**Capstone Project Final Report**

**Spring 2016**

This report is to be used in addition with the original documentation of the Iris application, developed during the Fall 2015/Spring 2016 software engineering capstone project course at the University of Virginia’s College at Wise, as an expansion regarding specific details of the planning, design, development, testing, and deployment of the developed software. Additionally, other implications of the software development process will be discussed in this report. Suggested questions, with responses given, will be separated individually.

1. What tools were used to keep track of scheduling and project milestones?

Prior to working on the software during both semesters, a project schedule was created as a way to ensure the team had an idea on the progress of the application based on where we currently were in the development cycle compared to where we projected ourselves to be according to the schedule. During both semesters, our team was consistently on, or ahead of schedule. Because of how development was tackled, project milestones were initially determined to be when certain documents were completed and later when the software achieved certain functionality. Additionally, Dr. Qureshi acted as a lead manager and assisted in maintaining our schedule, requesting specific items, such as documentation and source code, throughout both semesters as a means to ensure our team adhered to the determined schedule.

1. What tools and methodologies did you use to code your project?

Initially, Iris was intended to be developed under an agile methodology with frequent sprints to add new features to the application throughout the lifespan of this project with a wiki page to serve as documentation of the system’s requirements, design, and testing. Once it was determined that extensive documentation of the application would be needed, a waterfall methodology was adopted instead, requiring the requirements and design of the application to be formally defined before beginning construction. We performed this twice; once for the initial development of the software during the Fall 2015 semester, and again to review our initial requirements and design in order to accommodate a variety of new features to be added to the application during the Spring 2016 semester. Several tools were used during the development of Iris. The configuration management tool that was selected for use was Atlassian’s Bitbucket, a system that contains branches, wiki, change management and issue tracking all within a single location. Eventually, issue tracking was performed using IBM’s ClearQuest as an independent testing team formed from the Spring 2016 software testing and verification course could not be included in Bitbucket without needing to pay to accommodate more users. As the application was created for the Android mobile platform, Java was the language of choice for development and Android Studio was used as the IDE during development.

1. How were requirements of the software determined?

Iris was originally intended be an application for Apple devices as a means to have email messages read to the user, with users able to respond to messages in order to keep or delete the message. It was quickly determined that developing for the Apple platform would not be feasible due to licensing costs that would need to be incurred by the college. At this point, Dr. Qureshi accepted the position as our client and the target platform was changed to Android as it free to develop for with costs only required if the application is to be launched publicly on the Google Play Store. With Dr. Qureshi as the client, we were given specific functionality that our application was expected to meet. As progress on the application continued, both restrictions on what could be delivered with the product were reached, such as our application being limited to only Google accounts, and new functionality was added, such as support for multiple accounts simultaneously. Consult the software requirements sheet for a full list of requirements that the application was designed and built to meet.

1. How were defects captured, then tracked and resolved?

As mentioned above, an independent testing team formed from the software testing and verification course performed the bulk of testing. The initial plan was to utilize Bitbucket’s built in issue tracking as a means to maintain a list of issues with the application that needed attention. However, it was determined that Bitbucket limits the number of users under a free account, so we switched to IBM’s ClearQuest as a system for the testing team to submit bug reports and create issues to be tracked. From ClearQuest, issues with details on where or how they occur could be viewed and our team could make changes in the application to correct them and mark them as resolved. It should be noted that our team created and performed test cases once issues created by the testing team had been resolved in order to prevent duplicate issues from occurring.

1. How was the status from team members used for weekly meetings and reports?

The most common form of communication used by the team was email. Each week, an email was sent to the individual responsible for creating the weekly status report that included what the team had completed, what the team was currently working on, and what was upcoming next on the agenda for the team to complete. Meetings occurred for approximately one hour each Tuesday during both semesters, with additional meetings scheduled as needed to completed tasks together.

1. Was every requirement tested and included in the software test plan?

Yes, at least each of the functional requirements had at least one test case to ensure the proper functionality worked as expected. In total, there were 12 test cases created, some with multiple sections to accommodate different results for testing the same item in multiple ways.

1. Were there any unresolved issues after formally testing the software?

Upon the completion of both the independent team’s testing and our group’s own testing, there were several issues that became apparent and were tracked within ClearQuest. Our team successfully resolved all issues that were discovered by both teams and as of version 1.0, there are no major issues that are unresolved after the completion of formal testing.

1. What did you do to hand off the software to the customer?

Because our instructor is also our client, we have submitted all documentation of Iris, as well as the source code, in the case that a future team is tasked with creating enhancements for the application. The team has also created and submitted a research paper about the software development process of Iris along. Finally, instructions regarding how to deploy/publish the application to the Google Play Store were created and handed off.

1. How did you ensure the customer was able to maintain the software after delivery?

While there is currently no team or individual designated responsible for the maintenance of the application as it the property of the college and most individuals currently responsible for the application are planning to graduate this semester, the entire software development process and source code was documented and submitted to our instructor for future teams to pick up and modify at their will. As stated above, instructions on how to deploy/publish the software to the Google Play Store were created and submitted as well.

1. What are the ethical and social impacts of the software that was developed?

Our software is designed to give users a hands-free means of receiving email messages through their Google account(s). However, our software is not completely hands-free; it still requires both touch input and the user’s attention for navigational purposes through the application, manipulating settings, and for deleting and keeping message(s). Because the two experiences are mixed together, it should be understood that we do not intend our users to actively use the application while in scenarios that users should not use mobile devices due to distraction, such as driving. The application is designed with security to be handled by the Android operating system; the application does not store, manipulate, or see the user’s account passwords for their accounts, nor will it permanently manipulate messages received by the user. Additionally, the Iris application was not designed to act as a replacement for the user’s primary email client, but only as a means to alert the user when a message appears by reading it to them when the user may be busy performing other tasks.