Table 1: Revision History

Date	Developer(s)	Change
September 26th, 2016	Team NAR	Rev0 - completed txt file
December 7th, 2016	Team NAR	Rev1 - • Chose team name • More explicit rules for agenda • Labels added to Git workflow plan • Added "Project review" and revision history

SE 3XA3: Development Plan
TheLenaProject

Team 1, Team NAR
Abeed Alibhai - alibhaa
Rahul Bablani - bablanr
Nezar Dimitri - dimitn

Our team is redeveloping the open source project *"Marvin Frameworks"*, a program which uses many java frameworks with image processing algorithms to process plain old photos, into enhanced photos with various filters, much like Instagram does. A project schedule was established on a stable means of communication in addition to an agile workflow implementation.

1 Team Meeting Plan

After several discussions, it has been decided to hold two meetings weekly. The first meeting will be held after the Thursday lab section. This is due to the nature of the labs where we learn new development and documentation techniques every week as well as new project management tools such as *GanttProject*. In addition to acquiring new skills, there is the teaching assistant, who is always available to further clarify and guidelines that may be difficult to understand.

The second meeting will be held at the end of the week on Sunday nights. This day was a prime candidate in our decision process because it marks the end of the weekend, where it was agreed most progress will be achieved in terms of completing tasks. It is also a good day to plan accordingly for the week ahead and determine if any extra meeting need to be scheduled in.

More explicitly, we will be implementing a Scrum methodology for this project. Scrum is part of the Agile movement.Scrum emphasizes decision making from real-world results rather than speculation. Time is divided into short work cadences, known as sprints, typically one week or two weeks long. The product is kept in a potentially shippable (properly integrated and tested) state at all times. At the end of each sprint, stakeholders and team members meet to see a demonstrated potentially shippable product increment and plan its next steps. Scrum is a simple set of roles, responsibilities, and meetings that never change. By removing unnecessary unpredictability, were better able to cope with the necessary unpredictability of continuous discovery and learning. This will allow us to be organized for each upcoming milestone and make sure each requirement is met well in advance for each due date.

2 Team Communication Plan

The team dynamic is quite fluid, all team members have previously worked together on projects and as a result have a solid and accountable communication plan. The primary method of communication is through face-to-face, in person conversations. Due to being in the same program the team's schedules have many overlaps which allows for easy execution of the communication plan.

Of course there are times where the primary method may render useless, and as a result there are backup measures in place for such occurrences. The secondary method set for our communication plan is through several platforms of wireless communication. Each member has their counterparts' contact information, which permits for conversations through phone, text, or third party applications such as *Facebook Messenger*.

In addition to the plans implemented, meetings will be documented with progress on current deadlines as well as strategies on how to tackle upcoming ones.

3 Team Member Roles

The project and all of its tasks will be distributed evenly throughout the group. We have assigned certain roles to make sure work is not overlapped and time is used effectively. Rahul is our Project Manager, he over sees all milestones and focuses on making sure all tasks are done on schedule while focusing on documentation but helping out with code as well. Abeed and Nezar will be the main developers, their main focus is on redeveloping the application, debugging, and ensuring the end result is more user friendly.

4 Git Workflow Plan

Feature Branch Workflow
We will have one Centralized Workflow with featuring branches. This promotes communication and collaboration within the team. All changes and edits will be done on each team members individual branch. Once the change is pushed and committed, a merge request will be sent. At this point one we have set strict rules for accepting a merge request:

1- Additions of new files may be accepted by the requester.

2- Any changes made to a pre-existing file must be reviewed and accepted by another team member.

Labelling will provide an easy way to categorize issues or merge requests based on descriptive titles like bug, documentation or any other categorizes deemed necessary. They will have different colours, a description, and visible throughout the issue tracker or inside each issue individually. We will implement this to organize our issues so it is easy to navigate the issue tracker and filter any bloated information.

5 Proof of Concept Demonstration Plan

There are many risks to redeveloping a program that already works. While trying to implement new functionality, our redevelopment team may fall into the same problems as the original development team had fallen into but failed to document. If this is the case, our team has discussed and agreed to document such difficulties, possibly revise the functions in question and act in one of two ways: drop the function completely, or rethink and implement in a different way thus extending the deadline enough to do so.

There are two main goals we are trying to accomplish: a more user-friendly experience, and a an expansion of file types supported. These goals pair well together due to how the technological industry is growing and the strong influence of social media on that growth. The application must allow for a simple user experience, and we plan on accomplishing this by creating categorical buttons as well as incorporating an additional user interface allowing the user to select files from their local drive to edit. There are few risks to these changes, most of which are based on the user's level of expertise, for example, if the buttons are categorized in a confusing way that an avid photo editor might understand or vice versa, the buttons are confused in such a simple way that it is almost completely pointless for avid users but useful for a beginner. Other problems could be associated with the importing of files into the editor. While one of our goals is to allow for more file type support, a user may upload a file that isn't supported yet or cannot be supported such as an excel file for example. Most of these conflicts can be resolved through excessive testing and finding a user base of different levels of editing experience and allowing them to use the software to determine the level of ease and areas of improvement such as easy to understand error messages with an explanation of "next steps" and how to avoid such an error.

Although most of the struggles we might face are user based, there are the obvious software problems we may encounter. As one of our goals is to work on the extensibility of the application in terms of format compatibility, we must change the image processing algorithms already in place. This leaves a vast range of problems if the developer doesn't have a strong understanding of how the algorithm works and cannot follow the original source code. These problems can be avoided by coming up with reasonable deadlines, giving the developers enough time to learn the algorithms and the original source code. Another countermeasure is excessive step-by-step testing, upon completing each task we plan to run tests to make sure the code still runs as well as no other functionality was lost in the process.

6 Technology

The original project was developed in *Java* and we believe that the redevelopment should keep it in java as it easy to create a GUI and testing procedures are much more effective in this language. We are using *Eclipse* due to its strong functionality as an IDE as well as the developers being highly familiar with it. The *Marvin Framework* is being used to process the images using different plugins for different photo filters. The GUI will be constructed using *JPanel*. Laptops and Desktops are going to be used to build and execute the program. There is also a JavaDoc set in place for the original framework and if any changes are going to be made those files they will be evident there.

7 Coding Style

We plan on following Google's coding standards for source code in Java. The source file is considered as being in Google Style if it covers the aesthetic issues of formatting and their coding standards as well. The reason we chose this is due to how closely it matches our developers style of coding in terms of:

- Braces being used even when the body is empty or contains a single statement.
- Two space indentation.
- Column limits of 80 to 100 characters.
- No C-style array declarations.
- Default statements required in switch statements.

8 Project Schedule

On Gantt Chart linked below.
Gantt Chart

9 Project Review

Overall our project implemented the key features we expected such as: importing, exporting, applying filters and error handling. We will like to continue this project past the scope of the course and add MP4 and web cam capability as well as expand it to a web application. The group met all deadlines and worked in sync to complete all documents and code to a high degree of effectiveness. All our work was streamlined through the Gantt Chart to ensure we were meeting deadlines and on track to complete the project on time. Completeness on Gantt Chart shows "78%" because we plan to implement future features as discussed earlier. Communication went really well on Facebook and in team meetings twice a week which ensured we were ready for the upcoming weeks tasks. What did not go so well within our group was at times we forgot to review the final document before submitting which resulting in basic punctuational and structural grammatical errors. These could have been easily avoided but it taught us to finish our work a day early at least to leave enough time to read over the final documents before they were submitted. I would modify the development plan and make the main means of communication issue tracking on GitLab which would host all of our problems on 1 platform instead of having issues on Facebook, written on paper; basically all on different platforms. Making issue tracking the main means of communication would make our lives more organized and create a more simple outlook on our next steps. Team meetings would not need to take place as often as they did, possibly once a week instead of the two a week we had. Team roles worked quite well as we had one tester, one coder and one person doing the documentation but we could implement a system where we would rotate through each role so each member gets a taste of each section of the project. Time management could be managed better by following the Gantt Chart precisely and meeting every deadline as some milestones had to be extended due to lack of production at certain points.