**Module 1 Assignment - Fundamentals of Software Development**

**Introduction**

**Abstract**

This purpose of this report is to outline the development process of 'CheckMate', a console-based library management system tailored for librarians to facilitate the simple efficient management of books. The application has been developed in Python, following PEP 8 coding standards. It incorporates the use of colour and formatting to create a user-friendly experience. The project has made us of AI as a tool for research and any usage has been clearly described at each point for clarity. The report presents the adapted scrum development used in developing the CheckMate solution, along with justifying the decisions made throughout the course of the project.

**Introduction**

CheckMate is a solution that is designed to facilitate the efficient management of books within a library. The application is aimed to be used by borrowers and librarians. For both users, the application will have a clear user-interface that considers accessibility. The application will provide the ability to search for books using certain identifiers, as well as allowing users to view all books. There will be a log-in system allowing higher privileges to librarian users, giving them the ability to add and delete books from the system. As well as this, Librarians will be able to check-in and check-out books for borrowers. Book data will be saved and loaded, with the potential to store a large volume of book data. Goals for the system state that it will be optimised for performance, ensuring quick data retrieval and efficient user interactions. The following report will outline the development process for the application.

**Software Development Life Cycle**

Initially, use of the waterfall methodology was considered. This methodology involves the adoption of a rigid and sequential process of development, where each stage is completed before advancement to the next, mirroring a waterfall[[1]](#footnote-1). There are typically five phases within development that are requirement gathering, design, implementation, testing and deployment[[2]](#footnote-2). This would be well-suited for the development of this library management system as the project scope is clearly outlined and use of the waterfall methodology lends itself to fixed and clear requirements[[3]](#footnote-3).

Kanban was also considered as an option. Kanban is characterized by the aim to make workflow visible[[4]](#footnote-4), allowing teams to easily adapt to improve their methods. In order to implement this, a Kanban board is used. This is a physical or digital representation of workflow through the use of columns, where tasks are progressed through stages as completed[[5]](#footnote-5). Kanban ultimately aims to enhance workflow by ensuring a steady delivery of tasks, representing progress clearly, and promoting collaboration.

Alternatively, Scrum was considered. Scrum is an agile methodology that puts emphasis on teamwork, self-organisation, and productivity over documentation. Development takes place over short periods known as sprints, where daily meetings take place to ensure visibility across all team members[[6]](#footnote-6). Scrum development teams have distinct roles that work together to create a solution. The Product Owner is responsible for creating a backlog, these are lists of features that are required of the solution[[7]](#footnote-7). The Scrum Master ensures a teams effectiveness by facilitating the self-organisation of a team through running meetings, problem solving and open communication[[8]](#footnote-8).

Ultimately, it was decided that the use of an adapted form of the agile methodology scrum would be better suited to development of the solution.

**Adapted Scrum**

In order to implement this methodology as a single developer, I will adopt a multifaceted role that encompasses Product Owner, Scrum Master, and Developer. In my capacity as the Product Owner, my objectives will include prioritizing the project backlog and ensuring any outputs match with project objectives. While taking on the role of Scrum Master, I will orchestrate Scrum ceremonies. This will include sprint planning, sprint reviews, and retrospectives. As the sole Developer, I will take on responsibility for completing tasks outlined in the sprint backlog and ensuring that any deliverables have met the specified acceptance criteria. My sprints will be each be 2 days in length and will run across weekends to allow adequate time to develop increments and ensure time to produce the documentation. Through these adaptations, I can use Scrum principles within the process of my development, despite the limitations of solo development.

**Coding Standards**

To ensure a high standard of code quality and consistency throughout the application, this project will adhere to coding conventions that are outlined in the PEP 8 style guide for python[[9]](#footnote-9). Each indentation level will use 4 spaces, with line length kept to a maximum of 79 characters. Imports will have one line each and will be placed at the top of the file. Whitespace and commenting will be used sparingly to ensure readable code. These rules, along with others that are outlined within the document will be kept ensuring a high quality of readable code.

**Planning, Analysis and Design**

**Project Requirements**

Due to the nature of the project, I was initially given a list of basic requirements. These have been summarised below.

* Books should have a title, author, and unique ID.
  + Title and author can be between 1 to 50 characters.
  + ID must be unique within the storage solution
* Functionality to add new books to the library.
* Search function implemented that is functional for title, author, or book ID.
* Functionality to remove books from the library.
* Ability to view a list of all books within the library.
* Check out and return system to track book availability.
  + Both features should report if a book does not exist.
* Save and retrieval of library data.
* User interaction implemented through the console.
* Fast saving, loading, and searching of books that takes no longer than 5 seconds.
* Implementation of two user types. These are a borrower and Librarian.
* User experience is optimised, and usage of colour is explored.
* Appropriate algorithms used to store and retrieve data that only use native input and output functions.
* Memory footprint is minimized.
* Accessibility is both considered and demonstrated within the application.
* Error checking is implemented.
* Problems are identified and overcome through innovation.

**Expansion of Requirements**

Within my simulated role of Product Owner, I expanded on these requirements as I considered functionalities that would be desirable for a library management system that would ultimately create a greater experience for the end-user.

* The application will have a simple menu system that is easy to navigate. This will allow a user to view any actions they can take within the program. After a valid input from the user, this will take them to carry out the action they desire or display an error clearly.
  + This will have a 'help' section that allows a user to learn how the application functions.
* There should be two kinds of users within the application a Librarian and a Borrower. Where the Librarian has superior privileges compared to the borrower.
  + Borrower
    - Borrowers should have access to view all books within the library and have the ability to search for any book or genre. Borrowers should log-in to the system and be able to view books they currently have checked out.
  + Librarian
    - Librarians should have access to view all books within the library and have the ability to search for any book or genre. They should log-in to the system and have higher privileges compared to the borrower. Librarians should add and remove books to the library. As well as this, Librarians should have the ability to check out a book and return them. When a book is checked out, a librarian should be able to assign them to a borrower to give visibility.
* There should be a method to log-in to either a borrower or librarian account.
  + Account creation and deletion feature to allow new users to join the library system.
  + There should be considerations for privacy, to prevent passwords from being visible when typed into the console.
  + The implementation of encryption should be considered to ensure a secure application.
  + The prevention of multiple incorrect attempts to log-in to prevent brute-force attacks.
  + A system to reset passwords.

* Books should have several unique attributes assigned to them. They should have a title, author, unique ID, and availability. Books should have a value to indicate how many copies of a certain title is available within the library as there may be several available. Books should have genres to allow users to search for specific types of books.
  + Title and author should be between 1 to 50 characters.
    - Any character should be allowed for either of these attributes as there are many titles that may involve unexpected values. For example, 1984 by George Orwell includes numbers within the title and implementing systems to prevent this may create difficulty for the end-user.
  + There may be several copies of the same book so this should be stored and incremented when a new book is added.
  + Book ID codes should be unique and short. Any values used should be simple for a user to ensure accessibility and prevent the application from creating additional problems for the end-user.
* There should be a method to add a new book into the library.
  + Attributes such as the title and author of the new book should be assigned by the user. Other attributes such as availability and unique ID should be automatically assigned and require no input.
  + Feedback should be given to the user upon success.
* There should be a method to remove a book from the library.
  + Feedback should be given to the user upon success.
  + If the book does not exist or could not be deleted the user should be notified.
* There should be a method to view all books within the library that will list all books that the library.
  + There should be a method to view books regardless of availability and options to view available books.
* There should be a method to check-out a book for a librarian that uses the ID for the book to carry out the action.
  + Feedback should be given to the user upon success.
  + If the book does not exist or could not be deleted the user should be notified.
  + A Borrower should be assigned to the check-out for a book, enabling visibility for the Librarian to manage all books.
  + The availability should be updated accordingly within the attributes of the borrowed book.
* There should be a method to return a book.
  + Feedback should be given to the user upon success.
  + If the book does not exist, the user should be notified.
  + The borrowers account should then be updated to reflect this action.
  + The availability should be updated accordingly within the attributes of the returned book.
* The library data cannot be stored within a database and any algorithms to implement a method of storage should only use native input and output functions to the language. Memory footprint and performance should be considered for this decision.
* Positive and negative feedback should be given when an action is carried out by a user, allowing them to clearly understand if there have been any errors and how they can correct this.
* Error checking should be implemented to prevent users from carrying out invalid actions, such as removing a book that does not exist.
* Accessibility to users should be implemented where applicable, with limitations from the applications console-based nature being considered.
* User experience should be considered, and any solutions should be optimised towards a user’s interaction to create an application that is simple and efficient. Problems that are identified within development should be carefully considered to ensure that the adoption of the library management solution does not create new difficulties for the end-user.

**Project Limitations and Minimum Viable Product**

This project has to be developed within 4 weeks, creating a strict deadline. The time-boxed nature of this application has led to my decision to have 2-day sprints that will run across weekends. This limits the development time further and delays may lead to incomplete tickets. These reasons, coupled alongside with the nature of a single-developer team with limited experience, may prevent all requirements from being met. With this knowledge and using my role as Product Owner, I considered the key features that should be prioritised within development. After deciding on these features, I created a goal for the minimum viable product, these key features must be implemented to ensure success of the project and the creation of a functional library management application.

As product owner, I decided that the core features of the library management system should be catered for the librarian, as the solution can aid the librarian in managing a library. I decided that the functionalities that are catered to through the borrower user can be supplemented through the presence of a librarian. Therefore, I made the decision that success of the project does not rely on this feature.

Within this solution, books will have a unique ID code, availability, title, and author. There will be a feature to view all books that will allow a user to view a list of any book stored within the system that is displayed as a table.

This basic solution will have functionalities to allow a librarian to add and remove books. The length of book titles and authors will be between 1 to 50 characters, and any values entered outside this range will result in an error message and will not be added. The title and author of books will be taken from user input. Availability and book ID will be assigned automatically by the application. The length of book ID will be considered, and a decision will be met that considers ease for the end-user and preventing duplicates of the same ID code.

A basic search system will be implemented, allowing users to search for a book within the system using the unique book ID. It will involve error handling to display an error message for users that enter an invalid ID.

A function to check books in to the library and return them will be implemented through the use of unique book ID, allowing a Librarian to see availability clearly.

The menu system will be simple and ensure that a user can easily navigate the program. Accessibility will be considered and implemented to ensure the program can be used by many. One feature to ensure accessibility will be the implementation of a 'help' menu that allows a user to learn how to use the application. Another key feature for accessibility is the implementation of clear positive and negative feedback. This will be given to a user, indicating clearly if an action was successful or not.

In conclusion, the minimum viable product to ensure the success of the project only implements the most vital features. This will ensure that a basic system is created that is catered to librarians and allows them to search and view books. They will be able to add and remove books to the system, as well as viewing a help menu. Librarians will be able to check out and return books. Feedback will be given after an action is carried out and accessibility will be considered.

**Importance of Accessibility**

Ensuring any application is accessible is critical for the creation of a successful application. Accessibility ensures that an application is inclusive, allowing a wider user-base to reap the benefits of software[[10]](#footnote-10). For applications that require users to pay for licensing, this will expand customer base, improve engagement, and increase profit. Accessibility is required for legal compliance ensuring that any user regardless of sensory, cognitive, or physical impairments are able to benefit from software[[11]](#footnote-11). Improved accessibility benefits all users. Accessibility can look like many things within an application. For example, the use of legible fonts[[12]](#footnote-12) and clear and simple language[[13]](#footnote-13) ensures that all users have a better experience.

In consideration of the above points, I began to consider how I could create an accessible application. There are many accessibility features that are unable to be implemented due to the console-based nature of the program.

**Consideration of Accessibility**

To begin my research, I made use of ChatGPT to give examples of how I can create an accessible application. I entered a prompt and received a list of ideas to consider.

Prompt: I am developing a console-based application. I want to ensure my application is accessible. What are methods to ensure accessibility within an application that is limited to the console.

A screenshot of a computer

Description automatically generatedA screenshot of a black and white page

Description automatically generated

Figure 1 and 2: Generated response from ChatGPT.

This response enabled me to research further and add additional requirements for the application.

Through my role as Product Owner, I added additional requirements through my role to ensure accessibility. One of these new requirements were to implement a 'help' function, to allow the end-user to understand how to navigate the application. Another requirement was to implement consistent positive and negative feedback to the user, to clearly indicate the outcome of actions.

**MVP Use Case Diagram**

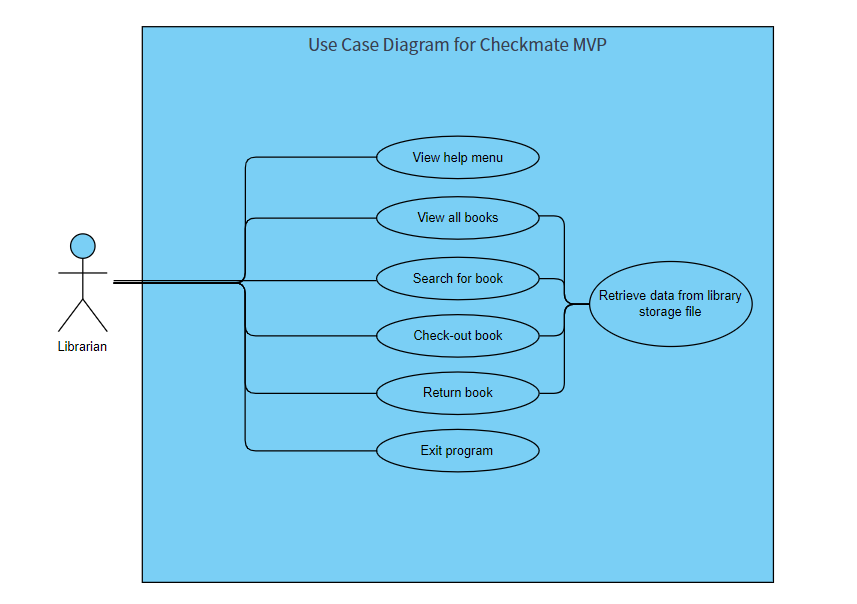


Figure3 : Use case diagram for the MVP

**Flowcharts**

The following flowcharts have all been created through Miro to consider the logic to implement each of these features.

A screenshot of a black screen

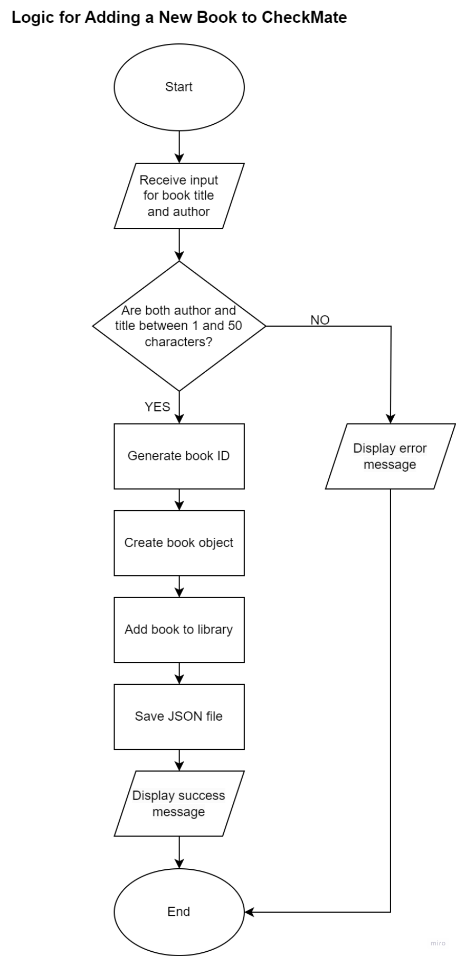
Description automatically generated

Figure 4: Logic for add\_book() function Figure 5: Logic for remove\_book() function

A black background with white squares

Description automatically generatedA screenshot of a computer screen

Description automatically generated

Figure 6: Logic for search\_by\_id() function Figure 7: Logic for check\_out\_book() function

A group of white squares with black text

Description automatically generated

Figure 8: Logic for return\_book () function

**Implementation**

**Sprint 1: Initial Implementation and Menu Navigation**

**Sprint Backlog and Goals**

The outcomes of the sprint planning meeting decided upon the goals for the sprint that are listed below.

**Basic Menu Structure:**

* **User Story:** As a user, I want a simple and functional menu in the CheckMate application for easy navigation.
* **Subtasks:**
  + **Implement a basic menu layout:**
    - **Acceptance Criteria:**
      * The menu should display options clearly.
      * Options should be numbered and easily distinguishable.
    - **Details:**
      * Create a function in the Menu class to print the basic menu structure.
      * Utilize the rich and pyfiglet libraries for enhanced text formatting.
  + **Experiment with colour schemes and layout for an accessible design:**
    - **Acceptance Criteria:**
      * Colours should be chosen for readability and accessibility.
      * Layout should be visually appealing and not cluttered.
    - **Details:**
      * Research colour combinations that enhance readability.
      * Experiment with different layouts to find an optimal design.

**Menu Navigation:**

* **User Story:** As a user, I want to navigate through the CheckMate menu to access different features.
* **Subtasks:**
  + **Implement menu navigation options:**
    - **Acceptance Criteria:**
* Users should be able to navigate to different menu options using numerical input.
  + - **Details:**
* Enhance the main\_menu function to handle user input and navigate accordingly.
  + **Handle user input for navigation:**
    - **Acceptance Criteria:**
      * Invalid input should be handled, providing feedback to the user.
    - **Details:**
      * Implement input validation to ensure users enter only valid menu options.
      * Display appropriate error messages for invalid inputs.
  + **Ensure the application responds appropriately to valid and invalid inputs:**
    - **Acceptance Criteria:**
      * Users should see the expected outcome based on their input (e.g., navigating to a help page, displaying books, or exiting).
    - **Details:**
      * Implement conditional statements based on user input to trigger the corresponding actions.

**Help Page:**

* **User Story:** As a user, I want a help page to guide me through the application and troubleshoot issues.
* **Subtasks:**
  + **Create a help page with an overview of CheckMate's features:**
    - **Acceptance Criteria:**
      * The help page should provide a clear overview of the application's functionalities.
  + **Details:**
    - Develop a new function in the Menu class to display the help page.
    - Include concise explanations of each menu option.
  + **Include examples and tips for navigation and troubleshooting:**
    - **Acceptance Criteria:**
      * Users should find examples and troubleshooting tips within the help page.
    - **Details:**
      * Incorporate sample usage scenarios for each menu option.
      * Include troubleshooting tips for common user errors.
  + **Ensure the help page is easily accessible from the main menu:**
    - **Acceptance Criteria:**
      * Users should be able to navigate to the help page from the main menu.
    - **Details:**
      * Add an option in the main menu to direct users to the help page.
      * Ensure the navigation is intuitive and straightforward.

**Research Storage options:**

* **User Story:** As a developer, I want to understand my options for file storage so I can begin implementing it in the next sprint.
* **Subtasks:**
  + **Research different storage options:**
    - **Acceptance Criteria:**
      * Make an informed decision on the most suitable storage solution for CheckMate:
    - **Details:**
      * A clear decision on the selected storage option is made, supported by the evaluation.

**Daily Scrum 1**

What was accomplished yesterday?

* Not applicable.

What will be done today?

* My goals are to work on creating the basic menu structure and then working on the navigation system.

What are my obstacles?

* Lack of knowledge around coding and best practice, this will be overcome through reading PEP 8 standards and learning through the implementation of these features.

**Daily Scrum 2**

What was accomplished yesterday?

* Completion of a basic menu structure and navigation system.

What will be done today?

* My goals are to work on writing a help page to add into the help menu function. Research will then be done to decide on the file structure for storage.

What are my obstacles?

* Lack of knowledge and experience with coding.

**Sprint Review**

Were the goals met?

* All tickets were completed throughout the sprint and the acceptance criteria was met for all.

What is next?

* The goals for the next sprint are to implement a method to store books within a json file and implement the basic librarian functions to meet the MVP.

**Sprint Retrospective**

What went well?

* All goals were met.
* Gained experience with coding.
* Made use of AI tools for research and productivity in researching possible file storage, which led to a decision.

What could be better?

* Lack of knowledge of file storage systems leads me to be concerned that I have not made correct decisions that may mean that the
* requirements are met. Example: Memory footprint is not minimized. and error checking is not correctly implemented.
* Concerns that testing is not thorough and complete.
* Commenting needs to be improved.

Action Items?

* Continue to comment more thoroughly.
* Practice coding skills through the second sprint.

**Sprint 2: Library Functionality Implementation**

**Sprint Backlog and Goals**

The outcomes of the sprint planning meeting decided upon the goals for the sprint that are listed below.

**Implement Add Book Function:**

* **User Story:** As a librarian, I want to add new books to the library easily.
* **Subtasks:**
  + **Create a method to add books:**
    - **Acceptance Criteria:**
      * Users should be able to add a new book by entering the title and author.
    - **Details:**
      * Implement the add\_book method in the LibraryFunctions class.
      * Ensure the method validates input and updates the library accordingly.

**Implement Remove Book Function:**

* **User Story:** As a librarian, I want to remove books from the library when necessary.
* **Subtasks:**
  + **Create a method to remove books:**
  + **Acceptance Criteria:**
    - Users should be able to remove a book by entering its ID.
  + **Details:**
    - Implement the remove\_book method in the LibraryFunctions class.
    - Ensure the method handles both successful removal and cases where the book is not found.

**Generate Book ID:**

* **User Story:** As a developer, I need a reliable method to generate unique book IDs.
* **Subtasks:**
  + **Develop a function to generate book IDs:**
    - **Acceptance Criteria:**
      * The generated IDs should be unique and consistent.
    - **Details:**
    - book\_id function in the LibraryFunctions class.
    - Use a reliable hashing algorithm to ensure uniqueness.

**Search Books by ID:**

* **User Story:** As a user, I want to find a specific book in the library by entering its ID.
* **Subtasks:**
  + **Implement a search function by book ID:**
    - **Acceptance Criteria:**
      * Users should receive information about the book with the entered ID.
    - **Details:**
      * Create the search\_by\_id method in the LibraryFunctions class.
      * Ensure the method returns relevant information or an appropriate message if the book is not found.
      * Make search\_by\_id accessible through shortening the id to 6 characters

**View All Books:**

* **User Story:** As a user, I want to see a list of all available books in the library.
* **Subtasks:**
  + **Enhance the 'View All Books' functionality:**
    - **Acceptance Criteria:**
      * The displayed list should include relevant information about each book.
    - **Details:**
      * Modify the display\_books method in the LibraryFunctions class to include additional details.

**Implement Book Class:**

* **User Story:** As a developer, I need a structured representation of books in the system.
* **Subtasks:**
  + **Create the Book class:**
    - **Acceptance Criteria:**
      * The Book class should have attributes for title, author, book ID, and availability.
    - **Details:**
      * Develop the Book class with necessary attributes and a constructor.
      * Ensure the class is structured to represent books in the library.

**Daily Scrum 1**

What was accomplished yesterday?

* Not applicable.

What will be done today?

* Start working on creating a book and library class. Work on creating the view all books, add books and book ID functions.

What are my obstacles?

* Lack of experience and the implementation of more complex features.

**Daily Scrum 2**

What was accomplished yesterday?

* Implemented the view all books, add books and book ID functions.

What will be done today?

* Start working on book removal and search feature.

What are my obstacles?

* Ensuring proper handling when removing books.
* Ensuring search feature is functional.

**Sprint Review**

Were the goals met?

* All tickets were completed throughout the sprint and the acceptance criteria was met for all.

What is next?

* The goals for the next sprint are to implement a log-in system, a check-out and return function.
* Ensure commenting is more thorough and complete the application.

**Sprint Retrospective**

What went well?

* All goals were met.
* Gained further experience with coding.

What could be better?

* Thorough testing.
* Comment code on each function for clarity.
* Some lines are greater than 79 characters, this must be fixed.

Action Items?

* Develop and execute a comprehensive testing plan in the next sprint.
* Comment on each function.
* Ensure lines are not over 79 characters.

**Sprint 3: Code Documentation, PEP8 Compliance, and User Authentication**

**Sprint Backlog and Goals**

The outcomes of the sprint planning meeting decided upon the goals for the sprint that are listed below.

**Code Documentation:**

* **User Story:** As a developer, I want comprehensive comments on the code to enhance readability and maintainability.
* **Subtasks:**
  + **Add comments to all functions and methods:**
    - **Acceptance Criteria:**
      * Every function and method should have descriptive comments explaining its purpose and usage.
    - **Details:**
      * Go through each function and method in the codebase and add comments for clarity.

**PEP 8 Compliance:**

* **User Story:** As a developer, I want code produced to be compliant with PEP 8 standards.
* **Subtasks:**
  + **Review the entire codebase for PEP 8 compliance:**
    - **Acceptance Criteria:**
      * Code adheres to PEP 8 standards without major violations.
    - **Details:**
      * Utilize linting tools and manual inspection to ensure compliance.
  + **Address and fix any PEP 8 violations:**
    - **Acceptance Criteria:**
      * All identified PEP 8 violations are resolved.
    - **Details:**
      * Make necessary adjustments to the code to align with PEP 8 standards.

**User Authentication System:**

* **User Story:** As a librarian, I want a login system to access higher privileges, while normal users should only be able to search and view books.
* **Subtasks:**
  + **Implement a login system:**
    - **Acceptance Criteria:**
      * Librarians can log in with a username and password.
    - **Details:**
      * Develop a login system using secure authentication methods.
  + **Grant higher privileges to librarians:**
    - **Acceptance Criteria:**
      * Librarians should have access to additional features, such as checking in and checking out books.
    - **Details:**
      * Define and implement higher-level privileges for librarian accounts through adjusting the implementation of current functions.
  + **Restrict borrower users to search and view functionality:**
    - **Acceptance Criteria:**
      * Borrower users should only be able to search for and view books.
    - **Details:**
      * Implement restrictions on functionality for non-librarian accounts.

**Return Book Feature:**

* **User Story:** As a librarian, I want the ability to check in books that have been returned.
* **Subtasks:**
  + **Implement a check-in feature:**
    - **Acceptance Criteria:**
      * Librarians can mark books as returned in the system.
    - **Details:**
      * Develop functionality to update the status of books when they are returned.

**Check-Out Feature:**

* **User Story:** As a librarian, I want the ability to check out books.
* **Subtasks:**
  + **Implement a check-out feature:**
    - **Acceptance Criteria:**
      * Librarians can mark books as checked out.
    - **Details:**
      * Develop functionality to update the status of books when they are checked out.

**Code Testing:**

* **User Story:** As a developer, I want to ensure the code is thoroughly tested.
* **Subtasks:**
  + **Review and update test plans:**
    - **Acceptance Criteria:**
      * Test plans cover all new features and functionalities.
    - **Details:**
      * Collaborate with the testing team to update test plans based on new code changes.
  + **Execute comprehensive testing:**
    - **Acceptance Criteria:**
      * All functionalities, including login, check-in, and check-out, should be tested.
    - **Details:**
      * Perform testing to identify and address any potential issues.

**Daily Scrum 1**

What was accomplished yesterday?

* Not applicable.

What will be done today?

* Begin implementing check-out and return functionality, comment on code as it is written.

What are my obstacles?

* Limited time to complete many tickets.

**Daily Scrum 2**

What was accomplished yesterday?

* Completed implementing check-out and return functionality, commented on code as it was written.

What will be done today?

* Continue commenting on functions, implement log-in system and make code PEP 8 compliant. Begin testing.

What are my obstacles?

* Last day of development.

**Sprint Review**

Were the goals met?

* Code documentation was improved, PEP8 compliance has somewhat been implemented. Testing is done.
* The log-in system was not implemented or functional, so it has been removed from the final code.

What is next?

* Evaluation and review of application.

**Sprint Retrospective**

What went well?

* Code documentation improved
* Testing complete.

What could be better?

* Too many items on the backlog due to incorrect weighting.
* Weighting incorrectly due to lack of experience.
* The code is not completely compliant with PEP 8 standards, 4 spaces have been used with sparing use of comments. However, there are several lines with too many characters.

Action Items?

* Improve time management for future sprints.

**Testing**

To test the final application, I have created a series of test tables that show the tests ran for each part of the application, as well as the results.

**Test Table 1: Menu Navigation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | 1 | Display the help message. | [Help message displayed] | Pass |
| 2 | 2 | Display a list of all available books. | [List of all books displayed] | Pass |
| 3 | 3 | Enter a valid book ID and display the book details. | [Book details displayed] | Pass |
| 4 | 4 | Enter valid title and author to add a new book. | "Book added successfully. ID: [unique ID]" | Pass |
| 5 | 5 | Enter an existing book ID to remove the book. | "Book with ID [ID] removed successfully." | Pass |
| 6 | 6 | Enter an available book ID to check out the book. | "Book with ID [ID] checked out successfully." | Pass |
| 7 | 7 | Enter a checked-out book ID to return the book. | "Book with ID [ID] returned successfully." | Pass |
| 8 | 8 | Exit the application. | [Application exits] | Fail |
| 9 | X | Display an error message. | "[red]Please enter a number from 1 to 8.[/red]" | Pass |

**Test Table 2: View Help Menu**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | N/A | Display help menu. | [Help message displayed] | Pass |

**Test Table 3: View all Books**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | N/A | Display all books in the library. | [List of all books] | Pass |

**Test Table 4: Search Books**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | Existing book ID | Display book details for the specified ID. | [Book details] | Pass |
| 2 | Non-Existent book ID | Display an error message about book not found. | "Error: Book not found." | Pass |

**Test Table 5: Add a New Book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | Valid title and author | New book is added to the library with a unique ID. | "Book added successfully. ID: [unique ID]" | Pass |
| 2 | Invalid title | Error message about invalid title/author. Book not added | "Invalid title or author length..." | Pass |
| 3 | Invalid author | Error message about invalid author/title. Book not added | "Invalid title or author length..." | Pass |
| 4 | No input | Error message about invalid author/title. Book not added | "Invalid title or author length..." | Pass |

**Test Table 6: Remove a Book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | Valid book ID | Confirmation of action success. Book is removed from the library. | "Book with ID [ID] removed successfully.". Removed from json file. | Pass |
| 2 | Invalid book ID | Error message about book not found. | "Error: Book with ID [ID] not found." | Pass |

**Test Table 7: Check Out a Book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | Available book ID | Confirmation of action success. Book status is updated to unavailable. | "Book with ID [ID] checked out successfully.". Book is available in file. | Pass |
| 2 | Valid but unavailable book ID | Error message about book already checked out. | "Error: Book with ID [ID] is already checked out." | Pass |
| 3 | Invalid book ID | Error message about book not found. | "Error: Book with ID [ID] not found." | Pass |

**Test Table 8: Return a Book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Result** | **Pass/Fail** |
| 1 | Unavailable book ID | Confirmation of action success. Book status is updated to available. | "Book with ID [ID] returned successfully.". Book is available in file. | Pass |
| 2 | Valid book ID but available book ID | Error message about book already checked out. | "Error: Book with ID [ID] is already available." | Pass |
| 3 | Invalid book ID | Error message about book not found. | "Error: Book with ID [ID] not found." | Pass |

**Evaluation**

**Mapping to Requirements**

Many requirements were not implemented into the final application. This was expected due to limited experience and the fixed deadline.

Books have a title, author, and unique ID, as well as an availability attribute. The title and author attributes must be between 1-50 characters. The implemented ID system makes use of hash values that are truncated, to prevent librarians from recalling long strings. This system means that book ID values can possibly be duplicated, however it is more accessible to users through this. In the future this can be built upon to create a truly unique ID system.

Books can be viewed, added, and removed. With error handling involved to prevent returned books from being returned again. The search system only works with ID values, this can be built upon to allow users to search for books more effectively, without requiring them to view all books first.

User interaction has been implemented through the console. There have been no formal tests to check saving, loading, and searching. The implementation of two user types has not been completed. However, colour is used to indicate positive and negative feedback.

Accessibility has been considered through consistent formatting of the Menu system and user actions. There is clear feedback of user actions and a clear help menu that describes in-depth how to navigate the software. Error checking has been implemented throughout. Native input and output functions are used retrieve data.

Therefore, the majority of the requirements have been met, along with meeting the MVP.

**Discussion of the Produced Software**

**Functionality**

The core functionalities that were outlined in the MVP have all been successfully implemented. Therefore, ensuring that the application development was a success. All features are functional, with the exception of the exit function. All have features have been fully tested.

**Usability**

The user interface and overall user experience are satisfactory. There are methods that could be implemented to improve the functionality. For example, if a user enters an incorrect book ID, they are taken back to the main menu. A user may want to attempt to enter the ID again and have the choice to exit to the main menu. The implementation of user testing would ensure that the usability is greater.

**Future Improvements**

While the project can be considered a success due to the achievement of the basic features outlined in the MVP, there is still more that can be done to improve the application. For example, the implementation of improved search functions to allow author and title to be entered to find books would create a more user-friendly experience. The implementation of an account system would ensure the application is available to a wider user-base. Due to the time constraints and limited experience, there are many additional requirements that are yet to be implemented.

In the future, this application could be developed further through continuing the sprints beyond the 4-week timeframe. Allowing more features to slowly be implemented through future iterations.

Transitioning away from a console-based interface would be a significant improvement. This could allow greater accessibility and visual appeal, enhancing user experience. Console-based interfaces are not accessible to all, they can seem overwhelming to those that are unfamiliar, therefore implementing a graphical user interface could create a more user-friendly experience. Potentially allowing the application to become marketable as a rival library management solution.

**Next Iteration**

The next iteration of CheckMate would focus on the implementation of the log-in system for users and librarians as well as focusing on improving the search functionality.

**Conclusions**

In conclusion, the CheckMate application can be considered a success, as the minimum viable product requirements have been met, therefore satisfying the product owner. That being said, there are many features within the requirements that were unable to be implemented. Along with the incomplete tickets from Sprint 3. The development process has served as a valuable learning experience. Through it, I have gained a greater understanding of coding conventions and the development process. As well as this, I have simulated different scrum roles, allowing me to understand the methodology in greater detail. The next steps for this project are to continue with development by working through the backlog to create a better solution for library management.

**Sources**

1. Atlassian (2023) *Waterfall Methodology: A Comprehensive Guide*

Available from: <https://www.atlassian.com/agile/project-management/waterfall-methodology> [Accessed 28th October 2023]

1. ProjectManager (2023) *What Is the Waterfall Methodology in Project Management?*

Available from: <https://www.projectmanager.com/guides/waterfall-methodology>

[Accessed 29th October 2023]

1. Lucidchart (2023) *The pros and cons of Waterfall methodology*

Available from: <https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology> [Accessed 30th October 2023]

1. Martins, J. (2022) *What is Kanban? Here’s what your Agile team needs to know*

Available from: <https://asana.com/resources/what-is-kanban>

[Accessed 31st October 2023]

1. Adobe (2022) *Kanban Methodology*

Available from: <https://business.adobe.com/blog/basics/kanban>

[Accessed 31st October 2023]

1. Drumond, C. (2023) *What is scrum?*

Available from: <https://www.atlassian.com/agile/scrum>

[Accessed 1st November 2023]

1. Scrum.org (2023) *What is a Product Owner?*

Available from: <https://www.scrum.org/resources/what-is-a-product-owner#:~:text=As%20described%20in%20the%20Scrum%20Guide%2C%20a%20Product,the%20team%20about%20a%20product%E2%80%99s%20vision%20and%20goal>

[Accessed 1st November 2023]

1. Rehkopf, M. (2023) *What is a scrum master & their responsibilities?*

Available from: <https://www.atlassian.com/agile/scrum/scrum-master>

[Accessed 1st November 2023]

1. Rossum, G (2001) *PEP 8 – Style Guide for Python Code*

Available from: <https://peps.python.org/pep-0008>

[Accessed 28th October 2023]

1. Apriorit (2023) *The Importance of Accessibility and Inclusiveness in UI/UX Design*

Available from: <https://www.apriorit.com/dev-blog/design-accessibility-in-ui-ux>

[Accessed 14th November 2023]

1. Mishra, A. (2023) *How to Incorporate Accessibility in Software Development*

Available from: <https://adevait.com/software/accessibility-in-software-development>

[Accessed 14th November 2023]

1. Williams, G. F. (2020) *A Guide to Understanding What Makes a Typeface Accessible*

Available from: <https://medium.com/the-readability-group/a-guide-to-understanding-what-makes-a-typeface-accessible-and-how-to-make-informed-decisions-9e5c0b9040a0>

[Accessed 14th November 2023]

1. Gov.uk *Accessible communication formats* Available from: <https://www.gov.uk/government/publications/inclusive-communication/accessible-communication-formats> [Accessed 14th November 2023]

1. Atlassian (2023) *Waterfall Methodology: A Comprehensive Guide* Available from: <https://www.atlassian.com/agile/project-management/waterfall-methodology> [Accessed 28th October 2023] [↑](#footnote-ref-1)
2. ProjectManager (2023) *What Is the Waterfall Methodology in Project Management?* Available from: <https://www.projectmanager.com/guides/waterfall-methodology> [Accessed 29th October 2023] [↑](#footnote-ref-2)
3. Lucidchart (2023) *The pros and cons of Waterfall methodology* Available from: <https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology> [Accessed 30th October 2023] [↑](#footnote-ref-3)
4. Martins, J. (2022) *What is Kanban? Here’s what your Agile team needs to know* Available from: <https://asana.com/resources/what-is-kanban> [Accessed 31st October 2023] [↑](#footnote-ref-4)
5. Adobe (2022) *Kanban Methodology* Available from: <https://business.adobe.com/blog/basics/kanban> [Accessed 31st October 2023] [↑](#footnote-ref-5)
6. Drumond, C. (2023) *What is scrum?* Available from: <https://www.atlassian.com/agile/scrum> [Accessed 1st November 2023] [↑](#footnote-ref-6)
7. Scrum.org (2023) *What is a Product Owner?* Available from: <https://www.scrum.org/resources/what-is-a-product-owner#:~:text=As%20described%20in%20the%20Scrum%20Guide%2C%20a%20Product,the%20team%20about%20a%20product%E2%80%99s%20vision%20and%20goal> [Accessed 1st November 2023] [↑](#footnote-ref-7)
8. Rehkopf, M. (2023) *What is a scrum master & their responsibilities?* Available from: <https://www.atlassian.com/agile/scrum/scrum-master> [Accessed 1st November 2023] [↑](#footnote-ref-8)
9. Rossum, G (2001*) PEP 8 – Style Guide for Python Code* Available from: <https://peps.python.org/pep-0008> [Accessed 28th October 2023] [↑](#footnote-ref-9)
10. Apriorit (2023) *The Importance of Accessibility and Inclusiveness in UI/UX Design* Available from: <https://www.apriorit.com/dev-blog/design-accessibility-in-ui-ux> [Accessed 14th November 2023] [↑](#footnote-ref-10)
11. Mishra, A. (2023) *How to Incorporate Accessibility in Software Development* Available from: <https://adevait.com/software/accessibility-in-software-development> [Accessed 14th November] [↑](#footnote-ref-11)
12. Williams, G. F. (2020) *A Guide to Understanding What Makes a Typeface Accessible* Available from:

    <https://medium.com/the-readability-group/a-guide-to-understanding-what-makes-a-typeface-accessible-and-how-to-make-informed-decisions-9e5c0b9040a0> [Accessed 14th November 2023] [↑](#footnote-ref-12)
13. Gov.uk *Accessible communication formats* Available from: <https://www.gov.uk/government/publications/inclusive-communication/accessible-communication-formats> [Accessed 14th November 2023] [↑](#footnote-ref-13)