|  |  |
| --- | --- |
| Ark: sinister  An AGILE SRS | Abstract  The following contains the functional requirements of the ARK: SINISTER game as of Dec 17, 2015.  A. Schlichter, C. Smukavic, J. Thompson, J. Broomfield, and K. Thomson-Diks  Project Design |

Contents

[Changelog: 3](#_Toc442105433)

[1. Introduction: 4](#_Toc442105434)

[1.1 Purpose: 4](#_Toc442105435)

[1.2 Document Conventions: 4](#_Toc442105436)

[1.3 Intended Audience: 4](#_Toc442105437)

[1.4 Scope: 4](#_Toc442105438)

[1.5 Contact Information/SRS Team Members: 5](#_Toc442105439)

[1.6 References: 5](#_Toc442105440)

[2. Overall Description: 6](#_Toc442105441)

[2.1 Product Perspective: 6](#_Toc442105442)

[2.2 Product Functions: 7](#_Toc442105443)

[2.3 User Classes and Characteristics: 7](#_Toc442105444)

[2.4 Operating Environment: 8](#_Toc442105445)

[2.5 Design/Implementation Constraints: 8](#_Toc442105446)

[2.6 Assumptions and Dependencies: 8](#_Toc442105447)

[3. Specific Requirements: 9](#_Toc442105448)

[3.1 External Interface Requirements: 9](#_Toc442105449)

[3.1.1 User Interfaces: 9](#_Toc442105450)

[3.1.2 Hardware Interfaces: 10](#_Toc442105451)

[3.1.3 Software Interfaces: 10](#_Toc442105452)

[3.1.4 Communication Protocols and Interfaces: 10](#_Toc442105453)

[3.2 Functional Requirements: 10](#_Toc442105454)

[3.2.1 Inventory System: 10](#_Toc442105455)

[3.2.2 Event Generation System: 11](#_Toc442105456)

[3.2.3 Rewards Generation System: 12](#_Toc442105457)

[3.2.4 HUD and User Interface: 12](#_Toc442105458)

[3.2.5 State Machine System: 13](#_Toc442105459)

[3.2.6 Internal Database: 14](#_Toc442105460)

[4. Other Non-Functional Requirements: 15](#_Toc442105461)

[4.1 Performance Requirements: 15](#_Toc442105462)

[4.1.1 Player User: 15](#_Toc442105463)

[4.1.2 Developer User: 15](#_Toc442105464)

[4.1.3 General Requirements: 15](#_Toc442105465)

[4.2 Safety Requirements: 15](#_Toc442105466)

[4.2.1 Player User: 15](#_Toc442105467)

[4.2.2 Developer User: 15](#_Toc442105468)

[4.2.3 General Requirements: 16](#_Toc442105469)

[4.3 Security Requirements: 16](#_Toc442105470)

[4.3.1 Player User: 16](#_Toc442105471)

[4.3.2 Developer User: 16](#_Toc442105472)

[4.3.3 General Requirements: 16](#_Toc442105473)

[4.4 Software Quality Attributes: 16](#_Toc442105474)

[4.4.1 Reliability: 16](#_Toc442105475)

[4.4.2 Availability: 17](#_Toc442105476)

[4.4.3 Maintainability: 17](#_Toc442105477)

[4.4.4 Portability: 17](#_Toc442105478)

[4.5 Project Documentation: 17](#_Toc442105479)

[5.0 Appendices: 18](#_Toc442105480)

[Appendix A: Terminology/Glossary/Definitions 18](#_Toc442105481)

[Appendix B: Work Flow Diagrams 19](#_Toc442105482)

# Changelog:

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Date of Change** | **Change Comment** |
| K Thomson-Diks | Dec 11, 2015 | Initial template. |
| K Thomson-Diks | Dec 14, 2015 | Section 1 and 2 finished. |
| K Thomson-Diks | Dec 15, 2015 | Section 3 and 4 finished. |
| K Thomson-Diks | Dec 15, 2015 | Appendices completed. Use cases added. |
| J Thompson | Dec 17, 2015 | Section 3.2 requirements – Room Generation |
| A Schlichter | Dec 17, 2015 | Section 3.2 requirements – Event, Rewards, State Machine |
| K Thomson-Diks | Feb 1, 2016 | Updates to modified requirements. |

# 1. Introduction:

## 1.1 Purpose:

The following document contains the functional and non-functional requirements for the ARK: SINISTER game. It does not cover game mechanics or summary information about the game itself. Requirements will include procedural generation of rooms and events, the structure of UI, and creation and/or assimilation of assets including floor, wall, and door tiles, in-game objects, character models, audio, and narrative elements. In addition, the requirements sections for each of the items listed above will also name the tools and programs used in their creation (if known).

For the purposes of this initial draft not all application requirements are known or are subject to change as per the application’s capabilities. See Team ARK’s Change Management Plan

## 1.2 Document Conventions:

This document follows a simple tree structure, where the main header of a section is at the top of the page and labelled with a single digit. Subheadings are given a second digit in the form of a single decimal place. Further subheadings are likewise separated with additional decimal places.

## 1.3 Intended Audience:

The intended audience of the SRS document are the primary stakeholders in the development of ARK: SINISTER: the development team including Aaron Schlichter, Chris Smukavic, Jordan Thompson, Josh Broomfield, and Kyle Thomson-Diks, and the project supervisor and adviser, Russ Foubert.

## 1.4 Scope:

The ARK: SINISTER game is comprised of a few major components that must function independently but in unison in order to provide a complete game experience.

* The room selection system: a system in place for selecting a particular room from a list of possible rooms by the player. Allows for the click and drag of rooms as objects from an inventory list to the game world and the creation of the room in the world as a playable area.
* The event generation system: a set of rules governing the procedural generation of in-game events. Allowable events are ones that provide a non-zero chance of success and whose options provide the player the option to use one or a combination of equipped items.
* The rewards and progression system: a set of rules governing the random generation of additional resources and abilities given to the player after clearing an event.
* The HUD and User Interface system: includes the main menu, any loading screens, and the heads-up displays in-game as well as the systems menu, character and inventory screens accessed as submenus of the HUD.
* A state machine system to handle conditional events and the availability of player options during events.
* An on-board storage system for saving the game state to local memory to be retrieved later.

For a complete explanation of the functional and non-functional requirements for each of the components listed above please see section 3: Functional Requirements and Section 4. Non-Functional Requirements.

## 1.5 Contact Information/SRS Team Members:

Aaron Schlichter: aarons@competers.com

Chris Smukavic: nadnadsetset@gmail.com

Jordan Thompson: nordan2@hotmail.com

Josh Broomfield: josh\_555555@hotmail.com

Kyle Thomson-Diks: felinix18@hotmail.com

## 1.6 References:

Donn Le Vie, Jr. Writing Software Requirements Specifications (SRS). Aug 29, 2010. <http://techwhirl.com/writing-software-requirements-specifications/>

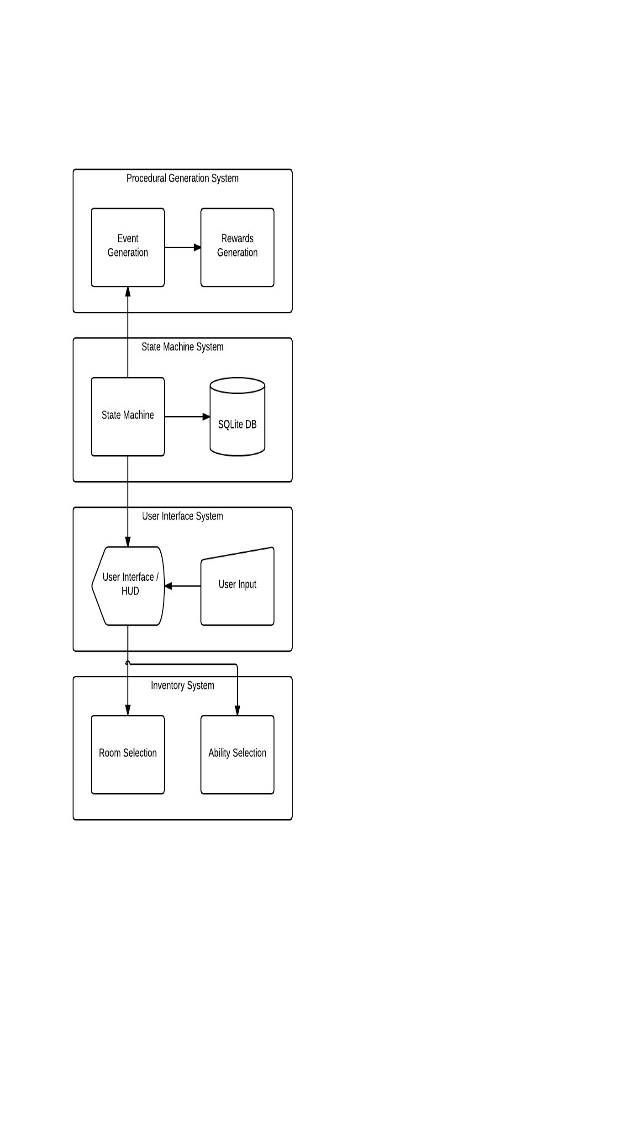
Geagea, Sarah. Software Requirements Specification: Amazing Lunch Indicator. <http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf>

# 2. Overall Description:

## 2.1 Product Perspective:

The ARK: SINISTER product will consist of three major systems all within the same parent system. The game itself will be completely self-contained and no additional software components will be necessary to run the game except for possible plugins related to playing the game, such as the Unity Engine.

The four main systems are shown in figure 1 to the right. The procedural generation system will be one of the more complicated systems and contain the most components. It is used for randomly generating the events when entering a new room and for generating appropriate rewards based on the difficulty of the event, the type and size of room, and the level of completion of the player.



Figure

When a player selects the type of room to create an event is generated based on the current game state and tied to the selected room. In order to access and use the room the player must clear the objectives of the event successfully.

The rewards generation is linked to the event system because certain types of rooms and events can spawn different varieties of resources, player abilities and unlockable rewards. This is randomized based on the current game state.

The state machine is one of the core systems responsible for overseeing the collective rules of the procedural generation systems and what is or is not displayed to the user via the HUD and in-game menus. This state machine keeps track of all potential variables that could exist within any particular stage of the game and updates certain components as necessary when the state changes. It is also responsible for committing changes to the on-board database during a save event for later retrieval. It is currently unknown how many separate game states will be saved to a single instance of the database.

The user interface is the primary way the user is able to interact with the game itself. The user interface includes the main menu system and the heads-up-display and is responsible for responding to any keyboard or mouse clicks the user creates as input.

Finally, the Inventory system is a catch all system for the implementation of a click and drag interface for user controls. Room selection is done through an in-game menu whereby the user can select a room type, shape, and size and then drag that room into the game world for construction. The rooms must fit like puzzle pieces on the game world so the Inventory system must be able to handle the bounds and rules for where a room can be placed. In addition, the player gradually unlocks additional abilities and functionality for their main handheld tool. The abilities unlocked are clicked and dragged from their inventory into equip able slots in the HUD. The Inventory system must be able to handle which abilities are unlocked and where they can be dragged.

## 2.2 Product Functions:

The game requires an install of the relevant files and assets to function properly and possibly requires the installation of the Unity framework on the host PC because it is built using the Unity engine.

Upon booting into the game the user is presented with the Main Menu, which branches to several optional menus for configuration and management of the game properties. When starting the game the user is given a short expositional narrative explaining the premise and current situation depicted in the game’s narrative. When they’ve finished the exposition they are given basic instructions on game controls and functions in order to use and play the game; the tutorial is very short and can be accessed through the HUD at any time.

During normal game play the user is not interrupted by system messages. The only messages or output they receive are messages relating to interactable objects or events they encounter.

The main menu is accessible from the HUD and provides the save, options, and exit buttons. Opening any sub menu from in-game pauses the game and closing the menus un-pauses the game and continues game play. By exiting the game the client closes and returns to their desktop.

## 2.3 User Classes and Characteristics:

The two User Classes intended for the ARK: SINISTER Application are Player Users and Developer Users.

A Player User has access to all features of the release version of the game. They can access all configurable options for volume, install location, and data management of the game. Because each copy of the game is independent of each other, all players are treated as administrative users for their owned copy.

During production an additional user type exists: the Developer User. The administrator has access to all debug configurations of the game as well as the base code, internal database and assets. The administrator class can also be a player, but a user assigned to be a player cannot also be an administrator.

## 2.4 Operating Environment:

The operating environment of ARK: SINISTER is intended to be the player user’s PC. The game is not intended to run on a mobile device or through web deployment. The user’s PC must meet the minimum requirements for installing and running the game. The current minimum requirements of the game are unknown.

A player user’s PC is defined as any PC owned and operated by the player user and can be a personal desktop system or laptop. Windows is currently the only compatible operating system for ARK: SINISTER, with the possibility of porting the game to the Mac OS in a later version.

## 2.5 Design/Implementation Constraints:

Because of the nature of household PCs, the game must be designed to run on the minimum specified hardware and not above. This limits the resolution of textures and assets we are able to include in the game without experiencing performance issues.

We must also consider the amount of hard disk space we use to be of limited quantity in order to be easily accessible for download from the Steam online store. The arbitrary limitation in this regard is approximately 100Mb.

## 2.6 Assumptions and Dependencies:

It is assumed that this game will always be run on PCs that meet or exceed the minimum requirements of the specification and that the player user understands and can read English. The game cannot be played on a PC or any other device without sufficient power or whose system does not meet the minimum specified requirements.

It is also assumed that the player user has installed ARK: SINISTER on a PC running the Windows operating system.

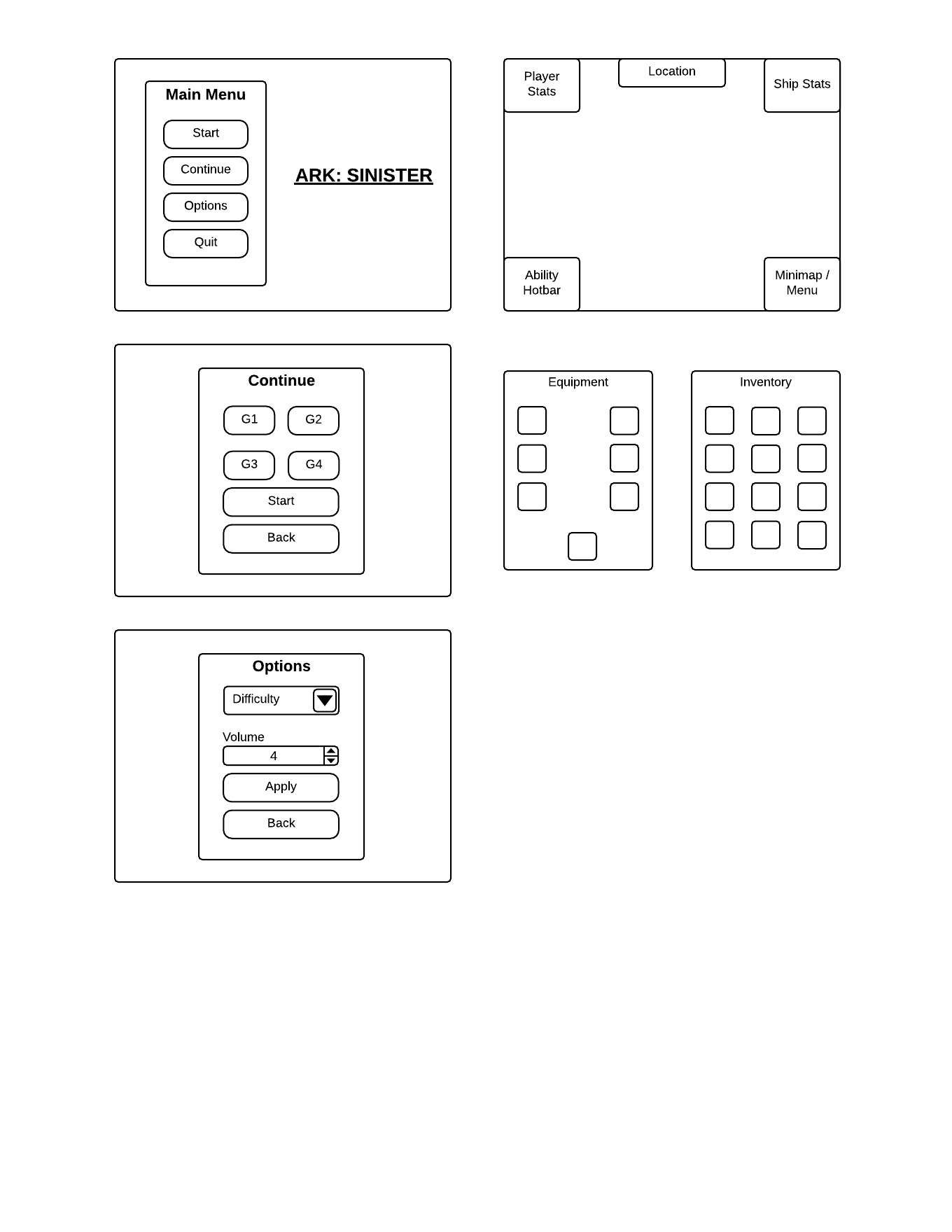
# 3. Specific Requirements:

The following sections provide detailed descriptions of all the software requirements for ARK: SINISTER and its component systems.

## 3.1 External Interface Requirements:

This section provides requirements for specific interfaces.

### 3.1.1 User Interfaces:



Figure

Figure 2 on the right is the basic wireframes of the proposed user interface design of the main menu system.

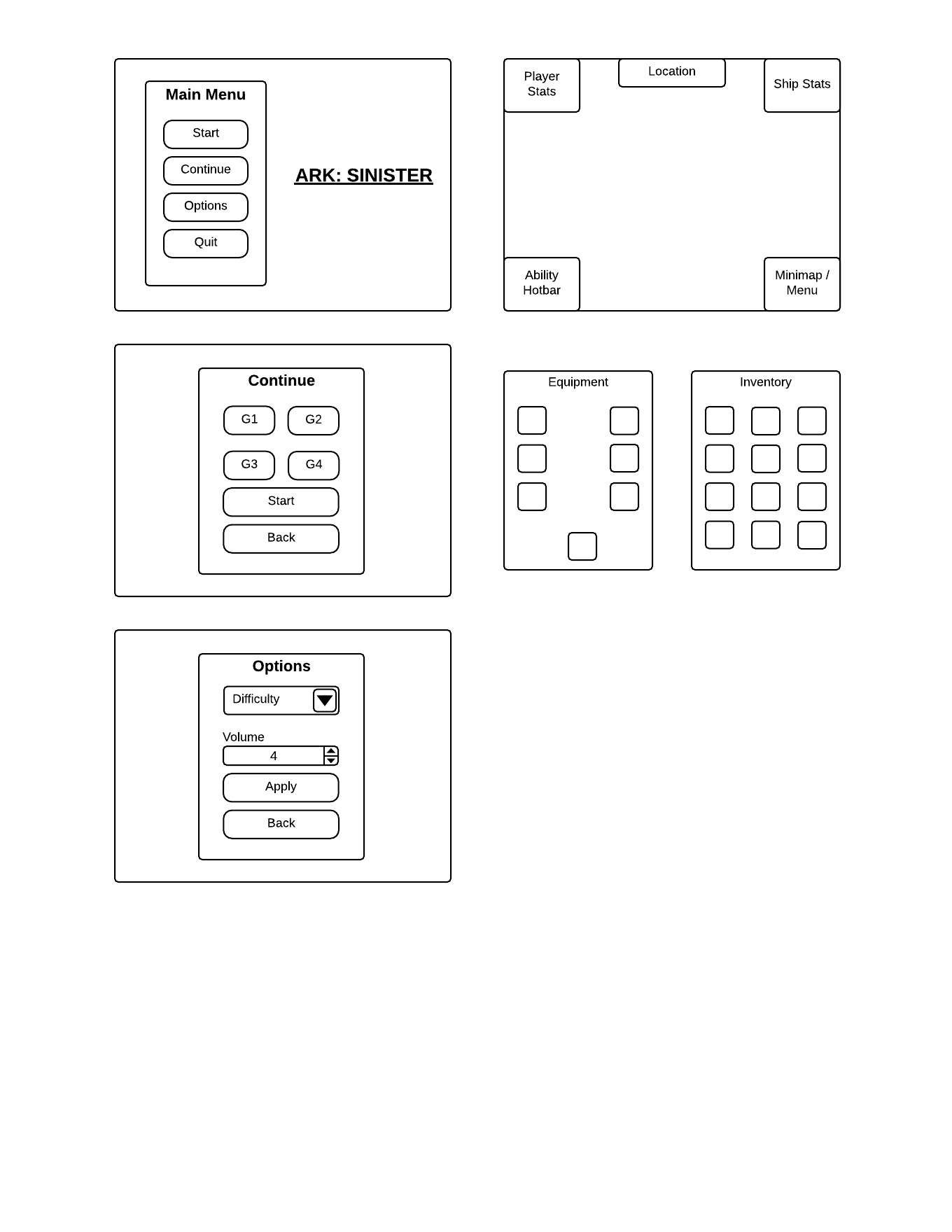
When starting the game, the user will be presented with the main menu page and be given the choices of Start Game, Continue Game, Options, and Quit Game. Starting the game will start a new game session; no other menus are necessary at this point. The continue button will bring the player user to the continue menu and they will be asked to select which of the previously saved games to load and then to start or go back to the main menu. Clicking the start button from here will load the selected saved state and start the game.

The options menu currently only has plans for adjusting the difficulty settings of the game and to adjust the master volume of game sounds. The user can apply these changes or drop them by going back to the main menu.

Quitting the game from the main menu will close the client. If there are any unsaved changes to configuration or to any open game states they will be lost.

Figure 3 is the proposed basic design of the heads-up-display during game play and a sample wireframe displaying the Inventory and Equipment menu. Both of these sub menus are launched from buttons in the mini-map section of the HUD.

The HUD itself is made up of five smaller components. In clockwise order starting in the top left there is the player stats, location, ship stats, mini-map, and ability hot bar. The player stats section keeps track of and displays the player’s vitals: health points and energy. The location displays the name of the player’s current location. The ship stats section keeps track of available resources: scrap, energy, and life support. The mini-map displays an overhead view of the ship and the current room, one at a time, and houses buttons to open the main menu, the equipment menu, and the inventory menu as seen in the second half of Figure 3. Lastly is the ability hot bar, which is a quick action toolbar mapped to three keys on the keyboard. Pressing one of these keys will activate the ability slotted into the associated section.



Figure

The inventory and equipment menus are similar. When the player collects a piece of equipment they can access it from the inventory menu and a double click on the icon will equip it to the character. Once equipped, the item will be shown in the appropriate slot in the equipment menu.

### 3.1.2 Hardware Interfaces:

The ARK: SINISTER project does not require any hardware interfaces.

### 3.1.3 Software Interfaces:

The ARK: SINISTER project does not require any software interfaces.

### 3.1.4 Communication Protocols and Interfaces:

In order to save game states to the on-board database there will be a basic protocol set up to handle serializing the game state in such a way that all the data can be automatically saved and retrieved at any time. The database will be made using SQLite for a light-weight infrastructure. This will be a custom protocol for serializing specific data from the state machine.

The ARK: SINISTER project does not require any communications interfaces.

## 3.2 Functional Requirements:

This section contains the specific requirements for components expressed in Section 1.4 Scope.

### 3.2.1 Inventory System:

#### 3.2.1.1 Player User:

* The Player shall open the Room Placement interface through key binds in the user interface.
* The Player shall open the Ability selection interface through key binds in the user interface.
* The Player shall be able to navigate to different rooms by moving the character sprite to a door connected to another room.
* The Player shall be able to choose the type, size, and shape of a room in the menu before dragging the room into the world.
* The Player shall have access to rooms based on the resources required compared to resources available.

#### 3.2.1.2 Developer User:

* The Developer shall be able to access all possible rooms regardless of resources required.

#### 3.2.1.3 General Requirements:

* The system shall generate an in-game explorable section to the game world when the room is properly attached to existing rooms.
* The Inventory system shall only allow rooms to be connected by their existing doors.
* The Inventory system shall only allow rooms to be dragged and placed in the world where no section of the room is bisected by the edge of the ship.
* The Inventory system shall colour rooms during the placement process to communicate a valid and invalid placement. Red for invalid, and green for valid.
* The system shall hide the new room’s layout from the Player until they are explored.
* The system shall highlight unlocked abilities and block out locked abilities in the Inventory menu.
* The system shall only allow interaction with unlocked abilities.
* The system shall allow drag and drop of abilities into the quick action toolbar in the bottom left of the HUD.
* The system shall return a clicked ability to the Inventory menu if dropped in an invalid spot.

### 3.2.2 Event Generation System:

*3.2.2.1 Player User:*

* The Player shall receive a procedurally generated event upon moving to a new unexplored room.
* The Player shall be shown a list of possible options to choose to resolve the event.
* The Player shall select an available option, and shall be told the outcome of the event

*3.2.2.2 Developer User:*

* All requirements from Section 3.2.2.1 Player User remain the same.

*3.2.2.3 General Requirements:*

* The Event Generation System shall randomly select an event out of a predefined set of events based upon the current game state and room being generated
* The Event Generation System shall determine whether the player passes or fails the event based on conditions contained in the event.
* The Event Generation System shall call the Rewards Generation System reward upon completion to determine which rewards the player will receive

### 3.2.3 Rewards Generation System:

*3.2.3.1 Player User:*

* The Player has no functional requirements for this system

*3.2.3.2 Developer User:*

* All requirements from Section 3.2.3.1 Player User remain the same.

*3.2.3.3 General Requirements:*

* Rewards shall be generated upon entering a room.
* Three tiers of rewards shall be generated.
* Rewards shall scale down in quality based on tier.
* Rewards shall be generated from similar lists of possible rewards based on room, event, and size.
* Rewards shall be eared based on level of completion of the event. Tired 1, 2 and 3 shall be received when the player achieves gold completion. Tier 2 and 3 shall be received when the player achieves silver completion. Their 3 shall be received when the player achieves bronze completion.

### 3.2.4 HUD and User Interface:

#### 3.2.4.1 Player User

* The Player shall be able to Start a new game, Continue a saved game, and Quit the client from the title menu.
* The Player shall be able to Configure game settings including (but not limited to) game volume and difficulty level from the Options menu.
* The player shall be able to access the Main Menu from within a game session using the Main button in the Mini-map section of the HUD.
* The player shall be able to open the Inventory and Equipment sub menus from the respective buttons in the Mini-map section of the HUD.
* The player shall be able to toggle between a full scale world view and a small scale room view of the mini-map using arrow buttons beside the display in the Mini-map section of the HUD.
* The user shall be able to click the ability portraits in the Ability Hot Bar to activate the effects of the equipped ability.
* The user shall be able to click the corresponding keyboard key to activate the effects of the equipped ability as shown in the Ability Hot Bar.
* The user shall be able to double click the ability or equipment icon from the Inventory menu to equip the ability.
* The player shall be able to drag-and-drop the ability onto the Ability Hot Bar to equip the ability to the desired location.
* An ability dragged onto a particular slot on the Ability Hot Bar will override the currently equipped ability in that slot and return the overridden ability to the inventory menu.
* The player shall not be able to interact with the Player Stats, Location, or Ship Stats sections of the HUD directly.
* The Health Bar in the Player Stats section of the HUD shall be coloured green while above 70% remaining health, turn to yellow when between 31% and 70% inclusive remaining health, and turn to red when at or below 30% remaining health.
* The Inventory and Equipment menu shall be interactable such that they can be moved to different locations on the screen, but never outside the screen, and will remember where they were closed in order to reopen at the desired locations.
* The Inventory menu shall be opened using the Inventory button in the Mini-map section of the HUD or by pressing the “V” key on the keyboard.
* The Equipment menu shall be opened using the Equipment button in the Mini-map section of the HUD or by pressing the “C” key on the keyboard.
* The Main Menu shall be accessible from the Main button in the Mini-map section of the HUD or by pressing the ESC key on the keyboard.

#### 3.2.4.2 Developer User:

* All requirements from Section 3.2.4.1 Player User remain the same.
* The Developer shall be able to enter debug mode from the Options menu to allow direct modification of any aspect of the state machine.
* Debug Mode is accessible from the Options menu wherever the Options menu appears.
* The Developer User shall be able to bypass specific requirements of game mechanics in order to test interactions between different game parts.
* The Developer User shall be able to create fundamentally impossible scenarios through state machine manipulation that should never occur.
* The Developer User shall have access to any unlockable item and ability.

#### 3.2.4.3 General Requirements:

* The Player User shall never have access to the Debug system.
* The Developer User shall be able to change context between Player and Developer User classes at will in order to test interactions between game states such that the game shall behave as though the Developer was a Player User while this context is changed.

### 3.2.5 State Machine System:

*3.2.5.1 Player User:*

* The Player shall save and load settings to and from the State Machine System via the User Interface.

*3.2.5.2 Developer User:*

* The Developer shall be able to directly modify any aspect of the state machine for debugging purposes.

*3.2.5.3 General Requirements:*

* The Player shall never have direct access to the State Machine System.
* The state machine shall track all variables associated with all aspects of the game and its systems at all times.

### 3.2.6 Internal Database:

#### 3.2.6.1 Player User:

* The Player shall have no direct access to or interaction with the internal database at any time.
* The Player shall be able to save their current game state to the database by clicking the Save button from the Main menu accessed in the Mini-map section of the HUD.
* The Player shall be able to load saved game states from the database by clicking on the desired saved game from the Continue Game menu accessed from the title screen.

#### 3.2.6.2 Developer User:

* The Developer shall have no direct access to or interaction with the internal database except where that limitation would interfere with the requirement that they are able to modify the state machine.

#### 3.2.6.3 General Requirements:

* The database will be built using SQLite and be built into the application itself to reduce application size and complexity.
* The database shall be relational and allow for multiple saved states at a time.
* The database shall currently limit the number of saved game states to four to decrease the storage requirement of the game.

# 4. Other Non-Functional Requirements:

## 4.1 Performance Requirements:

The following section contains information about specific performance requirements as they relate to particular User Classes.

### 4.1.1 Player User:

* The Player shall experience no load times between menus in the title screen.
* The Player shall experience no load times between starting a game in the title screen and starting the game session.
* The Player shall experience no load times while moving between rooms in the ship.
* The Player shall experience no latency between clicking a button and an in-game response.

### 4.1.2 Developer User:

* The Developer shall have the same experience as expressed in Section 4.1.1 Player User.

### 4.1.3 General Requirements:

* The Application shall require no more than 200Mb in hard disk space and preferably be below 150Mb.
* The Application shall require no more than 1 GB in RAM and preferably require less than 500Mb.
* The Application shall require a Windows compatible environment.
* The Application shall require the use of audio output devices for the full experience.
* The Application shall require the use of a computer monitor.
* The Application shall require the use of a keyboard and mouse setup for manual control.

## 4.2 Safety Requirements:

The following section contains information about particular Safety requirements as they pertain to specific User Classes.

### 4.2.1 Player User:

* The Player shall not be asked to input any personal information anywhere within the Application.

### 4.2.2 Developer User:

* The Developer shall not include any code that installs any unnecessary functionality including, but not limited to, viruses, either known or unknown, key loggers, location trackers, IP sniffers, or alternate code packets that do not relate to the core functionality of ARK: SINISTER.
* The Developer User shall not have access to the Debug system in the Production version of the Application.

### 4.2.3 General Requirements:

* The Application shall not track nor store any data of the Player’s nor the Player’s computer outside the realm of the state machine.
* The Application shall not be aware of the Player’s location data nor any Meta data of the computer’s environment where it does not concern the state machine.

## 4.3 Security Requirements:

### 4.3.1 Player User:

* The Player shall not be required to log in to the game itself. This requirement does not include logging into the gaming platform ARK: SINISTER is hosted on.

### 4.3.2 Developer User:

* The Developer shall not be required to log in to the game. This requirement does not include logging into the development studio.

### 4.3.3 General Requirements:

* See the requirements in Section 4.2 Safety Requirements sub-section 4.2.3 General Requirements.
* The Application shall not require a connection to the internet nor any network known or unknown to function.
* The Application shall function as a standalone product, complete unto itself.

## 4.4 Software Quality Attributes:

The following section outlines requirements for the Reliability, Availability, Maintainability, and Portability of the ARK: SINISTER system.

### 4.4.1 Reliability:

* The Application shall not interfere with processes, active or inactive, on the Player User’s computer.
* The Application shall not crash when run beside processes, active or inactive, on the Player User’s computer.
* The Application shall not introduce bugs into the computer to which it is installed.
* The Application shall not continue running during a power outage.
* The Application shall not continue running after the User presses the Quit Game button in any menu where the Quit Game button appears.
* The Application shall open and continue running as long as the User has not pressed the Quit Game button where this does not conflict with the requirement about a power outage.
* The Application shall not start new processes nor end existing processes not related directly to the ARK: SINISTER installation.

### 4.4.2 Availability:

* The ARK: SINISTER Application shall be available to any potential Player User as long as this does not interfere with local distribution laws.
* Once installed the Application shall be available to any Player User using the same shared location.

### 4.4.3 Maintainability:

* The Application shall be coded in compliance with SET Coding Standard listed in the SET Coding Standards notebook available online.
* The Application shall be coded to be modular and with naming conventions appropriate to the sections of code being written.
* The Application shall be well documented with function header and in-line comment sections explaining key aspects of functionality for all future development efforts.
* The Application shall be easily extendable and modifiable such that a new developer can spend a minimal amount of time learning the current code base and begin adding modifications and features.
* The Application shall be testable using specific test harnesses to test stress levels, data input and validation where that may apply, and specific interactions between in-game objects and game states.
* The test harnesses shall be automated.

### 4.4.4 Portability:

* The Application shall not be portable between individual computers outside of separate installations.
* The Application shall be designed for the Windows operating system.
* The Application shall be ported to the Mac operating system if time allows in the development cycle.

## 4.5 Project Documentation:

The following documentation for the ARK: SINISTER project has been created as is available upon request to any of the team members listed in Section 1.5 of this document.

* Project Charter
* Preliminary Scope Document
* Change-Management Document
* Preliminary Project Schedule
* Market Feasibility Study
* Preliminary Project Investigation Report and Game Design Document

# 5.0 Appendices:

## Appendix A: Terminology/Glossary/Definitions

|  |  |
| --- | --- |
| Term: | Definition: |
| User Class | One of two potential types of users expected to use the ARK: SINISTER Application. |
| Player User | The first of two User Classes. Refers to the general user of the Application with no special privileges to the system. |
| Developer User | The second of two User Classes. Refers to the administrative user of the Application with special privileges to the database and state machine. |
| Application | Refers to the ARK: SINISTER game. |
| HUD | The Heads-Up-Display. A piece of the user interface specific to an in-game session to display vital information to the player about the game state. |
| State Machine | A central controller to handle the run-time storage and management of all possible variables associated with running the game during a game session. |
| Procedural Generation | A type of random generation of particular game states to facilitate different scenarios for every game session. |
| Game Session | The time taking place from the start of a new game to the end of the game. |

## Appendix B: Work Flow Diagrams

