Software Development Description (SDD)
For
Bookwyrm

Our Mission: To increase focus and concentration during reading

Group 4: Ayrton Massamba, Sean Huber, Joey Maggitti, Ben Rieland, Peter Stein, Leon White

Due: 10/29/2020

1. Scope.

1.1 Identification.

This software is a focus/concentration software. Our plan is to find ways to stimulate reading habits. Time management in this exercise is therefore a central piece, as the goal is to isolate small chunks of time during one's day to enable constructive reading. Since we have observed that the lack of focus or attention during reading can lead to poor reading habits, a tool to improve focus would be very helpful. In conclusion, the software at its basic core is very simple, but the different features that can be added to it could be very beneficial.

Titles: Bookwyrm

Identification number: 2.0

Current version of software: Reading Space 2.csproj **Release number**: 0

1.2 System overview.

The purpose of the system/software is to promote active reading. The system works as a book database search engine, a follow-up checklist, a timer, a statistics report application, and a reader-reward-tool service.

The book database helps with the search of various book titles, authors/writers, and book covers/images (The data then can be added to the follow-up checklist, if a client wishes to do so.), the follow-up checklist is to be used to keep track of the diverse books of interest to the users (e.g. To-Be-Read,Completed, etc), the timer is used to organize reading time efficiently throughout the day (cutting a reader's reading objective in chunk of 15-30 minutes based a set schedule or a everyday 30 minutes a day reminder), a statistics report in then formulate to represent the user current and past usage of the application, and based on the use of the reader the reward system offers different badges/avatars (images).

The general idea of the software was first generated by Ayrton Massamba, but later on was polished by the team, which members are: Ayrton Massamba, Sean Huber, Joey Maggitti, Ben Rieland, Peter Stein, and Leon White. Pre-development, the team envisioned to use a programming platform such as visual studio or Unity. Moreover, the initial programming language selected for the software was C-Sharp on the visual studio platform. Operation for this project was to be done remotely via platforms such as Discord, Google Doc, and Blackboard Collaborate. Moreover, as a means of research/tutorials, various YouTube videos are promoted. The target demographic for this project are people with the desire to read, but who either lack motivation or time. Nonetheless, the team believes that potential other demographics can be essential users.

1.3 Document overview.

- Section 1: Scope
- Section 2: Referenced documents
- Section 3: CSCI-wide design decisions
- Section 4: CSCI architectural design
- Section 5: CSCI detailed design
- Section 6: Requirements traceability
- Section 7: Notes

Privacy and security concerns are considered beyond the scope of this project.

1.4 Relationship to other plans.

A Software Design Plan and Software Requirements Specification have been written for this software. Research on how to approach creating an app on Unity and a decision to use Unity as a base application and Google's books API for book information have been determined.

2. Referenced documents.

Youtube tutorial videos were used for this project and therefore they will be cited as sources/references. The group decided on the videos as a way to accommodate the team on a peculiar coding platform. Moreover, these videos served as a way of communicating ideas on which programming language should be selected. Therefore, the initial and current videos used so far are as follow:

Initial videos (for software identification number: 1.0)**:**

1) **How to Developed Application for Fast Foods Restaurant in C Sharp**

   https://www.youtube.com/watch?v=peb7kO3GDuQ&t=1045s

2) Designing a Modern Flat Desktop Application of a Fast Food Restaurant in Visual Basic VB NET

https://www.youtube.com/watch?v=znANTJNf8wI&t=522s

3) Modern Flat UI, Drop-down/Slider Menu, Side Menu, Responsive, Only Form - C#, WinForm

https://www.youtube.com/watch?v=JP5rgXO_5Sk&t=79s

4) C# Tutorial - Circle Progress Bar | FoxLearn

https://www.youtube.com/watch?v=o7MGaf8YW6s

Current videos (for software identification number: 2.0)**:**

1) **How to make UI in UNITY - EASY TUTORIAL**

   https://www.youtube.com/watch?v=_RIsfVOqTaE

2) **Making UI That Looks Good In Unity using Color Palettes, Layout Components and a Blur Panel**

   https://www.youtube.com/watch?v=HwdweCX5aMI

3) **START MENU in Unity**

   https://www.youtube.com/watch?v=zc8ac_qUXQY

3. CSCI-wide design decisions.

3.1 CSCI Inputs

The CSCI will have inputs for various actions in the form of clickable/pressable (for touch screen devices) buttons. Actions include but are not limited to: Load book, next page, back page, create note at page, view account, change settings.

3.2 Response Performance

Load Book: can be a little slow if loading the entire text, will require navigation to a file/uploading a file

Next/Back page: should be VERY fast, will load next page (one at a time, not mimicking 2 page book format)

Create Note at page: should be fast, should open new section to add text memo and should save to specific page

3.3 Appearance

Everything should be relatively minimalistic as we do not want to interfere with any of the material on screen from a given book

Menus will be mostly bound to the sides of the application so as to take advantage of the inherent blank space in page margins

3.4 Safety, security, and privacy

No real safety or security issues to deal with. Privacy should be as simple as requiring a password. No payment/personal ID is necessary for app functionality and therefore enhanced security does not require consideration

3.5 CSCI-wide design decisions

Game engine: Unity since it allows easy portability to other devices, operating systems, and platforms

Book information: google/library of congress api’s for book information procurement

Assumptions/Dependencies: Internet access, google/library of congress api’s active

4. CSCI architectural design.

4.1 CSCI components.

The following subsections address the hardware and the software components of the bookwyrm application. These components refer to the required hardware need to run the program and the interactive tools available to the user.

The hardware recommendations and components or buttons associated with C# scripts designed by the team are as described in the remaining subsections.

The hardware resources used are as follows:

- Local computers or smart devices: The machine that hosts the Bookwyrm application.

The software resources used are as follows:

- Menu Unit: Custom design of buttons associated with C# scripts that lead to a particular frame or panels. For example, the dashboard button would bring the user to the panel tool that visually tracks, analyzes and displays the user's performance with key messages/or notifications.

- Statistics Unit: It is the background process that collects data and uses it to feed the dashboard or other panels that display relevant information.

- Reading Tracking Unit: It is another background process that is utilized to track the actual reading activity of the reader. It is meant to be an automatic process (no direct user input necessary. The tracking is done by following where the user is on his/her ebook) and a manual process (where the user direct inputs data after reading when prompted to; this is a feature for readers who prefer physical books). It also encloses the timer as it allows the application to track the reading sessions of the user.

- Bookshelf Unit: This is the crucial part of the software as it helps in stimulating reading by allowing the user to create a collection of books to be read (TBR), books currently being read (CBR), and books completed (BC). The visual aspects of this section is the key element to keep the user engaged.

- Book Search Unit: This is an interactive and background process that is part of the Bookshelf unit where the user can request a book via search through a book database. The user after the search gets access to a list of affiliate books based on his/her search attempt. From there, they can decide which book they wish to add the bookshelf. Most likely in the books to be read (TBR) category.

- Rewards Unit: This is a dynamic rewards system that works based on information collected on the user reading progresses or accomplished goals. Rewards are assigned based on the current level of the user (e.g level 0 to level 100)

- Notification Unit: This unit is a background process that reminds the user to engage in reading activities.

- Scheduling Unit: This unit is used by the user to organize the date/time of the reading session. It would most likely be a dynamic calendar that works in close association with the *Notification Unit*.

- Note Taking Unit: This is a simple note feature that users may want to use during reading sessions to have a better understanding of the book.

- The Settings Unit: This unit is meant to give the user as much control as possible on the application. Some users, for example, are very self-disciplined and probably not want to have notifications popping up all the time. So, the settings would allow the user to control or adjust various application components.

- Group/Community Unit: This unit has to do with a social media environment that the team discussed for future development. It is a reading community based feature that prompts users to engage friends and discuss or critique a book. (Book clubs).

Describe ID, purpose, (requirements per unit see 6a), development status/type

4.1.1 Hardware Usage

4.1.1.1 *Local PC or smart devices*

Bookwyrm should be able to function on a variety of devices that meets or exceeds the following requirements:

- Operating System (OS) Requirements:
    1) Windows OS (7 and higher)
    2) Mac OS (10 and higher)
    3) Ubuntu or Linux (Newest versions)

- Hardware Requirements
    1) Graphic cards that enhance Unity softwares
    2) CPU that can handle Unity software applications

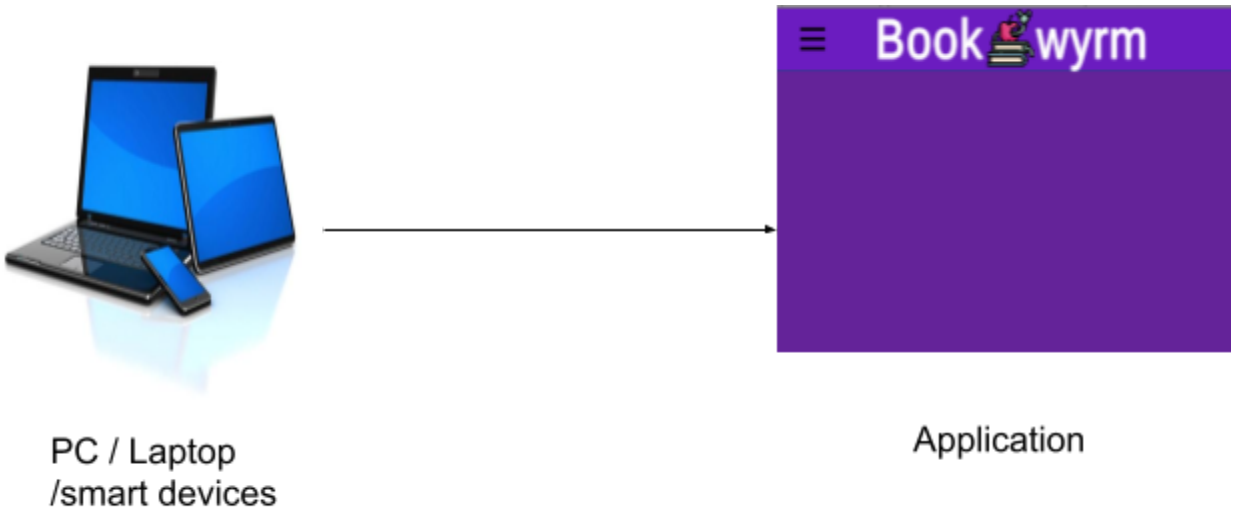


Figure 1

4.1.2 Menu Unit (BW_SWU_MU)

The menu panel consists of five buttons so far. Among which are the *dashboard* button, the *Reading Zone* button, the *My bookshelf* button, the *Rewards* button, and the *Analytics* button. These buttons are mostly to remain at the completion of the application. However, the team may decide on adding additional buttons or a change of those initial buttons along with their C# scripts.

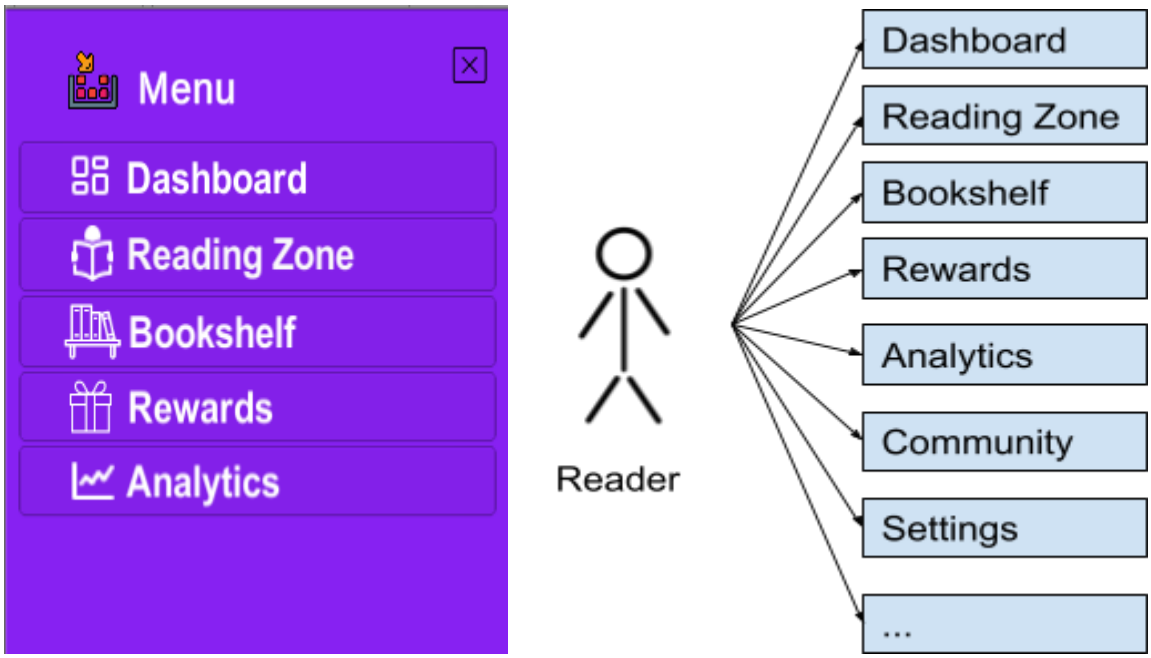


Figure 2

4.1.3 Statistics Unit (BW_SWU_SU)

The Static unit processes information based on the user reading habits and utilizes its given panel to display the information with graphs and other visual aspects. The image below (figure 3) represents

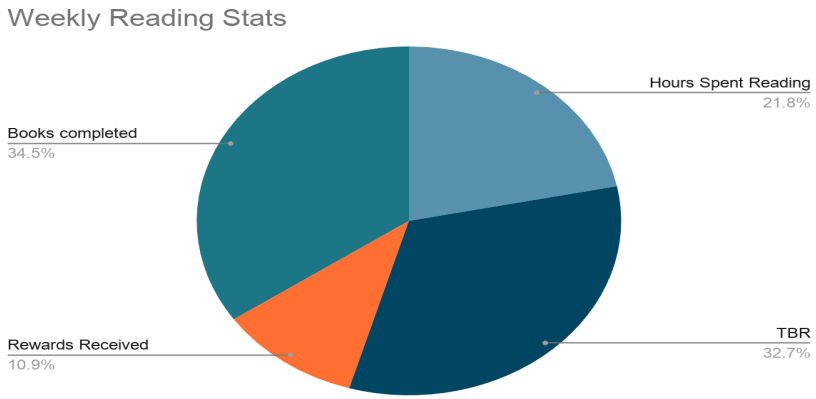


Figure 3

4.1.4 Reading Tracking Unit (BW_SWU_RTU)

This feature is used to collect information from the user about where they are reading. The team means to use an automatic approach and manual approach for this part. Also, the unit works in close proximity with the statistic unit as it feeds the *Statistics unit* data.

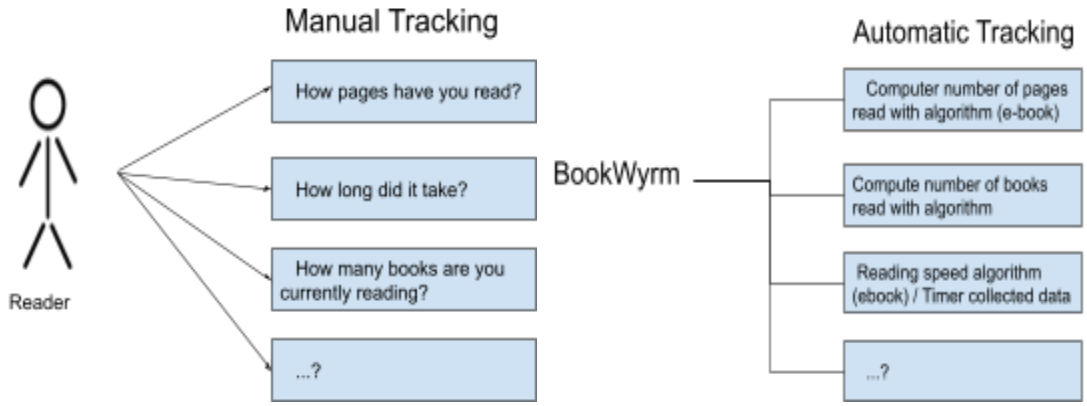


Figure 4

4.1.5 Bookshelf Unit (BW_SWU_BU)

This unit is the primary attraction of the application. It allows the reader to organize and collect books. Making the bookshelf visual visually attractive and informative is very important, as there is no doubt the user would spend a considerable amount of time on this unit.

Book Completed	To Be Read (TBR)	Information & Image
Categories (Scifi, Non-fiction, …)		
Book #1	**Book #2**	**Visual & Details**
Book #3	**Book #4**	**Visual & Details**
…	**...**	**...**
...	**...**	**…**

Table 1

4.1.6 Book Search Unit (BW_SWU_BSU)

This unit does the heavy-lifting for the Bookshelf Unit. It is a search engine that finds images, summaries, authors' names, and other information that would help the user create their bookshelf or just get specific information about a book. The picture below (figure 4) is an example of a Google search on "Harry potter books". Ideally, this unit would display results in a similar fashion.

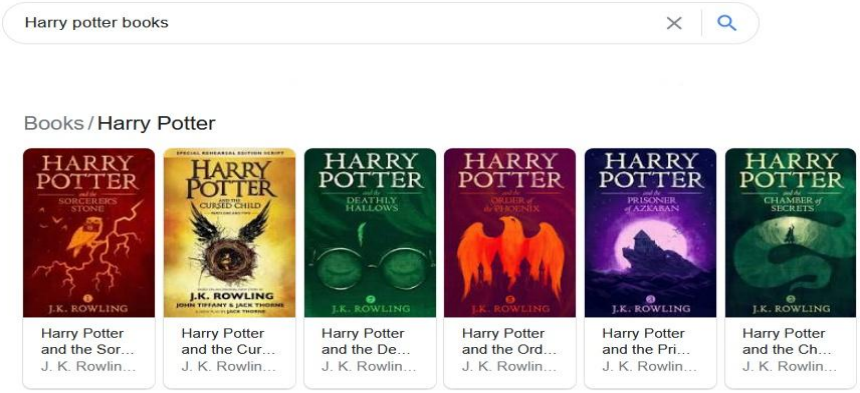


Figure 5

4.1.7 Rewards Unit (BW_SWU_RU)

The reward unit is the secondary most important component of this application in terms of keeping the reader interested or engaged. Research has proven that when an activity is associated with any sort of reward system, participants are prompted to willingly do the work. Just like the bookshelf, the reward unit is a visual/psychological stimulus.

Figure 6

4.1.8 Notification Unit (BW_SWU_NU)

The notification unit is very standard of the type of notification one can expect to receive on any application. The unit provides notifications or reminders via pop-up windows messages. (The possibility of sound associated with such notification is being considered by the team) Moreover, the notification unit works closely with the scheduling unit and statistics unit.

Figure 7

4.1.9 Scheduling Unit (BW_SWU_SCU)

This is a dynamic calendar that the users can use at their likings to choose the set of time they want to read. The users reserve a head date and hours which are convenient for reading.

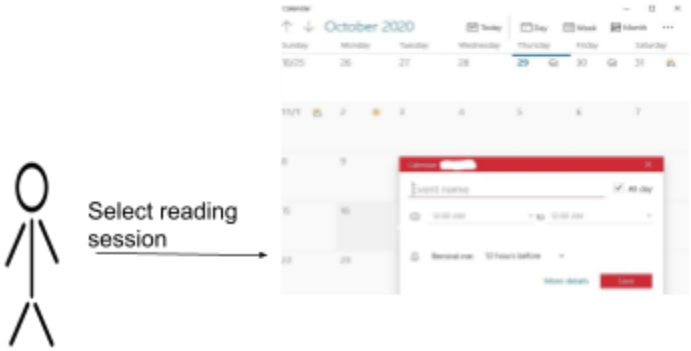


Figure 8

4.1.10 Note Taking Unit (BW_SWU_NTU)

The note taking unit or component is just a rumenditary note that the users utilize to process their current thoughts during the reading sessions.

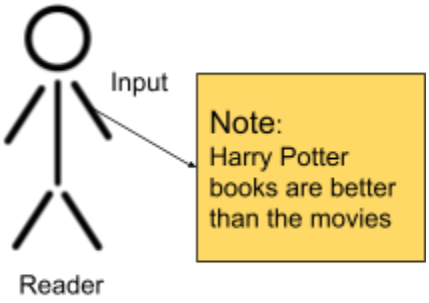


Figure 9

4.1. 11 The Settings Unit (BW_SWU_SEU)

This is a common component present in most applications and would be used appropriately so. Meaning that the team would create categories of factors that users can control making each individual experience unique or to one's preference.

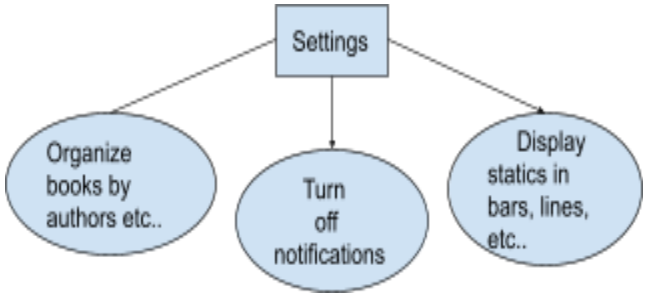


Figure 10

4.1.10 Software unit relationships/Community Unit: (BW_SWU_CU)

The purpose of this unit would be to create a social environment around the application. The team would most likely not fully launch with this unit, but would explore the possibility of it for future events. This is because the work needs in depth thinking. It is basically an application of significant size within another application of significant size. Therefore, a button may be added to the menu unit about it, but just to display ideas of future developments.

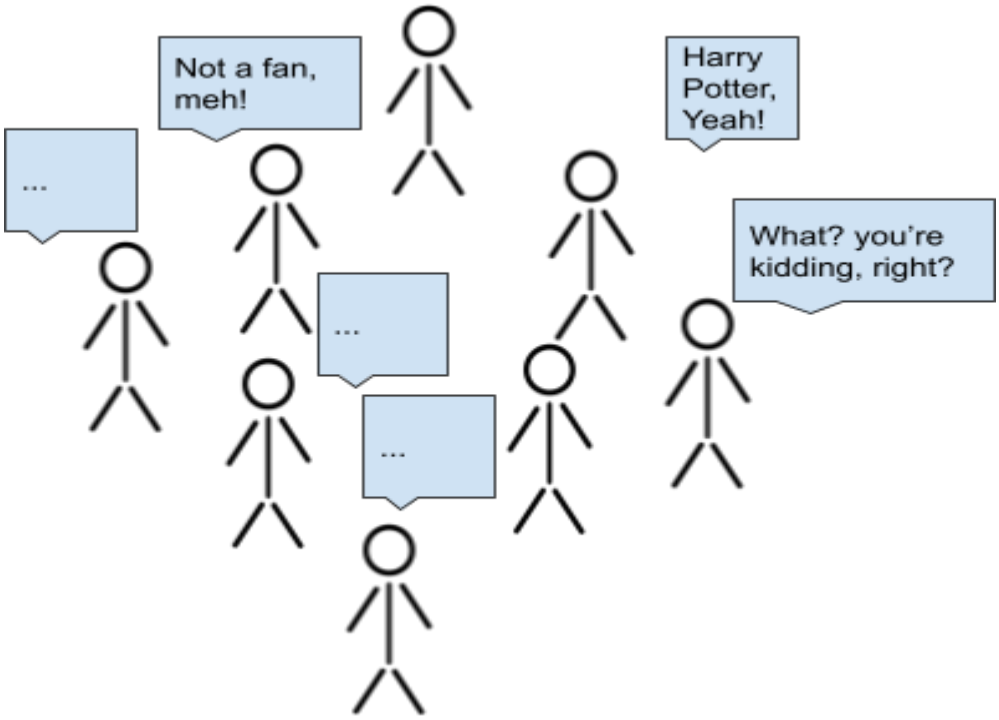
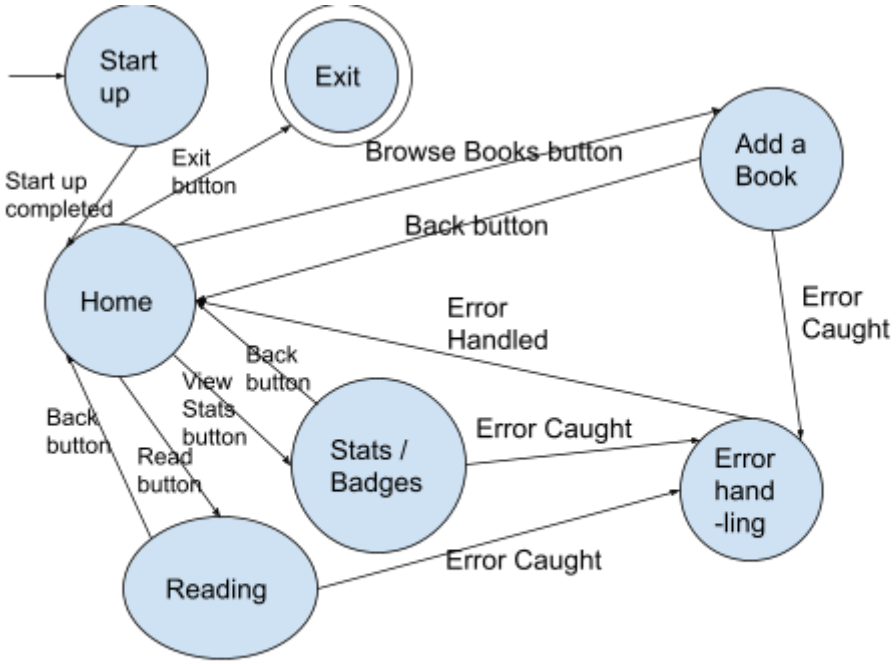


Figure 11

4.2 Concept of execution.

4.2.1 Flow of execution

The state diagram below (figure 11) is the high level depiction of the transitions between program states during execution. When the program loads up, it initializes to the home screen. Data is loaded when it is needed, such as book images and progress if looking at the bookshelf or reading data if statistics are being viewed.



Note: “Button” in this context is a stand-in for whatever method the UI is designed to use to let the user choose which state to go to.

Figure 12

4.2.2 Software unit control flow

The control flow state diagram below (figure 12) shows how the different software units pass around control in the program. Control flow specifically for adding a book can be seen in section 5.1.5.

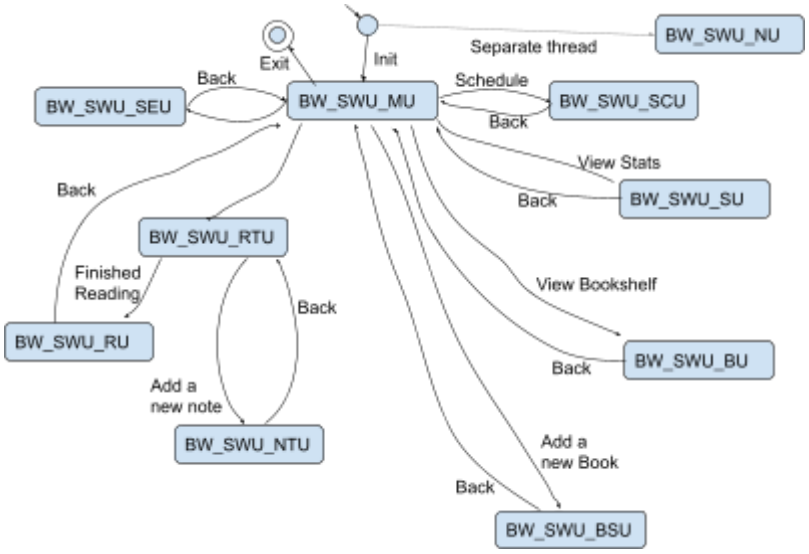


Figure 13

4.2.2 Software unit data flow

The data flow diagram below (figure 13) shows how each of the different software units accesses data from different local save files, or how it passes that data between software units. For a unit by unit description of where data is coming from and what data is saved, see section 5. Some data flow not depicted in this diagram for clarity includes settings values being sent to various software units that display things and user input is also not shown.

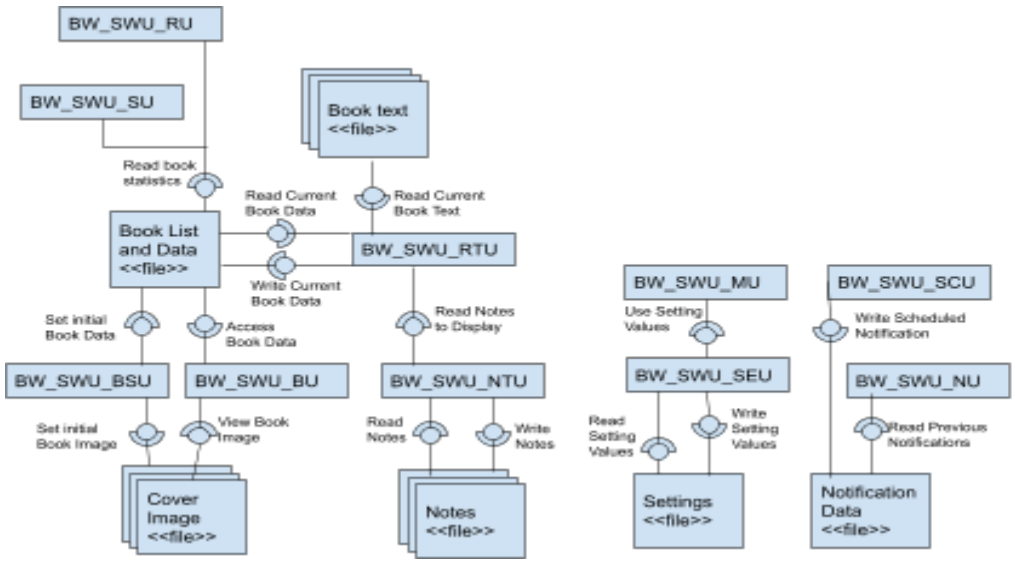


Figure 14

4.3 Interface design.

4.3.1 Interface identification and diagrams.

The project consists of three interfaces:

- Application Interface (4.3.1.1)
- Device Storage Interface (4.3.1.2)
- Google Books API Interface (4.3.1.3)

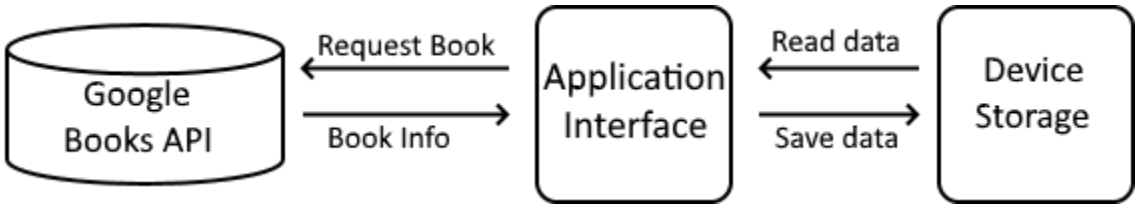


Figure 15

4.3.1.1 Application Interface

The Application Interface is the interface which initiates requests to the other interfaces. The Application Interface will initiate requests to the Google Books API Interface to request book information (see 5.1.5, figure 14, and section 4.3.1.3 for more details). After receiving data from the API, specific data will be cached and stored.

The Application Interface will also pull data from device storage by interacting with the Device Storage Interface. This is a separate interface because storage could be handled differently depending on the type of device and where the data is being stored. See 4.3.1.2

4.3.1.2 Device Storage Interface

The Device Storage Interface will retrieve data stored by the device and return it to the Application Interface. This interface is necessary because device storage could be complex depending on how and where the device stores the data. See section 5.1.4 on what kind of data is stored in the device.

4.3.1.3 Google Books API Interface

The Google Books API Interface is the interface which fetches book data and returns it to the Application Interface. The API only needs a single string search term as its input. For output, see section 5.1.4 on what type of data could be returned and used by the Application Interface.

5. CSCI detailed design.

5.1 BW_SWU_MU

The menu unit is responsible for navigating between the other software units and processing user input. The menu unit receives user input such as typing into textboxes and clicking/moving to different buttons. Upon exit, the menu is responsible for ensuring other units save user data locally. The menu outputs a navigation-related display and changes in relative program state. For how the navigation corresponds to state, see the diagram in section 4.2. All units are planned to be written in the C# programming language.

5.2 BW_SWU_SU

The statistics unit should calculate and display relevant statistics about the data recorded by the reading tracking unit (BW_SWU_RTU). It should include average reading time per week (or other unit of time), reading rate, total pages read, total books read, and average pages per book. This unit should also include badges the user has earned through reading. The badges should be displayed in such that the user can easily see how they were earned. The statistics unit will be responsible for determining if a badge has been earned using data from the reading tracking unit. For details, see the table in section 5.1.4. The input to the statistic unit is the local saved data from the reading tracking unit. The output is statistics which are displayed to the user. All units are planned to be written in the C# programming language.

5.3 BW_SWU_BU

The bookshelf unit should load the covers of each book to display on a virtual bookshelf with pages drawn to give the books depth depending on progress and book size. The unit does not receive input but does use the local book data stored (see table below). The output is the bookshelf display to the user. All units are planned to be written in the C# programming language.

5.4 BW_SWU_RTU

The reading tracking unit should keep the local data updated with information on books currently being read. The data to be included in the file is seen in the table. Since reading and writing to files is costly, the tracking unit should only save/load data to the main local save file at the beginning and the end of a reading session. A secondary temporary file may be used during reading if desired to restore lost progress in the event of a crash or unexpected exit. All units are planned to be written in the C# programming language.

The table below summarizes data recorded for each book:

Variable	Type	Time Recorded	Description
TotalElapsedReadTime	Integer	“Real time”	The amount of time the user has been recorded reading any book ever
TotalElapsedBookReadTime	Integer	“Real time”	The amount of time the user has been recorded reading the specific book ever
CurrentBookReadTime	Integer	“Real time”	The amount of time the user has been recorded reading the specific book
StartPage	Integer	At reading end	Page the app will start on next time
CurrentPage	Integer	“Real time”	The page of the current book the user is currently reading
TotalPages	Integer	At book addition	The total pages in the specific book
DateBookStarted	Integer	At first book open	The date the specific book was first opened
BookTitle	String	At book addition	The title of the specific book
CoverImagePath	String	At book addition	(Optional) The file system path to the cover image of the specific book.
TextPath	String	At book addition	(Optional) The file system path to the text of the specific book.

Table 2

5.5 BW_SWU_BSU

The book search unit will be responsible for generating a new book object to be used internally to follow each book. This includes book data such as title, author, page numbers, but also location of a cover image, if found, and location of a pdf of the text if available. The unit shall prompt the user for info, then run a search through Google’s Books API, (an alternative was the Library of Congress based API, but the Google API appeared easier to use). If information is found, it will auto populate the object, ask the user for confirmation and for any missing information, then add the book (See figure 15). The documentation for the API can be found at: https://developers.google.com/books/docs/v1/reference/ (also see section 2). For the logic to be used involving paths through this software unit, see the activity diagram and use case diagram and description below (figures 15 and 16). There is no language requirement for the bookshelf unit. Unity is designed to run on C# scripts so that is the desired language.

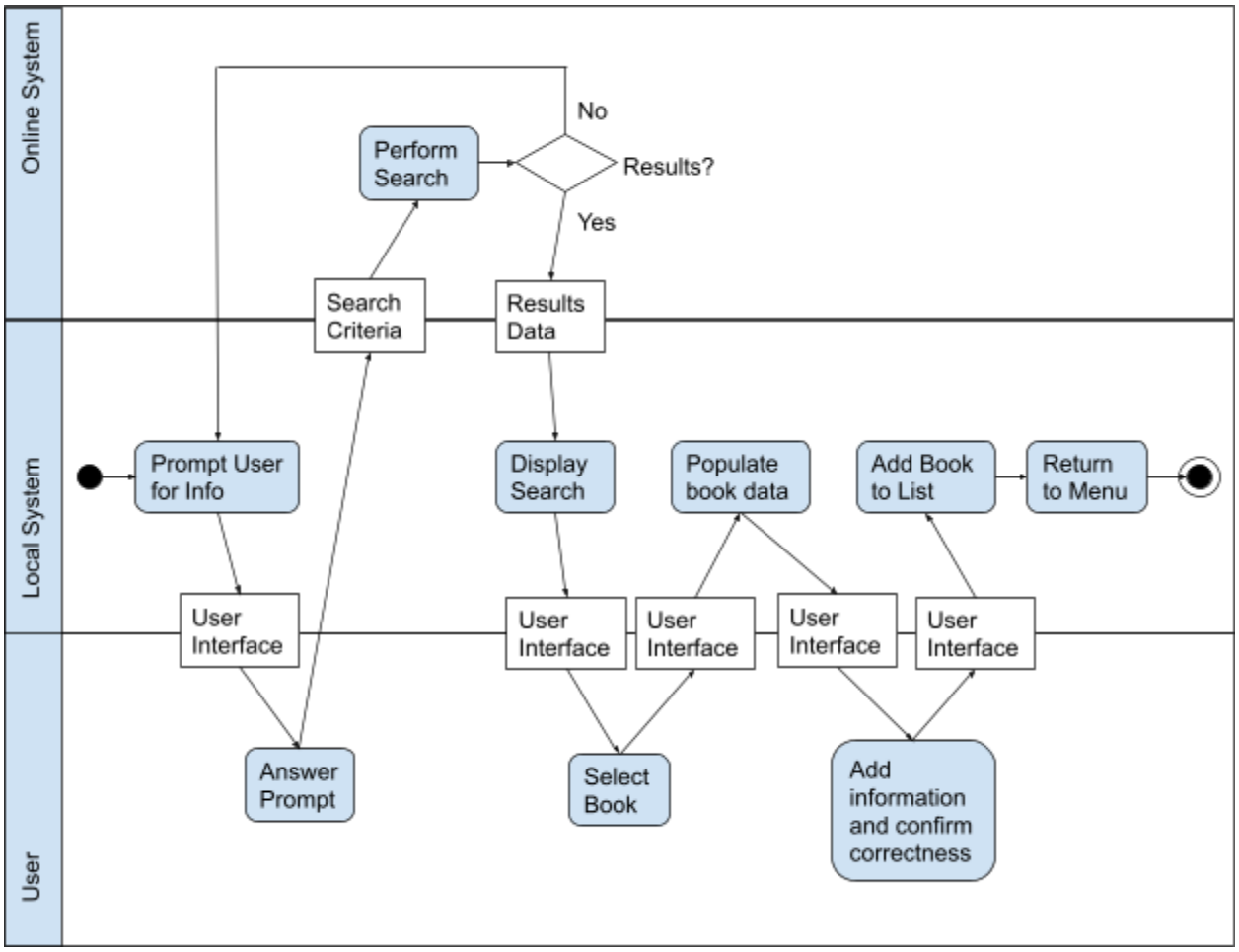


Figure 16

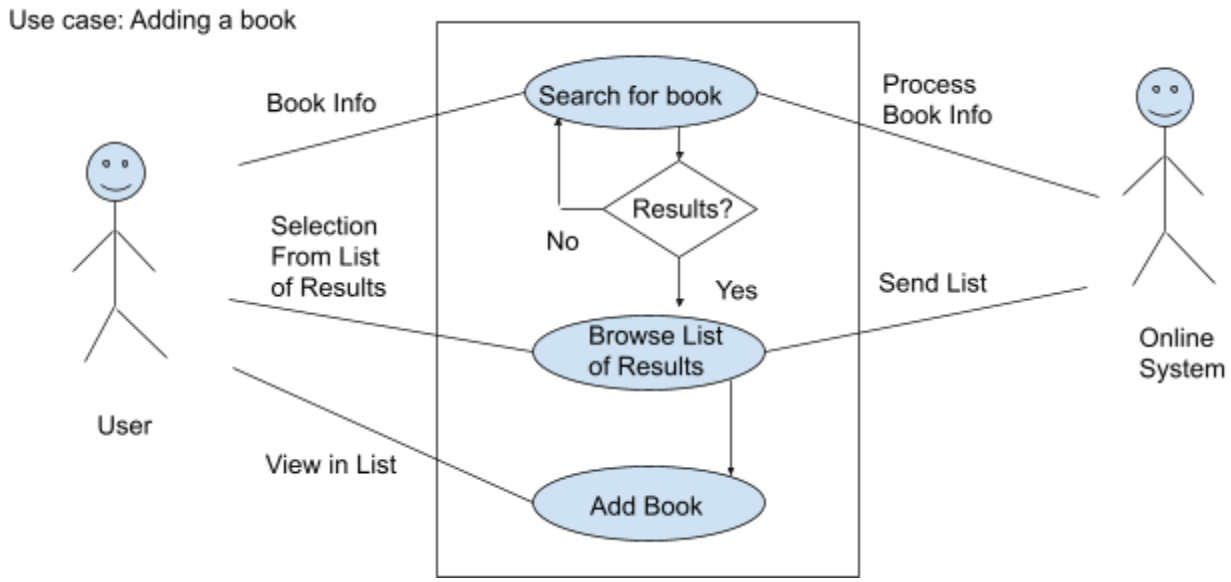


Figure 17

Name	User Adding a Book to be read
ID	BW_UC_002_UserAddBook
Description	The user wants to add a book to the to be read list so it can be tracked as he/she reads it later.
Actors	User, Online System
Organizational Benefits	This use case is vital to using the program. If books cannot be added, they cannot be tracked, and the app loses major functionality.
Frequency of Use	Weekly or more often (When browsing for books or when a book is completed)
Triggers	The user will select an option to add a book to be read.
Preconditions	There must be an internet connection (there must be access to the online system).
Postconditions	There will be a book added to the To Be Read list if the user completes the process without quitting.
Main Course	1. User chooses to search for a new book 2. They send some info to search on, which gets processed by the online system. 3. The main system displays the books that are a result of the search to the user 4. The user selects a book 5. The selected book is added to the to be read list for the user.
Alternate Courses	1. The user clicks a button to manually enter a book. 2. The user manually enters the books name 3. The user clicks a button to add the book to the system.
Exceptions	EX1: <No results are returned from the book search> 1. The user cannot continue through the process without results, so the user should be prompted to try a different search. 2. The user is also prompted to manually add a book via Alternate Course #1 EX2: <The user decides to exit the process> 1. The use case is abandoned, no books will be added to the to be read list. 2. This should be handled for every state in the process.

Table 3

5.6 BW_SWU_NU

The notification unit is responsible for running in the background and notifying the user when they should consider reading. It takes input from the scheduling unit, and it's output is a notification. It should save it's notifications locally, so in the event of being closed notifications are not lost. See the table in section 5.1.7 (table 4) for the types of data to be saved. All units are planned to be written in the C# programming language.

5.7 BW_SWU_SCU

The scheduling unit is responsible for helping the user set up times and rules for notifying themselves to spend time to read. The user is not required to schedule notifications. The input comes from the user and should be type-checked. The output goes directly to the notification unit to be saved, it also triggers the notification unit. See the table below (table 4) for the data and types to be sent to the notification unit. All units are planned to be written in the C# programming language.

This is the data sent per notification

Variable	Type	Description
Repeating	Boolean	Whether this notification should happen every so often or just once
StartTime	Integer	The first time the notification is set up to ring
TimeBetween	Integer	The time between repeats of the notification

Table 4

5.8 BW_SWU_NTU

The Note Taking Unit is responsible for allowing the reader to add notes on a page per page basis. These will be saved locally so that any time the reader returns to that page, the notes can be read. The note taking unit takes user input and saves it in association with the page. It's output is displaying the notes back to the reader. All units are planned to be written in the C# programming language.

5.9 BW_SWU_SEU

The settings unit is responsible for helping the user change ease of access settings like text size, resolution, and data management (like deleting saved data). The input comes from the user in the form of buttons and from a locally saved settings file. The output goes back to the locally saved settings file. See the table below (table 5) for the types of data to be saved. All units are planned to be written in the C# programming language.

Variable	Type	Description
TextSize	Floating point	A multiplier to scale the text in the menus for better readability
ResolutionWidth	Integer	The number of pixels across the app is
ResolutionHeight	Integer	The number of pixels tall the app is

Table 5

6. Requirements traceability.

This section shall contain:

1. Traceability from each software unit identified in this SDD to the CSCI requirements allocated to it. (Alternatively, this traceability may be provided in 4.1.)
2. Traceability from each CSCI requirement to the software units to which it is allocated.

Req.	Description	SRS Par. #	Software Unit
BW_F_1	Remember the list of books to be read between launches	3.2.1	BW_SWU_RTU
BW_F_2	Remember recorded statistics between launches	3.2.2	BW_SWU_SU
BW_F_3	Remember any in progress books and their state between launches	3.2.3	BW_SWU_RTU
BW_F_4	The software shall remember the list of books completed between launches	3.2.4	BW_SWU_RTU
BW_F_5	Type-check input and re-prompt the user if the type is wrong	3.2.5	BW_SWU_BSU
BW_F_6	Use the remote information source to acquire information on books to be added	3.2.6	BW_SWU_BSU
BW_F_7	Keep track of user progression through current books	3.2.7	BW_SWU_RTU
BW_F_8	Allow the user to record and display notes on a page by page basis.	3.2.8	BW_SWU_NTU

BW_F_9	Remember recorded notes and associated page numbers between launches	3.2.9	BW_SWU_NTU
BW_P_1	Include Alert times or reminders to motivate users	3.3.1	BW_SWU_SCU , BW_SWU_NU
BW_P_2	Include Books read and planned to be read checklists to keep track of user progress	3.3.2	BW_SWU_RTU, BW_SWU_BSU
BW_P_3	Include data that would help create statistics	3.3.3	BW_SWU_RTU
BW_P_4	Keep progress of current book on th to-be-read list up-to-date	3.3.4	BW_SWU_RTU
BW_P_5	Include an automated reward system	3.3.5	BW_SWU_SU
BW_P_6	Run on a system of the minimum requirements found in SRS section 2.4	3.3.6	All
BW_C_1	Initialize to the home state when loaded	3.4.1	BW_SWU_MU
BW_C_2	Have setting parameters	3.4.2	BW_SWU_SEU
BW_E_1	Do not corrupt saved data in the event of an internal error	3.6.1	BW_SWU_RTU, BW_SWU_SU, BW_SWU_BSU
BW_E_2	Do not crash nor exit due to connection to the remote book information source	3.6.2	BW_SWU_BSU
BW_E_3	Log any internal error it encounters, including error type	3.6.3	All
BW_E_4	Do not crash the user’s device upon encountering an error	3.6.4	All

Table 6

7. Notes.

We may add functionality based on our conversation with client Mariska van Delft. Any additions will be reflected in an updated version of this SDD.

A. Appendixes.

There are no appendices for this document.