Mission: Focus/Concentration Reading Application

DID: Software Requirements Specification (SRS)

Due Date: 10/15/2020

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

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.

Software title: Book Wyrm

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 (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.

1. Scope
    1.1. Identification
    1.2. System overview
    1.3. Document overview
    1.4. Relationship to other plans
2. Overall Description
    2.1. Product perspective
    2.2. Product functions
    2.3. User characteristics
    2.4. Constraints
    2.5. Assumptions and dependencies
3. Specific requirements
    3.1. External interface requirements
    3.2. Functional requirements
    3.3. Performance requirements
    3.4. Configuration requirements
    3.5. Security requirements
    3.6. Error handling requirements
    3.7. Software system attributes
    3.8. Qualification methods per requirement

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

1.4 Relationship to other plans.

No other plans for this project other than general outlines have been developed at the time of this document. Research on how to approach creating an app on Unity and a decision to use Unity over C# have been determined.

2. Overall description.

2.1 Product perspective.

The overall goal of the program is to help the user to read more, and to do this, it fills a specific role. The program is set up as a reading tracker and planner, to allow the user to look for new books they want to read, list them, and track them as they are read. As books are read, the program will provide the user with some positive feedback (like a badge), and the user can check their progress and statistics about their reading.

The program will be self contained aside from access to a database from which to pull book information. The lists it uses should be internal, as well as the calculation of statistics, to allow offline functionality of the statistics and tracking part of the program.

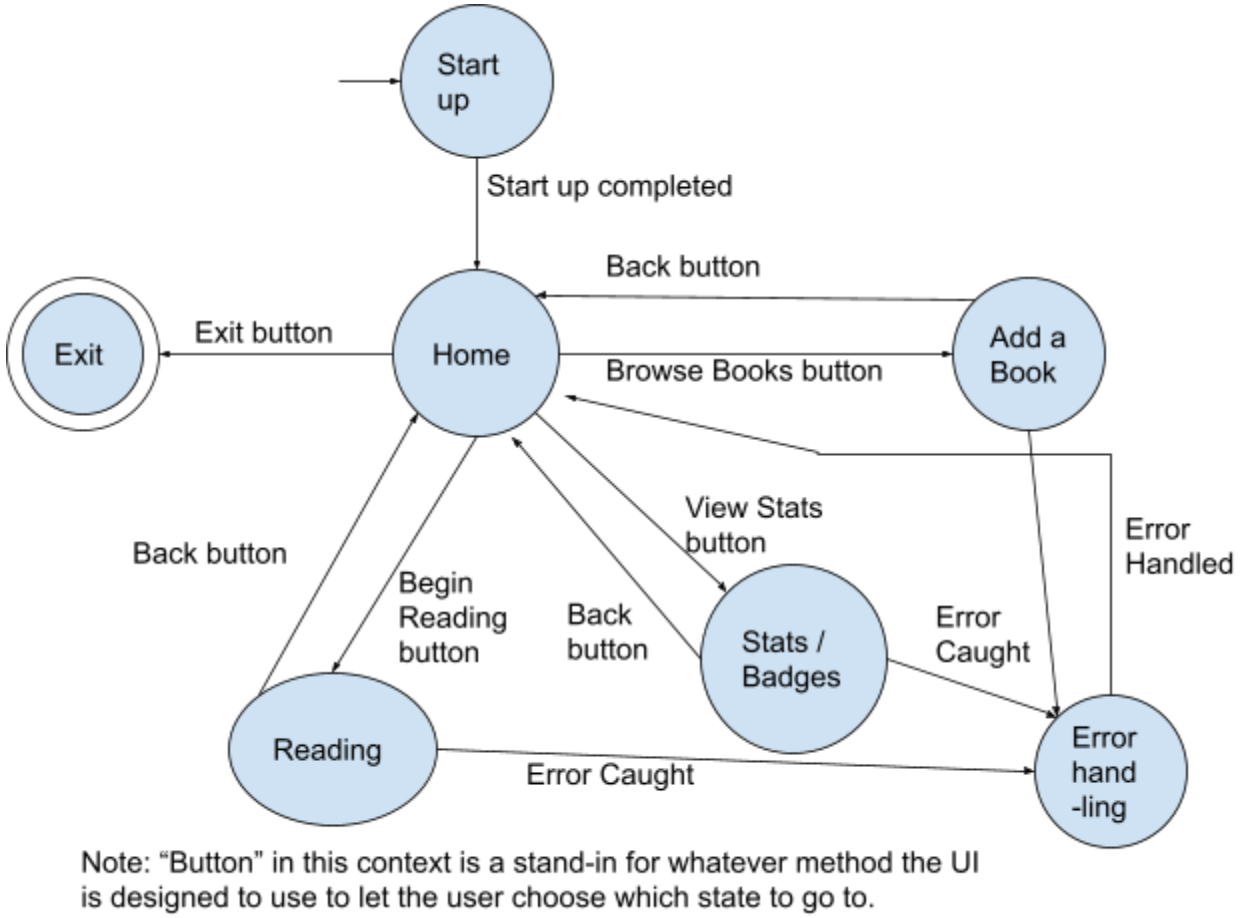

Figure 1: State diagram for basic usage

2.2 Product functions.

For the program to fulfill its goal to help people read more, it should have certain functionality. From a high-level view, the program needs to:

- Track planned books to read
- Track current books
- Track completed books
- Track progress in the current book(s)
- Show statistics about the user's reading
- Provide positive feedback for reading

For a more detailed description of what the program needs to do, see section 3, Specific requirements.

2.3 User characteristics.

The users of this software are expected to be of any reading age, and with access to a device. The program should be intuitive enough to not require any technical expertise or experience to operate. For more on constraints, see section 2.4, and for more on specific requirements, see section 3.

2.4 Constraints.

Software Constraints

- The software and any other supporting software for the client should be installable as a single package (or otherwise easy to install based on our expected broad audience).
- The software is expected to be developed on Unity for improved device compatibility

Hardware Limitations

The minimum hardware requirements to run a unity application (as of 15 October, 2020):

- A cpu with x86 or x64 architecture.
- At least 1 Gb or ram
- Opengl 3.2 and better or DirectX 10 and better
- Internet Connection required to access the online aspects of the app.

2.5 Assumptions and dependencies.

The software is built and run in Unity, and may use some imported libraries. It is assumed that operating systems and devices this software runs on will be compatible with Unity and the used libraries. This may come into question through future OS or hardware advances, but the timescale of such changes is beyond the scope of this project. It is assumed that the remote source of book information for searching will also not be deprecated, though in such a case, the program is expected to still be minimally functional.

3. Specific Requirements.

3.1 External interface requirements

3.1.1 User interfaces

The application has a dynamic user interface. Graphical and otherwise. The user has explicit control over the software based on an organized dashboard or configurations. Moreover, specific icons or logos have been selected to make the application appealing to the User. We believe that such an user interface is necessary to the full deployment and purpose of the application.

3.1.2 Hardware interfaces

No specific hardware interface are necessary for the application except for the device itself.

3.1.3 Software interfaces

The application is meant to interact with a chosen database. This point is pivotal to the deployment of the software. Without it, the application becomes less efficient in achieving its original intent, which is to push people to read by stimulating the exercise of reading with visuals. The visuals in this case would be a sort of visual library. Understandingly so, the database is very essential to the user experience.

3.1.4 Communications interfaces

No plan for now, but may consider it in the future as a way to further extend the interest of users in spending time on the platform, which afterwards would prompt strong reading habits.

3.2 Functional requirements

3.2.1 BW_F_1: The software shall remember the list of books to be read between launches (different program instances from the same file location)

3.2.2 BW_F_2: The software shall remember recorded statistics between launches (different program instances from the same file location). It may recalculate values.

3.2.3 BW_F_3: The software shall remember any in progress books and their state between launches (different program instances from the same file location)

3.2.4 BW_F_4: The software shall remember the list of books completed between aunches (different program instances from the same file location)

3.2.5 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.

3.2.6 BW_F_6: The software shall use user data in conjunction with the remote information source to acquire information on books to be added.

3.2.7 BW_F_7: The software shall keep track of user progression through current books.

3.3 Performance requirements

The application will be storing data on the user's device only. We do not currently plan to support multiple users/logging in/cloud storage. Therefore, we assume that only one instance of the application will be open on a single device at a time and the data in the device belongs to a single user.

The only information which will be handled is information the user inputs. This data will be stored locally on the user's device. We will not be storing any data about users externally. This information includes:

3.3.1 BW_P_1: The BW application should include Alert times or reminders to motivate users.

3.3.2 BW_P_2: The application should include Books read and planned to be read checklists to keep track of user progress.

3.3.3 BW_P_3: The application should include data about the amount of pages read and other data that would help create statistics related to the user's reading habits.

3.3.4 BW_P_4: The application should include progress of current book on TBR up-to-date

3.3.5 BW_P_5: The application should 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.

3.3.6 BW_P_6: The application should 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.

3.4 Configuration requirements

3.4.1 BW_C_1: The BW application shall initialize to the home state when loaded.

3.4.2 BW_C_2: The BW application shall have setting parameters, including the option to reset recorded data

3.5 Security requirements

Not applicable at this phase of the development. The assumption is that the data collected be stored on the user's computer or device at this time.

3.6 Error handling requirements

3.6.1 BW_E_1: The software shall not corrupt saved data in the event of an internal error.

3.6.2 BW_E_2: The software shall not crash nor exit due to connection to the remote book information source not being made.

3.6.3 BW_E_3: The software shall log any internal error it encounters, including error type to a local log file.

3.6.4 BW_E_4: The software shall not crash the user's device upon encountering an Error.

3.7 Software system attributes

3.7.1 Reliability

According to to castsoftware.com, "Application reliability is the probability of a piece of software operating without failure while in a specified environment over a set duration of time. In a perfect world, a reliable piece of software is completely defect free, does not create downtime, and performs correctly in every scenario. In reality, this is difficult to accomplish as an infrastructure becomes increasingly complex and developers must deal with strict time schedules for project completion."(castsofware.com). Knowing this, the above factors would be a requirement for the application. Meaning that the team would make sure to provide the probability of zero failure each time the application is launched.

3.7.2 Availability

Techopedia.com defines application availability as follows, "Application availability is the extent to which an application is operational, functional and usable for completing or fulfilling a user's or business's requirements. This measure is used to analyze an application's overall performance and determine its operational statistics in relation to its ability to perform as required.(techopedia.com). Accordingly, analyzing overall performance and determining its operational statics would be a requirement. Therefore, the team shall abide to the above criteria consistently.

3.7.3 Maintenance

Software maintenance is a part of the Software Development Life Cycle. Its main purpose is to modify and update software applications after delivery to correct faults and to improve performance. Software is a model of the real world. When the real world changes, the software requires alteration wherever possible. (The economicTimes.com)

Understanding that technology is always changing or evolving 40% to 80% of selected resources would be used to achieve this endeavor. This is important because without this standard the application can find itself to be obsolete with the growing years.

3.8 Qualification methods per requirement

Requirement	Demonstration	Testing
BW_F_1	x	
BW_F_2	x	
BW_F_3	x	
BW_F_4	x	
BW_F_5		x
BW_F_6	x	
BW_F_7	x	
BW_P_1	x	
BW_P_2	x	
BW_P_3	x	
BW_P_4	x	
BW_P_6		x
BW_C_1	x	
BW_C_2	x	
BW_E_1		x
BW_E_2		x
BW_E_3		x
BW_E_4		x