SOFTWARE REQUIREMENTS SPECIFICATION

for

Layers Implementation in Pyaint

Version 1.0 approved

Prepared by Group 8:
Subata Naveen Khan - 18119
Muhammad Usman Adnan - 23012
Danish Babar Qureshi - 22890
Saad Riazuddin Mahmood - 22803
Khansa Junaid - 23074

Section 6675

October 9, 2022

Contents

Revision History

Name	Date	Reason For Changes	Version
	9-10-2022		1.0

1 Introduction

1.1 Purpose

The purpose of this document is to provide a detailed overview of our implementation of layers in Pyaint. It includes the overall description covering the product perspective, functions, operating environment, user classes and characteristics, as well as design implementation and constraints. It also covers the system features relating to the layers component. It lays down the foundation of how we will execute the aforementioned aspect and lists down all the functional and non-functional requirements the user can expect.

1.2 Document Conventions

The font used is Times New Roman with size 12, which is a constant for official reports. Every requirement statement is to have its own priority.

1.3 Intended Audience and Reading Suggestions

The Target Audience is:

- Developers
- Program testers
- Documentation Writers
- End users

Reading suggestions can vary depending on who the reader is. To acquire an optimal level of understanding about the program and the component inclusion, a sequential reading of this document is suggested. This would be best for someone who is not well acquainted with Pyaint.

For someone who is looking for specifics and has some background knowledge of the program, sectional reading can be an efficient alternative (especially if they are looking for something in particular.

For a general understanding of the document and the program itself, view Overall Description (Section 2). For a detailed description of the program features, functional requirements, use case tables, and use case modeling see System Features (Section 4).

Technical standards to which the team will hold the project are laid out in Other Nonfunctional Requirements (Section 5).

1.4 Project Scope

The goal of our component addition is for the users to be able to apply the features of the layers module. This would allow them to play around with the different layers on Pyaint. The users would be able to add layers, remove them, bring them forward (ahead of another layer), move them backward (behind another layer), make them invisible and visible, and merge layers together. This would allow them to be more creative with their work. Essentially, artists of all skill levels can use this tool to make artwork more easily. This would give them more creative freedom over their output. The "Select" and "Click" based design endorses a user-friendly GUI. This is an offline program which, at the moment, is only available on desktops.

1.5 References

- GitHub. 2022. Pyaint/Software-Engineering-Modules at wiki · umairazfar/Pyaint. [online] Available at:

  https://github.com/umairazfar/Pyaint/wiki/Software-Engineering-Modules

- undefined [Lucid Software]. (2018, February 7). UML Use Case Diagram Tutorial [Video]. YouTube. Retrieved 9 October 2022 [online], Available at:

  https://www.youtube.com/watch?v=zid-MVo7M-E

- Miles, R., & Hamilton, K. (2006, May 16). Learning UML 2.0: A Pragmatic Introduction to UML (1st ed.). O'Reilly Media. [offline]

- Software Engineering (9th Edition) 9th (ninth) edition (authors) Sommerville, Ian (2010) published by Addison Wesley [Hardcover]. (n.d.). Addison Wesley. [offline]

2 Overall Description

2.1 Product Perspective

Pyaint is an offline platform that facilitates users in digital painting. Since its conception many features have been added to it, and now we are adding layers. Layers help to control parts of your painting. Layers give the user a better grip on their artwork and ease the mass transformations and make composition easier than before.

Consider a collage or piece of art produced from various paper stacks. Some sheets have cuts that reveal the paper beneath them, while others conceal it. If you wish to change a component, you can edit the sheet containing just that component rather than the entire artwork. For this reason, we use layers.

2.2 Product Functions

The perspective is to let users use layers function in their drawings. Individual layers can be edited. The main functionalities are:

- User can add a layer.
- User can remove selected layer.
- User can move selected layer up.
- User can move selected layer down.
- User can make selected layer invisible.
- User can make selected layer visible.
- User can merge two layers together.

2.3 User Classes and Characteristics

This system contains one type of user class and that is the user themself.

Drawing programs like this are used by students, graphic designers, animators, or digital marketers.

The features available to the users will be that after installing Pyaint they can easily work on it, there is no register or sign in option. They can add a picture add layers and work on each layer separately. By adding layers, they can only edit the part of the picture they want to edit instead of editing whole picture. They can work on different layers separately and can then merge them into single layers.

2.4 Operating Environment

Minimum Hardware Requirements:

- Microsoft Windows 8 or higher
- x86 processor
- 4GB RAM for Windows
- CPU: Intel Core 2 Quad Q9000

Software Requirements:

- Python 3.8
- Pygame

2.5 Design and Implementation Constraints

- No Administrator, so there is no check and balance in regards to who has access to the work.
- Only one window can open at a time in terms of GUI
- This is a short-term project so there may not be support available from the developers for the application
- Python must be installed in the system for the application to run

2.6 User Documentation

A user manual detailing all the new layers-related features will be included in the update.

2.7 Assumptions and Dependencies

This project is dependent on Python 3.8 and Pygame.

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

4.1 Add a new layer

Priority	High	
Related Requirements	REQ-1: One layer is added by default at startup of the software. REQ-2: A button to add layers.	
Goal in context	User opens a new layer.	
Preconditions	None	
Successful end condition	A new layer is opened.	
Failed end condition	A new layer is not opened.	
Primary Actors	User	
Trigger	Click the ”New Layer” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	User clicks the ”New Layer” button.
	2	A new layer is created.

4.2 Remove a layer

Priority	High	
Related Requirements	REQ-1: A button to delete layers. REQ-2: A layer must be selected first. REQ-3: If there is only a single layer in the panel, a new empty layer is created when the delete button is clicked.	
Goal in context	User removes the selected layer.	
Preconditions	User has selected a layer.	
Successful end condition	Selected layer is removed.	
Failed end condition	Selected layer is not removed.	
Primary Actors	User	
Trigger	Click the ”Remove Layer” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	User selects a layer from the layer panel.
	2	User clicks the ”Remove Layer” button.
	3	The selected layer is deleted.

4.3 Make a layer invisible

Priority	Medium	
Related Requirements	REQ-1: A ”Visibility” button to toggle between making the layer visible and invisible. REQ-2: Only a visible layer can be made invisible. REQ-3: If there is only one layer, it cannot be made invisible.	
Goal in context	Selected layer becomes invisible.	
Preconditions	A visible layer is selected.	
Successful end condition	Selected layer becomes invisible.	
Failed end condition	Selected layer does not become invisible.	
Primary Actors	User	
Trigger	Click on ”Visibility” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	User selects a visible layer.
	2	User clicks on ”Visibility” button.
	3	Selected layer becomes invisible.

4.4 Make a layer visible

Priority	High	
Related Requirements	REQ-1: A ”Visibility” button. REQ-2: Only an invisible layer can be made visible. REQ-3: If there is only one layer, it cannot be invisible.	
Goal in context	Selected layer becomes visible.	
Preconditions	An invisible layer is selected.	
Successful end condition	Selected layer becomes visible.	
Failed end condition	Selected layer does not become visible.	
Primary Actors	User	
Trigger	Click on ”Visibility” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	1. User clicks the ‘visibility’ toggle button of an invisible layer.
	2	2. The selected layer toggles back to being visible.

4.5 Move up one layer

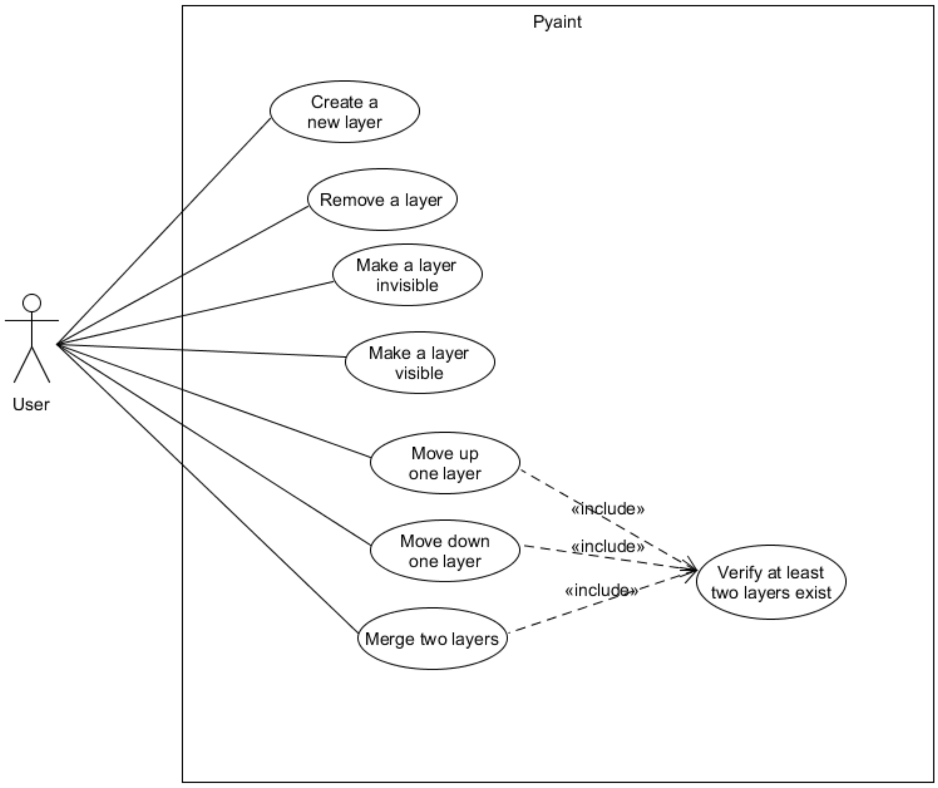
Priority	Medium	
Related Requirements	REQ-1: Up buttons for each layer. REQ-2: If user wants to move a layer up, there must be another layer present above it.	
Goal in context	User moves the selected layer up by one.	
Preconditions	More than one layer exists and one layer is selected.	
Successful end condition	Selected layer moves up.	
Failed end condition	Selected layer does not move up.	
Primary Actors	User	
Trigger	Click the ”Move Layer Up” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	User clicks ”Move Layer Up” button.
	2	At least two layers requirement is checked.
	3	Selected layer is moved up.

4.6 Move down one layer

Priority	Medium	
Related Requirements	REQ-1: Down buttons for each layer. REQ-2: If user wants to move a layer down, there must be another layer present below it.	
Goal in context	User moves the selected layer down by one.	
Preconditions	More than one layer exists and one layer is selected.	
Successful end condition	Selected layer moves down.	
Failed end condition	Selected layer does not move down.	
Primary Actors	User	
Trigger	Click the ”Move Layer Down” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	User clicks ”Move Layer Down” button.
	2	At least two layers requirement is checked.
	3	Selected layer is moved down.

4.7 Merge two layers

Priority	Low	
Related Requirements	REQ-1: There must be more than one layer. REQ-2: A button to select each layer. REQ-3: A button to merge layers.	
Goal in context	User merges selected layers.	
Preconditions	User selects two layers.	
Successful end condition	Selected layers are merged.	
Failed end condition	Selected layers are not merged.	
Primary Actors	User	
Trigger	Click on ”Merge Layers” button.	
Included cases	None	
Main flow	**Step**	**Action**
	1	User clicks on ”Merge Layer” button.
	2	At least two layers requirement is checked.
	2	Selected layers are merged.



5 Other Nonfunctional Requirements

5.1 Performance Requirements

The system must be interactive and have minimal delays. Showing the combined view of the layers should be fast to avoid lagging. Adding/deleting layers, merging layers, and toggling the visibility functions should have checks that raise errors if basic conditions are violated

5.2 Safety Requirements

The following safety requirements need to be paid attention to:

1. The software should not be too hardware intensive and should be able to run comfortably on even low-end machines without causing significant damage to the parts of the machine.

2. The users' device temperature must not exceed too much while using the software.

3. While the software is running, it should not cause disruption to system processes.

5.3 Security Requirements

The software fulfils a simple requirement for painting so it does not require any security implementation.

5.4 Software Quality Attributes

The app should be simple and easy to use. The buttons on the app's interface must be straightforward, intuitive, and visible.

5.5 Business Rules

This app allows everyone to access all functions available as they don't need to be kept restricted.

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>