

Table 1: Revision History

| Date | Developer(s) | Change |
|-------------|---------------------|--|
| Sept. 26 | Pavle Arezina | Added Team Meeting, Communication, Member Roles |
| Sept. 26 | Nicolai Kozel | Added project schedule section |
| Sept. 26 | Riley McGee | Added Proof of Concept Demonstration Plan, Technology, & Git Workflow sections |
| Sept. 29 | Pavle Arezina | Updated Introduction and Formatting |
| Sept. 29 | Nicolai Kozel | Final Proofreading |
| Sept. 30 | Riley McGee | Final additions to Rev 0, formatting corrected throughout |
| Nov. 30 | Nicolai Kozel | Edited for revision 1 |

SE 3XA3: Development Plan Gifitti

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

The Development Plan for Gifitti clearly states how the project will be created. The team meeting and communication sections demonstrate how we will work cohesively as a group within our clearly defined team member roles. To ensure that the project can be worked on without conflicts, the GIT workflow plan is defined and the risks pertaining to Gifitti are stated in order to develop plans to avoid them. The code and documentation has been constrained to guidelines to ensure a uniform project is produced on the schedule defined by the group members.

1 Team Meeting Plan

Team meetings will occur on Mondays and Tuesdays. The meetings on Monday will happen twice a day when possible during the lab periods. Tuesday meetings will occur once a day on the first floor of Thode Library. All three of the group members will contribute during the team meetings; however, Nick will act as the Team Leader and Riley will scribe the meetings. The agenda will follow the Harvard guidelines. (See Gifitti/ReferenceMaterial/HarvardGuidlines.pdf)

2 Team Communication Plan

Facebook messaging will be utilized to ask simple questions about the project to other group members. To track errors in the documentation or code, GitLab issue tracking will be used to help communicate these errors. A combination of texting and calling will be used to contact a group member when they cannot be reached through Facebook messaging. Skype will be the primary method to conduct calls for team meetings when physically meeting is infeasible. To contact a T.A. or Professor due to questions or issues with the project, McMaster e-mail will be used.

3 Team Member Roles

Nick will be in charge of the project and is designated as the Team Leader. His job is to assign tasks and provide a clear goal on what needs to be completed. He is knowledgeable on Gantt charts, GIT, image manipulation and C#. Riley will be responsible for scribing the team meetings and is experienced with GIT and C#. Pavle's responsibility will involve formatting and creating the documentation because he is an expert in GIT and LaTeX.

4 Git Workflow Plan

The development of Gifitti will follow the Feature Branch GIT workflow.

4.1 When and how to Create a Feature Branch

- The repository will have no direct code changes made on a local version of master. However, all documentation changes and additions may directly be added to master. All development of code is to occur on a feature branch from master where each branch is unique to the feature it addresses. Once a feature is completed and has been reviewed and tested, it is then to be merged into master. This merge should link the resolution of the issue on GitLab that it solves.
- Feature branch names should follow the following naming practices:
 - Brief yet descriptive
 - Keywords such as fix (for fixing bugs), ui (for view additions), feature (for general additions) followed by a brief of what the branch addresses. i.e. ui/file-menu, fix/console-fault, feature/save-as-sprite etc.
 - As seen in the examples above follow a <Category>/<Brief> with words separated with -
- Local feature branches must be committed and pushed often enough to act as a backup
- Pull requests are not needed for this project. We will all act as admins to the repository and will review feature branches and ensure they are up to date with master as a team before the feature branch creator merges it into master. This step is only needed for major changes.

4.2 Brief List of Commands for the Workflow

Make a new feature

```
git checkout -b <Category>/<Brief> master
```

Backup Local Feature Branch

```
git push -u origin <Category>/<Brief>
```

Publish Feature into Master

```
git checkout master  
git pull  
git pull <Category>/<Brief>  
git push
```

Committing

All commits must be done to the feature branch for code, and master for documentation.

4.3 Tags

- All deliverables should be tagged appropriately to mark that it is complete.
- All code releases should be tagged with a version, x.y.z where x is a major update, y is a minor update, and z is a bug fix.

4.4 Milestones

- The milestones will include all deliverables and their due dates.
- All milestones will be listed in the project schedule (Gantt chart).
- Deliverables will be considered major updates to the project.

5 Proof of Concept Demonstration Plan

Risk 1

The first risk will be reading in GIFs. GIF files must be read into Gifitti and parsed.

Plan to demonstrate risk 1's feasibility

An open source library that provides this functionality must be chosen and implemented into a base design by the project demo's deadline. This will be reflected on the Gantt chart.

Risk 2

The project will only be usable on a Windows platform, until .NET is able to be run on Linux and OSX platforms.

Plan for demonstrate risk 3's feasibility

All demonstrations and development must be done on a Windows platform.

Risk 3

Testing will be time consuming for all major components as the application is based on user input and experience.

Plan for demonstrate risk 4's feasibility

Some test cases that can be used to validate the system will be presented during the project demo to show that we can automate most of the testing that does not rely on user interaction. This saves time for the user tests.

6 Technology

Note: Currently, a Windows platform must be used to run and develop this project. In the near future .NET will be ported to OSX and Linux.

6.1 Programming Language(s)

The majority of all development will be done in C#; however, WPF will also be used for the UI.

6.2 IDE

The IDE that will be used to develop this project will be Visual Studio 2015 Community Edition.

6.3 Testing Framework

The testing framework that will be used will be the native Visual C# Test Suite, specifically unit testing. This platform comes integrated with Visual Studio 2015 Community.

6.4 Document Generation

- Doxygen will be used for code documentation.
- Visual Studio will be used for any post code software model generation
- Gantt Project will be used to document the project schedule

7 Coding Style

The coding standard that will be utilized for this project is the .NET Framework Development Guide.

8 Project Schedule

The Gantt Chart outlining the project schedule can be found [here](#).

9 Project Review

After completing Gifitti, we feel that our development plan worked well and aided in the successful completion of this project. Our GIT workflow plan was very helpful and reduced the amount of troubleshooting needed to fix merges or repository errors. Our master branch was clean throughout the entire development phase and as features were added and merged into master, it remained stable. We felt very comfortable within our team member roles and all performed within those roles to the best of our abilities. Our roles matched our personalities and we feel this made a big impact on everyone's motivation and attitude towards the project. The things that we felt did not work for us were the team meeting schedule and the coding style guidelines. We learned that because of time constraints or laziness, we did not adhere to the coding guidelines all the time and we each had our own styles of coding regardless of the guidelines. The team meetings were effective but we did not have them as often as we said we would. This was due to the nature of everyone's schedules and the fact that we did not want to meet on the weekends. Therefore, on a future project the only things that we would change would be to meet on dedicated days where we know everyone is available and choose less strict coding guidelines or have repercussions for not following them.