For Wolf House Points System

Produced for Sir James Whitney School for the Deaf

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 01/12/19 | 1.0 | Initial Report | Gabriela Braga |
|  |  |  |  |

Table of Contents

1. Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, Acronyms and Abbreviations 3

1.4 References 3

1.5 Overview 3

2. Overall Description 3

2.1 Use-Case Model Survey 3

2.1.1 Introduction 3

2.1.2 Survey Description 3

2.1.3 Use-Case Model Hierarchy 3

2.1.4 Diagrams of the Use-Case Model 3

2.2 Assumptions and Dependencies 3

3. Requirements 3

3.1 Use-Case Specifications 3

3.2 Functionality 3

3.2.1 <Functional Requirement One> 3

3.3 Usability 3

3.3.1 <Usability Requirement One> 3

3.4 Reliability 3

3.5 Performance 3

3.5.1 <Performance Requirement One> 3

3.6 Supportability 3

3.6.1 <Supportability Requirement One> 3

3.7 Design Constraints 3

3.7.1 <Design Constraint One> 3

3.8 Online User Documentation and Help System Requirements 3

3.9 Purchased Components 3

3.10 Interfaces 3

3.10.1 User Interfaces 3

3.10.2 Hardware Interfaces 3

3.10.3 Software Interfaces 3

3.10.4 Communications Interfaces 3

3.11 Licensing Requirements 3

3.12 Legal, Copyright and Other Notices 3

3.13 Applicable Standards 3

# Introduction

## Purpose

[Specify the purpose of this SRS. The SRS should fully describe the external behavior of the application or subsystem identified. It also describes nonfunctional requirements, design constraints and other factors necessary to provide a complete and comprehensive description of the requirements for the software.]

This Software Requirements Specification (SRS) will outline the description of a software application referred to as the Wolf House Point System to be created for Sir James

## Scope

[A brief description of the software application that the SRS applies to; the feature or other subsystem grouping; what Use Case model(s) it is associated with, and anything else that is affected or influenced by this document.]

This SRS will define the requirements for the Wolf House Point System (WHPS).

## Definitions, Acronyms and Abbreviations

[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to interpret properly the SRS.  This information may be provided by reference to the project Glossary.]

SRS – Software Requirements Specification

UML – Unified Modeling Language

WHPS – Wolf House Point System

## References

[This subsection should provide a complete list of all documents referenced elsewhere in the SRS. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

## Overview

[This subsection should describe what the rest of the SRS contains and explain how the SRS is organized.]

# Overall Description

[This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in section 3, and makes them easier to understand. Include such items as product perspective, product functions, user characteristics, constraints, assumptions and dependencies, and requirements subsets.]

## Use-Case Model Survey

[This section contains an overview of the use-case model or the subset of the use-case model that is applicable for this subsystem or feature.  This includes a list of names and brief descriptions of all use cases and actors, along with applicable diagrams and relationships. This section describes the use-case model comprehensively, in terms of how the model is structured into packages and what use cases and actors there are in the model. If you are using packages, the document shows the model structure hierarchically.]

### Introduction

[Introduction to the use-case model.]

The following Use-Case model will describe the functionality provided by the WHPS and the external entities that interact with the system, and it will define the boundary of the system.

### Survey Description

[Survey description of the use-case model.]

The following Use-Case model uses UML to describe the WHPS. Users are able to log into the system through an authentication function and change their password once logged in. Users may also manage user roles, modify rules and manage choices through the modify code tables function. The Faculty or Administration User may manage the student records. They may also import those records and then export them again with the assistance of the ITS Database. These users may also generate the report and calculate the distribution for the students into the possible disciplines. The administrator may manage the staff login information and reset the system.

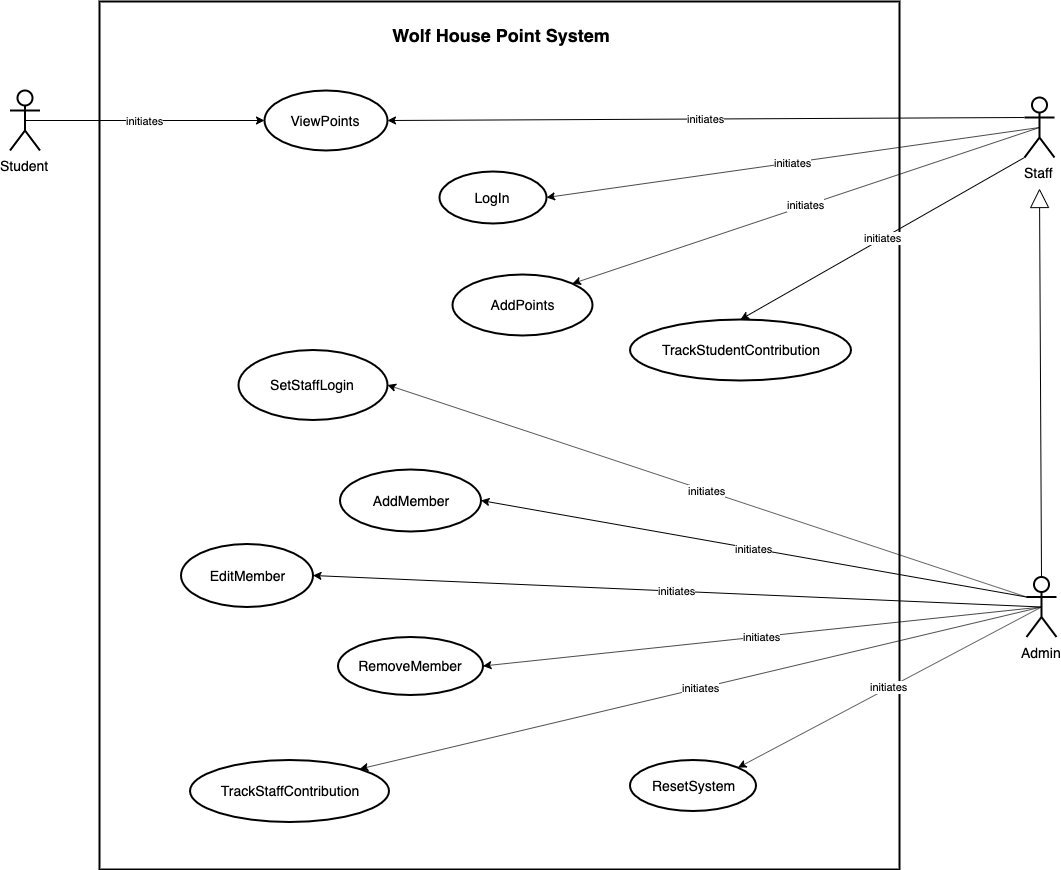
### Use-Case Model Hierarchy

[This section presents the use-case packages hierarchically, explains the dependencies among them, and shows the content of each package recursively. If the model has several levels of packages, those at the top-level are presented first. The packages within these are presented next, and so on, all the way down to the packages at the bottom of the hierarchy. For each package include:

* The Name.
* A Brief Description explaining the package's function and role in the system. The description must be understandable to any developer who wants to use the package.
* A list of the use cases owned by the package, including the name and brief description of each use case.
* A list of actors owned by the package, including the name and brief description of each actor.
* A list of relationships owned by the package, including the name and brief description of each relationship.
* A list of the packages directly owned by the package, with each package presented in the same hierarchical manner as above]

### Diagrams of the Use-Case Model

[Diagrams, primarily use-case diagrams, of the entire use-case model are included here.]



|  |  |
| --- | --- |
| **Use Case** | **Description** |
| ViewPoints |  |
| LogIn |  |
| AddPoints |  |
| TrackStudentContribution |  |
| SetStaffLogin |  |
| AddMember |  |
| EditMember |  |
| RemoveMember |  |
| ResetSystem |  |
| TrackStaffContribution |  |

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Student |  |
| Staff |  |
| Admin |  |

## Assumptions and Dependencies

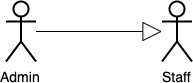
[This section describes any key technical feasibility, subsystem or component availability, or other project related assumptions on which the viability of the software described by this SRS may be based.]

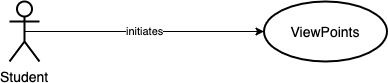
# Requirements

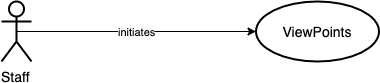
[This section of the SRS should contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements.   When using use-case modeling, the majority of these requirements are captured in the use cases.]

## Use-Case Specifications

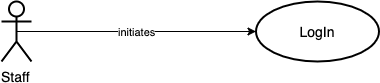
[In use-case modeling, the use cases often define the majority of the functional requirements of the system, along with some non-functional requirements. For each use case in the above use-case model, or subset thereof, enclose the use-case specification here. If you have documented use cases in a separate document, cross reference to all applicable external use-case specifications in this section. Make sure that each requirement is clearly labeled.]



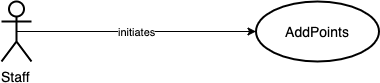




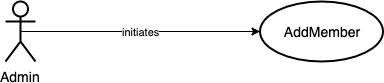
|  |  |
| --- | --- |
| Use case name | ViewPoints |
| Actors | Initiated by Student, Staff, or Admin |
| Entry Conditions | The user has browsed to the site homepage. |
| Flow of Events | 1. The three Wolf Houses are displayed with the number of House Points for each House. 2. A login labeled ‘Staff Login’ is displayed under the Houses, showing username and password entry fields and a button labeled ‘Log In’ |
| Exit Conditions | The user has been shown the house points. |



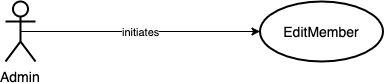
|  |  |
| --- | --- |
| Use case name | LogIn |
| Actors | Initiated by Staff or Admin |
| Entry Conditions | The login is displayed to the user. |
| Flow of Events | 1. The user enters a username into an input field labeled ‘Username’. 2. The user enters a password, which is protected with asterisks as they type, in a field labeled ‘Password’. 3. The user presses a button labeled ‘Log In’. 4. *If the login information is incorrect, the system responds by clearing the input fields and displaying a message informing the user, ‘Username and/or Password incorrect’.* 5. *If the login information is correct for the Staff account, the system responds by displaying the Staff account page. There are buttons labeled ‘Add Points’ and ‘Track Student Contribution’.* 6. *If the login information is correct for the Admin account, the system responds by displaying the Admin account page. There are buttons with the following labels:*    1. *‘Add Points’*    2. *‘Add Wolf Member’*    3. *‘Edit Wolf Member’*    4. *‘Remove Wolf Member’*    5. *‘Track Student Contribution’*    6. *‘Track Staff Contribution’*    7. *‘Set Staff Login’*    8. *‘Reset System’* |
| Exit Conditions | The user is logged in to the appropriate account and the appropriate menu options are displayed. |



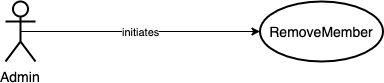
|  |  |
| --- | --- |
| Use case name | AddPoints |
| Actors | Initiated by Staff or Admin |
| Entry Conditions | The user is logged into the Staff account and has selected the ‘Add Points’ button. |
| Flow of Events | 1. The user starts to type the name of the Wolf Member into an input field labeled ‘Wolf Member’. 2. *The system responds by displaying a list of registered Wolf Member names that match the user’s entry as they type.* 3. The user selects the Wolf Member from the list of available names. Points cannot be added if the ‘Wolf Member’ field is empty, or is not a name registered in the system (ie. not selected from the list of available names). 4. *The system responds by displaying the Wolf House name associated with the selected Wolf Member in a field labeled ‘House’.* 5. The user selects the reason for the points being given from a dropdown list labeled ‘Reason’ with the following options:    1. ‘Paw’    2. ‘Club/Group Participation’    3. ‘Sports Team Participation’    4. ‘Volunteer Event/Collecting Hours’ 6. The user may or may not enter optional notes in a text field labeled ‘Optional Notes’ (ex: ‘honesty’, ‘Overcame their fear of presenting to the class after lots of practice’, etc). 7. The user selects the number of points to be added using a dropdown list with the following options: 5, 10, 15, 20, 25, 50, and 100. The default selected number is 5. 8. The user starts to type their own name into an input field labeled ‘Staff Sign-off’. 9. *The system responds by displaying a list of registered staff member names that match the user’s entry as they type.* 10. The user selects their name from the list of available names. Points cannot be added if the ‘Staff Sign-Off’ field is empty, or is not a name registered in the system (ie. not selected from the list of available names). 11. Once the user has completed all the required fields, they press a button labeled ‘Add Points’ to successfully add the points. This button is unable to be clicked if the user has not completed all the required fields. 12. At any time, the user may hover over a button labeled ‘Help’ to display a tool tip with instructions on how to complete the form. 13. At any time, the user may press a button labeled ‘Back’ to go back to the main menu, without saving. |
| Exit Conditions | The new total of House Points is updated and displayed under the house symbol. |



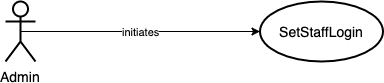
|  |  |
| --- | --- |
| Use case name | AddMember |
| Actors | Initiated by Admin |
| Entry Conditions | The user is logged into the Admin account and has selected the ‘Add Wolf Member’ button. |
| Flow of Events | 1. The user types the Wolf Member’s first name into an input field labeled ‘First Name’. 2. The user types the Wolf Member’s last name initial into an input field labeled ‘Last Name Initial’. 3. The user selects the Wolf Member’s associated House from a dropdown list labeled ‘House’ with the following options:    1. ‘Arctic Wolf’    2. ‘Timber Wolf’    3. ‘Grey Wolf’ 4. The user selects a radio button labeled ‘Student’ or a radio button labeled ‘Staff’. The default selected option is ‘Student’. 5. Once the user has completed all the required fields, they must press a button labeled ‘Add Wolf Member’ to successfully add the Wolf Member. This button is unable to be clicked if the user has not completed all the required fields. The Wolf Member cannot be added if that exact name is already registered in the system. 6. *If the Wolf Member information is invalid, the system responds by saying ‘\_\_\_\_\_\_ is already a name in the Wolf House System’. The fields remain filled so the user may edit the information and attempt to add the Wolf Member again.* 7. *If the Wolf Member information is valid, the system responds by providing a confirmation message to the user saying ‘\_\_\_\_ was successfully added to house \_\_\_\_\_’. The fields are cleared so the user may add another Wolf Member.* 8. At any time, the user may press a button labeled ‘Back’ to go back to the main menu, without saving. |
| Exit Conditions | The new member is added. |



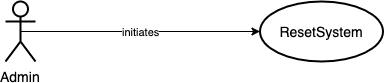
|  |  |
| --- | --- |
| Use case name | EditMember |
| Actors | Initiated by Admin |
| Entry Conditions | The user is logged into the Admin account and has selected the ‘Edit Wolf Member’ button. |
| Flow of Events | 1. The user starts to type the name of the Wolf Member into an input field labeled ‘Wolf Member’. 2. *The system responds by displaying a list of registered Wolf Member names that match the user’s entry as they type.* 3. The user selects the Wolf Member from the list of available names. A Wolf Member cannot be edited if the ‘Wolf Member’ field is not a name registered in the system (ie. not selected from the list of available names). 4. *The system responds by displaying a message to the user saying ‘Wolf Member being edited: \_\_\_\_\_\_’ from House: \_\_\_\_\_’. The field labeled ‘First Name’ is populated with the existing first name. The field ‘Last Name Initial’ is populated with the existing last name initial. The radio buttons labeled ‘Student’ and ‘Staff’ reflect the existing selection.* 5. The user may edit the Wolf Member’s first name in the input field labeled ‘First Name’. 6. The user may edit the Wolf Member’s last name initial in the input field labeled ‘Last Name Initial’. 7. The user may change the Wolf Member’s associated House using a dropdown list labeled ‘House’ with two of the following options (whichever are not the currently saved House):    1. ‘Arctic Wolf’    2. ‘Timber Wolf’    3. ‘Grey Wolf’ 8. The user may change the selection of the radio buttons labeled ‘Student’ and ‘Staff’. 9. Once the user has changed a field and all the required fields are valid, they press a button labeled ‘Save Changes’ to successfully save the changes to the Wolf Member. This button is unable to be clicked if the user has not completed all the required fields or has not changed any information. The Wolf Member changes cannot be saved if that exact name is already registered in the system. 10. *If the Wolf Member information is invalid, the system responds by saying ‘\_\_\_\_\_\_ is already a name in the Wolf House System’. The fields remain filled so the user may edit the information and attempt to add the Wolf Member again.* 11. *If the Wolf Member information is valid, the system responds by providing a confirmation message to the user saying ‘Changes were successfully saved’. The fields are cleared so the user may edit another Wolf Member.* 12. At any time, the user may press a button labeled ‘Back’ to go back to the main menu, without saving. |
| Exit Conditions | The member’s new information is saved. |



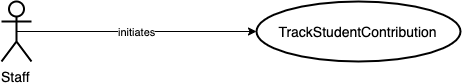
|  |  |
| --- | --- |
| Use case name | RemoveMember |
| Actors | Initiated by Admin |
| Entry Conditions | The user is logged into the Admin account and has selected the ‘Remove Wolf Member’ button. |
| Flow of Events | 1. The user must type the name of the Wolf Member into an input field labeled ‘Wolf Member Name’. 2. *The system responds by attempting to predict the rest of the name.* 3. The user selects the name from the available predictions. 4. *The system responds by displaying the Wolf House name associated with the selected Wolf Member in a field labeled ‘House’, and displaying the type of Member (student or staff) in a field labeled ‘Member Type’.* 5. Once a member is selected, the user presses a button labeled ‘Remove Member’. 6. *The system responds by displaying an alert to the user asking ‘Are you sure you want to remove the following member: \_\_\_\_\_?’ and displays two buttons labeled ‘Cancel’ and ‘Yes’.* 7. If the user selects ‘Cancel’, *the alert disappears and they return to the Remove Member form, with the fields filled.* 8. If the user selects ‘Yes’, *the alert disappears and they return to the Remove Member form, with the fields reset so they may remove another member. A message is displayed to the user saying ‘\_\_\_\_\_ was removed successfully’.* 9. At any time, the user may press a button labeled ‘Back’ to go back to the main menu, without saving. |
| Exit Conditions | The member is removed. |



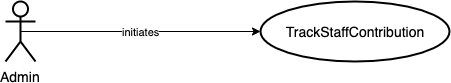
|  |  |
| --- | --- |
| Use case name | SetStaffLogin |
| Actors | Initiated by Admin |
| Entry Conditions | The user is logged into the Admin account and has selected the ‘Set Staff Login’ button. |
| Flow of Events | 1. The user enters the new password 2. *If the new password entered is identical to the current password, the system responds by clearing the password field and displaying a message to the user, ‘New password must be different than current password.’* 3. *If the new password is different than the current password, the system responds by displaying a message to the user, ‘The Staff account password has been updated.’* 4. At any time, the user may press a button labeled ‘Back’ to go back to the main menu, without saving. |
| Exit Conditions | The Staff account login information is updated. |



|  |  |
| --- | --- |
| Use case name | ResetSystem |
| Actors | Initiated by Admin |
| Entry Conditions | The user is logged into the Admin account. |
| Flow of Events | 1. The user selects a button labeled ‘Reset System’. 2. *The system responds by displaying an alert to the user asking ‘Are you sure you want to reset the Wolf House System? All house points will be reset to 0, and all member history will be cleared.’ and displays two buttons labeled ‘Cancel’ and ‘Yes’.* 3. If the user selects ‘Cancel’, *the alert disappears.* 4. If the user selects ‘Yes’, *the alert disappears. A message is displayed to the user saying ‘The Wolf House System has been reset successfully’. The system responds by removing all point data from* |
| Exit Conditions | The total of each house’s points is reset to zero, and all member history data is cleared. |



|  |  |
| --- | --- |
| Use case name | TrackStudentContribution |
| Actors | Initiated by Staff or Admin |
| Entry Conditions | The user is logged into the Staff or Admin account and has selected the ‘Track Student Contribution’ button. |
| Flow of Events | 1. The user views a list of all students in the system. By default, the students are listed in descending order of number of points earned. 2. The user may sort the students by number of points earned, either ascending or descending, by selecting the column labeled ‘points earned’ to toggle the sort order. 3. At any time, the user may press a button labeled ‘Back’ to go back to the main menu. |
| Exit Conditions | The user has been shown the list of students with points earned. |



|  |  |
| --- | --- |
| Use case name | TrackStaffContribution |
| Actors | Initiated by Admin |
| Entry Conditions | The user is logged into the Admin account and has selected the ‘Track Staff Contribution’ |
| Flow of Events | 1. The user views a list of all staff members in the system. By default, the staff members are listed in descending order of number of contributions entered (adding points to a house). 2. The user may sort the staff members by number of contributions entered, either ascending or descending, by selecting the column labeled ‘contributions’ to toggle the sort order. 3. The user may sort the staff members by number of points earned, either ascending or descending, by selecting the column labeled ‘Points Earned’ to toggle the sort order. 4. At any time, the user may press a button labeled ‘Back’ to go back to the main menu. |
| Exit Conditions | The user has been shown the list of staff members with points earned, and entries contributed. |

## Functionality

[This section describes the functional requirements of the system for those requirements that are expressed in the natural language style. For many applications, this may constitute the bulk of the SRS Package and thought should be given to the organization of this section. This section is typically organized by feature, but alternative organization methods, for example organization by user, or organization by subsystem may also be appropriate. Functional requirements may include: feature sets, capabilities and security.

Where application development tools (requirements tools, modeling tools, etc) are employed to capture the functionality, this section document will refer to the availability of that data and indicate the location and name of the tool which is used to capture the data.]

### <Password Protection>

[The requirement description.]

### <Password Protection>

[The requirement description.]

## Usability

[This section should include all of those requirements that affect usability. Examples:

1. Specify the required training time for a normal users and power users to become productive at particular operations.
2. Specify measurable task times for typical tasks, or
3. Base usability requirements of the new system on other systems that the users know and like.
4. Specify requirements to conform to common usability standards – e.g., IBM’s CUA standards, or the GUI standards published by Microsoft for Windows 95.]

### <Usability Requirement One>

The requirement description.

## Reliability

[Requirements for reliability of the system should be specified here. Suggestions:

1. Availability – specify % of time available ( xx.xx%), hours of use, maintenance access, degraded mode operations etc.
2. Mean Time Between Failures (MTBF) – this is usually specified in hours, but it could also be specified in terms of days, months, or years.
3. Mean Time To Repair (MTTR) – how long is the system allowed to be out of operation after it has failed?
4. Accuracy – specify precision (resolution) and accuracy (by some known standard) that is required in the systems output.
5. Maximum bugs or defect rate – usually expressed in terms of bugs/KLOC (thousands of lines of code), or bugs per function-point.
6. Bugs or defect rate – categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a “critical” bug (e.g., complete loss of data, complete inability to use certain parts of the functionality of the system).]

#### *<Reliability Requirement One*>

[The requirement description.]

## Performance

[The performance characteristics of the system should be outlined in this section. Include specific response times. Where applicable, reference related Use Cases by name.

1. Response time for a transaction (average, maximum)
2. Throughput (e.g., transactions per second)
3. Capacity (e.g., the number of customers or transactions the system can accommodate)
4. Degradation modes (what is the acceptable mode of operation when the system has been degraded in some manner)
5. Resource utilization: memory, disk, communications, etc.]

### <Performance Requirement One>

[The requirement description.]

## Supportability

[This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including coding standards, naming conventions, class libraries, maintenance access, maintenance utilities.]

### <Supportability Requirement One>

[The requirement description.]

## Design Constraints

[This section should indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, etc.]

### <Design Constraint One>

[The requirement description.]

## Online User Documentation and Help System Requirements

[Describes the requirements, if any, for on-line user documentation, help systems, help about notices, etc.]

## Purchased Components

[This section describes any purchased components to be used with the system, any applicable licensing or usage restrictions, and any associated compatibility/interoperability or interface standards.]

## Interfaces

[This section defines the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, etc, so that the software can be developed and verified against the interface requirements.]

### User Interfaces

[Describe the user interfaces that are to be implemented by the software.]

### Hardware Interfaces

[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, etc.]

### Software Interfaces

[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application, or components being developed for subsystems outside of the scope of this SRS, but with which this software application must interact.]

### Communications Interfaces

[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, etc.]

## Licensing Requirements

[Defines any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]

The use of the Wolf House Point System is restricted to Gabriela Braga and Sir James Whitney School for the Deaf. All future updates to this software must be done by Gabriela Braga or with the express written permission of Gabriela Braga.​

## Legal, Copyright and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notice, word mark, trademark, or logo compliance issues for the software.]

Gabriela Braga and Sir James Whitney School for the Deaf have all rights to the Wolf House Point System described by this document. This system may not be used by any other party unless given written consent by both Sir James Whitney School for the Deaf and Gabriela Braga.

## Applicable Standards

[This section describes by reference any applicable standards, (and the specific sections of any such standards that apply to the system being described). For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, etc.]