

SE 3XA3: SRS
Gifitti

Team #2, Gifitti
Nicolai Kozel kozeln
Pavle Arezina arezinp
Riley McGee mcgeer

December 8, 2016

Contents

1	Project Drivers	3
1.1	The Purpose of the Project	3
1.2	The Stakeholders	3
1.2.1	The Client	3
1.2.2	The Customers	3
1.2.3	Other Stakeholders	3
1.3	Mandated Constraints	3
1.4	Naming Conventions and Terminology	4
1.5	Relevant Facts and Assumptions	4
2	Functional Requirements	4
2.1	The Scope of the Work and the Product	4
2.1.1	The Context of the Work	5
2.1.2	Work Partitioning	5
2.1.3	Individual Product Use Cases	5
2.2	Functional Requirements	5
3	Non-functional Requirements	10
3.1	Look and Feel Requirements	10
3.2	Usability and Humanity Requirements	11
3.3	Performance Requirements	11
3.3.1	Speed	12
3.3.2	Precision	12
3.3.3	Reliability/Availability	12
3.3.4	Capacity	12
3.3.5	Safety Critical	12
3.4	Operational and Environmental Requirements	13
3.4.1	Expected Physical	13
3.4.2	Expected Technical	13
3.4.3	Partner Applications	13
3.5	Maintainability and Support Requirements	13
3.6	Security Requirements	13
3.7	Cultural Requirements	14
3.8	Legal Requirements	14
3.9	Health and Safety Requirements	14

4	Project Issues	14
4.1	Open Issues	14
4.2	Off-the-Shelf Solutions	14
4.3	New Problems	15
4.4	Tasks	15
4.4.1	Planning of Development Phases	15
4.5	Migration to the New Product	16
4.5.1	Developers	17
4.5.2	Users	17
4.6	Risks	17
4.6.1	Developer Based	17
4.6.2	User Based	18
4.7	Costs	18
4.8	User Documentation and Training	18
4.9	Waiting Room	18
4.10	Ideas for Solutions	19
5	Appendix	20
5.1	Research Articles	20
5.2	Symbolic Parameters	20

List of Tables

1	Revision History	2
----------	-----------------------------------	----------

List of Figures

1	UML Use Case Diagram	6
2	Example of UI design	11
3	Example of User Guide	19

This document describes the requirements for 'Gifitti', gif viewer and frame extractor. The template for the Software Requirements Specification (SRS) is a subset of the Volere template (?).

Table 1: **Revision History**

Date	Version	Notes
October 4	1.0	Adding non-functional and functional requirements.
October 4	1.1	Added section 2.*
October 5	1.2	Adding Project Drivers and Issues
October 6	1.3	Proof read and added Ideas for Solutions
October 6	1.4	Adding information to appendix.
October 6	1.5	Finished Section 4.4-4.6
November 30	2.0	Revised for Rev 1

1 Project Drivers

1.1 The Purpose of the Project

The creation and manipulation of GIFs has always been a complicated process and has prevented the general public from participating in the creation of GIFs. Giffiti removes the barrier of entry to the creation and manipulation of GIFs.

1.2 The Stakeholders

Stakeholders are people, groups, organizations that might be affected by the outcome of a project. They have direct influence on how the project will be developed.

1.2.1 The Client

The client for this product is Professor Spencer Smith.

1.2.2 The Customers

The general public requires a tool to manipulate GIFs, both for personal use, or for publishing to social media such as Facebook or Reddit. This set of stakeholders is composed of individuals with a wide variety of ages and technical skills.

1.2.3 Other Stakeholders

Another group of stakeholders are graphic designers who are skilled in manipulating GIFs but want to create a product quickly, can utilize this tool to produce high quality GIFs.

1.3 Mandated Constraints

The only constraint to the project is that Giffiti does not implement the feature of taking screen shots of the user's computer. This constraint was imposed by the client, Spencer Smith. The only other constraint worth mentioning is that the cost should be \$0. Otherwise, the project is not

constrained in any manner. It has been left up to the developers to create Giffiti in any way they choose.

1.4 Naming Conventions and Terminology

1. GIF-Graphic Interchange Format; A file type similar to a video (without audio) and high compression.
2. Sprite Spreadsheet- A single image that contains all the frames of a simple animation.
3. Framerate - The speed the GIF is played at in the playback window.
4. Windows - Common desktop and laptop operating system. Version 7, 8, and 10.
5. Playback - The window that plays the GIF the user loaded into the program.
6. Manipulation - Any changes made to the GIF are considered manipulations. This includes framerate, length, and any other changes.

1.5 Relevant Facts and Assumptions

There are several assumptions and facts that are necessary for this project to be completed. For example, Giffiti will only run on the Windows operating system since it is being developed through C#. Additionally, the only devices able to run Giffiti will be computers, laptops, and ultrabooks. The users of this project can be of any age as long as they understand basic computer tasks such as downloading a file, opening a file in a program, and can recognize different file formats.

2 Functional Requirements

2.1 The Scope of the Work and the Product

Whenever a project is decided upon, there are boundaries that need to be defined to deliver a definite solution to the problem proposed. Background information must be given to help identify what the developers will need to produce a successful project.

2.1.1 The Context of the Work

The context of the application is that it will be executed on a windows machine, the machine itself cannot be harmed by execution of the application. The developers must be able to understand image manipulation on a software level to tackle many of the problems at hand.

2.1.2 Work Partitioning

Event List		
Event Name	Input and Output	Summary
User Selects the Load Button	User mouse click (in)	This will allow user to load in the GIF.
User Adjusts the play-back slider	Playback value (in)	This will change the how fast the GIF is playing.
User changes the frame numbers	User inputted values(in)	This will change which frames the GIF will play.
User Selects the export frames option	User mouse click (in)	This will allow the user to grab which frames they want.
User Selects the resize option	User mouse click (in)	This will allow the user to change the size of the GIF.

2.1.3 Individual Product Use Cases

The following image is the Use Case representation of Gifitti using UML

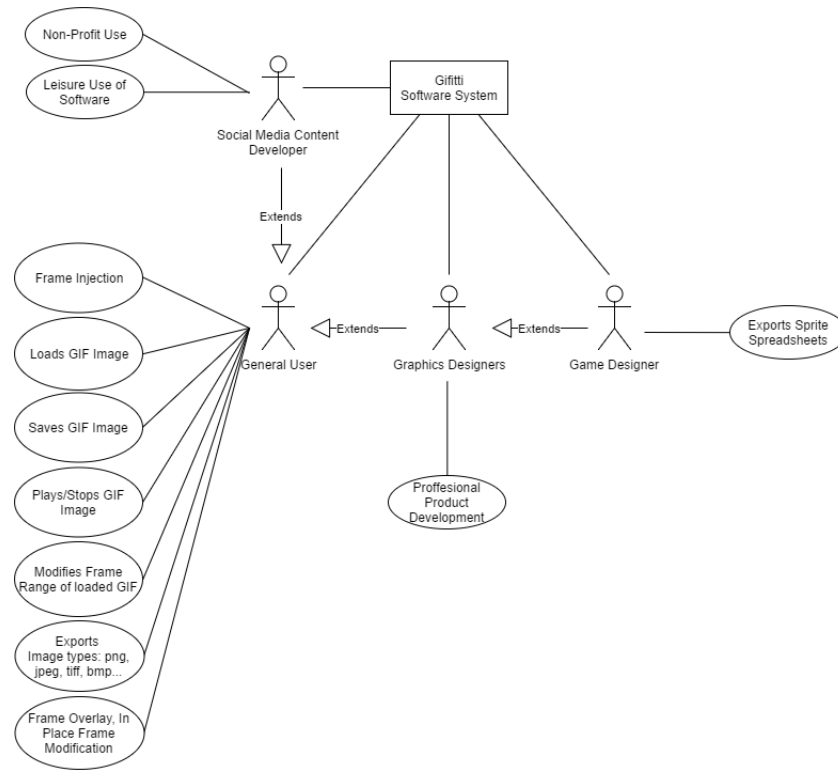


Figure 1: UML Use Case Diagram

2.2 Functional Requirements

1. The user is able to open a GIF from a specified location.

Rationale: Allow a user to modify a GIF or pull frames from it.

Fit Criterion: When GIF is selected to load, the program will display the selected GIF.

2. The user is able to save a GIF to a specified location.

Rationale: Give user control over where to save their finished product.

Fit Criterion: After saving the GIF in the program, at the specified location there will be a newly created GIF.

3. The user can specify the saved name of the GIF.

Rationale: Allow user full control over how they want their GIF to appear to the file system.

Fit Criterion: The user selected name will appear as the name of the GIF.

4. A Command must exist to allow the GIF to be played, this command only works if the GIF is stopped.

Rationale: Allow the user to be able to resume the animation of the GIF to inspect it.

Fit Criterion: When the play button is pressed, the GIF should run properly.

5. A Command must exist to allow the GIF to be stopped, this command only works if the GIF is playing.

Rationale: Allow the user to be able to stop the animation to inspect specific frame.

Fit Criterion: When the stop button is pressed, the GIF freezes on frame.

6. The user is given control over the stop and start commands.

Rationale: Only the user should be able to control of the animation.

Fit Criterion: The start and stop of the animation is only done on user input.

7. A range of frames may be specified and extracted as another GIF image.

Rationale: User may only want to have that specific set of frames from the original GIF.

Fit Criterion: Only the specified frames are shown in the program.

8. Any GIF or specified range of frames in a GIF can be exported as a series of frames.

Rationale: User may only want specific set of frames to be exported.

Fit Criterion: When the frames are exported, only the specified ones are there.

9. Series of frames may be PNG, JPEG, BMP, TIFF, or any other standard image format.

Rationale: User might want the exported frames for graphical art and may want it in different file types.

Fit Criterion: Images of the gif in the selected file format are located in the file the user saved them in.

10. The system must ensure the all files read in is of a proper format.

Rationale: To ensure that the system is getting the correct input to perform its tasks properly.

Fit Criterion: Only GIF files can be accepted as input for the system.

11. The system can set the playback speed of the loaded GIF.

Rationale: Allow the user creative control of the speed the animation is playing.

Fit Criterion: As the playback speed is adjusted, the speed of the animation is also changed.

12. The system must be integrated with a help context available to the user.

Rationale: Allow a way for the user to teach themselves how to use

the system.

Fit Criterion: User is able to access the help section and understand it.

13. Frame modification should be able to be done in place on the application.

Rationale: Any frame modification to the system is done should be reflected on the gif shown in the program.

Fit Criterion: Any frame adjustments are visible to the user before they save the GIF.

14. Frame modification allows users to resize the GIF.

Rationale: Give the user the choice to make a smaller or bigger version of the GIF for their needs.

Fit Criterion: The resized GIF should have the dimensions given by the user when saved as a GIF file format.

15. Error handling will catch any incorrect input from the user.

Rationale: User should be able to put in any input and the program should be able to prevent errors from crashing the system.

Fit Criterion: The program should be able to prevent negative values from crashing the program.

3 Non-functional Requirements

3.1 Look and Feel Requirements

1. The application will not have any background music to prevent from detracting from a pleasant user experience.
2. In the event of a user error, such as importing an invalid file type, a message box will appear to help indicate an error.

3. The form window should have a large enough display (relative to the screen and resolution it is being displayed on) such that any imported gif is viewable without squinting.
4. The gif playback should be at the maximum frame rate encoded in the gif so that it appears as a smooth playback.
5. The majority of the UI elements and buttons should be responsive and single click to ensure a user does not become frustrated with the program.
6. The design of the UI should extend on the current design of 'Gif Viewer'

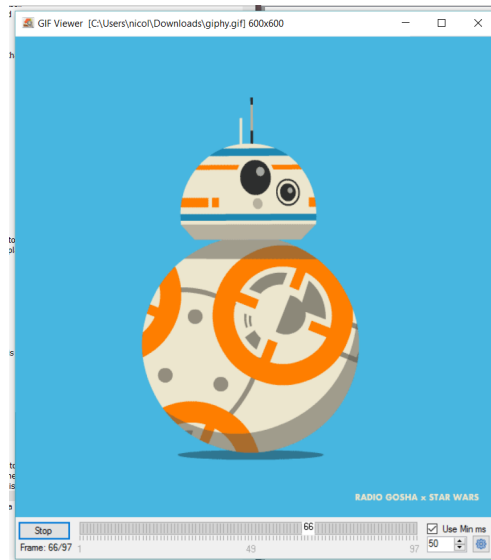


Figure 2: Example of UI design

3.2 Usability and Humanity Requirements

1. The program should be easy to use by people older than MIN_USE_AGE years old. This can be verified by seeing if a group of test users can manage to export a frame from a gif given simple oral or visual instructions.
2. Experienced users should be able to navigate the program's UI and export frames within under a MIN_USE_TIME.
3. The program should be designed to force a linear workflow to export GIFS. (I.e Load file, manipulate file, save file)

3.3 Performance Requirements

One of the main non-functional requirements is the performance of the system. The performance can be broken down into subsections to define more accurately what a good performance of the system should look like.

3.3.1 Speed

1. When exporting a reasonable number of frames (around 50-100), the operation should take no longer than MAX_EXPORT_TIME seconds. This can be verified through timing from the start of exporting to when the needed files have been exported.
2. All other UI elements should load within MAX_UI_LOAD seconds (or at least as fast as the current competitor Program 'Gif Viewer' benchmarked on system with Intel i5 3GHz, GTX 940, 8 GB RAM)

3.3.2 Precision

1. The application should export only and exactly the frames from the GIF that the user specifies.
2. The normal (or initial) play speed of a GIF when loaded shall be determined from the encoded information within the GIF.

3.3.3 Reliability/Availability

1. The program should be available 24/7, 365 days a year (or 366) because it does not rely on a server or internet connection.
2. Normal operation of the program, such as trying to import an invalid file type, should not cause it to crash or exit.

3.3.4 Capacity

1. The product only needs to be able to accommodate a single user at a time since it is run and hosted on each user's local machine.

3.3.5 Safety Critical

1. When saving frames from a GIF, these saved files should not overwrite existing files without prompting the user first.
2. Additionally, the remaining disk space must be checked before saving the frames to ensure we do not run out of room while saving.
3. In the event there is not enough room, the user shall be asked to choose a different location or free up memory space.

3.4 Operational and Environmental Requirements

External forces on the system need to be also be taken into consideration when creating a project.

3.4.1 Expected Physical

1. The product is expected to be used by a single person sitting down at a desktop or on a laptop in a climate controlled building.

3.4.2 Expected Technical

1. The software is expected to run on a desktop or laptop computer running Windows 7 or higher. Linux and Mac versions will not be available until '.NET' applications are ported to these OS.

3.4.3 Partner Applications

1. The product will utilize some third party extension to enable the ability to work with GIF's within C#. This extension will be decided upon before coding begins. The proposed solution is the same extension that the competitor, 'Gif Viewer' currently uses.

3.5 Maintainability and Support Requirements

1. Maintenance of this product will be provided through new versions (manually downloaded) that users will have to download and re-install to gain access to new features and bug fixes.
2. Support for this product will be provided to users via a FAQ section in a help menu and through a help email that will be set up once the program is finished.

3.6 Security Requirements

1. There are no security requirements for this program because there are no security issues with this application.

3.7 Cultural Requirements

1. The program shall not display or use any vulgar or obscene text, images, or media that will offend those in the countries that download it. However, this does not include content loaded by the user (I.e the GIFs they are trying to manipulate). The GIFs loaded by the user should not be filtered in any way. Any offensive images displayed through the program are the user's own doing.

3.8 Legal Requirements

1. Since the software is a redesign of 'Gif Viewer', it must comply with all [GPLv3.0 license conditions](#).

3.9 Health and Safety Requirements

1. There are no health and safety requirements for this program because there are no issues that apply to this application.

4 Project Issues

4.1 Open Issues

We are currently unsure on what third party API will be utilized to manipulate the GIFs in a more advanced method. Furthermore, the group is unsure of all the file types Giffiti will support in the final iteration of the project. This decision is dependent on the difficulty of implementing the tools necessary to convert the files from one form to the other. The final design of the user interface and how the features will be displayed to the user, still needs to be decided. A meeting should be held by the end of October 2016 to review possible design ideas.

4.2 Off-the-Shelf Solutions

There are multiple GIF editors available to the general public. Several editors are online versions that can be accessed through web browsers while others include software tools that encompass GIF editing such as ShareX. Photoshop

is another image manipulator that can be utilized in editing GIFS. Due to the multiple software tools that allow GIF manipulation, there are components that can be referenced to aid in the creation of Giffiti. Certain ideas from the online GIF editors can be incorporated into Giffiti to give the same functionality as the online editors but with a more streamlined experience.

4.3 New Problems

The implementation of Giffiti will be self contained and will not create any problems in the environment that it will be implemented in. Existing stakeholders who regularly use GIF editors will not experience any adverse effect by Giffiti. However, those who are very experienced and often create complex manipulations of GIFs will respond negatively to the product due to the less complex tools available. Currently, Giffiti will only be implemented in the Windows operating system and will only be supported on computers, not mobile devices.

4.4 Tasks

When developing a project, the content that will be developed for each section of a projects developmental lifecycle needs to be defined to allow proper deadlines to be created and fulfilled.

4.4.1 Planning of Development Phases

Phase 1 Architecture Construction:

This phase requires the software definition of the top level system layout. Controllers, and Major view elements link models. API not specified. This allows all system components to be worked on independent of others, parallelism of the project begins.

No requirements are satisfied at the end of this phase, it is the initial set up to allow for all requirements to be satisfied by the system.

Phase 2 Project demo:

The proof of concept for the system should be completed, this includes file reading and viewing. Mock view should be shown, not all view elements are required for this phase.

Requirements Satisfied by this phase:

- 2.2.1, 2.2.2, 2.2.3, 2.2.4, 3.1.1

Phase 3 Recompletion of old system:

As the old system is a base level for what is needed for the new system, all functionality of the old system should belong to the new system upon phase completion.

Requirements Satisfied by this phase:

- 2.2.6, 2.2.7, 2.2.8, 2.2.9, 2.2.10, 2.2.11, 2.2.13, 3.3.4, 3.3.5, 3.3.6, 3.3.7

Phase 4 Addition of frame injection and sprite spreadsheet exports:

These two tasks are to be done in parallel and mark a new revision of the software, the system may be released at this time, however there are still more features to complete. Error handling must be completed at this phase.

Requirements Satisfied by this phase:

- 2.2.5, 2.2.12, 2.2.14, 2.2.16, 2.2.17, 3.1.2, 3.3.3, 3.2.*, 3.3.*

Phase 5 Addition of all additional features that make the system complete:

Tasks such as frame injection and editing of images with overlays are to be completed in this phase. On completion the product is to be released as completed.

Requirements Satisfied by this phase:

- 2.2.15, 2.2.18, 2.2.19, 2.2.20, 3.2.*, 3.3.*

4.5 Migration to the New Product

When developing a different version of an existing product, users and developers of older systems need to know what steps they need to take to move to the new system.

4.5.1 Developers

The developers are starting the software from scratch. The system as-is will be understood completely by the development team before the Code is started. Since there is no Code Migration directly, please refer to the project Gantt Chart for more information on deadlines for implementation of the new system.

4.5.2 Users

Users of the current system will be able to just as easily use the new system. The new system will load any gif, and image type that the current system utilizes so there is no need to undergo any file conversions from the old to new system. The system will require no back-up on the user end as all user data is simply the images and GIFs they use.

4.6 Risks

Risks need to be considered to better identify what potential problems might occur. Risk management entails assessing which risks are most likely to apply to the project, deciding a course of action if they become problems, and monitoring projects to give early warnings of risks becoming problems.

4.6.1 Developer Based

Time Management.

Probability High. All members of the team have a full engineering course load of 6 courses. With this in mind we have to ensure all courses are completed at a level considered subjectively good to each member. Thus time management can be difficult due to distractions from other parts of academia.

File Conversion.

Probability High. The project is to take in all standard image files for GIF frame injection. Thus all image types must be converted to a standard format, before use.

Frame Injection of images of a different format.

On frame injection a different sized image may be specified, these images need scaling options for the GIF or the frame to be allotted by the system. This task will be one of the most difficult issues to face, as it is a sub system of its own.

4.6.2 User Based

Loading in files that have loadable extensions but whose data is not relevant to the system.

Probability Moderate. This risk pertains to users loading files that are not supported by the system, this issue can occur from either a file with different data having a readable extension or specifying a non-supported file type. The system must handle these inputs.

File corruption.

Probability Low. Loading files whose data is corrupted, causes the user to be dis-satisfied with the system, even though this risk is not controlled by the system directly.

4.7 Costs

This project is being developed by a group of students for McMaster's 3XA3 course and has zero cost except for time. This is also a mandated constraint. Each group member's time has been allocated in the Gantt chart's resources section and should be referred to for any concerns relating to the cost of the project.

4.8 User Documentation and Training

A very simple user guide will be provided within the program under a 'Help' menu. This user guide will be a short series of images and text descriptions showing how to load a GIF, select a playback speed, select a subset of frames, and export these frames to a desired location on the user's PC. It will follow a format similar to below [Figure 3]. This user guide will be completed by a developer once the application is finished. This document will be updated to include the user guide when it is completed.

4.9 Waiting Room

The below requirements may not be included in the initial release of the product but may be implemented further in the development process. They are listed here so that the ideas are organized and not lost.

1. The program must allow users to record a short clip of their screen and directly convert it to a GIF format.

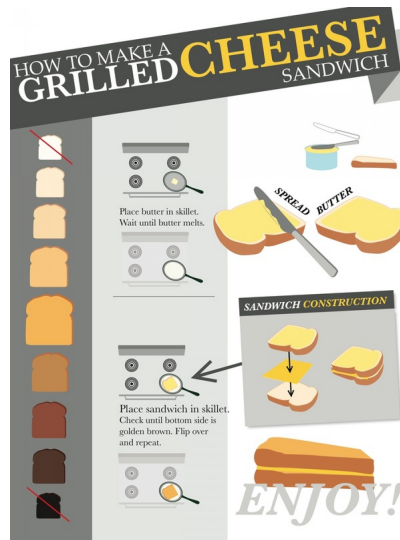


Figure 3: Example of User Guide

2. The program will allow users to sign in through Facebook or Twitter and directly upload and share the manipulated GIF.
3. The application must allow users to add filters, image overlays, or text to GIFS.
4. The application must be able to export the file as a animated GIF instead of a sequence of images.

4.10 Ideas for Solutions

1. For a intuitive UI design, see [here](#).
2. For better functionality of the UI, see [here](#).
3. For image manipulation in a C# library, see [here](#).
4. For file converting, see [here](#).

5 Appendix

The appendix for this document contains any other relevant information to the project. This section will continue to expand as the project progresses.

5.1 Research Articles

These sample articles illustrate why there is a need for products like Gifitti. GIFS are used all across the internet and on all sorts of websites. They enrich the user experience and encourage them to share their content. It has gone so far that Google has implemented a separate GIF filter to help search for animated images related to certain keywords. These animated images have been around for 25 years, and are not going anywhere soon.

1. [The Rise of The Animated GIF](#)
2. [GIFs, The Language of The Web: Their History, Culture, and Future](#)

5.2 Symbolic Parameters

1. $\text{MAX_EXPORT_TIME} = 10$
2. $\text{MAX_UPLOAD} = 3$
3. $\text{MIN_USE_TIME} = \text{A minute}$
4. $\text{MIN_USE_AGE} = 10$