

**Mission:** Focus/Concentration Reading  
Application

**DID:** Software Test Report (STR)

**PREPARED by:** Group 4

Ayrton Massamba,  
Sean Huber,  
Joey Maggitti,  
Ben Rieland,  
Peter Stein,  
and Leon White

**DATE:** 12/8/2020

# 1. Scope.

This section shall be divided into the following paragraphs.

## 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.

**Names of the software:** Book Wyrn

**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 wish 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.

The software test report displays the results of test runs on the software to confirm that it meets its requirements.

## 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=znANTJNf8wl&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](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](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](https://www.youtube.com/watch?v=zc8ac_qUXQY)

### Software Requirement specification (SRS)

[https://docs.google.com/document/d/1t0MRxjIOIWxOdMWuwaQs7we6WcvF1\\_pFbE2bt0\\_tKGs/edit?usp=sharing](https://docs.google.com/document/d/1t0MRxjIOIWxOdMWuwaQs7we6WcvF1_pFbE2bt0_tKGs/edit?usp=sharing)

### Software Test Plan (STP)

[https://docs.google.com/document/d/1lQsevcybAjl6DTtMN0znj\\_1tP-WQoGnLgWJi2WO8qcw/edit?usp=sharing](https://docs.google.com/document/d/1lQsevcybAjl6DTtMN0znj_1tP-WQoGnLgWJi2WO8qcw/edit?usp=sharing)

## 3. Overview of test results.

This document shall describe the results of the action of the BookWorm Software Test Plan as defined by the Software Test Description. This section shall be divided into the following paragraphs to provide an overview of test results.

### 3.1 Overall assessment of the software tested.

Through the tests performed the Bookworm software performed as designed. It was successful in performing the requirements expected. The Bookworm application successfully added books based on the users input and from a remote database, was successful in saving and loading the state of the application and the data stored within it, was able to persist and not corrupt data through early termination or through unexpected crashes, and was able to smoothly operate even without a connection to any remote databases.

The only deficiency noted was with the use of blank data when testing the remote functionality of the app. As there was nothing to search for, or the data used as the search term wouldn't yield results, not having results would be the expected outcome. It's noted as a deficiency as that would also be the outcome of a failure on the application. There is no impact on the

performance of the software as other tests proved successful and have shown a fully functional application when a remote connection is established.

### **3.2 Impact of test environment.**

The application will be built in Unity then tested on multiple operating systems (Android, Windows, Mac, Ubuntu, and CentOS). This will ensure that the test environment represents the live environment which the application will be deployed into.

One of the advantages of using Unity is that the built application should perform the same as the operational environment. Thus, we can reasonably expect that the results of our application in the test environment will reflect how the application does in the operating environment. As a result, there should be no impact from using a test environment.

### **3.3 Recommended improvements.**

The app may function better if coded as a web app using JS instead of Unity.

## **4. Detailed test results.**

### **4.1 BW\_T\_EUI tests**

Erroneous user input tests ensure that users cannot crash the application or corrupt data through bad input. They are run at the software component level primarily, but may also be tested at the CSCI level. This is a testing method of test (as opposed to demonstration), and can be designed to use automatic input and output recording to automate the test. If developing this is prohibitive, the same test can be completed by a developer directly working with the program.

### 4.1.1 Summary of test results.

Requirement	Description	Test ID	Status
BW_F_5	The software shall type-check input and re-prompt the user if the type is wrong or if the data could not be interpreted	BW_T_EUI_1	all results as expected
BW_F_6	The software shall use user data in conjunction with the remote information source to acquire information on books to be added.	BW_T_EUI_1	deviations required
BW_E_3	The software shall log any internal error it encounters, including error type to a local log file.	BW_T_EUI_1	all results as expected

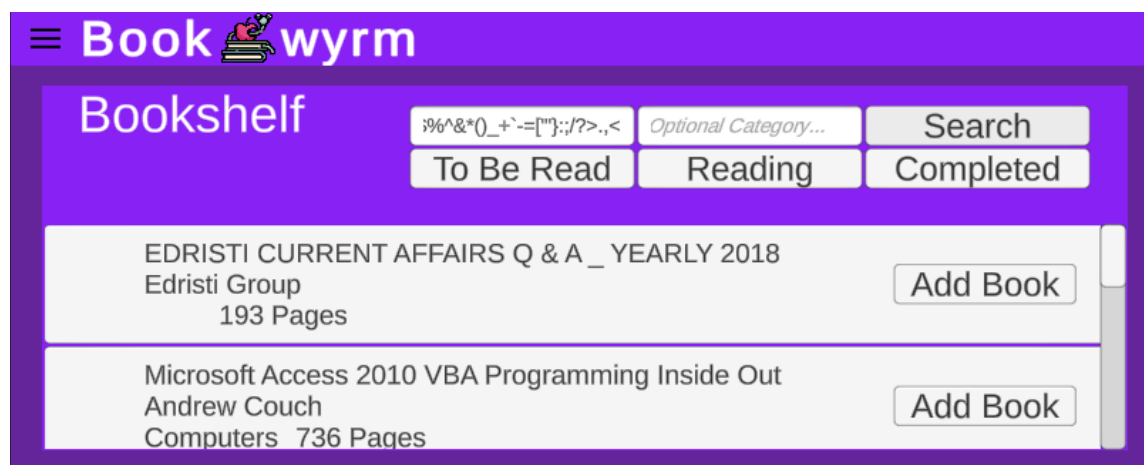
#### 4.1.1.1 BW\_T\_EUI\_1 Details

Process: Input characters that would be difficult for the program to handle. If the characters are invalid, check for a reprompt and details in a log file.

Data: Used ~!@#\$\$%^&\*()\_+`-="';:/?>.,< as the input string

Results: The string of special characters didn't create an internal error, as accurate results were returned. Since there was no error, the local log file was empty, and the user was not re-prompted due to a type-checking error.

Search results of string of special characters.



### 4.1.2 Problems encountered.

There were no software problems discovered during erroneous user input tests.

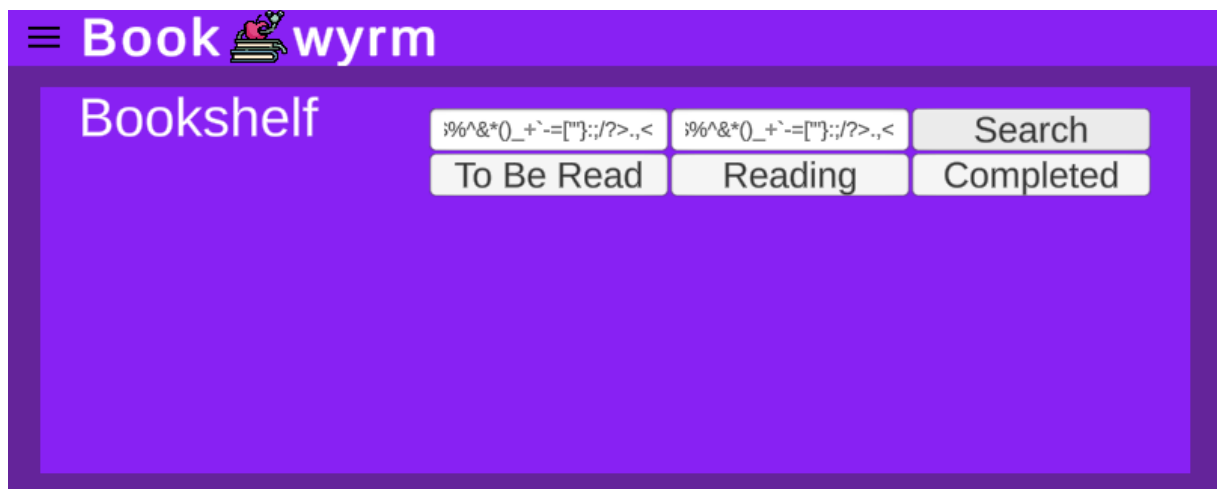
### 4.1.3 Deviations from test cases/procedures.

There were no deviations from procedures during erroneous user input tests.

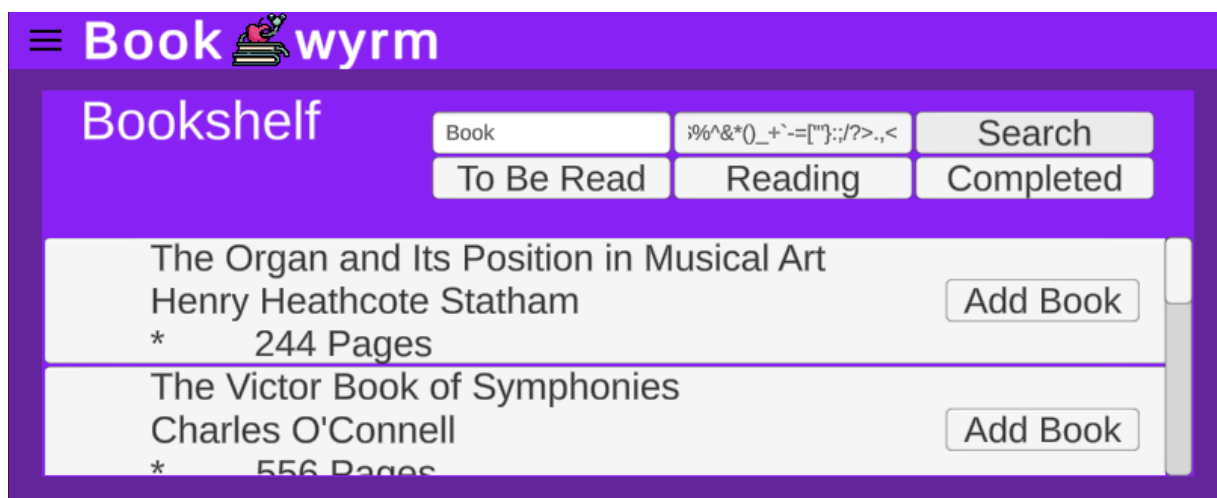
#### 4.1.3.1 BW\_T\_EUI\_1

When testing the string of special characters in the category output, no results were found. The test was repeated once more before changing the procedure to use an input that would return results. Both attempts before the change were inconclusive. A different search string was used to see if the problem was due to lack of results or an internal problem. This deviation was to explore an inconclusive result, and does not impact the validity of the previous test case.

Search results of string of special characters in both search term and category.



Search results of "book" in the category of the string of special characters.



## 4.2 BW\_T\_CSD tests

Corrupt save data tests ensure the application is capable of handling and responding to invalid saved data. They run at the CSCI level and are another testing method type of test. These involve designed incorrect save data to test loading problems. After the save file is written, it is manually loaded in the system. This test is run by a developer manually, and data is recorded through screen capture of the program's state and by resaving to see loaded data.

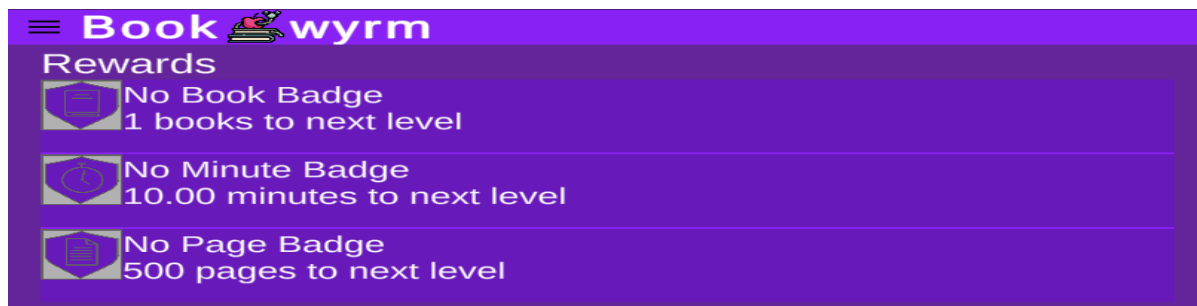
### 4.2.1 Summary of test results.

Requirement	Description	Test ID	Status
BW_E_1	The software shall not corrupt persistent data in the event of an internal error.	BW_T_CSD_1	all results as expected
BW_E_2	The software shall not crash due to the connection with the book information database being unsuccessful or the database returning invalid data.	BW_T_CSD_1	all results as expected
BW_E_3	The software shall log any internal error it encounters, including error type to a local log file.	BW_T_CSD_1	all results as expected

#### 4.2.1.1 BW\_T\_CSD\_1 Details

Process: Recompile the program to make it save in such a way that the output is corrupt. Reset to the previous version and run the program to have it load the corrupt file. Check for errors.

Results: The string of special characters didn't create an internal error, as accurate results were returned. Since there was no error, the local log file was empty, and the user was not re-prompted due to a type-checking error. Saving and using the program without internet access did not corrupt the save data. (tested in BW\_T\_DAR\_1). Program launched on empty save file.

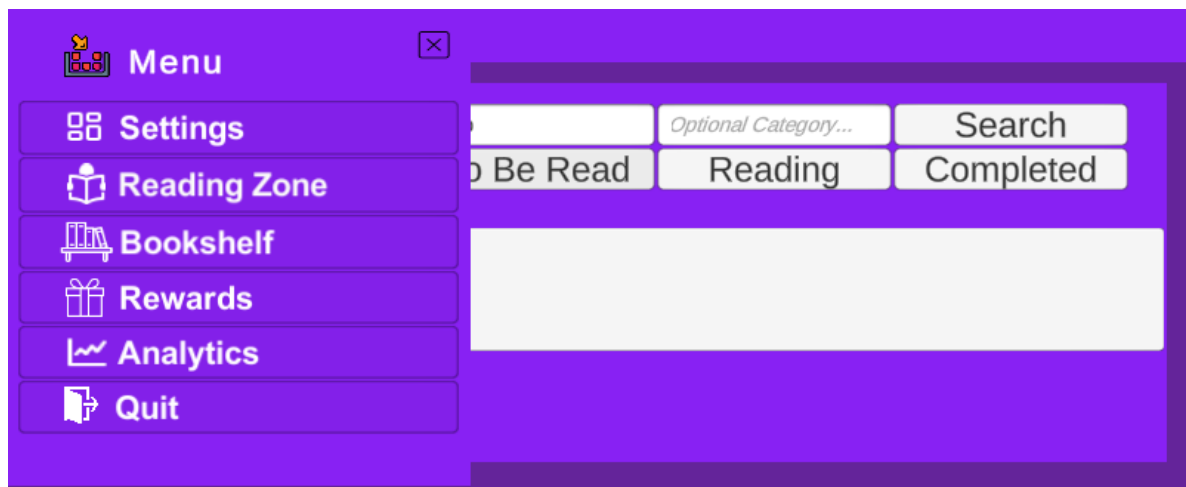




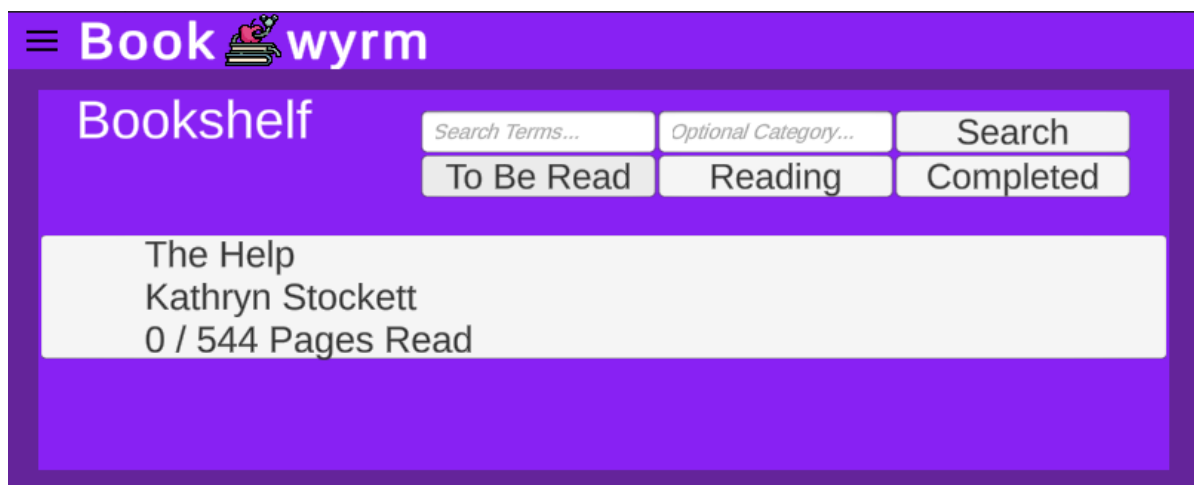
Error code as seen in the log file.

```
System.Runtime.Serialization.SerializationException: Attempting to deserialize an empty stream.  
  at System.Runtime.Serialization.Formatters.Binary.BinaryFormatter.Deserialize  
(System.IO.Stream serializationStream,  
System.Runtime.Remoting.Messaging.HeaderHandler handler, System.Boolean fCheck,  
System.Boolean isCrossAppDomain,  
System.Runtime.Remoting.Messaging.IMethodCallMessage methodCallMessage) [0x0003c]  
in <9577ac7a62ef43179789031239ba8798>:0
```

Program Functional after error. Added a book.



Added book still exists after relaunch (save file works).



#### 4.2.2 Problems encountered.

There were no software problems discovered during corrupt save data tests.

#### 4.2.3 Deviations from test cases/procedures.

There were no deviations from procedures during corrupt save data tests.

### 4.3 BW\_T\_AT tests

Application termination tests crash the application suddenly to ensure data persistence is maintained. This phase should consist of rebooting the application multiple times in order to determine possible crash events. Analyzing those events would be then the next course of action to fix the related or identified issues.

#### 4.3.1 Summary of test results.

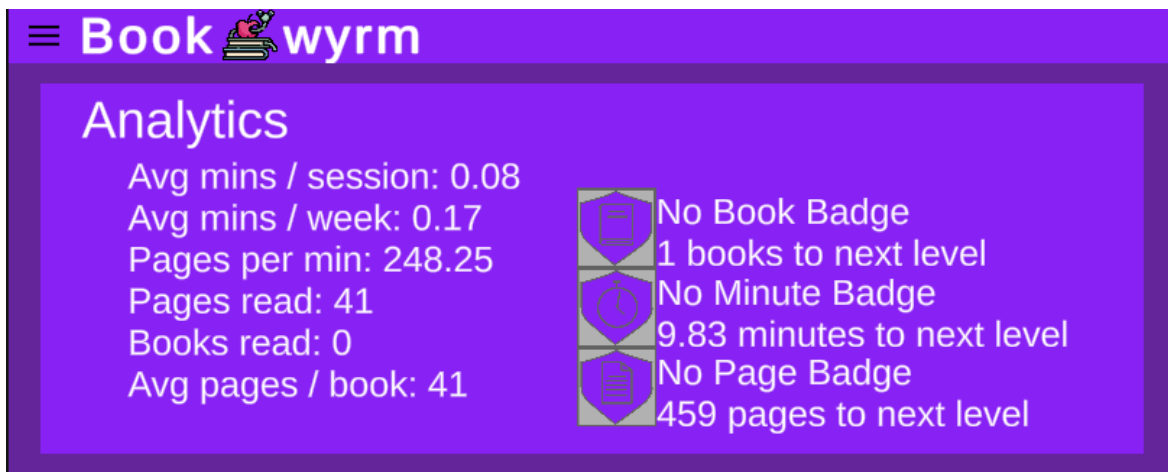
Requirement	Description	Test ID	Status
BW_F_1	The software shall store a list of books to be read and ones planned to be read which remain persistent between launches.	BW_T_AT_1	all results as expected
BW_F_2	The software shall store recorded statistics which shall be persistent between launches. It may recalculate statistical values.	BW_T_AT_1	all results as expected
BW_F_3	The software shall store data on books in progress and display the data. The data will persist through launches.	BW_T_AT_1	all results as expected
BW_F_4	The software shall store the list of books completed which shall be persistent between launches (different program instances instances from the same file location)	BW_T_AT_1	all results as expected
BW_E_1	The software shall not corrupt persistent data in the event of an internal error.	BW_T_AT_1	all results as expected
BW_E_3	The software shall log any internal error it encounters, including error type to a local log file.	BW_T_AT_1	all results as expected

#### 4.3.1.1 BW\_T\_AT\_1 Details

Process: A functional save data file was generated by using the program as intended. The program was then recompiled with a line to crash the program during a save. The program is then recompiled without the line, relaunched, and save data is checked to see if it is loadable or not. It should not crash the program.

Results: The list of books, data on books, and statistics were still loadable after a crash during the saving procedure. Data was not corrupted and no internal errors were encountered to be logged. (The internal error we created was not expected to be logged as it did not involve data from the program itself).

Statistics after relaunching after the crash.



The screenshot shows the 'Bookworm' application interface. At the top is a purple header with a hamburger menu icon, the text 'Bookworm', and a small icon of a book with a red apple. Below the header is a light blue box titled 'Analytics'. Inside this box, on the left, are several statistics: 'Avg mins / session: 0.08', 'Avg mins / week: 0.17', 'Pages per min: 248.25', 'Pages read: 41', 'Books read: 0', and 'Avg pages / book: 41'. On the right side of the analytics box are three rows of status information, each with a small icon in a blue square: 'No Book Badge' (book icon), '1 books to next level'; 'No Minute Badge' (clock icon), '9.83 minutes to next level'; and 'No Page Badge' (document icon), '459 pages to next level'.

Bookworm

### Analytics

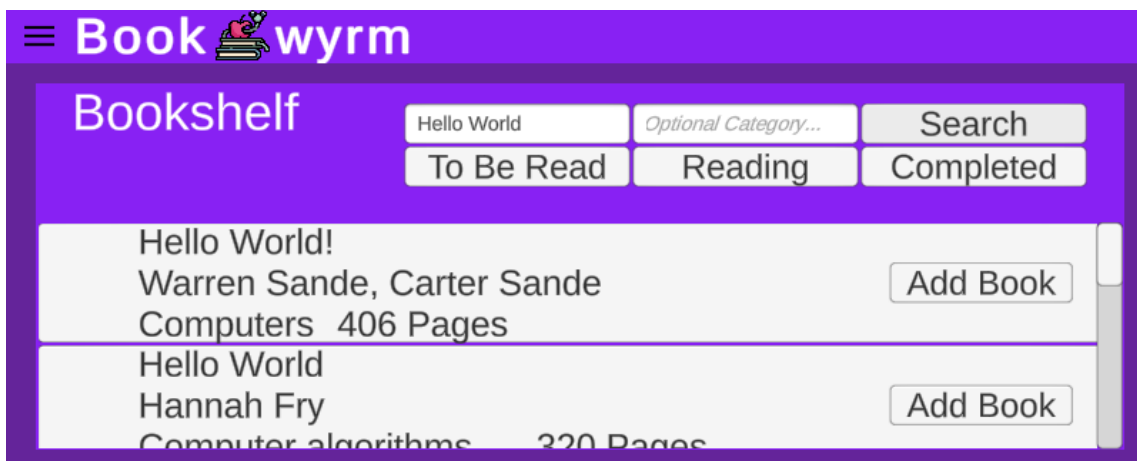
Avg mins / session: 0.08  
Avg mins / week: 0.17  
Pages per min: 248.25  
Pages read: 41  
Books read: 0  
Avg pages / book: 41

No Book Badge  
1 books to next level

No Minute Badge  
9.83 minutes to next level

No Page Badge  
459 pages to next level

To Be Read list prior to crash.



The screenshot shows the 'Bookworm' application interface. At the top is a purple header with a hamburger menu icon, the text 'Bookworm', and a small icon of a book with a red apple. Below the header is a light blue box titled 'Bookshelf'. At the top of the bookshelf are three input fields: 'Hello World', 'Optional Category...', and 'Search'. Below these are three buttons: 'To Be Read', 'Reading', and 'Completed'. The main area of the bookshelf is a list of books. The first book is 'Hello World! Warren Sande, Carter Sande Computers 406 Pages' with an 'Add Book' button to its right. The second book is 'Hello World Hannah Fry Computer algorithms 320 Pages' with an 'Add Book' button to its right. A vertical scrollbar is visible on the right side of the book list.

Bookworm

### Bookshelf

Hello World Optional Category... Search

To Be Read Reading Completed

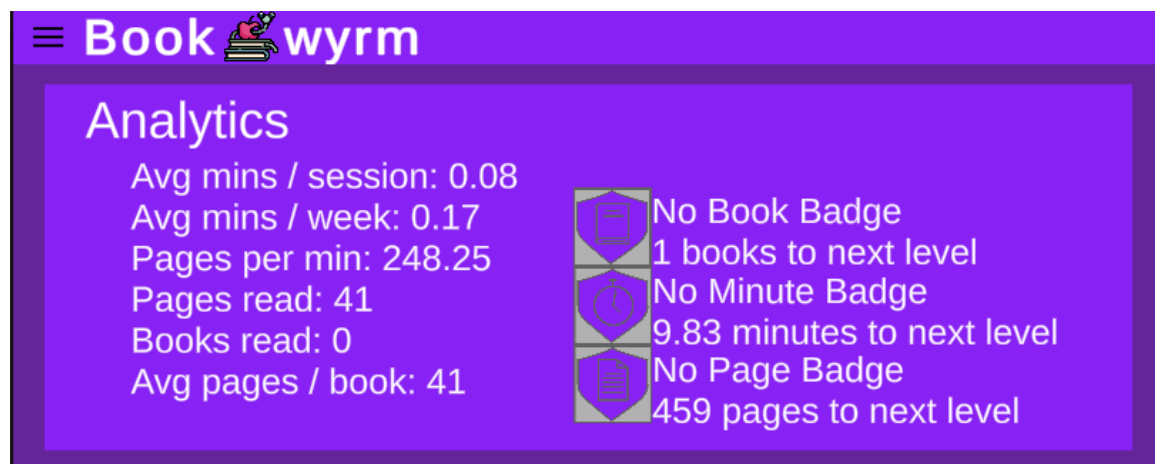
Hello World!  
Warren Sande, Carter Sande  
Computers 406 Pages  
Add Book

Hello World  
Hannah Fry  
Computer algorithms 320 Pages  
Add Book

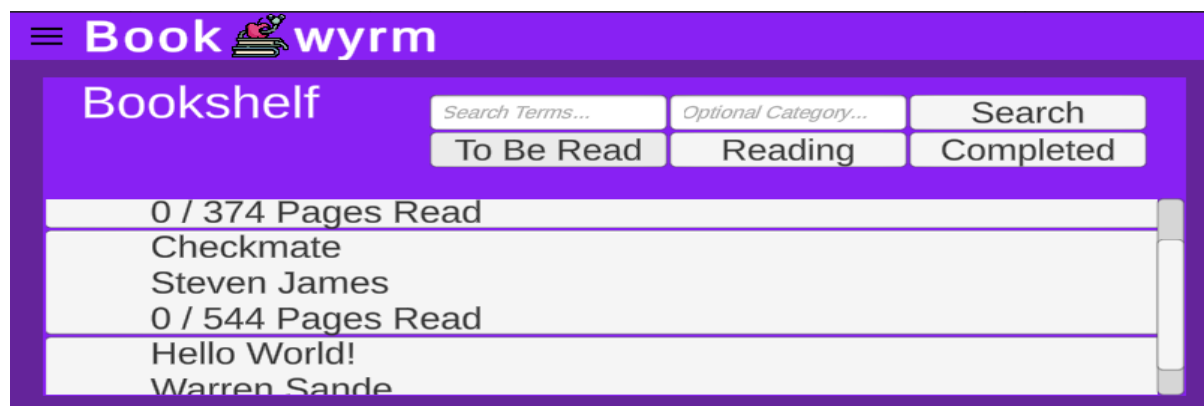
Adding a book (which causes a save event, thus a system freeze).



Statistics after relaunching after the crash.



To Be Read list after relaunching after the crash (it includes the book added as the software crashed).



### 4.3.2 Problems encountered.

There were no software problems discovered during application termination tests.

### 4.3.3 Deviations from test cases/procedures.

There were no deviations from procedures during application termination tests.

## 4.4 BW\_T\_SD tests

Software demonstration tests involve the user running the program and using features to confirm their function. They run at the CSCI level and are demonstration type tests. They are manual tests completed by a developer. Recording results should be done through screenshots of the program's state before and after the test.

### 4.4.1 Summary of test results.

Requirement	Description	Test ID	Status
BW_F_1	The software shall store a list of books planned to be read which remain persistent between launches.	BW_T_SD_1	all results as expected
BW_F_2	The software shall store recorded statistics which shall be persistent between launches. It may recalculate statistical values.	BW_T_SD_1	all results as expected
BW_F_3	The software shall store data on books in progress and display the data. The data will persist through launches.	BW_T_SD_1	all results as expected
BW_F_4	The software shall store the list of books completed which shall be persistent between launches (different program instances instances from the same file location)	BW_T_SD_1	all results as expected
BW_F_6	The software shall use user data in conjunction with the remote information source to acquire information on books to be added.	BW_T_SD_1	all results as expected
BW_F_7	The software shall sort book search results by category, and allow the user to search amongst individual categories.	BW_T_SD_1	all results as expected
BW_P_1	The application shall include data about the amount of pages read and other data that would help create statistics related to the user's reading habits.	BW_T_SD_1	all results as expected

BW_P_2	The application shall include an automated reward system or algorithm that functions with the data (input) fed to the software and based on such information appropriate rewards should be given to the user.	BW_T_SD_1	all results as expected
BW_C_1	The BW application shall initialize to the home state when loaded.	BW_T_SD_1	all results as expected
BW_C_2	The BW application shall have the option to reset recorded data	BW_T_SD_1	all results as expected

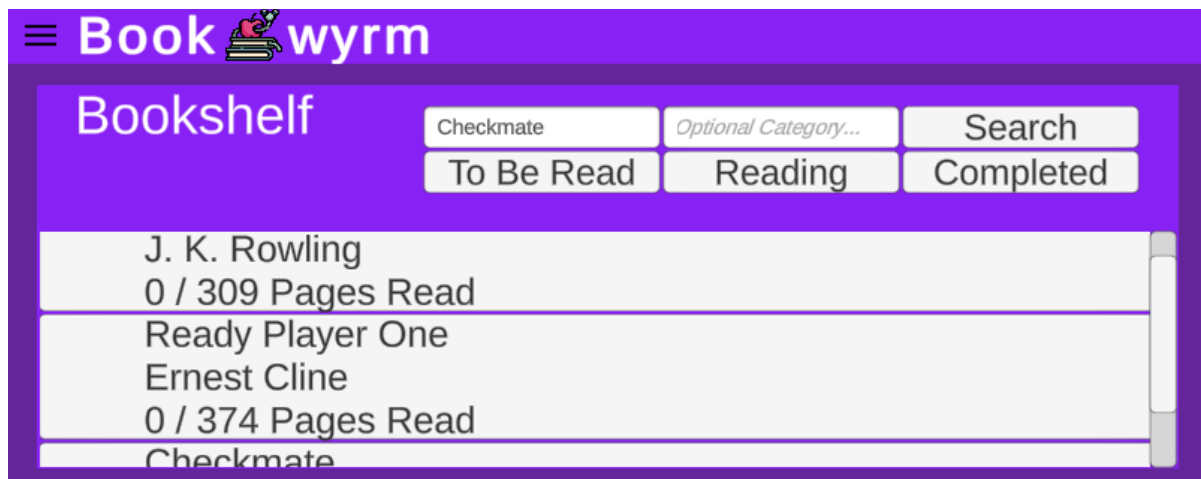
#### 4.2.1.1 BW\_T\_SD\_1 Details

Process: Software demonstration tests involve taking screenshots of features described to show that they function.

Results: See screenshots for each feature as demonstrated below.


BW\_F\_1:

List of books to be read: (same after relaunch)



BW\_F\_2:

Statistics page (same after relaunch).

Book wurm

## Analytics

Avg mins / session: 0.03


Avg mins / week: 0.06

Pages per min: 118.01

Pages read: 7


Books read: 0

Avg pages / book: 7




No Book Badge

1 books to next level



No Minute Badge

9.94 minutes to next level




No Page Badge

493 pages to next level

BW\_F\_3:

List of books currently reading (same after relaunch).

Book wurm

## Bookshelf

Search Terms...

Optional Category...

Search

To Be Read

Reading

Completed


Harry Potter and the Sorcerer's Stone

J. K. Rowling

7 / 309 Pages Read

BW\_F\_4:

List of books completed: (same after relaunch)

Book wurm

## Bookshelf

Train Song

Optional Category...

Search

To Be Read

Reading

Completed

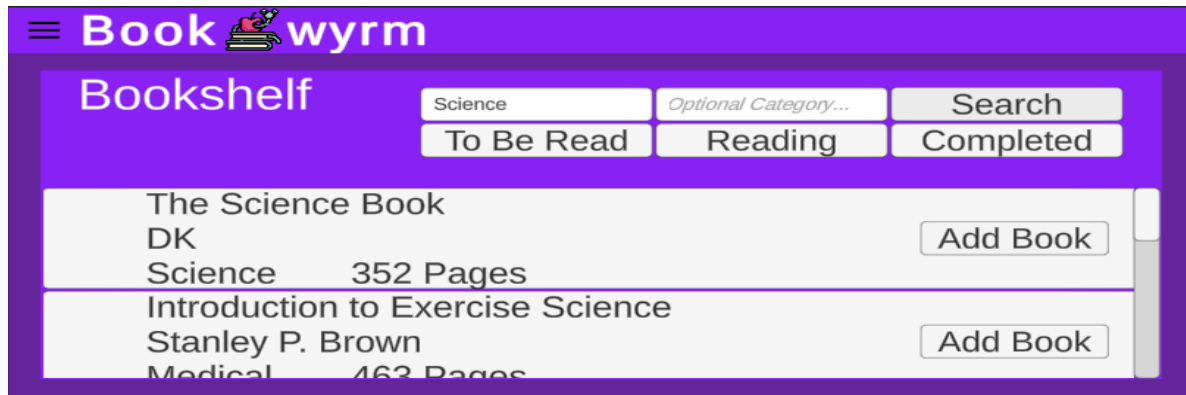
Train Song

Diane Siebert

32 / 32 Pages Read

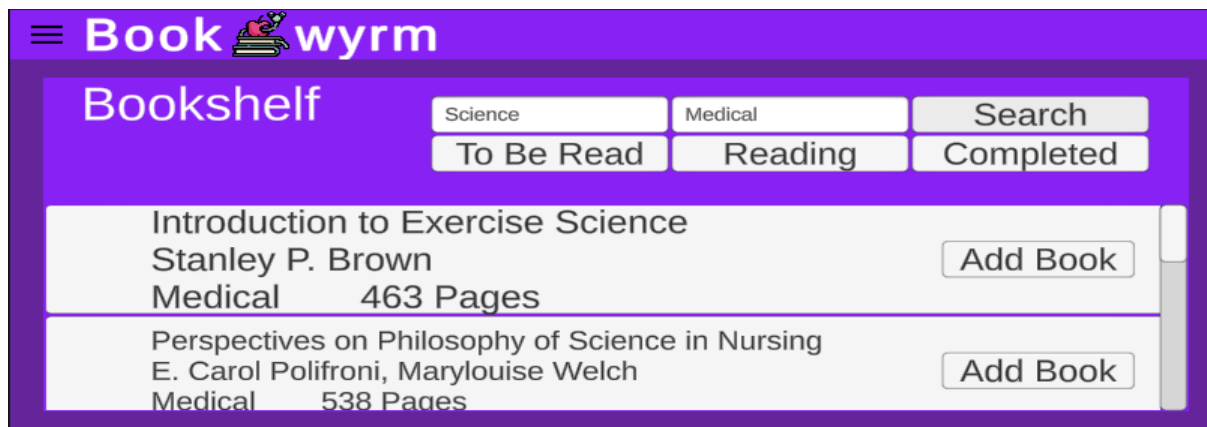
BW\_F\_6:

Results found online through the Google Books API when someone searches "science".



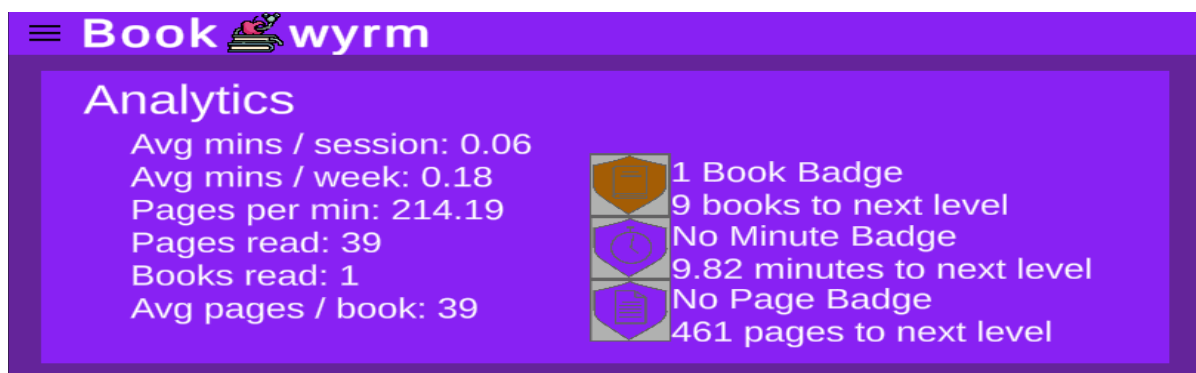
BW\_F\_7:

Results found online through the Google Books API when someone searches "science" in the category "Medical"..



BW\_P\_1:

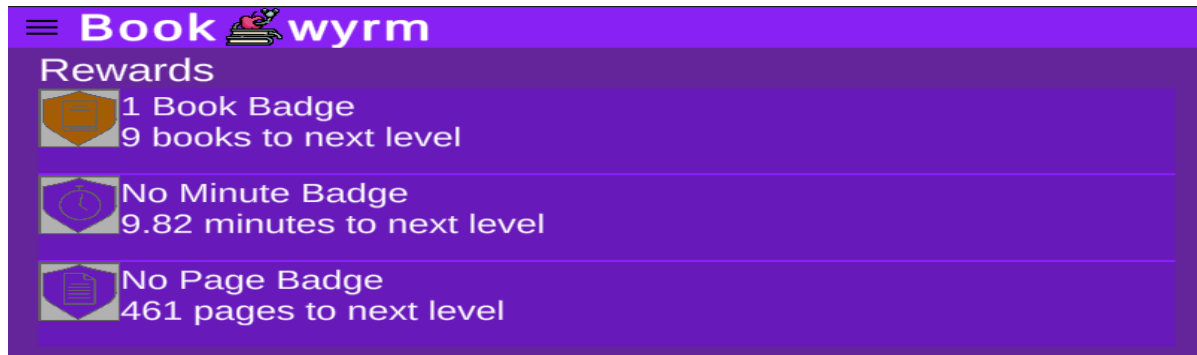
Pages read and other data stored by the program to calculate statistics as seen on the analytics page.



BW\_P\_2:

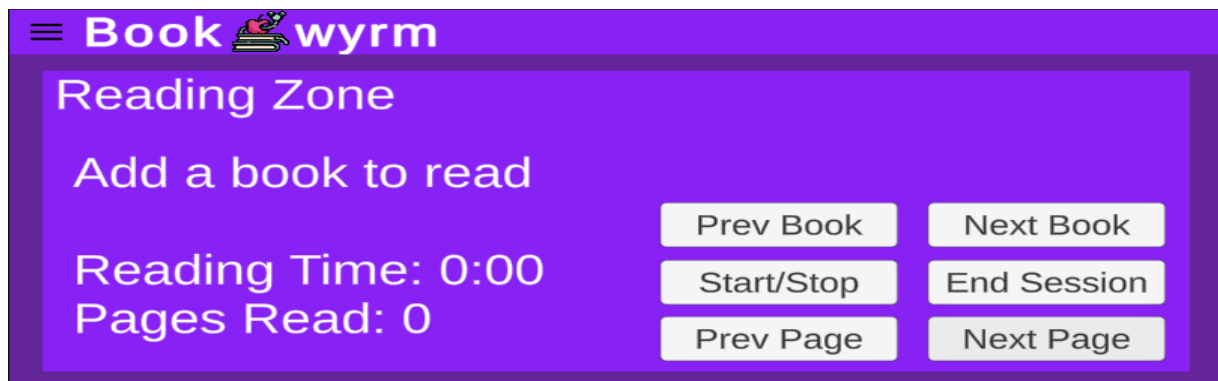
Badges based on the statistics data stored (also can be seen on the analytics page).





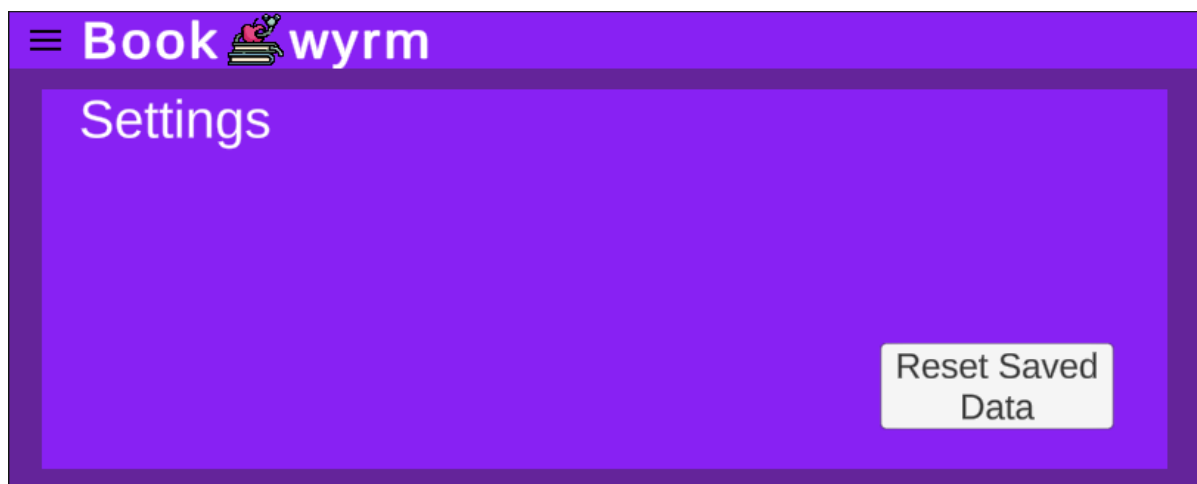
BW\_C\_1:

The state of the program immediately after launching. For this application, the reading zone in the home state, it always launches to this state on startup.



BW\_C\_2:

The button to reset save data.



The analytics page after a data reset.

## Analytics

Avg mins / session: 0.00

Avg mins / week: 0.00

Pages per min: 0.00

Pages read: 0

Books read: 0

Avg pages / book: 0



No Book Badge

1 books to next level



No Minute Badge

10.00 minutes to next level



No Page Badge

500 pages to next level

### 4.4.2 Problems encountered.

There were no software problems discovered during software demonstration tests.

### 4.4.3 Deviations from test cases/procedures.

There were no deviations from procedures during software demonstration tests.

## 4.5 BW\_T\_DAR tests

Database availability/reliability tests are tests that involve running the program while attempting to query the remote information source without connection to it. They also involve sending fake or bad input to the program into the channels where the program expects data from the server.

### 4.5.1 Summary of test results.

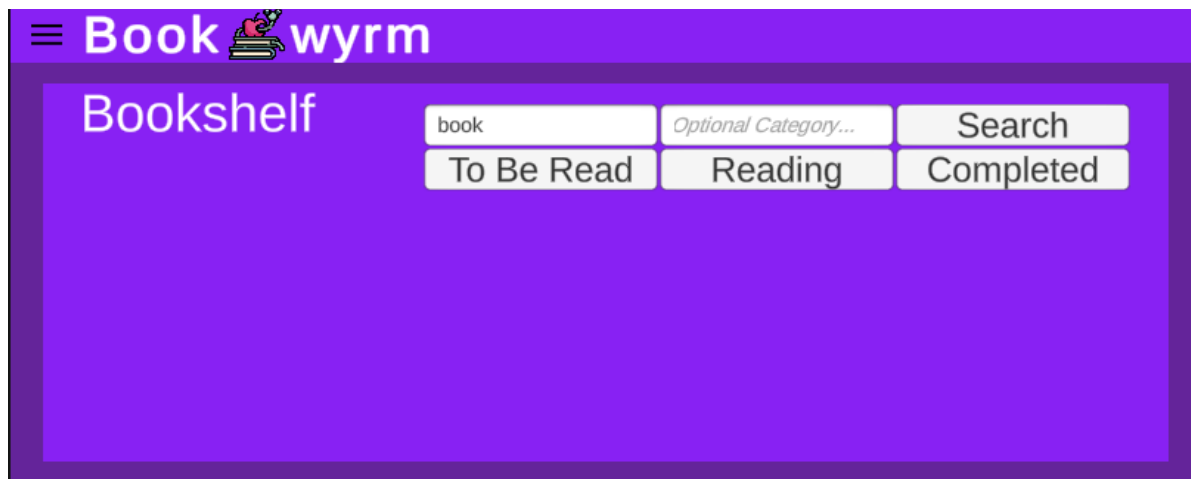
Requirement	Description	Test ID	Status
BW_F_5	The software shall type-check input and re-prompt the user if the type is wrong or if the data could not be interpreted.	BW_T_DAR_2	all results as expected
BW_E_2	The software shall not crash due to the connection with the book information database being unsuccessful or the database returning invalid data.	BW_T_DAR_1	all results as expected

#### 4.5.1.1 BW\_T\_DAR\_1 Details

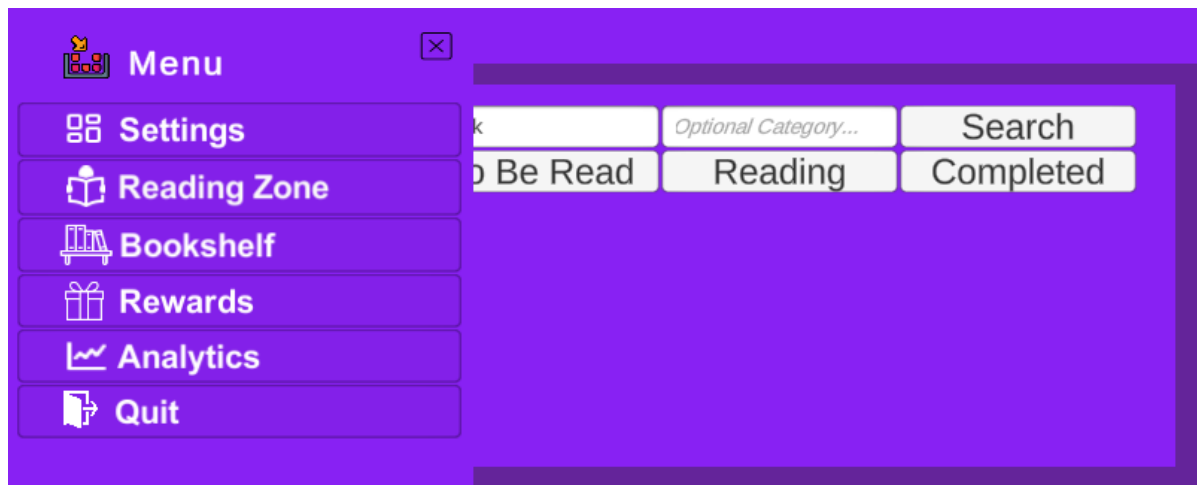
Process: Run the program and attempt to query the remote information source without connection to it. Make sure the program handles the case without crashing.

Results: The program caught a http exception from a get request to a nonexistent source, and put the error data in the log file as expected. The program did not crash from this error and kept operating effectively.

Search performed without network access.



Program still responsive after search.



Error log from local log file:

```
WebException: Error: NameResolutionFailure
  at System.Net.HttpWebRequest.EndGetResponse (System.IAsyncResult asyncResult)
[0x00058] in <ef151b6abb5d474cb2c1cb8906a8b5a4>:0
  at System.Threading.Tasks.TaskFactory`1[TResult].FromAsyncCoreLogic
(System.IAsyncResult iar, System.Func`2[T,TResult] endFunction, System.Action`1[T]
endAction, System.Threading.Tasks.Task`1[TResult] promise, System.Boolean
requiresSynchronization) [0x0000f] in <9577ac7a62ef43179789031239ba8798>:0
```

#### 4.5.1.1 BW\_T\_DAR\_2 Details

Process: Run the program and attempt to query the remote information source for books with known strange values. Make sure the program handles it. If an error is encountered check the error log for expected output.

Results: Books were found with missing page numbers and strange or missing category values. None of these results caused internal errors, and though the output looks strange, a slick UI was not a requirement for this project. They were also able to be added successfully to the To Be Read list.

Results of books with missing pages and categories.



Those books added to the To Be Read list.

 Bookworm

Bookshelf

Illustrated Collection

Optional Category...

Search

To Be Read


Reading

Completed

Harry Potter  
J. K. Rowling  
0 / 0 Pages Read

Harry Potter - The Illustrated Collection  
J. K. Rowling  
0 / 0 Pages Read

One of the books with missing pages in the Reading Zone.

 Bookworm

Reading Zone

Harry Potter - The Illustrated Collection

Reading Time: 0:05  
Pages Read: 0 / 0

Prev Book

Next Book


Start/Stop

End Session

Prev Page

Next Page

Search results with missing and strange categories.

 Bookworm

Bookshelf

Book

\*

Search

To Be Read

Reading

Completed

Elmer Hewitt Capen  
242 Pages

The Organ and Its Position in Musical Art  
Henry Heathcote Statham  
\* 244 Pages

Mozart

Add Book

Add Book

Search results with missing and strange categories. (Cont.)

The screenshot shows the Bookwurm Bookshelf interface. At the top, there is a header with a hamburger menu icon, the text 'Bookwurm', and a small icon of a book with a red bookmark. Below the header, the word 'Bookshelf' is displayed. To the right of 'Bookshelf' are three input fields: 'Book', '\*', and 'Search'. Below these fields are three buttons: 'To Be Read', 'Reading', and 'Completed'. The main content area displays two book entries. The first entry is 'Charles O'Connell' with a '\*' symbol and '556 Pages', followed by an 'Add Book' button. The second entry is 'Encyclopaedic Dictionary of Buddhism' by 'Samir Nath' with a '.' symbol and '801 Pages', followed by an 'Add Book' button. A third entry, 'Watteau', is partially visible at the bottom.

Book	Category	Action
Charles O'Connell	*	Add Book
Encyclopaedic Dictionary of Buddhism	.	Add Book
Watteau		

Those books added to the To Be Read list.

The screenshot shows the Bookwurm Bookshelf interface with the 'To Be Read' button selected. The main content area displays three book entries. The first entry is 'Samir Nath' with '0 / 801 Pages Read'. The second entry is 'The Victor Book of Symphonies' by 'Charles O'Connell' with '0 / 556 Pages Read'. The third entry is 'Occasional Addresses'.

Book	Category	Progress
Samir Nath		0 / 801 Pages Read
The Victor Book of Symphonies		0 / 556 Pages Read
Occasional Addresses		

#### 4.5.2 Problems encountered.

There were no software problems discovered during database availability/reliability tests.

#### 4.5.3 Deviations from test cases/procedures.

There were no deviations from procedures during database availability/reliability tests.

## 5. Test log.

Chronological record of the test events covered by this report.

1. BW\_T\_EUI tests
  - a. Date and Location: December 9, 2020 at Windows 10 test site
  - b. Software
    - i. Unity 2019.4.13f1 Personal <DX11>
    - ii. Windows 10 Pro, 10.0.18363 Build 18363
    - iii. BookWyrms Github commit c3b2bd2
  - c. Hardware
    - i. 16 GB RAM
    - ii. Intel i5-6600K
    - iii. NVidia GTX 1050TI
  - d. Test performed at 3:20pm EST by Peter Stein
  - e. Test (with deviation) performed at 3:55pm EST by Peter Stein
2. BW\_T\_CSD tests
  - a. Date and Location: December 9, 2020 at Windows 10 test site
  - b. Software
    - i. Unity 2019.4.13f1 Personal <DX11>
    - ii. Windows 10 Pro, 10.0.18363 Build 18363
    - iii. BookWyrms Github commit c3b2bd2
  - c. Hardware
    - i. 16 GB RAM
    - ii. Intel i5-6600K
    - iii. NVidia GTX 1050TI
  - d. Test performed at 7:50pm EST by Peter Stein
3. BW\_T\_AT tests
  - a. Date and Location: December 9, 2020 at Windows 10 test site
  - b. Software
    - i. Unity 2019.4.13f1 Personal <DX11>
    - ii. Windows 10 Pro, 10.0.18363 Build 18363
    - iii. BookWyrms Github commit c3b2bd2
  - c. Hardware
    - i. 16 GB RAM
    - ii. Intel i5-6600K
    - iii. NVidia GTX 1050TI
  - d. Test performed at 5:10pm EST by Peter Stein
4. BW\_T\_SD tests
  - a. Date and Location: December 9, 2020 at Windows 10 test site
  - b. Software
    - i. Unity 2019.4.13f1 Personal <DX11>
    - ii. Windows 10 Pro, 10.0.18363 Build 18363

- iii. BookWyrms Github commit c3b2bd2
  - c. Hardware
    - i. 16 GB RAM
    - ii. Intel i5-6600K
    - iii. NVidia GTX 1050TI
  - d. Test performed at 8:35pm EST by Peter Stein
- 5. BW\_T\_DAR tests
  - a. Date and Location: December 9, 2020 at Windows 10 test site
  - b. Software
    - i. Unity 2019.4.13f1 Personal <DX11>
    - ii. Windows 10 Pro, 10.0.18363 Build 18363
    - iii. BookWyrms Github commit c3b2bd2
  - c. Hardware
    - i. 16 GB RAM
    - ii. Intel i5-6600K
    - iii. NVidia GTX 1050TI
  - d. Test performed at 6:29pm EST by Peter Stein

## 6. Notes.

Reference the Software Requirement Specification and the Software Test Plan for more detail and rationale behind all the test activities.

4.x.1.y (Project unique identifier) Details sections added to template for clarity

Glossary of document acronyms:

- API - Application Programming Interface
- DCMA - Defense Contract Management Agency
- CSCI - Computer Software Configuration Item
- HWCI - Hardware Configuration Item
- RAM - Random Access Memory
- SDD - Software Design Description
- SDP - Software Development Plan
- SRS - Software Requirements Specification
- STD - Software Test Description
- STR - Software Test Report