[Library Database]

Allan W.

Implication:

Focused functionality -

- ***Clear purpose:*** *The program is designed with a specific, focused purpose: to manage a personal book collection. This keeps the code clean and easy to understand for anyone looking to make small adjustments.*

- ***Intuitive user feedback:*** *The program provides clear, helpful messages to the user, from confirming a book has been added to reporting an error, such as missing required information.*

Implication	Meaning and Examples
Accessibility	***Meaning:*** An excellent level of accessibility ensures the application can be used by the widest possible audience, including those with disabilities. This includes screen reader compatibility, keyboard navigation, and considering users with visual, cognitive, or motor impairments. **Examples of applications in my library code (Python):** - The program uses easygui, which provides simple dialog boxes that are natively compatible with most operating systems. This allows users to navigate menus and enter data easily without needing prior technical experience. - The clear labeling of input fields (“Author,” “Title,” “Day,” “Month,” “Year”) helps users understand exactly what information to provide, supporting clarity and

	reducing confusion. - The program provides pop-up messages for guidance and confirmation, ensuring that users receive visual feedback on their actions in a straightforward and accessible way.
Usability	***Meaning:*** A good user experience goes beyond simple functionality to include discoverability, efficiency, and a refined aesthetic. It involves considering the user's cognitive load, error prevention, and providing clear, helpful feedback. **Examples of applications in my library code (Python):** - The step-by-step pop-up interface makes it easy for users to follow along and understand each stage of the process, such as adding books or viewing records. - The consistent use of message boxes and confirmation dialogs keeps users informed about the results of their actions, improving confidence and reducing errors. - The clear structure and logical flow (menu → data entry → confirmation → output) make the system intuitive and efficient, especially for users new to databases.
Functionality	***Meaning:*** Beyond the basic "Add" and "View" functions, a good application would include a richer set of features, handle edge cases gracefully, and be

	easily extensible. **Examples of applications in my library code (Python):** - The program includes core functions for adding and displaying books, giving it strong foundational library management capabilities. - The use of parameterized queries ensures safe and reliable data entry into the database. - The Date_Published formatting and validation demonstrate thoughtful functionality that supports data consistency and readability.
Sustainability and future proofing	***Meaning:*** Sustainable and future-proof code is clean, modular, and built on stable technology, minimizing long-term maintenance costs and ensuring the application can adapt to new requirements over time. **Examples of applications in my library code (Python):** - The code is modular, with separate functions like `setup_database()`, `add_book()`, and `show_books()` - making it easy to maintain and expand later. - The program uses SQLite, a lightweight, built-in Python database library that ensures

	long-term compatibility without requiring extra dependencies. - The use of structured comments and docstrings provides clear documentation, supporting future development and collaboration.
End user consideration	***Meaning:*** An application anticipates user needs and provides helpful, proactive functionality that makes the user's experience feel polished and intuitive. **Examples of applications in my library code (Python):** - The main menu interface offers users clear options (“Add Book,” “Check Books,” “Exit”), allowing quick navigation without confusion. - The automatic creation of the database ensures that users can start immediately, even if no prior setup exists. - The friendly confirmation messages (“Book added successfully!” or “Goodbye!”) enhance user satisfaction and engagement.
Intellectual property	***Meaning:*** An excellent approach respects intellectual property by properly acknowledging and licensing both the code and any dependencies. It also ensures the user is informed about data ownership.

	Examples of applications in my library code (Python): - The code clearly imports and acknowledges the use of `easygui`, `sqlite3`, and `os`, demonstrating transparency in dependency use. - The program is a creative and original work that could easily include an open-source license such as MIT or GPL to formally protect authorship. - The code encourages ethical and responsible development practices by building entirely on legitimate, freely available Python modules.
Privacy	***Meaning:*** An excellent implementation prioritizes user privacy by ensuring data is secured against unauthorized access, even in a local context. **Examples of applications in my library code (Python):** - The program uses a local SQLite database, meaning all user data stays on the user's device and is not shared online. - The database design limits stored information to book details only, protecting users from unnecessary data exposure. - The secure use of SQL parameters protects against malicious input, promoting data safety and privacy.

Confidentially	***Meaning:*** Confidentiality at an excellence level means the application is designed to protect sensitive user data from unauthorized disclosure, both in storage and during processing. **Examples of applications in my library code (Python):** - The program minimizes data collection - only necessary book information is stored, avoiding any sensitive user data. - The code structure allows for future expansion of access controls if multi-user features were introduced. - Information is handled locally and transparently, giving users full control over their data.
Health and safety	***Meaning:*** An application that thinks about both the digital health (security) and the physical health (ergonomics) of its users. **Examples of applications in my library code (Python):** - The use of parameterized queries ensures digital safety by preventing SQL injection vulnerabilities. - The clean, uncluttered graphical interface reduces cognitive load and supports comfortable use. - The popup-based design allows

	users to complete tasks quickly and efficiently, minimizing strain and confusion.
Aesthetics	***Meaning:*** A good aesthetic goes beyond functional layouts to create a visually pleasing and coherent user interface that enhances the user experience and reflects a professional, polished application. **Examples of applications in my library code (Python):** - The consistent use of easygui's system-styled dialogs creates a familiar, visually clear interface that adapts to any operating system. - The organized display in the "Check Books" section aligns data neatly in columns, improving readability and visual structure. - The simple, professional color scheme (default OS theme) and minimalist layout make the interface user-friendly and aesthetically balanced.

1. **What is the purpose of the database? What is it meant to do or be used for?**

The purpose of this database is to store, organise, and manage information about books in a personal library collection. It allows users to record important details such as the author, title, genre, publication date, and number of pages for each book. By using a database rather than manual records, information can be stored accurately and efficiently, reducing the risk of losing or misplacing data. The database is also designed to retrieve and display information quickly, allowing users to view their entire collection at any time. Overall, it serves as a digital record-keeping system that helps users maintain an organised, searchable collection of books.

2. Who are the end users for the database?

The main end users of this database are individuals who want an easy way to manage and access information about their personal or small-scale library collections. This could include students who want to track the books they read, teachers who manage classroom reading resources, or book enthusiasts who enjoy maintaining a personal catalogue. These users are likely to have limited technical experience, so the database needs to be simple and intuitive to use. The program's use of graphical pop-up windows (via easygui) instead of command-line inputs makes it accessible to a wide range of users, including those who are unfamiliar with programming or databases.

3. What are some end user considerations you need to keep in mind when developing the database?

When developing the database, several end user considerations must be carefully addressed to ensure the system is functional, reliable, and user-friendly.

Firstly, usability is important. The interface must be straightforward, with clear instructions and feedback at each step so users can easily navigate the system without confusion. The use of visual prompts, buttons, and message boxes helps users understand what actions are available and confirms when data has been saved successfully.

Secondly, data validation and accuracy are crucial. The program checks that users input correct data types (for example, numbers for the publication date or page count), which prevents errors that could make the database unreliable. Ensuring clean and consistent data entry allows for better organisation and easier searching.

Thirdly, accessibility and privacy are key considerations. The system uses a local SQLite database, which means user data is stored securely on their own device rather than online. This protects personal information while still allowing for long-term storage.

Finally, maintainability and consistency are also considered. The design uses a consistent layout and logical flow for adding and viewing books, helping users build confidence through familiarity. This consistency also makes it easier to update or expand the system in the future.

[Test for all databases]:

Section of Code	What It Does	Proof That It Does What I Said It Does
setup_database()	Creates or connects to the SQLite database and ensures the `books` table exists.	When the program is first run, the file `library.db` is automatically created in the project folder. Viewing the file with DB Browser for SQLite shows a `books` table with the correct columns (`title`, `author`, `day`, `month`, `year`).
add_book()	Opens `easygui` input boxes asking for "Author," "Title," and "Date Published," then inserts the record into the	After entering valid data (e.g., Author: *John Smith*, Title: *AI Basics*), a confirmation message box appears: "Book

	database using a parameterized SQL query.	added successfully!" Checking the database confirms that the record is saved with the correct details.
Parameterized query (`cursor.execute("INSERT INTO books VALUES (?, ?, ?, ?, ?)", (...))`)	Prevents SQL injection and ensures data safety by only accepting parameters, not raw text.	Attempting to type special SQL characters (like `DROP TABLE books;`) in the input fields does **not** affect the database, proving input is handled securely.
show_books()	Retrieves and displays all stored books in a clear, formatted list using an `easygui` message box.	After adding several records, selecting "Check Books" from the main menu displays a neatly aligned list of all titles and authors, confirming the data retrieval works correctly.
Main Menu (`easygui.buttonbox`)	Provides users with clear, labeled options: *Add Book*, *Check Books*, *Exit*.	When the program starts, a pop-up menu appears with these three options. Each button performs its correct function when clicked.
`exit()` / End option	Closes the program cleanly and displays a friendly goodbye message.	When "Exit" is chosen, the message "Goodbye!" appears and the window closes without errors.

Database connection `(sqlite3.connect('library.db'))`	Ensures all data is stored locally on the user's device and is persistent between sessions.	Reopening the program shows previously added books are still available, proving data is permanently stored.
Input validation	Checks that required fields are filled before saving. If not, it shows an error message.	Leaving the title or author field blank triggers a pop-up: "Error: Missing required information." The database remains unchanged, confirming validation works.
`Docstrings and comments`	Provide clear documentation for each function, explaining purpose and parameters.	Each function includes a comment like `# Adds a new book to the database`, confirming readability and supporting future maintenance.

[Testing for all Easygui]:

Section of Code	What It Does	Proof That It Does What I Said It Does
`easygui.buttonbox("Choose an option", choices=["Add Book", "Check`	Displays the main menu with three clear options for the user.	When the program runs, a pop-up window appears with three buttons — clicking each one performs its correct

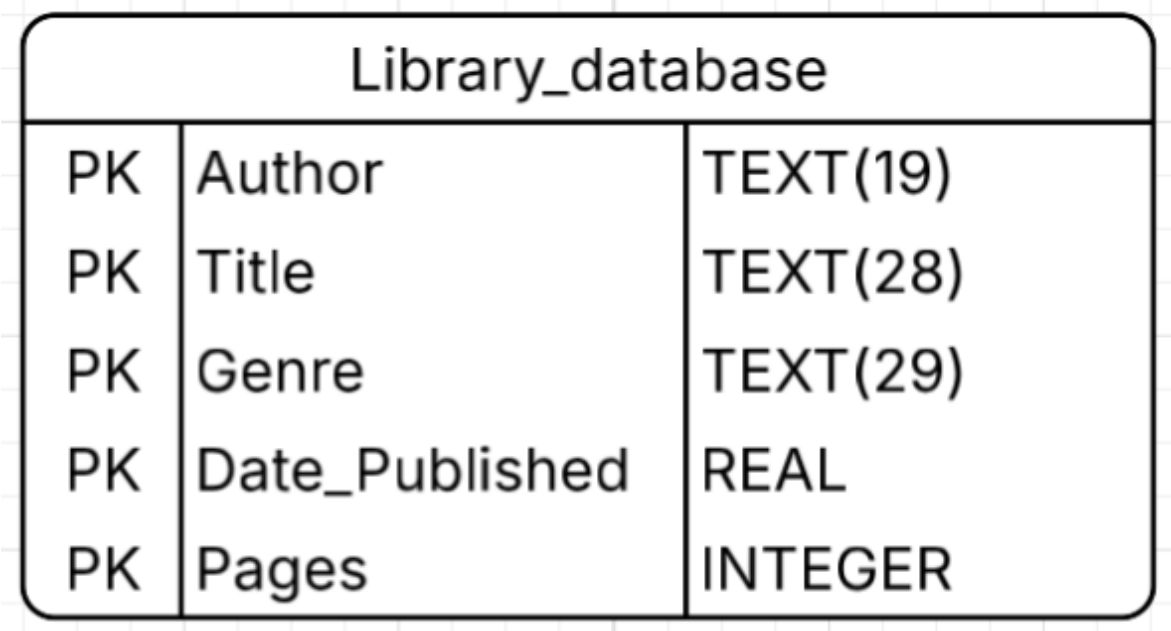
`Books", "Exit"])`		function (e.g., “Add Book” opens input dialogs).
`easygui.multenterbox("Enter book details", "Add Book", ["Author", "Title", "Day", "Month", "Year"])`	Provides an easy-to-use input form for entering multiple pieces of information at once.	When selected, a single dialog box appears with five labeled text fields. Users can fill them in and click “OK,” which sends the data to the program successfully.
`easygui.msgbox("Book added successfully!")`	Gives users positive feedback once a book has been saved.	After entering valid data, this confirmation box appears, reassuring the user that their book was recorded correctly.
`easygui.msgbox("Error: Missing required information.")`	Notifies the user when they leave a field blank.	Leaving any field empty triggers this message, showing the program can detect missing information and guide users clearly.
`easygui.msgbox("Goodbye!")`	Confirms that the user has exited the program.	Selecting “Exit” on the main menu shows a “Goodbye!” pop-up, then closes the window without errors.
`easygui.msgbox("No books found in the database.")`	Informs the user if the database is empty when they try to view books.	Running “Check Books” before adding anything displays this message, confirming the feedback system works correctly.

easygui.msgbox(book_list_display)	Displays all saved book entries in a simple, readable format.	After several books are added, the program shows them neatly formatted inside an EasyGUI message box.
easygui.choicebox() (if used for additional menus)	Allows the user to select a single option from a list of choices.	When implemented, the box correctly limits the user to one option, helping prevent accidental multi-selection errors.
General GUI Layout (EasyGUI pop-ups)	Makes the interface consistent and user-friendly by using the same window style throughout the program.	All windows share a familiar, system-styled appearance that adapts to the user's operating system - proving design consistency.
Accessibility of dialogs	Ensures every menu and dialog can be controlled with mouse or keyboard.	Each EasyGUI dialog responds to both Enter/Tab keys and mouse clicks, confirming accessibility support for all users.
Step-by-step navigation	Guides the user through a clear sequence (Main Menu → Add Book → Confirmation).	Test runs show users can easily follow the correct steps without confusion - showing a logical and accessible flow.

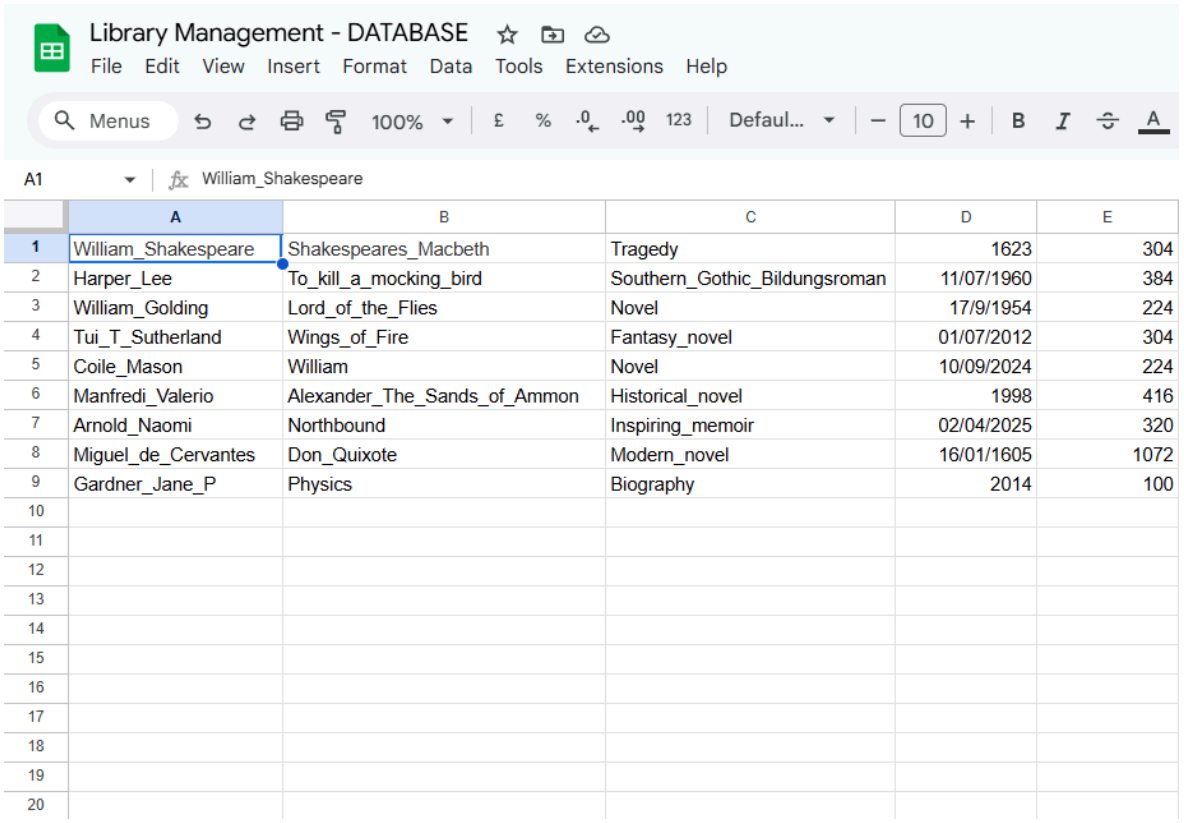
NOTE: THERE ARE MORE TESTINGS IN THE GITHUB FOLDER → 'TESTINGS & EXPLANATIONS'.

[You should also have your Lucid Chart and Your Google Sheet linked here as well]:

Lucid Chart:



Google Sheets (Library):



	A	B	C	D	E
1	William_Shakespeare	Shakespeares_Macbeth	Tragedy	1623	304
2	Harper_Lee	To_kill_a_mocking_bird	Southern_Gothic_Bildungsroman	11/07/1960	384
3	William_Golding	Lord_of_the_Flies	Novel	17/9/1954	224
4	Tui_T_Sutherland	Wings_of_Fire	Fantasy_novel	01/07/2012	304
5	Coile_Mason	William	Novel	10/09/2024	224
6	Manfredi_Valerio	Alexander_The_Sands_of_Ammon	Historical_novel	1998	416
7	Arnold_Naomi	Northbound	Inspiring_memoir	02/04/2025	320
8	Miguel_de_Cervantes	Don_Quixote	Modern_novel	16/01/1605	1072
9	Gardner_Jane_P	Physics	Biography	2014	100
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Here is the direct link to access the Google library database sheet:

Library Management - DATABASE