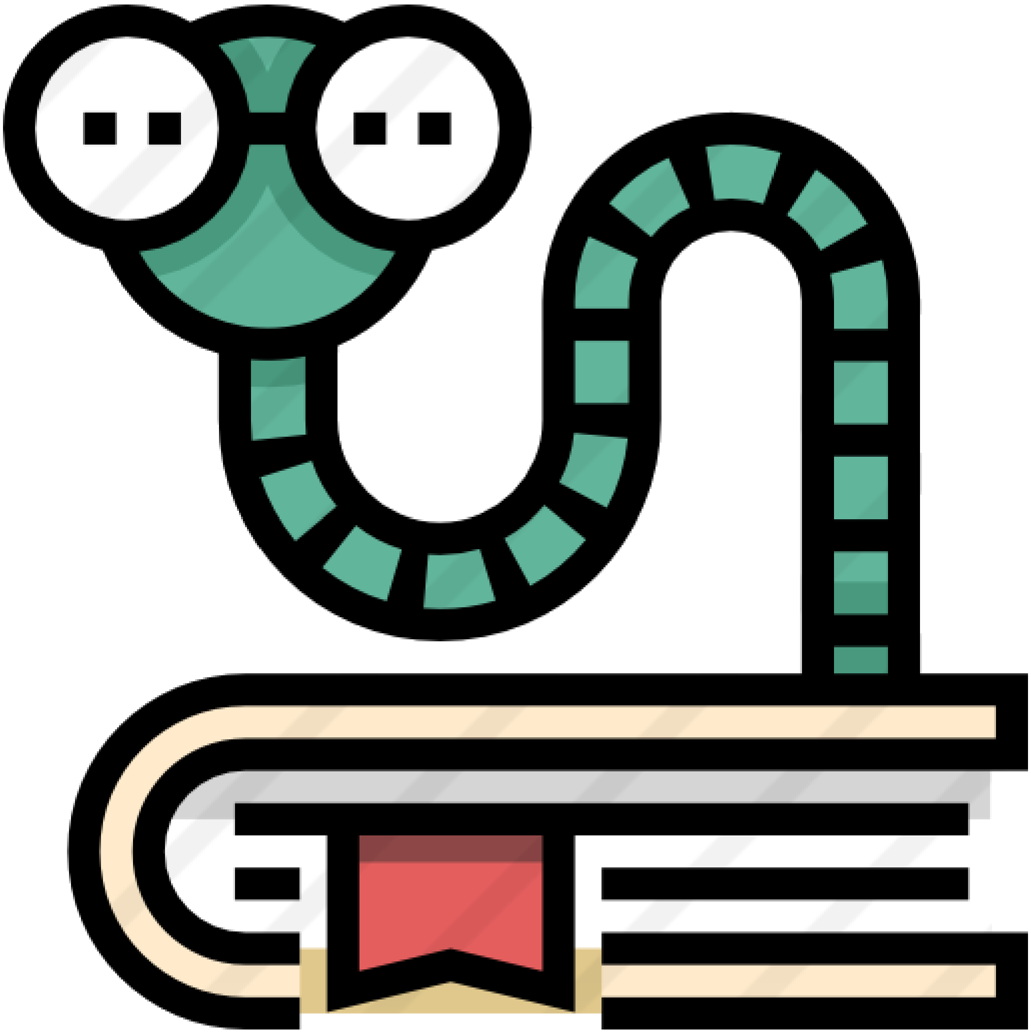


Project Bookworm

A Level Computer Science Project Development Write-up and Documentation

AQA A Level Computer Science Project Write Up

Analysis

Introduction

Hampshire County Council's School Library Service (SLS) loans books to over 400 schools in Hampshire and has four centres located in Basingstoke, Farnborough, Fareham and the New Forest. The SLS provides various services as well, including loaning books and e-resources, training and advisory/support services. The school's exchange books by visiting a centre or receiving delivery via a small fleet of vans. The standard allocation of books is three per pupil at primary level and two per pupil at secondary with a maximum loan of two hundred books per exchange.

Description of current system

Currently, the SLS only tracks the quantity of books at a given location using an Excel spreadsheet for each centre and one tab for each school. Each tab contains the following details about the school (see figure 1 below) as well as the details of each exchange including the number of books and the date. When a school visits a centre to exchange books, the library assistants at the centre physically count all of the books as they come in and go out and record this in the spreadsheet, which then calculates the total for each individual school.

The clients requirements were investigated by speaking with one of the library assistants as well as conducting a survey amongst the rest. I also went to a showcase day for library management systems aimed for internal use in a school library. These apply a similar principle to my project however they work on a larger scale in terms of volumes of users and books.

The main issue with the current system is that the library assistants have no way of knowing where books are located. Currently a school cannot request a list of books that they have since there is no tracking data available to look this up with so currently an accurate stock taking exercise be undertaken. Another issue is that one school is not supposed to have more than two copies of any book but this is difficult to manage since there is not a list of what books they have to check against.

General Details for a School (See Fig 1.)

Each school currently has its own tab in the spreadsheet. At the top of each tab is a header containing the following information:

- School Name
- Head Teachers Name
- Address
- Contact Details
- The DfE Number (unique number assigned by the Department for Education to each school)
- Date of the Last Exchange
- Number of Pupils

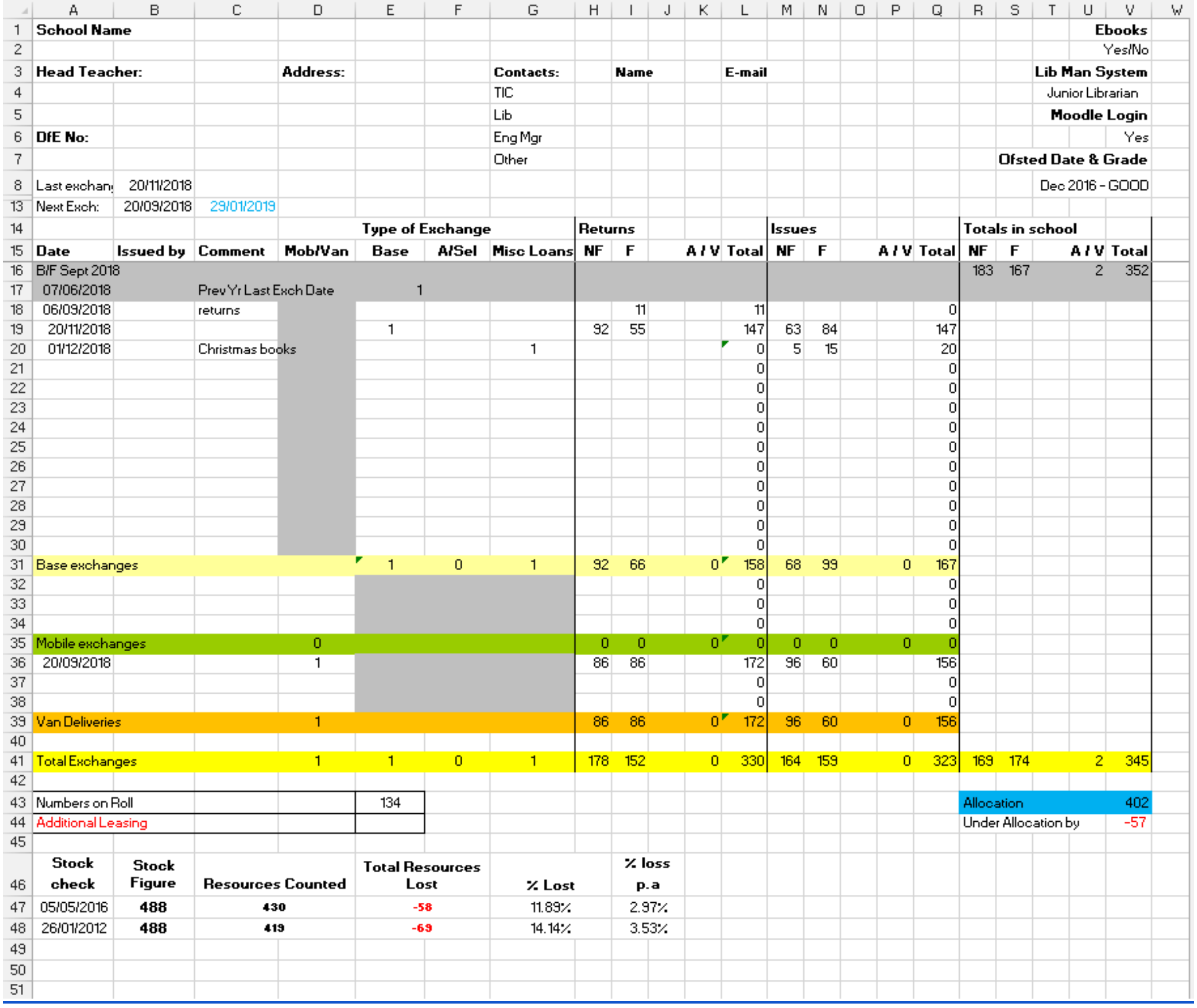
Below these details, there is a table of the exchanges and details of them including:

- Date
- Comments/Notes
- Type of exchange
- Number of books returned
- Number of books issued

At the bottom of the sheet, is a tracking table for stock check visits including:

- Date
- Stock they should have
- Stock counted
- Total items lost
- Percentage of loan that has been lost

Fig 1. Screenshot of current system spreadsheet example for a school.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	School Name																					Ebooks	
2																						Yes/No	
3	Head Teacher:			Address:			Contacts:		Name			E-mail										Lib Man System	
4							TIC															Junior Librarian	
5							Lib															Moodle Login	
6	DfE No:						Eng Mgr															Yes	
7							Other															Ofsted Date & Grade	
8	Last exchan	20/11/2018																				Dec 2016 - GOOD	
13	Next Exch:	20/09/2018	29/01/2019																				
14					Type of Exchange			Returns					Issues					Totals in school					
15	Date	Issued by	Comment	Mob/Van	Base	A/Sel	Misc Loans	NF	F		A / V	Total	NF	F		A / V	Total	NF	F		A / V	Total	
16	B/F Sept 2018																	183	167		2	352	
17	07/06/2018		Prev Yr Last Exch Date		1																		
18	06/09/2018		returns						11			11					0						
19	20/11/2018				1			92	55			147	63	84			147						
20	01/12/2018		Christmas books				1					0	5	15			20						
21												0					0						
22												0					0						
23												0					0						
24												0					0						
25												0					0						
26												0					0						
27												0					0						
28												0					0						
29												0					0						
30												0					0						
31	Base exchanges				1	0	1	92	66		0	158	68	99		0	167						
32												0					0						
33												0					0						
34												0					0						
35	Mobile exchanges			0				0	0		0	0	0	0		0	0						
36	20/09/2018			1				86	86			172	96	60			156						
37												0					0						
38												0					0						
39	Van Deliveries			1				86	86		0	172	96	60		0	156						
40																							
41	Total Exchanges			1	1	0	1	178	152		0	330	164	159		0	323	169	174		2	345	
42																							
43	Numbers on Roll				134													Allocation				402	
44	Additional Leasing																	Under Allocation by				-57	
45																							
46	Stock check	Stock Figure	Resources Counted		Total Resources Lost		% Lost		% loss p.a														
47	05/05/2016	488	430		-58		11.89%		2.97%														
48	26/01/2012	488	419		-69		14.14%		3.53%														
49																							
50																							
51																							

Data collection and research:

Identification of End Users:

The users would be the librarians at the Hampshire School Library Service who could use it to manage the loans easier.

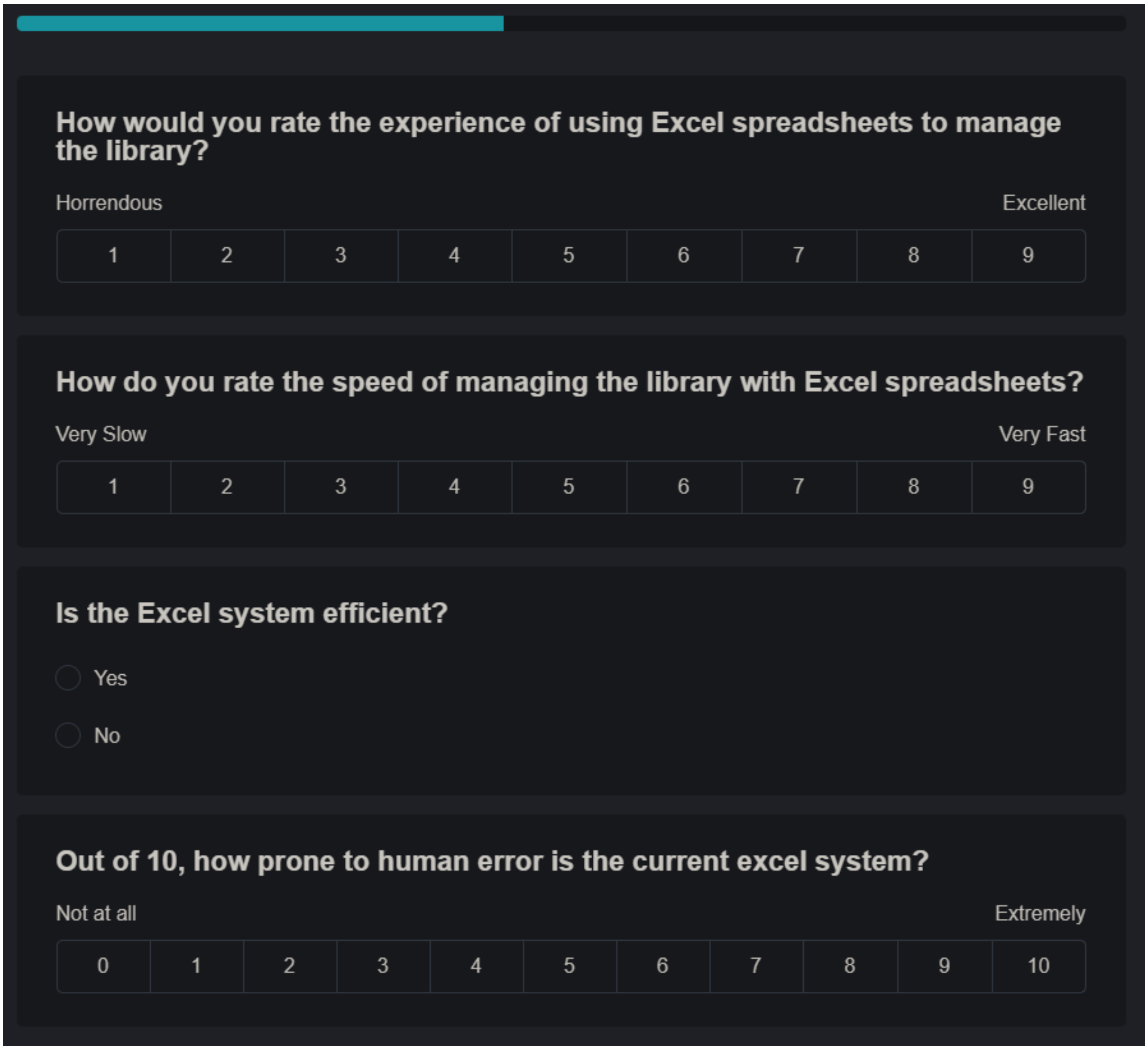
Questionnaire:

In order to gauge the needs of the users and therefore the requirements of the system, I created an online survey that could be passed around to the SLS library assistants to easily collect the data needed. I then used that data to form the Key Objectives and Additional Objectives that my system should meet.

In order to create the survey, I used a free online survey creator called Enalyzer

Questions about the current system:

Fig 2. Question about respondents interest in a new system.



How would you rate the experience of using Excel spreadsheets to manage the library?

Horrendous — Excellent

1	2	3	4	5	6	7	8	9

How do you rate the speed of managing the library with Excel spreadsheets?

Very Slow — Very Fast

1	2	3	4	5	6	7	8	9

Is the Excel system efficient?

○ Yes

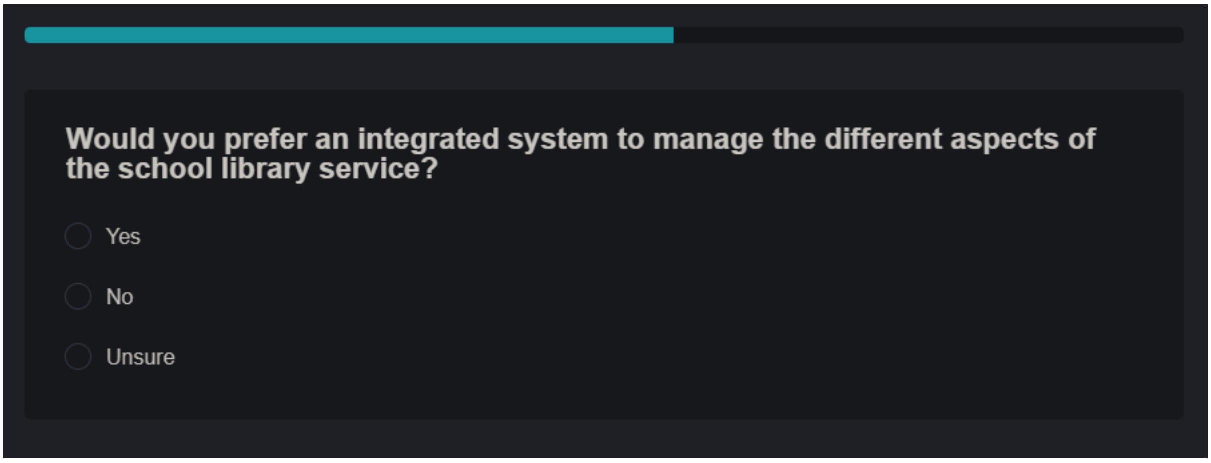
○ No

Out of 10, how prone to human error is the current excel system?

Not at all — Extremely

0	1	2	3	4	5	6	7	8	9	10

Fig 3. Questions about desired features in the new system.



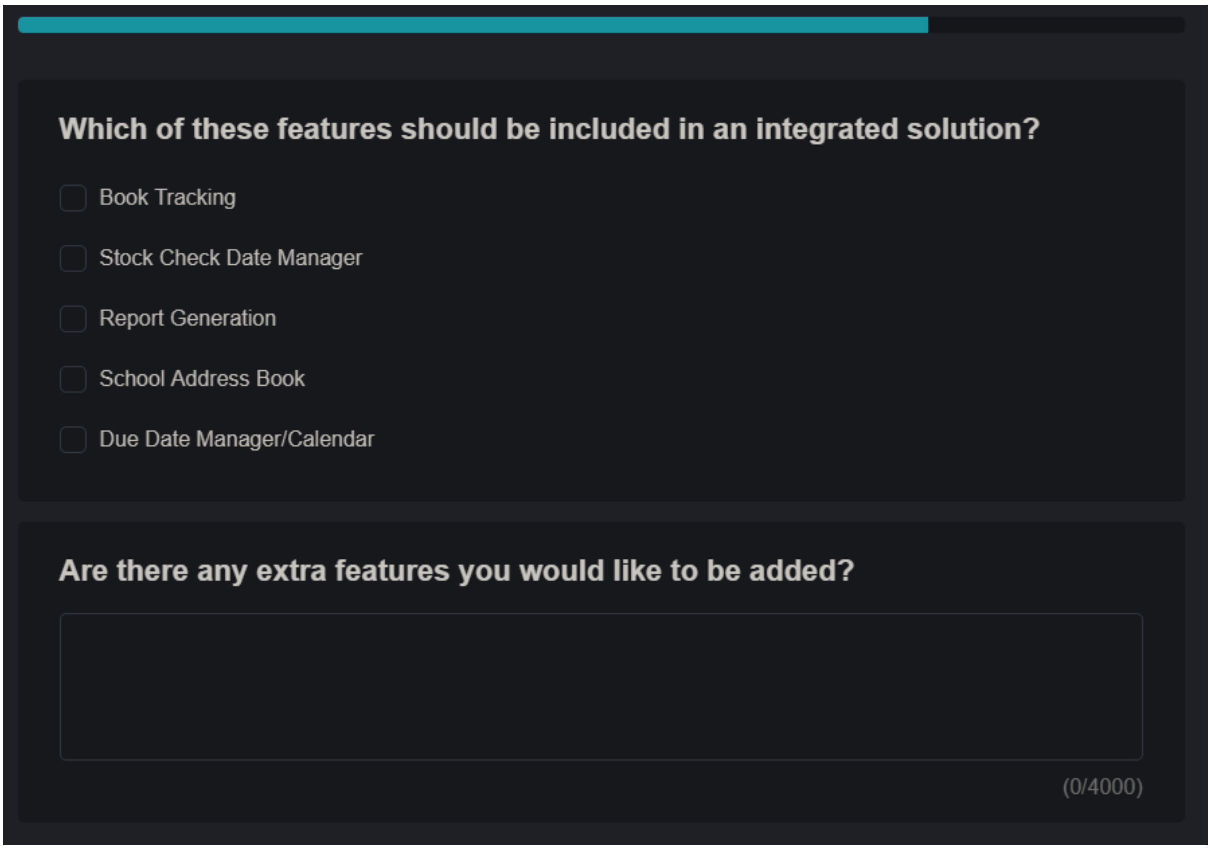
Would you prefer an integrated system to manage the different aspects of the school library service?

○ Yes

○ No

○ Unsure

Fig 4. Questions about a report creation feature.



Which of these features should be included in an integrated solution?

□ Book Tracking

□ Stock Check Date Manager

□ Report Generation

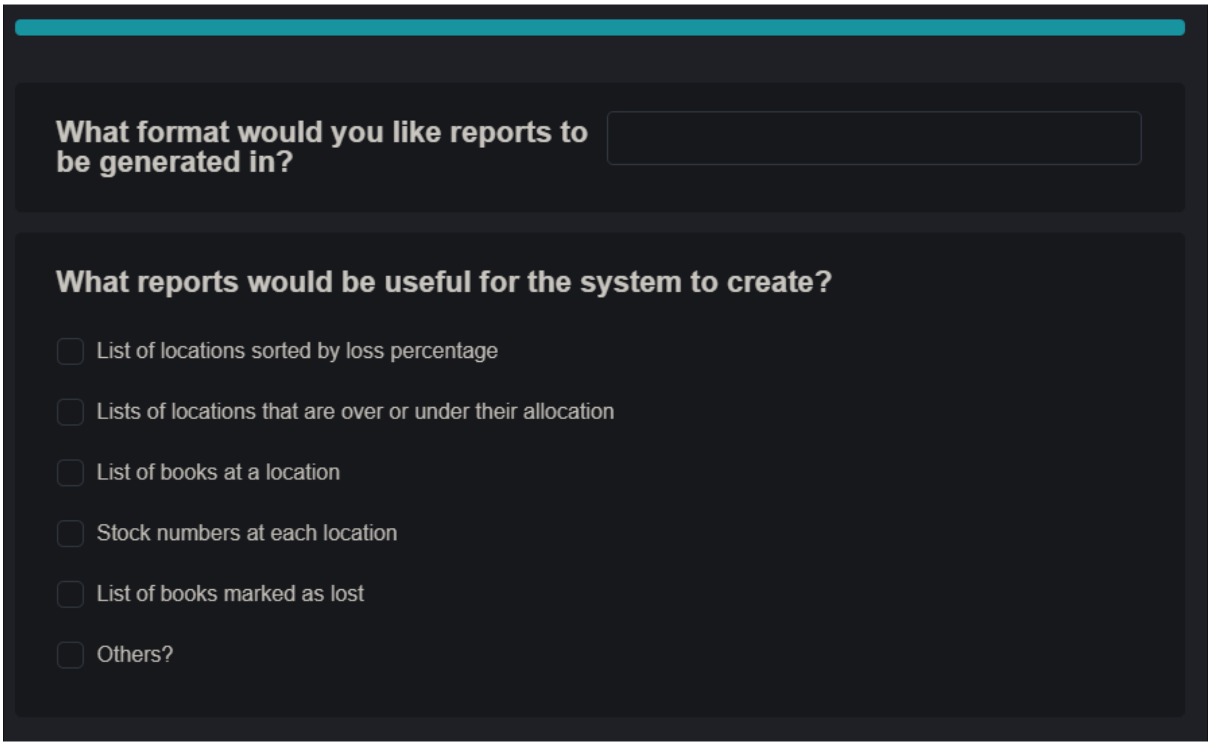
□ School Address Book

□ Due Date Manager/Calendar

Are there any extra features you would like to be added?

(0/4000)

Fig 5. Questions regarding reports being generated by the system.



What format would you like reports to be generated in?

What reports would be useful for the system to create?

□ List of locations sorted by loss percentage

□ Lists of locations that are over or under their allocation

□ List of books at a location

□ Stock numbers at each location

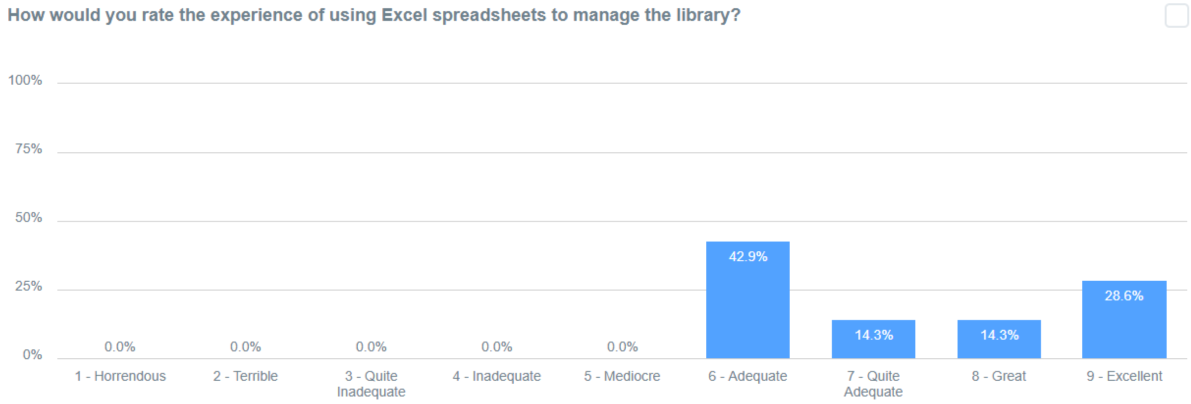
□ List of books marked as lost

□ Others?

Results:

Experience of using the current system:

Fig 6. Results of the question regarding the experience of using the current system.

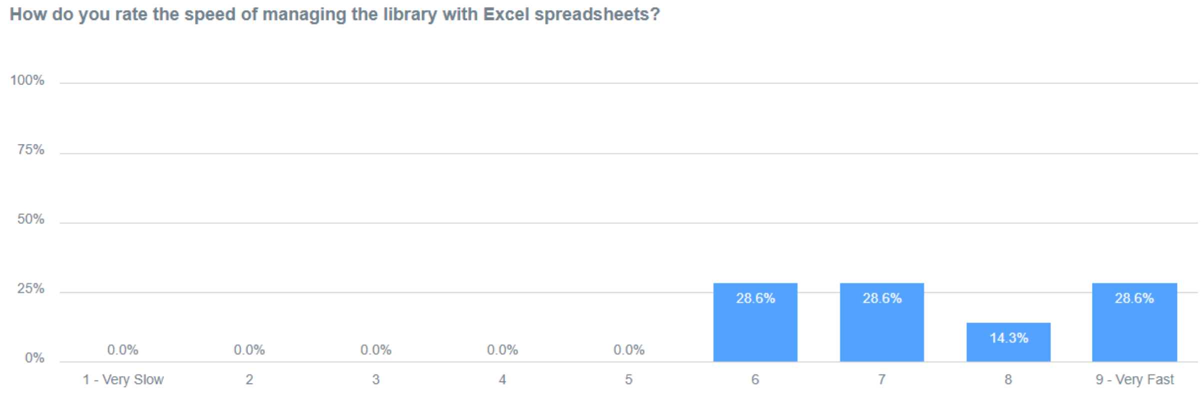


How would you rate the experience of using Excel spreadsheets to manage the library?

This question asked about how the users found the experience of using the current system. All of the surveyees gave a response of adequate to excellent. This shows that the users are used to using the current system and that a change would possibly take a while to get used to or be initially rejected. This means the new system should be intuitive for the user, in order to have a smooth transition to the new software.

Thoughts regarding the speed of using the current system:

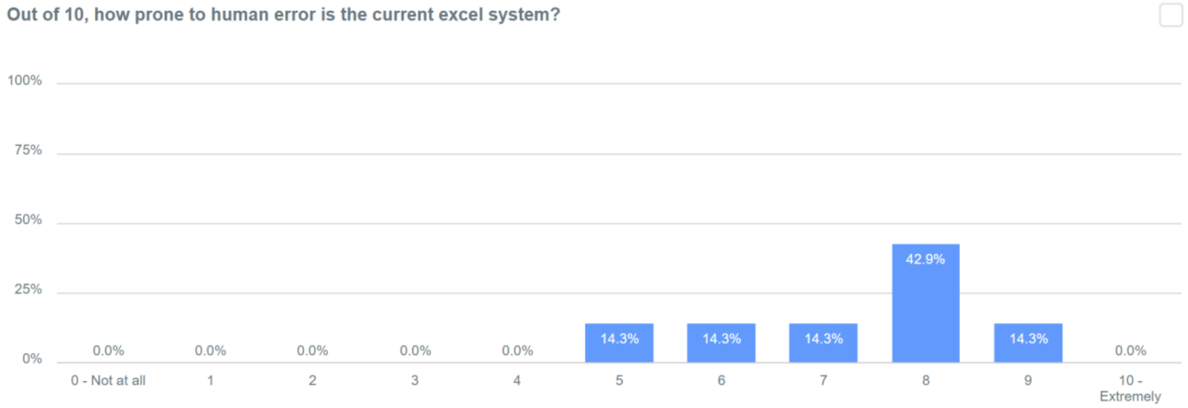
Fig 7. Results of the question regarding the speed of using the current system.



All those surveyed noted that the speed of the current system as being somewhere between 6 and 9 on a scale of 1-9 with 1 being “Very slow” and 9 being “Very fast”.

Proneness of the current system to human error:

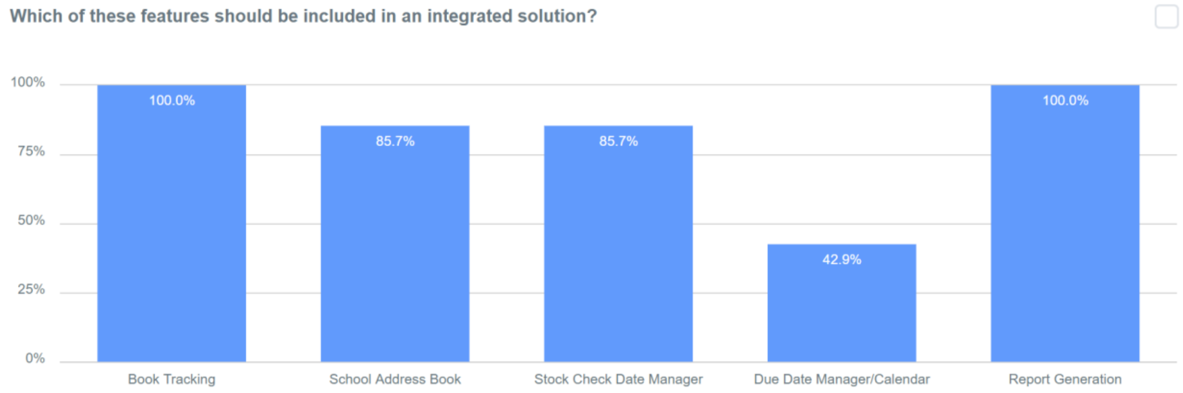
Fig 8. Results of the question regarding how prone the current system is to human error.



This question asks the surveyees how much the current system for monitoring exchanges is prone to human error. All of the surveyees responded with a score of 5 or above on a 0-10 scale which shows that every user surveyed finds the excel system more error prone than not. The score with the highest number of responses was 8/10 which shows that it is typically found very error prone. The fact that every respondee said it was error prone shows that a system that is less error prone would be a good alternative to the current system so an intuitive user design to minimise errors with straightforward step by step processes to accomplish various tasks would be good.

Main feature requests:

Fig 9. Results of the question regarding features wanted by the users.



All those surveyed requested a book tracking feature to monitor where different copies of books are located.

Another feature that was requested by all of the surveyees was Report Generation. This is a feature for creating printouts of useful data based on statistics from the system. These reports include a list of books at a certain location, list of visits to and from a certain location, how many books each school has on loan and how many they are supposed to have, and a list of books that have been lost.

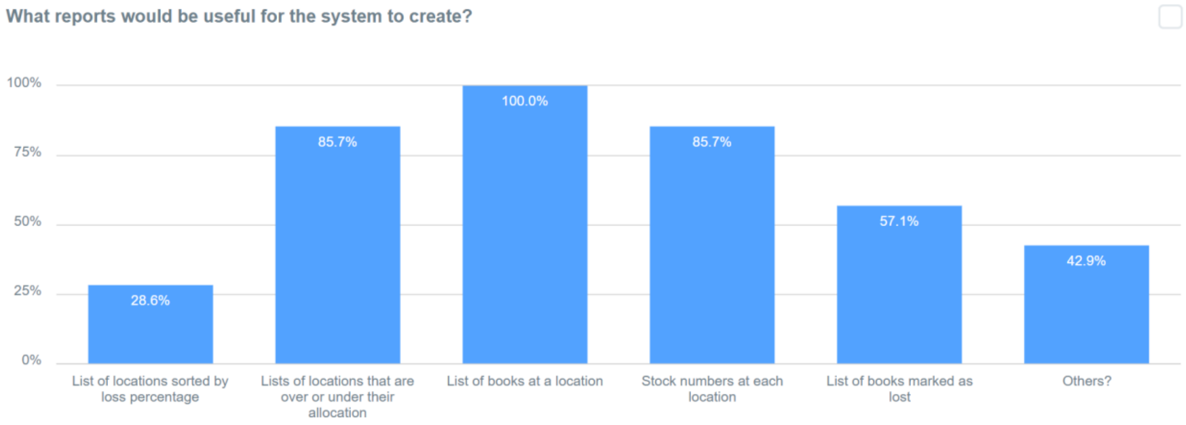
One feature that most of the users requested was an integrated School Address Book. This would store the address and contact details for each school as well as possibly have a button that opens the directions to the school in Google Maps in a browser.

Another frequently requested feature was a Stock Check Date Manager. This would be used to book dates for SLS library assistants to visit a school to verify the number of books that have been loaned out and are still at the school and to mark down any lost books.

The least requested feature of the options, chosen by under half the surveyees was a Due Date Manager/Calendar. However, since this is quite close to the Stock Check Date Manager, it would make sense to implement it as a whole calendar feature with event types for either stock checks or visits to return old books.

Reports Requested:

Fig 10. Results of the question regarding the reports that the system should create.



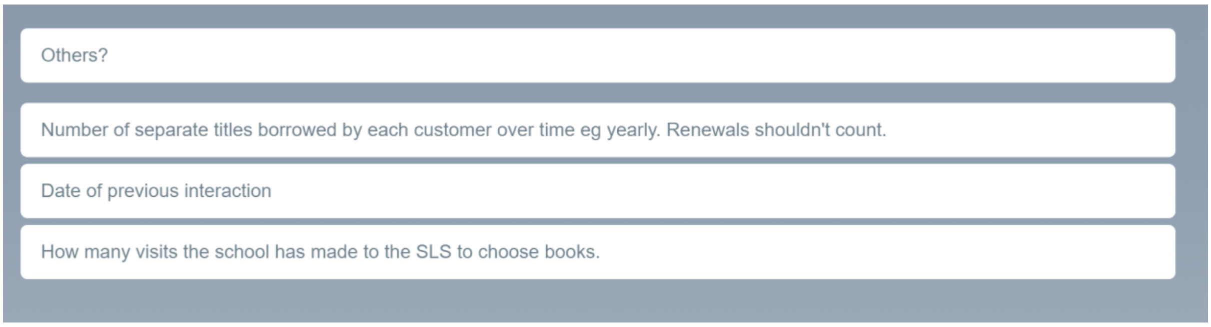
There was one report that every surveyee requested which was a list of books at a location. This would be useful for taking stock checks and returning books so that the school can be informed in advance which books need to be made available for the SLS library assistants to count.

The second most requested reports are a list of locations that are over or under their allocation and the stock numbers at each location. These reports could easily be combined into one with a list of schools, their current stock numbers, allocation amounts and whether they are over or under their allocation.

The third most requested feature was a list of books marked as lost. This would be useful for reordering lost books as well as finding out which schools are the worst at losing books so that the problem can be addressed.

The least requested report was a list of locations sorted by loss percentage. This would be used for identifying problem schools when it comes to losing books but, as shown in the survey, isn't a high priority feature and similar functionality could be achieved with the list of lost books report, albeit less efficiently.

Fig 11. Open responses to the question regarding the reports that the system should create.



Others?

Number of separate titles borrowed by each customer over time eg yearly. Renewals shouldn't count.

Date of previous interaction

How many visits the school has made to the SLS to choose books.

The survey also had an open response box on question regarding reports. Three surveyees put extra reports in this section.

The first suggested report was the number of unique titles borrowed by each school over a certain time frame. This could be useful for analysing whether a school typically takes many of the same books (indicated by a low number of unique titles) or whether they take one or two of many different titles (indicated by a high number of unique titles).

The second suggested report was a list of schools with the date of the previous interaction. This could be useful if a school didn't book a new date for their next exchange and the SLS library assistants can then work out when their next exchange should take place and contact the school. To improve upon the suggest functionality of this report, a list of next exchange dates pulled from the calendar feature would allow the SLS library assistants to see whether the school has booked another exchange from the same window.

The third suggested report was how many visits the school has made to the SLS to choose books. This would be useful to analyse how often a certain school typically visits and then base the next visit from the average elapsed time between previous visits.

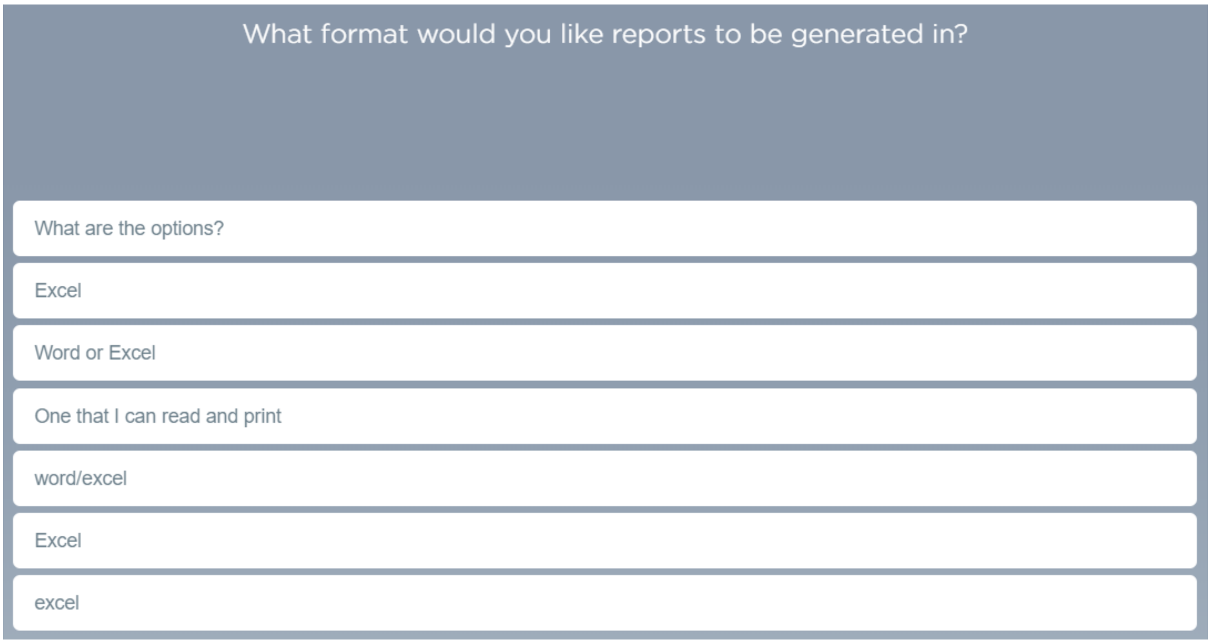
Of all these reports, there are three high priority reports based on amount of requests and observed importance based on the brief and then four low priority reports where only one or two requests were made or its additional functionality that could be built into one of the other reports at a later date.

The three high priority reports that will developed first before the first release version of the software is finalised are:

1. A List of Books at a Location
2. A List of Allocation Statistics regarding number of books at a location.
3. A List of Books Marked as Lost

The other reports that are considered useful would be developed after the initial release of the software as part of a functionality update.

Fig 12. Responses to the open question regarding what format the reports should be in.



What format would you like reports to be generated in?

What are the options?

Excel

Word or Excel

One that I can read and print

word/excel

Excel

excel

To determine the format that the reports should be created as, I put an open answer box with the question "What format would you like the reports to be generated in?" which resulted in the above responses. The general consensus was that a format from a standard office application would work best as the users already have those applications installed and available to them. It would also be beneficial to use an already existing standard for the reports as that wouldn't require any additional training in order to open and print the reports. As for the final file type, the most requested format was an excel file. This file type also makes the most sense from a programming perspective as there is an easy to use and open-source library available for creating spreadsheets from the report data as stored in the code.

Evaluation:

This questionnaire has a few flaws in the way it was carried out. Firstly, the sample size (n=7) that completed the questionnaire was smaller than the already limited target population that is the SLS library assistants. This means that my responses are less likely to be varied or represent the views of library assistants from other services offered by other counties who could be interested in using the system.

Identification of User Needs

- Reports
  - A List of Books at a Location
  - A List of Allocation Statistics regarding number of books at a location.
  - A List of Books Marked as Lost
- Calendar
- no more than 2 of a certain title per school.
- format for reports

Acceptable Limitations:

- Only able to use it on one computer because the database is kept locally and does not support multiple concurrent database connections.

Data sources and destinations

The first main data source for the program is the database in which most data will be stored. In the database there are five tables for the user logins, book details that have been modified, school details, loan information and what loan a book is on.

The second main data source for the program is the Google Books API. The API will allow the program to collect information about a book for various reasons. The most important feature for the user is that it can use the data from Google Books to generate a list of books that are currently checked out to a school so that the school can make sure they don't miss any books because they didn't know it was part of the SLS loan.

The data coming out of the program will be the lists of books at locations as well as reports on the loss rate and other statistics that can be calculated about the loans.

Data Volumes

The main data volume is the SQLite database backend of the program. It contains all of the data on books and loans as well as schools. The size of the database, once implemented in a production environment, will be much higher than the testing version because of the difference in quantities of books and schools which means I can't yet estimate the size of an implemented database.

Schools table: 3 digit integer for id, 40 characters for name, 210 characters for address, 70 characters for HT, 10 characters for date, 10 characters for DFE, 11 characters for contact, 2 digits for items per pupil and 4 digits for no of pupils adds to 360 characters per row which approximately equals 360 bytes per row. There are currently 85 schools in the test database based on a sample version of the current system from 6 or 7 years ago. So an estimated maximum number of schools could be upto about 200. Therefore for the schools table, there would be 72,000 bytes of data or about 72KB.

Login table: 2 digits for user id, 40 characters max for username, 128 characters for the sha512 password hash and 5 characters for admin or not, adds to 175 bytes per row. There would be about 10-20 users at the base location so assuming 20 users/rows with 175 bytes per row, the estimated maximum for this table is 3500 bytes or 3.5KB.

Loans table: 4 digits for loan id, 10 characters for the date of loan, 3 digits for school id and 5 characters for whether the loan is active or not, adds to 22 bytes per line. Each school visits approximately 3-6 times a year so assuming max 6 visits a year and max 200 schools, the total for the table would be 26.4KB per year.

Changed books: 13 characters for isbn, 100 characters for title, 50 characters for author, 30 characters for genre, 10 characters for release date, 4 characters for binding, 10 characters for age, 4 characters for label, 500 characters for blurb and 5KBs for the cover image adds to 5707 bytes per row. The users will choose how many books get modified but with an estimate of 200 books being modified, it would need 1.1MB.

Calendar table: 3 characters for event id, 3 characters for school id, 10 characters for date, 20 characters for event type, 500 characters for notes and 5 characters for time adds to 541 bytes per row. Assuming a max of 6 exchanges and 6 stock checks for each of a max of 200 schools per year, one year of events would be 1.3MBs.

Books table: 13 characters for isbn, 2 digits for copy number, 4 digits for loan id and 5 characters for lost or not, adds to 24 bytes per row. Assuming about 15,000 books in the system, it would total to 360MB.

The total size of all the tables after one year of active deployment would be about 362.5MB which would increase by about 1326KB each year.

Objectives for Proposed System:

Key Objectives:

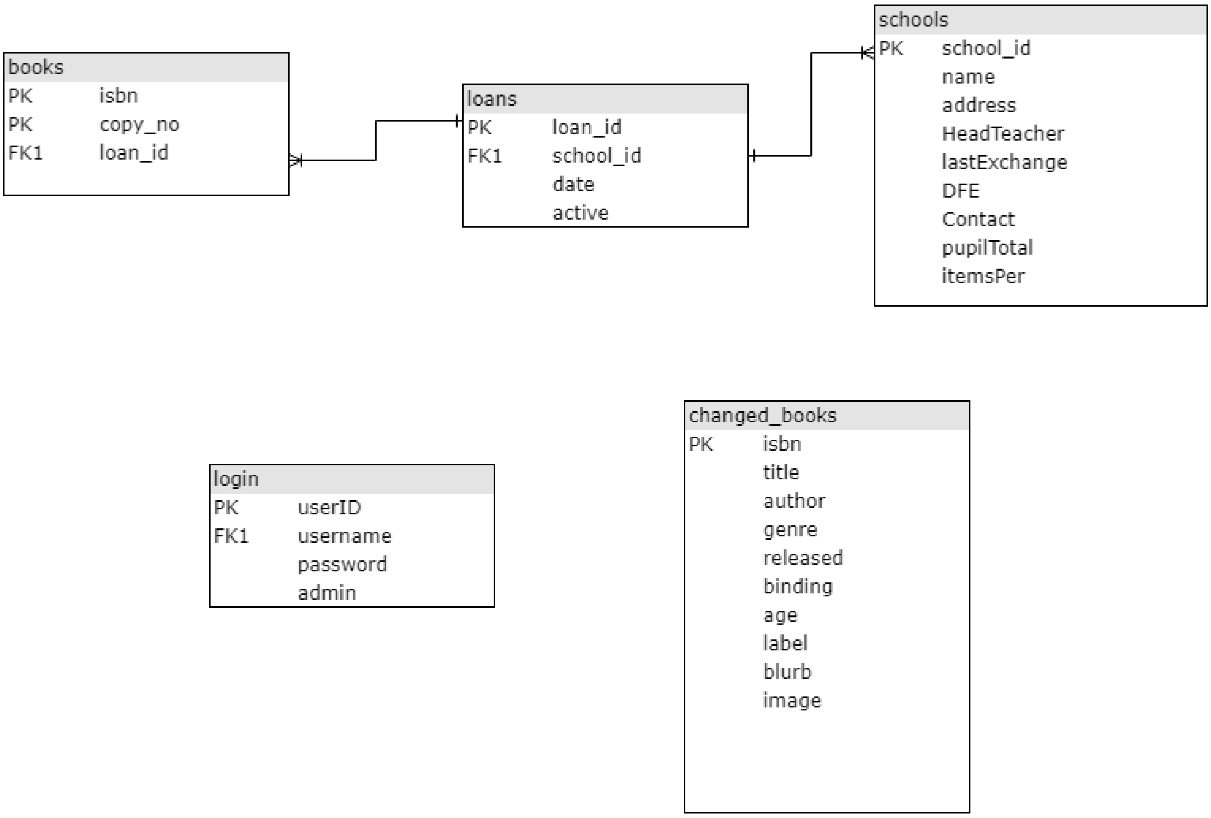
1. The system must be able to sign books in and out to different schools.
2. The system should keep track of books in-stock and at different the schools.
3. The system should have a database to store details about the schools that subscribe to the service.
4. The system must be able to produce reports about the loan history of individual schools and the service as a whole.
5. The system should have a login policy to protect sensitive data about the subscribed schools.
6. The graphical user interface must be user friendly and intuitive.
7. The system should protect the database from SQL Injection attacks that could otherwise cause irreversible damage to the structure and contents of it.
8. The system should be able to look up the details of book from the internet or a database with modified details and use those details in various places.

Additional Objectives:

1. The system should use Object Oriented Programming where possible.
2. The system could have a settings menu where the user can customise their experience (e.g. colour themes) and change the locations of any required files.
3. The system could have a calendar system to manage school visits.
4. The systems reporting function could calculate various statistics about their visits, borrowing history and amount of lost/damaged books.
5. The system could have a lookup system for books pertaining to a particular topic with a way to check their location.
6. The system should be able to make use of a barcode scanner for faster signing in and out of books.
7. The system should be simple to setup and install on the target users computers.

Analysis data dictionary and ERD

Fig 13. Database ERD



books	
PK	isbn
PK	copy_no
FK1	loan_id

loans	
PK	loan_id
FK1	school_id
	date
	active

schools	
PK	school_id
	name
	address
	HeadTeacher
	lastExchange
	DFE
	Contact
	pupilTotal
	itemsPer

login	
PK	userID
FK1	username
	password
	admin

changed_books	
PK	isbn
	title
	author
	genre
	released
	binding
	age
	label
	blurb
	image

Data Dictionary

Table Name	Columns	Description
books	isbn, copy_no, loan_id	Links a copy of a certain book isbn to a loan via it's loan id.
changed _books	isbn, title, author, genre, binding, age, label, blurb, image	Stores the details of a modified book.
loans	loan_id, date, school_id, active	Stores the details of a loan linking it to a school.
login	userID, username, password, admin	Stores the login details of users for to use when accessing the system.
schools	school_id, name, address, HT, lastEx, DFE, Contact, pupilTotal, itemsPer	Stores the details about a school.

Books

Column	Required	Type	Field Length	Default Values	Notes
isbn	✓	Text	10 or 13	--	
copy_no	✓	Integer	N/A	--	
loan_id		Integer	N/A	--	

changed_books

Column	Required	Type	Field Length	Default Values	Notes
isbn	✓	Text	N/A	--	
title		Integer	N/A	--	
author		Integer	N/A	--	
genre		Integer	N/A	--	
released		Integer	N/A	--	
binding		Integer	N/A	--	
age		Integer	N/A	--	
label		Integer	N/A	--	
blurb		Integer	N/A	--	

image		Blob	N/A	--	

loans

Column	Required	Type	Field Length	Default Values	Notes
loan_id	✓	Integer	N/A	--	Auto-Increment
dates	✓	Text	N/A	--	
school_id	✓	Integer	N/A	--	
active	✓	Text	N/A	--	

logins

Column	Required	Type	Field Length	Default Values	Notes
userID	✓	Integer	N/A	--	
username	✓	Text	N/A	--	
password	✓	Text	128	--	
admin	✓	Text	N/A	--	

schools

Column	Required	Type	Field Length	Default Values	Notes
school_id	✓	Integer	N/A	--	Auto-Increment
name		Text	N/A	--	
address		Text	N/A	--	
HT		Text	N/A	--	
lastEx		Text	N/A	--	
DFE		Text	N/A	--	
Contact		Text	N/A	--	
pupilTotal		Integer	N/A	0	
itemsPer		Integer	N/A	3	

Realistic appraisal of the feasibility of potential solutions:

Key:

Character	Note
□	Very difficult but still possible
✓	Possible but time consuming
✔	Yes
✕	No or not practical

Project Platform	Windows	MacOS	Linux	Android	iOS	Experienced with	Has libraries to use	Supports USB Scanners
Python Text Interface	✓	✓	✓	✕	✕	✓	✓	✓
Python Tkinter GUI	✓	✓	✓	✓	□	✓	✓	✓ (Bluetooth version required for Android and iOS)
Python Kivy GUI	✓	✓	✓	✓	✓	□	✓	✓ (Bluetooth version required for Android and iOS)
Visual Basic	✓	✕	✕	✕	✕	✓	✕	✓
Java	✓	✓	✓	✓	✓	✕	✕	✓
WebApp	✓	✓	✓	✓	✓	✕	✓	✓ (more setup required on user end + Bluetooth version for mobile)

Justification of chosen solution:

- Python with Tkinter GUI

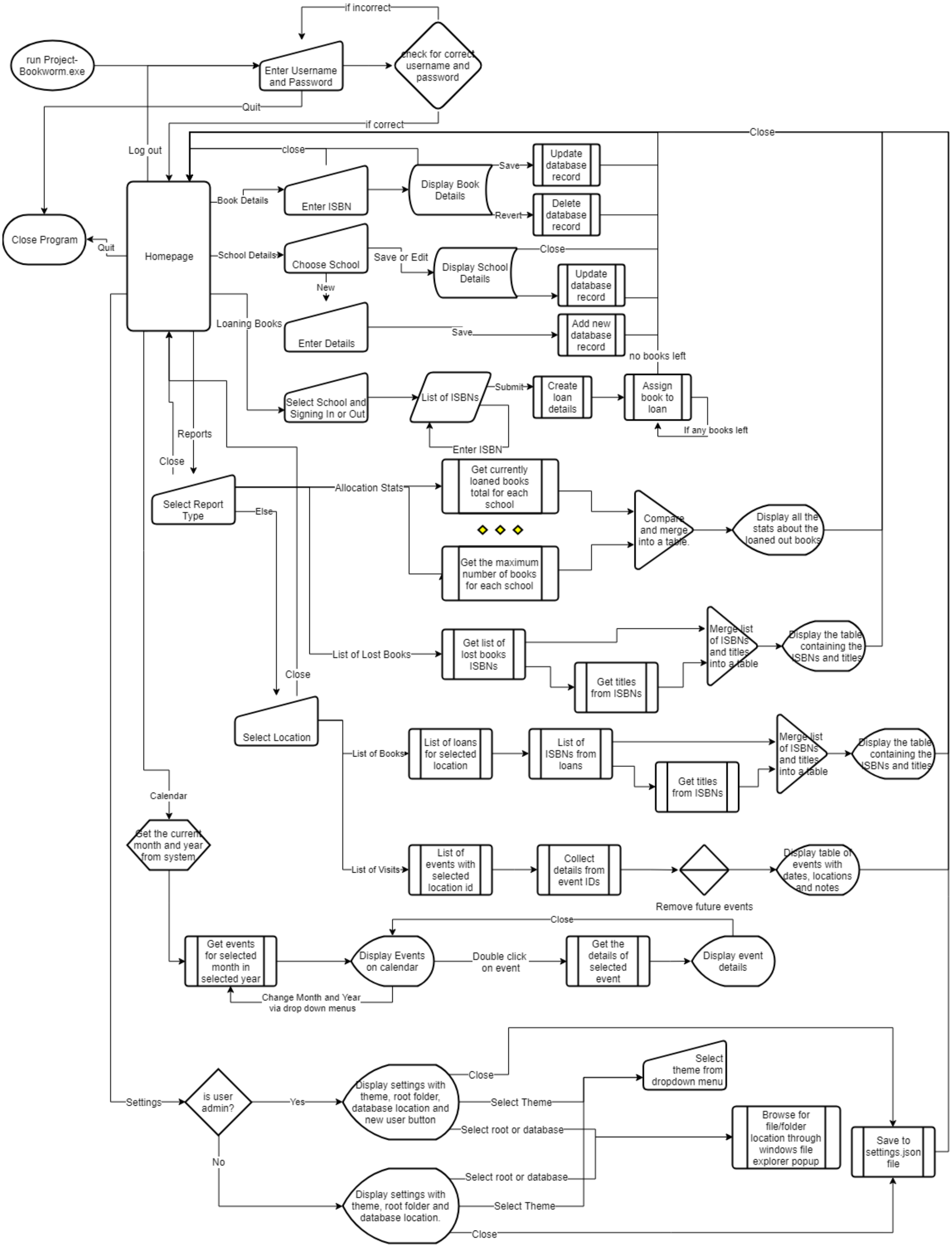
I have chosen to use Python with Tkinter for the GUI as my method of choice for my project. I am using it because I know Python quite well (compared to Visual Basic which is the only other language I have used before) and I have already learnt Tkinter as a GUI implementation technique for Python as opposed to learning Kivy just for being able to easily put the program on mobile devices.

I also know that the users have Windows machines as their desk computers at the site meaning mobile devices are not a required feature.

Design

Overall System Design

Fig 14. System Design Flowchart.



if incorrect
run Project-Bookworm.exe
Enter Username and Password
check for correct username and password
Quit
if correct
Log out
close
Close
Close Program
Quit
Homepage
Book Details
Enter ISBN
Display Book Details
Save
Update database record
Revert
Delete database record
School Details
Choose School
Save or Edit
Display School Details
Close
Update database record
New
Loaning Books
Enter Details
Save
Add new database record
no books left
Select School and Signing In or Out
List of ISBNs
Submit
Create loan details
Assign book to loan
If any books left
Enter ISBN
Reports
Close
Select Report Type
Else
Allocation Stats
Get currently loaned books total for each school
Get the maximum number of books for each school
Compare and merge into a table.
Display all the stats about the loaned out books
List of Lost Books
Get list of lost books ISBNs
Get titles from ISBNs
Merge list of ISBNs and titles into a table
Display the table containing the ISBNs and titles
Close
Select Location
List of Books
List of loans for selected location
List of ISBNs from loans
Get titles from ISBNs
Merge list of ISBNs and titles into a table
Display the table containing the ISBNs and titles
List of Visits
List of events with selected location id
Collect details from event IDs
Remove future events
Display table of events with dates, locations and notes
Calendar
Get the current month and year from system
Get events for selected month in selected year
Display Events on calendar
Change Month and Year via drop down menus
Close
Double click on event
Get the details of selected event
Display event details
Settings
is user admin?
Yes
No
Display settings with theme, root folder, database location and new user button
Close
Select Theme
Select root or database
Select theme from dropdown menu
Browse for file/folder location through windows file explorer popup
Save to settings.json file
Display settings with theme, root folder and database location.
Select root or database
Select Theme
Close

Description of modular structure of system

Modules:

All of the code has been separated into different python modules (*.py). These modules can be imported into other parts of the code to allow modularity in the structure of the program. The modules also reduce the amount of code needed to be written as they can imported to allow the calling of the functions inside the modules.

Objects:

Objects are callable by any other part of the code where the modules are imported.

Some objects that I have implemented include:

- Entry form object
- book details object
- Homepage object
- Multi Entry init form object
- Multi Entry form object
- settings menu object
- School Details View menu
- School Details Buttons menu
- Calendar object
- Calendar Event details object
- Reports menu object
- Settings menu object
- Add New User object

Object:	Description:	Uses:	Called by:	Calls:
Entry Form	For entering a single value to be processed.	Entering an ISBN to view the details of that book.	Homepage	Book Details
Book Details	Displays the title, author, etc. that is passed into the object.	Displaying the details of a book.	Entry Form	Homepage
Homepage	The main menu of the program. Has buttons to call each of the parts of the program.	Calling smaller independent parts of the program such as Settings and the Multi Entry form.	Starting the program, Book Details, Settings, Multi Entry, Entry Form.	Entry Form, Multi Entry, Settings Menu
Multi Entry Init	For choosing a school to sign books in from or out to.	Configures the multi entry page to the correct school and correct method.	Homepage	Multi Entry

Multi Entry	For entering multiple values to be processed.	Entering many ISBNs to be assigned to a loan.	Multi Entry Init	Homepage
Settings Menu	Changing settings that apply globally to the program.	Changing the colour theme of the program.	Homepage	Homepage
School Details View	For the viewing, editing or creating of school profiles in the database.	Can be used to create a new school profile in the database, edit a pre-existing one or viewing the details of one.	School Details Init.	Homepage.
School Details Init	Initialising the school details view object.	Opens the School Details View in new school mode for creating a new school or selecting an existing school	Homepage.	School Details View.
Calendar Object	Viewing, editing and creating events in a typical calendar	Has an interactive calendar with which the user can change the month and year to view the corresponding events and can also create events using a button and the resulting popup and edit the events with	Homepage	Homepage.

	format	double clicking on the event name on the calendar.		
Reports	Interface for generating reports based on data from the system	Has a series of dropdown menus that the user can use to select a report type and any required options for it to be generated. The button to generate a report will appear once all the required options have been selected. Once clicked it will trawl the database for the data needed by the report and create a table in a new window with the report. Once generated, there is an option to print the report as a Excel spreadsheet that can be sent to schools or fellow library assistants, or can be used to print the report onto paper.		
Settings	Modify settings pertaining to the functionality and aesthetics of the program.	This menu allows the user to change settings stored in a json file through a user friendly interface. It has fields for the database location and the root folder with buttons to bring up the standard windows file/folder selection window. It also has a drop down menu to select the theme for	Homepage	Homepage, Add New User

		the colours of the system. If the user is an admin, a button for adding a new user appears alongside the other settings.		
Add New User	Create the credentials for a new user to login to the system.	This menu allows an admin to create a new login profile for a user to access the system with.	Settings	Settings

Function Modules

Some of the modules written only contain functions and algorithms that other parts of the program can import and use.

The function modules that I have written are:

- books_api
- gui
- img2gif
- misc_python
- sql

The "books_api" module is for interacting with and processing the data from the Google Books API. It contains three functions.

The first function takes an ISBN from the program and returns properly formatted python dictionary with all of the details about the book.

The second function is used to get a specific detail about a book from the dictionary returned by the first function.

The third one runs the second function for all of the details required in the book details window and returns them in a format that can be accepted by the book details object for being displayed or compared against the modified data from that window.

The "gui" module is written to make bringing up a new window easier.

Each function in the module contains the few lines of code required to call a new GUI object so that other parts of the program can do so in one line. It also makes the program code easier to read since the same lines of code are repeated less often.

The "img2gif" module is for the manipulation of the book cover images before they get displayed in the Book Details window.

The tkinter canvas object on the window only accepts image files in the GIF type whereas the Google Books API stores all of the images as JPEG files. This means in order to display them correctly, the program needs a way to convert between the different file types.

As well as the file type discrepancy, there are often differences between the resolution of the image file from the Google Books API and the tkinter canvas object in which the images are needed to be displayed. This module also solves that issue by resizing the images to the correct resolution to reduce the blank space around the image and to keep as much of the cover from being cropped.

The "misc_python" module is different to the other function modules in the fact that it wasn't written specifically for this program. Instead, it is a collection of useful functions and algorithms that I have written and collated into one module for use in any program I write.

One example of a function it contains is a binary search algorithm for searching through lists for a specific item.

Another function is a sorting algorithm which sorts alphabetically and numerically to make sure that multiple digit numbers are sorted correctly and not by the first digit as happens with algorithms like quicksort.

The last function module I've written is a module for interacting with the sqlite database that the program is based around. This module is made up of lots of functions, each with its own SQL query, that can be executed in a connection to the database using variables that are passed into the function from the main program.

Modular Design

- Login Screen
    - Login Button
        - Main Menu Screen
            - Book Details
                - Single Entry
                    - Submit
                        - Book Details (Title, Author etc of the book specified by isbn)
                            - Save Changes
                            - Close
                            - Revert to online data
            - Sign Out Books
                - Destination Select Screen
                    - Sign In
                        - Multi-Entry
                            - Sign In
                    - Sign Out
                        - Multi-Entry
                            - Sign Out
            - School Details
                - School Selection Screen
                    - Edit School
                        - School Details (Edit Mode)
                    - New School
                        - School Details (New Mode)

        - View School
            - School Details (View Mode)
- Settings
    - Settings
        - Theme Select
        - Database Location
        - Root Folder
        - Add New User (Only when admin)
            - Add New User
        - Close
- Log Off
- Quit Program

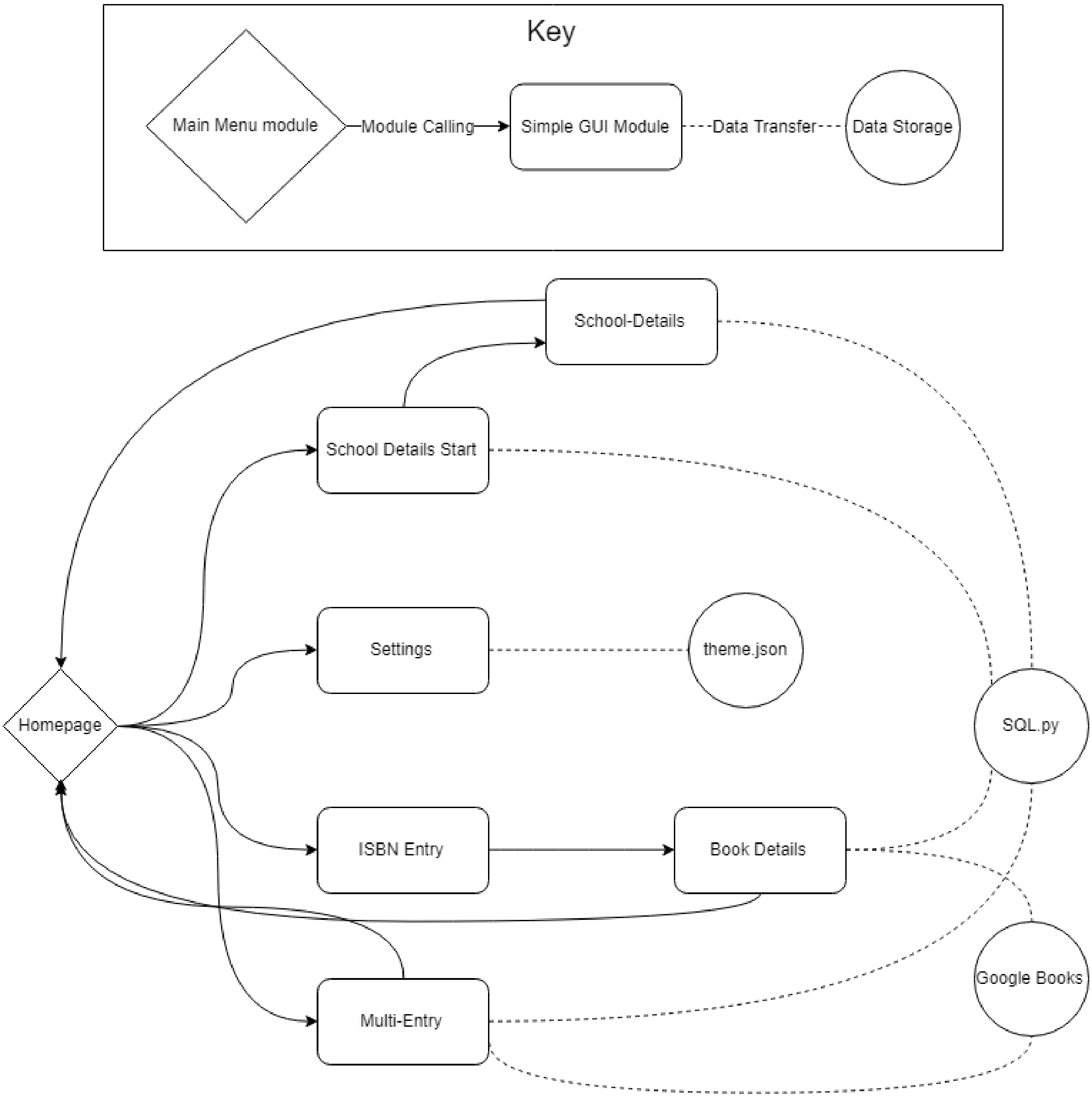
Definition of data requirements

Identification of appropriate storage media

Entity relationship diagram(Normalised)

Entity Relationship Diagram:

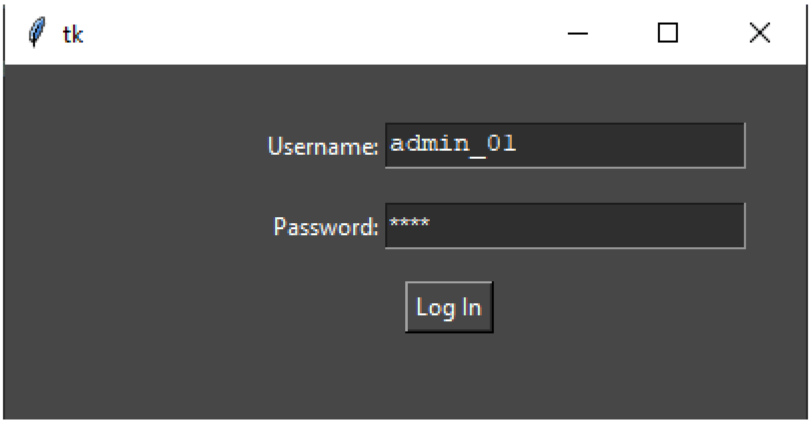
Fig 15. Program Entity Relationship Diagram.



User interface design (HC)

Login Screen

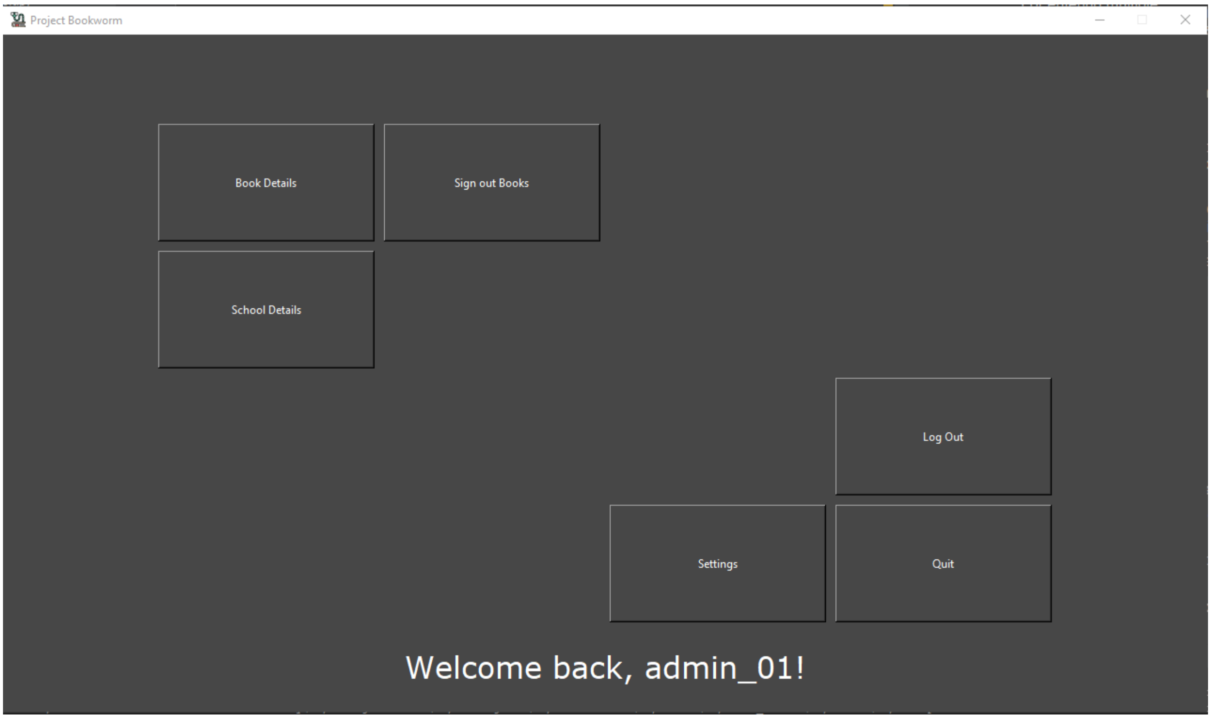
Fig 16. Login screen.



This is the Login screen that is presented when the user opens the application or logs out from the homepage. The menu is characterised by a pair of entry fields, one for the username and one for the password, as well a submit button. The button isn't the only way to run the authentication process as the user can also press `Enter` to submit the details as is standard of most login systems. The password field is obscured upon entry with asterisks (`*`) replacing each character as is standard with almost all Windows-based login systems as well as on many websites. When submitted, the password is hashed with an SHA512 cryptographic hash function before being compared with the hashed password associated with the given username that is stored in the databases login's table.

Homepage

Fig 17. Program Homepage.

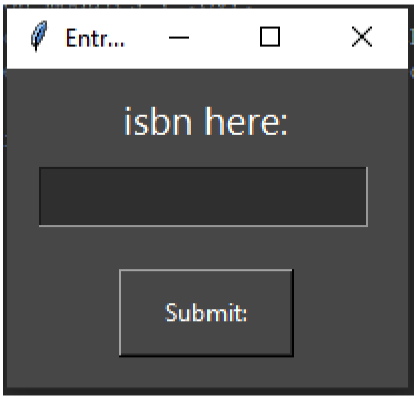


Welcome back, admin_01!

This is the homepage for the application where the user can navigate to all of the various parts. The homepage is made up of various buttons linked to the different sections of the program. There are also buttons for `Log Off` and `Quit` to allow the user to end their unique session or to completely close the application respectively. At the top of the window is the program name and logo and at the bottom is a welcome back message with the current username.

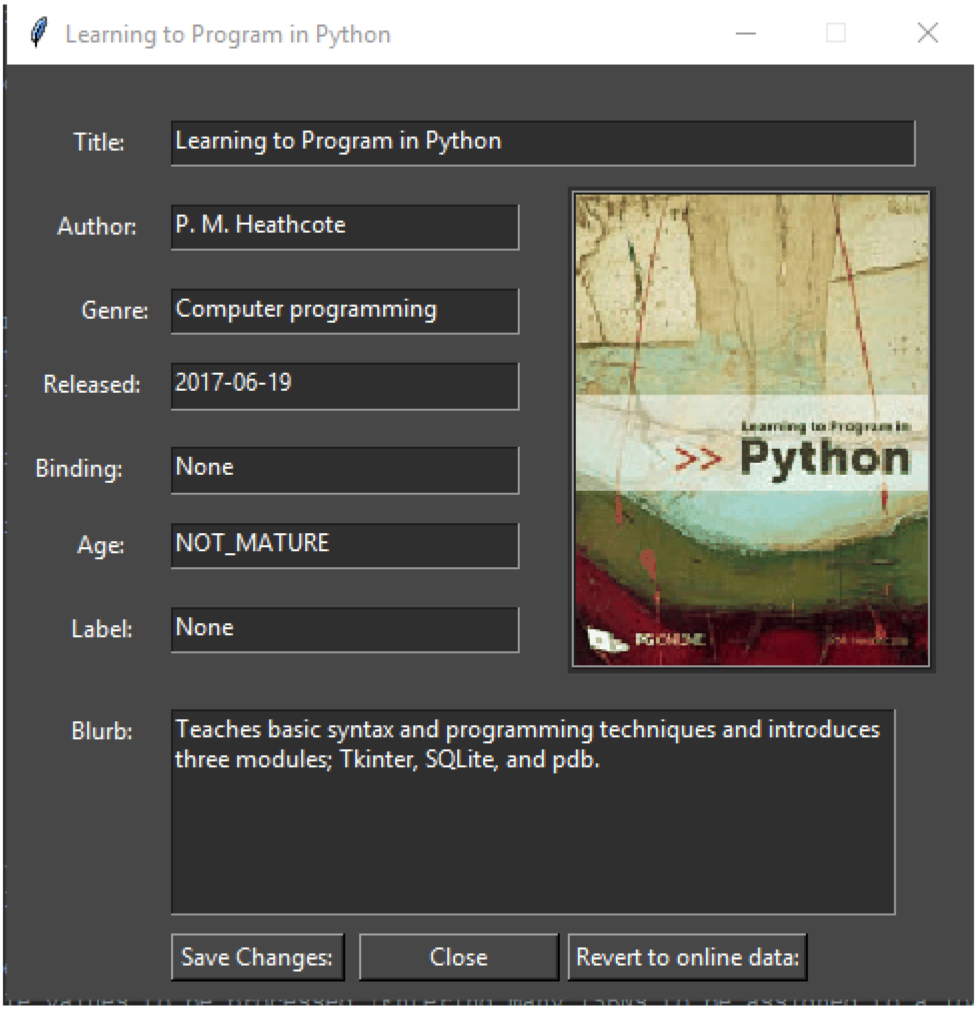
Book Details

Fig 18. Entry form for single ISBN.



This part of the book details process is for taking the inputted ISBN for a book and passing it through to the next section. The menu has a entry box for the ISBN and a label above it to instruct the user on what to input. Below the entry box is a button which will submit the contents of the entry box to the next bit of the Book Details section.

Fig 19. Book Details form



Title: Learning to Program in Python

Author: P. M. Heathcote

Genre: Computer programming

Released: 2017-06-19

Binding: None

Age: NOT_MATURE

Label: None

Blurb: Teaches basic syntax and programming techniques and introduces three modules; Tkinter, SQLite, and pdb.

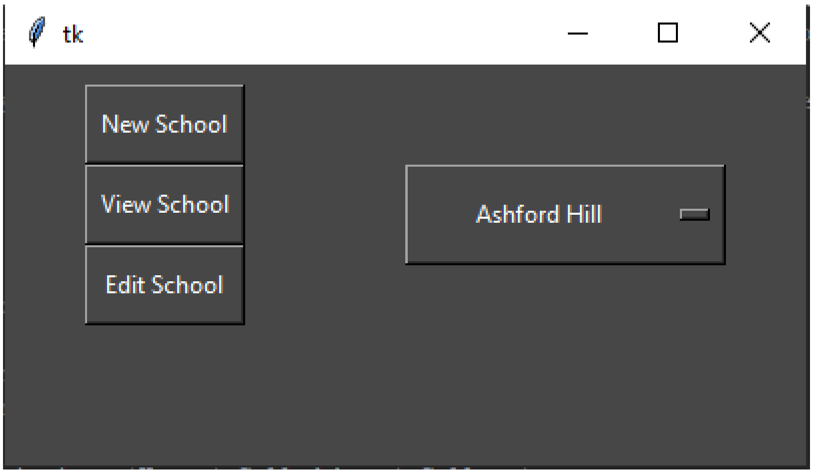
This is the window that is used to display the details of a book. The details are pulled from either the sqlite database (if there is an entry for that), ISBN or the Google Books API if there isn't. The details that are available for being displayed are:

- Title (and subtitle if applicable)
- Author(s)
- Genre
- Release/Publish Date
- Age Rating
- Blurb
- Cover Image

This also includes two placeholder fields for anything else requested by the end users. At the bottom of the window are three buttons. One of these will save any changes made to the details by the user to the sqlite database. The next one will close this part of the program and return to the homepage of the application. The last button will delete any entry in the database and allow the program to use the original Google Books data for the details.

School Details

Fig 20. Menu for selecting the a school to show the details of.



This window is for selecting a school to view or edit the details of or to add a new school to the system. There are three buttons on the left of the window:

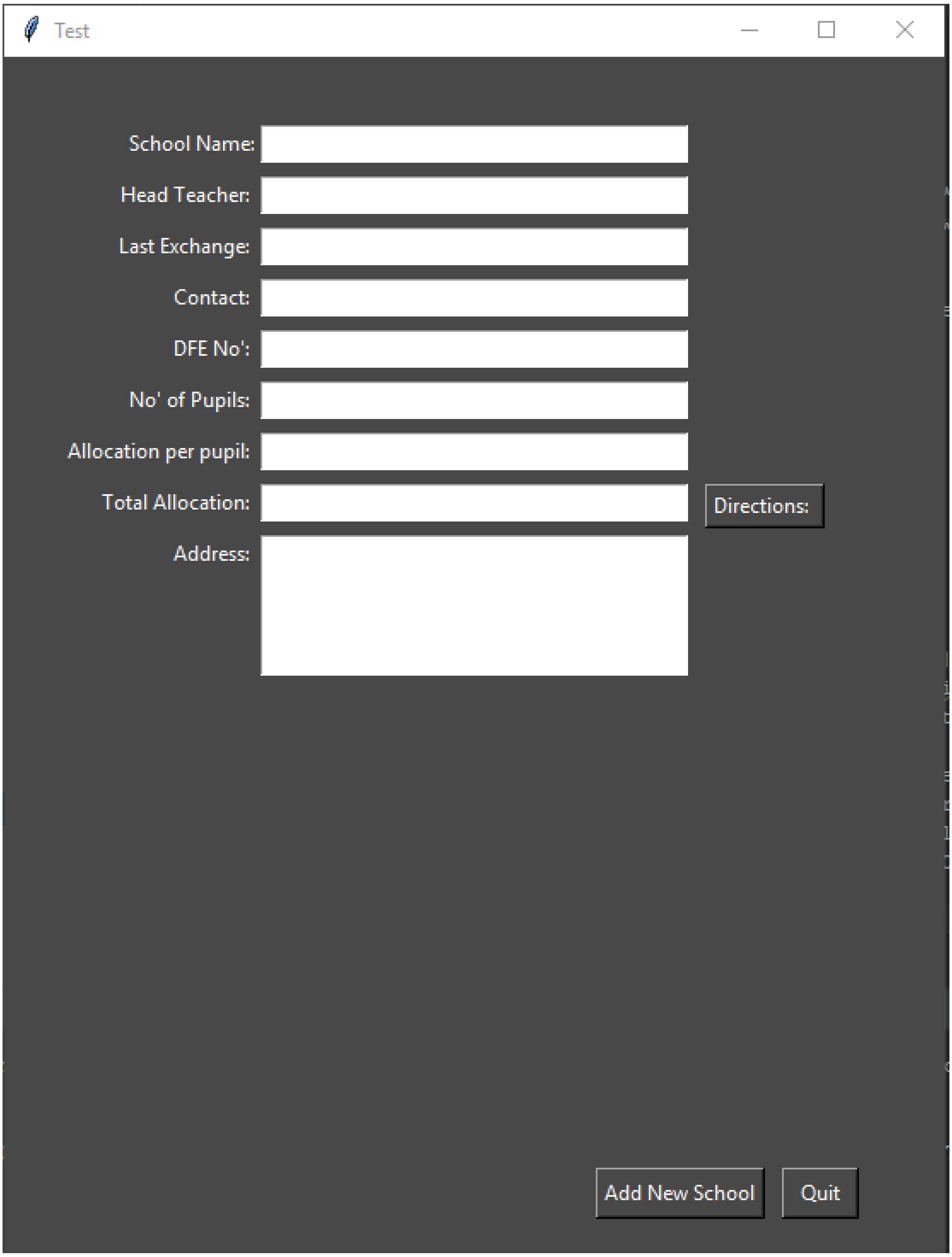
- `New School` for creating a profile of a new school in the database
- `View School` for viewing the details of an existing school profile
- `Edit School` for editing the details of an existing school profile

On the right of the window is drop down menu containing the list of existing school profiles to select from for use in the `View School` or `Edit School` windows.

The School Details main window has three variants which are chosen in the previous window. All of the variants are have the same fields to fill in or be viewed but they start differently depending on the variant. The fields are:

- School Name
- Head Teacher
- Last Exchange
- Contact:
- DFE No'
- No' of Pupils
- Allocation per pupil
- Total Allocation
- Address There is also a button next to the address field which will open Google Maps with the location of the school in the default web browser. At the bottom of all of the variants is a button to close the details menu and return to the homepage.

Fig 21. Empty School Details form for adding a new school.

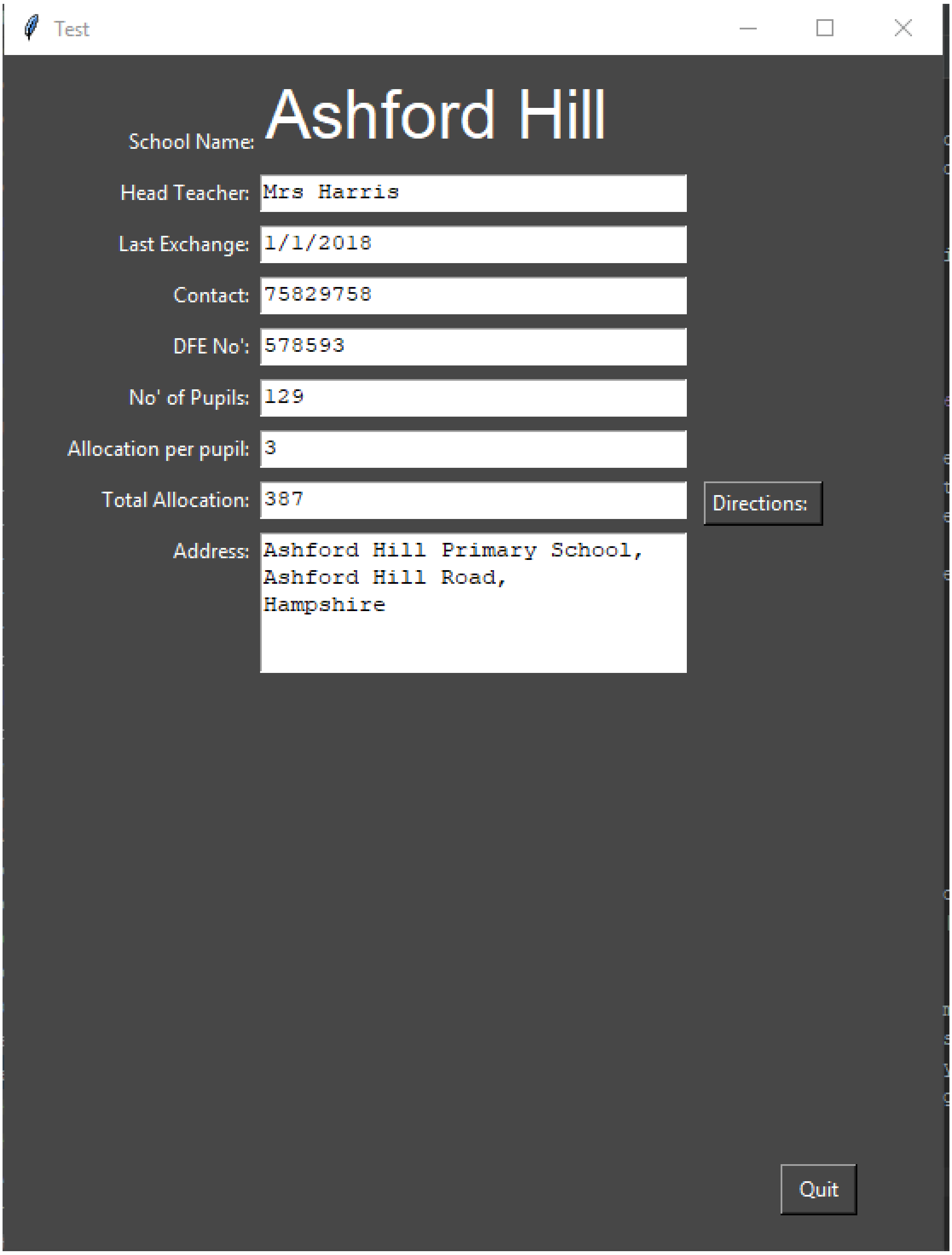


Test

School Name:	
Head Teacher:	
Last Exchange:	
Contact:	
DFE No':	
No' of Pupils:	
Allocation per pupil:	
Total Allocation:	
Address:	

The first variant of the School Details window is for create a new profile for a school. It has the same fields as the other variants but all are left blank for the user to fill in. At the bottom of the field is an `Add New School` button to save the entered details to the database as a new school.

Fig 22. School Details form in view mode with no save button for any changes.



School Name: Ashford Hill

Head Teacher: Mrs Harris

Last Exchange: 1/1/2018

Contact: 75829758

DFE No': 578593

No' of Pupils: 129

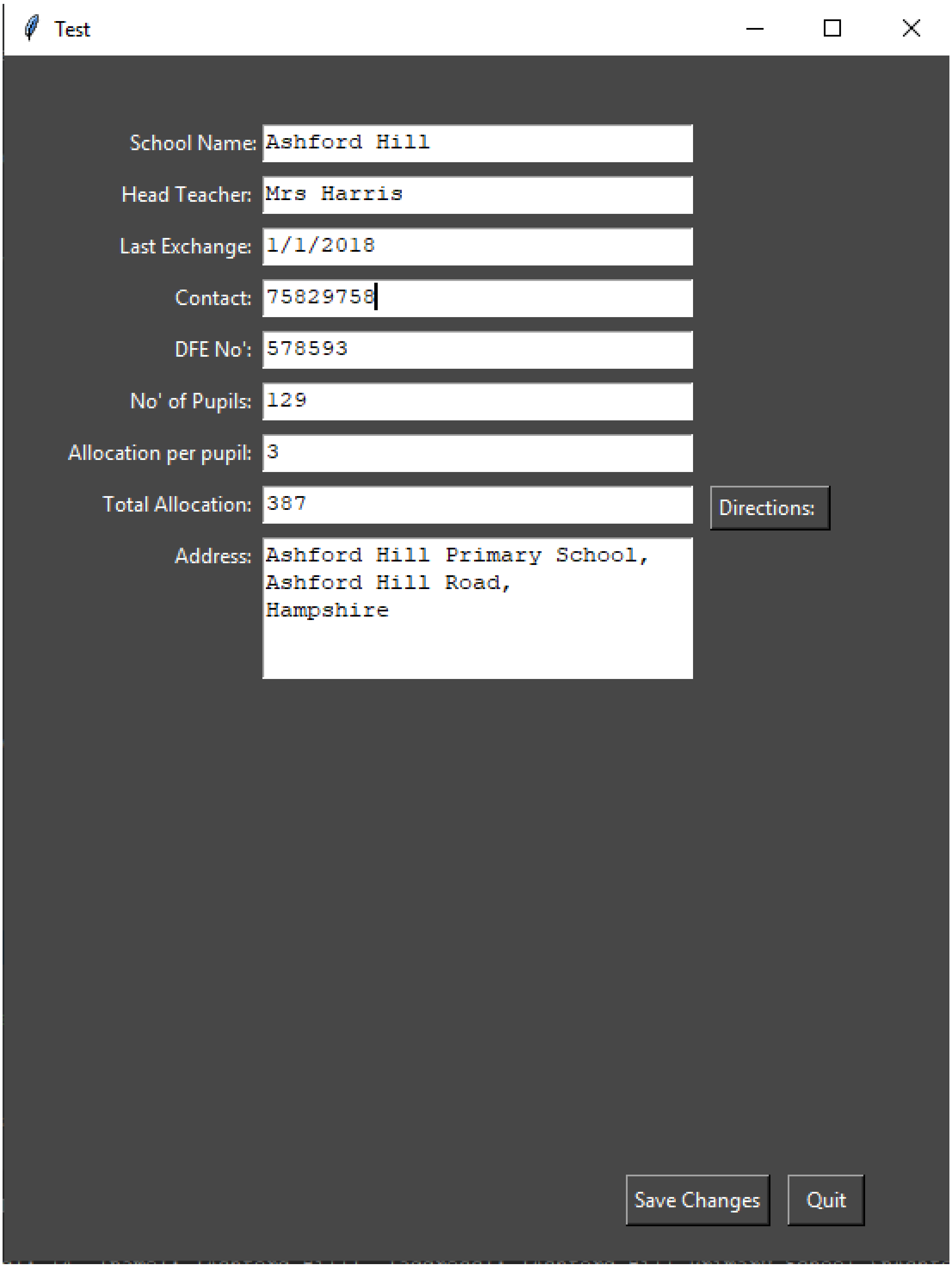
Allocation per pupil: 3

Total Allocation: 387

Address: Ashford Hill Primary School,
Ashford Hill Road,
Hampshire

The second variant does not have editable field for the school name or a button to save changes as it is only for viewing the details of a chosen school.

Fig 23. School Details form filled out with values from the database.



School Name: Ashford Hill

Head Teacher: Mrs Harris

Last Exchange: 1/1/2018

Contact: 75829758

DFE No': 578593

No' of Pupils: 129

Allocation per pupil: 3

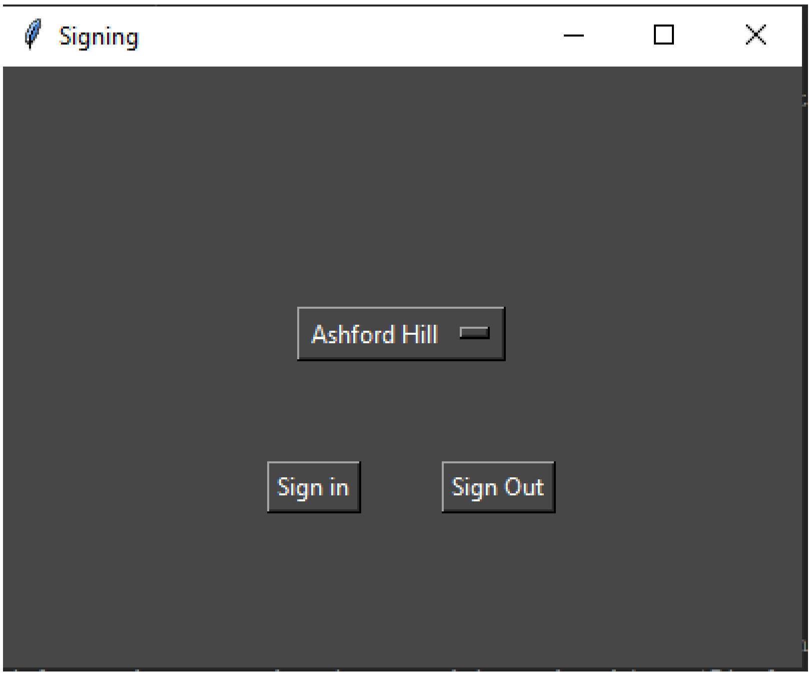
Total Allocation: 387

Address: Ashford Hill Primary School,
Ashford Hill Road,
Hampshire

The third and final variant is for changing the details of a school. As such, it starts with all of the details fields filled with the details in the database ready for being changed by the user and has a `Save Changes` button at the bottom for committing any changes to the database.

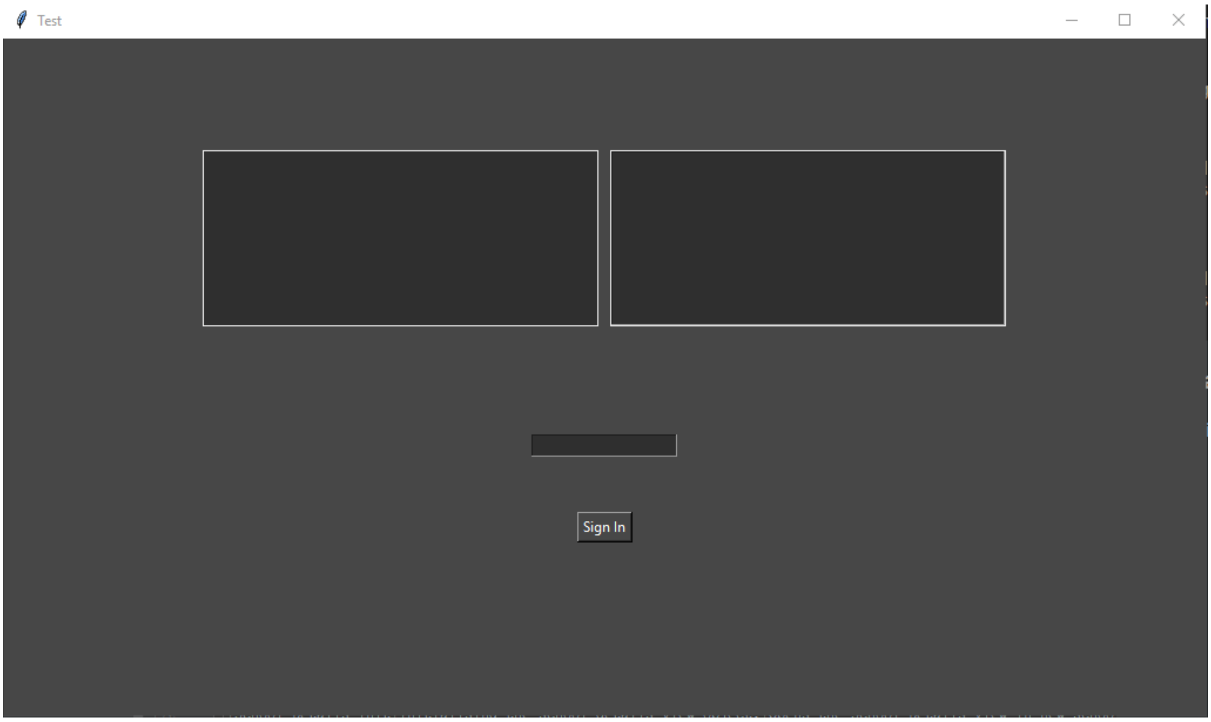
Sign Out Books

Fig 24. Menu for initialising the loaning and return process.



This window is for selecting a school to sign books out to or in from. It has a list of the schools in the system in a drop down menu to select one and two buttons to choose whether you're signing books in or out with respect to that location.

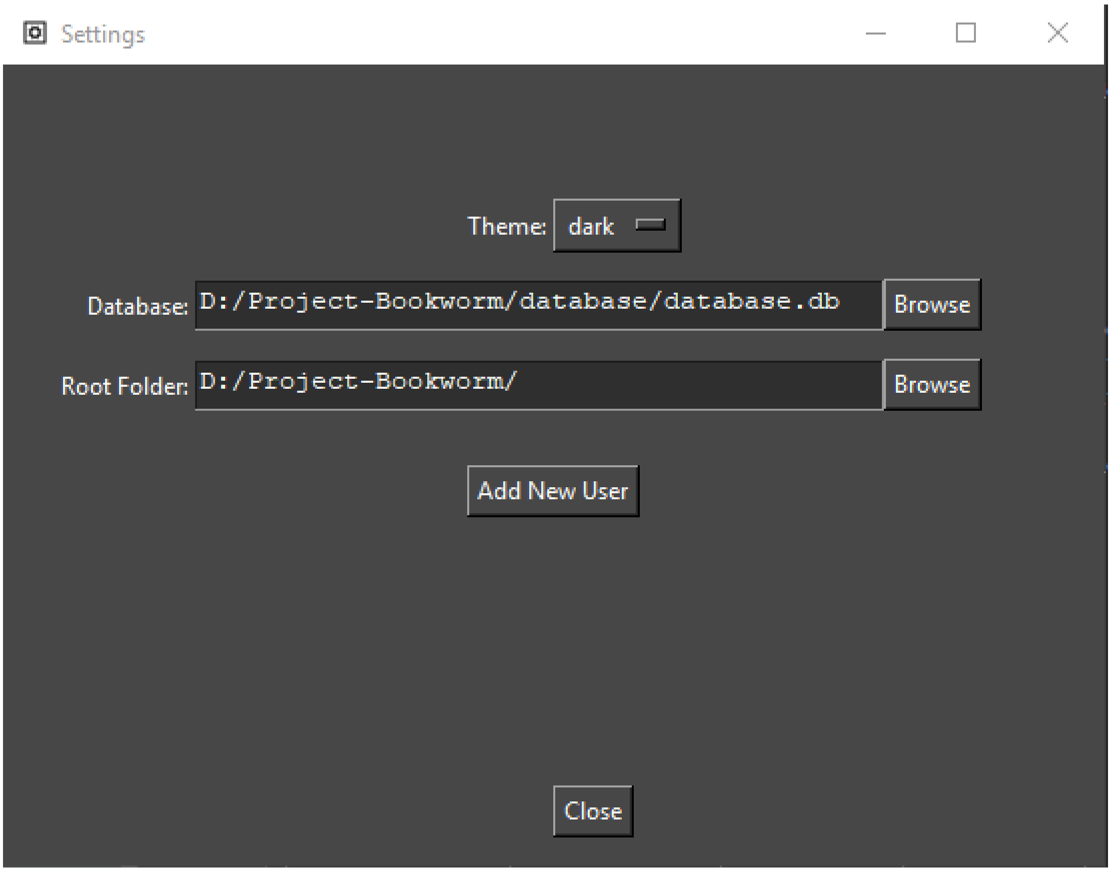
Fig 25. Empty Loaning/Returning form



This window has a single entry box for entering an ISBN of a book. The user enters the ISBN and presses the `Enter` key. This puts the ISBN in the list box on the left and collects the title from the database or the Google Books API and puts it in the list on the right to help make sure the correct book has been entered. Once all of the ISBNs are entered, the user clicks on the `Sign In`/`Sign Out` button which will sign the books out to the location selected in the previous window.

Settings

Fig 26. Settings menu as seen when signed in as an admin.



Settings

Theme: dark

Database: D:/Project-Bookworm/database/database.db

Root Folder: D:/Project-Bookworm/

This window is where the user can change any settings pertaining to the application that might need to be modified. The first setting is the application-wide theme option. This is changed via a drop down menu with a list of the available themes that can be chosen from. The next setting is the location of the sqlite database. This is chosen via a file path to the database file which can be selected via the Windows Open File dialog. The next setting is the root folder location for the program to use as a file location for the temporary storage of cover images for books and long term storage of icons required for the program. The last setting is an `Add New User` button for creating a new user profile which can be used on the login screen to access the application. This setting is only available to accounts with administrator privileges.

Hardware specification

Input Devices

Currently Required: Keyboard, Mouse. Optional: Barcode Scanner (untested)

- Keyboard: allows manual entry of barcodes and editing of book details.
- Mouse: for interaction with buttons on the forms. (tab is untested.)
- Barcode scanner: easier input of isbns stored as barcodes on most books.

Output Devices

Required: Monitor.

- Monitor: viewing of GUI.

Storage devices

Required: Program files and sqlite database. unknown sizes as unfinished. database may be moved to a server.

Processor and Memory requirements

By using Python to create the program, I have been able to keep the program quite lightweight in terms of system resources needed to run it.

Table:

	Processors	Memory	Storage	OS	Screen Resolution	Peripherals
Minimum:	i5-520M	6GB SODIMM DDR3	80.1MB w/out third-party modules	Win7 32bit		Keyboard and Mouse
Recommended:	Ryzen 7 1800x	16GB DDR4	8GB with all required modules	Win10 64bit	1920x1080 @60Hz	Barcode Scanner
Also Working:	i5-4460	8GB DIMM DDR3			1920x1080 @60Hz	

Testing

Description of measures planned for security and integrity of data and system security

- sql injection prevention

Overall test strategy

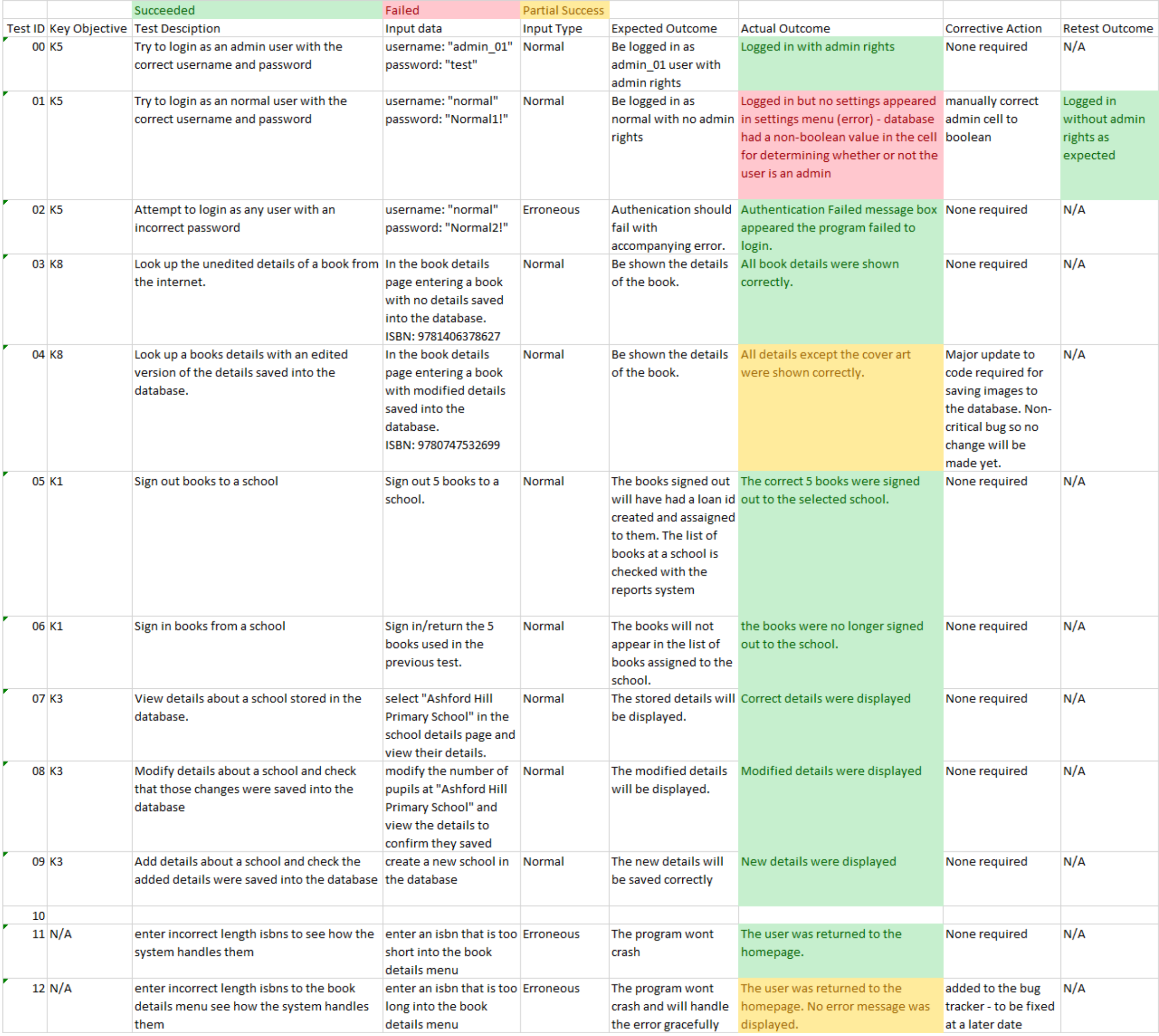
- test for sql injection
- see if it breaks under normal use cases
- test under extreme use cases to see if it breaks

Once I have finished creating the program, I will test it in a few main areas. Firstly, I will test against the key objectives to check that required functionality is working. This is done by attempting to complete a task that fits with the key objective being tested, for example signing books out to or in from a school to test Key Objective I.

Secondly, I will stress test the parts of the system where the length of the process varies based on amount data entered or stored in the database by using larger amounts of data than is realistic in typical use of the system, for example, excessively large numbers of books being loaned in one transaction (the library have a recommendation of about 200 books per exchange so 400 books in one go should be enough to stress the system).

Testing Details

Fig 27. Table of Tests conducted.



		Succeeded	Failed	Partial Success				
Test ID	Key Objective	Test Desciption	Input data	Input Type	Expected Outcome	Actual Outcome	Corrective Action	Retest Outcome
00	K5	Try to login as an admin user with the correct username and password	username: "admin_01" password: "test"	Normal	Be logged in as admin_01 user with admin rights	Logged in with admin rights	None required	N/A
01	K5	Try to login as an normal user with the correct username and password	username: "normal" password: "Normal1!"	Normal	Be logged in as normal with no admin rights	Logged in but no settings appeared in settings menu (error) - database had a non-boolean value in the cell for determining whether or not the user is an admin	manually correct admin cell to boolean	Logged in without admin rights as expected
02	K5	Attempt to login as any user with an incorrect password	username: "normal" password: "Normal2!"	Erroneous	Authenication should fail with accompanying error.	Authentication Failed message box appeared the program failed to login.	None required	N/A
03	K8	Look up the unedited details of a book from the internet.	In the book details page entering a book with no details saved into the database. ISBN: 9781406378627	Normal	Be shown the details of the book.	All book details were shown correctly.	None required	N/A
04	K8	Look up a books details with an edited version of the details saved into the database.	In the book details page entering a book with modified details saved into the database. ISBN: 9780747532699	Normal	Be shown the details of the book.	All details except the cover art were shown correctly.	Major update to code required for saving images to the database. Non-critical bug so no change will be made yet.	N/A
05	K1	Sign out books to a school	Sign out 5 books to a school.	Normal	The books signed out will have had a loan id created and assaigned to them. The list of books at a school is checked with the reports system	The correct 5 books were signed out to the selected school.	None required	N/A
06	K1	Sign in books from a school	Sign in/return the 5 books used in the previous test.	Normal	The books will not appear in the list of books assigned to the school.	the books were no longer signed out to the school.	None required	N/A
07	K3	View details about a school stored in the database.	select "Ashford Hill Primary School" in the school details page and view their details.	Normal	The stored details will be displayed.	Correct details were displayed	None required	N/A
08	K3	Modify details about a school and check that those changes were saved into the database	modify the number of pupils at "Ashford Hill Primary School" and view the details to confirm they saved	Normal	The modified details will be displayed.	Modified details were displayed	None required	N/A
09	K3	Add details about a school and check the added details were saved into the database	create a new school in the database	Normal	The new details will be saved correctly	New details were displayed	None required	N/A
10								
11	N/A	enter incorrect length isbns to see how the system handles them	enter an isbn that is too short into the book details menu	Erroneous	The program wont crash	The user was returned to the homepage.	None required	N/A
12	N/A	enter incorrect length isbns to the book details menu see how the system handles them	enter an isbn that is too long into the book details menu	Erroneous	The program wont crash and will handle the error gracefully	The user was returned to the homepage. No error message was displayed.	added to the bug tracker - to be fixed at a later date	N/A

System Maintenance

System Overview

When the system is opened by the user, they are greeted with a log on form. This log on form is setup using entry boxes to collect the users' login credentials to allow the user to log on and access the system. The user will type in their log on credentials and press the `Enter` key or click the `Log In` button on the form. If the credentials match any of the stored credentials in the database, the user will be taken to the system homepage.

The system homepage is the way the user accesses the other parts of the system from one central form. It has buttons for the user to press to take them to the various other parts. One of the other parts is a way to look up the details of a book by its ISBN. The user enters the ISBN into a form that appears when the `Book Details` button on the homepage which then passes it through to the details form. The details form displays the books title, authors, genre, release/published date, age rating, blurb and cover image.

Another section is the part for loaning books out to schools and returning them to the base location. The first part of this sub-system is a form to select the school that is having books exchanged with using a drop-down menu and a pair of buttons either `Sign Out` books to that school or `Sign In` books from that school. The second part of the sub-system is a form for entering the ISBNS that are part of the exchange. The ISBNs are entered into an entry box and stored visually in a list box with the corresponding titles in another list box next to it. Once the ISBN have been entered and the submit button has been pressed, the system processes all the books and adds them to the new loan in the database for signing out books or to the existing loan for returning them to the base.

Another section is the part for retrieving data about a school in the database. The first part of this sub-system is a form for selecting the school from a drop-down menu and three buttons to select which mode the second form should open in, whether the user is viewing or editing a school or adding a new school into the system.

Another section is the method for generating reports based on the data in the database. Through the use of a set of drop down menus to select the report type and any other parameters for the report, the user can generate one of a few reports pertaining to the service.

Another section is the calendar part of the program. This is for managing events such as stock checks and exchanges through the system. It is made of a simple, normal calendar layout to view a months' worth of events at once and dropdown menus to select the month and year. Double clicking on an event in the calendar will bring up a detailed view of the events including all the details stored along with notes about the event.

The last section of the system is a settings menu for making changes to the functionality of the system including file locations for icons and the database file, and the aesthetic theme of the program. If the user is logged in as an administrator then an option to add a new user to the system is also present as a button to bring up the relevant form.

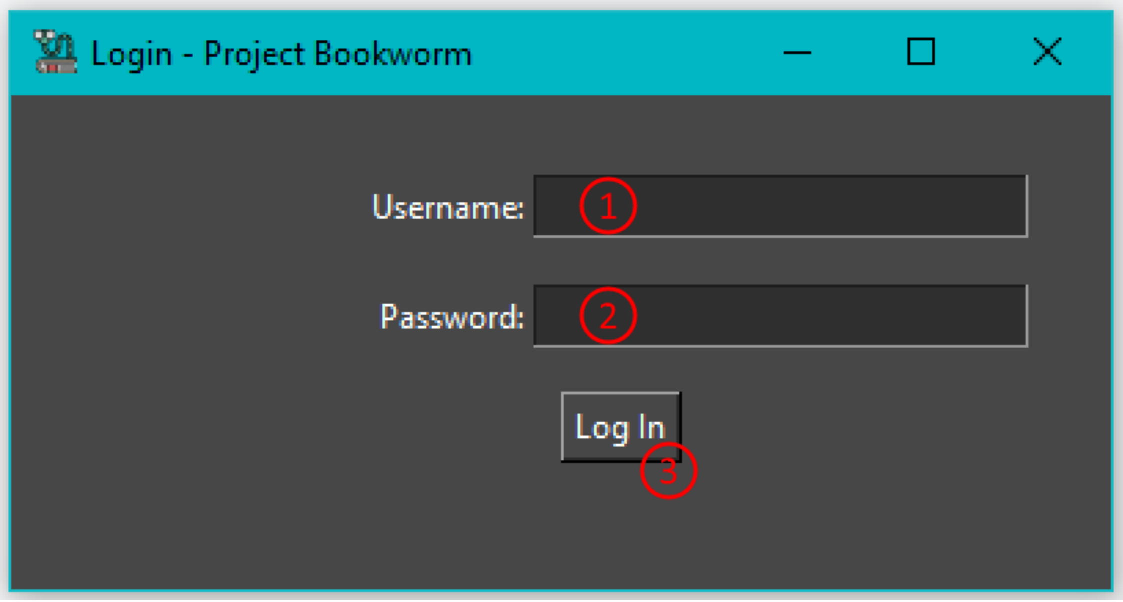
Sampled Detailed Algorithms

Guide

Accessing the System via the Login Menu:

1. Start the program.
2. Select the **Username** field (Fig 28, Ref ①) with the mouse or the *Tab* key.
3. Enter the username for the system.
4. Either using the *Tab* key on the keyboard or the mouse to navigate to the **Password** field (Fig 28, Ref ②).
5. Enter the password associated with that username. As a hint, in case the password has been forgotten, the password requirements are at least one of each of the following:
    a. Uppercase Characters.
    b. Lowercase Characters.
    c. Numbers.
    d. Special Characters.
6. Press the *Enter* key on the keyboard or click the **Log In** button (Fig 28, Ref ③) with the mouse to attempt an authentication.
7. If the username and password are correct, the user will be taken to the programs **Homepage** (Fig 29.). If not, an error box will appear saying that the authentication failed and that the user should try again.

Fig 28. Login menu with references for user guide.



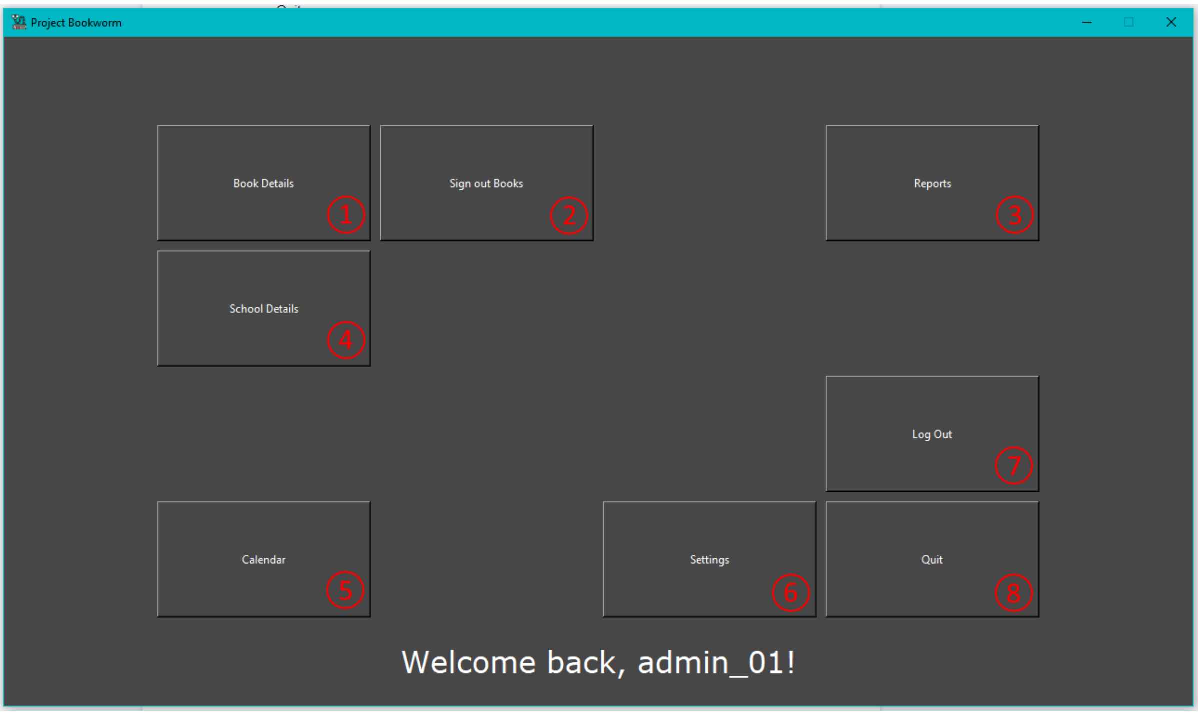
Homepage:

Once signed in, the user will be brought to the homepage of the program. From this page, they can access the various parts of the system through the use of clearly labeled buttons. Each main function of the program has its own button including:

- **Book Details** (Fig 29, Ref ①)
- **Loaning/Returning Books** (Fig 29, Ref ②)
- **Reports** (Fig 29, Ref ③)
- **School Details** (Fig 29, Ref ④)
- **Calendar** (Fig 29, Ref ⑤)
- **Settings** (Fig 29, Ref ⑥)
- **Log Out** (Fig 29, Ref ⑦)
- **Quit** (Fig 29, Ref ⑧)

Log out (Fig 29, Ref ⑦) will return the user to the login menu so the system is secure in the case of unauthorised people with physical access to an unlocked computer in the library. **Quit** (Fig 29, Ref ⑧) will close the program completely and return the user to the desktop.

Fig 29. Homepage screenshot with references for user guide.



Welcome back, admin_01!

Viewing and Editing Book Details:

The first button on the homepage is labelled Book Details. This feature is for looking up the details of a book by its ISBN.

1. Click the **Book Details** button (Fig 29, Ref ①) on the Homepage.

2. Enter the ISBN either with a Barcode Scanner or typing it into the entry field (Fig 30, Ref ①) and pressing the **Enter** key or the **Submit** button (Fig 30, Ref ②).
3. The system will now produce the following details about the book.
    a. Title (Fig 31, Ref ①)
    b. Author (Fig 31, Ref ②)
    c. Genre (Fig 31, Ref ③)
    d. Release Date (Fig 31, Ref ④)
    e. Age Rating (Fig 31, Ref ⑥)
    f. Blurb (Fig 31, Ref ⑧)
    g. Cover Image (Fig 31, Ref ⑨)
4. The details are currently displayed inside Entry fields which the user can edit. Using these, the user may correct errors or add missing details for that book.
    a. To save any changes that have been made, click the **Save Changes** button (Fig 31, Ref ⑩) to commit the modifications to the database. These saved details will be prioritised over data from the internet when the system needs any data about the book.
    b. If the user changes their mind about saved modifications, clicking the **Revert to Online Data** button (Fig 31, Ref ⑫) will delete any details about the current book from the database allowing the system to use the data from the Google Books API.
5. Once finished in the window, clicking the **Close** button (Fig 31, Ref ⑪) will return the user to the programs Homepage.

Fig 30. ISBN Entry form.

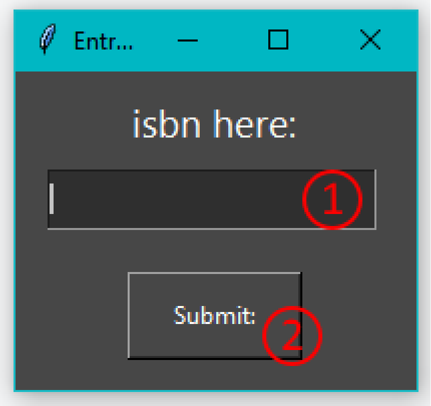
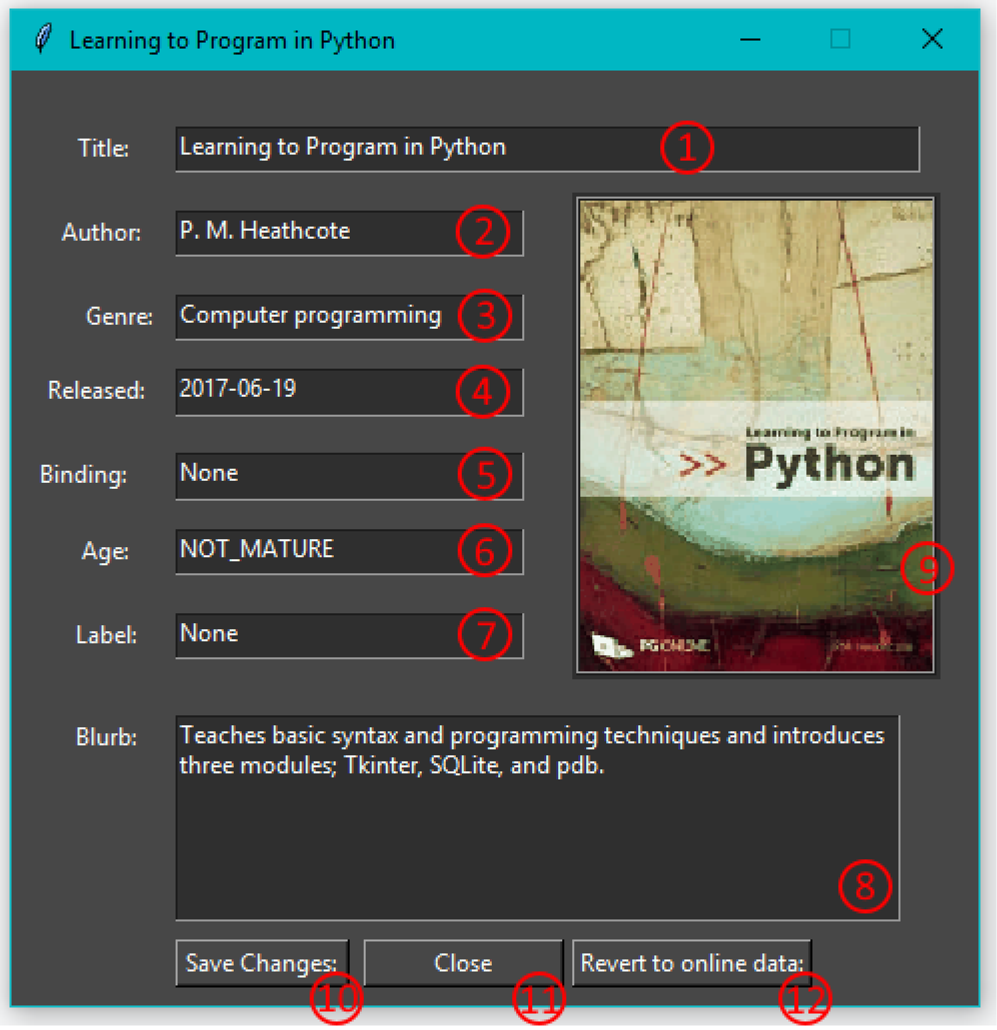


Fig 31. Book Details form.



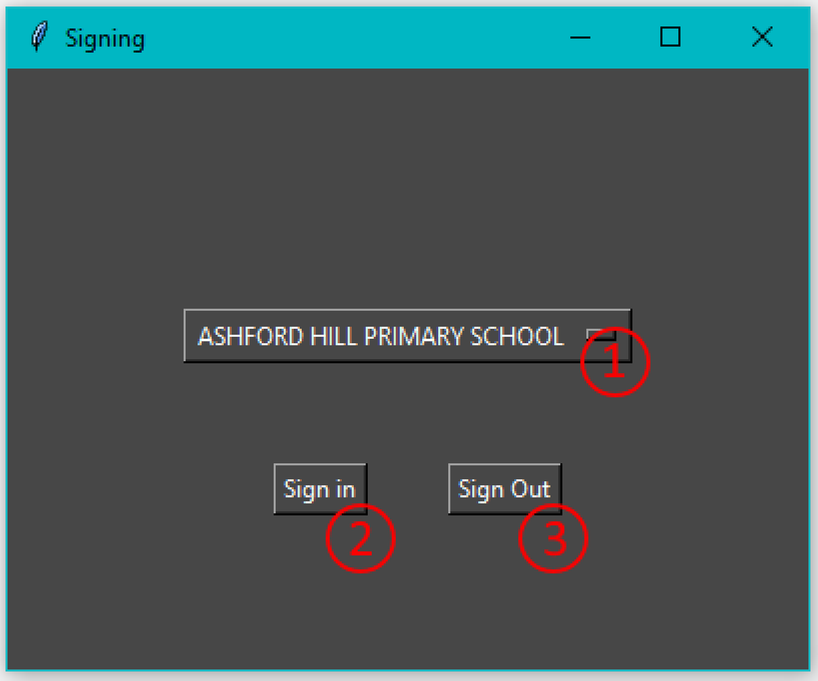
Learning to Program in Python

Title:	Learning to Program in Python
Author:	P. M. Heathcote
Genre:	Computer programming
Released:	2017-06-19
Binding:	None
Age:	NOT_MATURE
Label:	None
Blurb:	Teaches basic syntax and programming techniques and introduces three modules; Tkinter, SQLite, and pdb.

Loan Books out to a School

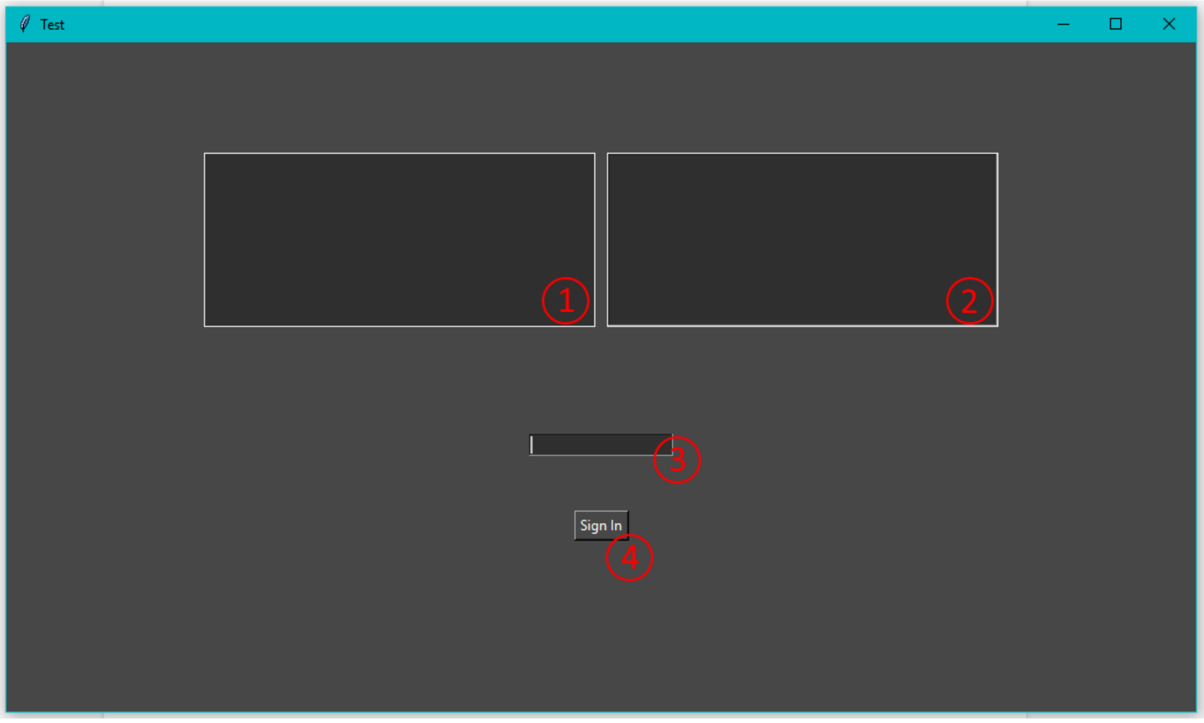
1. Click the **Sign Out Books** button (Fig 29, Ref ②) on the Homepage.
2. Select the school from the drop down menu (Fig 32, Ref ①) .
3. Click the **Sign Out** button (Fig 32, Ref ③)

Fig 32. Loaning starting menu.



4. Select the **ISBN entry** field (Fig 33, Ref ③)
5. Enter the ISBN:
    a. If using a keyboard, type it in and press the Enter key
    b. If using a barcode scanner, scan the barcode.
6. The ISBN that has been entered will appear in the List Box on the left (Fig 33, Ref ①) and the book title will appear in the List Box on the right (Fig 33, Ref ②).
7. Repeat **Step 5** until all the books have been entered.
8. To confirm the loan, click the **Sign Out** button (Fig 33, Ref ③).

Fig 33. Multiple Books entry form.



Return Books from a School

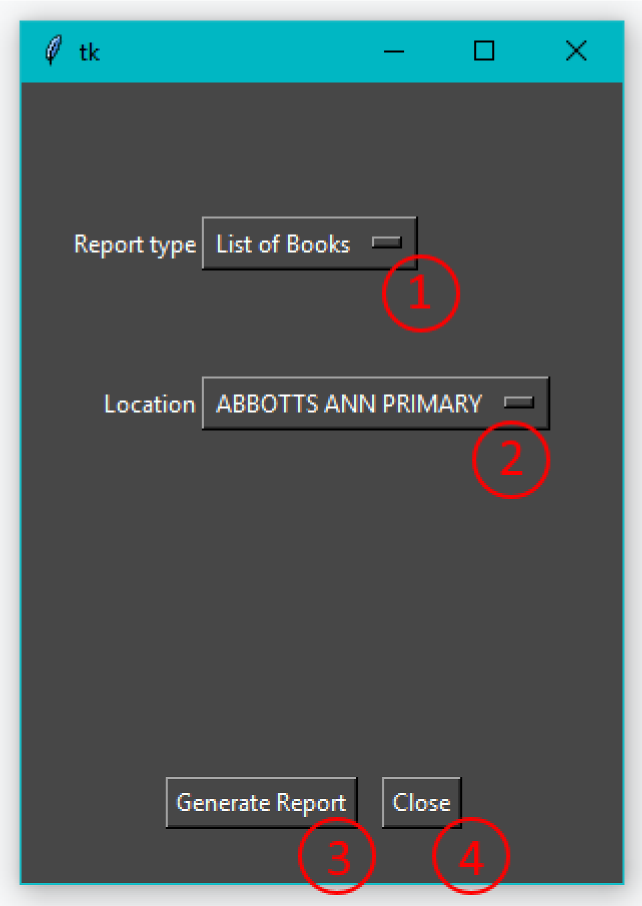
1. Click the **Sign Out Books** button (Fig 29, Ref ②) on the Homepage.
2. Select the school from the drop down menu (Fig 32, Ref ①) .
3. Click the **Sign In** button (Fig 32, Ref ②)
4. Select the **ISBN entry** field (Fig 33, Ref ③)
5. Enter the ISBN:
   a. If using a keyboard, type it in and press the Enter key
   b. If using a barcode scanner, scan the barcode.
6. The ISBN that has been entered will appear in the List Box on the left (Fig 33, Ref ①) and the book title will appear in the List Box on the right (Fig 33, Ref ②).
7. Repeat **Step 5** until all the books have been entered.
8. To confirm the loan, click the **Sign In** button (Fig 33, Ref ③).

Generating Reports:

One of the features provided by the program is a process for generating reports of useful data based on statistics from the system.

1. Select the report wanted from the **drop down menu** (Fig 34, Ref ①) at the top of the form.
2. If the selected report needs a location to gather data about, select that from the **second drop down** menu (Fig 34, Ref ②) that has appeared.
3. Once all the drop down menus have been filled out, the **Generate** button (Fig 34, Ref ③) will appear at the bottom. Click the button to generate a report in the form of a table in a new window.
4. On the window with the report table there is a button labelled **Print** (Fig 35, Ref ①) . This button will convert the report in the window into an Excel spreadsheet saved into a folder in the users documents folder. With the spreadsheet version, you can send it to anyone who needs it (ie a list of books at a school can be sent to the school in question) or it can be printed out onto paper for filing or taking to a location.

Fig 34. Reports Wizard

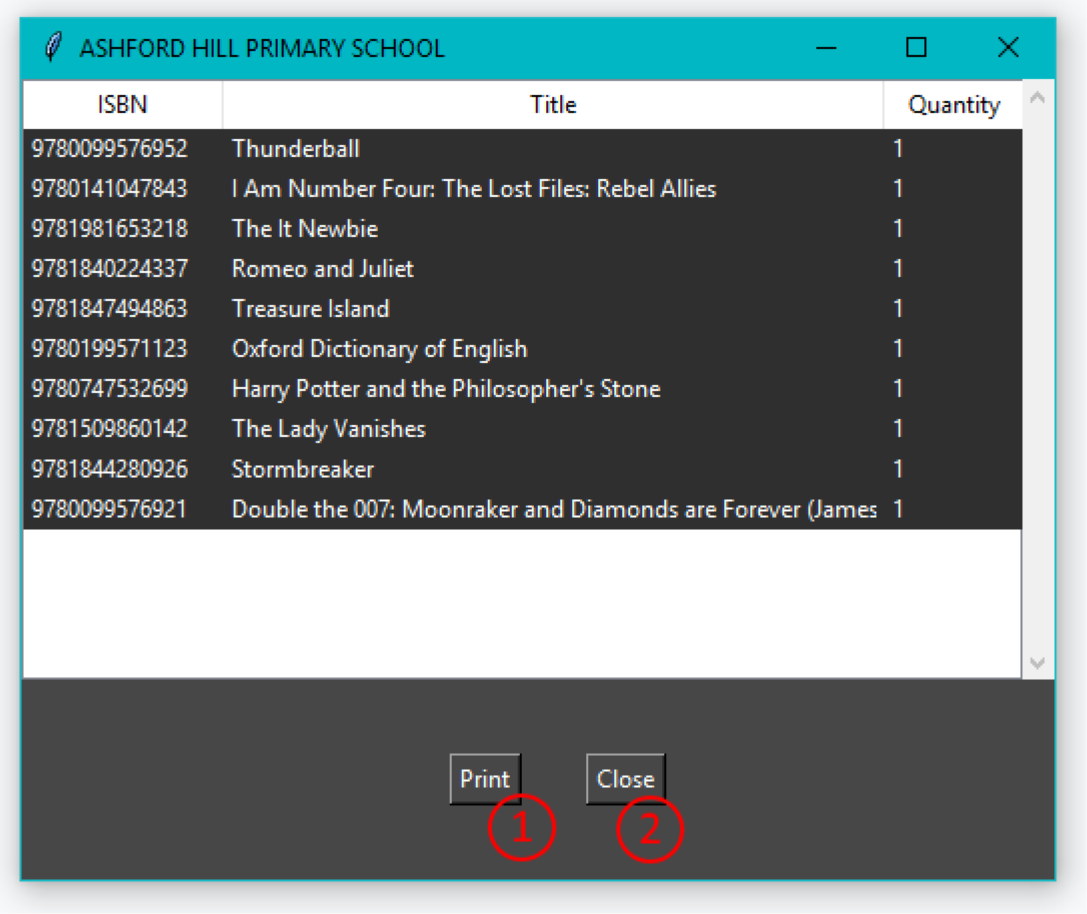


tk

Report type: List of Books

Location: ABBOTTS ANN PRIMARY

Fig 35. Example of a generated Report



ASHFORD HILL PRIMARY SCHOOL

ISBN	Title	Quantity
9780099576952	Thunderball	1
9780141047843	I Am Number Four: The Lost Files: Rebel Allies	1
9781981653218	The It Newbie	1
9781840224337	Romeo and Juliet	1
9781847494863	Treasure Island	1
9780199571123	Oxford Dictionary of English	1
9780747532699	Harry Potter and the Philosopher's Stone	1
9781509860142	The Lady Vanishes	1
9781844280926	Stormbreaker	1
9780099576921	Double the 007: Moonraker and Diamonds are Forever (James	1

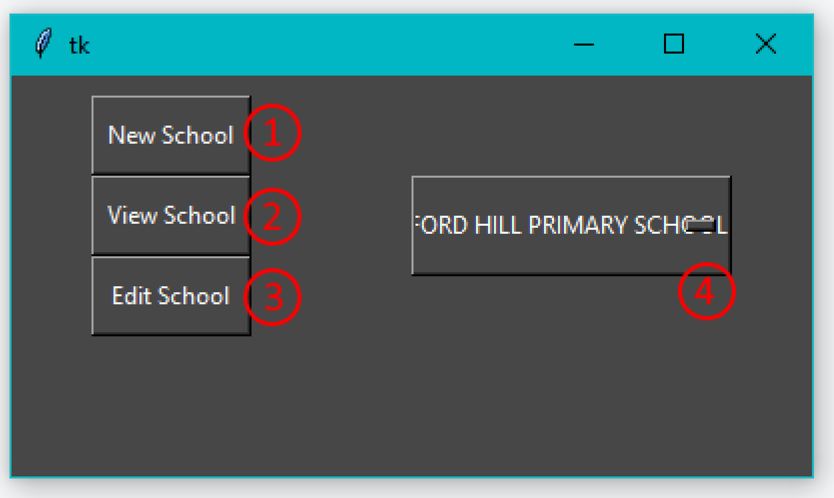
Adding, Editing and Viewing School Details.

One of the buttons on the homepage is for viewing the details about a school that is subscribed to the SLS service.
The details include:

- School Name (Fig 37 a/b/c, Ref ①)
- Headteachers name (Fig 37 a/b/c, Ref ②)
- Date of last exchange (Fig 37 a/b/c, Ref ③)
- Phone Number (Fig 37 a/b/c, Ref ④)
- Department for Education number (Fig 37 a/b/c, Ref ⑤)
- Number of pupils at the school (Fig 37 a/b/c, Ref ⑥)
- Number books allowed per pupil (Fig 37 a/b/c, Ref ⑦)
- Number of books allowed total (Fig 37 a/b/c, Ref ⑧)
- Address (Fig 37 a/b/c, Ref ⑨)

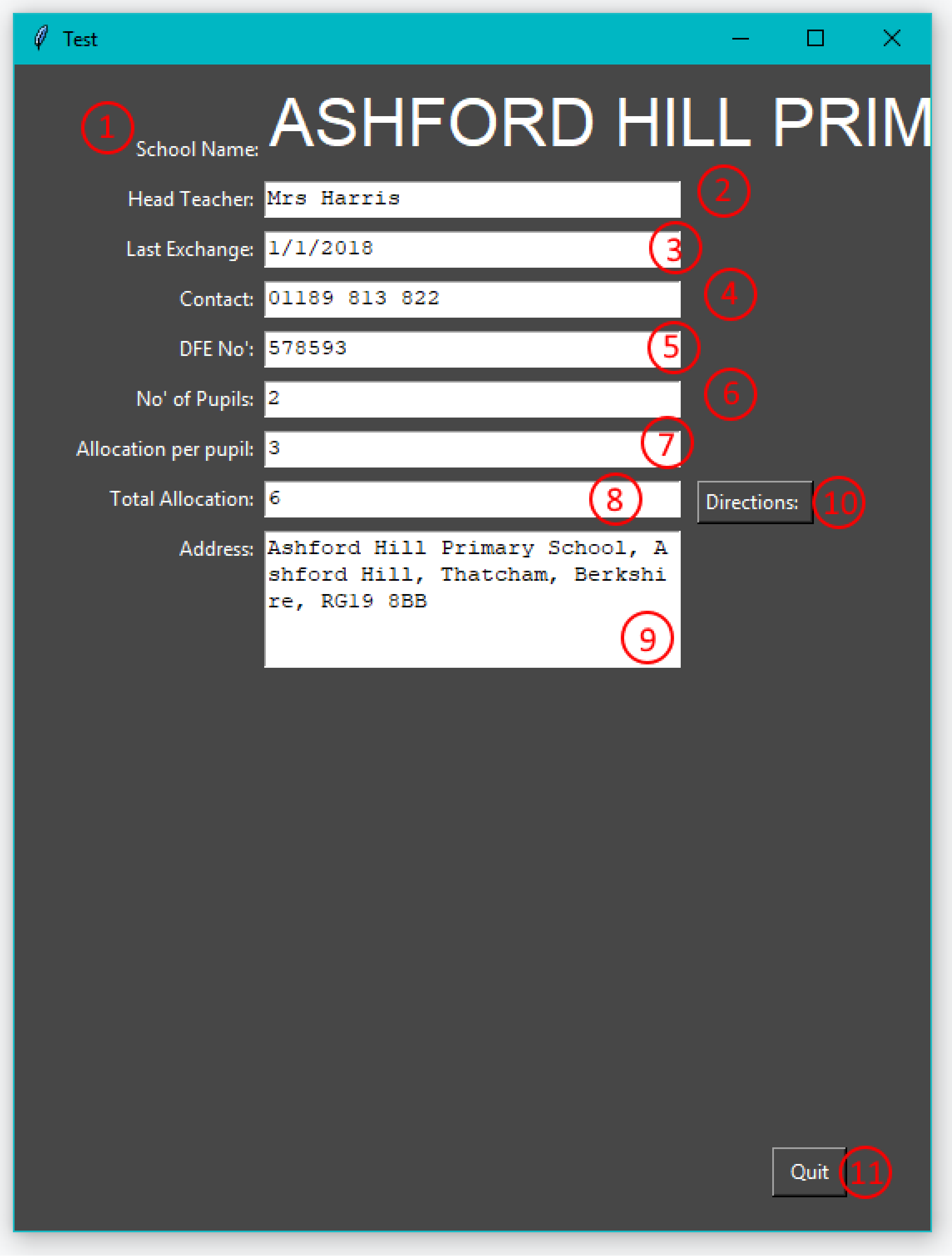
1. Click the **School Details** button on the homepage.
2. Select the school from the **drop down menu** (Fig 36, Ref ④).
3. Using the buttons on the form, choose whether you are viewing the details (**View School,** Fig 36, Ref ②), changing the details (**Edit School,** Fig 36, Ref ③) or adding a new school to the system (**New School,** Fig 36, Ref ①).

Fig 36. School Details form.



4.
    a. **Viewing the details.** On this screen (Fig 36 a) there will be no option to save so do not make any changes to the details here. A **Directions** button (Fig 37a. Ref ⑩) is available next to the address field to get directions to the address in that box. This button will open up a new tab in the default browser with the Google Maps page for that address.

Fig 37 a. Viewing School Details



Head Teacher: Mrs Harris

Last Exchange: 1/1/2018

Contact: 01189 813 822

DFE No': 578593

No' of Pupils: 2

Allocation per pupil: 3

Total Allocation: 6

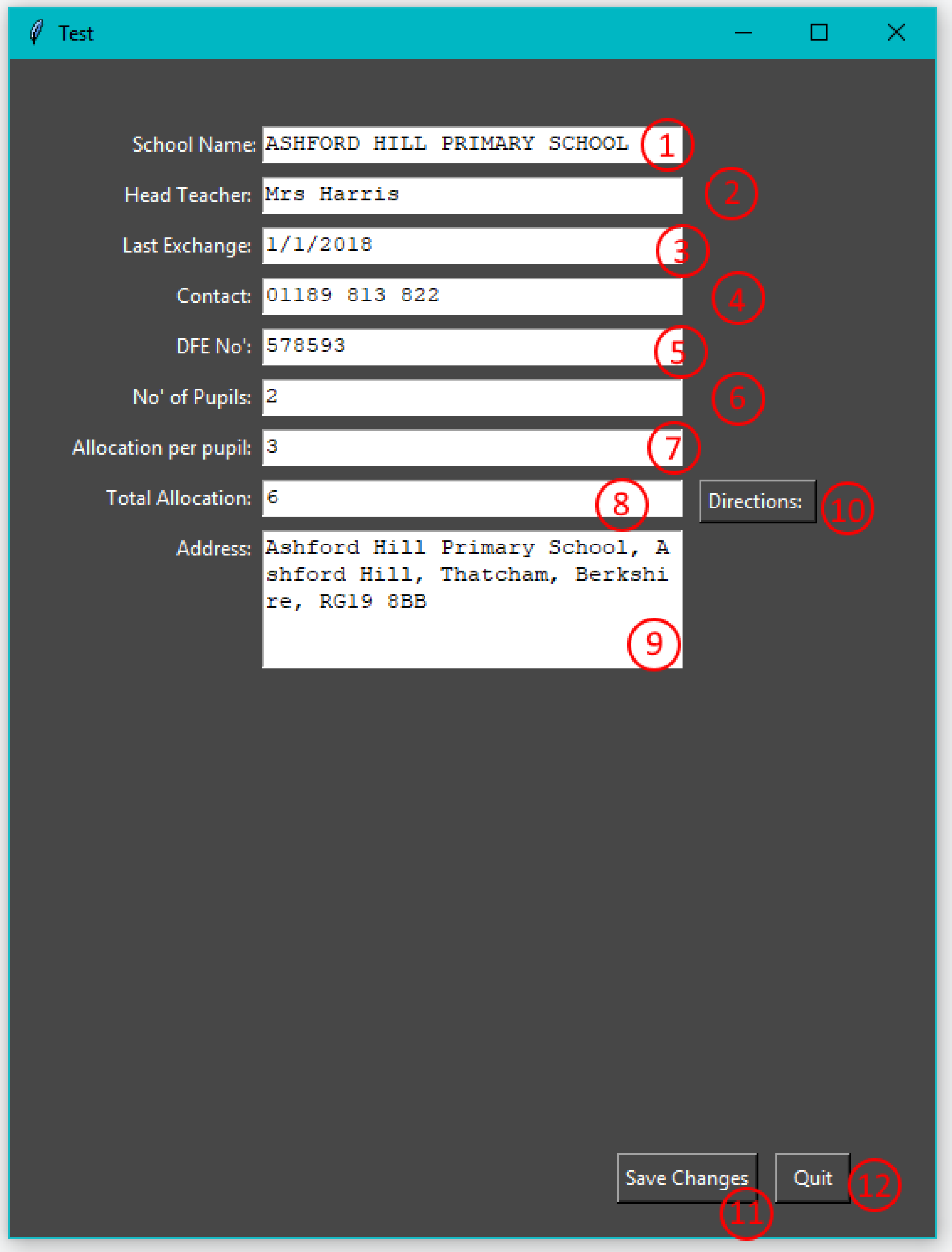
Address: Ashford Hill Primary School, Ashford Hill, Thatcham, Berkshire, RG19 8BB

b. **Modifying the details.** On this screen (Fig 36 b), there will be **entry fields** (Fig 37b. Refs ①-⑨) pre-filled out with the details of the school selected in the previous form.

   i. To change them, simply click on the field and change the contents using the keyboard.

   ii. Once done, press save at the bottom to commit those changes to the database.

Fig 37 l



Test

School Name: ASHFORD HILL PRIMARY SCHOOL

Head Teacher: Mrs Harris

Last Exchange: 1/1/2018

Contact: 01189 813 822

DFE No': 578593

No' of Pupils: 2

Allocation per pupil: 3

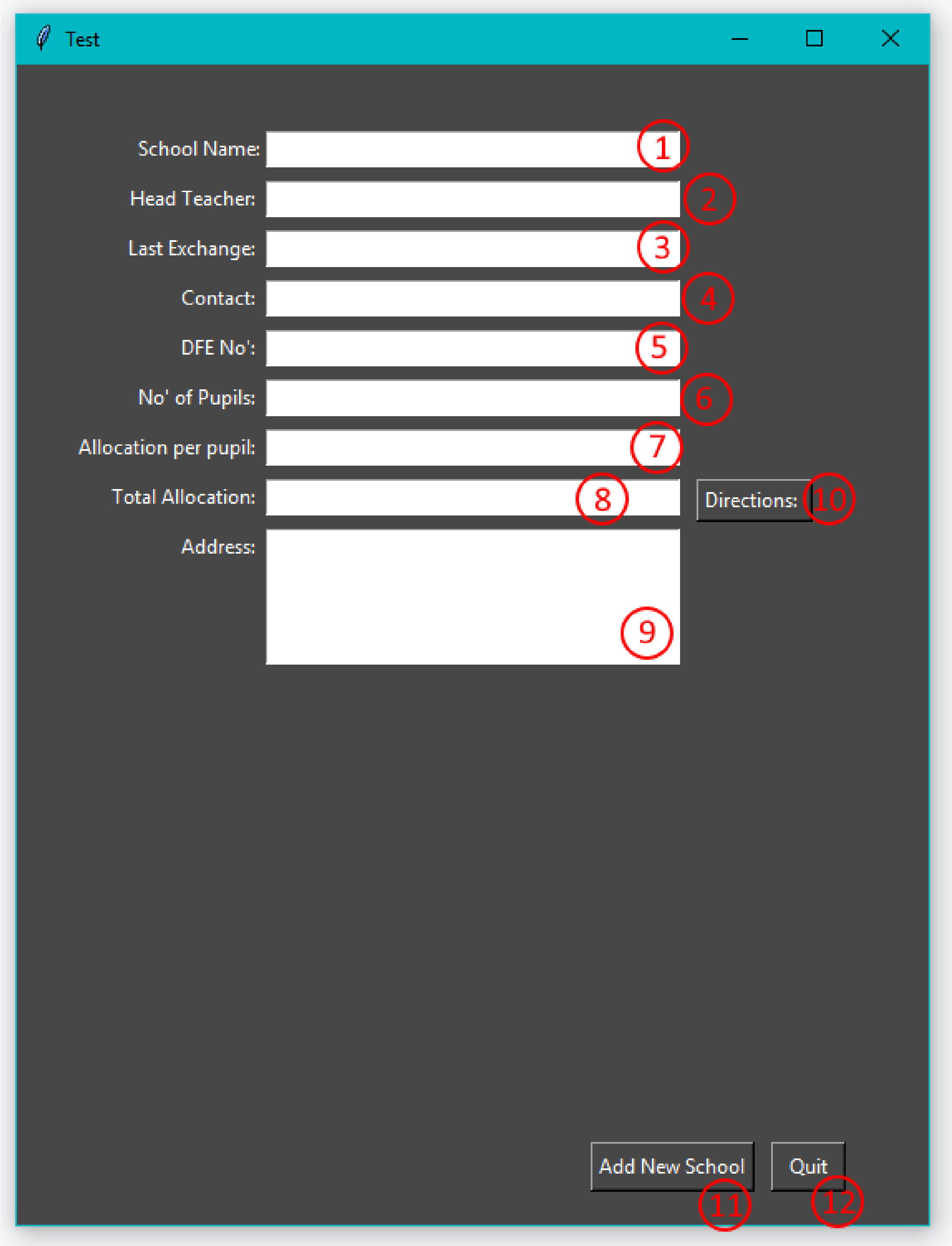
Total Allocation: 6

Address: Ashford Hill Primary School, Ashford Hill, Thatcham, Berkshire, RG19 8BB

c. **Adding a New School.** On this screen (Fig 36 c), the **entry fields** (Fig 37c. Refs ①-⑨) will all be empty.

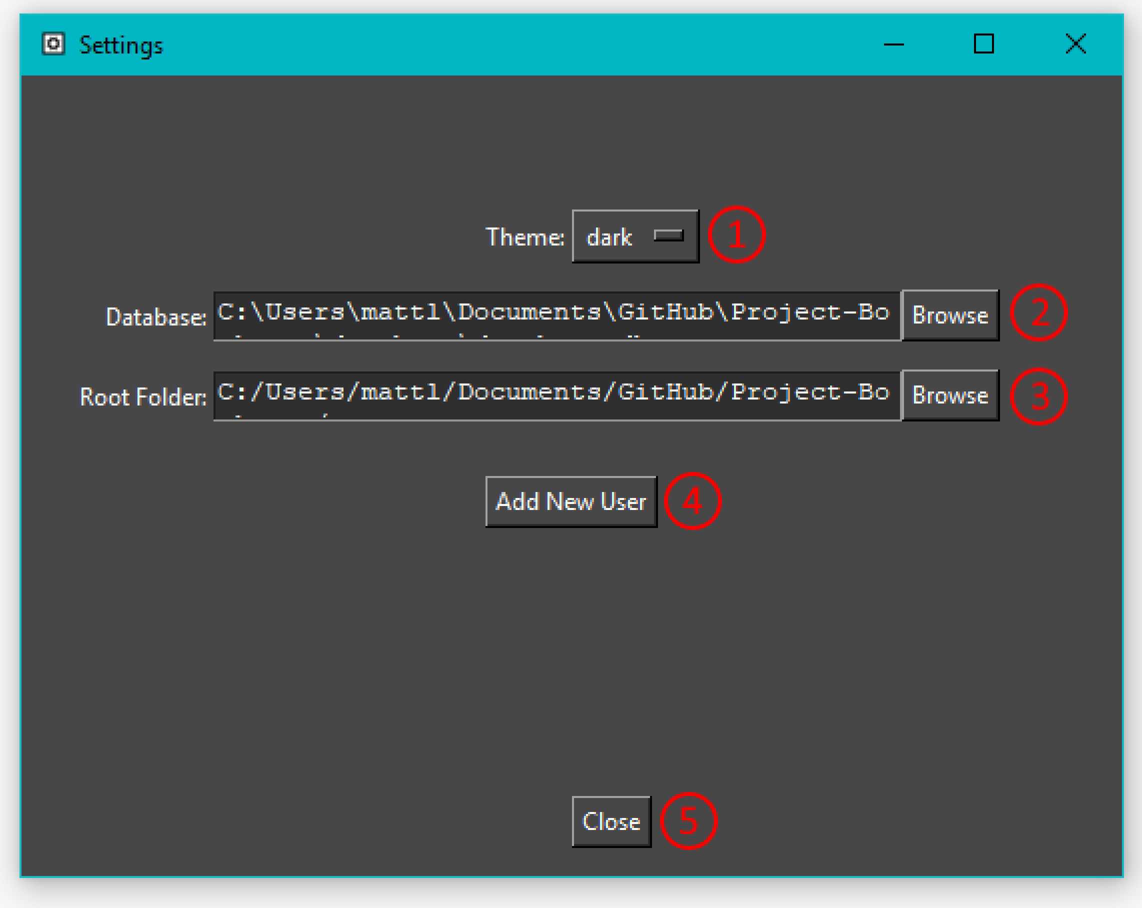
   i. To add the details, click on each **entry field** and type in the corresponding value.

   ii. Once done, press the **Add New School** button (Fig 37c. Refs ⑪) at the bottom to commit the changes to the database.



Settings:

Fig 38. Settings Menu



Theme: dark

Database: C:\Users\mattl\Documents\GitHub\Project-Bo

Root Folder: C:/Users/mattl/Documents/GitHub/Project-Bo

Changing the program theme:

1. Login to the system with any user.
2. Click the **Settings** button on the homepage.
3. Using the **drop down menu** (Fig 38, Ref ①) at the top, select the new theme.
4. To apply the theme, click the **Close** button (Fig 38, Ref ⑤) at the bottom of the menu.
5. You will be returned to the Homepage with the new theme having been applied.

Changing the Database Location:

1. Login to the system with any user.
2. Click the **Settings** button on the homepage.
3. Click the **Browse** button (Fig 38, Ref ②) next to the Database entry field.
4. Using the file browser, select the database file.
5. Click the **Close** button (Fig 38, Ref ⑤).

Changing the Assets and Temporary Files Location:

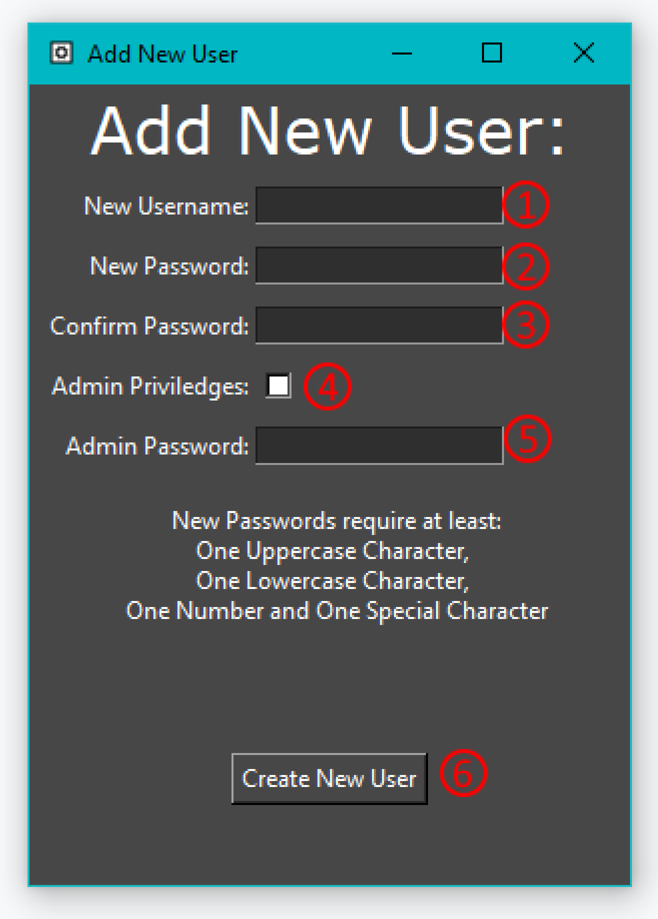
6. Login to the system with any user.
7. Click the **Settings** button on the homepage.
8. Click the **Browse** button (Fig 38, Ref ③) next to the **Root Folder** entry field.

9. Using the file browser, select the database file.
10. Click the **Close** button (Fig 38, Ref ⑤).

Adding a New User:

1. Sign in to the system as an Admin user.
2. Click the **Settings** button on the Homepage.
3. Click the **Add New User** button (Fig 38, Ref ④) (note: this will only be available if the user is signed in as an administrator)
4. Fill out the required details
    - Username (Fig 39, Ref ①)
    - Password (Fig 39, Ref ②)
    - Password Confirmation (Fig 39, Ref ③)
    - Admin Rights check box (Fig 39, Ref ④)
    - Admin Password (Fig 39, Ref ⑤)
5. Click submit. If any of the details don’t match the requirements, an error message will appear instructing you on what needs to be changed to correct it.
6. Once successfully submitted, a message box confirming the setup is complete will appear.
7. Check that the new user has been created successfully by:
    - Closing the settings menu
    - Logging out of the program using the **Log Out** button on the Homepage.
    - Log back in using the new user credentials.

Fig 39. Add New User menu



New Passwords require at least:
One Uppercase Character,
One Lowercase Character,
One Number and One Special Character

Evaluation

Key Objectives:

1. The system must be able to sign books in and out to different schools.

    I successfully managed to implement this feature into my program. There is a dedicated button on the Homepage to start the process of loaning or returning books to and from schools. The loans are created in the database and the books are tied to the loans.

    The part of this Objective that I could have improved upon is the menu for entering the books to exchange. This form is rather rough around the edges with a lack of labelling regarding the input box and lists. With more time, I could have improved this by adding labels to describe the different UI elements which would make the form more intuitive.

2. The system should keep track of books in-stock and at different the schools.

I managed to include this feature in the program however the system doesn't know what copy of a book is being processed, only that it is one of the copies at the start location in whatever process is being performed.

I could have improved on this by having unique barcodes for each copies on the books so that system can track the location of each actual book instead of just how many copies are at a location.

On the other hand, doing it by number of copies is fine since it is easier to code and reduces the work required in order to use the system as the SLS library assistants don't have to print a unique barcode for each book they have.

3. The system should have a database to store details about the schools that subscribe to the service.

    I successfully achieved this objective by storing all of the data in the system in a relational SQLite database. By using table relationships, I have removed duplicate data with everything normalised.

    One way I could have improved on this objective is by hosting the database on a server. This would allow me to create the program in a way that multiple users could process exchanges of different machines at the same time improving the efficiency of the library.

    However, since the SLS library assistants rarely have multiple schools visiting for exchanges at the same time, this feature isn't as necessary as the main functionality and won't hinder the processing of exchanges to a significant extent.

4. The system must be able to produce reports about the loan history of individual schools and the service as a whole.

    I successfully created most of the reports that the sample users requested in the questionnaire. With more time available for development, I could improve on this by completing all of the reports. But since the most important ones have been completed, it would be acceptable for an initial version of the system to be missing some of the less vital reports.

5. The system should have a login policy to protect sensitive data about the subscribed schools.

    I successfully implemented a secure login procedure upon launch of the program. The passwords are securely hashed in the database and associated with the username so the system can verify the users login without the passwords being visible to anyone who can view the database file in a separate program. However, the user id of the last person to log in is stored in the plain text settings file so that the program can work out whether or not they are admin but this is a small flaw since someone could manually edit it to be a user id with admin rights.

6. The graphical user interface must be user friendly and intuitive.

   I believe I have met this objective by using large buttons with simple yet descriptive labels which should make the use of these buttons clear for users to follow in the use of the system. If the user does become confused, I have written a straightforward step-by-step guide on the various functions of the system.

   To improve on the intuitiveness of the system, I could spend more time with someone who would use the system to make the design clearer to the target users.

7. The system should protect the database from SQL Injection attacks that could otherwise cause irreversible damage to the structure and contents of it.

   This feature is currently untested but based on the way the SQL database queries were set up with the user input, it should automatically escape any malicious queries that have been attempted to be entered into the system.

8. The system should be able to look up the details of book from the internet or a database with modified details and use those details in various places.

   I successfully achieved the first half of this objective (looking up details on the internet) by using the Google Books API to access the data about the books in the system. By using an API key and the Python Requests module, I am able to search for the details by ISBN.

   The second objective was achieved by saving book details if that have been modified by the user. By only saving the changed details, I am able to reduce the size of the database by a significant amount.

   In order to improve on this objective, I could have spent more time on the algorithm that parses the data from the Google Books API to verify the correct details are being used as, occasionally when the Google Books API returns the wrong book as the first one in the search results, the wrong book details are being used. Another way I could improve is to add a method to save the image of the book cover in the database so that it doesn’t have to resort to the error image.

Additional Objectives:

1. The system should use Object Oriented Programming where possible.

   I have successfully achieved this objective as every window and form in the program is coded as a Class object which matches the idea of Object Oriented Programming.

2. The system could have a settings menu where the user can customise their experience (e.g. colour themes) and change the locations of any required files.

   This objective was successfully achieved with a simple settings menu using drop down menus, wizards and standard file selection menus that are used by any Windows program that need to select a file which the users should be used to from other applications.

   To improve, I could make some of the more important settings, like database file location, administrator only settings so the everyday users can’t interrupt functionality by accidentally changing an important setting.

3. The system could have a calendar system to manage school visits.

   I have mostly completed this objective by making a Calendar that can show events on the correct day in the month and change the events and dates to match the selected month and year.

   The way I could improve upon it is to add a create an event button to the calendar window to allow the users to book in dates for exchanges and site visits.

4. The systems reporting function could calculate various statistics about their visits, borrowing history and amount of lost/damaged books.

   This objective was mostly achieved with one statistic report about the current allocation numbers which can be generated through the reports menu.

   I could improve on this by making more report types able to be generated in the reports menu.

5. The system could have a lookup system for books pertaining to a particular topic with a way to check their location.

    This objective was not achieved in the development of the system. This was because the nature of the objective is such that the system need an understanding of the topics of all of the books in the system which, in the timeframe of development, was un-achievable.

6. The system should be able to make use of a barcode scanner for faster signing in and out of books.

    This objective was successfully achieved. All of the entry fields where ISBNs need entering are able to make use of a standard barcode scanner. This is important for the functionality of the system as it dramatically reduces the time it takes to loan out and return large numbers of books compared to having to type them in manually.

7. The system should be simple to setup and install on the target users computers.

    This objective was unable to be achieved. In the time frame of development, researching and implementing both executable files and installers was time-consuming with inconsistent results on the execution of the compiled files.

Overall, I think I was able to achieve more than just enough of the key and additional objectives defined in the Objectives for the Proposed System however there are a few ways I could improve upon but these were mostly down to a lack of effective time management in the development process.

Program Code

See folder attached. The file to run is the login.py module.