Griswold University Post-Implementation Report

Matthew E. Heino

Western Governors University

Abstract

The Griswold University software solution addressed the problem of hardware utilization problem at the university. The previously implemented solution only allowed university resources to be used to access the students' information. This solution did not allow for mobility and ease of access by the student body. The solution to this problem was the development of a mobile application. This mobile application shifted most of the resources from the university equipment to the student's devices. This project was conducted with other projects that were occurring on campus. Resources for this project were allocated as they became available. The project was composed of many different stakeholders. These stakeholders had a different view of what would constitute a successful project. The solution was one in which the student's tasks would be completed using a mobile application. This will remove the requirement of using the university's resources and free up resources for other academic tasks. There were a few points that were developed in this solution. The IT staff learned new skills. A reduction of the administrative requirements. Students received an application that they were deeply involved in. The administration saw a reduction in costs and realized labor and money savings to maintain the new solution. This paper seeks to document the project that ran from August 1st, 2022, till the end of October 2022.

Keywords: efficiency, mobile application, resources allocation, software, students

Table of Contents

Abstract....2

Post Implementation Report Title....5

Quality Assurance....6

Formative Evaluation and QA Metrics....7

Testing Methodology....10

Test Cases....12

Acceptance Criteria....15

Project Review....16

Assumptions....17

Project Phases....19

Timeline Deviations....26

Project Dependencies....26

Resource Requirements....29

Implementation Project Milestones....31

Project Deliverables....35

Documentation Deliverables....36

Formative Evaluation Results and Revisions....37

Summative Evaluation Plan and Results....37

Stakeholder Communication Plan and Reports....40

Ongoing Support and Maintenance....43

Resources for Post-Implementation Support....44

Short and Long-Term Maintenance Plan....46

Post-Implementation Project Summary....49

Documentation Deliverables....50

Success Criteria for Each Project Outcome....53

Justification of Proposed versus Actual Project Outcomes....54

Lessons Learned....55

References....57

Tables....59

Post Implementation Report Title

The mobile application was developed to address a few critical needs of the university. The solution solved the resource usage problem by moving most tasks from university computers and kiosks to the student's devices. The IT department saw reduced labor used to administer the application. The administration saw a reduction in costs both in terms of labor and costs to maintain the solution. The previous solution required using aging kiosks and increasing costs to maintain.

The project addressed a few points from design to implementation. The requirements of the students drove the design. These requirements were integrated into the application. This was important because, over the course of the project, we received feedback from the student body about what worked and did not work. We took the feedback to heart and tried to implement the feedback into the creation of the application.

The IT department was most concerned with reducing hardware usage and the labor involved in maintaining the previous solution. The IT department needed to maintain two different systems on campus. The first system was the lab computers and other equipment used to accomplish academic tasks. The other was the system used to provide the students with their information and process their tasks. These two systems could not be used together. There were information kiosks that were in use. Still, the kiosks had limited functionality for processing student requests, and they relied on other computer systems to handle most of the requests when it came to processing the student's requests.

The administration wanted to see a freeing up of capital and human resources. This project did just that. The resources that were required to run and maintain the previous solution

can now be allocated to other activities on campus. The money spent on the aging hardware for the kiosks can currently be assigned to different campus needs.

Quality Assurance

The goal of the project was to create a solution that was quality. But what does quality include? We approached quality assurance by taking a few approaches to ensure quality is built into the project. The approaches will recognize the following areas (Project Management Institute, 2013):

- **Customer satisfaction**. We looked at the customer as the student body. We wanted to ensure they were happy with the features implemented in the first version of the application. We needed to address that this application will be supplanting the old system and to ensure that the functionality is maintained and some new and better functionality added. We need to make sure that the IT department realizes a reduction in their workload, which was realized. They also wanted to see the previous solution resources be freed up for other activities. This was also realized with the new solution.
- **Prevention over inspection**. During our project, we tried to prepare for all types of risks and circumstances that may arise. We planned for both expected and tried to categorize risks. We categorize risks, whether human or environmental risks to the project. These precautions will make sure that these risks can be mitigated. We looked at the software produced to ensure that the product created is quality. We made sure to implement procedures that will try to have viable code that follows the best software design practices.
- **Continuous improvement**. Over the course of the project, we tried to get feedback from the stakeholders. We wanted to keep them involved throughout the whole project. We

especially sought to get feedback from the student body. We must keep these stakeholders happy to ensure they will engage with the application when deployed. We want their buy-in the most they will ultimately determine the solution's success.

- **Management responsibility**. While I was the project manager on this solution, I tried to devise metrics that would be used to make sure that there was a quality product being produced. Quality is needed to meet the needs of the stakeholder but also the financial and regulatory requirements of the application.
- **Cost of quality**. We were developing an application that had never been done before by the IT staff. We unavoidably had to rework the project to ensure it conformed to what we needed in a final project. This added to the cost both in terms of time and financial expenditures.

The above bullets laid the framework for developing a way to ascertain the quality of the software. These bullets allowed for a well-reasoned approach to guaranteeing that there will be a quality solution produced and that it will meet the needs of the stakeholders.

We used different QA metrics to ensure this product was being produced promptly. These metrics were used to get a feel for the progress of the project and the quality of the code created for the mobile application. The following section will discuss the metrics used to ensure quality was built into the project and the software.

Formative Evaluation and QA Metrics

As documented in the previous paper, we will use several key performance indicators. These indicators will revolve around the code and how often the code is released as a stable feature that can be used and tested by the student body.

As discussed in the previous document, we will look at several performance indicators regarding software development. These indicators were cycle time, development velocity, change failure rate, development frequency, and PR requests (5 Software Development KPIs for a Savvy Engineering Leader, 2022). We can use these in tandem to help gauge the progress of the project and the quality of the code being produced.

With cycle time, we looked at when a work package was started until it was finished. Ideally, we want the overall turnaround time to be as short as possible (5 Software Development KPIs for a Savvy Engineering Leader, 2022). Over the project, we saw a decrease in the time it would take to create a viable work product. We used this metric and compared it against other work products produced.

We could see a decline over the project's course compared to the project's opening weeks. The opening weeks of the project were used as the baseline to understand the project's progress. This was because the developers became more accustomed to the new development environment, and therefore there was a reduced time to complete each new work package.

Another metric we used to see if the project was on track was development velocity. Development velocity is a metric that measures how much work is being produced over time (5 Software Development KPIs for a Savvy Engineering Leader, 2022). This metric is very similar to the cycle time metric, but it looks at work produced over a given period for all work products produced within that timeframe.

We used the Agile methodology. We wanted to see consistent work products produced over the project. Over the project, we noticed that as time went on, more products could be produced as the developers became more proficient with the development environment. We

used the first few weeks of the project as the baseline, and we could see that as time went on, the production rate increased.

The change failure rate measured the code's quality and whether it produced a failure in the software solution (5 Software Development KPIs for a Savvy Engineering Leader, 2022). We made a very robust product that did not seem to have too many failures when integrating it into the existing code base. We aimed for a rate that was as close to zero as possible. It is not possible to produce a change failure rate of zero. This is especially true when you have new developers in the development environment. We tried to get to the lowest percentage possible. We saw these percentages decrease as the project progressed. We attributed this reduction to the developers' getting better at producing code using this new platform.

With deployment frequency, it measures how often our developers can push code into production (5 Software Development KPIs for a Savvy Engineering Leader, 2022). As the project progressed, we saw code pushed with increasing frequency. As stated earlier, this was due to the developers becoming more proficient with the platform.

The last performance indicator was the PR size. This metric looks at how long it takes to review the currently sitting code waiting to be pulled and reviewed (5 Software Development KPIs for a Savvy Engineering Leader, 2022). We strived for relatively short reviews, which were accomplished as the project progressed.

These metrics were easy to measure and chart. We can easily see the progress and the improvements in the turnaround time for the various features of the project. We could even gauge the improvement of the developers as they become more knowledgeable about the platform. While these are quantitative metrics, we also tried to get some qualitative metrics as well.

Over the project, we tried to get some feedback about the use of the application. We strictly engaged with the student body. When we did this, we asked the students to assess the features and GUIs produced thus far. This ensured that all aspects of the application worked as the students expected. We needed to ensure that the application fit their idea of what it should do and what it should look like. Any negative feedback or feedback that was thought to be a great idea was implemented into the feature. While this approach took more time, it was a way to ensure that the students would get what they deemed as a suitable application to meet their needs.

Testing Methodology

As discussed earlier, we used an Agile approach to the project and the software development. We used the Agile model to ensure a feature was always completed that the stakeholder could see. With each feature, the stakeholder could have a tangible piece of the application and see how it will function.

The Agile methodology entails an incremental approach and feedback from interested parties (Hamilton, 2022b). Why was this approach used? We used this approach to minimize risks during the implementation and development process. We could have used the waterfall development approach but did not see it as a worthy option. With the waterfall method, there is very little input from the stakeholders. Most of the input comes at the beginning of the project, and there is further input at the end. This leaves much room for error (Hamilton, 2022b).

With Agile, we can get feedback whenever we develop a needed feature. We can get immediate feedback from the stakeholders – like the students. There are other advantages to using this approach. We were new to this platform for producing an application. With this approach, we can constantly check the product to see if there were any errors in the code

produced or in the interpretation of the wants of the stakeholders. With Agile, we can make adjustments as they appear, and any mistakes can be corrected while it is in an early iteration of the product.

The course of the project employed three basic methods that are integral to Agile development. We used the concepts of behavior-driven development, acceptance-driven development, exploratory testing, and session-based testing (SeaLights, 2019).

Behavior-driven development was utilized to ensure that both the developers and the stakeholders mutually understood what would be produced with the project. With the method, we devised scenarios that mapped out how the students would use the application. These scenarios were the students' actions to accomplish the tasks previously done by the old IT solution. These scenarios would become the foundation for the tests that would be used to test the functionality of the application feature before that feature is integrated into the whole application (SeaLights, 2019).

Acceptance test-driven development was utilized to develop the test that would be used to define the acceptance tests. These tests would be used to gain an idea of the application from the student's perspective (SeaLights, 2019). The goal of this is to make sure to develop tests that closely approximate the way the students will use the application. We wanted to build the functionality around these tests. If we built around these tests, we were assured that the functionality would be something the students wanted and an application that the students would be happy to engage with.

During the early phases of the project, we used exploratory testing to test the project to see if we could get it to fail (SeaLights, 2019). We did this mainly to see if there were any bugs in the code or the logical layout of the applications. This testing allowed us to explore the

functionality as the developers understood the requirements of the various features. We did this testing before we passed the feature on for testing by the students.

As major sections were completed, we would implement session testing. We would ask the students to come in and test the software. During the session, we would ask them to complete different tasks based on the completed feature set. We wanted to see if the students could complete these tasks. We asked for feedback from them and made adjustments to these features as needed.

This approach to creating the application ensured that the stakeholders, mainly the students, had an integral part in the development of the application. During these methods, many tests were created to justify moving on to the next feature and adding the newly created feature to the whole application.

Test Cases

Test cases were developed to test the different areas of the application. We tested things like the flow of the GUIs and the application's functionality. We tested each feature as it was completed and integrated with other completed components of the application.

The first part of the application that we tested was the GUIs. We developed the GUIs first since these will need to perform the functionality and display information to the users. Tests in this regard were most subjective and did not affect the application's functionality.

In GUI testing, we tested things that the user cared about. We checked to see if the elements were correctly displayed on the screen. Do the buttons forward to the right screen or perform the correct action when pressed? This is with other items like checking to ensure all items on the screen are readable and pleasing to the eye (Hamilton, 2022a).

Most of these tests were performed manually by asking the students to perform tasks they would typically accomplish while using the application. These test cases were based on the features implemented on each screen. For example, the students would be asked to enter information to add a course to their schedule. During this phase, this functionality may not be implemented. We could see how easy it is for the student to complete this task and get any feedback from them.

A quick list of the test cases is the following (Hamilton, 2022a):

- Testing the font (size, position, height, etc.)
- Testing different sections of the screen for display aesthetics.
- Testing of hyperlinks.
- Testing error messages, e.g., does it display the appropriate error message if a required field is left blank?

The above list is not exhaustive but is included to show the types of items that were tested during the testing of the GUIs. The application's functionality will come with the other test cases in the following paragraphs.

Test cases were documented using the following template that can be saved for future use and when we implement other features discovered during the survey, or that may need to be implemented to meet future demands.

The template has the following structure (Software Testing Help, 2022).

Project Name: Griswold University Software Solution	
Test Case Id:	Test Designed by:
Test Priority (Low/Medium/High):	Test Designed Date:
Module Name: Name of the item being tested	Test Executed by:
Test Title:	Test Execution date:

Description:						
Precondition:						
Dependencies:						
Step	Test Step	Expected Result	Actual Result	Status (Pass /Fail	Notes:	

Test Case Template for Documenting Test Cases

This template was created to help document the tests performed mainly on the mobile application's functionality. It will form some project documentation to aid in future development and additional troubleshooting if required.

We developed the actual test cases based on the application's functionality. The functionality in the project's first iteration was to match the functionality of the previous solution closely. We tested the following functionality or feature:

1. **View academic record feature**. This test case tested whether the functionality worked as intended and provided the result that we were expecting. This initial testing of this case was done by one of the developers. This was primarily to ensure that the feature functions as the developers expected. This was later passed on to the students to see their

reaction to the feature and gain their input on what to add or if the feature was acceptable to them.

2. **View class schedule**. This feature was tested in the same manner as the first feature. We tested by the developers and then by the students.
3. **Add course**. Feature tested in the same manner as the previous.
4. **Delete course**. Feature tested the same way.

While this is not an exhaustive list of the test cases, it is included here for illustrative purposes. Satisfying these tests in the manner prescribed made sure that the mobile application would have input from the students and that the application accomplished everything they desired in the way they expected.

Acceptance Criteria

Acceptance criteria for the mobile application for this application entailed creating an application that met the needs of the students and the other stakeholders. To think of the acceptance criteria is a series of scenarios that need to happen.

The first scenario is that we produce an application that functions like the previous solution. The new solution meets the goals of this scenario. The mobile application can do all the functionality that the previous solution could and can be added to at a later date with very little problem.

For this, we have devised a template for the required functionality of the application. Using the template, we can ascertain if the functionality has been achieved. The template has the following form (Editor, 2019).

Feature Name:	
Scenario:	Description of the scenario, aka feature
Given	The event that needs to happen
When	What has to happen
...	
Then	Final action

Acceptance Criteria Template

The template above was used to help provide steps that a feature needs to be implemented to accept the feature. The table can be added based on the steps necessary to complete the desired feature. It can be accepted if these steps are met and the feature performs as expected.

While this template was a good way to ensure that the feature meets the criteria for what it is supposed to do, we need to conform to other areas like standards for the documentation. We tried to conform to ISO 29119 for software testing during the project. This set of standards sets forth guidelines on systems and software engineering. This standard provided us with what types of documentation will be created and the techniques implemented in this project. The techniques were discussed in the previous section and throughout this document (Software Development Standards: ISO Compliance and Agile, n.d.).

The university has much experience creating applications for their use, and the acceptance criteria were used on this project and for any further project that may involve this mobile application.

Project Review

The project went well for the most part—a few items cropped up during the project's evolution. We made a few assumptions about the project discussed in the previous document.

Some of the assumptions helped throughout the project and did not cause a problem for the project. While others did not hold as expected. Both assumptions that held and those that did not go as planned are discussed in the next section.

Assumptions

The project had many assumptions. These assumptions will be discussed as bullet points below. These assumptions were made at the beginning of the project. This section will look at whether these assumptions were held over the course of the project. The assumptions are the following:

- **Many of the required resources will be available in-house**. As discussed in the previous document, the only outside resources needed were the survey processor and the trainer. This assumption was held during the course of the project. Once the survey was completed, this resource was no longer needed. After the developers completed their training, they did not require further training from outside resources. They could consult online materials and the training provider's learning materials when they have questions. So, this assumption was valid.
- **Software developers had the required basic skills for the project**. This assumption held. The developers had the basic skill set. There was no need to bring in outside help. We did not need to bring in additional developers or subject matter experts.
- **The university administration will support the project 100%**. During the early phases of the project, there was a little apprehension as to whether this project could be pulled off with the resources allocated. As more of the features were delivered, the administration could see that project was starting to bear fruit. With each successive feature completion, there was a decrease in the reluctance of some individuals in the

administration. As a bonus, the IT department began to see that this application would accomplish their desire to reduce maintenance overhead.

- **Students have a vested interest in the project**. In the initial weeks of the project, there was a feeling that this project may not accomplish the whole student buy-in. As time progressed, the students began to consider the project a viable application that would meet their needs. Using the student body as testers added to the feeling of being vested in the project. The students felt they had a say in what the application does and looks like.
- **Project manager had authority**. The project manager had the authority to manage the project as they thought fit. The only problems that arose were when procuring the servers for the project. It took a little time to get the funds from the administration. The administration still had complete control over the dispersal of funds as they related to the project.
- **Current server and other university hardware can handle the new work model**. The current infrastructure was able to handle the new workload. This is because their workload never really changed. The servers handled the requests as they usually did and did not need to be modified in any meaningful manner. With the addition of the new servers, the requests and translations of these requests were handled by these servers, and the older infrastructure did no additional work.
- **IT staff can conduct the day-to-day operations whenever they need to do so**. Since this project was a back burner project. If resources were required for other projects or activities that could happen. During this project, this event did not occur. While there were instances where the developer and the technician were called away to handle other It events, it did not dramatically affect the project's overall outcome.

- **Student volunteers will be able to test the application after each feature**. Early on, this was a problem, but getting student volunteers to test the application was easier as the project progressed. This was attributed to the fact that they saw that their feedback was being considered while the application was being created and tested.
- **There is money allocated to the project to see it completed**. While there was a drop in enrollment as predicted in the previous document. There was enough money to complete the project for the proposed budget.
- **All students' needs will be covered, and they will adopt the new application**. After the mobile application was deployed, there was a noticeable drop in the use of the kiosks and the computers allocated to providing the functionality the previous solution provided. Most of the requests for student information now come from requests that are routed to the new servers and from off-campus as well.
- **Technician can configure and provision the appropriate resources**. The technician was able to accomplish this task with no real hindrance. The only hiccup was that getting server hardware was a little challenging, but it was made available before this project phase. So, the hardware was available but not as soon as desired.
- **Staff will not go on vacations or take unapproved vacations**. The only exception to this was that one of the developer's family members passed away, and they needed to attend to family affairs. This did not have too much of an impact. This lack of impact was due to the project being a little ahead of schedule. The schedule was ahead because the developer was learning to use the new platform more efficiently as time passed.

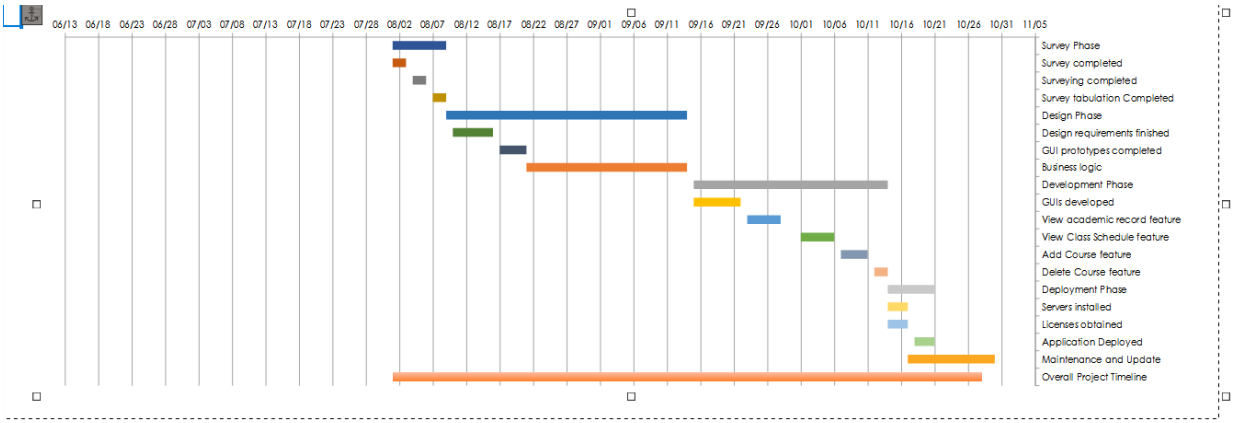
Project Phases

This project was completed at the end of October 2022. This evaluation is commencing the last week of November 2022. Below is a planning and timeline chart that covers the most important phases and milestones.

Project Timeline

Phase	START DATE	END DATE	DURATION in days	COMMENTS
Survey Phase	08/01	08/09	8	
Survey completed	08/01	08/03	2	milestone
Surveying completed	08/04	08/06	2	milestone
Survey tabulation Completed	08/07	08/09	2	
Design Phase	08/09	09/14	36	
Design requirements finished	08/10	08/16	6	milestone
GUI prototypes completed	08/17	08/21	4	milestone
Business logic	08/21	09/14	24	milestone
Development Phase	09/15	10/14	29	
GUIs developed	09/15	09/22	7	milestone
View academic record feature	09/23	09/28	5	milestone
View Class Schedule feature	10/01	10/06	5	milestone
Add Course feature	10/07	10/11	4	milestone
Delete Course feature	10/12	10/14	2	milestone
Deployment Phase	10/14	10/21	7	
Servers installed	10/14	10/17	3	milestone
Licenses obtained	10/14	10/17	3	milestone
Application Deployed	10/18	10/21	3	milestone
Maintenance and Update	10/17	10/30	13	An ongoing phase
Overall Project Timeline	08/01	10/28	88	

06/13 06/18 06/23 06/28 07/03 07/08 07/13 07/18 07/23 07/28 08/02 08/07 08/12 08/17 08/22 08/27 09/01 09/06 09/11 09/16 09/21 09/26 10/01 10/06 10/11



Project Timeline with Chart

Note: Larger versions can be found in the Tables section of this document

As described in the previous document, the project was divided into five basic phases. These phases were Survey, Design, Development, Deployment, and Maintenance and Update. Each of these phases and their milestones will be discussed in detail. We will review each phase,

the milestones accomplished, and whether they were completed on time or took longer than expected.

Survey. The survey phase ran from August 1st to August 9th. This is the phase in which the survey was created by the survey processor and eventually handed off to the students. Survey completed milestone ran from August 1st until August 3rd.

The subsequent milestone Surveying completed ran from August 4th till August 6th. The students were tasked with delivering the survey to as many students as possible within the given time frame. The survey was delivered by email and via the computers that the survey company provided. Only some of the students needed to answer the survey. We needed a consensus on what the student body wanted and needed for the application to be successful.

Some items found during this stage may only be implemented late in the project or during subject updates to the software solution. Our key objective was to ensure that the new solution met the current functionality. If we had time and the budget allowed, we would start to design and develop other features.

The survey tabulation completed milestone ran from August 7th until the 9th. During this phase, the students tabulated the results of the survey. We did this to save a little money on administering the survey. We wanted to make sure that the students felt like they were contributing in a beneficial and they were. We also valued their opinion on what constituted needed features versus wanted features. If we were doing it ourselves, we would have to make too many assumptions about needs and wants. We would risk getting this wrong and jeopardize the project by prioritizing features that were not wanted or needed at this time.

The work product from this was delivered to the developers so they could begin the design phase of the mobile application. At the same time, we had a good idea of what the project will need based on the previous solution. We wanted to see what other complimentary features could be implemented during this project to increase student body adoption of the application.

Design. This phase of the project ran from August 9th to September 14th. This phase was conducted after the features were gathered and ranked by the students in the previous phase.

The design requirements milestone ran from August 10th to August 16th. The developers looked at the features presented by the students. The developers choose features that the current solution does not provide but will either add value to the application or make the application more user-friendly.

Since we already had an idea of the base features of the application, we went through the list. We looked for complimentary features that could be easily implemented during this project. Taking this approach, we aimed to increase the application's usefulness and the students' acceptance since this approach will address any shortcomings of the previous solution.

The developers will lay out the application's GUIs during the design phase. The completed GUI prototypes milestone ran from August 17th to August 21st. They can begin to create these GUIs based on what the current system does. The developers will be able to add to it as they get feedback from the students. The developers did not produce any code during this time but used software like Figma to create prototypes that could be displayed to show the flow of the application and what it would look like. Software like Figma allows you to build prototypes quickly without creating the code. This method was very efficient since prototypes can be created and changed in a drag-and-drop manner, and change can be made effortlessly.

The business logic milestone began on August 21st and ran to September 14th. During this milestone, the developers are implementing the business logic that will be used to bring about the desired functionality. At the same time, the previous solution will already implement most of the application logic.

Once the design and the functionality flow were fleshed out, we moved on to the next phase – development.

Development. The development phase ran from September 15th to October 14th. Since we used Agile in our overall methodology, this phase became very iterative in nature. It was iterative because we would develop and test a new feature. The developers would test the feature based on their QA standards, and then they would ask the student body for feedback about the features and how they were implemented. Any significant feedback would force the feature to be reworked to meet the desires expressed in the feedback. This was the most time-consuming part of the project. The developer created processes that made it a little faster as time went by. They were able to automate tests that tested the functionality of the application. This would free up time to work on other items, e.g., new features or implement feedback.

As each feature passed the quality section of this phase, it would be implemented into the application and then tested again to ensure that it integrated properly with the rest of the application. Likewise, students would be utilized to test the application and take notes on what worked and what didn't.

This development phase was composed of quite a few milestones. Most of these milestones corresponded to the features that were provided by the older solution.

GUIs developed milestone ran from 9/15 until 9/22. This milestone is when the GUIs for the application were produced and tested. These were produced first since they make it easier to do user testing with the students. These were tested for acceptance by the student body. While no functionality was yet associated with them, the student testers could give feedback on the overall flow from one screen to the next.

View academic record milestone (9/23 – 9/28) was the ability for the application to show the student's academic record. This was chosen first since it was easiest to complete and still provided the functionality. Using this as the base the developers could build upon is advantageous since our developers were still learning the new development platform.

View class schedule (10/01 – 10/04) was a feature implemented in the previous solution. This feature did not take as long as I first thought. Much of the logic was already implemented in some way in the last milestone. It just needed to be augmented to fulfill this feature's expected functionality. This allowed the project to pick up a few days of development time.

Add course (10/05 – 10/07) was another feature implemented in the previous solution. Most of the development time was spent creating the logic compatible with the mobile application. We gained a few days here since the logic was straightforward and only pertained to how the application would handle the functionality as opposed to the previous solution.

Delete course (10/08 – 10/10), a feature from the previous solution. It did not take as long to implement. We were able to pick up some time here. We were able to pick up a day here.

When all the features were created, we could move on to the deployment phase of the project. This is where we could deploy a fully functioning application that the whole student body could use.

Deployment. The deployment phase ran from October 11th to October 18th. Licenses obtained milestone ran from October 11th to October 12th. The licenses came at a reduced cost since we are a non-profit educational institution. This only took a day since the application was straightforward and approved the same day. We could deploy the application to the appropriate app store with completed feedback and testing.

The servers installation ran from October 11th to October 12th. During this phase, we had the technician set up the servers and provision them as needed. These servers provided access to the school's infrastructure outside the school's computer network.

We were able to begin the application deployed milestone early since we were able to gain a few days during the development phase of the project. This milestone started on October 13th and was completed on the 14th. This phase also took less time than expected. During this time, one of our developers had a family issue. Since most of the project was completed, it did not cause a problem. This event did not have a significant effect on the project. The remaining developer could cover any emergency, and two developers were not required to deploy the application.

With the mobile application deployed, we can terminate the deployment phase and move on to the current phase of the mobile application – maintenance and update.

Maintenance and Update. This phase's time frame is ongoing and will run until the current application is retired or replaced with a different solution. When we deployed the

application, it did not mean the project was over. During this phase, we will document the project. This documentation will aid in the future development of similar projects and document the lessons learned from this endeavor.

We will monitor the application for any problems, whether they be bugs not caught during development or spring up in deployment. The developers will monitor feedback in the app stores for suggestions that could improve the application and benefit the students. The developers will look at the students' features list and try to prioritize them. This prioritization will add features as time and budget allows.

Timeline Deviations

The project was brought in within the time frame allotted it. We were able to bring in the project ahead of the proposed timeline. This is due to many factors. Since caution was abundant when estimating the time to complete the development, we had a few extra days.

As the developers gained more experience with coding with the new platform, they learned tricks to produce code more quickly and efficiently. While this gain is not expected to hold up for future projects. It is nice to know that this is possible.

Developers could also devise automated testing tasks, so they do not have to do all manual testing tasks.

This time-saving was significant since this project depends on the phases being completed before the next task or feature could be initiated. These project dependencies will be discussed in the next section.

Project Dependencies

This project was dependent on the previous phases being completed before the next one could be initiated. This pertained to the phases and milestones contained within each phase.

Some could be started concurrently. For example, we could begin the implementation of the servers and get the licenses. Neither one of these operations was dependent on the other. The milestones of the project need to be completed in this order.

1. **Survey completed**. This milestone began the survey phase of the project. This was the first significant milestone of the project. We needed to complete this survey to move on to survey the students. This survey would ask questions to ascertain the student body's needs.
2. **Surveying completed**. We were achieving this milestone and allowed for the next milestone to begin. We did not need to get input from the whole student body, but we needed to get enough of the student body that we had a good feel for what they wanted.
3. **Survey tabulation completed**. This milestone could only commence after the surveying milestone was achieved. The tabulation of the surveys could only be conducted after the previous milestone had ended. The completion of this milestone indicated the end of the survey phase of the project, and we could begin the project's next phase – the design phase.
4. **Design requirements finished**. We could only begin this milestone once we had the tabulated survey data from the students. This milestone marked the beginning of the design phase of the project.
5. **GUI prototypes completed**. We needed to complete the design requirements that were created during the previous milestones.
6. **Business logic**. This milestone could be completed before the GUIs prototypes were completed, but it seems better to create, begin and finish this milestone after the GUI

prototypes were created. This milestone also marked the end of the Design phase of the project

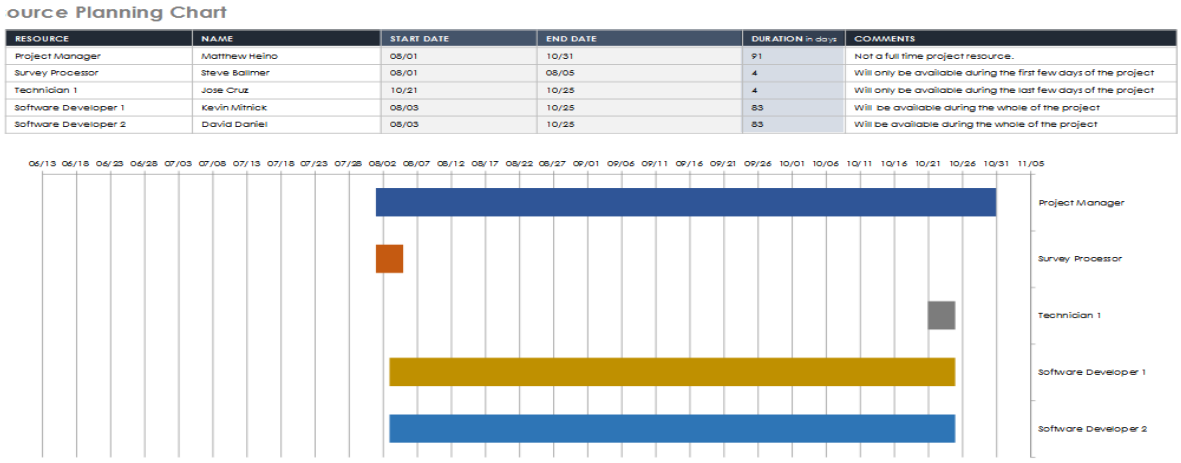
7. **GUIs developed**. This milestone began the development phase. To ensure we have a way to allow the developers and students to test the functionality. These needed to be created before any of the functionality.
8. **View academic record feature**. This milestone needed to be completed before any of the following features. This feature milestone was used as a warm-up since it was a good way for the developers to implement what they had learned. This milestone could be used as a template to create the other features found in this project.
9. **View class schedule**. The milestone could have been created either before or after the previous milestone. The order did not matter as long as it was completed before the other milestone features.
10. **Add course feature** and **Delete course feature**. These milestones could have been completed in any order during the project. The only requirement was that they were created after the **View class schedule** milestone. The **View class schedule** was used to show the updates after the add and delete courses were implemented.
11. **Servers installed,** and **Licenses obtained**. These milestones could occur at any time during the project. Server installation relied mainly on the timing of the server hardware procurement. As discussed in the previous paper, there were questions as to the availability of hardware due to supply chain issues.
12. **Application deployed**. This milestone depended on completing the milestone in the number 11 bullet. We couldn't deploy the application until the servers were installed and the appropriate licenses were obtained.

This project was very linear in the way it progressed. There was very little room to move the component milestones around. If we had a delay in one milestone, it would hold up the start date of the dependent milestone.

While this project used many in-house resources, we had a few resources outside the university. In the next section, we will discuss the resources that were used in the project.

Resource Requirements

The project required a few types of resources to bring the project to completion. Most of the resources used for the project were from the university's staff. We did consume a few outside resources. These were mostly the hardware for the computer labs and the servers needed to process the student's requests from the Internet. The following chart shows the resource allocation for the project's human resources.



ource Planning Chart

RESOURCE	NAME	START DATE	END DATE	DURATION in days	COMMENTS
Project Manager	Matthew Heino	08/01	10/31	91	Not a full time project resource.
Survey Processor	Steve Ballmer	08/01	08/05	4	Will only be available during the first few days of the project
Technician 1	Jose Cruz	10/21	10/25	4	Will only be available during the last few days of the project
Software Developer 1	Kevin Mitnick	08/03	10/25	83	Will be available during the whole of the project
Software Developer 2	David Daniel	08/03	10/25	83	Will be available during the whole of the project

Human Resource Plan Chart

The chart above depicts how the human resources were allocated during the project. The survey processor was only active for the project's first week, mainly during the survey creation. Surveying the student body was handled by student volunteers. The project manager was included for the project's duration, but the project manager worked only part of the time on this project. The project manager split their time between other projects that were being engaged on the campus.

The technician was a human resource utilized in the project's deployment phase, and they only worked a few days. The technician was the one responsible for the server installation.

The only actual full-time employees of the project were the developers. They produced the code as well as tested the code. The developers ensured that the code functioned and would be suitable for the test by the student volunteers.

This chart did not reflect student volunteers as their needs were temporary. We would ask the students via email if anyone were interested in testing the application in their free time. We always had a good response and never wanted for testing throughout any phase.

The financial resources for the project were slated to be about $27,054. We were able to bring this cost down a little because we saved a little time on the project. We cost savings because our developers became more proficient at producing code and became more familiar with the platform. We saved about $3000, reducing the project cost to $24,054. This was derived from the hourly rate of the developers of $37.50 and a savings of 80 hours. This financial resource was allocated to paying for the developers' labor, the survey processor, the technician, and the hardware needed to create the project.

The hours that were initially quoted for the project were 380. This we reduced for the reasons given previously. We were able to bring the project to completion within 300 hours. This led to savings that were discussed in the previous paragraph.

The project was punctuated with milestones marking important events in creating the mobile application. Each of these milestones was implemented in a way that allowed the project to be completed in a time-efficient and cost-effective manner. The following section will explore the implementation of the milestones.

Implementation Project Milestones

The project's milestones were used to gauge the project's progress and to provide features that the stakeholders could see and implement. The implementation for most of the milestones was very similar and will be discussed in the bullets below.

- **Survey completed** (8/1 – 8/3). To implement this milestone, we needed a key outside resource. We needed the survey processor. This was one of the few outside resources unavailable to the university staff. We wanted to have an expert prepare the survey because we wanted to make sure that we asked the right questions. We understood that we were on a time limit for this milestone of the survey phase. We wanted to pass this document on to the student volunteers so they can begin surveying the students.
- **Surveying completed** (8/4 – 8/6). This milestone was largely implemented with help from volunteers from the student body. They were tasked with getting as many students as possible to answer the survey. The volunteers were only allotted a few days to accomplish this. We did not need the whole student body to answer this survey. We just needed enough to ensure we have a good feel for what they require of the mobile application.

- **Survey tabulation completed** (8/7 – 8/9). To help save money and help the students feel involved in the application, we allowed them to tabulate the data. This tabulated data was presented to the developers who will use it in the next milestone.
- **Design requirements finished** (8/10 – 8/16). With students' major involvement mostly over. The student's tabulated data can be used to create the requirements documentation for the application. The developer will look at the tabulated data and look for requirements that would be logical and complimentary to include with this version of the application. At this point in the project, we did not want to add any feature that was not part of the original. If we were to add a feature, it must be complimentary to the application's main features.
- **GUI prototypes completed** (8/17 – 8/21). With the requirements firmly established, we can develop some ideas for the GUIs. This played an important part in the project. It gave the project the ability to appeal to the stakeholders. We were able to present the look and feel of the application. We could show how the application would flow from one screen to another. We can get feedback from the students and make sure they are happy with the feel of the application and what the application will be able to do. Any negative feedback voiced by the student testers was implemented into new designs for the user interfaces. It was a great way to keep the student body engaged with the project. We wanted to show them that we valued their opinions.
- **Business logic** (8/21 – 8/14). We defined the requirements in earlier milestones, and in this milestone, we looked at how to ensure the new application could use earlier features. We did not want to reinvent the wheel, only tweak features and the background code to work with the requirements of the mobile application.

- **GUIs developed** (9/15 – 9/22). As discussed in the GUI prototype milestone, we wanted to accomplish a few things with this milestone. We wanted to show an actual living and breathing user interface. Next, we wanted something to test the application's functionality. We wanted to be able to display the information for testing by the developers and by the student volunteers. We want the student body to continue with the role of a tester to ensure they still feel involved and get honest feedback about the features as they are implemented.
- **View academic record feature** (9/23 – 9/28). This milestone was one of the first components that would add some functionality. It was also used as a stepping stone for the developers. Since our developers had no exposure to this development platform. We thought it best to develop something that implemented all the concepts they learned in training. The developers eventually discovered that many of the components used to develop this feature could be used again if they modularized them properly. This would decrease the turnaround time, and the project will eventually be completed ahead of time. The developers and the students extensively tested this feature. Any deficiencies were addressed and tested again.
- **View class schedule** (10/1 – 10/4). This milestone would be a component needed for all the development milestones that would follow. This one feature would allow the student to view their schedule, but it also allowed the student to view changes like adding courses or deleting courses. The developers and the students tested this milestone. Any deficiencies were addressed and tested again.

- **Add course feature** (10/1 – 10 /4). This milestone was implemented so students could add a course to their schedule. The students tested this for ease of use, and any negative feedback was addressed in subsequent versions of this product feature.
- **Delete course feature** (10/05 – 10/10). This milestone was implemented so students could delete a course from their schedule. The feature was tested for ease of use by the students, and any negative feedback was addressed in subsequent versions of this product feature.
- **Servers installed** (10/11 – 10/12). This milestone marked the completion of the installation of the servers. This milestone could have been implemented at any time during the project, but we waited till the end just in case the application needed other requirements. The technician performed this function and marked a step closer to the application's deployment to the app stores.
- **Licenses obtained** (10/11 – 10/12). This milestone was important because, without the right license, we will not be able to deploy our mobile application to the app stores for Android and iOS. These stores require licenses to deploy to their stores. The university also wanted to get licenses at a slightly cheaper rate since we are a non-profit university entitled to a cheaper rate for our licenses. This cost-saving idea was mentioned in the previous paper.
- **Application deployed** (10/13 – 10/14). The application is deployed to the app stores for Android and iOS. The students began to use the application as they became aware of its availability.
- **Maintenance and update** (10/15 -?). This is the phase that the project is in now. We are now maintaining the application across all the platforms and actively listening to the

students for feedback to improve the application and fix any defects in the mobile application.

Project Deliverables

The project produced many deliverables as a work product. The project produced documents, the software code base, manuals, and other documents. The project produced documentation throughout its five phases. I will discuss what was produced in each phase.

Survey phase. In this phase, the documents created are the survey used to query the students about what they wanted in the application. This will aid in future application improvements as we have a pool of candidates to draw from to add more functionality. The student volunteers created a tabulated document that will aid in ranking what improvements will make next in the application.

Design phase. This phase produced items like the GUI prototypes for the various screens that will provide the functionality for the application. We will have documented feedback from the student testers. We will keep this for future reference and track the changes to the application over its lifetime. We will also create some preliminary testing documentation during this phase. These test documents will be added over the course of the project and the lifetime of the mobile application.

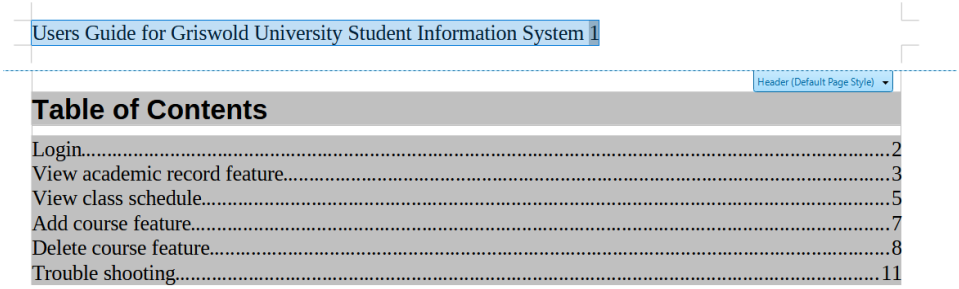
Development phase. This is where the code base is produced and finalized. We will have documentation from testing from the developers and the student testers. We will have the essential project deliverable – the completed mobile application.

Deployment phase. We have the licenses and any documentation created by the technician during the installation of the servers.

After the project, we will have abundant documentation produced as a result. The documentation will be discussed in more detail in the next section of the document.

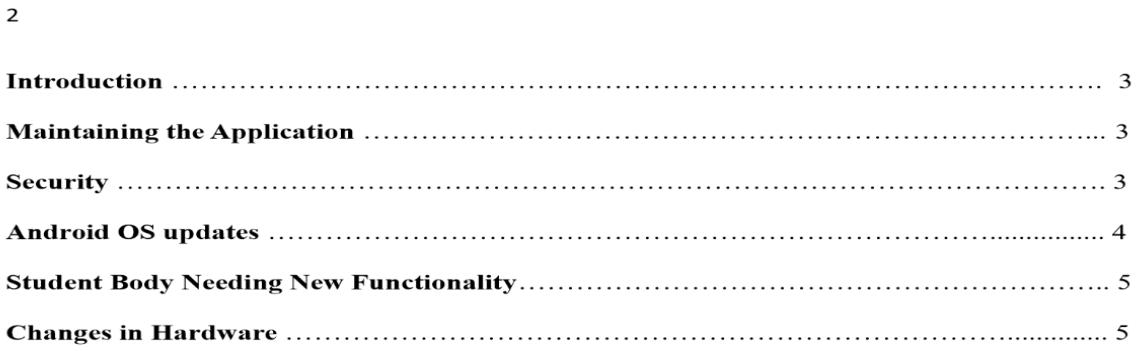
Documentation Deliverables

The project produced many types of formal documentation. The documentation included items like a user manual for the project that can be posted for the student on how to use the application. A sample of the table of contents page can be seen in the image below.



Screenshot Griswold University User's Guide

Additional documentation included with the project included a maintenance guide compiled by the IT staff who worked on the project. This guide provides information about how to add functionality to the application. You will also see how to handle updates to the Android OS and how the technician configured the servers to meet the project's needs. You can see a screenshot of the guide's table of contents below.



Screenshot Griswold University Maintenance Guide

Documentation about acceptance criteria has been created. A sample of this document can be found below.

Feature Name:	
Scenario:	Description of the scenario, aka feature
Given	The event that needs to happen
When	What has to happen
...	
Then	Final action

Acceptance Criteria Template

We have generated reports that documented the project over time. These reports included graphs that show the trend of work products being produced. We retained them for future reference.

In the next section, we will look at the evaluation results and any revisions needed to keep the project on track and ensure it stays on course and reaches its scheduled completion date.

Formative Evaluation Results and Revisions

During the project, we used many different types of measurement to see how the project was progressing. Most metrics were based on software development methodology since this project was mainly about developing a software solution.

Summative Evaluation Plan and Results

We needed to keep this project on track and on schedule as much as possible. We used the QA metrics discussed in this document's Formative Evaluation and QA Metrics section. These QA metrics were based on good software development practices. We could use the data from these metrics to view the progress and the efficiency of the project as it went on.

As described earlier, we can look at metrics that describe code production as the best metric of the project's progress. This was a software project, and using these metrics seemed most appropriate for the project. We used the following metrics cycle time, development velocity, change failure rate, development frequency, and PR requests (5 Software Development KPIs for a Savvy Engineering Leader, 2022).

We felt these metrics also allowed us to gauge how well the developers adapted to the new software development platform – Flutter and Dart. For instance, as time went by, cycle time decreased. Cycle time is the amount of time that it takes complete a work package. As the developers went on, this time decreased as they became more proficient with the platform, and they were able to streamline processes for creating code. They accomplished this in many ways. They came up with ideas on how to reuse code by changing it a little to meet the program's needs. These tweaks saved time and kept the project on and eventually ahead of schedule.

Since we are working with a relatively small group to complete the project, there was excellent communication among all parties. I, as project manager, was always open to suggestions from all parties. The most important parties in this project were the developers and students.

Neglecting the opinions of these individuals would hamper the project then and into the future. From the developer's perspective, I looked at their opinions on how the features were coming along. Did they feel the features were coming along at a good pace? Did they think we could add features at some point in time or during the current phase of the project? If they felt we could add a feature, then we did it. We wanted to ensure they were not overly tasked. Above all, we wanted to maintain the timeframe to deliver the mobile application by the proposed date – the end of October.

While we used metrics to measure the progress, their items aided in keeping the project on time and ensured that the application produced was quality in every sense of the word.

When we first started, we had the Survey phase. During this phase, we were able to get the requirements of the project. We did know the basic requirements, which were handled by the previous solution. But we went one step further by using the survey to see if we could add some of the wants of the students. This did not detract from the mobile application only added to it on many levels (2020).

We tried to create meaningful milestones that were attainable. With attainable milestones, we would have a much-needed component or a feature completed that we could use for other phases. When we got to the development phase, each feature would become a working part of the application. This aided in the completion in two basic ways. First, it gave the stakeholders a functional piece of the application, and second, it gave the developers more confidence in creating an application they had never done before (2020).

During a milestone, the developers created their own daily goals. They aimed to develop a specific part of the milestone by a given time. These could be considered mini-milestones, and they are the building blocks that would eventually lead to the completion of the milestone. They were an excellent way to understand progress at a particular milestone (2020).

To accomplish this, we had a very comprehensive stakeholder communication plan. In the next section, we will look at how each of the stakeholders was communicated with regarding the project.

Stakeholder Communication Plan and Reports

We developed a stakeholder communication plan to ensure the project was off to a good start. The chart below shows the stakeholders along with other information.

Stakeholder	Title/ Role	Interest: How much does the project affect them (1,2,3)	Influence: How much do they have (1,2,3)	What are the Stakeholder's Most Important Goals?	How will they Contribute?	Best Way to Manage
Matthew Heino	Project Manager	1	1	To stay on time and budget	Will be the daily lead, delegate project tasks	Phone Calls, milestones updates, weekly emails summaries of tasks, in-person meetings
IT Department (Main contact: Yolanda Cruz)	Head of the IT Department	1	1	Reduction in the resources used	Provide the IT resources to accomplish the project	Phone Calls, milestones updates, daily emails summaries of tasks, in-person meetings
Student Body (students)	NA	1	1	Mobile application for day-to-day tasks	Desired features	Email
Administration (Main contact: Priya Thomas)	VP of University	2	1	Reduction in costs and resources capital	Monetary human resources	Phone Calls, milestones updates, daily emails summaries of tasks, in-person meetings
Developers	Developer	1	1	Create the application	Code and testing	Phone Calls, milestones updates, daily emails summaries of tasks, in-person meetings
Technician	Technician	2	2	Complete server installation	Installation of servers	Email and phone calls

Stakeholder Communication Plan

We will look at how each of the above stakeholders was communicated with during the project. The first stakeholder was the project manager, a part-time manager for this project. The project manager contacted all the developers, the administration, and the technician. He was

instrumental in informing the administration about the project's progress and ensuring that the project funds would be available when needed. He would conduct in-person meetings when necessary to discuss any problem or concerns the stakeholder might have. This was important when trying to engage the administration and keep them involved and enthusiastic about the project. He notified the technician mainly about the status and for updates on when the technician would have the servers available to them.

The IT department was contacted through their primary contact Yolanda Cruz. This was done mainly through emails from the project manager or the developers. The IT department was primarily concerned with the status of the project. The IT department only provided the developers and the technician. They were contacted more often as the project reached the deployment phase. We needed to ensure the infrastructure was ready to handle the new application. We needed to ensure that the technician would be available during the deployment phase to install the servers and that they were not required to work on other campus projects.

The student body was communicated with mainly through email. These emails came from both the project manager and the developers. The project manager was responsible for informing them about the application’s progress. The developers would reach out to the student body when they desired feedback from student volunteers. We only needed to keep them involved so they could see what milestones were completed. We wanted to keep them in the loop because we would need volunteers to test the application and wanted the student body to stay updated on the completed features. We wanted to ensure they could voice an opinion on the project as it progressed.

The Administration was contacted by several members developers, the project manager, and the technician. Most of this communication was via email about the status of different

project phases and milestones. The project manager conducted in-person meetings with Priya Thomas and other interested parties to answer any questions about the project and address any concerns. These meetings were scheduled over the course of the project and when there was an urgent question or problem that needed to be addressed.

The developers conducted most of their communication by meeting with the project manager. They would meet in person or via phone to inform the manager of the progress and have any questions answered. Toward the end of the development phase, they would ensure that the servers would be available for deployment. They would contact the manager whenever a milestone in the development phase was completed so the manager could inform other stakeholders about the project's status. The developers contacted the only other group, the students. This was to get volunteers and provide the students with feedback after all their test feedback was considered.

The technician was contacted at the beginning of the project to inform them of the type of project and the skills needed to install the servers. The technician was informed via email as each milestone was completed, so they could have a timeframe for when they will need to install the servers. The developers and the project manager would contact them via email when they could begin the server installation.

In the next section, we will discuss what is and will be required for the ongoing support of the mobile application.

Ongoing Support and Maintenance

With the application deployed, the project did not end. We would begin the maintenance and update phase. This is an ongoing phase of the project. We will need to maintain the application and the infrastructure that supports it.

We deployed a mobile application, so we need to be cognizant of changes to the various operating systems. We will need to make changes to the application to keep up to date with new operating systems as they are released. With each new OS, functions will be added that may make our application easier to use. We should think about implementing them. We will need to rewrite some functionality as methods become deprecated. The application will need to be rewritten because of this change.

As we move forward with a mobile application, we need to be aware of threats that may be encountered through the application. To combat these threats, we must keep abreast of any known vulnerabilities to the OSs we have deployed to. Support needs to be given to the servers as well. Any new firmware or other upgrades must be applied to the server once they are known. Waiting could have disastrous consequences.

We need to keep in touch with the student body to ensure that the application performs the tasks they need and in the manner they need. We expect that this application will undergo many modifications. These modifications will enhance the application to provide new or better functionality or the required security.

We will implement some ideas discovered during the survey phase as time and finances permit. Some of these were added as complimentary features during this project. Others need to wait till other versions of the software are released. There were many good suggestions for features, but not all of them could have been implemented in the time allotted.

You will find a proposed support and maintenance plan showing some items that will be maintained over the application's lifetime.

Item	Start Date	End Date	Comments
Server Updates	10/15	Ongoing	Every month and as needed. Updates are based on changes in the application's requirements or to meet security concerns.
Feature Improvements	11/1	Ongoing	It will commence as time, and financial resources allow.
Survey of Students	NA	NA	Will be conducted when there is a noticeable increase in negative feedback about the application's functionality.
Review of Code	Ongoing	NA	It will be conducted whenever there is a new mobile OS release.
Hardware Maintenance	12/1	NA	Depending on the diagnostic software data, it will be conducted monthly or less.

Support and Maintenance Plan

Most of the items in the plan are ongoing with no actual end date. These items will be conducted the whole time the application is active. We have given a rough interval on when these activities will begin, what intervals they should occur, and any triggers that may change the timeframe.

Resources for the maintenance and support plan will be discussed in the next section of the document.

Resources for Post-Implementation Support

Resources will be needed to maintain the application and items from the previous section. This section seeks to break down what resources will be required. This section will try to break down the resources in terms of finances and human resources.

Server Updates. The server updates will be ongoing. The timeframe will be at least once a month and as needed. The servers will be updated to patch any security vulnerabilities and to update the servers to meet the needs of any changes in the application. There may be a

need for hardware replacement, but this will not happen immediately. The human resource that will be needed will be the technician. The technician will use their hourly rate, which is currently $25 per hour. Any hardware expenditure will need to be approved, and the costs will be variable based on the component needing replacement or upgrade.

Feature Improvements. Feature improvements will be conducted during the lifetime of the application. There are no set dates for these improvements. The upgrades will be undertaken when there is a need and the funds are available. Improvements that are required to meet security or defect in the application will take precedence. The resources that will be needed for this item will be the developers and the students. Students will be used to test the functionality of the improvement.

The developers will use their regular hourly rate of $37.50 an hour to create these features. Another project may need to be initiated if the improvement is very complicated. Suppose the improvement can be created in about ten days. In that case, this improvement will not need a new project but only approval from the administration for the time and resources involved in creating the improvement. The exact time to complete the improvement will vary, so no definite timeframe can be stated.

Survey of Students. This survey will only be conducted when there is a need to do so. There is no set timeline for this activity. We will conduct this as needed. This will be in response to feedback that students have left in the app stores or if there is a noticeable deficit in the application's functionality. The time to complete this will be the same as the survey phase of this project. The only omission is that we will not need the survey processor to create the survey. We can use the current one as it will address the same questions. We can still use the students to

survey the student body and process the data. There will be no actual cost with this item from the maintenance and support plan.

Review of Code. This item will be conducted whenever changes in the OS have been released or when a new OS enters the market. We want to make sure that the code meets the standards of the new OS, and we want to look for additions to the OS that may make it easier or more secure to perform different application functions. The developers will be responsible for keeping up to date about releases for the OSs. They will bill the standard rate of $37.50. The time to accomplish this task is variable. The cost will need to be approved by the administration.

Hardware Maintenance. This item will be conducted at least once a month or as often as diagnostic tests deem appropriate. The time to accomplish this will vary depending on the item being maintained. The technician will bill this at the standard rate of $25 per hour. Hardware that needs to be replaced will require approval from the administration, and this cost will vary.

Short and Long-Term Maintenance Plan

We developed both a short-term and long-term plan to maintain the mobile application. We will look at a few areas for developing a maintenance plan for the mobile application. These types will be corrective, preventive, risk-based, and condition-based maintenance. Each of these types seeks to address a key aspect of the application. Corrective maintenance looks to correct errors not discovered during application testing. Preventative maintenance looks to prevent errors or unexpected functional events from occurring. Risk-based seeks to look for threats to the application. Conditional maintenance is triggered when certain events within the application happen (*4 Types of Maintenance Strategy, Which One to Chose?*, 2022).

Short-Term Plan.

Item	Start Date	End Date	Comment
Corrective Maintenance			
Correct defects that were not found in the initial testing of the application.	12/15	11/15	This is an ongoing part of the project. It will commence when a defect is found in the code that has been released to the students.
Server's firmware	12/1	11/1	Update the firmware servers and other hardware.
Preventative Maintenance			
Servers are patched	12/2	12//2	Security patches applied to the servers
Development platform updates	12/4	12/6	Updates to the development platform
Risk-based Maintenance			
Review applications for known vulnerabilities	12/7	12/12	To review the application code for known vulnerabilities.
Rework code to account for vulnerabilities	12/12	12/30	Rework the code to account for the vulnerabilities and test again.
Condition-based Maintenance			
Update code based on negative student feedback	10/15	Ongoing	The code will be updated based on any negative feedback from the students in the app stores.

Short Term Plan

This is a short-term plan to maintain the application just after release. We are mainly addressing the defects not caught in the initial testing during the project's development phase. Our long-term plan will follow a similar layout and use the same types as the short-term plan.

Long-Term Plan.

Item	Start Date	End Date	Comment
Corrective Maintenance			
Correct defects that were not found in testing the application.	10/15	Ongoing	This is an ongoing part of the project. It will commence when there is a defect found in code that has been released to the students or as new features are released.

Servers firmware	First of the month	NA	Update the firmware servers and other hardware. Will be conducted on the first of the month or as needed if no firmware updates are available
Preventative Maintenance			
Servers are patched	The 15th of the month	NA	Security patches will be applied to the servers on this date if they are available.
Development platform updates	As needed	NA	Updates to the development platform as needed and available
Risk-based Maintenance			
Review applications for known vulnerabilities	End of month	NA	To review the application code for known vulnerabilities periodically.
Rework code to account for vulnerabilities	As needed	NA	Rework the code to account for the vulnerabilities and test again. It will be conducted only if the previous maintenance task finds issues.
Condition-based Maintenance			
Update code based on negative student feedback	Every other month starting two months after deployment	Ongoing	The code will be updated based on any negative feedback from the students in the app stores.

Long Term Plan

This is the long-term plan for maintaining the application over the course of the application's life cycle. We can add features during the condition-based maintenance part of the plan. We will look at features that may have come up in the student's feedback about the application and will take the opportunity to add them at that time.

Post-Implementation Project Summary

Overall, the project turned out very well. The project met all the desired outcomes. These outcomes were a reduction in the students' resources to access their information. They no longer

need to use the university's resources to access this information. They can use their mobile devices and the new application to accomplish this.

They are happy to engage in other activities not afforded them using the previous solution. The students also gained an understanding of how the application was created because they were directly involved. They saw it start as an idea on paper, and through their input, they began to have a vested interest in the application's success. They had their opinions listened to, and I think they appreciated that. They also added valuable input on what needed fixing and deficiencies in the previous solution.

The IT department no longer allocates as many campus resources to service the student's requests. They can now use these resources for other campus activities. They no longer have to maintain the kiosks that were aging and no longer easy to obtain affordable parts for. The maintenance is made more accessible since technicians no longer need to keep a larger percentage of computers running. We have the labs to service academic requests, which are more than sufficient to handle the current load. Excess computers can be allocated to other campus needs, like research opportunities, as discussed in the previous paper. These resources are now available for use.

The administration saw a cost reduction. They no longer need to budget for the maintenance of the two computing systems. They no longer had to budget for hardware for the kiosks, which was becoming increasingly more difficult to source, and the price for the hardware was increasing as it became scarcer.

The mobile application solution addressed all these outcomes in a manner that was cost-effective and in a reasonable timeframe. We left the opportunity to add the application as time and funding permit. We chose to use in-house talent to develop the application. This will pay for

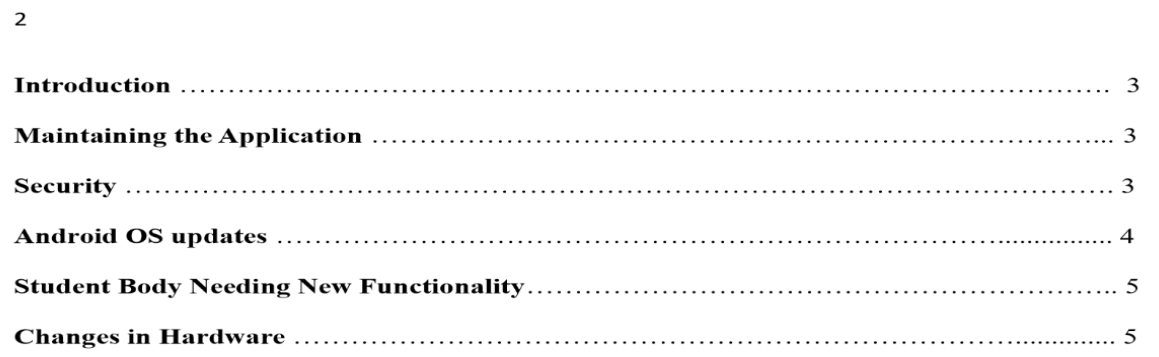
itself many times over. We now have the talent to create additional features and maintain the application. We would not have this ability if we outsourced the application's production to an outside firm. We save money anytime we need to add or fix something for the application.

To summarize, we were able to satisfy all the stakeholders by completing this project. We solved the inefficient use of campus computing resources. We saw a reduction in the administrative overhead that the IT department dearly wanted. The project accomplished the students' desires by providing a mobile application that performed the tasks the on-campus solution provided but allowed them to be mobile. The administration saw reduced costs by removing the kiosks and their expense and a reduction in labor to maintain the university's computing infrastructure.

Documentation Deliverables

This project produced many documentation deliverables. The documentation was discussed in a previous section, but we should explain what each deliverable means to the project now that it is completed.

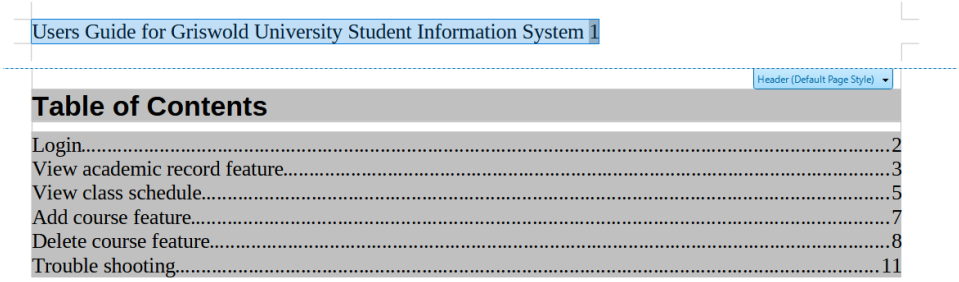
One document that will be important to the project will be the Maintenance Guide. A screenshot of the guide is shown below.



Screenshot Griswold University Maintenance Guide

This document is essential because it lays out the framework for maintaining the application for the foreseeable future. It will describe how to go about adding new features to the application. The guide discusses what will need to happen if there are updates to the OS that the application is deployed to. It will have suggestions on how to keep the application secure. It will have a troubleshooting section that the developer put together to aid in troubleshooting certain parts of the application. During the application's useful life expectancy, this document will be amended and added to as needed. We want to document as much of the project as possible.

Another helpful document that was produced was the User's Guide. We produced it to inform all parties as to the functionality of the application. It was also meant to save calls to the help desk. Students could look up the tasks here and only call the help desk if they still have problems. This document was also provided to the help desk associates, so they have a record to study to help them support the application. A screenshot of the user guide is shown below.



Screenshot Griswold University User's Guide

We developed a few useful items that may be beneficial on this project but on any future application development project. We developed an acceptance criteria template and a test case template. The test case template allowed us to devise reusable test case scenarios, and we came

up with a format that could easily document the test case and give useful information about the test case. An example of the test case documentation is given below.

Project Name: Griswold University Software Solution						
Test Case Id: GUI_1			Test Designed by: Steve Mitnick			
Test Priority (Low/Medium/High): High			Test Designed Date: 8/3			
Module Name: GUI			Test Executed by: Steve Mitnick			
Test Title: Test Login			Test Execution date: 8/5			
Description: Test the login screen of the application						
Precondition: None						
Dependencies: None						
Step	Test Step	Expected Result	Actual Result	Status (Pass /Fail	Notes:	
1	Enter information into fields	The field accepts the text	Field accepted text	Pass		
2	Press Enter button	Able to press enter button	The button is pressed successfully	Pass		
3						
4						

Test Case Sample Document

These documents will aid in creating an archival history of the project for testing and development purposes. Other documents that were made would be the tabulated data from the survey phase of the project. We will keep this and refer to it often as we add new features over the next few months or years.

Success Criteria for Each Project Outcome

Success criteria for the project had to be viewed from the vantage point of each stakeholder. We needed to address the needs of very disparate stakeholder groups. One group, the student body, only looked at the application as to what it could do for them and whether or not they could use this application on the go.

With this application, we were able to meet this outcome and provide an application that the students can use at any time and any place they want. They no longer had to use the campus infrastructure to accomplish their activities. They can achieve them on the go and using the mobile devices of their choice. We could easily see this as a successful outcome since using the dedicated campus resources saw decreasing usage. This was especially true of the kiosks. Within the first month and a half of application deployment, kiosk use dropped to almost zero. With some insight, the kiosk use would be zero if students did not forget their mobile devices at home.

We developed a plan to keep the students involved in the application creation process. We valued their input while they tested the application at various stages of production. We tried to incorporate this feedback to improve the product to meet their standards. The students, after all, will be the ones using the application, and we need to ensure that they are happy with all the features the application will have. With the proposed Agile framework for developing the application, we sought out and got an application accepted by the student body.

Another outcome of this software solution was a reduction in the resources used by the university to process the students' requests for information. We did see a drop in the use of resources. As stated earlier, kiosk use dropped to almost zero. We have a demonstrable decrease in using other earmarked computers to service student requests. After deployment, we began to withdraw the computers from servicing student requests and removed the kiosks since they were

no longer useful to the student body. The IT department devoted resources to the kiosk dismantling, but only if there was time to do so since there was no pressing priority to do it. The computers were removed, placed into storage, or pressed into service for other campus activities.

The Administration wanted to see a reduction in costs as a desirable outcome. They wanted to see that the new solution would do that. Since the students no longer used the kiosks, they no longer had to budget for maintenance. They no longer had to procure new computing equipment for some campus activities as these resources could be sourced from the no longer used computers from the previous solution. This saves the administration money in terms of procurement costs. Now the only time the administration needs to spend money on computing equipment is for specialized equipment used for intensive academic research.

Justification of Proposed versus Actual Project Outcomes

The project mainly went according to plan. As discussed in a previous section, we were able to complete the project a little ahead of schedule. The last schedule was a little on the cautious side. I factored in a steep learning curve. I wanted to give the developers ample time to learn the new platform, but that time was unnecessary. Some phases took the expected time. The survey phase took the allotted time and did not experience any hiccups.

We were able to meet or exceed the proposed outcomes. We brought the project slightly under budget since we could shorten the development period. This was due to the developers quickly becoming very adept at the new platform.

Lessons Learned

This project has quite a few lessons learned. We were able to develop and refine processes for creating code. We developed documentation templates that could drastically shorten the time to document steps, like creating acceptance criteria and test case documentation.

The developers, over the course of the project, learned to document everything for future reference. This will aid in creating new features since the developers can refer to these "notes." The developers learned through multiple iterations of the same feature how to make that code better but apply this knowledge to make other code features better. The developers also realized they did not need to say yes to every requirement in the student's feedback. The developers sifted through them and only added bona fide features. Features that were useful and practical (Quint, 2022).

Lessons learned from this project will make the future project easier to undertake. We have learned various skills that will aid in creating new software for the university. We learned a unique skill set that can be applied to developing new software applications. We created new documents that can be used and added to as the new features are added or can be used on other projects. For example, the templates created can be used on other projects to help document the project.

This is a summation of the lessons learned by undertaking this software solution. As the maintenance and update phase continues, we will continue to learn other lessons that can be applied to other projects at the university.

References

4 types of maintenance strategy, which one to chose? (2022). ABB. https://new.abb.com/medium-voltage/service/maintenance/feature-articles/4-types-of-maintenance-strategy-which-one-to-choose

5 Software Development KPIs for a Savvy Engineering Leader. (2022, October 20). LinearB. *https://linearb.io/blog/5-software-development-kpis/*

Editor. (2019, December 9). *Acceptance Criteria: Purposes, Formats, and Best Practices*. AltexSoft. https://www.altexsoft.com/blog/business/acceptance-criteria-purposes-formats-and-best-practices/

Hamilton, T. (2022a, November 5). *Software Testing Methodologies: Learn QA Models*. Guru99. https://www.guru99.com/testing-methodology.html

Hamilton, T. (2022b, November 19). *GUI Testing – UI Test Cases (Examples)*. Guru99. https://www.guru99.com/gui-testing.html

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)–Fifth Edition* (Fifth Edition, Fifth edition).

Quint, A. (2022, February 26). *10 Lessons I've Learned in 10 Years of Software Development*. Medium. https://levelup.gitconnected.com/10-lessons-ive-learned-in-10-years-of-software-development-113b5cd9910a

SeaLights. (2019, November 7). *Understanding Agile Testing Methodology and 4 Agile Testing Methods*. Sealights. https://www.sealights.io/agile-testing/understanding-agile-testing-methodology-and-4-agile-testing-methods/

Software Development Standards: ISO compliance and Agile. (n.d.). *https://www.softkraft.co/software-development-standards/*

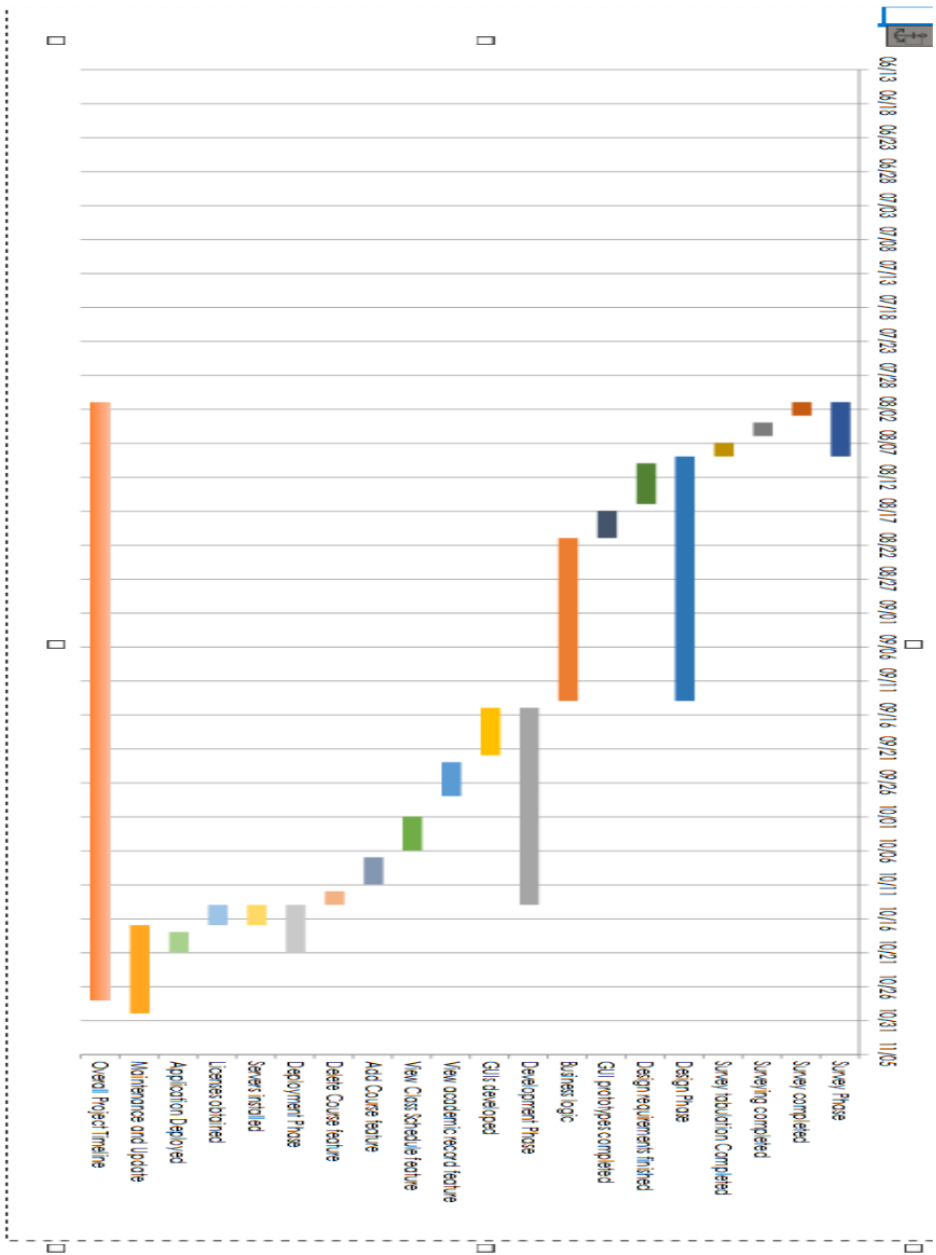
Software Testing Help. (2022, October 25). Software Testing Help. https://www.softwaretestinghelp.com/test-case-template-examples/

(2020, July 10). *Keeping on Track: 14 Project Management Tips*. Project Management Articles, Webinars, Templates and Jobs. https://www.projecttimes.com/articles/keeping-on-track-14-project-management-tips/

Project Timeline

Phase	START DATE	END DATE	DURATION in days	COMMENTS
Survey Phase	08/01	08/09	8	
Survey completed	08/01	08/03	2	milestone
Surveying completed	08/04	08/06	2	milestone
Survey tabulation Completed	08/07	08/09	2	
Design Phase	08/09	09/14	36	
Design requirements finished	08/10	08/16	6	milestone
GUI prototypes completed	08/17	08/21	4	milestone
Business logic	08/21	09/14	24	milestone
Development Phase	09/15	10/14	29	
GUIs developed	09/15	09/22	7	milestone
View academic record feature	09/23	09/28	5	milestone
View Class Schedule feature	10/01	10/06	5	milestone
Add Course feature	10/07	10/11	4	milestone
Delete Course feature	10/12	10/14	2	milestone
Deployment Phase	10/14	10/21	7	
Servers installed	10/14	10/17	3	milestone
Licenses obtained	10/14	10/17	3	milestone
Application Deployed	10/18	10/21	3	milestone
Maintenance and Update	10/17	10/30	13	An ongoing phase
Overall Project Timeline	08/01	10/28	88	

Tables


Survey Phase
Survey completed
Surveying completed
Survey tabulation Completed
Design Phase
Design requirements finished
GUI prototypes completed
Business logic
Development Phase
GUIs developed
View academic record feature
View Class Schedule feature
Add Course feature
Delete Course feature
Deployment Phase
Servers installed
Licenses obtained
Application Deployed
Maintenance and Update
Overall Project Timeline
04/13
04/18
04/23
04/28
07/03
07/08
07/13
07/18
07/23
07/28
08/02
08/07
08/12
08/17
08/22
08/27
09/01
09/06
09/11
09/16
09/21
09/26
10/01
10/06
10/11
10/16
10/21
10/26
10/31
11/05