SE 3XA3: Software Requirements Specification
PyCards

Team 2,
Aravi Premachandran premaa
Michael Lee leemr2
Nikhil Patel patelna2

October 31, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
October 11, 2016	1.0	Initial Revision
October 30, 2016	1.1	Revision 1

1 Project Drivers

1.1 The Purpose of the Project

PyCards is a collection of solitaire (single-player) card games. It is designed to be a source of entertainment for the end user(s). The main objective for the product is to PyCards to be a user-friendly application that our users can use to pass the time, have fun, relax, and also challenge themselves.

1.2 The Stakeholders

1.2.1 The Client

Our client is Dr. Smith, a professor at McMaster University

1.2.2 The Customers

Our customer is any user who downloads and runs our program.

1.2.3 Other Stakeholders

Other stakeholders include the original developer of PySol, Markus F.X.J. Oberhumer, as well as the many developers and contributors PySolFC, the fork which we have based our program on. In addition to these, as the project is open source, the last group of stakeholders are future contributors and developers to this software project.

1.3 Mandated Restraints

Constraint 1

The source code of the application shall be implemented using the Python programming language, version 2.7.xx

Rationale 1

The product should preserve compatibility for existing users of the original implementation. The client should not be required to upgrade or install new software in order to run the application.

Fit Criterion

The product shall only require a Python interpreter (and/or the inclusion of additional Python packages) in order to operate. Exception will be made if either the entirety of the product is ported to another language or if the necessary runtimes and dependencies can be bundled with the product.

Constraint 2

The product shall be compatible with Windows operating systems (Windows 10), Mac OSX, and Ubuntu provided that Python version 2.7.xx is compatible with the target system.

Rationale 2

The client should not be required to migrate to newer (or different) software or hardware in order to utilize the product.

Fit Criterion

The product shall be tested on each of the specified systems by the developers to ensure that it operates as expected and is fully functional.

Constraint 3

The product shall be made available under the GNU GPLv3 or later, along with any non-permissive terms added in accord with section 7 of the GNU GPLv3.

Rationale 3

The original implementation was conveyed with this license and as per the conditions modified source versions must be licensed under the same license.

Fit Criterion

The product must adhere to the conditions of the GNU GPLv3 license.

Constraint 4

The source code for the product shall be publicly available but the end product must also be deliverable as an standalone executable or application for both Windows operating systems and Mac OSX operating system.

Rationale 4

The source code for the product must be publicly available as per the conditions of the license. Users should not be required to be familiar with the command-line/terminal in order to operate the product.

Fit Criterion

The final product should be available as a .exe executable for Windows-based systems and as an application for OSX systems. The code should be available in a publicly accessible repository.

1.4 Naming Conventions and Terminology

API (Application Program Interface): A set of routines, protocols, and tools for interacting with a program

GNU GPLv3: The GNU GPL (GNU General Public License)is a software license, focused on freedom and free software. Our software project is licensed under version 3 of this license.

More information on the terms and conditions of this license can be found at its website: https://en.wikipedia.org/wiki/GNU_General_Public_License
Free Software: Software distributed under terms that allow users to run the software for any purpose, as well as to study, change, and distribute the software and any adapted versions.
PySolFC (PySol Fan Club Edition): The program which our project is based off of.
PySol: The original program which PySolFC is based off of
Compatibility: The ability of the product to operate on a given **Klondike:** The most common solitaire game. Most people think of this variation when they think of solitaire, and it is the variation included in the solitaire program found in older versions of Windows (pre-windows 8). system. Full functionality is expected, unless otherwise specified, but performance, appearance, and other characteristics may vary
Implementation: The object code, binaries, and/or executable form of the product, along with the source code used to create it
Product: The entirety of the project, including but not limited to its source code, binaries, and documentation

1.5 Relevant Facts and Assumptions

Relevant Facts

- The existing implementation is implemented purely in the Python programming language.
- Mac OSX and Ubuntu come with a Python environment pre-installed. The installed version may or may not be compatible with the product.

Assumptions

- It is not feasible for the product to be tested for compatibility on all of the various operating systems. As such it will only be tested on the latest version of each major operating system as previously specified.
- The product assumes that the system that it will be operated on includes peripherals such as a mouse, keyboard and display.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

Our product shall be based upon the implementation PySol Fan Club Edition however it shall be redeveloped in order to satisfy the constraints and requirements defined in this document.

Table 2: **Work Partitioning**

Event Name	**Input/Output**	**Summary**
User changes game	Game ID(IN)	The user selects a different game. The system prompts for confirmation to discard the current game. The system destroys the current game and creates the new game
User changes background	File Path(IN) or Color(IN)	The user selects a new background image from the file system or chooses a solid color to use. The system redraws the window with the new color
User changes cardset	File Path(IN)	The user selects the new cardset to use. The system attempts to load the cardset and then redraws the game using the new cardset images
User selects a card	ButtonPress(IN)	The user clicks on a card. The system checks if the selection is valid and then highlights and tracks the cursor with the selected card(s)

2.1.2 Work Partitioning

2.1.3 Individual Product Use Cases

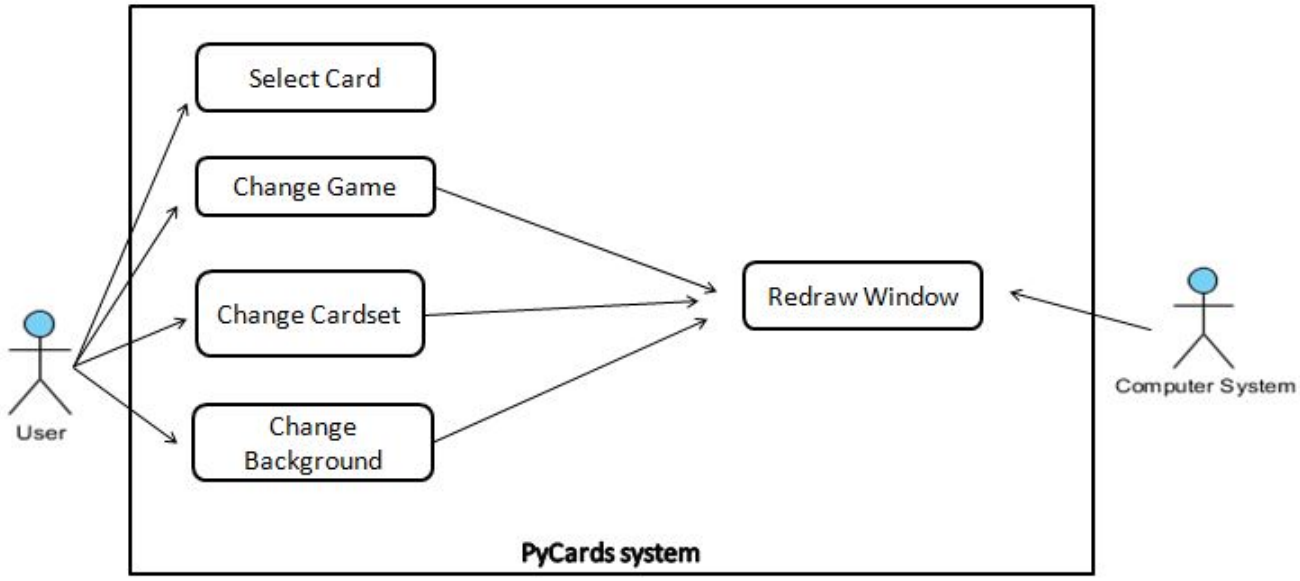

Figure 1: Use Case Diagram for PyCards

2.2 Functional Requirements

Functional Requirement 1

The program will be executed on a computer compatible with Python version 2.7.xx

Fit Criterion

The functionality of the program on the major operating systems - Windows 10, Ubuntu, Mac OSX - will be tested before release

Functional Requirement 2

The program will execute in its own graphical window

Fit Criterion

Check that the graphical user interface is successfully created and fully functional on the target system(s).

Functional Requirement 3

The users game statistics will persist even after the application is terminated

Fit Criterion

Upon re-launching the application the player statistics will correspond to the statistics present when the application was most recently terminated.

Functional Requirement 4

The program shall provide the user with the option to start a new game

Fit Criterion

The user interface should contain a button that when pressed will trigger the start of a new game.

Game Aspect Specific Requirements:

- The program shall provide users with the ability to start new games, save games, and load already saved games.
- The program shall provide users with an interface to customize the game window, namely the background and cardset to be used.
- The program shall adhere to the rules of the game in progress
- Upon completion of a game, the user shall be prompted to start a new game

Klondike Requirements:

- The playing cards are organized into NUM_DECKS deck stack, NUM_WASTES waste stack, NUM_FOUNDATIONS foundation stacks and NUM_NORMAL_STACKS alternating-colour stacks
- Cards cannot be added to the deck or waste stack(s) by the player
- The waste stack(s) can be recycled to the deck stack NUM_REDEAL times
- If the player clicks on the deck stack, the top BURN_NUM cards from the deck stack will be moved to the waste stack
- The player can only select and move the current top card from the waste stack
- The foundation stacks are initially empty.
- The player can build the foundation stacks by placing on it cards of the same suit in order of increasing rank (FOUNDATION_BASE to King). These cards must be placed one at a time.
- If the player successfully builds all four foundation stacks (to King) then (s)he wins the game and a dialogue is shown confirming the successful completion of the game.
- The player can build the alternating-suit stacks by forming stacks of cards of opposite colours and decreasing (by 1) rank
- Multiple cards in the same stack can be moved at the same time provided that sequence of cards to be moved is alternating in colour and the rank of each subsequent card is one lower than that of the previous card (with the exception of the first card at the bottom of the stack of selected cards which has no previous)
- If an alternating-suit stack is empty, only sequences starting with STACK_BASE can be moved to it
- If as a result of moving cards, the top card of a stack is face-down, that card is then flipped over to become face-up

3 Non-functional Requirements

3.1 Look and Feel Requirements

- The product shall represent cards with images of playing cards and additional areas that can be interacted with (ex. where cards can be placed) shall be outlined for the user to see.

3.2 Usability and Humanity Requirements

- The product shall be easy for a child with basic reading and computer abilities to use

3.3 Performance Requirements

- Normal interaction actions with the game shall take no longer than if an average user were playing the same game with a physical deck of cards.

3.4 Operational and Environmental Requirements

- Users will interact with the product using a mouse and keyboard connected to their computer

3.5 Installability Requirements

- The game will not need to be installed to the user's device; instead it will be packaged and self-contained, runnable as a portable application.

3.6 Maintainability and Support Requirements

- The product is expected to run on Windows 10, Mac OSX, and Ubuntu systems that have Python version 2.7.xx installed.

3.7 Security Requirements

- The source code for the product is publicly available and as such anyone can download, modify, and produce modified version of the project. However, only the developer team is allowed to directly make changes to the repository
- When the product is made ready for distribution, the packaged product shall have a checksum (ie. MD5 checksum) for verifying authenticity

3.8 Cultural Requirements

- The product will not contain images or text that would be considered offensive.

3.9 Legal Requirements

- The product is licensed under the GPLv3 license and must conform to the license and any non-permissive terms added in accordance to section 7 of the GNU GPLv3 license.

3.10 Health and Safety Requirements

- This product requires the use of a pointing device and that the user views and interacts with a graphical user interface. Excessive use of this product may cause strain or injury including but not limited to eye strain and injury to arms, wrists, and/or hands including carpal tunnel syndrome.

4 Project Issues

4.1 Open Issues

- The main open issue is the possibility of the graphics extension provided by the open source project being incompatible. Seeing as the team has permission to use and modify the existing implementation, many of the issues a new project would face have been accounted for
- Python includes a lot of built-in libraries that will be used in the implementation. One such library is Tkinter, used for the graphical portion of the product
- The existing implementation PySolFC is an off-the-shelf solution, and so is the project that PySolFC was based upon, the original PySol

4.2 New Problems

- Any changes to the original Tkinter graphics library or the custom interfaces that PySolFC provides to the library could make it incompatible with our reimplementation and would thusly affect the work of our developer team. It would require the adaptation of our code to the new API which could introduce problems
- The program should not require a lot of RAM when its running or have high disk usage or requirements of the users system. Overall the project should not have overly high resource consumption on the target system

4.3 Tasks

The team plan on completing this project by completing the problem statement and requirements documents before beginning programming. With these documents the team will have a strong understanding of what needs to done. From then onwards developers will work on implementing the program using the Model View Controller software architecture and referring to the existing project if and when needed. To manage the project well be using

version control via git.

4.4 Migration to the New Product

There are no major requirements for migration as we intend to distribute the program as a standalone executable (while making the source available to interested parties). As such the product will be able to coexist with the existing products PySol and PySolFC. Furthermore, packaging or bundling as an executable (or application) allows the product to be self-contained with minimal external dependencies.

The existing implementation however, if installed, adds its modules to the python directory for 3rd party packages. Facilitating its removal would require us to reverse that modification and by removing or updating the specified files

4.5 Risks

- Excessive pressure due to time constraints for deliverables
- Inaccurate quantification of objectives
- Inadequate testing of graphical user interface

4.6 Costs

The product being developed is open-source as is the implementation it is based upon. To follow and propagate the concept of open-source software, only freely available software shall be used in the development process.

The primary and only significant cost of the project to date is the time and effort spent by its developers. The cost is anticipated to be around 5 hours or more of time spent per person per week for the duration of September-December 2016.

4.7 User Documentation and Training

In-person training will not be required and therefore will not be provided for this product. However, user documentation will be included with the packaged product. A specifications document including APIs, modular decomposition, and also build and modification instructions shall accompany the source for the product.

The user documentation shall provide enough guidance that a user with no previous exposure to the product shall after reading be able to achieve basic objectives. These objectives include but are not limited to navigating the interface, starting a new game, and accessing the demo feature.

The user documentation shall include a section (or multiple sections) that display the rules of the available games.

4.8 Waiting Room

Some possible requirements we could address if time permits:

- Create more games (version 1.1)
  By creating more games users will have more options and be less likely to become bored or disinterested
- Improve the graphics (version 1.1)
  By improving the graphics, the program will avoid the risk of seeming outdated or becoming obsolete
- Create a leaderboard (version 1.2)
  By creating a leaderboard, it will keep people more engaged and bring out their competitiveness which will motivate them to use the product more

4.9 Ideas for Solutions

5 Appendix

5.1 Symbolic Parameters

- NUM_REDEALS = Unlimited
- FOUNDATION_BASE = Ace
- STACK_BASE = King
- NUM_DECKS = 1
- NUM_WASTES = 1
- NUM_FOUNDATIONS = NUM_SUITS = 4
- NUM_NORMAL_STACKS = 7