**Software Requirements Specification for**

**DePauw Sorry**

**Version 1.0**

Prepared by Aadhikya Ghimire, Alyssa Fisher, Emma LeMasters, Chris Hemsworth

DePauw University

**1  Introduction**

**1.1  Purpose**

The Sorry Game system will allow the simulation of a game of sorry with a variety of kinds of players and ways of moving around the board. The player will interact with the menu in the role of one of the games players

**1.2  Intended Audience and Reading Suggestions**

This document was developed both as a guideline for student developers, and also as a reference for potential simulated game customers.

**1.3  Project Scope**

The goal of Sorry is to support the simulation of a traditional game of Sorry with 4 players and a game board. Interaction will be through a menu displayed on the console. The implementation should be flexible enough that different types of cards can be both drawn and played.

**2  Overall Description**

**2.1  Product Perspective**

This software product is being developed by students of the Object-Oriented Software Development course at DePauw University and is intended for use by anyone wanting to simulate a game of Sorry. The goal of this project is to develop a feature-rich application which will serve as a functioning prototype for a more comprehensive application which could be developed by extending the codebase.

**2.2  Product Features**

The main features of this product are:

• managing the menu-oriented interface to the simulation

• maintaining the state of a number of players as players move their pieces around the board.

• providing a flexible design to support expansion or revision of the simulation

**2.3  User Classes and Characteristics**

The users will include those who simply wish to use the simulation, as well as those (the developers) who are designing and testing the simulation or extensions to the simulation.

**2.4  Operating Environment**

This application is designed to work with a Java Virtual Machine in a desktop environment. Users of this application are expected to be running either a Windows, MacOS, or Linux desktop operating system.

**2.5  Design and Implementation Constraints**

This application may use the filesystem as a means of saving bank state and simulation history.

**2.6  User Documentation**

A programmer’s guide to working with the software components developed as part of this application will be provided for those who wish to continue development on this application. Additionally, a user’s guide will be provided for those who wish to use this application to simulate Sorry transactions. Within the simulation, a limited amount of help information will also be available.

**3  System Features**

The following features, with their associated requirements, will be implemented in the final revision of this software system:

**3.1  Menu Interface**

**3.1.1  Description and Priority**

A user can interact with the simulation through a series of menus.

**3.1.2  Functional Requirements**

**REQ-1:** There will be a main menu from which the user can select common operations or navigate to more specialized menus.

**REQ-2**: The user will interact with the menu by clicking on icons around the board and typing in specific key words to the console

**REQ-3:** Any operations that need additional information will also communicate with the user through the console.

**REQ-4:** It should be easy to navigate through the menu system, and to get help from the system in doing so.

**3.2  Rules and Card Layouts**

**3.2.1  Description and Priority**

The system allows the user to play several variants of cards, which differ in characteristics and actions.

**3.2.2  Functional Requirements**

**REQ-1:** The system must keep track of rules and card layouts for several different players.

**REQ-2:** The system must deal cards into the appropriate layout when a new game is started.

**REQ-3:** The system should follow rules to know when cards may be played, and should give an appropriate error response when an illegal move is attempted.

**REQ-4:** The system should detect when the player has achieved an arrangement of pieces that wins the game.

**REQ-5:** The user should be able to view a summary of the rules.

**3.3  Starting and Ending a Game**

**3.3.1  Description and Priority**

The game has well-defined starting and ending states, and the user can track their progress toward completion.

**3.3.2  Functional Requirements**

**REQ-1:** When a new game is started, the player may choose which color they want to be.

**REQ-2:** The system should use a random number generator to determine some aspects of the initial setup; such as the type of cards each player starts with.

**REQ-3:** At any time during play, the player may start a new game, or quit the game system altogether.

**REQ-4:** When the player wins the current game, the system should provide an appropriate indication, then ask the player whether they want to start a new game or quit.

**3.4  Scores and Statistics**

**3.4.1  Description and Priority**

The system keeps track of the state of the game, including scores and statistics for previous games. A user may save all information about the current state of the game so that it may be resumed at another time.

**3.4.2  Functional Requirements**

**REQ-1:** The system should keep track of the player’s current positions on the board

**REQ-2:** The system should keep track of how many wins each player has had in the session

**REQ-3:** The system should keep track of the player’s game play statistics, including number of games started, games won, and games lost.

**REQ-4:** The system should display the current score while the game is being played, and other statistics on demand.

**4  External Interface Requirements**

**4.1  Hardware Interfaces**

The software will run on a desktop or laptop and no additional hardware is needed.

**4.2  Software Interfaces**

The game does not need to interface with any software other than the Java platform.

**5  Other Nonfunctional Requirements**

**5.1  Performance Requirements**

Management of the graphical interface and game state must consume minimal system resources so as to be accessible in real-time by users of the system. This application is intended to be used interactively, so users should not be expected to wait for the completion of any of the operations provided by the application.

**5.2  Security Requirements**

The application will not require any sensitive information from the user. It will rely on existing user-based security on the host operating system to keep saved game states private.

**5.3  Software Quality Attributes**

This application will ship with a suite of tests which insure its proper function, even if third-party updates to the source-code are integrated. Additionally, at run time, this application will verify the correctness of any data files it uses, or of any input provided by the user, and issue appropriate error messages in the cases of unexpected or erroneous input.