Project Plan

Version 2.1

for

SHELVD

Prepared by

Arun Ezekiel S/O Richard
Cheong Yu Qing, Crystal
Kenneth Goh Jing Wei
Muhammad Sufyan Bin Mohd Jais
Nyan Maw Htun
Ryu Hyunsun
Zhu Yuhao

Mawsters , Nanyang Technological University

2024-03-27

Revision History

Revision Number	Date	Primary Author(s)	Comments
1.0	Mar 10th, 2024	Arun Ezekiel S/O Richard, Kenneth Goh Jing Wei, Muhammad Sufyan Bin Mohd Jais, Nyan Maw Htun	First version
1.1	Mar 11th, 2024	Muhammad Sufyan Bin Mohd Jais	Review Sections 1-4
1.2	Mar 11th, 2024	Kenneth Goh Jing Wei	Review Sections 8 -10
1.2	Mar 12th, 2024	Nyan Maw Htun	Review Section 5
2.0	Mar 13th, 2024	Mawsters Team	Final Version
2.1	Mar 27th, 2024	Mawsters Team	Confirmation of Project Name change from BookCatalogue to shelvd

Table of Contents

Revision History 1

Table of Contents 2

1 Introduction 3

- 1.1 Project Overview 3
- 1.2 Project Description and Scope 3

2 Project Organization 4

- 2.1 Team Structure 4
- 2.2 Roles and Responsibilities 4
- 2.3 Team Communication 5

3 Process Definition 6

- 3.1 Lifecycle Model 6

4 Schedule 7

- 4.1 Activity Dependencies and Schedule 7
- 4.2 Work Breakdown Structure 8
- 4.3 Work Packages 9
- 4.4 Activity Dependencies 9
- 4.5 Work Package Details 10

5 Project Estimates 14

- 5.1 Code Size Estimation using Function Points 14
  - 5.1.1 Unadjusted Function Points 14
  - 5.1.2 Adjusted Function Points 19
  - 5.1.3 Lines of Code 20
- 5.2 Efforts, Duration and Team Size Estimation 21
  - 5.2.1 Distribution of Effort 21
- 5.3 Cost Estimates 22

6 Product Checklist 23

7 Best Practice Checklist 24

8 Risk Management 25

9 Quality Assurance 26

10 Monitoring & Control 27

1 Introduction

1.1 Project Overview

Shelvd is a web application designed to facilitate users' exploration and engagement with books. Serving as a platform for accessing a diverse range of books, shelvd offers users the ability to conduct searches, view detailed book information, access best-selling lists, and receive personalised recommendations. Additionally, users can leverage the application's functionality to create, edit, and manage book collections according to their preferences and interests. By integrating recent cache data, shelvd ensures seamless access to previously viewed content.

1.2 Project Description and Scope

The shelvd application represents a school project undertaken by our team. Our objective is to develop a comprehensive book catalog functionality within the application framework, enabling users to explore and engage with a vast array of literary works. The primary focus of the shelvd system is to provide users with an intelligent search engine capable of accommodating various preferences and constraints.

Key features of the shelvd application include:

- A user-friendly interface facilitating seamless navigation and intuitive search functionality.
- Comprehensive search capabilities allowing users to filter books based on genres, authors, publication dates, formats, and other criteria.
- Integration with free external APIs to access up-to-date book data, including titles, authors, summaries, and reviews.
- Personalisation features enabling users to receive recommendations based on their reading history, preferences, and browsing patterns.
- Support for the creation, editing, and management of custom book collections, allowing users to organise their reading lists according to their interests.
- Seamless offline access utilising recent cache data to ensure uninterrupted browsing and exploration, even in low-connectivity environments.

The shelvd application is designed to provide a robust platform for users to discover, select, and engage with books tailored to their individual preferences and interests. Our team is committed to delivering a user-centric experience that enhances the enjoyment and accessibility of literature for all users.

2 Project Organization

2.1 Team Structure

The following is the list of executive roles, as required by CMM level 3.

- Senior Management: Maw Htun
- Software Configuration Manager: Crystal
- Software Engineering Project Group: Sufyan, Yuhao, Hyunsun, Maw Htun, Arun
- Software Quality Assurance Engineer: Kenneth
- Representative of Customer: Crystal

2.2 Roles and Responsibilities

Project Manager: Maw Htun

- Oversees project progress
- Approves and executes project plan
- Assigns tasks and reports status of project to team members
- Manages and motivates team members
- Represents the team to the outside world

Front End Developer: Sufyan, Yuhao, Arun

- Implements the user interface design
- Integrate front-end components with the back-end system
- Utilises front-end technologies to develop responsive and interactive web interfaces
- Conducts usability testing to identify and address any issues with the user interface design

Back End Developer: Hyunsun, Maw Htun

- Implements the back-end functionality of the system
- Develops server-side logic
- Integrates external APIs and databases to support system functionality and data retrieval
- Works closely with the front-end developer to ensure smooth communication and integration between front-end and back-end components

Release Manager: Crystal

- Coordinates the release process, ensuring that software updates and new features are deployed smoothly and efficiently
- Manages release schedules and coordinates with the development team to prioritise and schedule releases
- Conducts risk assessments and manages release dependencies

Quality Assurance Manager: Kenneth

- Ensures acceptable software quality
- Designs testing strategies
- Creates and manages test plan
- Verify software requirements
- Executes test procedures

2.3 Team Communication

Our communication channels include the following:

- Weekly meetings are held on Thursdays.
- Group announcements and updates are sent through our Telegram group.
- Zoom discussions are held as necessary.
- Split up into subgroups as necessary, in order to work more co-operatively on specific problems.

3 Process Definition

3.1 Lifecycle Model

We intend to use the Incremental Development Model throughout the shelvd project. This methodology is more flexible than the traditional Waterfall SDLC due to repeated iterations involving design, coding, unit testing, integration, and quality assurance. The Waterfall SDLC is not a viable choice due to the short timeline available for the shelvd app to reach delivery quality.

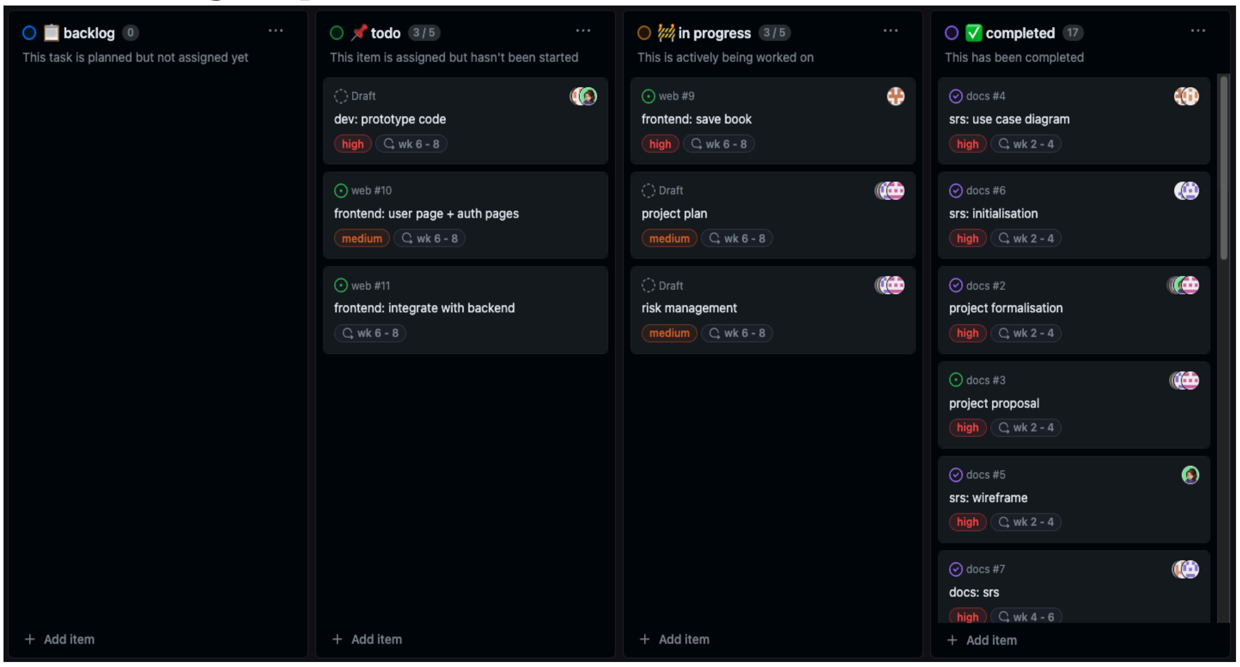
We have chosen to avoid such methodologies as Spiral because of concerns over the short timeline. Should design procedures, for example, need to be revisited within the first release date, it is likely that the project will overshoot its critical schedule.

We intend to deliver the first iteration of functionality on the System Delivery date indicated in the Estimations section of this document. After further client interaction, further iterations should occur as necessary.

Due to the intelligent nature of this agent, further iterations interleaved with client interaction and live testing should hone the algorithms used such that they behave in an accurate and logical manner, providing a more effective search experience for users.

4 Schedule

4.1 Activity Dependencies and Schedule



📋 backlog 0
This task is planned but not assigned yet

+ Add item

📌 todo 3 / 5
This item is assigned but hasn't been started

- Draft
  dev: prototype code
  high · wk 6 - 8
- web #10
  frontend: user page + auth pages
  medium · wk 6 - 8
- web #11
  frontend: integrate with backend
  wk 6 - 8

+ Add item

🚧 in progress 3 / 5
This is actively being worked on

- web #9
  frontend: save book
  high · wk 6 - 8
- Draft
  project plan
  medium · wk 6 - 8
- Draft
  risk management
  medium · wk 6 - 8

+ Add item

✅ completed 17
This has been completed

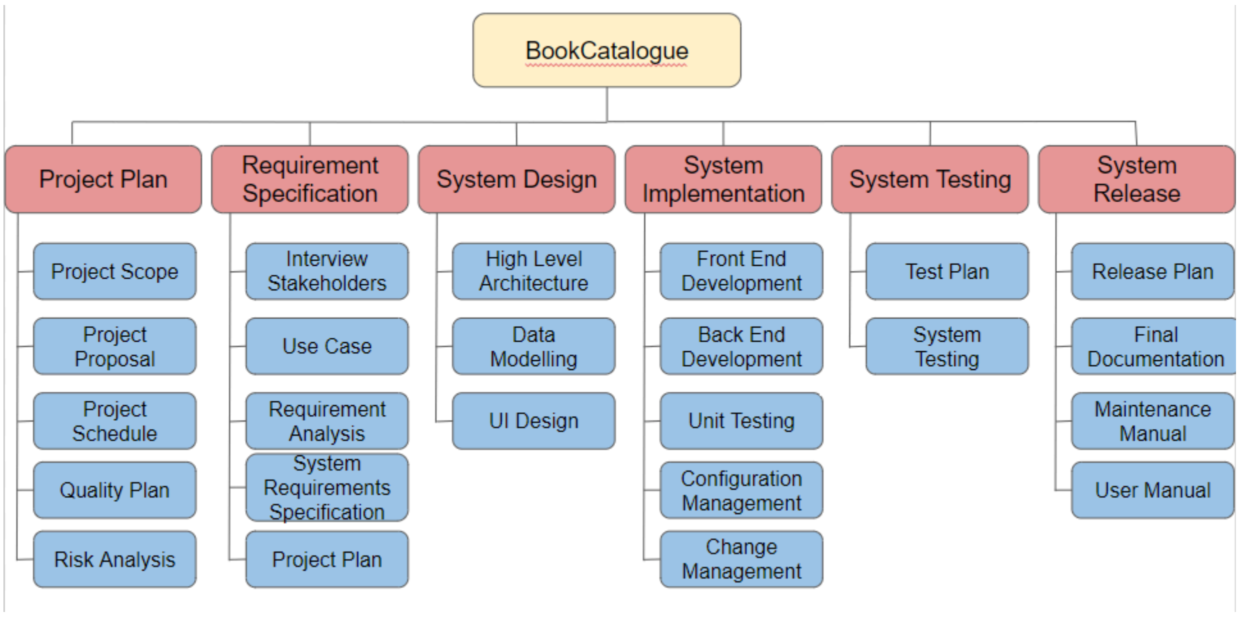
- docs #4
  srs: use case diagram
  high · wk 2 - 4
- docs #6
  srs: initialisation
  high · wk 2 - 4
- docs #2
  project formalisation
  high · wk 2 - 4
- docs #3
  project proposal
  high · wk 2 - 4
- docs #5
  srs: wireframe
  high · wk 2 - 4
- docs #7
  docs: srs
  high · wk 4 - 6

+ Add item

The detailed bar chart can be viewed on the project's Github site, accessible via:

- Backlog: https://github.com/orgs/mawsters/projects/3/views/1
- Roadmap: https://github.com/orgs/mawsters/projects/3/views/4

4.2 Work Breakdown Structure



4.3 Work Packages

The entire project work is broken down by the important phases of the software development life cycle. They include the following:

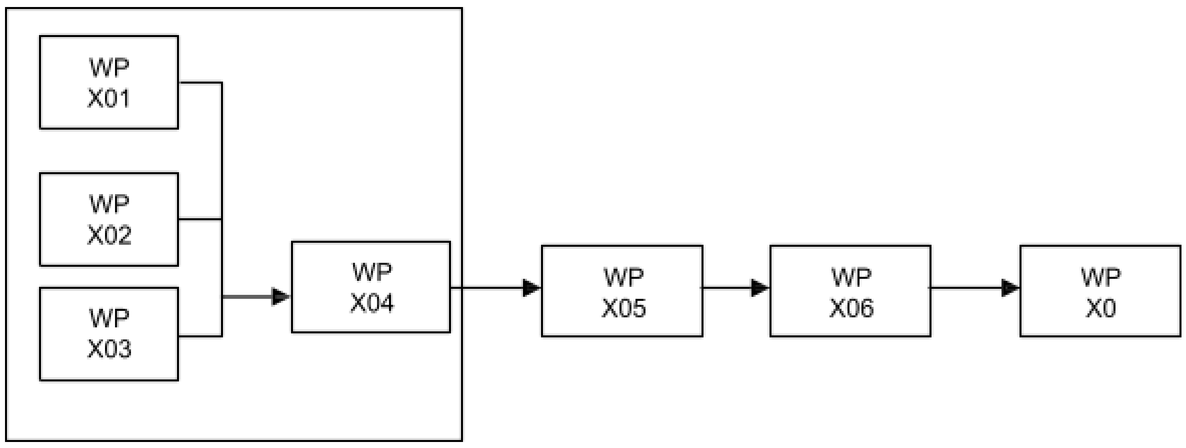
1. Project Plan
2. Requirement Specification
3. User Interface
4. Technical Architecture
5. Data Modeling
6. Coding & Unit Testing
7. Integration & Quality Assurance

4.4 Activity Dependencies

The following table describes the dependencies of the deliverable work packages:

Work Package #	Work Package Description	Duration	Dependencies
X01	Project Plan	7 days	--
X02	Requirement Specification	7 days	--
X03	User Interface	7 days	--
X04	Technical Architecture	12.1 days	X01,X02,X03
X05	Data Modeling	7 days	X04
X06	Coding & Unit Testing	16.2 days	X05
X07	Integration & System Testing	16.2 days	X06

The following Activity Network Diagram describes the above in more graphical detail:



Note that work package X05 is dependent on all work packages encapsulated by the larger boxes linked to its left. For instance, WP X05 may not start until WP X01- X04 has been finished.

4.5 Work Package Details

Work packages are listed below. A team member, indicated in bold, has been assigned as primarily responsible for each work package and will coordinate that package.

Project	shelvd Software System
Work Package	X01— Project Plan (1 of 7)
Assigned To	**Maw Htun,** Crystal, Sufyan, Yuhao, Hyunsun, Kenneth, Arun
Effort	7PD
Start Date	Thursday, 01/25/24
Purpose	To determine an introductory overview of the project, to be refined in later work packages.
Inputs	None
Activities	This work package includes providing a brief overview of the project, its objectives, and a set of proposed project deliverables throughout the development of the software cycle. The people responsible for this work package will also be transcribing ideas brought up in the group meeting discussion into a formal report.
Outputs	A written document of the Project Plan Introduction.

Project	shelvd Software System
Work Package	X02— Requirement Specification (2 of 7)
Assigned To	**Crystal,** Maw Htun, Sufyan, Yuhao, Hyunsun, Kenneth, Arun
Effort	7PD
Start Date	Thursday, 02/01/24
Purpose	To establish a common understanding between the customer and the software project team of the customers' requirements to be addressed by the project
Inputs	Customer's requirements
Activities	Identify "the customer", interview customer, write and inspect customer requirement and build requirements.
Outputs	A written document of the requirement specification.

Project	shelvd Software System
Work Package	X03— User Interface (3 of 7)
Assigned To	**Sufyan**, Crystal**,** Maw Htun, Yuhao, Hyunsun, Kenneth, Arun
Effort	7PD
Start Date	Friday, 02/01/24
Purpose	To build the user interface between the system and the customer, to make it easy use, and friendly to the customer
Inputs	User information
Activities	To get the user information, user request, display the dialog between system and user, display the result of request
Outputs	User Interface

Project	shelvd Software System
Work Package	X04— Technical Architecture (4 of 7)
Assigned To	**Yuhao**, Sufyan, Crystal**,** Maw Htun, Hyunsun, Kenneth, Arun
Effort	12.1PD
Start Date	Friday, 02/01/24
Purpose	To do the high level architecture design
Inputs	Project Plan Work Packages (X01 to X03 inclusive).
Activities	High level design entails defining the architecture of the software system and identifying the various components and how they are inter-related to and interactive with each other. Designers also need to decide on the software and hardware infrastructures, such as what operating system on which the software is built, the language used to implement the software, and so on. Design topics including maintainability, portability, and reusability will be addressed here as well.
Outputs	High Level Design and Architectural Specification.

Project	shelvd Software System
Work Package	X05— Data Modeling (5 of7)
Assigned To	**Hyunsun**, Yuhao, Sufyan, Crystal**,** Maw Htun, Kenneth, Arun
Effort	7PD
Start Date	Thursday, 01/15/24
Purpose	To build the project's database
Inputs	Project Plan Work Packages (X01 to X05 inclusive).
Activities	Analyze the data flow relationships, entity relationships
Outputs	A written document of the data modeling

Project	shelvd Software System
Work Package	X06— Coding & Unit testing (6 of 7)
Assigned To	**Kenneth**, Hyunsun, Yuhao, Sufyan, Crystal**,** Maw Htun, Arun
Effort	16.2PD
Start Date	Thursday, 02/28/24
Purpose	To implement the system as per the requirements specification and other associated documents. This work package includes such additional activities as preliminary unit testing.
Inputs	Project Plan Work Package X06.
Activities	Programmers will implement the modules according to the design specifications noted in the Specification document.
Outputs	Source code and header files

Project	shelvd Software System
Work Package	X07— Integration & System Testing (7 of 7)
Assigned To	**Arun**, Kenneth, Hyunsun, Yuhao, Sufyan, Crystal**,** Maw Htun
Effort	16.2PD
Start Date	Thursday 03/14/24
Purpose	To identify and fix logical and syntactical errors produced during the implementation of the System, and setting up drivers and stubs to see how the module responds to various inputs. Black box testing as well as white box testing might be conducted to check for logical errors. All the testing procedures will be documented in the Test Plan report. If problems are found, they will be noted and fixed at the earliest possible time.
Inputs	Project Plan Work Package X07.
Activities	The Integration testing team may try to simulate how a user might interact with the system. Similar to Unit Testing, Integration Testing may require the development of stubs and drivers as well, but here this is more geared towards the higher (overall system) level. Testers may also examine issues such as system performance and integrity. Heuristics assessment plays an important role in this work package, as intelligence components will define eventual system success.
Outputs	A test report.

5 Project Estimates

5.1 Code Size Estimation using Function Points

We calculated unadjusted function point based on the complexity of functions provided by this system. Code size is then estimated by adjusted function point.

5.1.1 Unadjusted Function Points

The shelvd system supports the following proposed functions, as detailed in the Software Requirements Specification (SRS):

Shelvd user:

- Book Search
- Login
- Register
- Logout
- Reset / Change Password
- Bookmarking (Save Books, Create Collection, Add to Collection)
- View Book Details
- Manage Book Collections

The assessment of unadjusted function points is determined through five core components of these functions: Inputs, Outputs, Inquiries, Logical Files, and Interfaces. The complexity of each element is classified into three categories: Low, Medium, and High. The complexity of each function is analysed based on the Function Point Counting Practices Manual Release 4.1.1 by The International Function Point Users Group. The complexity estimation uses a matrix based on the number of Files Type Referenced (FTRs)/Record Element Types (RETs) against the number of Data Elements (DEs).

Rating Inputs:

Matrix:

File Types Referenced (FTR)	**Data Elements**		
	1-4	5-15	Greater than 15
Less than 2	Low	Low	Average
2	Low	Average	High
Greater than 2	Average	High	High

Complexity Estimation:

Functionality	**FTR/DE Count**	**Description**	**Complexity**		
			L	**A**	**H**
Login (for Users)	FTR: 2	The system interacts with the User Database for verification and the ClerkJS API for login processes.	✔		
	DE: 2	The user inputs a username and password.			
Register (for Users)	FTR: 2	This interacts with the User Database to check the existence of the account and to store new account	✔		

		details, and with the ClerkJS API for email verification.			
	DE: 4	Inputs include email, password, confirm password, and potentially a verification code. The password has multiple criteria, but these are checked as one input.			
Reset Password	FTR: 2	Interactions occur with the User Database to verify the email and to update the password, as well as with the ClerkJS API for sending and verifying the reset code.	✔		
	DE: 3	The user provides an email, new password, and a password reset code.			
Logout	FTR: 2	This may involve interaction with a session management database to invalidate the session token and the ClerkJS API.	✔		
	DE: 1	If you include user action confirmations, then DE might be 1.			
Save Books	FTR: 1	This would involve at least one database where the collections are stored.	✔		
	DE: 2	User actions here include selecting a book to save and choosing a collection.			
Create Collection	FTR: 1	Involves the collections database.	✔		
	DE: 1	The user inputs the name of the new collection.			
View Books on Home Page	FTR: 2	Interacts with caches or APIs (New York Times API, Server for recommendations).			
	DE: 0	Typically no direct user input for viewing operations.			
Search Books	FTR: 2	Involves interaction with the search cache and Google Books API.	✔		
	DE: 4	The user enters search criteria such as title, genre, author, and ISBN.			
View Collections	FTR: 1	Interacts with the user's collection database to retrieve the list of collections.			
	DE: 0	Typically no user input for simply viewing collections.			
View Book Details	FTR: 1	May involve interaction with a book details cache or database.			
	DE: 0	If the user is simply viewing details, there is no DE.			
Delete a book from Collection	FTR: 1	This would involve interaction with the collections database to delete a book.	✔		
	DE: 1	The user action of selecting a book for deletion is 1 DE.			
Count			8	0	0

The assessment of unadjusted function points is determined through five core components of these functions: Inputs, Outputs, Inquiries, Logical Files, and Interfaces. The complexity of each element is classified into three categories: Low, Medium, and High. The complexity of each function is analysed based on the Function Point Counting Practices Manual Release 4.1.1 by The International Function Point Users Group. The complexity estimation uses a matrix based on the number of Files Type Referenced (FTRs)/Record Element Types (RETs) against the number of Data Elements (DEs).

Rating Outputs:
Matrix:

File Types Referenced (FTR)	**Data Elements**		
	1-5	6-19	Greater than 19
less than 2	Low	Low	Average
2 or 3	Low	Average	High
Greater than 3	Average	High	High

Complexity Estimation:

Functionality	**FTR/DE Count**	**Description**	**Complexity**		
			L	**A**	**H**
Login (for Users)	FTR: 1	User session information or authentication token.	✔		
	DE: 2	Typically includes messages for successful login or errors.			
Register (for Users)	FTR: 2-4	User table, email validation records, default book collections.	✔		
	DE: 3	Multiple outputs like account creation confirmation, errors, and email validation.			
Reset Password	FTR: 1	User table for resetting password.	✔		
	DE: 2	Includes success message and potential error messages.			
Logout	FTR: 1	Session data.	✔		
	DE: 1	Includes logout confirmation message.			
Save Books	FTR: 1	This would involve at least one database where the collections are stored.	✔		
	DE: 2	Notifications for success or failure when saving books.			
Create Collection	FTR: 1	Involves the collections database.	✔		
	DE: 2	Includes success message and error messages for collection creation.			
View Books on Home Page	FTR: 2	bestsellers cache, recommendations cache.	✔		
	DE: 4	Outputs could include lists of bestsellers and recommendations (4 estimated DE - 2 lists, each with a success or error state).			
Search Books	FTR: 2	Involves interaction with the search cache and Google Books API.	✔		
	DE: 3	Search results, possibly including pagination and error messages.			
View Collections	FTR: 1	Interacts with the user's collection database to retrieve the list of collections.	✔		
	DE: 2	List of collections, potentially an error message if no collections are found.			
View Book Details	FTR: 1	May involve interaction with a book details cache or database.	✔		
	DE: 3	Book details including description, metadata, and links.			
Delete a book	FTR: 1	This would involve interaction with the collections	✔		

from Collection		database to delete a book.			
	DE: 2	Success or error message after deletion attempt.			
Count			11	0	0

Rating Inquiries:
Matrix:

File Types Referenced (FTR)	**Data Elements**		
	1-5	6-19	Greater than 19
less than 2	Low	Low	Average
2 or 3	Low	Average	High
Greater than 3	Average	High	High

Complexity Estimation:

Functionality	**FTR/DE Count**	**Description**	**Complexity**		
			L	**A**	**H**
Login (for Users)	FTR: 1	User account validation.	✔		
	DE: 2	Username, password verification result.			
Register (for Users)	FTR: 2	User account creation, email verification.	✔		
	DE: 3	Email verification result, account creation status, error messages.			
Reset Password	FTR: 1	Password reset mechanism.	✔		
	DE: 2	Email verification, new password acceptance.			
Logout	FTR: 1	Session management.	✔		
	DE: 1	Session termination confirmation.			
Save Books	FTR: 2	Book details, collection management.	✔		
	DE: 3	Book ID, Collection ID, addition status.			
Create Collection	FTR: 1	Collection management.	✔		
	DE: 2	New collection name, creation status.			
View Books on Home Page	FTR: 2	Bestsellers cache, recommendations cache.	✔		
	DE: 4	List of bestsellers, list of recommendations, each with a status.			
Search Books	FTR: 2	Book cache or direct API response, search criteria management.		✔	
	DE: 4+	Search criteria, search results, pagination, error message. Complexity is likely Average, due to the dynamic nature of search criteria and results.			
View Collections	FTR: 1	Collections data.	✔		
	DE: 2	Collections list, retrieval status.			
View Book Details	FTR: 1	Book details.	✔		
	DE: 3	Book metadata, availability in collections, external links.			
Delete a book from Collection	FTR: 1	Collection management.	✔		
	DE: 3	Book ID, Collection ID, deletion status.			
Count			10	1	0

Rating Logical Files:
Matrix:

Record Element Types (RET)	Data Elements		
	1 to 19	20 - 50	51 or More
1 RET	Low	Low	Average
2 to 5 RET	Low	Average	High
6 or More RET	Average	High	High

Complexity Estimation:

Functionality	Complexity		
	L	A	H
User Account Management (including Login, Logout, Register, Reset Password)		✔	
Book Collection Management (including Save Books, Create Collection, Delete a Book from Collection)			✔
Search and Display Books (including View Books on Home Page, Search Books, View Book Details)			✔
Count	0	1	2

Rating Interfaces:
Matrix:

Record Element Types (RET)	Data Elements		
	1 to 19	20 - 50	51 or More
1 RET	Low	Low	Average
2 to 5 RET	Low	Average	High
6 or More RET	Average	High	High

Complexity Estimation:

Functionality	Complexity		
	L	A	H
ClerkJS API (for Login, Logout, Register, Reset Password)	✔		
Google Books API (for Search Books, View Book Details)		✔	
New York Times API (for View Books on Home Page)	✔		
Count	2	1	0

Calculation of Unadjusted Function Points:

Characteristic	Low		Medium		High	
Inputs	8	× 3	0	× 4	0	× 6
Outputs	11	× 4	0	× 5	0	× 7
Inquiries	10	× 3	1	× 4	0	× 6
Logical Files	1	× 7	0	× 10	2	× 15

Interfaces	2	× 5	1	× 7	0	× 10
Unadjusted FP	115		11		30	
Total=L+M+H	156					

5.1.2 Adjusted Function Points

Scoring (0 – 5)
0 = No influence
1 = Insignificant influence
2 = Moderate influence
3 = Average influence
4 = Significant influence
5 = Strong influence

Influence Factors	**Score**	**Description**
Data Communications	2	The application interacts with external APIs for authentication and data retrieval, indicating a moderate level of data communication.
Distributed Functions	1	The functionalities require some interaction with distributed services like APIs, but the bulk of the processing seems centralized.
Performance	3	Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.
Heavily used	2	Some security or timing considerations are included.
Transaction rate	2	Regular transactions like logins and book management will occur. Daily peak transaction period is anticipated.
On-line data entry	3	Multiple functionalities involve direct online data entry, such as registering, resetting passwords, and saving books.
End-user efficiency	3	There is emphasis on the user interface and experience.
On-line data update	2	Regular updates are likely due to online data entry and updates, like saving books or creating collections.
Complex processing	3	Interactions with external APIs and real-time data handling suggest a moderate level of complexity.
Reusability	1	No special considerations were stated by the user.
Installation Ease	0	The application is to be web-based, which typically means there is little to no need for installation.
Operational Ease	3	Operational ease is to be a priority given the focus on user efficiency and quick operations like login.
Multiple sites	1	As a web application, it's naturally accessible from various locations.
Facilitate change	3	The system is likely to facilitate change, considering the modern web-based environment and interactions with external APIs that suggest adaptability.

Total score	29
Influence Multiplier = Total score × 0.01 + 0.65 = 29 × 0.01 + 0.65 = 0.94	
Adjusted FP = Unadjusted FP × Influence Multiplier = 156 × 0.94 = 146.64	

5.1.3 Lines of Code

According to QSM Function Points Languages Table, each Function Point requires 47 lines of code if the application is implemented using JavaScript.
Therefore, we have: **Lines of Code** = 146.64 FP × 47 LOC/FP = **6893 LOC**

5.2 Efforts, Duration and Team Size Estimation

To estimate the effort and duration required for the project, we use function points as the basis to calculate Effort, Duration, Team size and finally the schedule. The estimates are expanded to account for project management and extra contingency time to obtain the total average effort estimates. From these averages, the duration of each work package in working days is estimated based on the following calculations.

- Working days include 5 days in a week.
- Effort = Size / Production Rate = (6893 LOC) / (62 LOC/PD)[1] = 111.18 PD
- Duration = 3 × (Effort) $^{1/3}$ = 3 × (111.18) $^{1/3}$ = 11.35 Days
- Initial schedule = 14.34 Days / 5 days a week = 2.27 Weeks
- Team size = 111.18 PD / 11.35 D = 9.79 P = 10 Persons
- Working hours include 8 hours in a working day.
- Total person-hours (PH) = 111.18 PD × 8 hours = 889.44 PH

Based on real world constraints, we have a team size of 7. Adjusting for this while maintaining the effort required of 111.18 PD, and maintaining 8 hours of work per day with a team size of 7 members, the adjusted duration would be approximately 3.18 weeks.

5.2.1 Distribution of Effort

1990's Industry Data	Work Package	Distribution	Estimates
Preliminary Design 18 %	Project Plan	9%	80.05
	Requirement Specification	9%	80.05
Detailed Design 25 %	User Interface	7%	62.26
	Technical Architecture	11%	97.84
	Data Modeling	7%	62.26
Code & Unit Testing 26 %	Code & Unit testing	21%	186.78
	Online Documentation	5%	44.47
Integration & Test 31 %	Integration & Quality Assurance	31%	275.73
	Extrapolated total effort		889.44
	2% for project management		17.79
	3% for contingency		26.68
	Total effort		993.91

These duration estimates are based on the assumption that each team member works an equal amount on any given work package.

[1] Lines of code per Person Day statistics based on Industrial Benchmarks, 1997: 31 LOC/PD for United States; 62 LOC/PD for Canada

5.3 Cost Estimates

Hardware:

Developer workstations:

7 - Personal Laptops	Total $0.00

Software:

Free License-based Software:

Github	$0.00
ClerkJS API	$0.00
Google Books API	$0.00
New York Times API	$0.00

Software License Provided by Third Party:

Microsoft 365 Apps for enterprise	$0.00
Google Apps	$0.00
draw.io	$0.00
Canva	$0.00

Other Resources:

Staff:

7 Employees with 993.91 working hours with $18.00/hour	$17,890.38

Total: $17,890.38

Mawsters 's hardware and software responsibilities relate only to our own development needs to accomplish the project we have been asked to complete, and which has been described in the introduction section of this document. Mawsters will also demonstrate the completed product.

6 Product Checklist

The plan is that the items listed below will be delivered on the stated deadlines.

Deliverable	**Estimated Completion Date**	**Final Deadline**
Meeting Minutes	Every meeting	
Backlog	Updated after every lab	
Project Proposal	5/2/2024	8/2/2024
Use Case Model	5/2/2024	8/2/2024
Software Requirement Specification	19/2/2024	22/2/2024
Quality Plan	19/2/2024	22/2/2024
Project Plan	11/3/2024	14/3/2024
Risk Management	11/3/2024	14/3/2024
Prototype Code	12/3/2024	14/3/2024
Prototype Documents	12/3/2024	14/3/2024
Design Report on Software Maintainability	25/3/2024	28/3/2024
Configuration Management Plan	25/3/2024	28/3/2024
Change Management Plan	25/3/2024	28/3/2024
Release Plan	25/3/2024	28/3/2024
Presentation Slides	8/4/2024	11/4/2024
Test Plan	8/4/2024	11/4/2024
Test Cases & Requirements Report	8/4/2024	11/4/2024
Final Documentation	9/4/2024	11/4/2024

7 Best Practice Checklist

Practice	9
Document what we do; all documentation must be in a standardised format.	
Pay attention to requirements, and check for ambiguity, completeness, accuracy, and consistency. The requirement documentation must contain a complete functional specification.	
Keep it simple. Complexity management is one of the major challenges. Strive to: • Minimise interfaces between modules, procedures and data. • Minimise interfaces between people, otherwise exponential communication cost • Avoid fancy product functions, and design as long as the functionality meets the customer requirements	
Require Visibility. We must see what we build otherwise we can measure the progress and take management action. This includes: the manager must have good communication with his or her employees; require developers to make code available for review; and review design for appropriateness.	
Plan for continuous change. We must: • All manuals designs, tests, and source code should have revision numbers and dates revision history comments, and change marks to indicate the changes • New revisions should be approved before being made and checked for quality and compliance after being made • Use a configuration management system and make processes • Required maintenance	
Don't underestimate. We must be careful to obtain accurate estimates for: time, effort, overhead, meeting time, and especially effort on integration, testing, documentation and maintenance.	
Code reviews are a much more efficient method to find software defects. Plan and manage code reviews between team members	
Software testing will use both black box and white box testing. It will involve unit, functional, integrating and acceptance testing.	

8 Risk Management

Besides the general risk management, the following risks have been identified for the shelvd project:

More changes to requirements than anticipated
Impact Severity: High
Probability: 25%
Impacts: Depending on the stage at which changes occur, could range from needing to update the requirements documentation to needing to do a complete redesign.
Risk Reduction: Be rigorous in eliciting requirements. Make customers aware of the potential repercussions of requirement changes.

Unable to meet Project Deadline
Impact Severity: High
Probability: 1%
Impacts: Reduced functionalities or delays in deployment.
Risk Reduction: Have regular meetings and updates from the development team to check on their progress.

Staff leaving before project completion
Impact Severity: Extreme
Probability: 1%
Impacts: There would be more work for remaining employees, and any specialized skills or knowledge would be lost.
Risk Reduction: Offer benefits and incentives to staff.

Updates to code not communicated to team
Impact Severity: Moderate
Probability: 30%
Impacts: Overlapping/overwriting of codes may lead to errors in running the application
Risk Reduction: Ensure updates are communicated whenever new code is pushed into the repository.

Customer cancels the project
Impact Severity: Extreme
Probability: 1%
Impacts: All work done will have been wasted.
Risk Reduction: Keep in close contact with customers. Ensure that they have some market research indicating a demand for this project.

9 Quality Assurance

The project will achieve quality assurance by following the standards set by the team. The specific procedures and details shall be provided in the Quality Plan.

In addition, two testing methodologies shall be used:

- **Unit Testing** involves testing system components individually.
- **In-place testing** involves testing the whole system as a unit.

Furthermore, these methodologies will be used to test two important of shelvd

- **System Function** will be tested to ensure that software flaws are eliminated.
- **Algorithmic Functions** will be tested to ensure that heuristic aspects of the project perform realistically to provide value to users.
- **Load Capability** involves testing the system performance under heavy traffic

Shelvd methodology makes use of realistic test cases. Detailed test data is an important aspect of the final project delivery. Shelvd shall provide a comprehensive and detailed subset of this data for testing purposes.

10 Monitoring & Control

Many procedures are required to be able to successfully monitor the progress of a software project. Some of the most important are:

Quantitative measurement of resource consumption: Estimates resources needed for the project with its function points. Enables tracking of progress and aids in determining the necessity of resource reallocation

Identification of major project risks: Early identification of major risks to the project allows for placement of preventative measures before problems can develop. Major risks have been identified in the Risk Management section of this document, along with the measures being taken to avoid them.

Regular reviews of project progress: Throughout the project, the team shall meet weekly to review the progress of all project tasks, including management, planning, analysis, development, and testing.

Timeline Planning and task decomposition: This document outlines an estimated timeline for the project. A reasonably accurate timeline can be assembled by hierarchically decomposing tasks into measurable subcomponents and estimating requirements for each. At the same time, this decomposition can assist in task assignment and balancing. Throughout the implementation phase, these subcomponents can allow for fine-grained measurement of progress. Project subcomponents and timeline estimates are included in the Estimates and Work Breakdown Structure sections of this document.