, which only contains one property which is a MySQLConnection. When the application is first launched it creates a new MySQL Connection and tries to connect to the MySQL Database, using the **IPAddress**, **Database**, **Username** and **Password** values stored within the Settings file. If it successfully connects to the Database, it stores this MySqlConnection within the Connection Class, allowing the Connection to remain open and used within the many different forms inside the application.

3.3 Temporary data structure

Automatic Emails Application

Within our AutomaticEmails Console based Application we have a single Class called Customer Details. Customer Details is a temporary data structure which contains the combined information of the customers name (with the first and last name joined), using only the information from the database which is needed for the application's functions.

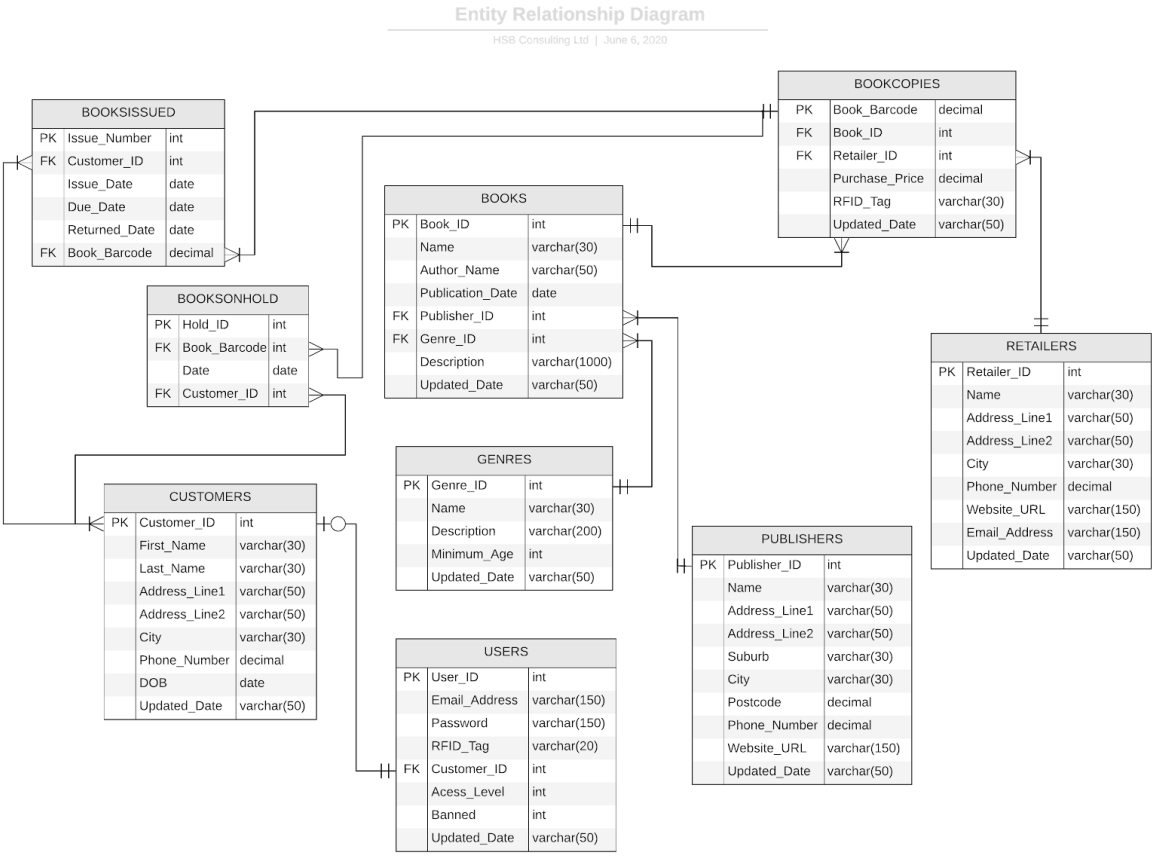
Once it is finished using the CustomerDetails object, it is destroyed and a new one is created for the next customer.

Snatch and Go

Within this application we have created a BookDetails Class, which temporarily stores the data structure of the information of a book that has been checked in/out. This BookDetails object stores the information such as the Book_ID, Book Name, Author, Book Barcode, RFID_Tag, Age (recommended Age retrieved from the Genres Table) and the DueDate.

We have also created a CustomerDetails Class which temporarily stores the data structure of the information of the current customer logged into the Self Checkout Application. This class stores the details of the Customer_ID, Name, Age, Banned and their Email Address. When performing functions such as Issuing new Books it uses the stored Customer ID to insert into the BOOKISSUED Table. If a user chooses to be sent a receipt of their books, the Application uses the stored Email Address in the CustomerDetails Class to send this receipt to the user.

3.4 Database description



This is an Entity Relationship Diagram showing the structure of our MYSQL database which is hosted on a Raspberry Pi. Within this diagram, we have decided to use several tables to ensure that the entities have the correct attributes.

The Retailers table stores information about the retailers which *The Library* purchases books from. Each entity has a unique Retailer_ID value which is the primary key.

The Publishers table stores information about the different publishers of the books. Each entity within this table has a unique Retailer_ID value which is the primary key.

The Genres table stores information about the different genres of each book in the library. Each entity within this table has a unique Genre_ID value which is the primary key. There is also a Minimum Age Attribute within this table, which specifies the minimum age a customer must be in order to be able to issue a book with this genre.

The Books table stores information about each Book in the Library. This table stores the master information about a Book, and not about each copy of a Book. This ensures data integrity, while also ensuring that the data is normalised. Each entity within this table has a unique Book_ID value which is the primary key. The Books table has several foreign

relationships with other tables, which include a one to many (1:M) relationship with the Publishers Table (referring to the publisher of the Book), and a one to many (1:M) relationship with the Genres Table (referring to the necessary genre & age limit of the Book).

The BookCopies table stores information about the different Book Copies that the Library has. This table contains the details of the different copies of a book, such as the Book Barcode as well as the retailer's details. Each entity within this table has a unique Book_Barcode value which is the primary key. This table includes several foreign relationships with other tables, including a one to many (1:M) relationship with the Book Table (to identify which copy refers to which book), as well as a one to many (1:M) relationship with the Retailers Table (which refers to the retailer who sold that book copy to *The Library*). We chose to put the foreign relationship with the retailers table within the BookCopies table and not the Books table due to how different copies of a book can come from different retailers.

The Books Issued table contains information about books that currently are/or have been previously issued. Books which are currently issued and have not been returned, have a value of NULL on the Returned_Date, allowing them to be distinguished by books which have been returned in the past (these entries have a value of the date the book was returned). The BookIssued table also contains the information of which customer issued what book, through a forigen relationship with the Customers Table. Each record within this table has a unique Issue_Number value which is the primary key. This table has several foreign relationships which is a one to many (1:M) relationship with the BookCopies Table (to identify which Book Copy was issued) and a one to many (1:M) relationship with the Customers Table (to identify who issued what book).

There is a Customers table which contains the personal details of the customer, such as their name and email address. Each record within this table has a unique Customer_ID value which is the primary key.

The Users table contains information about the login users, such as their username (which is the email address) and password. The access level within this entity sets the permissions within the application, ensuring that functions and forms are hidden for users who do not have permission to access them. This table has a foreign one to one (1:1) relationship with the Customer's table in order to associate and allow customers to login and perform actions. Each record within this table has a unique User_ID value which is the primary key.

The BooksOnHold Table contains information about the current holds that have been put on books which are currently not available to issue. This allows customers to be notified when books they have put a hold on are available to collect. Each record within this table has a unique Hold_ID value which is the primary key. This table contains a foreign relationship of a one to many (1:M) relationship with the Customers Table (to identify the customer who

placed the hold), as well as a one to many (1:M) relationship with the BookCopies table (to identify which copy of the book the customer has place a hold one)

4.0 Architectural and component-level design

4.1 Program Structure

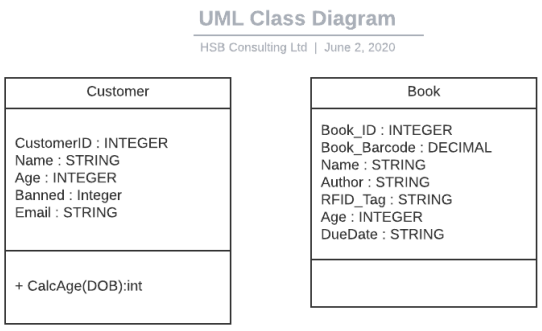
Our System incorporates three C# Applications (2 Form Based, and 1 Console Based) as well as a MySQL Database.

One of the Form Based Applications we have developed for our system is our *Rent a Leaf* Application which is the main application within this system. This application allows the user (Librarian or Customer depending on the user level) to access features and acts as a Front-End User Interface to our MySQL Database. Book Copies can be identified either via a Book Barcode or RFID Tag, while also allowing them to lookup and perform functions within the database. Customers can see their current and previous book issues, while also allowing them to place holds (and see their current holds).

Another Form Based Application we have developed for our system is our *Snatch and Go* Application. This application has been created as a user driven application, allowing customers to issue and return books. We have tried to optimise the application for Touch Screen Devices, allowing the user to interact with the application through a touch screen interface. Recognition of Customer Identities and Book Copies are performed through Radio Frequency Identification Cards that are embedded in the Books and Customer ID Cards.

We have also developed a C# Console Based Application which is a simple application that reads information from the MySQL Database and uses this information to send out reminder emails and SMS notifications. This application is run automatically at 6am on our Linux Server (Raspberry Pi) through a Crontab Job. There is no user interaction with this application, however there is the ability to alter the Configuration Settings (such as the SQL Connection String) through a Settings File.

4.1.1 Class Diagram (Snatch and Go)

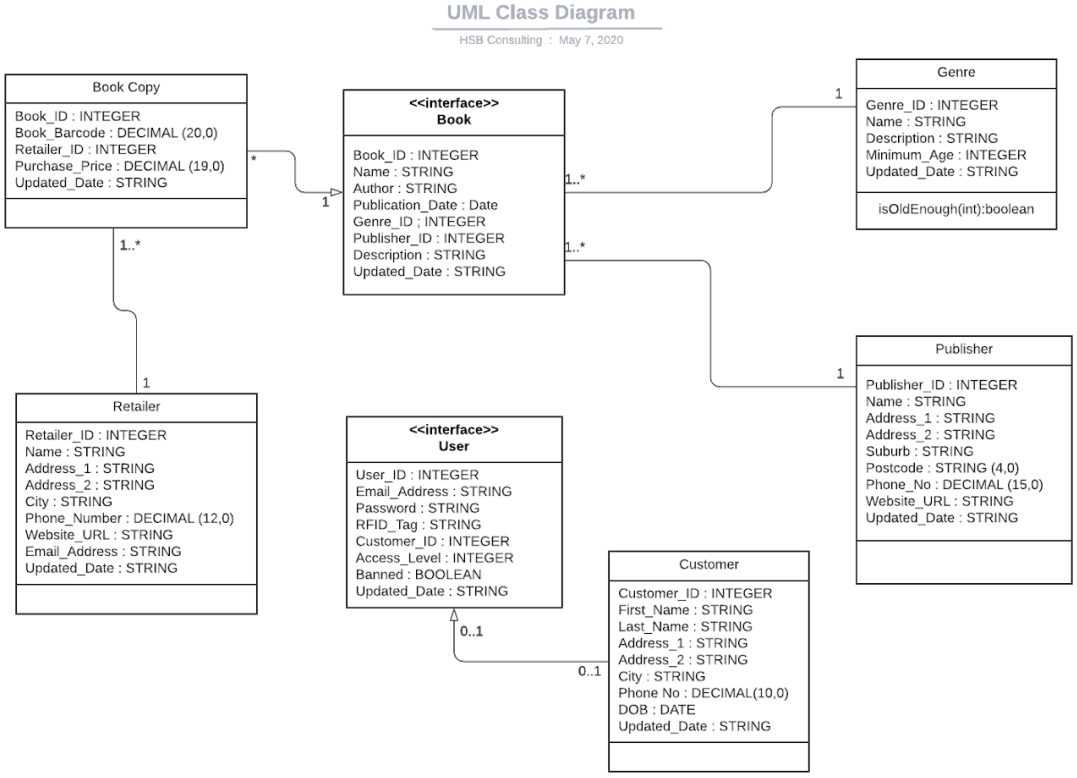


This UML Class Diagram shows our Temporary Class of Customer and Book which are used within our *Snatch and Go* Application.

The Customer class stores the information from the database of the CustomerID, Name, Age, Banned and their Email Application which are used in our Application to Issue Books. There is also an internal CalcAge method within this customers class which calculates the current age of the customer based on the current DateTime Value, to ensure that the Book they are issuing is age appropriate for them.

The Book Class stores the details of the Book Copy (and the Book that relates to that Book Copy), such as the Book_ID, Book_Barcode as well as other information including the RFID_Tag (which is used to retrieve the Book Details from the Database). The Name and Author of the Book is retrieved from the Database; which is done to provide a visual aid to the customer (or staff member) when issuing or returning the book. When issuing a book the user can choose to receive a receipt of the books they have issued. When performing this feature the Customer and Book classes are used to personalise and create the message that is sent.

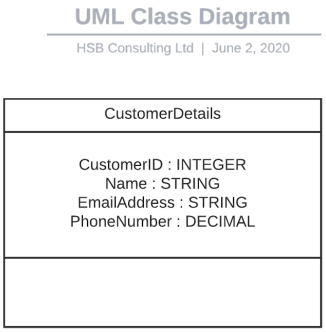
4.1.2 Class Diagram (Rent a Leaf)



This UML Class Diagram shows how our Objects within our C# Application will be created based on the Entity Relationship Diagram.

All Objects on the Class Diagram are created from tables in the MYSQL Database, allowing us to perform client-side validation within our application before sending it to the MYSQL Database to be validated through server-side validation. The objects only exist for the lifetime of the intended purpose of their use, once they are no longer needed they are destroyed.

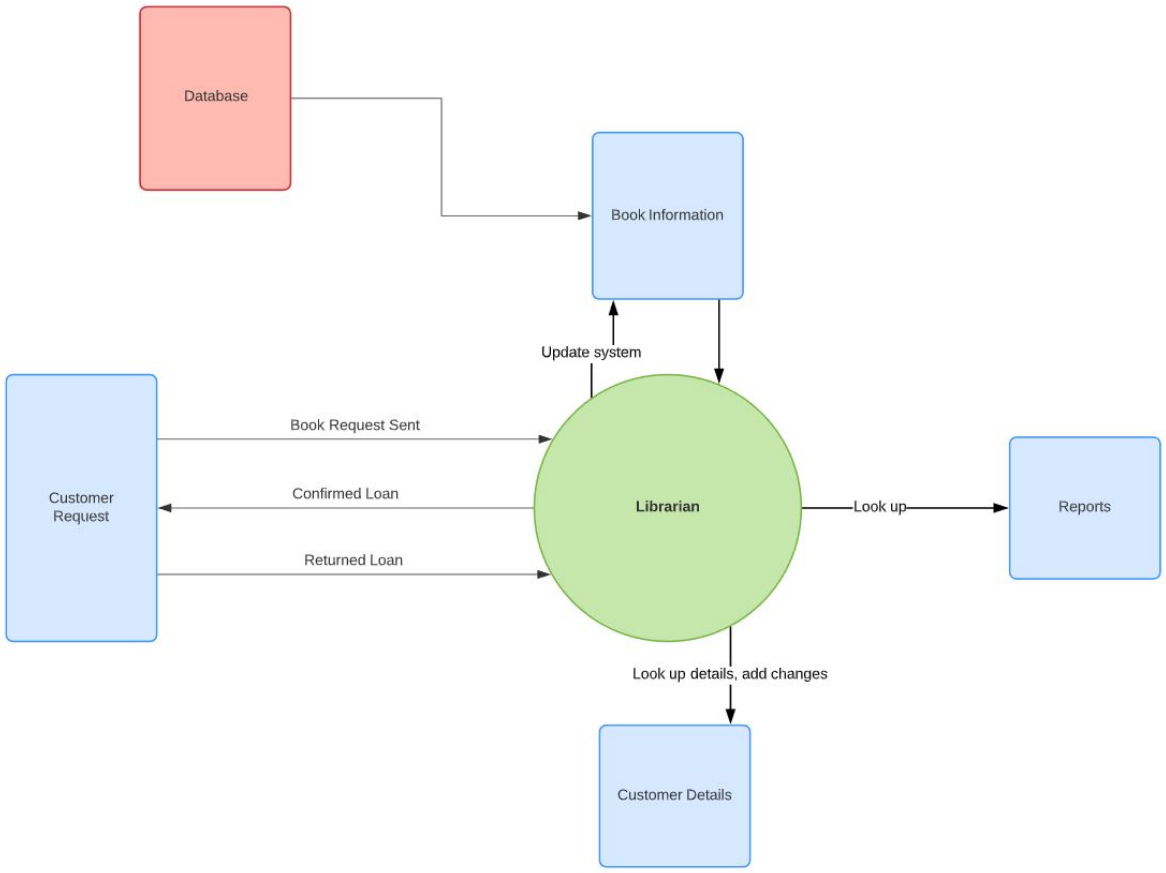
4.1.3 Class Diagram (AutomaticEmails)



This UML Class Diagram shows our Temporary Class of CustomerDetails which is used within our AutomaticEmails Application.

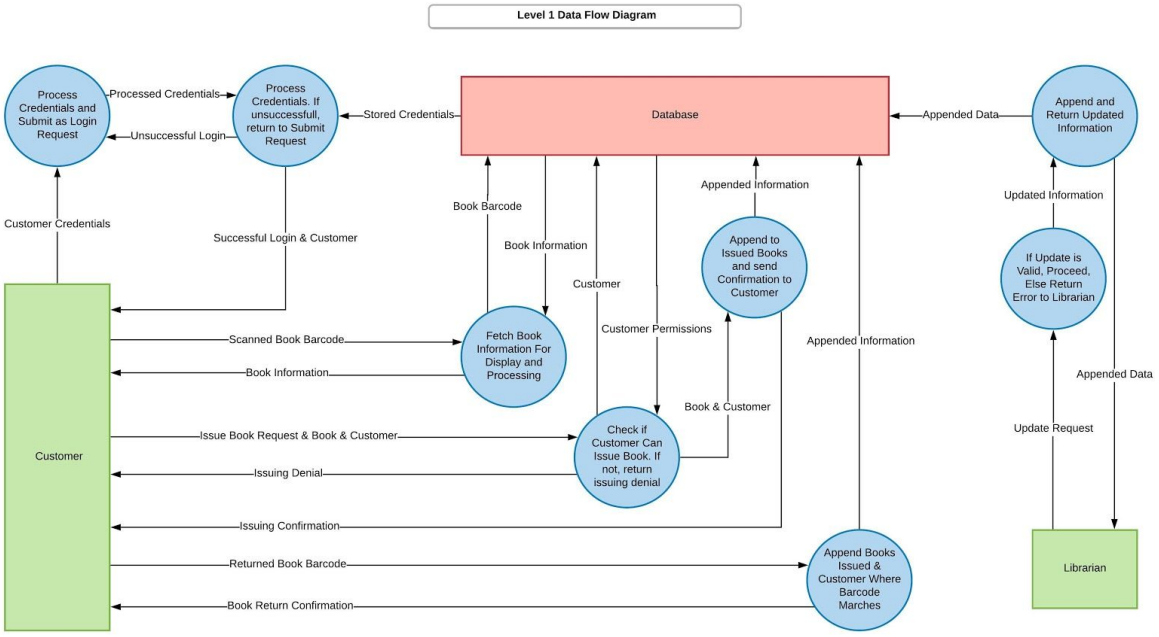
The CustomerDetails class stores the information from the database of the CustomerID, Name, Email Address and their Phone Number which are used in our Application to send out Reminders and other useful emails automatically via a CronJob on the Linux server.

4.2 Data Flow Diagrams



4.2.1 Process Description 1

This is a Level 0 Logical Data Flow Diagram, outlining the basic data interactions within the library system. The customer sends a book issuing request that goes to the library system containing information about the book and customer. The system pulls any additional information regarding the book and customer for processing the loan. If the loan is permitted, the database is updated with new user and book information (book is deemed “issued”, and the customer is assigned the book).



4.2.2 Process Description 2

This is a Level 1 Data Flow Diagram describing the same issuing process above in greater detail. The customer first logs into the Library system, submitting their credential information. This is then processed, and if this is not successful, the customer must redo the login process before continuing. If login is successful, the user credentials are sent to the Database so as to retrieve their user access rights, and a successful login confirmation is sent to the customer.

The customer then scans or enters in a book that they want to issue. This is used to retrieve book information from the Database for the user. If this is the book the user wants to issue, they can then issue this book, and the information and issue request is processed. If they have permission to do so, the database is updated and the loan is confirmed. If not, the customer receives a loan denial. When the customer goes to return the book, the book is scanned into the system and the database is updated again, and the customer is returned a return confirmation.

The Librarian can also manually enter information into the system. If the changes the Librarian is trying to make are valid, the altered information is appended to the Database. Once this has been successfully appended, the information is then returned to the Librarian to confirm the changes have been made.

4.2 Description of Component Snatch and Go

4.2.1 Component Snatch and Go Details

One of the applications within our system is the **Snatch and Go** application. This application is a small Self-Checkout/Self-Return application which easily allows the user to issue and return books.

Features (Issuing Books):

- Ability to inform the user when logging in if they are banned, which occurs when they have overdue books for more than 14 days.
- Ability to look up and count how many books the user has currently out on issue. This then shows and limits the customer to the limit specified in the settings file (by default 20 books).
- Ability to check whether the book placed on the RFID Reader has a hold on it (meaning it can't be issued).
- Ability to Issue Books to be due back at a set specific time preset in the settings file (by default 28 days).
- Ability to send a Receipt to the user's email address (stored in the database) when they logout, listing the books they issued during the session and the return date for them.
- Ability to then issue books to a new user credential without having to restart the application.

Features (Returning Books):

- Ability to return books by looking up the book in the database and altering its Returned_Date entry from NULL to the current Date.
- Ability to check whether the book has a hold on it, and if so email the Library (notifying the librarians so they are alerted that there is a hold on it), and informing the user that the book is available to pick up.
- Showing which books have been returned in the *session* of returning
- Ability to return back to the default mode of issuing books

4.2.2 Processing of Information in Snatch and Go

Within the Snatch and Go Application, on first load up it creates and establishes a Connection to our MySQL database which contains the tables and information relating to *The Library*.

Issuing Books

Once the Connection is established with the MySQL Database the Login Form for the Self Checkout System is shown, asking for the user to present their ID Card which contains an RFID Tag. When the User scans their Card on the RFID Scanner built into the Self Checkout Machine's Visual Interface, the application reads the unique ID value of the Customers ID Card. If a value is not found in the database which matches the Tag, an Error Message is shown. Otherwise the user is then logged into the Self-Checkout Application and can issue a book.

A book may be issued by the Customer by placing it on the RFID Scanner, as the Book has an embedded RFID Tag within it. Once placed on the sensor and detected, the scanner reads the unique ID value of the RFID Tag (which like the customer's ID Card, is unique). With this value it reads the database to retrieve the Book and Book Copies value which matches the RFID_Tag value stored as an entry in the BookCopies Table, combining the values needed as shown in the Class Diagram to create the Book Class. The Book Barcode as well as the Issue & Due Date are inserted into the Database's BooksIssued Table. The Due Date is automatically calculated by adding the specified number of days from the current date (reading this specified number of days from the Settings File). The Listview that shows the current books issued since the user logged in adds a new List View Item, showing the Book Name, Author and Due Date with the remaining number of books that can be issued below the List View Button.

If the same book copy is placed on the RFID Reader while logged in during this session, the application will inform the user that they have already issued it out.

The user can then choose to Logout with a Receipt, or just Logout. If the user selects to logout with a receipt an email will be automatically sent to the customers email address with a list of books that were issued (using the customers email address stored in the Customer object). Otherwise the user is just logged out and a new user can present their ID to issue Books. When the user logs out of the application, the book and customer objects are destroyed and new objects are created when the next user logs in.

Returning Books

If the user selects to return books (clicking the Button at the Bottom Left Corner) the Self-Checkout Application prompts them to place a book on the RFID Scanner. Once the

book is placed on the scanner, the application reads the RFID Unique ID Value and looks up the BooksIssued Table for an entry where the Return Date is NULL (meaning the book is still currently issued). It alters the entry, changing the Return Date to the Systems Current Date. Once this has been performed, it displays a message informing the user that it has been returned. The application also checks at the same time whether a book has a hold on it, sending an email to the librarians (and to the customer) while also displaying a message within the application.

Throughout Snatch and Go most of the Application is user driven in a linear flow via the placement of the RFID Tag on the scanner, or the users interaction via buttons (using the Self-Checkout Devices' Touch Screen). The user does not need to enter any values in themselves, as the values are all retrieved from and inserted or altered into the database via SQL Commands in the background.

4.2.3 Component Rent a Leaf Details

Another application we have developed within our system is the Rent a Leaf application. This application is a complicated application that is used by Librarians and Customers. Customers have more restricted access to the features within this application compared with the Librarians and Admins. The application just like the Snatch and Go Application connects to our MySQL Database, and performs SQL Commands.

Features (Issuing Books):

- Ability for users to login to the application, hiding and showing features of the application depending on the user level.
- Ability to Add/Edit/Search/Remove Users
- Ability to Add/Edit/Search/Remove Customers
- Ability to Add/Edit/Search/Remove Books
- Ability to Add/Edit/Search/Remove BookCopies
- Ability to Add/Edit/Search/Remove Genres
- Ability to Add/Edit/Search/Remove Retailers
- Ability to Add/Edit/Search/Remove Publishers
- Ability to lookup (search) Books and place a hold on them
- Ability to see current and previous books issued.

- Ability to see current reserves
- Ability to generate and print out a range of Reports.
- Ability to logout, and have a new user login giving them their necessary permissions and details.
- Ability to look up Users via their ID Card (using RFID Technology)
- Ability to look up Book Copies via RFID Technology or using a Barcode Scanner.
- Sending out Welcome Emails to new Customers/Users once they are created.
- Hashing Passwords to ensure they are stored securely.

4.2.4 Processing of Information in Rent a Leaf

Within the Rent a Leaf Application, on first load up it creates and establishes a Connection to our MySQL database which contains the tables and information relating to *The Library*. Any information that is inputted within our application is validated, and then inserted into the Database, while performing server-side validation.

There are options within our application that allows the user to edit and alter existing entries, such as renaming books or changing the email address of a Customer. These are all performed through a Graphical User Interface that we have designed, while performing SQL Commands such as INSERT or ALTER commands in the background.

There are also options within our application to remove entries, such as books and book copies. The User is warned before deleting an entry, and if they select to continue and delete the entry a message box appears informing the user it has been deleted. The ListView is refreshed, showing the up to date entries within the database. In the background the deleting of this entry is performed by using a DELETE sql statement, which uses the ID of the selected entry within the WHERE clause (retrieving the ID from the selected Object's Tag).

4.2.5 Component Automatic Emails Details

To facilitate the feature of sending reminder and overdue emails we have created an AutomaticEmails Application. This is a C# Console Application which once deployed will be launched via a CronTab Job on our Linux Server (the Raspberry Pi) at 6am each day of the week.

This application performs several different functions.

One of the features it does is that it sends a reminder notification, reminding customers of any books which are due back in 7 days time. An email is sent out, with information about which books are due back (an email can contain information of more than 1 book), as well as a polite message informing them that the books are due back in 7 days as seen below.

HI Ben King
The Following Books are due back in 7 days time:

Harry Potter & the Half Blood Prince by J.K. Rowling
Issued Date: 1/05/2020 12:00:00 am
Due Date: 16/06/2020 12:00:00 am
Barcode: 9780123456786

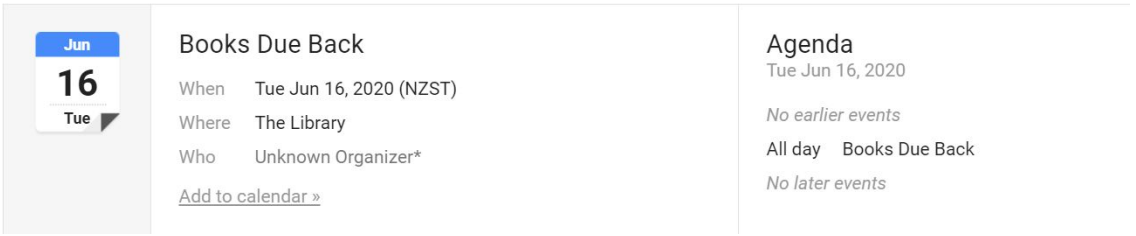
FEAR by Micheal Grant
Issued Date: 1/05/2020 12:00:00 am
Due Date: 16/06/2020 12:00:00 am
Barcode: 9780123456799

If you are unable to get these books back in time, Please login to your account and see if you can renew them. Otherwise get in touch with us

Thanks,

The Library

Attached with the email is a reminder iCal event file which can be added to the customers calendar if they want to have a calendar reminder. The description of the calendar event is a copy of the email message, listing which books are due back. The iCal's generated event date is the date that the book is due back (an all day event), and will send reminders to the user via the calendar app depending on their calendar settings.



By default if the Customer's email address is a Gmail Account being accessed through the Gmail Website the notification will be added to their calendar (by default, nothing we have caused).

The application also sends out Overdue Emails to customers if their book is overdue, and needs to be returned. Just like our Reminder Emails the overdue email lists all the books that are overdue on the day (with the overdue email containing information about one or more overdue books), as well as a polite message informing them that the books are now overdue and will need to be returned.

HI Ben King
The Following Books are now Overdue and will incure a fine of 20 cents per item, per day:

FEAR by Micheal Grant
Issued Date: 1/05/2020 12:00:00 am
Due Date: 8/06/2020 12:00:00 am
Barcode: 9780123456799

Please ensure you get these back promptly to ensure that you are not charged a Lost Fee. Any issues please get in touch with us

Thanks,

The Library

--DO NOT REPLY

The application also sends out *blocked* Emails to customers if they are blocked by the system. This happens when a user is blocked by having books overdue for more than 14 days. To be unblocked the customer needs to get in touch with the Library and follow their instructions in order to reactivate their account and to be allowed to issue books again.

HI Ben King
Unfortuatly due to how the following books have been overdue for 14 days, you are now blocked from issuing any new books.

This block will remain in place on your account until you return all overdue books and talk to the Libarians in person:

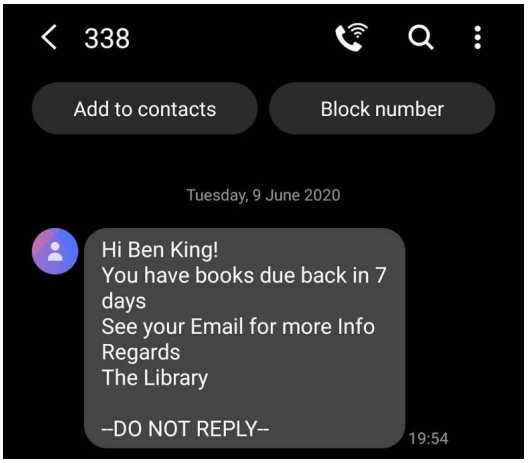
Harry Potter & the Half Blood Prince by J.K. Rowling
Issued Date: 1/05/2020 12:00:00 am
Due Date: 26/05/2020 12:00:00 am
Barcode: 9780123456786

Please return these books as soon as possible to ensure that you are not charged a Lost Book Fee. Any issues please get in touch with us

Thanks,

The Library

Whenever one of the above reminder emails are sent out we have also implemented SMS messaging, which sends an SMS message to the phone account on record telling them that they have **Books Due Back**, **Overdue Books**, or **that they are Blocked** depending on the method and customers current account status. SMS messages are automatically sent using Twilio, an external Provider which provides virtual numbers to developers. While for this development stage we have used a trial account for this application to be deployed. *The Library* would need to create a professional account, costing 10 cents per text sent out.



To allow this application to be adapted for configuration changes we have stored the following configuration values in a settings file which can be altered in Visual Studio or a Text File editor.

Configuration Settings

- IPAddress - the IP Address or Server Domain Name of our MySQL Database.
- Database - the database in the MySql Server which contains our Library Database.
- Username - the MySQL Username used to access the MySQL Database.
- Password - the MySQL Username's Password which is used to access the MYSQL Database.
- VirtualNo - the Twilio Virtual Cell Phone Number used to send SMS Messages
- CellUsername - the Twilio API Username used to access and send SMS's from the Virtual Number
- CellPassword - the Twilio API Username's Password used to access and send SMS's from the virtual Number.
- ReminderPeriod - the number of days before a book is due back that the reminder email is sent. The default for this is 7.
- BlockedPeriod - the number of days after a book is overdue which the user is blocked, sending out a message informing them and marking them as blocked in the database. The default for this is 14.
- SMTPServer - the SMTP (mail) server for our Email Address
- SMTPPort - the SMTP Port for our Email Address's server
- SMTPEmail - the Email Address that emails will be sent from
- SMTPPassword - the password for the email address that emails will be sent from

For the development of this application we have configured the application so that it sends out SMS messages from our Twilio Trial Account, and a custom email address of **library@benking.co.nz**

4.3 Object interaction details

4.3.1 Snatch and Go

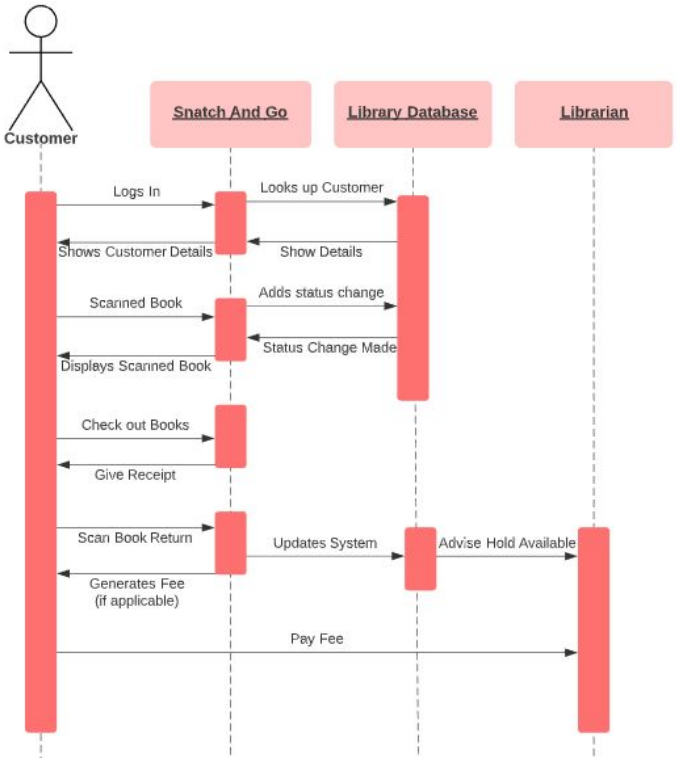
The below diagram shows the interaction between the customer and the Self Checkout System (Snatch and Go). Snatch and Go has a few features that it cannot do automatically and the Librarian is required for this. However having a self-checkout is intended to free up the librarian's time and improve efficiency.

A Customer using Snatch and Go:

When they log in, the Snatch and Go screen will show their details. They will then scan the books they have (limited to 20) and all the scanned books will appear on the screen. Once they have scanned all the books they will check out all the books and the Snatch and Go system will ask if they want a receipt. If the user selects they do, they will be sent it automatically and logged out. Snatch and Go can also be used to return books.

Snatch and Go to the Database: Snatch and Go is connected to the library database, the same database connected to the Librarians (Rent a Leaf) System. Any changes made with Snatch and Go will connect to the Library database and will update automatically for Rent a Leaf.

Returned Books: Once returned, the Library Database automatically updates and then it will notify the Librarians if a book on hold has become available.

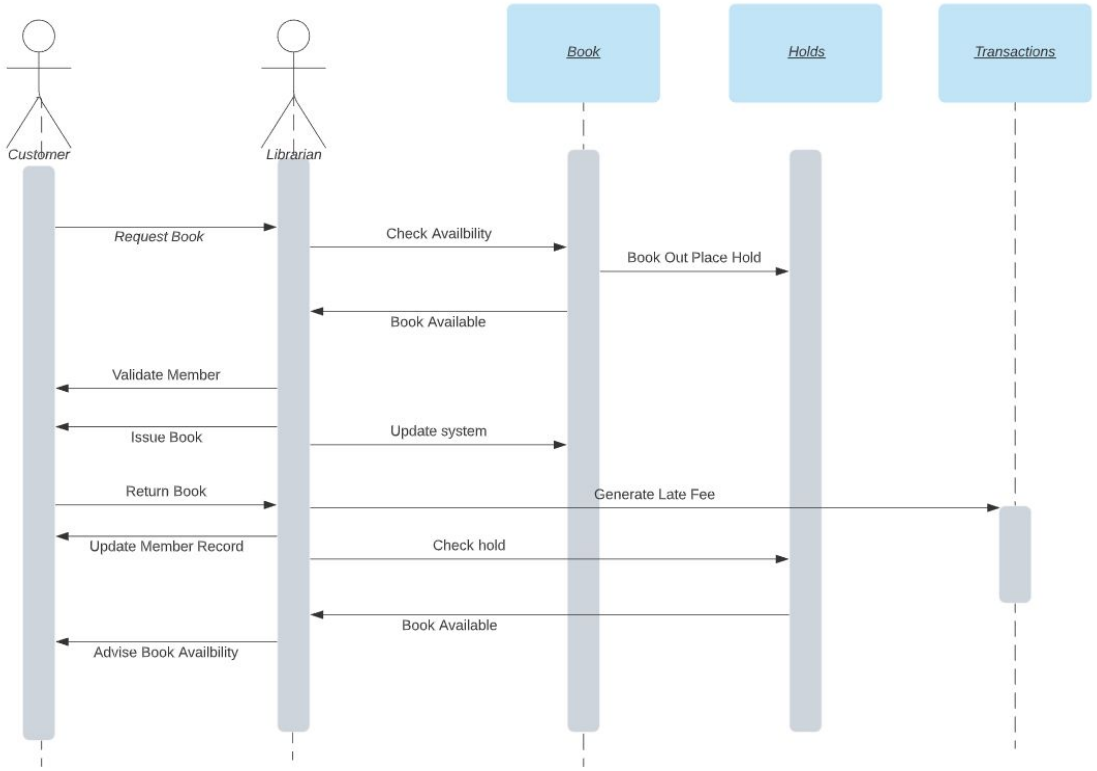


4.3.2 Rent a Leaf

Rent a Leaf involves Interactions between the customer and the Librarian. The below diagram displays how the customer interacts with the Librarian while the Librarian is using the Library System (Rent a Leaf).

Renting a book: Customers can request a book which the Librarian will look up in their system to see if it is available. If not they can place a hold for when the book is returned. If the book is available the customer can check it out once their membership login has been checked.

Book Return: When a book is returned the system is updated to show the return and available for the next customer however, the system will also check for book holds. If a book has a hold request against it, this will notify the librarian and they will notify the customer with the request. Another part of the returned book is the late fees which occur when a book is late or damaged. This fee will go against the customer who returned it and be noted on their membership details. With the requirements for this project *The Library* is using an external service to calculate and record fees for customers, allowing customers to pay for these fees through a Web Interface. Due to this we have not implemented a Fee System within our MySQL Table, or our application.



4.4 System behaviour

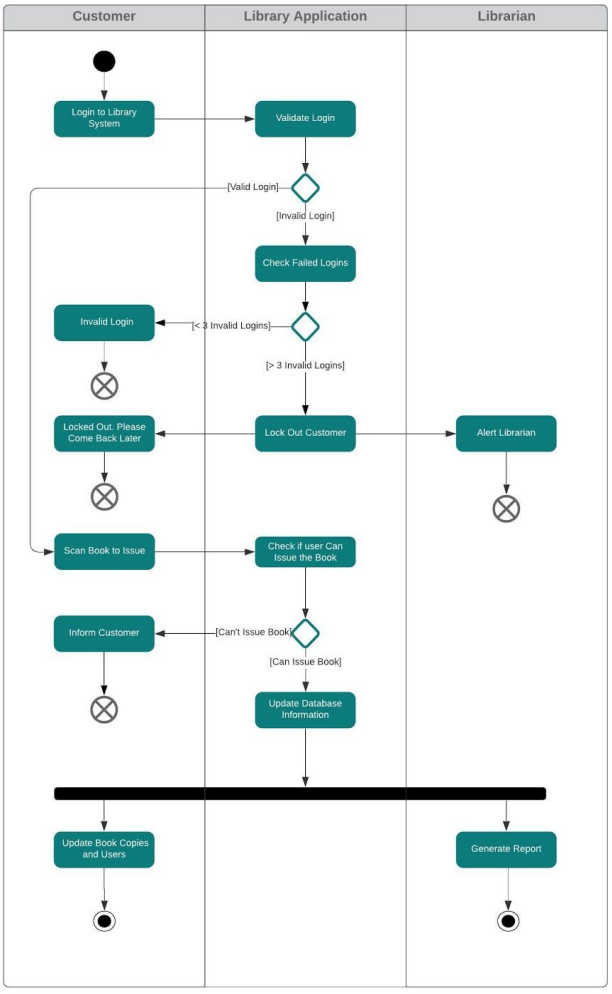
4.4.1 Activity Diagram

4.4.1.1 Activity Diagram - Issuing Book

Below we have our Book Issuing Activity Diagram including our secure login.

Login: A customer cannot log in without having the correct credentials. If the customer logs in incorrectly they will be locked out. If you are locked out the Librarians will be notified. You cannot rent a book without logging in.

Scan Book: Once logged in you can scan books for loan. If there are any issues the Librarian will notify the customer eg book limit or overdue fines. At the end of the transaction, the database will be updated with changes. This, in turn, will update the reports.

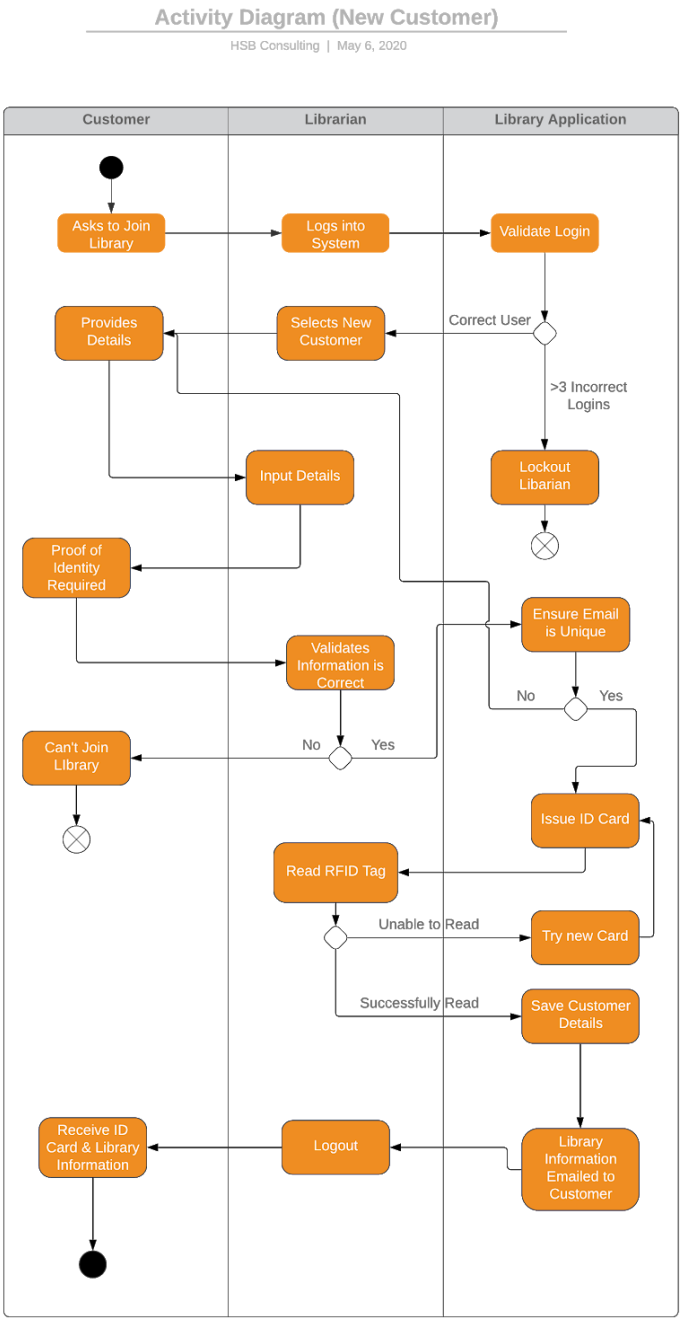


4.4.1.2 Activity Diagram - New Customer

Below is an activity diagram on how to add a new customer.

The Librarian has to log in before they can do anything within the system. Once logged in, the librarian will have a step-by-step process to follow to add a new customer.

Every customer will have their own ID card with their own login. It is essential this process is done correctly for security purposes.

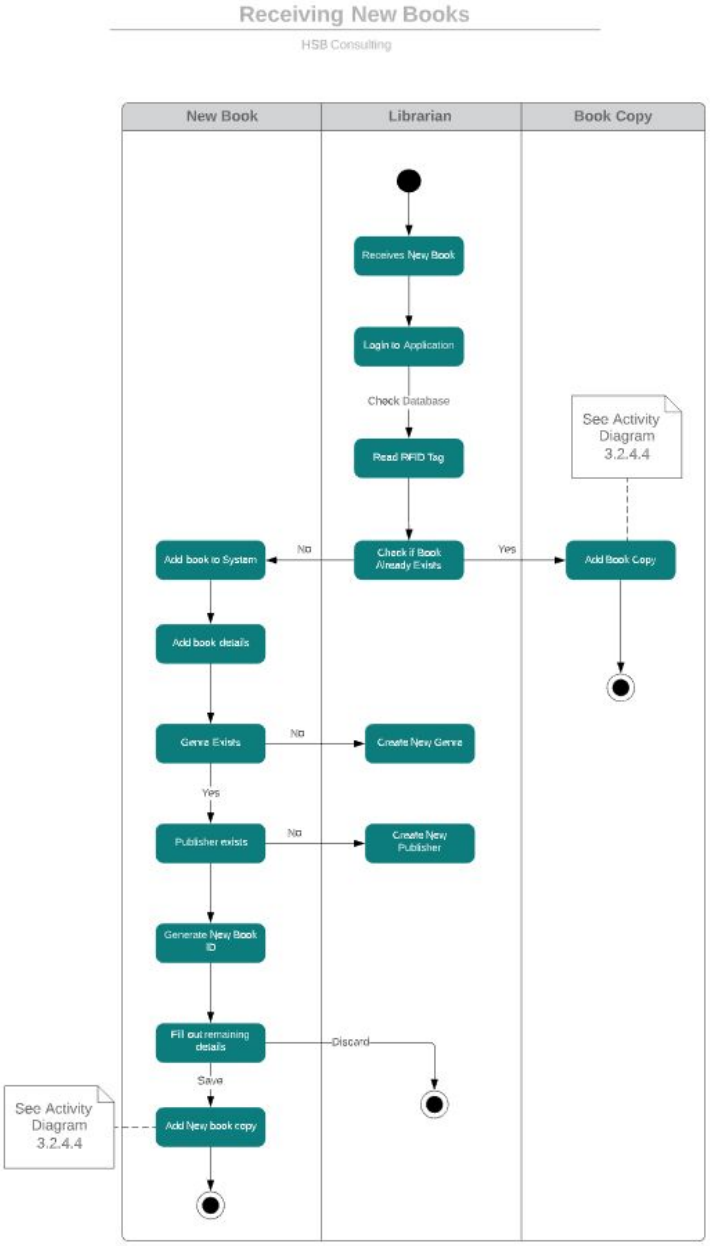


4.4.1.3 Activity Diagram - Add New Book

The Library is continuously changing their book supply as new books are always out there. Below is the activity diagram showing how a new book is added to the system.

Once a Librarian has logged in they will need to check the database to check whether the book is new or if they have it already in their system.

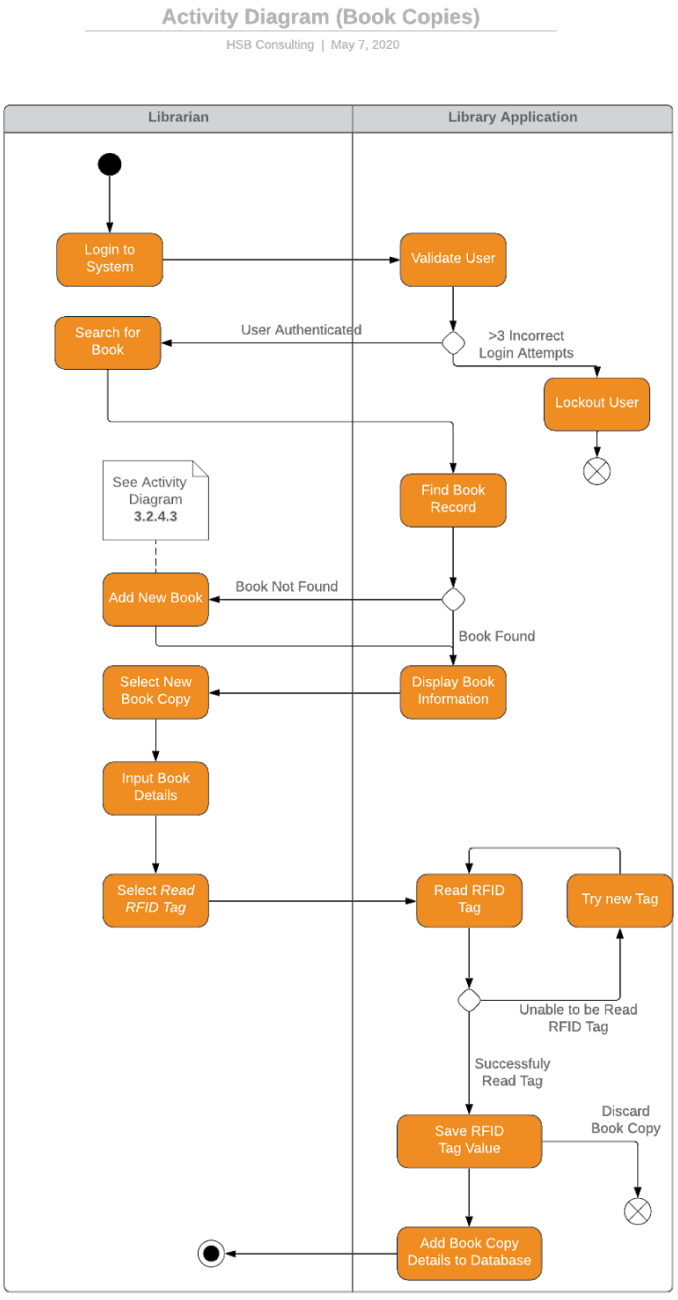
If the book is in the system then the Librarian will follow the diagram 3.2.4.4 (Adding New Book Copies), if not they will follow the process of adding a new book to the system. This includes adding the book genre, publisher and generating a new book ID.



4.4.1.4 Activity Diagram - Add New Book Copy

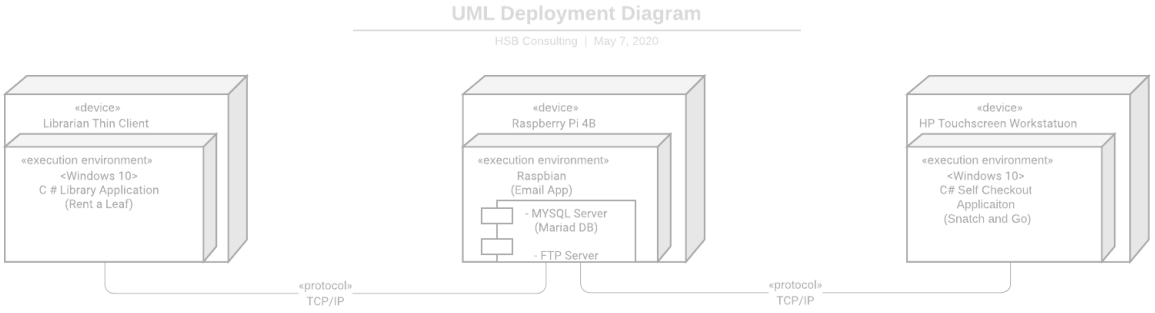
Adding a new book copy to the Library application by the Librarian. As per the system, the librarian is required to log correctly to move forward with the application. Once they are logged in, they can search the system for the book, if no record of the book is found they will then follow the process of adding a new book (3.2.4.3).

If the book is found in the system, the information will be displayed and the librarian will add the new copy/copies) to the database.



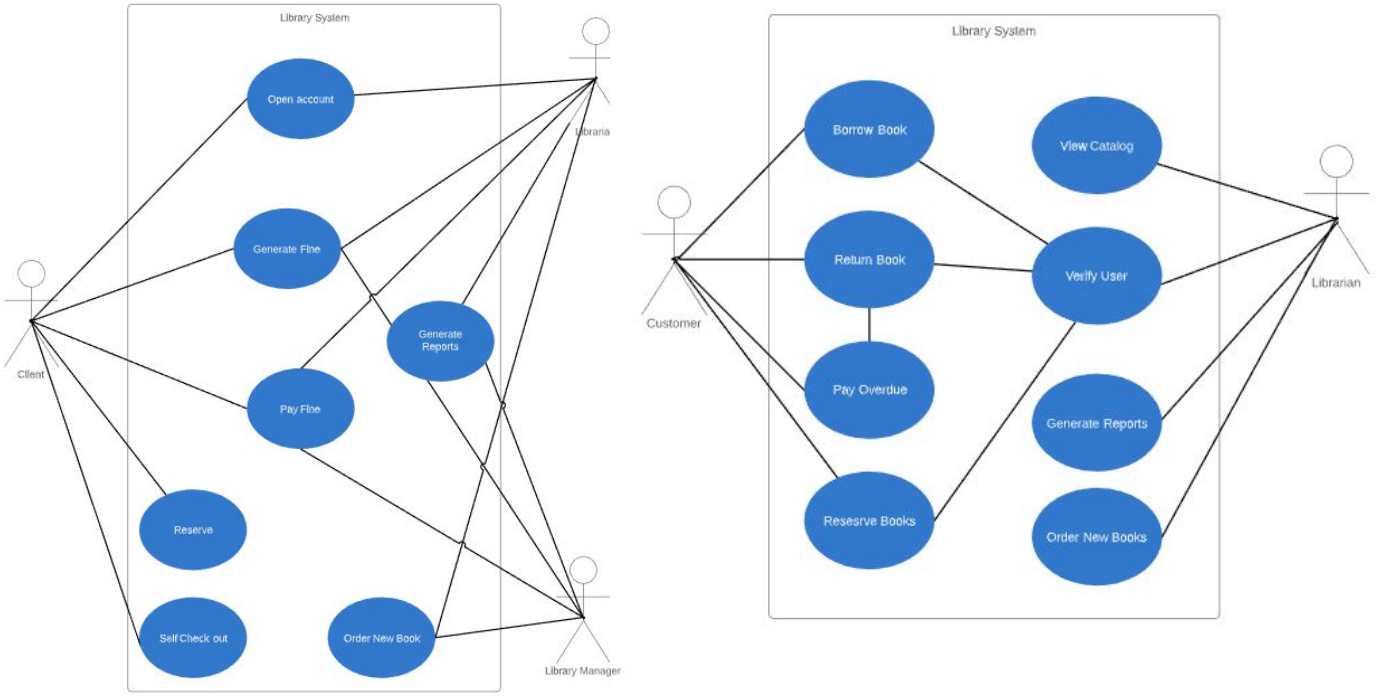
4.5 System Structural Diagram

4.5.1 UML Deployment Diagram



This Deployment Diagram shows the deployment of our finished library system. Both applications will connect to the Linux Server through the TCP/IP Protocol.

4.5.2 Use Case Diagram - Rent a Leaf

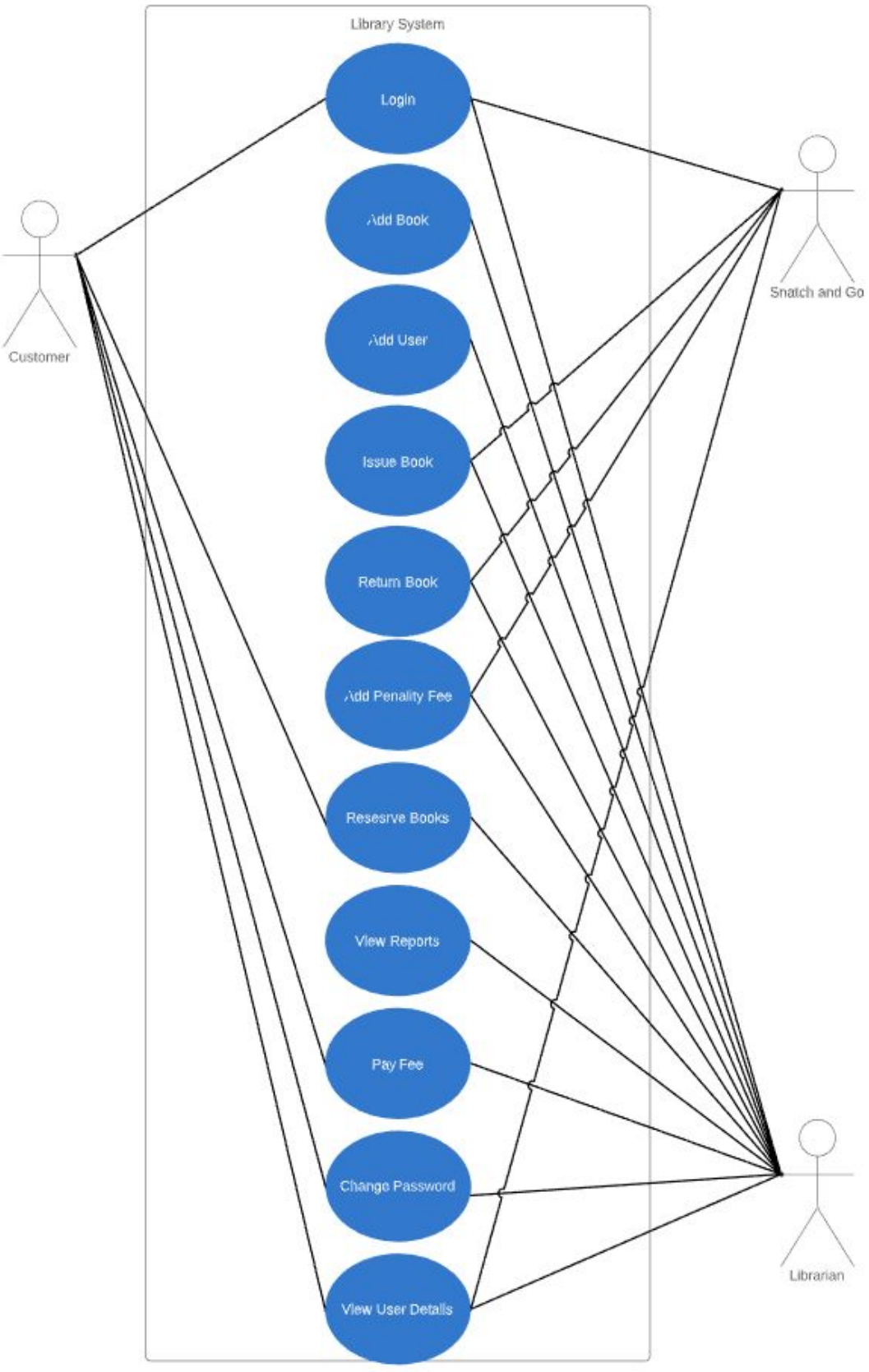


4.5.2.1 Rent a Leaf Description

ID	ucdLib1, and ucdLib2
Title	Use Case Diagram Library 1, and Use Case Diagram Library 2,
Description	This use case diagram provides a high-level view of the overall library system with assistance from the librarian and the software.
Primary Actor	Customer
Secondary Actor	Librarian
Pre-Conditions	Customers must be a registered user with the Library in order to engage with the Library's systems. Administrative interaction with the library software (i.e Librarian duties) will require the user to login and meet authentication criteria before they interact with the system.
Post-Conditions	The system itself will remain unchanged however, the information

	used within the system will be updated/changed
Main Success Scenario	The main success scenario for the overall library system is where the customer can successfully return and issue books while meeting the relevant verification criteria.
Extensions	Secondary success scenarios include the procurement of additional books, the generation of reports, and the successful display of data/information. When a customer is issuing, returning, or reserving a book, they must pass a range of validation criteria. Failure to pass these criteria will result in an error case, and the customer will not be able to complete their action. Irrelevant and improper data entry will also trigger an error event and cause the system to await proper information or drop out of the action being taken.
Frequency of Use	As a system overview, the system will be used with a high level of frequency, as this system encompasses key features of the Library software, and reflects the majority of interactions with that software.
Development Status	On-Going; Mid-Development
Ownership	Sharmaine
Priority	Extreme - As a general overview, this is the most likely to reflect the general software system, and will likely be used the most to model general software interactions.

4.5.3 Use Case Diagram - Snatch and Go



4.5.3.1 Snatch and Go Description

ID	ucdLib3
Title	Use Case Diagram Library 3,
Description	This use case diagram provides a high-level view of the Self-checkout Software, the Librarian Software and the difference between the options with the customer
Primary Actor	Customer
Secondary Actor	Librarian, Snatch and Go
Pre-Conditions	Customers must be a registered user with the Library in order to engage with the Library's systems. Administrative interaction with the library software (i.e Librarian duties) will require the user to login and meet authentication criteria before they interact with the system.
Post-Conditions	The system itself will remain unchanged however, the information used within the system will be updated/changed
Main Success Scenario	The main success scenario for the overall library system is where the customer can successfully return and issue books while meeting the relevant verification criteria using the self-checkout or seeing the librarian.
Extensions	Secondary success scenarios include the procurement of additional books, the generation of reports, and the successful display of data/information. When a customer is issuing, returning, or reserving a book, they must pass a range of validation criteria. Failure to pass these criteria will result in an error case, and the customer will not be able to complete their action. Irrelevant and improper data entry will also trigger an error event and cause the system to await proper information or drop out of the action being taken.

	The customer can only reserve books with the Librarian directly and not use the self-checkout system.
Frequency of Use	As a system overview, the system will be used with a high level of frequency, as this system encompasses key features of the Library software, and reflects the majority of interactions with that software.
Development Status	On-Going; Mid-Development
Ownership	Sharmaine
Priority	Extreme - As a general overview, this is the most likely to reflect the general software system, and will likely be used the most to model general software interactions.

5.0 Testing

Software testing is a key component of any development project, be it large or small. Throughout the development lifecycle the team has systematically tested their work for every new feature and component introduced into the Library Software. The key testing and debugging approach the team has undertaken is to check functionality throughout development and rout out any additional issues as they arise.

5.1 Step-By-Step Testing and Debugging Throughout Development

The testgrid displayed below provides an example of how each project requirement, both required and non-required, was systematically tested, checked and debugged throughout the course of development. The grid shows a clear range of login, validation, and general-functionality testing as different components were introduced into the software, and the issues that arose thereafter. However despite the range of component and requirements testing, the test-grid above doesn't fully represent the large breadth of testing that was yet to be undertaken during the final, dedicated testing phase of development.

ID	Input	Expected Result	Actual Result	Status
1	"brian"	Incorrect Username/Password	Incorrect Username/Password	Resolved
2	"0130012231"	RFID Tag not Found	RFID Tag not Found	Resolved
3	"A Barcode"	Barcode not entered in correct format	Exception Occured, Value is in incorrect format	To be Resolved
4	No Password Entered when attempting to login	Incorrect Username/Password	Incorrect Username/Password	Resolved
5	No Book Selected when viewing Book	"Please select a Book"	Exception Occurred - value can not be null	To be resolved
6	First Name not entered on Create Customer	"Please ensure all the fields are filled out"	"Please ensure all the fields are filled out"	Resolved
7	Email Address not entered on Create User	"Please ensure all the fields are filled out"	"Please ensure all the fields are filled out"	Resolved
8	Nonexistent Email or other required database value on login/validation	Prompt user "Incorrect username/ password"	Throws null value exception error	Resolved - reorganised validation checks

9	Data submission for admin user details update	Update details to database	Throws unknown column exception for sql statement	Resolved - Incorrect target table in sql syntax
10	Initial Connection to Database	Successful connection	Successful connection	Resolved
11	Close Main Form	Recursively close all other app features and forms	Main connection and Loading screen still open	Resolved
12	Open Application	Loading screen hidden on load	Loading screen remains visible	Resolved
13	Open Application without logging in	Application features hidden until logged in	Application features still accessible	Resolved
14	Open Application without Elevated Privileges	Specific application features disabled and inaccessible	Specific application features still accessible	Resolved
15	Logged Out	Disable features and clear current user details	Current user details are not cleared	Resolved

5.2 - *Snatch and Go Testing*

One area that we focused on testing and debugging is within the *Snatch and Go* Self-Checkout Application, a key component of our system for *The Library*.

Due to how all of the Object Mapping and Manipulation is done automatically, the following testing that has been performed is user testing for the *Snatch and Go* Application.

5.2.1 User Authentication

A common issue that might arise is, if a user was to present an invalid User ID (such as an ID Card which has an RFID Tag that has not been correctly stored into the database), or if they were to accidentally present a book on the scanner during the login screen.

Screenshot of issue (and resolution)

Testing Parameters

Input Value	**An Invalid ID Card has been presented (one not Registered, or a book is placed on the sensor)**
Expected Output	A message is displayed informing the user that it is not able to be identified
Actual Output	A message is displayed informing the user that it is not able to be identified
Status of Issue	Resolved

5.2.1 Book Age Appropriate

Another common scenario within our application that might occur is that a customer may try to issue a book which is not recommended for their age based on the genre of the book. Due to the way we have setup and configured the system, we do not want to allow a customer to issue a book if it is not age appropriate for them.

Screenshot of Issue (and resolution)

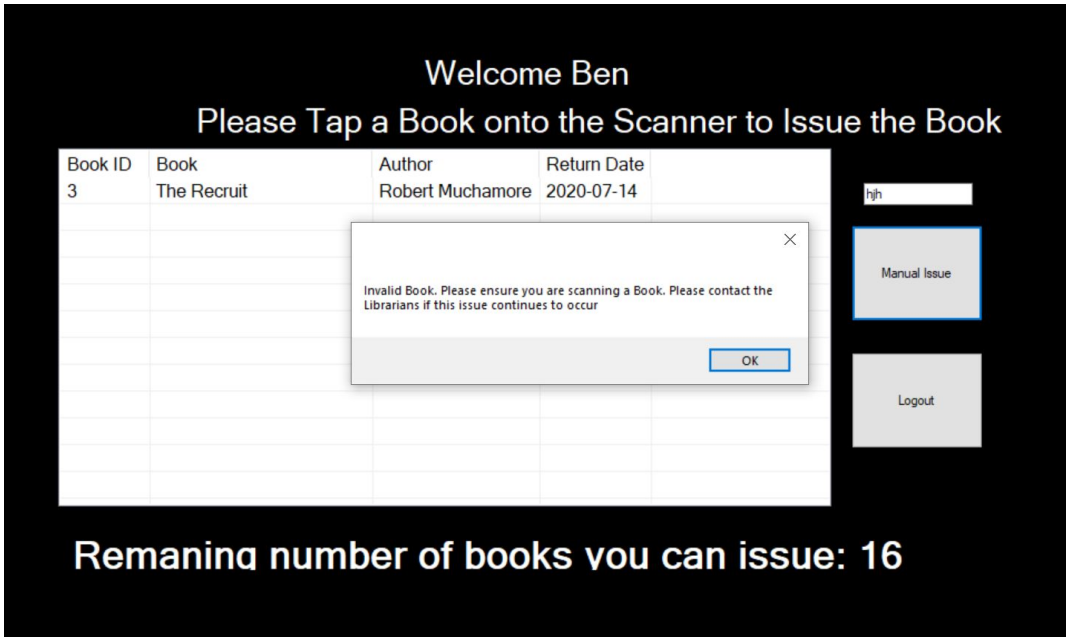
Testing Parameters

Input Value	**A Book is placed on the RFID Scanner to be issued which has a higher age rating then the age of the customer.**
Expected Output	A message is displayed informing the customer that the book is not age appropriate for them.
Actual Output	A message is displayed informing the customer that the book is not age appropriate for them.
Status of Issue	Resolved

5.2.2 Book Authentication - Issuing Books

Another common scenario that might occur within our application is if a user was to present an invalid book (e.g. a Book whose RFID Sensor is blank and not stored in the database), or a Customer ID Card on the Issue Screen.

Screenshot of Issue (and resolution)



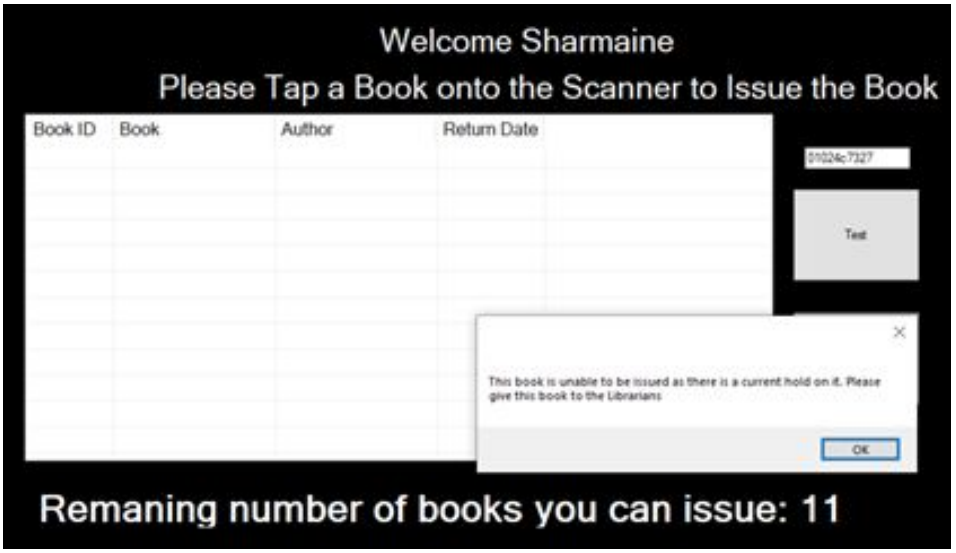
Testing Parameters

Input Value	**An Invalid Book is placed on the sensor, or a Customer ID is placed on the sensor**
Expected Output	A message is displayed informing the customer that the book is not able to be identified.
Actual Output	A message is displayed informing the customer that the book is not able to be identified.
Status of Issue	Resolved

5.2.3 Book currently on Hold

Holds are able to be placed on books which are not currently available for issue (are currently out). While the procedure within *The Library* is to check whether a book has a hold on it when it's returned, there is always the potential that a book may be misplaced and be put on the shelf and selected by a customer before a librarian can put it on the hold shelf.

Screenshot of Issue (and resolution)



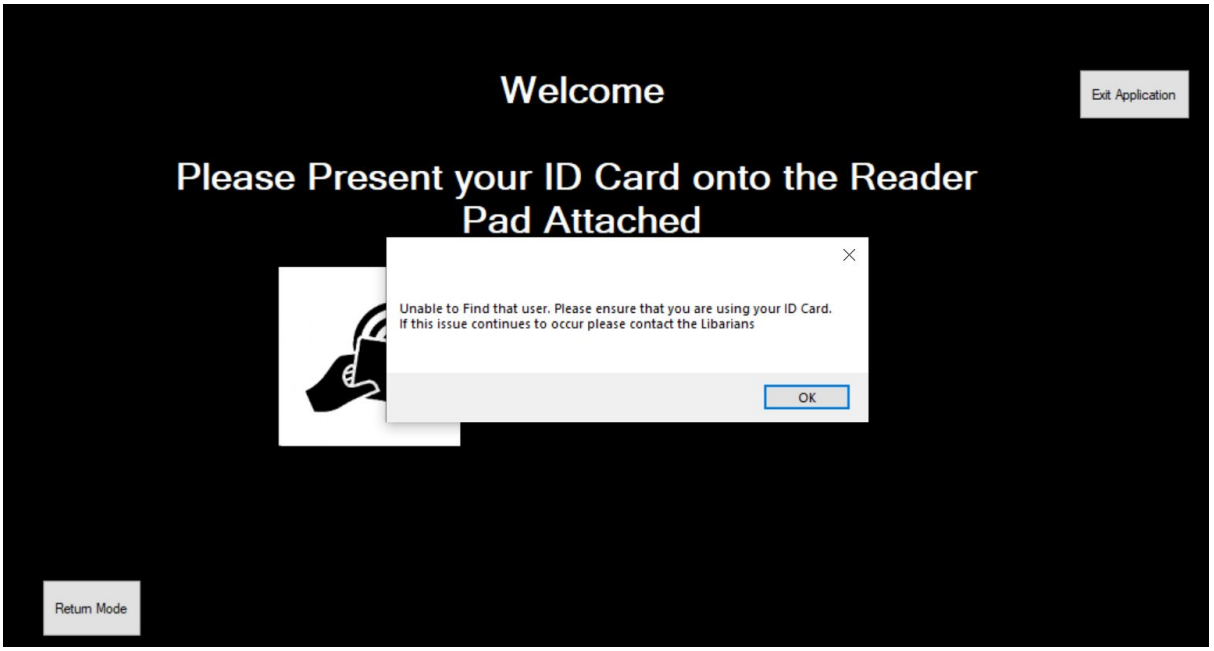
Testing Parameters

Input Value	**A Book is placed on the scanner which currently has a hold on it.**
Expected Output	A message is displayed to inform the customer that the book is unable to be issued and to give it to the librarians
Actual Output	A message is displayed to inform the customer that the book is unable to be issued and to give it to the librarians
Status of Issue	Resolved

5.2.4 SQL Injection

A customer could try to perform SQL injection, by typing in SQL Statements (such as DROP TABLE BOOKSONHOLD) within the textbox that allows a user to manually login, or tries to do this within the manual issue textbox.

Screenshot of Issue (and Resolution)



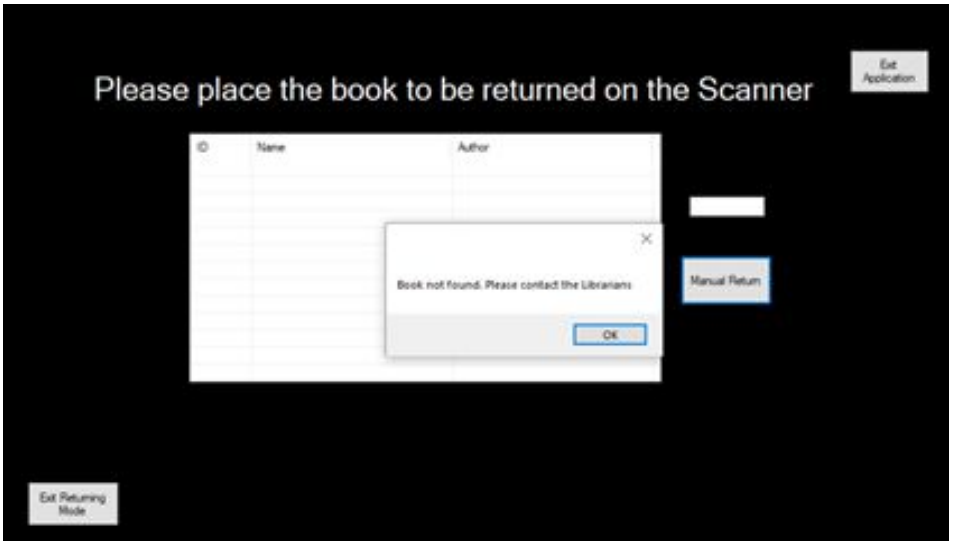
Testing Parameters

Input Value	**DROP TABLE BOOKSONHOLD**
Expected Output	The SQL Injection Statement is ignored, passed through as a Parameter resulting in the error message of the Book/Customer unable to be found
Actual Output	The SQL Injection Statement is ignored, passed through as a Parameter resulting in the error message of the Book/Customer unable to be found
Status of Issue	Resolved

5.2.5 Book Authentication - Returning Books

Another common scenario that might occur within our application is if a user was to present an invalid book (e.g. a Book whose RFID Sensor is blank and not stored in the database), or a Customer ID Card on the return Book Screen.

Screenshot of Issue (and resolution)



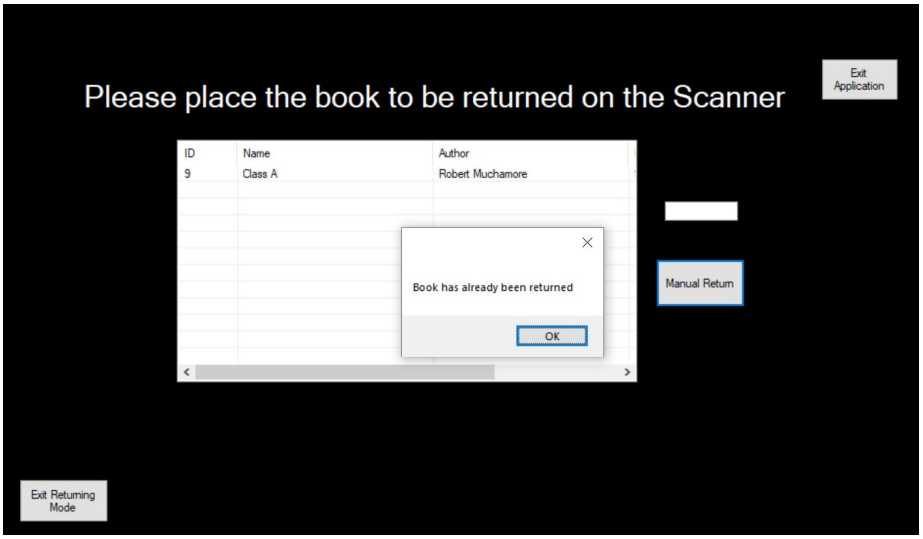
Testing Parameters

Input Value	**An Invalid Book is placed on the sensor, or a Customer ID is placed on the sensor**
Expected Output	A message is displayed informing the customer that the book is not able to be identified.
Actual Output	A message is displayed informing the customer that the book is not able to be identified.
Status of Issue	Resolved

5.2.6 Book Repetition - Returning Books

Another common scenario that might occur within our application is if a user was to present the same bookcopy twice or more, after the book has been previously returned during this returning session.

Screenshot of Issue (and resolution)



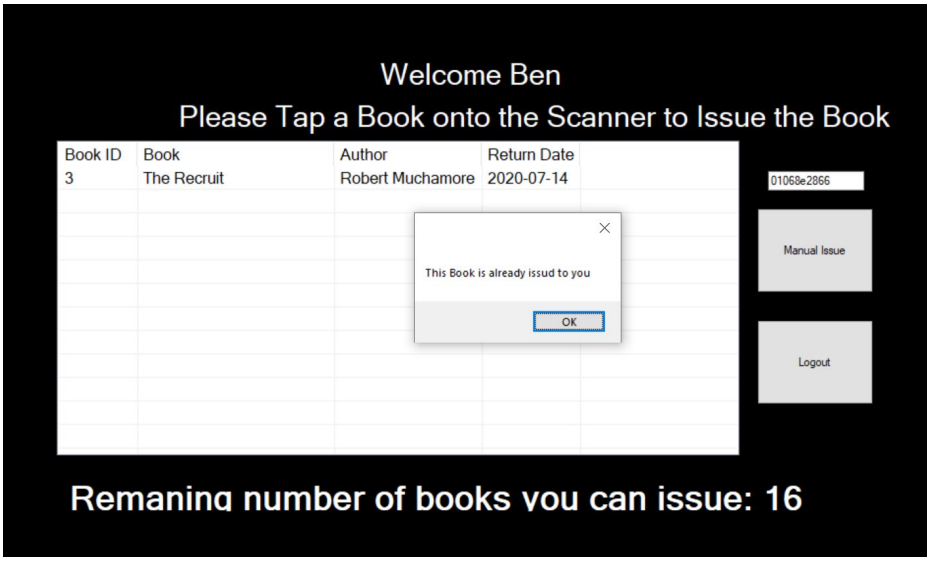
Testing Parameters

Input Value	**The Same Book is placed on the RFID Sensor again (more than once)**
Expected Output	A message is displayed informing the customer that the book has already been returned
Actual Output	A message is displayed informing the customer that the book has already been returned
Status of Issue	Resolved

5.2.7 Book Repetition - Issuing Books

Another common scenario that might occur within our application is if a user was to present the same bookcopy twice or more, after the book has been previously issued during this issuing session.

Screenshot of Issue (and resolution)



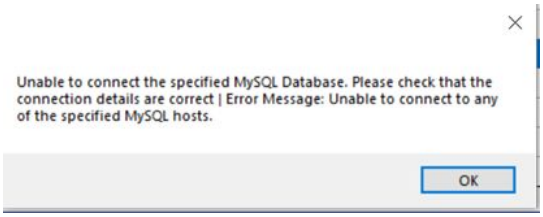
Testing Parameters

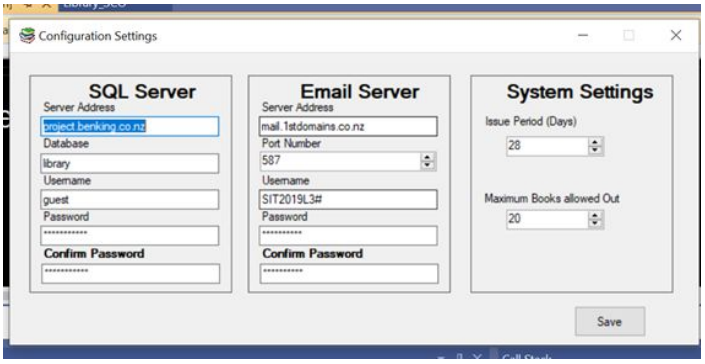
Input Value	**The Same Book is placed on the RFID Sensor again (more than once)**
Expected Output	A message is displayed informing the customer that the book has already been returned
Actual Output	A message is displayed informing the customer that the book has already been returned
Status of Issue	Resolved

5.2.8 Connection Issues

Another common scenario that might occur within our application is if a user was to try to run the application, trying to establish the connection with our MySQL Database. If for some reason it was unable to connect (e.g. Internet Outage, Configuration Settings to connect to the MySQL Database need updated), there would need to be an appropriate message informing the user

Screenshots of Issue (and resolution)





Testing Parameters

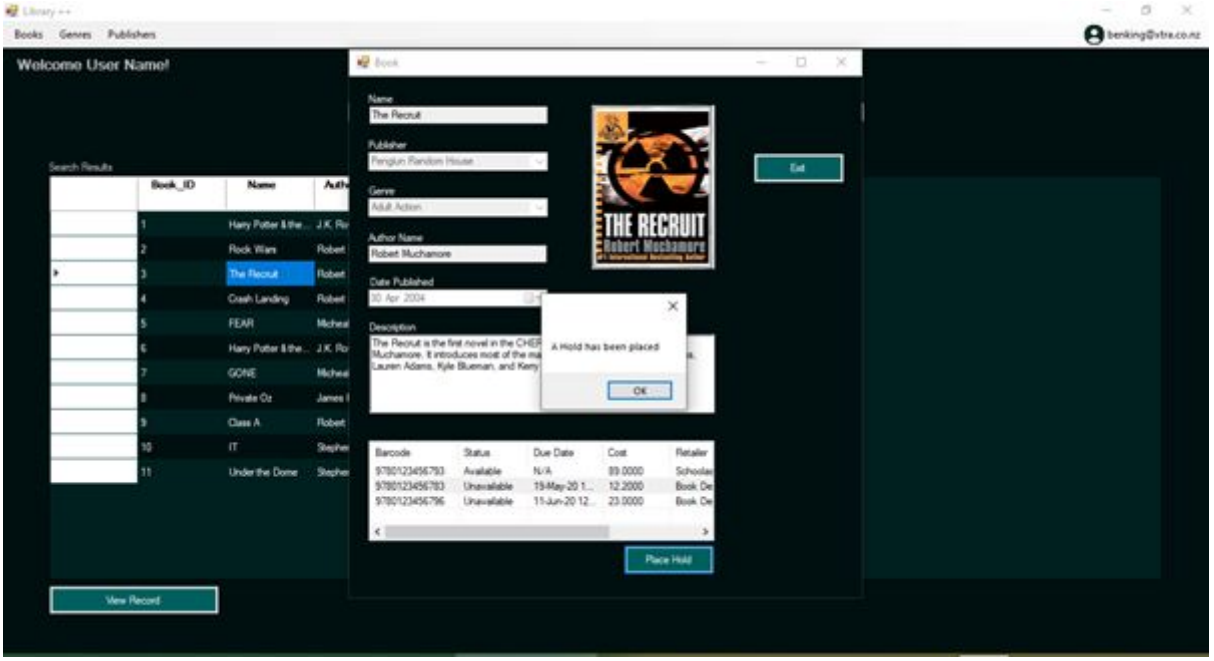
Input Value	Unable to Establish a Connection to the MySQL Database
Expected Output	A message is displayed informing the customer that it is unable to connect to the database, and loads the Configuration Screen
Actual Output	A message is displayed informing the customer that it is unable to connect to the database, and loads the Configuration Screen
Status of Issue	Resolved

5.3 Library Management Software Testing (Rent a Leaf)

5.3.1 Customer Details and Book Management Interoperability

As a Library Management System, it is expected that customers will regularly be utilizing the Book Management component of the software for the purpose of viewing books and placing them on hold. As such this component of the software will be expected to operate alongside customers without fail.

5.3.1.1 Screenshots of Interoperability Testing

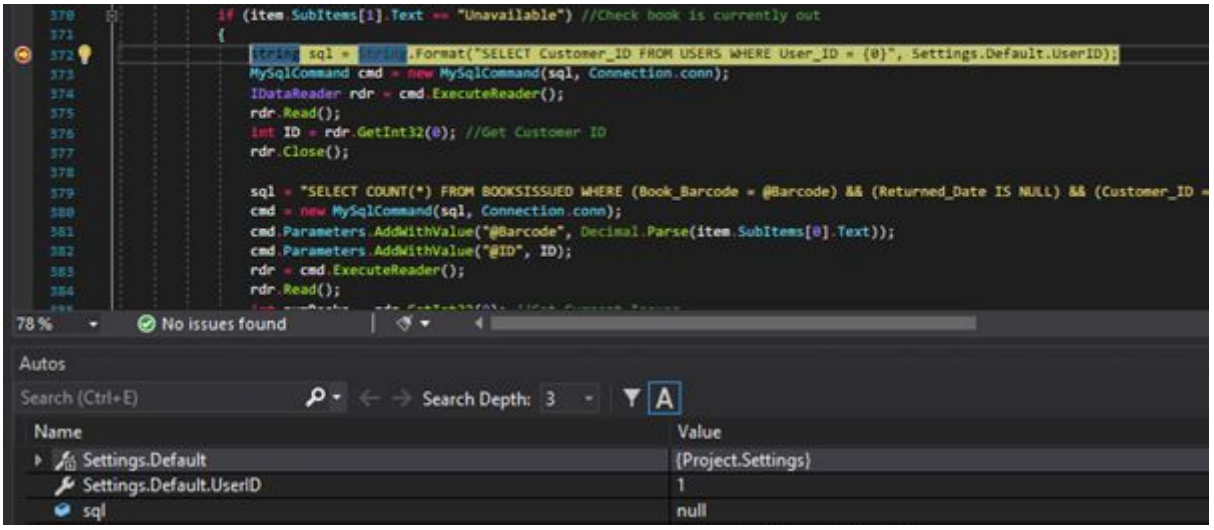


5.3.1.2 Testing and Debugging Process

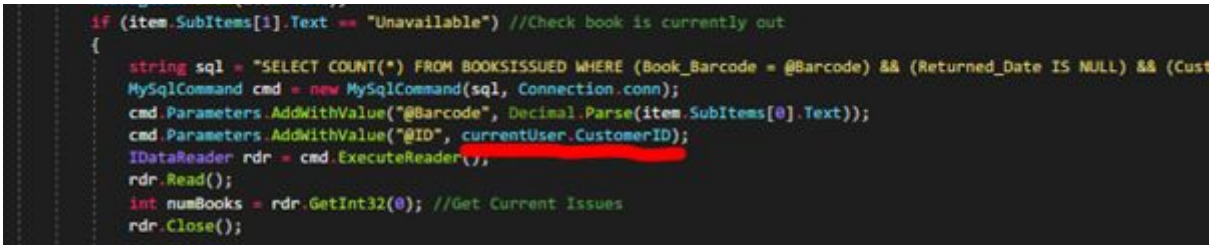
Input Value	A Valid Hold is Placed on a Book By Clicking the Place Hold Button for a Targeted Book
Expected Output	It was expected that when a customer placed a book on hold within the Book form, this reserved book would be assigned to that customer. This would be accessible within the Customer Details form under the “Holds” tab.
Actual Output	As depicted by the screenshots above, the reserved book was not appearing within the Customer Details Form, indicating that either the book was not assigned, or the book was not being retrieved by the Customer Details Form.
Status of Issue	Resolved

When viewing other customer details and analyzing the SQL command used to retrieve information for Customer Details it was found that the issue did not lie with Customer Form as it was performing its intended role in information retrieval.

The issue could therefore be narrowed down to the way Holds were being placed within the Book Management Form itself. Upon inspecting this, it was found that the SQL query being used to retrieve information from the Current User for future processing was being passed the default Customer_ID, likely as derelict code from previous testing. This resulted in any holds placed by any customer to be registered with Customer_ID 1, where 1 is set as the default Customer_ID.



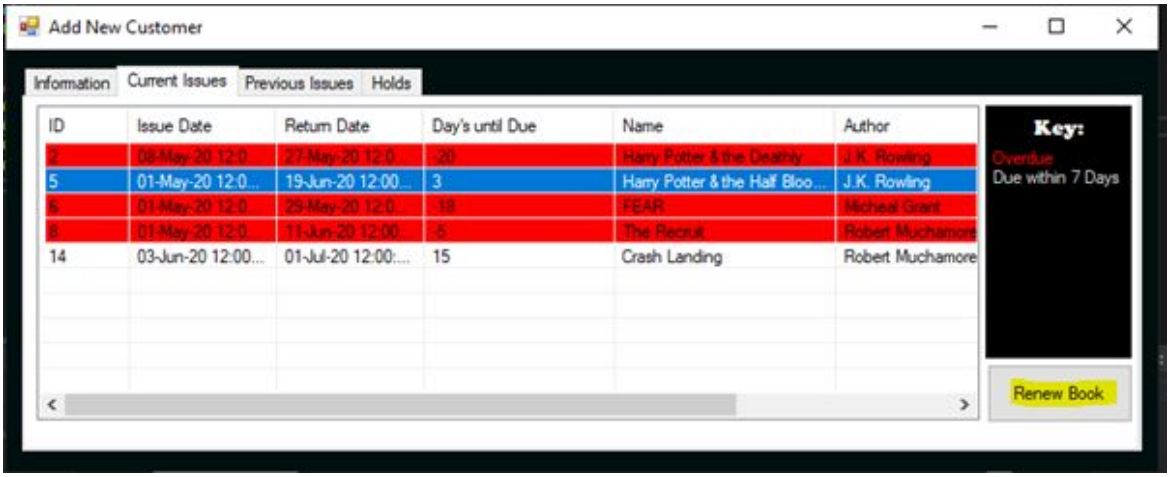
It was found that this initial step in placing a hold in of itself was redundant, as the customer ID of the current Customer was already being passed to the Book Management Form for the purposes of validation and assignment, so removing this and setting the Hold method to use the correct ID proved to correct the error.

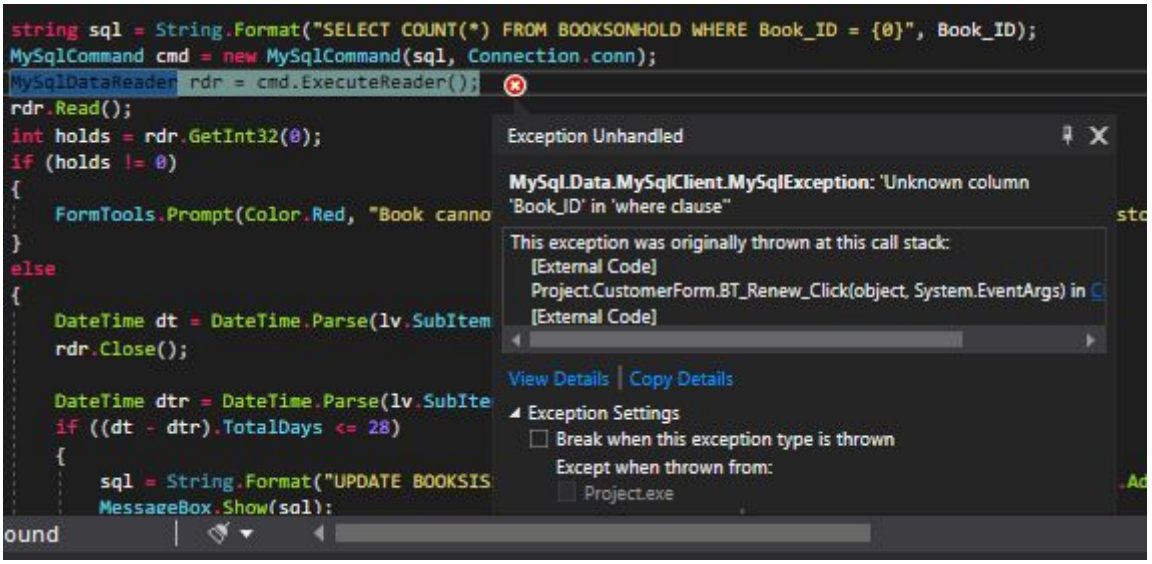


5.3.2 Customer Renew-Issue Management Testing

Likewise with placing books on hold, it is also expected that the ability to Renew books will also be a heavily utilised component of the software. The importance of this component necessitates thorough testing, and debugging where needed.

5.3.2.1 Screenshots of Book Renew Testing

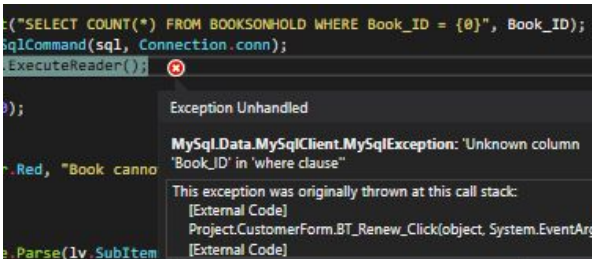


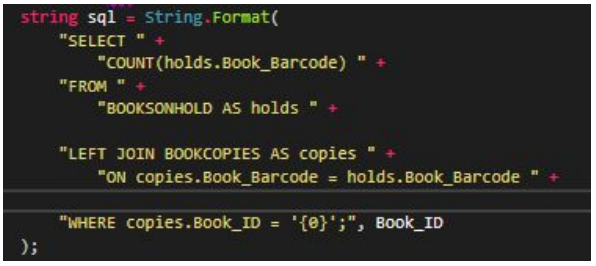


5.3.2.2 Testing and Debugging Process

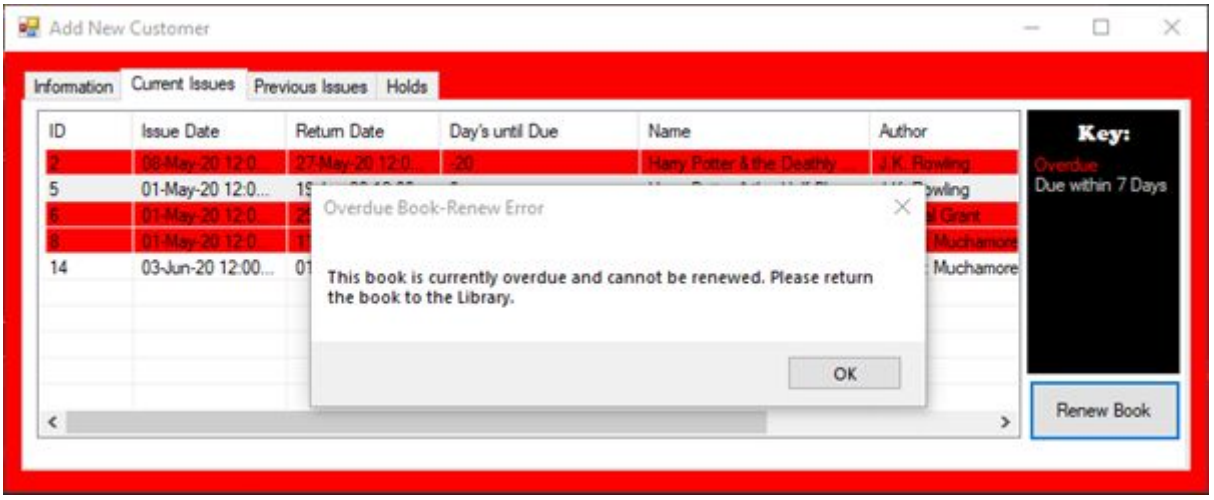
Input Value	**A customer highlights the book they would like to renew and clicks the Renew button located in the bottom left of the Customer Details Form on the Current Issues page.**
Expected Output	If the book is not overdue, and the book has not been placed on hold by another customer, the book is renewed, and the due date is extended.
Actual Output	The application crashes on a valid book renewal, and all other renewal selections return as being invalid due to the book being overdue suggesting a logic issue within both the programming and SQL of the Renew Book component of the Customer Details form.
Status of Issue	Resolved

When looking into this issue, it became immediately apparent that the SQL being used to select the number of holds placed on the renewed book did not reflect that of our data structure, as the Book_ID of the books on hold is not represented in our BOOKSONHOLD table. This was fixed by replacing the original query (left) with one that takes advantage of a LEFT JOIN to retrieve the relevant information (right).



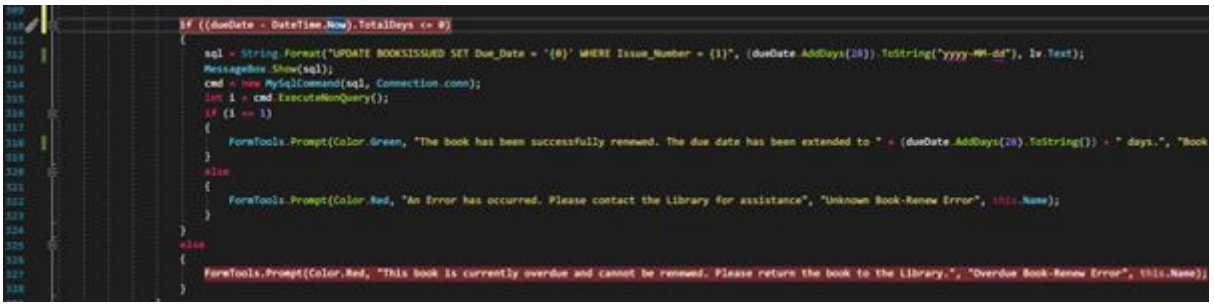


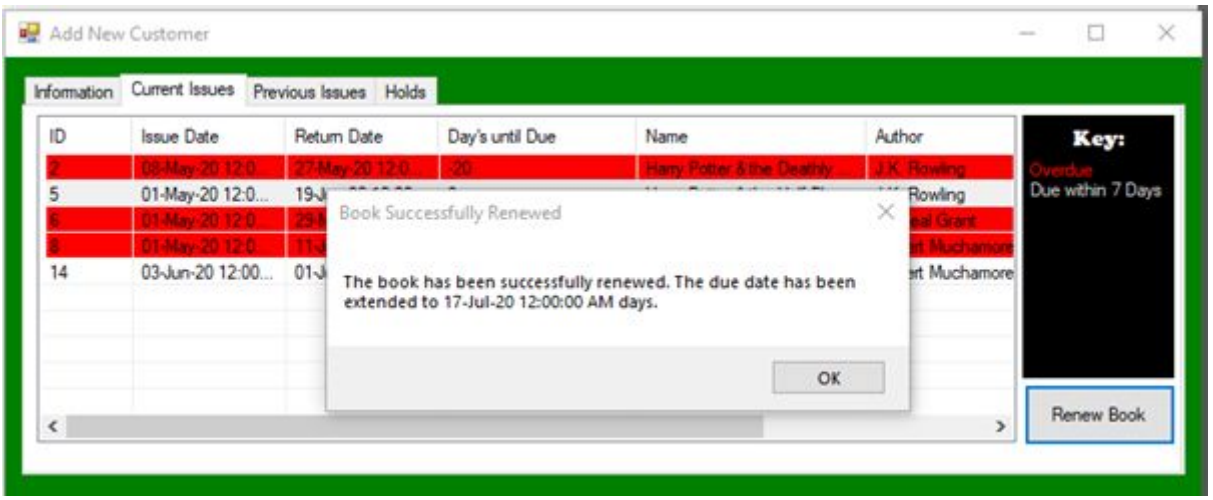
This only proved to remedy the original crash however, and all book renewals were still returning as being overdue, cementing our previous hypothesis that the primary issue is logic-based within the programming of our application.



Upon analysis of our code it was found that the issue lay with the “if” condition intended to check the amount of time left before the book is due. The calculation involved would always return the same positive value, as the condition was getting the Date Due minus the Date Issued which chronologically would never be negative.

Replacing the Issue-Date value with the current date returns how long until the book is overdue. This will be 0 or negative if the book passes it’s Due-Date, and will be positive if it is not yet due. This means the book can be renewed as it is in this instance. The ternary operator involved in checking if the book is overdue was also incorrect as it is testing for a number of days left that is less-than or equal to 0 before the book is due. In other words, that the book must be overdue to return true, which would have been impossible due to the previous equation. This was promptly amended, as shown in the two screenshots on the next page.

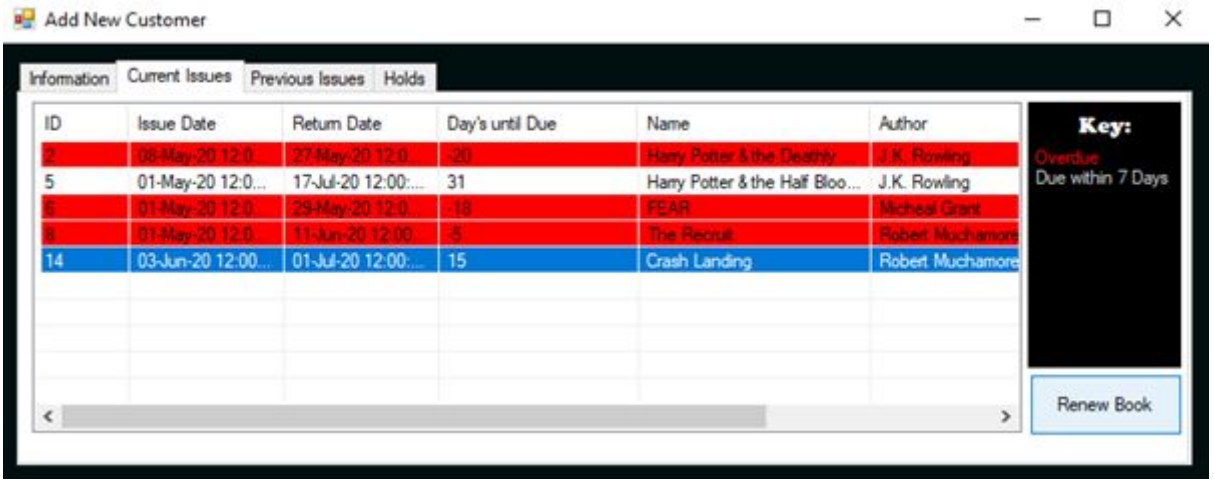


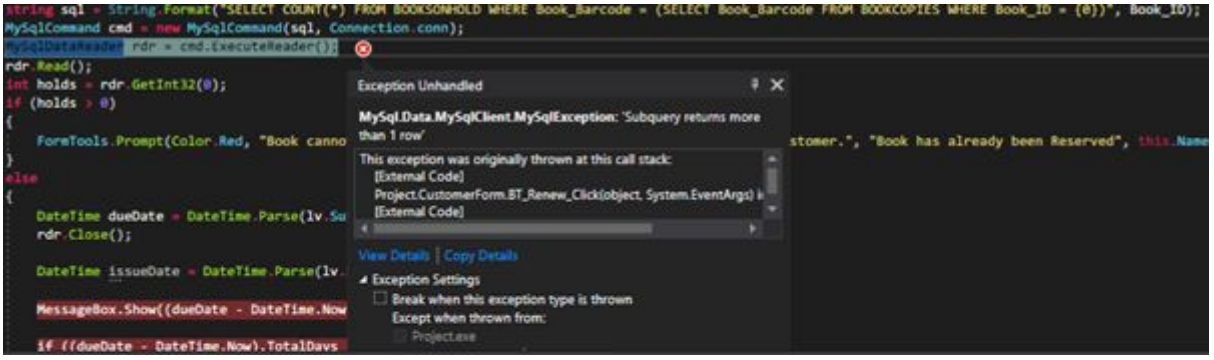


5.3.3 Renewing Reserved Books

With the core renew-book component stabilized another issue arose with the renewing of books that had been placed on hold by another customer.

5.3.3.1 Screenshots of Renew Book Testing





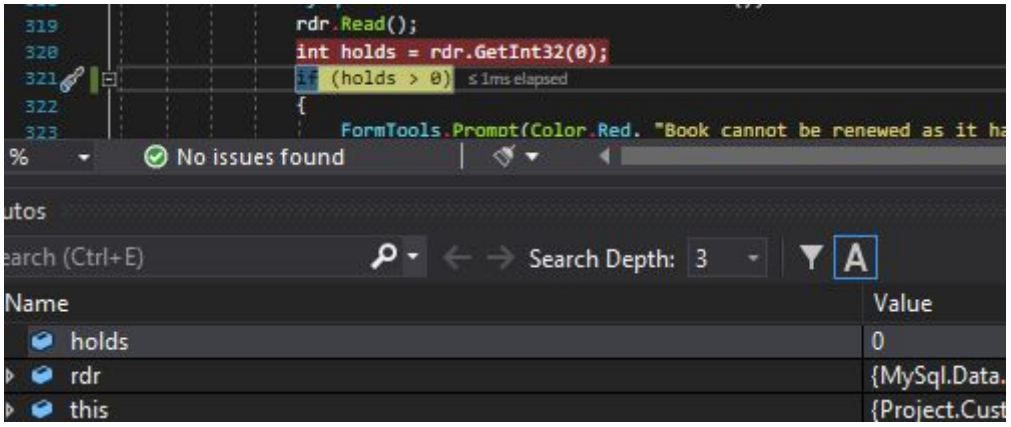
5.3.3.2 Testing and Debugging Process

Input Value	**A customer highlights the book they would like to renew and clicks the Renew button located in the bottom left of the Customer Details Form on the Current Issues page.**
Expected Output	If the book has not been placed on hold by another customer, the book is renewal is denied
Actual Output	The application encountered a fatal application crash due to a thrown exception
Status of Issue	Resolved

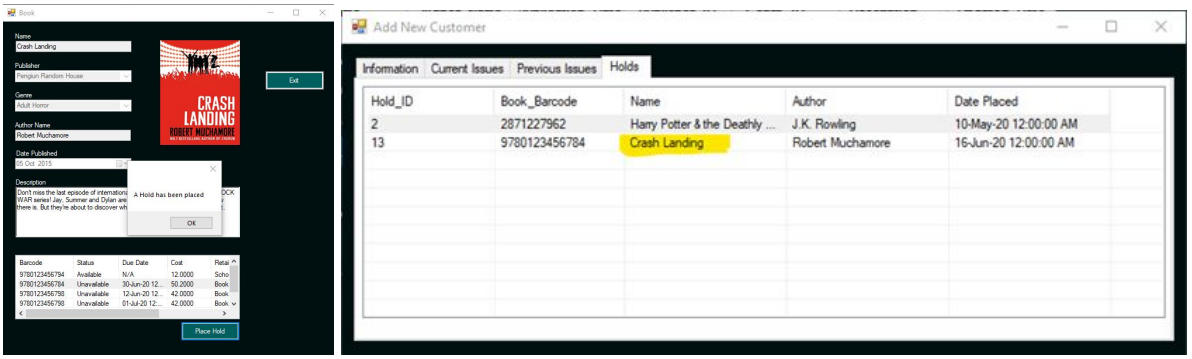
After analysing the error message, it became apparent the issue lay with the SQL select statements being used. The desired results could also be achieved using a JOIN or UNION statement similar to the one previously discussed. Below is a screenshot showing the previous SQL query highlighted in green, and the new SQL query highlighted in yellow.

```
            string sql = String.Format(
                //"SELECT COUNT(*) " +
                //"FROM BOOKSONHOLD " +
                //"WHERE Book_Barcode = " +
                //    "(SELECT Book_Barcode " +
                //    "FROM BOOKCOPIES " +
                //    "WHERE Book_ID = {0})", Book_ID);


        "SELECT " +
            "COUNT(holds.Book_Barcode) " +
        "FROM " +
            "BOOKSONHOLD AS holds " +

        "LEFT JOIN BOOKCOPIES AS copies " +
            "ON copies.Book_Barcode = holds.Book_Barcode " +

        "WHERE copies.Book_ID = '{0}';", Book_ID);
```

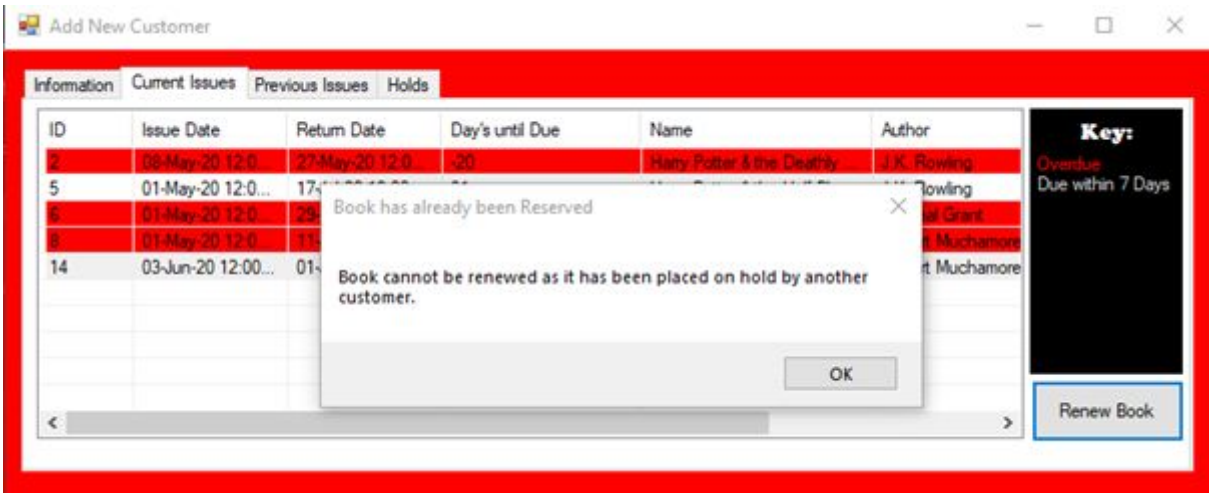
After replacing the old SQL query with the new functioning query, the program no longer threw an error, however, to ensure the new SQL statement was working as intended, we decided to ensure there are no holds placed on the book by analysing the data being retrieved.



Searching the database revealed that there are no holds on the target book, reflected by the image above. For further testing purposes, we decided to place this book on hold in the database to ensure that if there is an existing hold, it is picked up by the SQL and that the remainder of the code functions to prevent us encountering any additional uncaught errors.



Running through the renewal process again proved that the number of holds placed on the target book was being properly returned by the SQL query. This allowed us to use this for validating weather the book could be placed on hold. This allowed us to confirm that the renewal process has been adequately tested, as books renewed when already reserved are denied, when the book is overdue it is denied, and when within the issued timeframe without being reserved, the renewal is accepted.



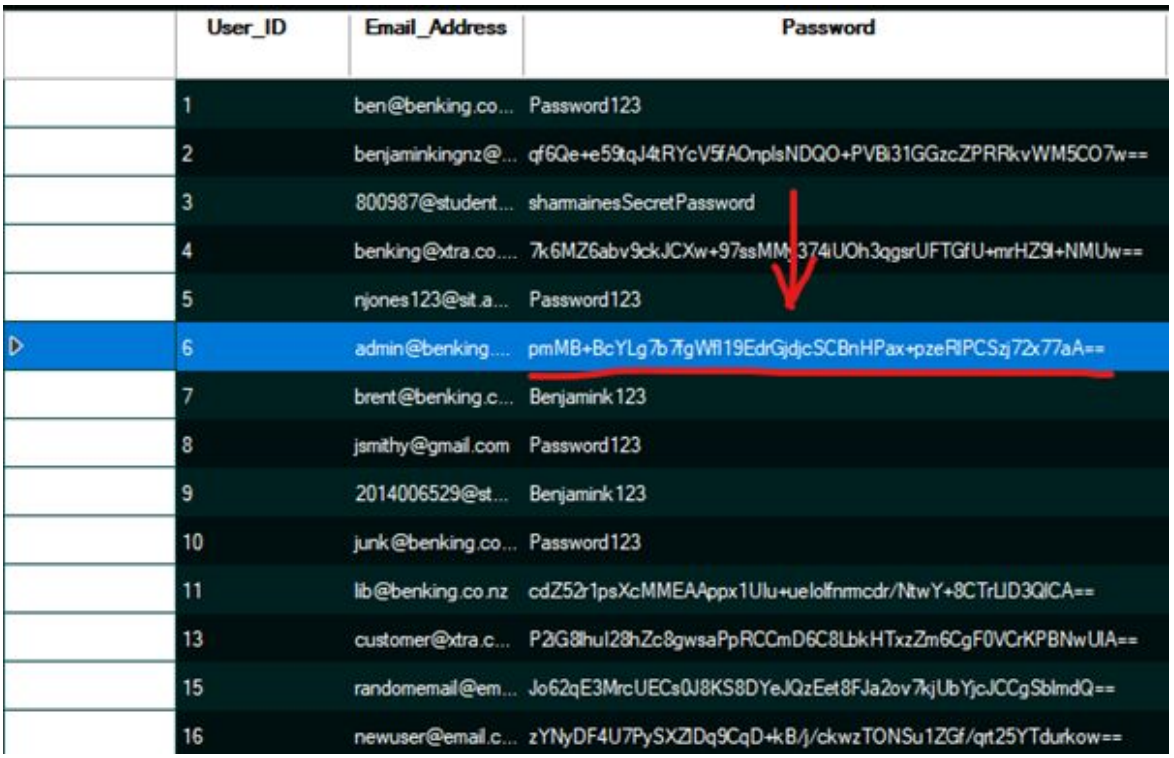
5.3.4 Password Hashing and Storage

The security of personal information is vital whenever it is handled. There is a heavy legal and ethical obligation to make sure that personal information is stored safely and our customers will expect a high degree of security and professionalism when dealing with information such as passwords. This means that any security measures we undertake for the management of user information and accounts are of the utmost importance, and need to function one hundred percent of the time.

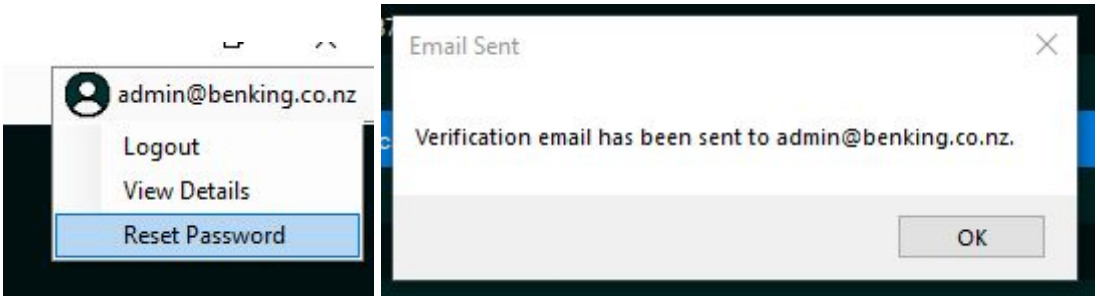
5.3.4.1 Password Hashing Testgrid

Input Value	**A customer has reset their password using the "Reset Password" form by filling in their details and clicking the button in the bottom right corner.**
Expected Output	The password entered will be encrypted with a one-way hash function for secure storage.
Actual Output	The password entered has been encrypted with a one-way hash function for secure storage.
Status of Issue	Resolved

5.3.4.2 Supporting Images



	User_ID	Email_Address	Password
	1	ben@benking.co...	Password123
	2	benjaminkingnz@...	qf6Qe+e59tqJ4tRYcV5fAOnplsNDQO+PVBi31GGzcZPRRkvWM5CO7w==
	3	800987@student....	sharmainesSecretPassword
	4	benking@xtra.co....	7k6MZ6abv9ckJCXw+97ssMMy374iUOh3qgsrUFTGfU+mrHZ9I+NMUw==
	5	njones123@sit.a...	Password123
▷	6	admin@benking....	pmMB+BcYLg7b7fgWfl19EdrGjdjcSCBnHPax+pzeRlPCSzj72x77aA==
	7	brent@benking.c...	Benjamink123
	8	jsmithy@gmail.com	Password123
	9	2014006529@st...	Benjamink123
	10	junk@benking.co...	Password123
	11	lib@benking.co.nz	cdZ52r1psXcMMEAAppx1Ulu+uelolfnrmcdr/NtwY+8CTrLID3QlCA==
	13	customer@xtra.c...	P2G8lhuI28hZc8gwsaPpRCCmD6C8LbkHTxzZm6CgF0VCrKPBNwUlA==
	15	randomemail@em...	Jo62qE3MrcUECs0J8KS8DYeJQzEet8FJa2ov7kjUbYjcJCCgSblmdQ==
	16	newuser@email.c...	zYNyDF4U7PySXZlDq9CqD+kB/j/ckwzTONSu1ZGf/qrt25YTdurkow==



	Customer_ID	User_ID	Email_Address	Password
	1	1	ben@benking.co...	Password123
	2	2	benjaminkingnz@...	qf6Qe+e59tqJ4tRYcV5fAOnplsNDQO+PVBi31GGzcZPRRkvWM5CO7w==
	3	3	800987@student....	sharmainesSecretPassword
	4	4	benking@xtra.co....	7k6MZ6abv9ckJCXw+97ssMMy374iUOh3qgsrUFTGfU+mrHZ9I+NMUw==
	5	5	njones123@sit.a...	Password123
▷		6	admin@benking....	uKXS97TM2Ga5tm+mB0HgsK1IEXvX8xRG6Tg9vPKnw4UjCGO5Z3ZI2w==
		7	brent@benking.c...	Benjamink123
	6	8	jsmithy@gmail.com	Password123
		9	2014006529@st...	Benjamink123
		10	junk@benking.co...	Password123
		11	lib@benking.co.nz	cdZ52r1psXcMMEAAppx1Ulu+uelolfnrmcdr/NtwY+8CTrLID3QlCA==
	9	13	customer@xtra.c...	P2G8lhuI28hZc8gwsaPpRCCmD6C8LbkHTxzZm6CgF0VCrKPBNwUlA==
	10	15	randomemail@em...	Jo62qE3MrcUECs0J8KS8DYeJQzEet8FJa2ov7kjUbYjcJCCgSblmdQ==
		16	newuser@email.c...	zYNyDF4U7PySXZlDq9CqD+kB/j/ckwzTONSu1ZGf/qrt25YTdurkow==

6.0 User interface designs

6.1 Description of the User Interface, Objects, and User Actions

For this section we will be talking about the user interface design of our main systems component, which is the Rent a Leaf (Librarian/Customer) Application. The Rent a Leaf Application is a form based user driven application, where events and actions are driven by the users interaction with the form, such as by clicking buttons and selecting options. Within this section we will be referring to the Screen Captures of the application in section **6.1.2 Screen Images.**

We have designed and developed the Rent a Leaf application with a Multi Level Tree Menu, allowing multiple parents and multilevel transversal. The parent of our application is the menu/search screen where all of the interaction within our application starts from and finishes at. Most of the interaction throughout our application is form based, allowing the user to enter information through a form.

UI Design Properties and Selected Values		
Properties	**Property Value**	**Property Sample**
Background Colour	RGB 0,16,16	
Primary Button Colour	RGB 0,96,96	
Secondary Button Colour	RGB 0,64,64	
Container/Area Colour	RGB 0,32,32	
Primary Text Colour	HTML Silver	
Warning Text Colour	HTML Red, Italicised	
Font Family	Microsoft Sans Serif, 8.25pt	This is an example with some *warning* text as well

6.1.1.1 Initial Login

As shown in **Figure 1 of 6.1.2**, the user is initially shown a brief loading screen. This is both to provide user feedback, and to load background assets such as automated application features and database connections. If these processes are unsuccessful, the loading screen will display this to the user along with an error message in the bottom left corner to assist with troubleshooting potential issues.

When the application loads successfully, the user is shown a login window to gain entry to the main application. If the credentials entered are wrong, the user will be notified. If they are wrong 3 or more times in a row, they will then be notified they have been locked out for 30 seconds, the timer doubling after each third consecutive failed login.

Should the user forget their password, there is a button located beneath the password field on the right-hand side to allow users to allow the user to reset it. This will open a prompt window for the user to enter their new password with their confirmed email address. If the new credentials are valid the final prompt will appear, and a 6 digit confirmation key will be sent to their email address for the user to enter. If the key is valid the user is provided a popup confirmation of a successful password reset.

If the user closes the login prompt without logging in, the main application will still load, however the user will be unable to use any of the applications features except for the login button located in the menustrip in the top-right of the application. Clicking this button will re-initiate the process laid out in **Figure 1, 6.1.2**, without the loading screen.

6.1.1.2 Book and Book Copy Management

The user interaction map shown in **Figure 2, 6.1.2** shows how the user can access a variety of features to manage books and book copies within the library infrastructure. The user can view a list of books within the library infrastructure through the main application window. This can be done by either selecting the “Books” dropdown selection just above the dataviewer within the search bar component, or by selecting the “View Books” option in the sub-menu for “Books” located in the top menu strip.

To view the details of a book in detail, the user then has the option to double-click the specific record they would like to view, or optionally, they can highlight the record and click the “Update/View Record” button located in the bottom right corner of the dataviewer. This method for accessing book details was undertaken so as to ensure that navigation and utilization of the Library application is intuitive and easy for a wide range of users who are used to navigating different software. E.g: It may not be immediately apparent for some users that detailed information can be viewed by the double-click access method. This standard of practice is kept consistent throughout the application, and can be seen on display in **Figures 2 through 5 of 6.1.2**, and is discussed in greater depth in **6.2.2**.

Within the individual viewing form of **Figure 2, 6.1.2,** there is a range of input fields containing information on each book. These input fields are disabled by default, and can be enabled by users with the correct login privileges by selecting the “Enable Editing” checkbox in the top left corner of the form. For the purpose of security and data integrity, this option will be unavailable to customers and users without the correct access rights, and user interactions with the form will end at general information viewing. This practice is also reflected in all other forms, where the user can view details in a more legible format via the two described methods, and edit those details if they are authorized to do so.

At the bottom of the page, there is a button that allows the user to place a book on hold. Selecting a book copy and clicking this button will return with one of three prompts based on whether the book can be put on hold. If the user possesses edit permissions, two additional buttons will be visible beneath the book copy information displayed at the bottom of the form. These buttons let the user add and edit copies of the book that is selected, allowing the library infrastructure to effectively track and manage multiple copies of a single book.

Located on the book copies form are two buttons next to text input boxes for barcode and RFID entry. The user can enter these codes manually through the text-boxes, or by clicking the buttons. By clicking the buttons the user will be prompted to scan in the barcode or RFID using a physical barcode scanner or RFID reader.

Elevated users can also add a new book via the “Add Book” option in the “Books” submenu of the top menustrip, or by clicking the “Add Book” button in the bottom left of the dataviewer in the main form. Both of these will be invisible to Customers and regular users as they do not possess the authorization to create new data entries. These will open a blank version of the view details form discussed above with editing enabled by default. A button will also be made visible below the book-cover picturebox that will enable the user to select a jpeg copy of the new book’s cover. Finally in the top left of the form, the user can select the submit button in the top left corner of the window to submit the new book copy to the database.

6.1.1.3 Customer and User Management

The user can access customer and user forms via the same methods discussed earlier, for both “Update/View forms and “Add New Record” forms as depicted in Figures 7 and 8 of 6.1.2. When the user opens a customer as detailed in Figure 7 of 6.1.2, the user will be displayed a Tab-Page window, starting with the customer’s details which can be edited by selecting the “Enable Editing” checkbox at the bottom of the form and editing the relevant checkboxes. This includes the customer’s email address, which can only be updated by the user if both the customer’s email and confirmed email address are the same. From there the customer can navigate the other tab pages by selecting a tab from the list at the top of the

form, to view the books they have issued, their issue history, and the books they have on hold respectively.

When viewing the books a customer has issued, they will be able to see the books that are issued, due back soon, and overdue within the List View, highlighted in white, grey, and red respectively. This is indicated to the user within the key displayed on the right hand side of the Tab-Page. Within this section, the customer can also select a book, and click the "Renew" button to renew the book they have issued provided it is not overdue (i.e Highlighted in red), previously been renewed by the user, or currently has a hold on the book.

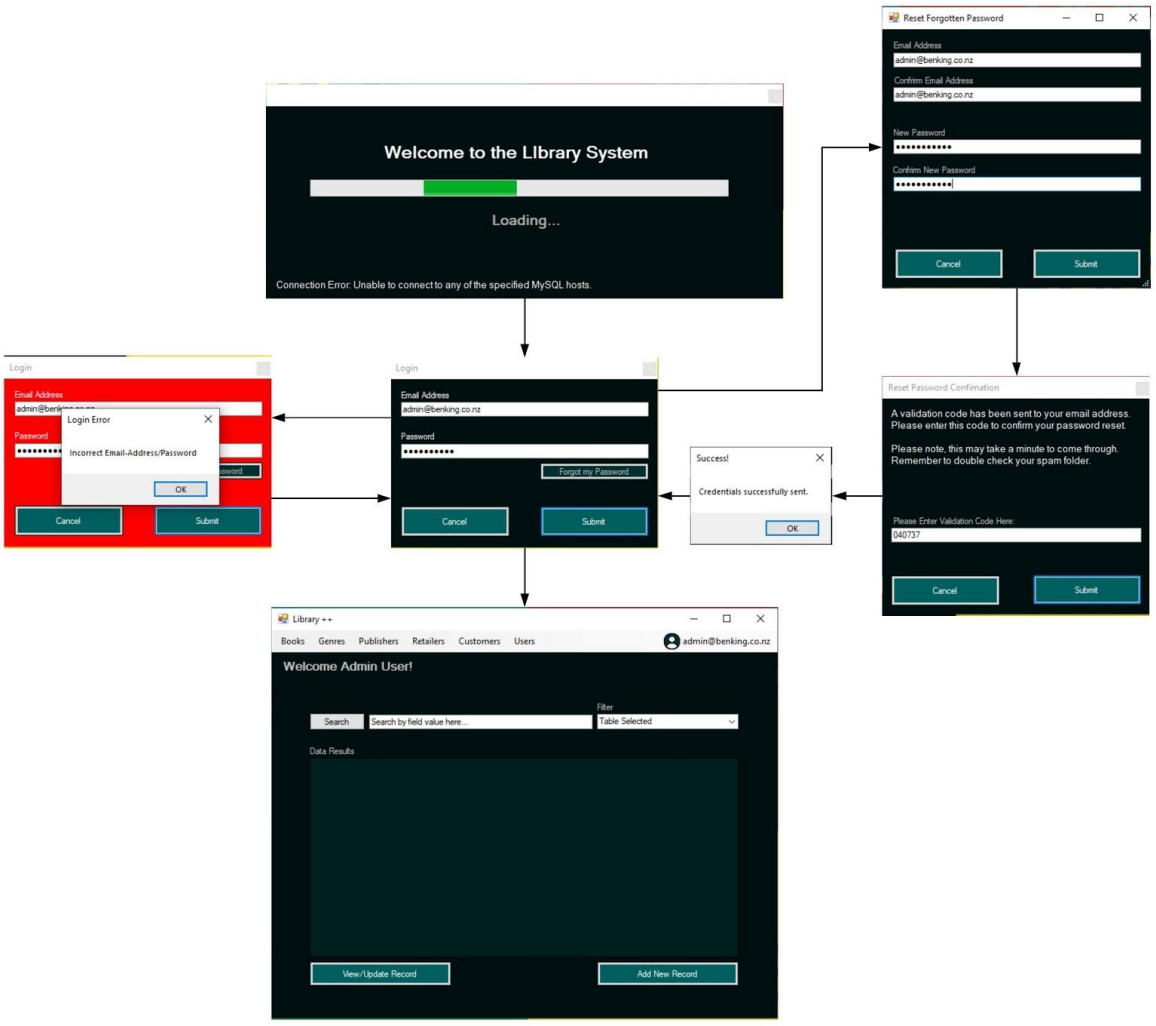
When adding a new customer, the Tab Pages will be confined to a single page with the customers main details enabled for editing. This is to limit confusion for the user, as a new customer will not yet have been assigned any issued books or have any books on hold. Once the customer has been created, a user account will be partially generated using these details and a new window will be opened for the current user to finish generating the customer's user account.

6.1.1.4 Logged in Users and Current User Details

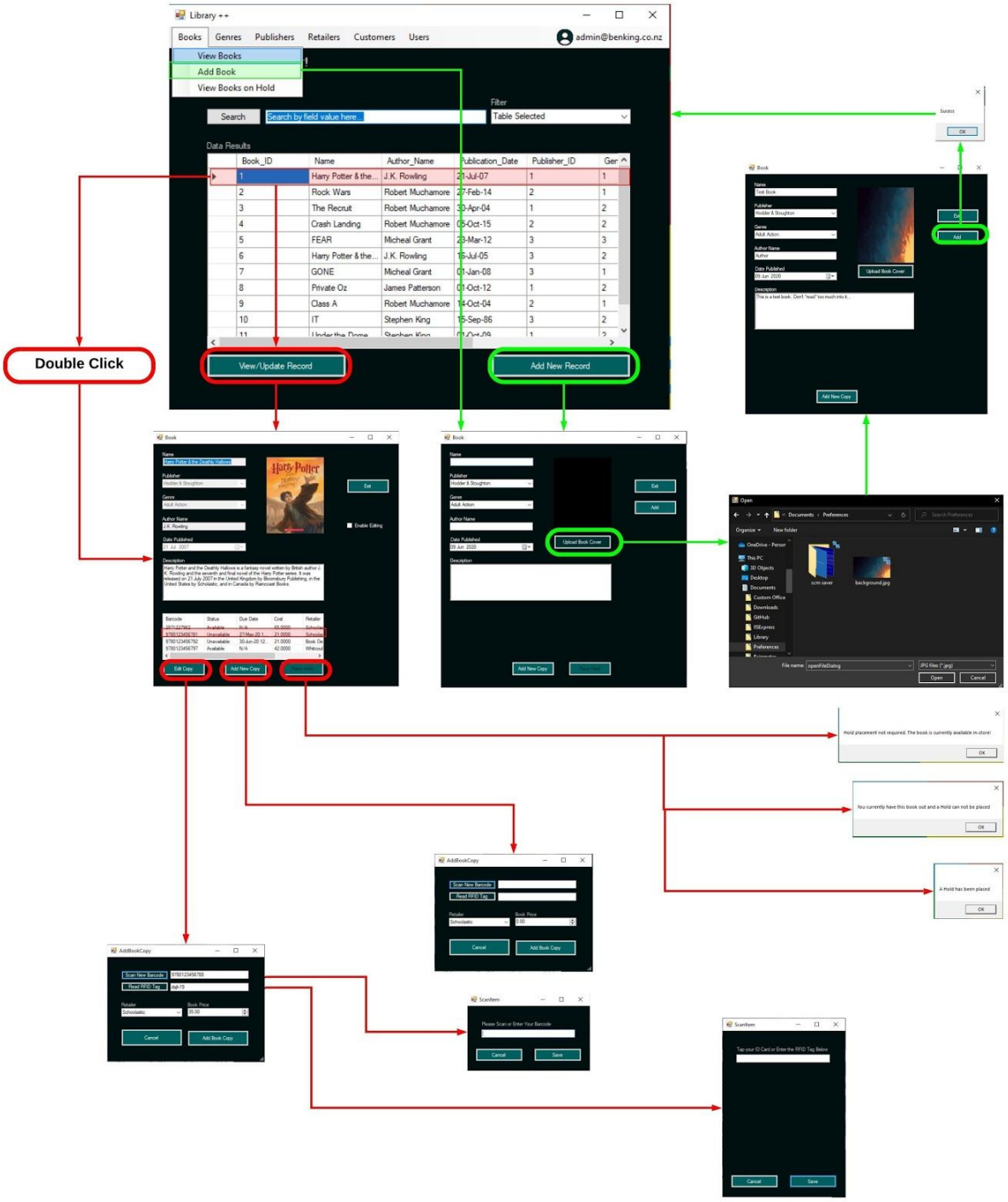
When the user is logged in, the user's email address will be displayed next to the user icon in the menustrip of the top right corner. As depicted in **Figure 6 of 6.2.1**, when the user clicks this component of the Menu Strip, the user will be given three options in the resulting submenu. This gives the user access to view and edit their own customer/user details (customer details if they are a normal user, and user details if they are a Librarian/Admin), the option to reset their password using the same process as outlined in **Figure 1, 6.1.2**, and the option to log out of the account.

Should the user log out of the account, the current-user held within the application will be discarded, and all GUI components will be disabled except for the login user option in the top right corner. This is to ensure that only users who are logged in have access to the application's features. This disabled main-form view is also the same screen that users who do not log into the application see on startup. Selecting the current user details in this instance will prompt the user with the same login process previously discussed in 6.1.1.1 and Figure 1 of 6.1.2.

6.1.2 Screen images

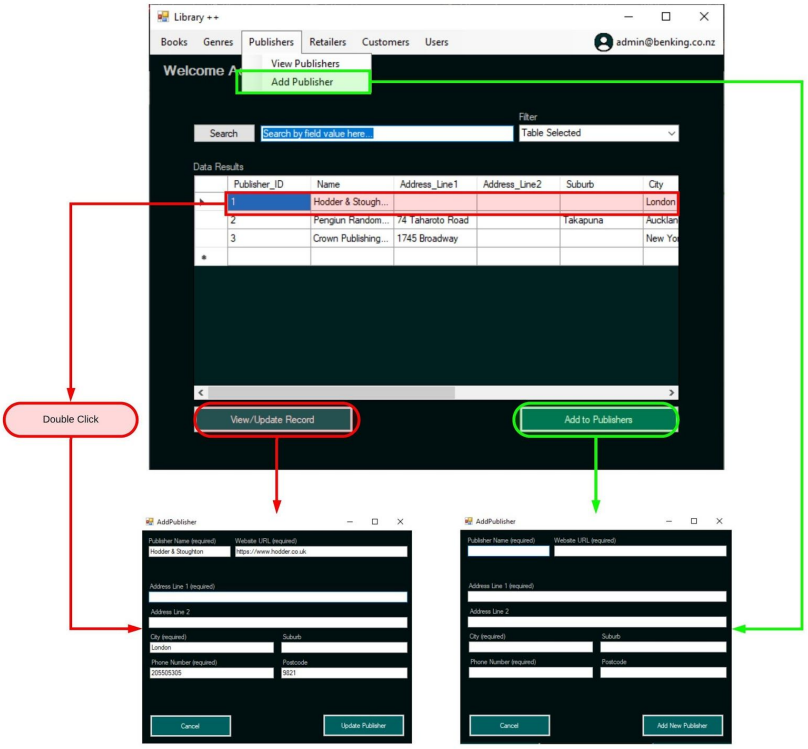


(Figure 1)

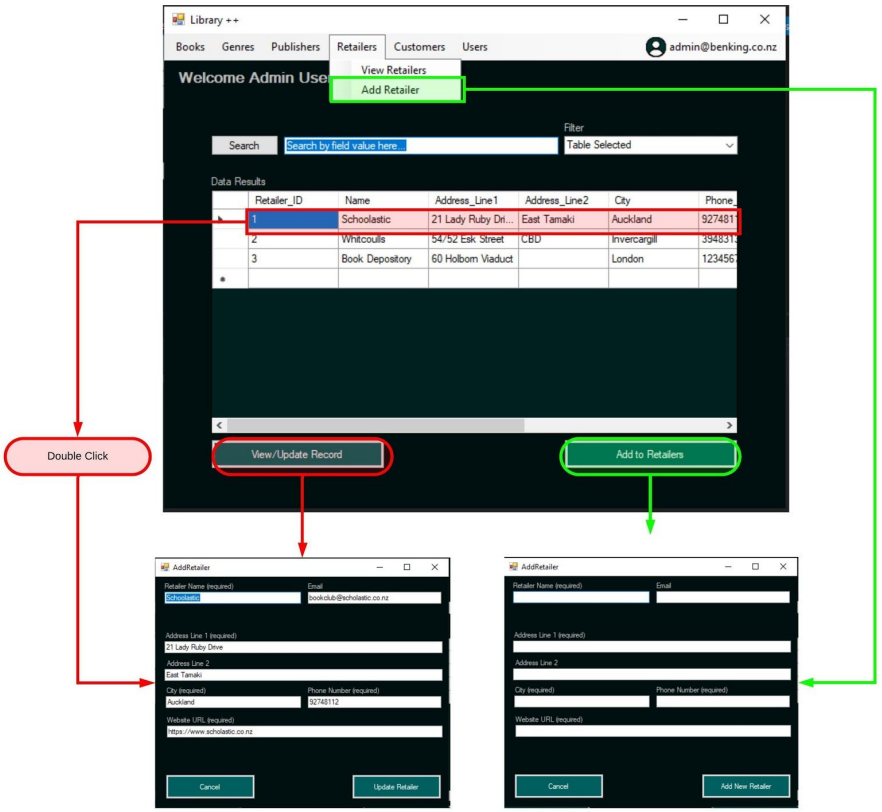

Library ++
Books
Genres
Publishers
Retailers
Customers
Users
admin@benking.co.nz
View Books
Add Book
View Books on Hold
Search
Search by field value here
Filter
Table Selected
Data Results
Book_ID
Name
Author_Name
Publication_Date
Publisher_ID
Double Click
View/Update Record
Add New Record

(Figure 2)

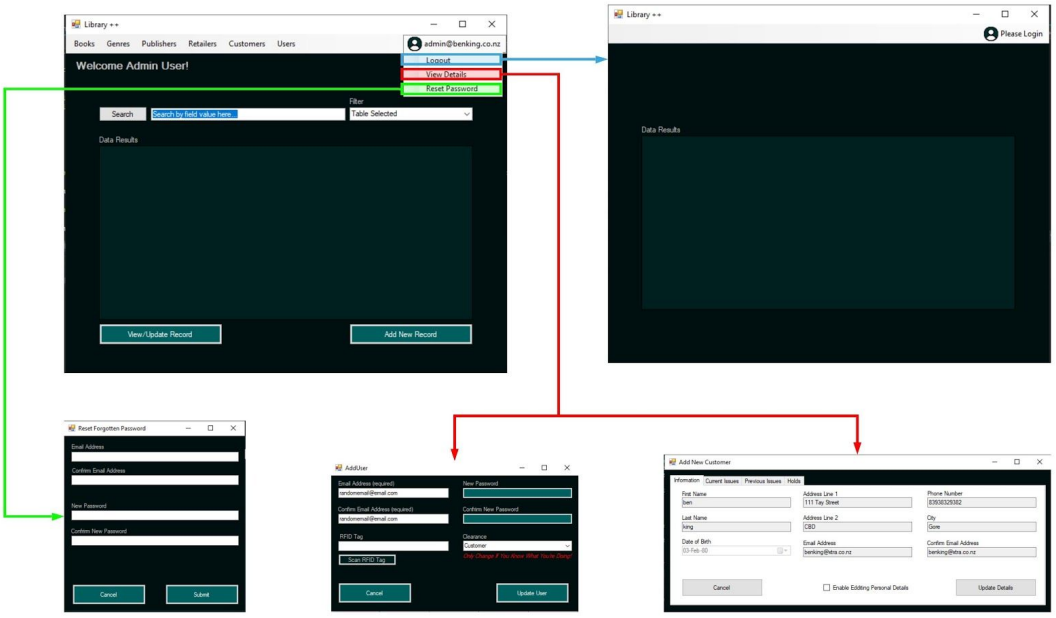
(Figure 3)



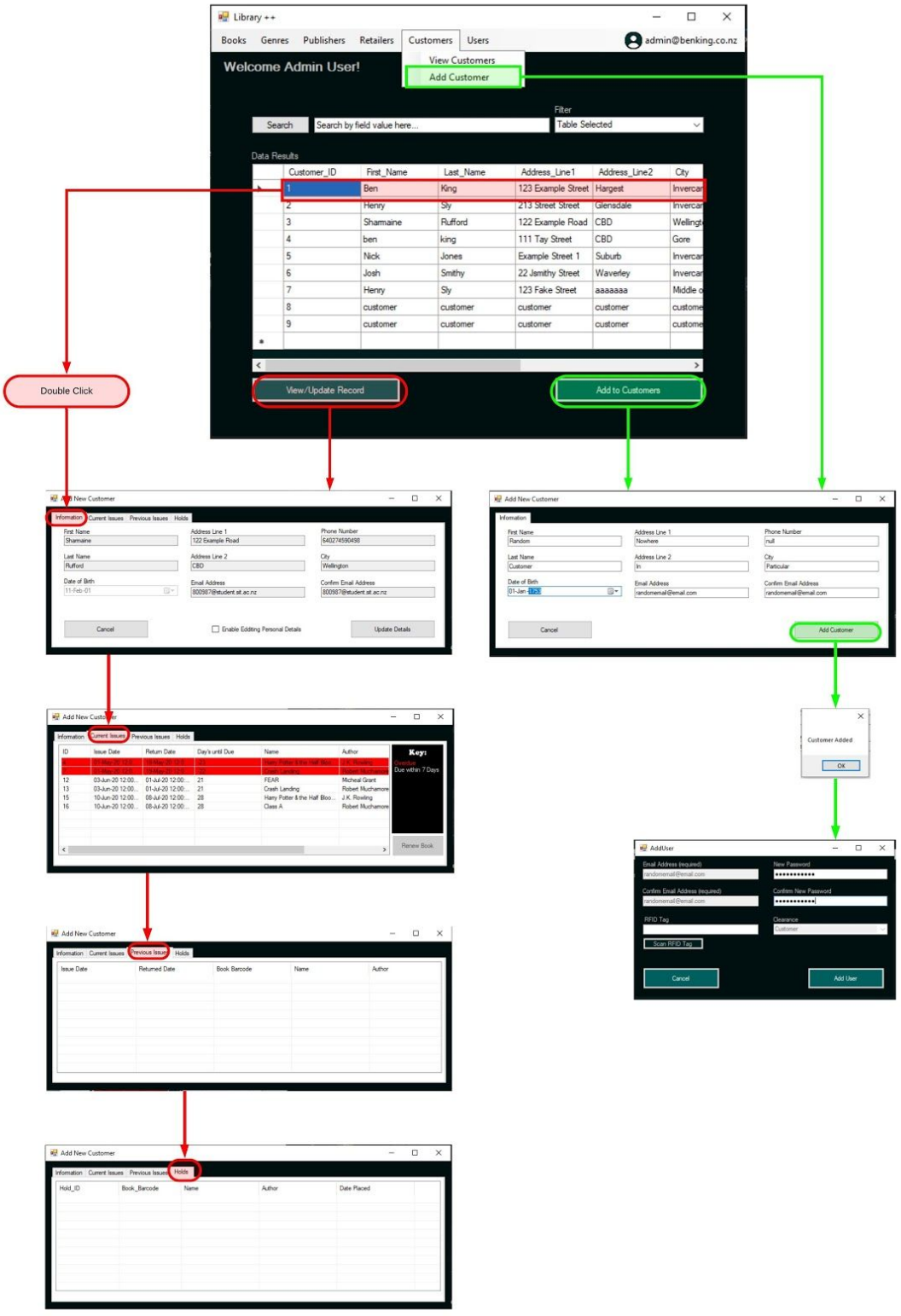
(Figure 4)



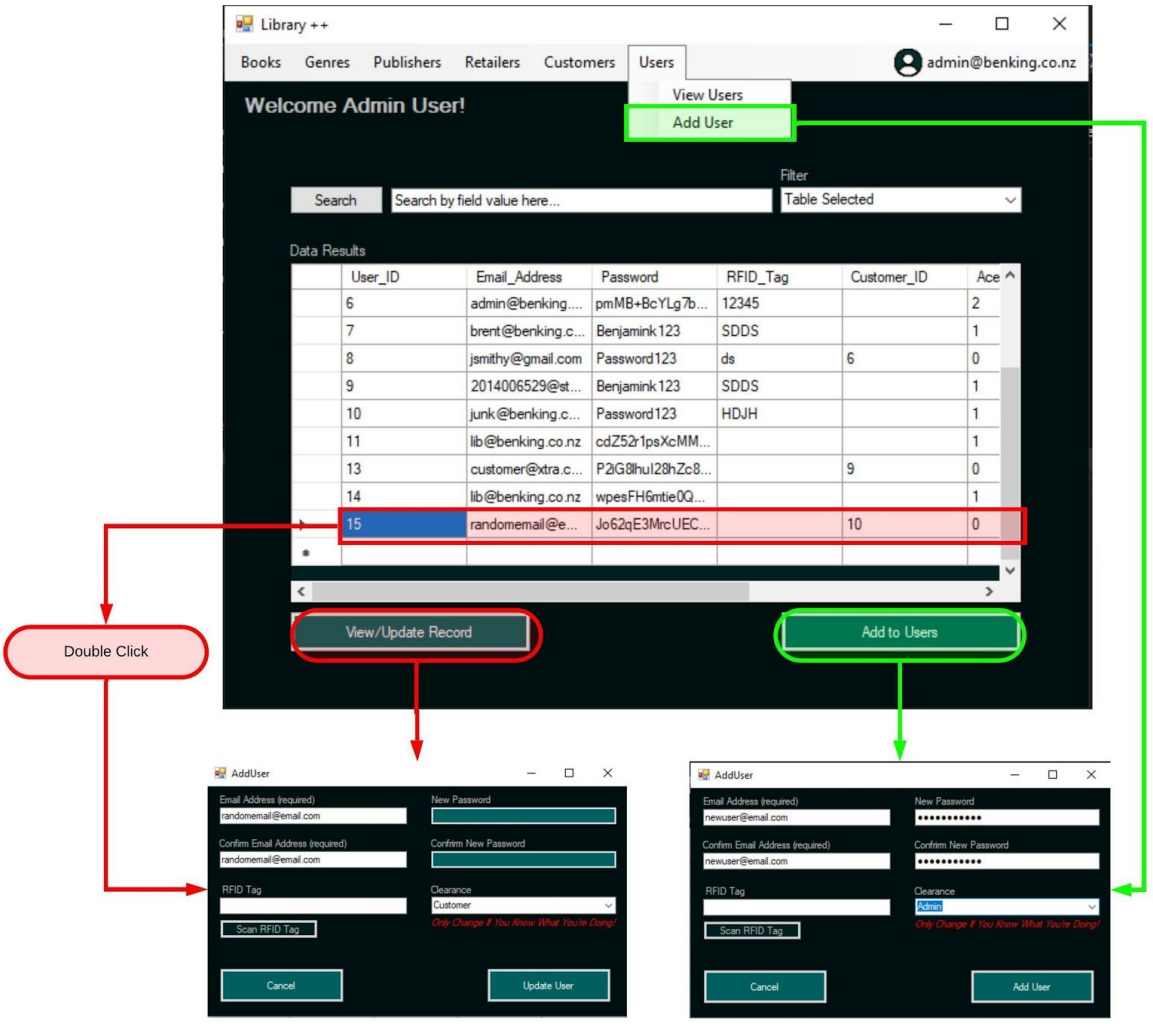
(Figure 5)



(Figure 6)


Library ++
Books
Genres
Publishers
Retailers
Customers
Users
admin@benking.co.nz
View Customers
Add Customer
Welcome Admin User!
Search
Search by field value here...
Filter
Table Selected
Data Results
Customer_ID
First_Name
Last_Name
Address_Line1
Address_Line2
City
Double Click
View/Update Record
Add to Customers
Add New Customer
Information
Current Issues
Previous Issues
Holds
Cancel
Update Details
Add Customer
Customer Added
OK
Key:
Due within 7 Days
Renew Book
AddUser
Scan RFID Tag
Add User

(Figure 7)


Library ++
Books
Genres
Publishers
Retailers
Customers
Users
admin@benking.co.nz
View Users
Add User
Welcome Admin User!
Filter
Search
Search by field value here...
Table Selected
Data Results
User_ID
Email_Address
Password
RFID_Tag
Customer_ID
Ace
6 admin@benking.... pmMB+BcYLg7b... 12345 2
7 brent@benking.c... Benjamink123 SDDS 1
8 jsmithy@gmail.com Password123 ds 6 0
9 2014006529@st... Benjamink123 SDDS 1
10 junk@benking.c... Password123 HDJH 1
11 lib@benking.co.nz cdZ52r1psXcMM... 1
13 customer@xtra.c... P2lG8lhuI28hZc8... 9 0
14 lib@benking.co.nz wpesFH6mtie0Q... 1
15 randomemail@e... Jo62qE3MrcUEC... 10 0
Double Click
View/Update Record
Add to Users
AddUser
Email Address (required)
randomemail@email.com
Confirm Email Address (required)
randomemail@email.com
New Password
Confirm New Password
RFID Tag
Clearance
Customer
Only Change If You Know What You're Doing!
Scan RFID Tag
Cancel
Update User
AddUser
Email Address (required)
newuser@email.com
Confirm Email Address (required)
newuser@email.com
New Password
Confirm New Password
RFID Tag
Clearance
Admin
Only Change If You Know What You're Doing!
Scan RFID Tag
Cancel
Add User

(Figure 8)

6.2 Interface design rules

When undertaking the development and design of the Library application, we undertook several interface design rules and philosophies in order to maximise user experience and ensure overall human-application functionality.

6.2.1 Clarity and Consistency are Key

One of the main interface design rules taken into consideration during development was that the user interface had to be both clear and consistent. This meant the use of consistent colorization, font-size, spacing, and asset placement wherever possible to ensure the experience is fluid and intuitive for users and customers. This also meant limiting the amount of visual clutter, adopting the “less is more” mentality discussed in 2.3.1, so that users would not be overwhelmed by the amount of content we were trying to disseminate.

6.2.2 Navigation Must be Easy and Intuitive

Given the scale of the content and information we were aiming to provide the Library application users, it became very apparent that UI navigation and accessibility would play a major role in our UI design. As such we decided to streamline this process by utilizing a mix of menu, forms, and object based interaction. This baseline consideration also encouraged us to provide multiple points of entry for UI functionality. E.g: Most forms are accessible via object buttons in relevant forms and UI locations as well as the top menu strip in the main form.

6.2.3 Application must be Easily Understood

The Library application is intended to have a low barrier of entry for users, both familiar with, and new to existing Library management systems. This means that the functional interface components have to be easily understood. This necessitates the use of clear and accurate labeling, clear and visible points of interaction, and the use of extensive UI affordances (e.g: buttons, input boxes, menu strips) supported by clear and legible UI signifiers (e.g: labels, titles, and instructions).

6.2.4 Content Mapping and Element Placement Must be Effective

Due to the large number of focal points the end user will be exposed to within the Library application, it is vital that visual spacing and proximity are utilized effectively within the application’s UI design. This includes maximising the functionality of blank space and object clustering to afford the user functional similarities and differences, and implementation of sub-forms to reduce screen cluttering. This can also be utilized to draw user attention to important focal points and assist in intuitive UI navigation.

6.2.5 Information Must Be Effectively Communicated

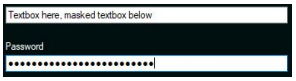
Any information communicated by the application UI in the form of text and signifiers must be easy to read and understand. This means that text must have a high enough contrast and font-size to be legible. Text must also be kept to a consistent format, breaking from that format only for the purpose of highlighting vital information to prevent JaRRiNg the user.

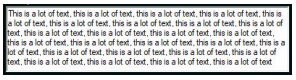
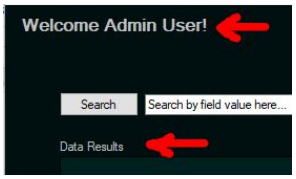
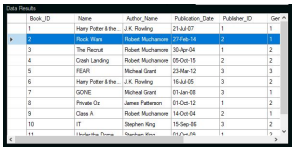
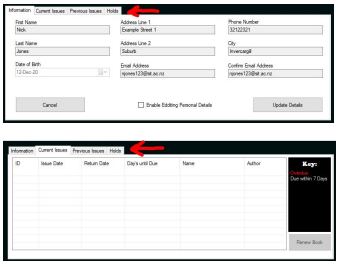
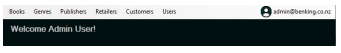
6.2.6 General Formatting Must Be Unified

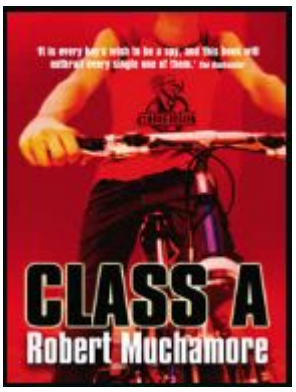
As a general rule to achieve design consistency and improve the rate of user understanding, objects that afford a priority action such as submit and update buttons will be placed on the right hand side, and secondary actions such as cancel buttons will be placed on the left. Optional buttons will be given a darker shade of colour to submit and cancel buttons as these incite less immediacy to the user.

6.3 Graphical User Interface Components and Considerations

There are a large range of Graphical User Interface (GUI) components that are utilized within the Library application. In the table below, we discuss the GUI components that were used, and the purposes for which they were utilized.

Currently Used GUI Components		
Component	Image	Purpose
Buttons	 	Buttons are small distinct components that can be clicked on or “pressed” to initiate an action. They were elected as a primary choice for user interaction based on user familiarity and ease of implementation.
Combo Boxes		Combo boxes are dropdown menus that supply a list of predefined inputs. These proved ideal in scenarios where we wanted to limit user entry to a handful of controlled values, such as table and access-level selection.
Text-Boxes and Masked Text-Boxes		Text boxes are the most basic GUI components implemented in the application. They are used regularly to present user defined inputs and data.

Rich-Text Boxes		We have used several rich text boxes within our application as these allow a larger amount of text to be inputted into our application.
Labels		Labels are used within our applications to provide visual aids to users of what information should be entered in the Text Boxes.
Data Viewers		Data viewers are graphical components designed to present raw data to users in a way that is easy to understand. This proved extremely helpful when providing the user high volumes of information from within the user interface.
List Views		List Views are used throughout our applications due to how this allows the user to see the information from the database inside a table. This easily displays information for more than one item such as the results of a search query.
Tab Pages		Tab pages are a type of menu display that allows users to switch between views or “tabs” within a set amount of space, similar to having multiple tabs open in one's browser. This was useful when dealing with the customer page, where a large amount of loosely-interconnected information had to be displayed within a confined area.
Menu Strips		Menus are collections of various other GUI components clustered together to condense navigation and improve accessibility to functionality. We decided to implement a menu-strip at the top of our application for this purpose.

Picturebox		We have used pictureboxes within our application with how these allow the images (such as the cover page of a book) to be displayed, presenting visual content to the user.

When considering the future of the application, there is a significant amount of room to expand and improve the application's existing GUI components, as well as implement additional GUI features to improve user interactions and experience. An example of this would be the expansion of the Menu-Strip and Tab-Page menus already existing in the application to include a wider range of features for user accessability. This can also include the introduction of application specific context menus to better increase functionality for users more used to using the right click button.

6.4 Development system description

To generate and design the Library's user interface, the team utilized the form building tools available within Microsoft Visual Studio 2019. This allowed us to develop clean looking UI rapidly, as the development system is fully implemented within our existing development suite. This offered a distinct advantage over other user integrated development systems (e.g: generating UI elements and assets via photoshop), as this offered us clear insight as to how the software will look as a finished product while functional development was taking place. This also assisted in creating a greater degree of design parity as the development system was readily available and could be uniformly used by the team.

To develop and generate the SQL Scripts needed to make the MySQL Database, Visual Studio Code was used as the Integrated Development Software. Exploring and adding manual entries within the database tables were done through SQLYog, allowing easy insertion and viewing of the tables.

FileZilla was also used to check and perform testing, that uploaded Book Covers were being saved intact onto the server, without any corruption and were being appropriately named with the relevant filename.

7.0 Evaluation

When we first conceptualised the idea of creating a Library System, we weren't originally too sure of what would work or the specifics of application that would be needed to be created. However over the last 15 weeks we have expanded on our original conceptualised idea, creating 3 fully functional applications and a MySQL Database; where the information is selected from, and inserted into.

Originally we conceived the idea of creating a simple C# Form based application which would act to the user as a front end user interface to an SQL database that we would develop. We originally conceived the idea that it would be a simple application and database, which would require minimalistic coding and would reuse forms, simplifying the developing process. However we have expanded on this application, while ensuring that most of the effort we did focused on this application, our *Rent a Leaf* application.

Using knowledge from other classes, we have successfully incorporated RFID Readers (from our Human Computer Interaction Class - IT705) allowing the user to identify themselves and books by scanning their ID Cards/RFID Chips. We have also used our knowledge and understanding from our last year Data Management Class (ITC607) to create sophisticated SQL queries consisting of subqueries to enhance this application and retrieve information from the MySQL Database. By connecting and linking our application to a MySQL Database, we used the knowledge gained from our last year Advance Programming Class (IT607).

Despite having a pandemic and mandatory lockdown occurring during this semester's class, we have used the extra time and concentration that was a side effect of this to research and add additional features and abilities to the application; such as researching how to store the passwords securely in our database (Hashing them).

Other external learning that has helped us enhance our application to add and improve new features includes the ability of incorporating a 3rd Party External Library and Service **Twilio** to send automated Reminder SMS Messages to customers, and providing the ability to also send email messages using **Mailkit**, an External Library using our SMTP Provider. We have also researched and incorporated additional features such as creating an ical event, which automatically is generated and sent by the Automatic Emails application when it sends reminder emails, allowing the customer to add the reminder notice to their calendar. We also decided to use a dedicated File Transfer Protocol Server (FTP Server) for the Book Cover Images, requiring extensive research of how to upload and download images from the Server.

Having to submit the 3 assignments for this paper, allowed us to use our knowledge from our previous years of the course, especially the System Analysis Design (ITC 604) class. This allowed us to gain a deeper understanding of how the diagrams we have been taught relate to each other and are important factors and components in the designing and development of a system. Using Component, Class and ERD Diagrams allowed us to see and understand a deeper understanding of the system.

For all of us, this has been the best (and funest) project to work on and develop, with the lectures from class corresponding directly to the creation of this project and its documentation.

We have managed to keep well within our timeframe originally set at the start of the project. We allowed extra time for study (e.g. new abilities), illness, and other commitments. As we kept to this strict schedule we have been able to add extra value to the project which include extra features. We collect information for future development options if The Library wishes to add them *see* **8.0**.

However now, on completion of the project, all of us are extremely proud of the Library System we have created. We believe it is one of a kind, and are all extremely proud of the finished product and have enjoyed working as a team to develop it.

Notable shoutouts:

Ben - Twilio (SMS Messages), MailKit (Email Messages), Self Checkout Application, iCal File, core frameworks of the applications and SQL Database.

Henry - Hashing (security of Passwords), core programmer of *Rent a Leaf* and designing the User Interface

Sharmaine - Creation and documentation of all the assignments and finalised Project Documentation, creation of a lot of the diagrams required for the assignments.

Favourite Moment of the Project:

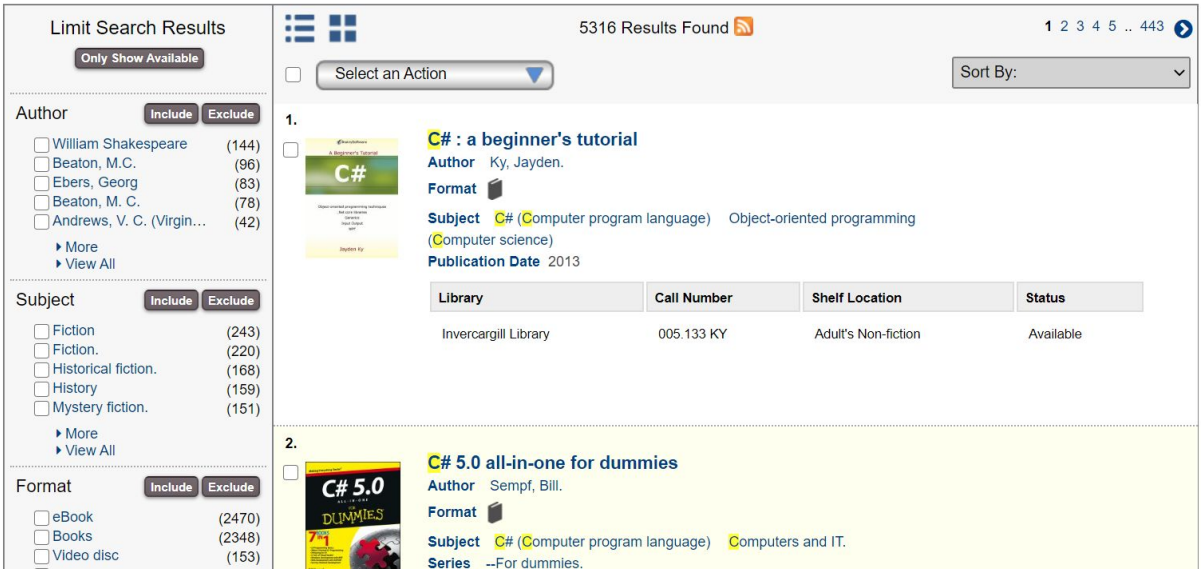
Ben	Being able to develop and use knowledge from other classes and external sources especially getting the Twilio Text System to work!
Henry	One of the key highlights for me on the project was the thorough communication and coordination we had as a team. I also really enjoyed the research and development components of the project, learning about effective user and security management and how to implement this

	into our Library system.
Sharmaine	Working so well as a team and completing the projects required within the timeframe set. The motivation was not lost because of the pandemic and we managed to complete a lot at home while this was on. I also enjoyed making the diagrams and seeing them in use.

8.0 Future Developments

8.1 Website

To improve and modernise this system a possible future development would be converting this C# form based application to an ASP.Net MVC application, easily allowing users to see and perform functions through a website. This would allow easier facilitation of Authentication, easily allowing customers to create an account and using External Logins to login to the system. It would also allow people to easily see and perform functions without needing to download and install the Rent a Leaf Application. Search Features could be implemented allowing the user to search for a book and see results such as the below screenshot from the Invercargill City Libraries Catalogue Website:



(Figure 9 - Invercargill City Libraries Online)

More extensive filters could be applied, allowing users to select books from multiple authors and the ability to easily click the author's name to search for books by them. However due to time constraints and technical ability constraints we have not implemented this.

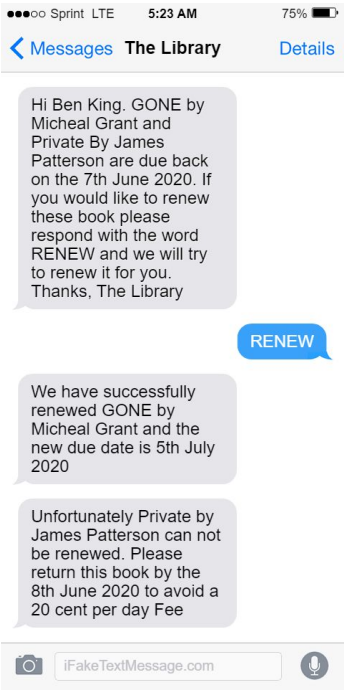
8.2 Security

To also improve the security of our application another possible future development could be to allow users to login and have the ability of using 2 Factor Authentication, requiring the user to enter a generated number to authenticate their identity. This number could be generated and sent via an SMS Message, or by using an Time Based Authenticator App (which the account has been activated in the App through a QR Code). While we have set this sort of up in a similar method with the Reset Password ability (via Email confirmation key), this could be

expanded to logins through an App or SMS. (Figure 1 - Screenshot of Authy 2 Factor Authentication).

8.3 Enhanced Reminder Texts

Another future development that could also be implemented is the ability for customers to respond to text and email messages sent to them, allowing the system to perform functions. This could be as simple that if a Reminder Text is sent to the customer, they could respond with the word **RENEW** and the system on receipt of their SMS Message would try to renew their books for them. The responding text to the message could specify which books were renewed, and which books were unable to be renewed, such as the image to the right. (Figure 2 - Screenshot of Possible Expansion, Simulated through https://ifaketextmessage.com/)



8.4 Increased Testing and Debugging

To improve the integrity and security of this system, future developments could include more testing and debugging to ensure that all the bugs and errors in our application have been resolved. While we have performed some testing, more rigorous testing would.

8.5 Mobile Application - Customer

Another future development that could improve and make this system greater would be the ability to link the system to a mobile application, allowing people to lookup books, place holds and see their current account status from a Mobile Application. The ability to incorporate push notification reminding users that books are due, or holds are available to pick up would improve the usability and accessibility of the system.

8.6 Mobile Application - User (Librarian/Admin)

A Mobile Application could also be developed for the Librarians or Admins of the *Rent a Leaf* System, easily allowing them to issue and return books as well as perform similar tasks on a Mobile Phone (or Tablet such as an iPad) to the Rent a Leaf Application. This could allow them to perform tasks and interact more one on one with customers compared with sitting at a Service Desk, or use computers around the Library. Books (& Customers) could be identified by using a device which supports the ability to read RFID/NFC Tags.

8.7 Cross Compatible Device Support

The Applications we have created (*Rent a Leaf*, *Snatch & Go, Automatic Emails*) could be further developed to support additional Operating Systems such as macOS or native Linux Support. This would allow the library to support a larger range of devices, allowing the library to upgrade devices regardless of operating system.

8.8 Enhanced Customer Identification

The Applications we have created could support enhanced customer identification, allowing customers to easily identify themselves through other identification methods. These other identification methods could be via Fingerprint, Facial Recognition or Biometric Identification (Blood/DNA Samples). This would increase security within our application, while also incorporating future technologies, where security and privacy of information is becoming more of an ongoing issue.

9.0 Conclusion

In conclusion, this report shows an in-depth software design description of The Library's systems consisting of the applications Rent a Leaf and Snatch and Go, that HSB Consulting has created. HSB Consulting has evaluated the Snatch and Go self-checkout system and found this system acts as an easy use option for the customer, with a clear and concise layout. Whereas, the Rent a Leaf system shows superior qualities for the librarian's use and detailed response with the software use. These two systems are linked through The Library's database system, which means the same information is duplicated on both systems. These two systems shall have enduring qualities to stand the test of today's technology.

As a Team we are **very proud** of our work, and our strong ability to work as a team, not encountering any personal or complicated team difficulties, but instead strengthening our team bond throughout the project.

10.0 References

10.1 External References Used in the development of our project

C# Language - PBKDF2 for Password Hashing: c# Tutorial. (n.d.). Retrieved June 17, 2020, from https://riptutorial.com/csharp/example/10258/pbkdf2-for-password-hashing

Guardado, D. M. (2019, October 4). PBKDF2 Hash a secure password. Retrieved June 17, 2020, from https://dev.to/demg_dev/pbkdf2-hash-a-secure-password-5f8l

Jstedfast. (2020, June 16). jstedfast/MailKit. Retrieved June 17, 2020, from https://github.com/jstedfast/MailKit

Karelz. (2018, June 26). How to: Download files with FTP. Retrieved June 17, 2020, from https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-download-files-with-ftp

Karelz. (2018, June 26). How to: Upload files with FTP. Retrieved from https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-upload-files-with-ftp

Rianjs. (2019, March 21). rianjs/ical.net. Retrieved June 17, 2020, from https://github.com/rianjs/ical.net

Twilio SDK for C# and .NET. (n.d.). Retrieved June 17, 2020, from https://www.twilio.com/docs/libraries/csharp-dotnet

As well as hundreds of Stackoverflow and Microsoft Forums when trying to diagnose issues and bugs

10.2 Tools used in the development process:

Lucid Charts for the creation of diagrams
http://lucidchart.com/

SQLYog Community for the visualisation of the MySQL Database
https://github.com/webyog/sqlyog-community/wiki/Downloads

Phidget Control Panel for RFID testing and debugging
https://www.phidgets.com/docs/Phidget_Control_Panel

Visual Studio Community 2019 for Application Development
https://visualstudio.microsoft.com/downloads/

Visual Studio Code for SQL Development
https://code.visualstudio.com/download

Application/SQL Hosting - free hosting provided by Project Manager (Ben) through his raspberry Pi, which was port forwarded to his Domain (http://benking.co.nz).