Web Application Project
Summer School, 2022

Milestone Submission[1]: 9am Monday 30th January 2023

Final Submission: 5pm, Friday 10th February 2023

Worth: 45% of COMP636 Marks

Scenario

You have been asked to develop a web-based library management system for Waikirikiri Library. It is used to manage books, borrowers and loans. You are provided with a Flask app with some template files, or you can build on your existing Jinja + CSS code.

It has the following functional requirements:

You will provide two interfaces:

1. The first, not requiring any log in, should allow borrowers to search the catalogue to find out which books are available, whether they are on loan or not (and if they are, when they are due back).
2. The second interface, not requiring any log in, is focused on library staff, allowing them to issue books to borrowers and return books that have been on loan.

Using the first interface (public) borrowers must be able to:

- Search the books by title and/or author, allowing for partial text searches
- See the availability of all copies of a book, whether a copy is on loan and, if so, the due date.
- List all books available in the library at the /booklist route Include book title, author, category and year of publication.
    - This is the only route is required for the milestone submission see submission information.

Using the second interface (staff) library staff must be able to:

- Search the available books by title and/or author
- See the availability of all copies of a book, whether a copy is on loan and, if so, the due date.
- View the details of a borrower, searching by name or by library card id (borrower id).
- Update the details of a borrower, except their id number.
- Add a new borrower
- Issue a book to a borrower. Physical Books can only be loaned once at a time (i.e., are not already on loan, eBooks and Audio Books can be loaned multiple times simultaneously)

[1] This involves an early submission of the GitHub repository and PythonAnywhere set-up to ensure that all parts of the project are correctly configured for marking. See the marking section for details.

- Return a book that has been on loan.
- Display a list of all overdue books & their borrowers. Group all overdue books by borrower (i.e., display the details of each overdue borrower only once).
- Display a list (Loan Summary) showing the number of times each book has been loaned in total.
- Display a list (Borrower Summary) showing all borrowers and the number of loans (past and current combined) in total they each have had.

The standard loan period is 28 days (due date). Books are reported as overdue (list of overdue books) once they have been on loan longer than 35 days.

General requirements

The public interface must be displayed when the default route is accessed (/)

The staff interface must be accessed by an alternative route (/staff)

Borrowers and staff do not need to log in.

The functions for each role should be accessible from the home page for that role (route / for public, route /staff for staff).

Do not provide a link to the staff interface from the public interface.

See also the Project Requirements later in the document.

Report

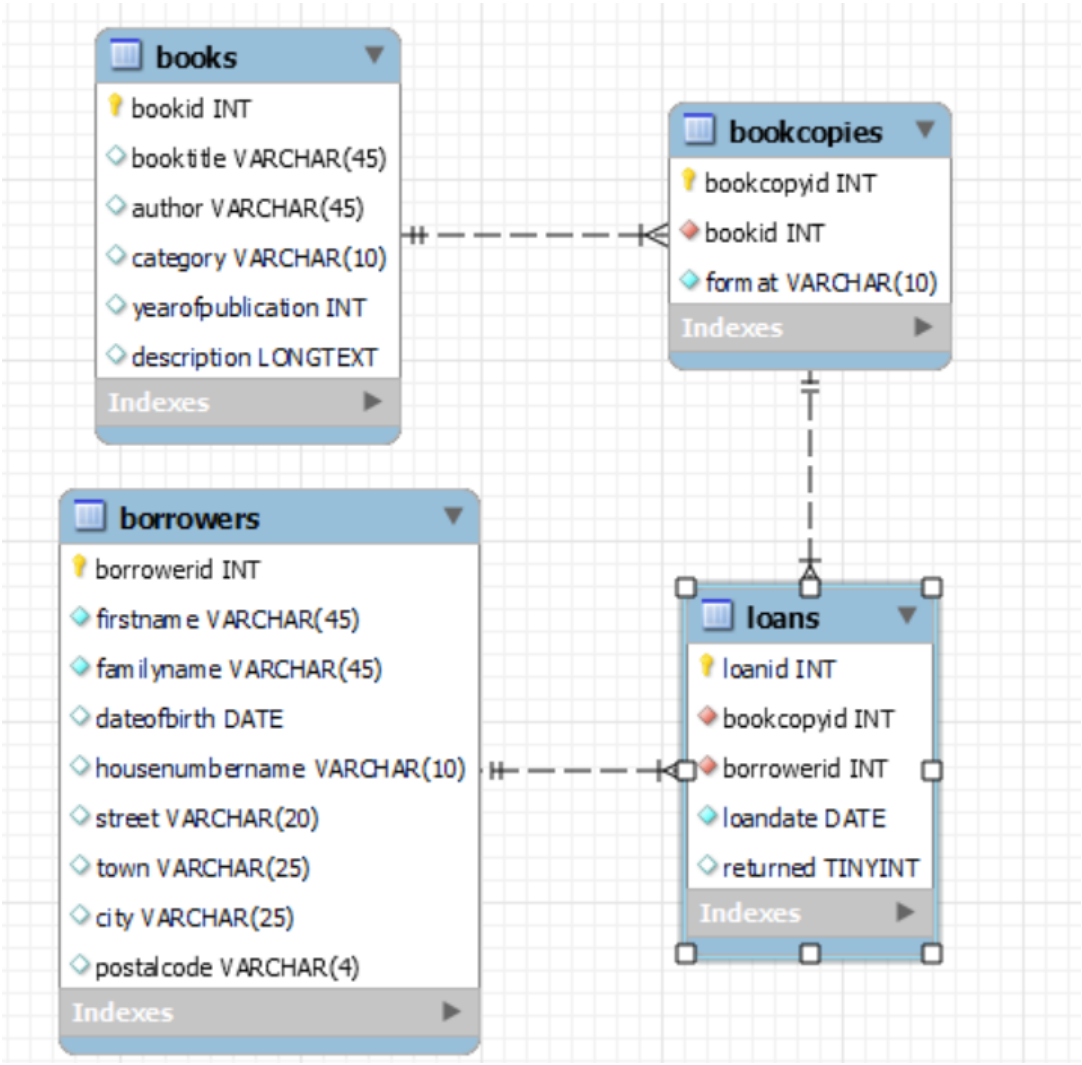
Include a brief project report:

- Outlining the structure of your solution (routes & functions). This should be brief, but be sure to indicate how your routes and templates relate to each other and what data is being passed between them, do not just give a list of your routes.
- Detailing any assumptions and design decisions that you have made. For example, did you share a page template between public and staff or use two separate pages? Did you detect GET or POST method to determine what page displays? If so, how and why? Note these types of decisions and assumptions as you work.
- A discussion that outlines what changes would be required if your application was to support multiple library branches. This should include:
    - Changes required to database tables (new tables and modifications to existing tables)
    - Changes required to the design and implementation of your web app.
    - (Do not implement these changes in your app.)

This report must be created using GitHub Markdown and saved in the README.md file of your GitHub repository.

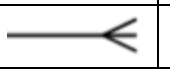
Data Model

Note: The lines shown indicate only the relationships between the tables. They do not indicate which fields are included in the table relationships. See the table below the diagram for the field details.



Foreign keys

Foreign keys enforce 'referential integrity' – which means that a value in the child table field must match an entry in the parent table field, e.g., a **bookid** cannot be entered into the **bookcopies** table unless it is already in books. This prevents entering bookcopies for non-existent books (and ensures loans are issued to only existing borrowers and existing copies).

Parent table and field(s)		Child table and field(s)
books.bookid		bookcopies.bookid
bookcopies.bookcopyid		loans.bookcopyid
borrowers.borrowerid		loans.borrowerid

Loans that have been returned have the value 1 in the returned field.

Project Requirements

You must:

- Use only the COMP636 technologies (Python & Flask, Bootstrap CSS, MySQL). Do not use SQLAlchemy or ReactJS (or other similar technologies) in your solution.
- Use the provided SQL file to create the database within your MySQL database & PythonAnywhere. This also creates initial data in the database.
- Create a Flask web app that:
    - meets the functional requirements
    - is appropriately commented
    - connects to your database on PythonAnywhere
    - provides appropriate routes for the different functions
    - provides templates and incorporates HTML forms to take input data
    - uses Bootstrap CSS to provide styling and formatting
- Include your report as outlined above
    - This report must be created using GitHub Markdown and saved in the README.md file of your GitHub repository.
- Create a private GitHub repository that contains:
    - All Python, HTML, images and any other required files for the web app
    - A requirements.txt file showing the required pip packages.
    - An extract of your database from MySQL – schema and data
    - Your project report as the README.md document
    - Your repository must have a .gitignore file and therefore not have a copy of your virtual environment
    - You must add lincolnmac (computing@lincoln.ac.nz) as a collaborator to your private GitHub repository
- Host your system (including database) using PythonAnywhere
    - You must add lincolnmac as your "teacher" via the site configuration

Project Hints

Create your GitHub repository first and create all your required code/files in your local folder. Regularly commit and push changes to your GitHub repository.

PythonAnywhere is case sensitive so test your app early – we will mark the PythonAnywhere version of your app.

Spend some time sketching the structure of your application before you start developing. Think about which features could share the same (or nearly the same) templates. Remember that you can nest templates.

Take note of your design decisions, compromises, workarounds, etc. for your report as you develop your web app. Otherwise you may struggle to remember all of the issues you worked through afterwards.

Milestone Submission (10 marks)

This is an early submission of the basic web app set-up and a single route to ensure that all apps are correctly configured before the final web app is submitted.

Submit via the link on the Learn COMP636 page:

- Your PythonAnywhere URL (e.g., joebloggs.pythonanywhere.com/)

Your GitHub repository name (e.g., joebloggs99)

For this submission:

- Your GitHub repository setup
- You must have a the /booklist route working
- Your database set-up on PythonAnywhere
- Your app hosted on PythonAnywhere
- Granted access (lincolnmac as teacher) to PythonAnywhere
- Granted access (lincolnmac / computing@lincoln.ac.nz as collaborator) to GitHub repository

At this submission we will check your GitHub, PythonAnywhere and database setup

Project Set-up Element	Marks Available
GitHub Repository set up and shared	3 marks
PythonAnywhere web app hosting correctly configured, including home URL and database setup	5 marks
/booklist route and page	2 marks
TOTAL	**10 marks**

Final Submission (90 marks)

Submit via the link on the Learn COMP636 page:

- Your PythonAnywhere URL (e.g., joebloggs.pythonanywhere.com/)
- Your GitHub repository name (eg, joebloggs99)

Marking

Project set-up (milestone submission)	10 marks
General project aspects	35 marks
Functional project aspects	55 marks
TOTAL	100 marks

General Project Aspects:

Project Element	Marks Available
Project Report – Part 1: • outlining the structure of your solution (routes & functions) • detailing any assumptions and design decisions that you have made. • report created using GitHub Markdown and saved in the README.md file of your GitHub repository Project Report – Part 2: • Outline of changes required to support multiple branches	20 marks • Almost half of these marks are assigned to Assumptions and Design Decisions 10 marks • Up to 3 marks of these 30 are awarded for spelling, punctuation, grammar and presentation
•	
Consistent 'Look and Feel' (interface, Bootstrap styling & templates)	5 marks
TOTAL	**35 marks**

Functional Project Aspects:

Within each of the functional areas we are looking for:

- The functionality specified works
- Well commented and formatted HTML, SQL, and Python code throughout
- A user interface that looks and functions to a professional standard, including colour and styling choices
- Primary key identifiers are for internal system use only and should not be made visible[2] to system users
- Data Validation on Forms
- Functionally correct Python
- Well-structured SQL queries
- Appropriate naming, both of variables and labels
- Appropriate and well formatted content

[2] Exception: it may be useful to display borrower IDs when updating borrowers and issuing books, to distinguish between any borrowers with the same name

Functionality	Applies to	Expectations	Marks Available
Access	Public	/ route exists and operates	1
	Staff	/staff route exists and operates	2
Search Books	Public and Staff	Book Search allows specification of which fields are searched. Search accepts partial text matches in search terms. Results show book availability. Accessible to both public and staff users via their respective interfaces.	20
Search/Add/Edit Borrower	Staff	A new borrower can be added by staff member. Appropriate interface to find borrowers to edit Existing borrowers' details can be edited.	12
Issue book to borrower	Staff	Physical Copies can only be lent once, eBooks and Audio Books can be issued an unlimited number of times	4
Return Books	Staff	A book (copy) on loan can be returned.	2
Overdue Books Report	Staff	Report showing Overdue books and who has borrowed them	5
Loan Summary Report	Staff	Report showing how many times each bookcopy has been borrowed, grouped by book	5
Borrower Summary Report	Staff	Report listing borrowers and the number of loans (past and present combined) they have had.	4
TOTAL			**55**

Functionality not required

Your web app is **not** required to:

- Delete any records from the database.

This feature may be desirable in a full system but is outside the scope of this assessment.

You may modify data in your database for testing purposes and may add new data. Markers may also add additional or alternative data to your system as part of the marking process.

Do **not** modify the schema (the model shown on page 3).