Software Test Plan (STP)
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: 11/12//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: Software test environment
- Section 4: Test identification
- Section 5: Test schedules
- 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, Software Requirements Specification, and Software Development Document 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. Software test environment.

This section shall be divided into the following paragraphs to describe the software test environment at each intended test site. Reference may be made to the Software Development Plan (SDP) for resources that are described there.

3.Platform Testing (PC/Mac/iOS/Android).

Our application does not require a dedicated test location. Each member of the team will test the application on their own personal device. Operating systems and hardware may vary from device to device. Sites will include a windows PC, a Mac, an Android device, and an iOS device.

3.1 Software items.

Testing the data and statistics produced will be done through a testing file. The software controls, output, reward systems, and layout, will all be tested within the application itself.

<u>Operating System testing environments:</u>

iOS 14 (any revision should be acceptable), Android KitKat (any revision should be acceptable), Windows 10 (any revision should be acceptable), macOS Catalina (any revision should be acceptable)

<u>Compilers:</u>

Created and compiled with Unity (version 2019.4.11f1)

<u>Databases:</u>

For pulling book information and cover images we call the Library of Congress - Data exploration API, listed on github in the following link: https://github.com/LibraryOfCongress/data-exploration/blob/master/LOC.gov%20JSON%20API.ipynb

Same service through google as a backup in case book is missing from LoC database, listed on google API library in the following link: https://developers.google.com/books/docs/overview

3.2 Hardware and firmware items.

The hardware requirements are minimal as the application is designed to run on both mobile and stationary systems. The only true technical requirements would be fully updated and functioning devices running Windows 10, macOS Catalina, Android KitKat, or iOS 14 .

3.3 Other materials.

We may consider creating invalid saved data for testing to ensure the application can handle corrupt/invalid data. Otherwise, no additional material is necessary for testing.

3.4 Proprietary nature, acquirer's rights, and licensing.

I. **Unity Licensing**:

The team decided on utilizing a free student Unity license or package, which terms are as follow according to the Unity website:

- Financial eligibility: Eligible if revenue or funding is less than $100K in the last 12 months
- Student plan eligibility: Students enrolled in an accredited educational institution of legal age to consent to the collection and processing of their personal information, e.g., age 13 in the US, 16 in the EU. Must join the GitHub Student Developer Pack to be verified.
- Monetization eligibility: The licence or package includes Unity Ads and In-App Purchase plugin as ways to monetize on a created application.

Note: Please, be advanced to fully read the licensing terms and conditions that Unity offers for the above package in further details on the official Unity website.

II. **API Licensing Terms**:

To have access to various book assets, the team decided on abiding the terms of service that Google APIs provide and any other APIs if needed. In general, these licenses follow the below categories according to the Google APIs Terms of service:

- Section 1: Account and Registration
- Section 2: Using our APIs
- Section 3: Your API clients
- Section 4: Prohibitions and Confidentiality
- Section 5: Content
- Section 6: Brand Features, Attribution
- Section 7: Privacy and Copyright Protection
- Section 8: Termination
- Section 9: Liability for our APIs
- Section 10: General Provisions.

Note: Please, be advance to expand on those sections in particular for the Google APIs Terms of Service at the following URL address

https://developers.google.com/terms/#section_10_general_provisions

III. GitHub Licensing

To be able to share codes and application assets, the team decided on a free GitHub student license, which section to abide are as follows according to the official GitHub website:

- Definitions
- Account Terms
- Acceptable Use
- User-Generated Content
- Private Repositories
- Copyright Infringement and DMCA Policy
- Intellectual Property Notice
- API Terms
- GitHub Additional Product Terms
- Beta Previews
- Payment
- Cancellation and Termination
- Communications with GitHub
- Disclaimer of Warranties
- Limitation of Liability
- Release and Indemnification
- Changes to These Terms
- Miscellaneous

Note: Please be advised to read the above section in details at the following URL:

https://docs.github.com/en/free-pro-team@latest/github/site-policy/github-terms-of-service

IV. Open Codes

The team agrees to give credit to open codes on the internet and abide on their terms of services. However, to the extent a protection to ourselves and prior proprietary agents. The Bookwyrm development team chose to adjust any codes that a proprietary agent deems unvalid for utilization. Accords to not use their codes shall be explored as to not in any case burden the Bookwyrm development team or assets.

3.5 Installation, testing, and control.

The drivers for the software test environment for input testing will have to be developed, but much of the testing is done through demonstration which only requires Bookwyrm to be installed. These will be hosted in a shared git repository to be cloned onto testing devices. This repository can also be used to share and store the output of the tests.

3.6 Participating organizations.

No external organization will be assisting with the testing. The only testers will be the development group.

3.7 Personnel.

The members of the development group will begin thoroughly testing the application on agreed upon dates and times as seen in section 5. Testing does not require any specific skills, although we must ensure that all members are aware of the expected functionality and output of the application. The length of testing will be as long as needed for the tests to pass. An estimate is given in the table in section 5.

3.8 Orientation plan.

Prior to thoroughly testing the application, the group should be fully aware of what functionality and output is expected for each section of the application. Members need to know exactly how the application functions so they will know when the application is not working as expected. No additional training is required.

3.9 Tests to be performed.

Since the different sites correspond to different platforms for the program to be run on, and since the program will have the same functionality between platforms, the full set of tests in section 4 will be run individually in each test site.

4. Test identification.

This section shall be divided into the following paragraphs to identify and describe each test to which this STP applies.

4.1 General information.

This paragraph shall be divided into subparagraphs to present general information applicable to the overall testing to be performed.

4.1.1 Test levels.

The Bookwyrm program is being developed at the CSCI level, being a software component instead of a system, and as such, it will mostly be tested at the CSCI level. Some tests may be developed at the CSC level, unit testing the specific software components as described in the SDD. More specifically, all demonstration method tests will be performed on the CSCI level, testing the whole system for proper functionality. Input and type checking tests will be performed on the unit level earlier in development as part of integration testing.

4.1.2 Test classes.

The test classes which will be performed include:

- Erroneous user input tests to ensure that users cannot crash the application or corrupt data through bad input
- Corrupt save data tests to ensure the application is capable of handling and responding to invalid saved data
- Application termination tests that crash the application suddenly to ensure data persistence maintains
- Demonstration tests that involve the user running the program and using features to confirm their function.
- Database availability/reliability test to ensure that the application can handle an unsuccessful connection to the database or invalid data received from the database

4.1.3 General test conditions.

4.1.3.1 Test coverage

The complete set of tests shall check maximum, minimum, nominal, wrong data types, and out of range values for all user input sections. Users are the biggest source of malformed input to this program, so by checking these four types of values, most edge cases will be checked, while still running relatively few tests to save time.

4.1.3.2 Retesting approach

The program must pass all tests. If a test fails, the program will be required to retest that test, and any other tests for the same software component (if it is being tested at the component level). This is to attempt to catch new bugs caused by an attempted fix on a particular failed test.

4.1.4 Test progression.

This set of tests has no particular order, except that any planned integration testing shall be completed before the software is tested on the CSCI level.

4.1.5 Data recording, reduction, and analysis.

Any demonstration-method test shall be recorded through screenshots showing the specific steps and output for the feature being tested. This is a manual process. Input tests shall record input that is sent and output received, along with expected output. This can be done by printing or writing into a document or through screen capture. This can be a manual or automatic process. The data can be saved in a shared repository.

4.2 Planned tests.

This paragraph shall be divided into the following subparagraphs to describe the total scope of the planned testing.

4.2 Item(s) to be tested

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.2.4 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.

4.2.5 BW_T_DAR tests

Database availability/reliability tests are tests that involve sending fake or bad input to the program into the channels where the program expects data from the server. This runs at the software component level and should be run automatically through a driver program, which can also log the input and output.

5. Test schedules.

Purpose	Site(s)	11/13 - 11/20	11/21	11/22	11/23-11/24
Developing Tests	PC, Mac, iOS, Android	X	X		
Conducting Preliminary Test Run (including integration testing)	PC, Mac, iOS, Android		X	X	
Analysis of Output	PC, Mac, iOS, Android		X	X	X
Conducting Final Test Run	Any site that failed preliminary test run			X	X
Compiling results into Software Test Report	(none)				X

6. Requirements traceability.

Requirement	Description	Method	Tested by
BW_F_1	The software shall store the list of books to be read between launches (different program instances from the same file location)	Demonstration	BW_T_SD BW_T_AT
BW_F_2	The software shall store recorded statistics between launches (different program instances from the same file location). It may recalculate values	Demonstration	BW_T_SD BW_T_AT
BW_F_3	The software shall store and display data on books in progress	Demonstration	BW_T_SD BW_T_AT
BW_F_4	The software shall store the list of books completed between launches (different program instances instances from the same file location)	Demonstration	BW_T_SD BW_T_AT
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.	Testing	BW_T_EUI BW_T_DAR
BW_F_6	The software shall use user data in conjunction with the remote information source to acquire information on books to be added.	Demonstration Testing	BW_T_SD BW_T_EUI
BW_F_7	The software shall allow and remember display notes on a page by page basis inputted by the user	Demonstration	BW_T_SD
BW_P_1	The BW application shall include Alert times or reminders to motivate users.	Demonstration	BW_T_SD
BW_P_2	The application shall include Books read and planned to be read checklists to keep track of user progress.	Demonstration	BW_T_SD
BW_P_3	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.	Demonstration	BW_T_SD
BW_P_4	The application shall include an automated reward system or algorithm that functions with the data (input) fed to the software and	Demonstration	BW_T_SD

	based on such information appropriate rewards should be given to the user.		
BW_P_5	The application shall run on a system of the minimum requirements found in section 2.4 with a minimum frame rate of 30 frames per second when not loading data from disk.	Demonstration	BW_T_SD
BW_C_1	The BW application shall initialize to the home state when loaded.	Demonstration	BW_T_SD
BW_C_2	The BW application shall have setting parameters, including the option to reset recorded data	Demonstration Testing	BW_T_SD BW_T_EUI
BW_E_1	The software shall not corrupt saved data in the event of an internal error.	Testing	BW_T_CSD BW_T_AT
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.	Testing	BW_T_CSD BW_T_DAR
BW_E_3	The software shall log any internal error it encounters, including error type to a local log file.	Testing	BW_T_AT BW_T_CSD BW_T_EUI
BW_E_4	The software shall not crash the user's device upon encountering an error.	Testing	BW_T_AT BW_T_CSD BW_T_EUI

7. Notes.

Glossary of document acronyms:

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