**Software Requirements Specification for**

**Timed Patience**

**Version 1.1**

Prepared by Brian Howard

DePauw University

Last Updated: 12.6.14 by Lauren Owensby

**1 Introduction**

**1.1 Purpose**

The Timed Patience system will allow players to play a variety of card-based solitaire games on the computer. Timed Patience will be flexible enough that it can be modified to play additional versions of solitaire with minimal extra programming.

**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 game designers and players.

**1.3 Project Scope**

The goal of Timed Patience is to support the playing of traditional card-based solitaire games, also known as “patience”. Interaction will be through a graphical user interface, where the player may move cards between piles to achieve an overall goal. The system will allow the player to choose from among several solitaire variations, each with a different layout of cards, end goal, and rules for which cards may be moved where. The implementation should be flexible enough that new variations could be added with relatively little programming effort.

**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 experience traditional card-based solitaire games. 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 graphical interface to the game: displaying cards in a layout and allowing the player to move them between piles

• maintaining game state (positions of cards, statistics such as number of moves) and enforcing game rules

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

**2.3 User Classes and Characteristics**

The users will include those who simple wish to play the game, as well as those (the developers) who are designing and testing the game or extensions to the game.

**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 game state and statistics.

**2.6 User Documentation**

Within the game, rules of play will 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 Graphical Interface**

**3.1.1 Description and Priority**

A user can interact with a visual representation of playing cards on the screen.

**3.1.2 Functional Requirements**

**REQ-1:** The game shall simulate one or more decks of playing cards, allowing them to be shuffled, dealt into piles, turned face-up or face-down, and shifted from pile to pile.

**REQ-2:** The user shall be able to move cards by direct manipulation on the screen, using the mouse or a touch screen.

**REQ-3:** The game will allow the player to undo and redo at least the most recent move.

**REQ-4:** The user shall be able to choose from a variety of screen backgrounds and card designs.

**REQ-5:** The game shall simulate a clock and record of user victory time at the start and finish, along with the score.

**3.2 Rules and Card Layouts**

**3.2.1 Description and Priority**

The system allows the user to play several variants of solitaire, which differ in where cards are placed, how they may be moved, and how the game is won.

**3.2.2 Functional Requirements**

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

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

**REQ-3:** The system should follows rules to know where cards may be moved, 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 cards that wins the game and record the time from the clock.

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

**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 variant to play.

**REQ-2:** The system should use a random number generator to determine some aspects of the initial setup; it may allow the player to choose an initial seed to select a particular instance of the game.

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

**REQ-4:** When the player wins the current game, the system should provide an appropriate indication.

**3.4 Scores and Statistics**

**3.4.1 Description and Priority**

A user can 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 score, which may depend on the number of moves made or the amount of time passed.

**REQ-2:** The system should keep track of the high scores for each solitaire variant.

**REQ-3:** The system should display the current score and time while the game is being played.

**REQ-4:** The system should keep track of the current time spent from the start of the game up until the user finishes the game and display that time at the conclusion of the game.

**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.