Project Management Plan

Employee Tracker
COP 4331C, Fall, 2017
Team 6

Team Members:

- David Simoneau
- Navon Francis
- Brandon Bradley
- Jordan Martin
- Marc Simon
- Ryan Hoeck

Project Overview

This project is the Employee Tracker App. It is a design to aid companies in keeping the employees safe and creating an efficient work atmosphere. Our design will be modern with all the bells and whistles that the customer needs and wants, without getting in the way of the user interface. We know there are concerns with employee tracking, but our app gives the power to both the companies and their valued employees. We are aiming to keep the product low-cost, while exceeding current systems already in place.

Applicable Standards

Our Coding Standard will be heavily influenced by having clean-looking, well-formatted, and maintainable code. However, we all agree that we will not compromise functionality and maintainability for “pretty code”. These characteristics will be heavily influenced by the Google Style Guide (cited below). We will stick to using tabs instead of spaces for indentation. Our naming conventions will be descriptive in the sense that each method will describe exactly what will be accomplished, but won’t reach extremely large lengths. Code will be commented throughout the process, this is so that other teammates will be up to speed and can help in different areas of the application. Documentation will be kept in the Git repo and as well as our local machines. We will have technical documents explaining the requirements that must be met in our application. README’s will be created, describing what is in our Git repo. We will be stressing Object Oriented design patterns.

github.com/google/styleguide

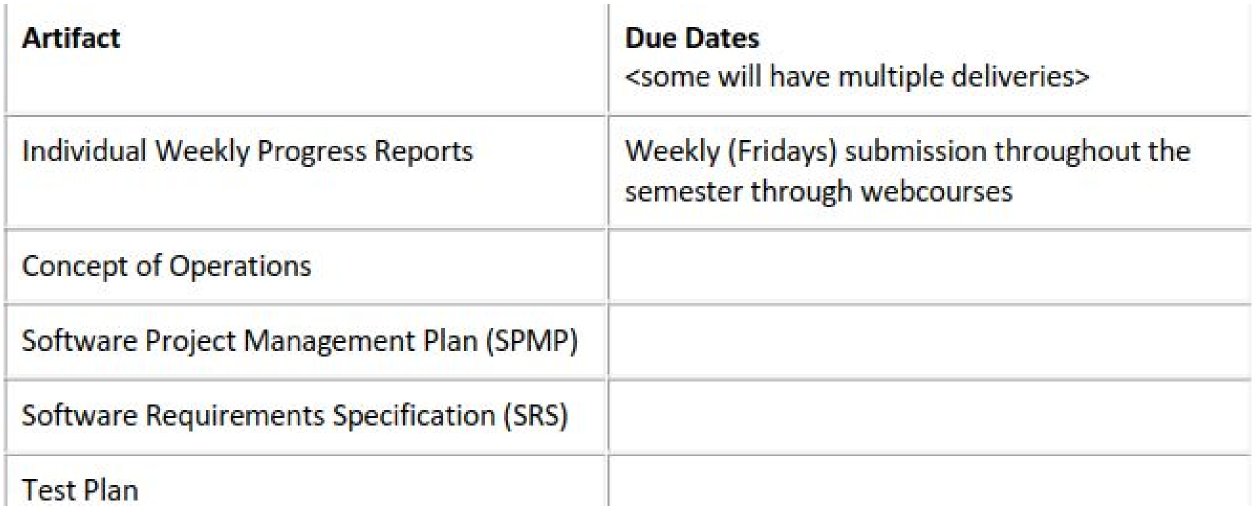
The Artifact Size Metric Standard for our application will be broken into 2 sections. We will have documentation and code. Our documentation will cover various aspects about the application. This section will be concise with information that pertains to the technicalities of the application. Our goal is to have 7-10 pieces of documentation to correlate with our project. The code itself will be heavily done in each respective corresponding React component. The will be around 7-10 main files of code (there will be many other files but these are auxiliary React files.)

Project Team Organization

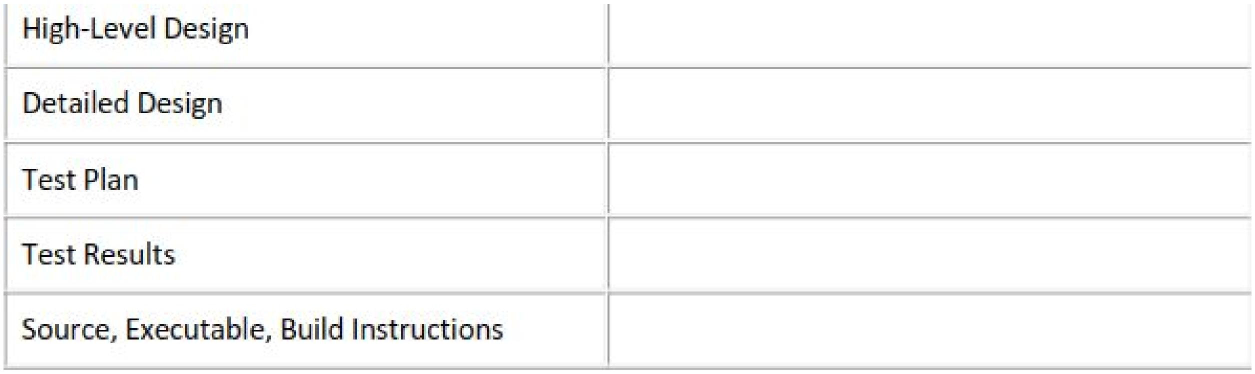
Our team consists of David Simoneau, Ryan Hoeck, Navon Francis, Brandon Bradley, Jordan Martin, and Marc Simon, Thus far, our team has been very efficient with our time. As soon as we were assigned our team, we established a good medium for communication, which was Slack, and started conversing and throwing ideas around. Sharing the work has not been hard because we’ve only had to write documentation for ConOps and now this, the Project Management Plan. When we begin coding, we have discussed using a git service and we are leaning more towards BitBucket because of their private repositories that they offer. As far as the role of project manager, the role has yet to be needed. Yet, if need-be, I’m sure we can solve the issue democratically, and it wouldn’t pose a problem.

Deliverables

Artifacts	Due Dates
Concept of Operations	September 27th
Software Project Management Plan	October 5th
Software Requirements Specification	October 15th
Software Design Description	October 23rd
Peer Review Evaluations	November 3rd
Test Plan	November 13th
Software Acceptance Test	December 1st
Code Submission	December 1st



Artifact	**Due Dates** <some will have multiple deliveries>
Individual Weekly Progress Reports	Weekly (Fridays) submission throughout the semester through webcourses
Concept of Operations	
Software Project Management Plan (SPMP)	
Software Requirements Specification (SRS)	
Test Plan	



High-Level Design	
Detailed Design	
Test Plan	
Test Results	
Source, Executable, Build Instructions	

Software Life Cycle Process

Our team will be following the Agile - Scrum methodologies and processes. Most of us are already familiar with Agile so using this methodology With Agile we take the approach of making stories with points that weigh how important something is and how long it is going to take. We use a Scrum board to track the progress of each of our modules that need to be completed. We have 3-5 “stand-up” type meetings a week (more realistic than 5 days a week) where we talk about what we accomplished yesterday, what we will do today, and if there was any roadblocks.

Tools and Computing Environment

For our application, being both mobile and web friendly, we’ve established that we are mostly going to use React.js and Python. These are great programming languages for what we are developing due to the fact that they work on all modern operating systems. Since we will all be working from a git account, our environments will be different, but Ryan has a home server that he is going to set-up so we can host and test all on the same system. Our production will most likely be live on the server that everyone has access to mostly because we won’t have any confusion on testing environments. As far as databases go, we’re planning on using postgres due to the SQL language being fairly known among the group and it’s easy to use. The testing OS will probably be an Ubuntu server with apache being run on it as well as the necessary dependencies for our languages, dev-tools, and everything else we might need. In the end, the plan is to have the application friendly for all smart-phone devices and web browsers so our customers will have an easy experience.

Configuration Management

For our version control we are using Github to manage and control edits to the project code. By merging Github's capabilities with Slack, our communication app, we'll be able to discuss and apply responsibilities as needed. Slack and Github are both common development tools that students in the group are familiar with, or are capable of learning and using.

Quality Assurance

Test-driven development will be on everybody's mind as they code. We plan to automate casual-usage as well as any edge-cases we can think of as we are developing the app; that way our automation can detect if old code breaks when new features are added. Thus, this process will happen continue to happen each and every time we are coding and push a feature to the master branch of our code. Through Slack, we will have a chat room specifically for any bugs to notify each other and use the Bitbucket Open-Issues feature to track our bugs. This way, development doesn't have to stop, but we can stay on top of the issues without losing track.

Risk Management

Our biggest potential risks for this project include communication, scope, and resources.

Communication is a potential risk to this project that could impact the final outcome of the software. The team needs to clearly understand the requirements so that they understand what the user expects as a final product. The team also needs to clearly communicate thoughts, concerns, and instructions so as to not assume that the team already knows any of these. Under-communication will cause a disconnect which will most definitely affect the final product. To address this risk, the team will be creating a Software Requirements document that will specify all requirements that the user is expecting. In addition, the team has set up an online workspace where the team can communicate easily and ask questions if there is any sort of confusion with a certain task.

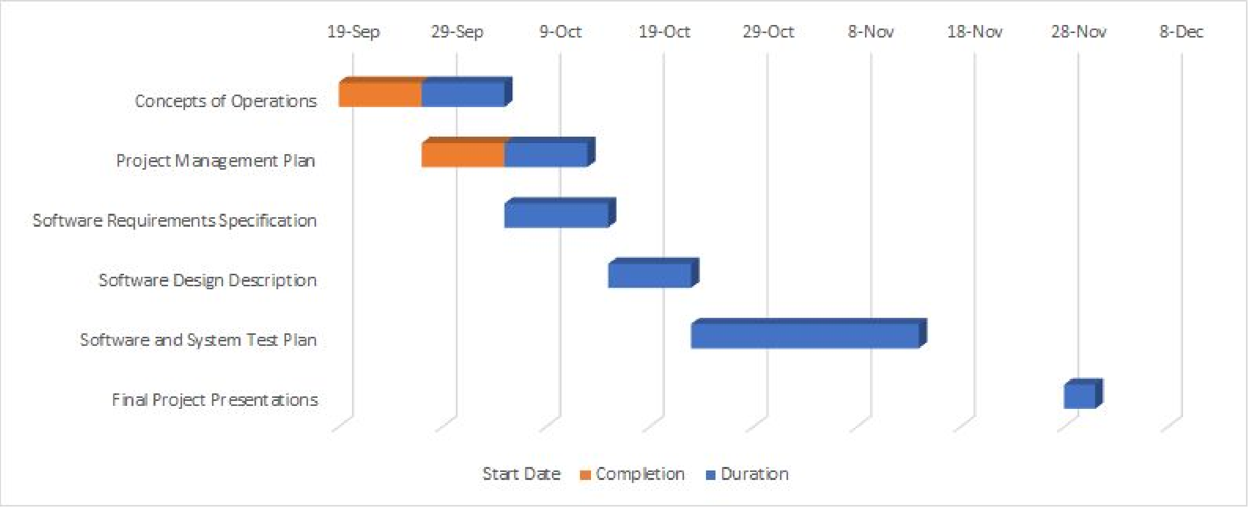
Scope is another potential risk to the project's success. If the scope is not clearly defined, the team may end up producing a product that does not include all the functionality the the user needs. At the same time, the team may also end up producing a product with additional features that are not needed and may clutter the total outcome. Additionally, if the scope is ill-defined, estimates and dependencies may be inaccurate. This can cause slippage and possibly even a failed project. To address this risk, the team will make sure to define the scope of this project when listing out our Software Requirements. This way, we have a scope that aligns with the detailed tasks that need to be completed. Once our scope is defined in terms of our requirements and the intended functionality, our team can work on setting realistic estimates with as much detail as possible. Dependencies will be addressed ahead of time to make sure that they do cause issues for us later on in development.

The resources that the team has available can be another potential risk. Resources that may be in high demand of our team include working hours, training, software licensing. Also, team motivation and commitment may also be considered an in-demand resource. In short, our team may end up surprised when we cannot complete a certain task in time due to missing some of these items if we do not plan ahead. To address this risk, we will note when members of our team can work so that this information is publicly available. We will also choose software that will be most familiar to all members of the team as to minimize training time. This software will also be free to obtain a license for as we not have funds to afford commercial software. Lastly, our team motivation and commitment will a priority among all members, as we plan to communicate on a daily basis over the project's details. If we do lose this, the leaders of the team will make sure to keep our motivation and commitment in check.

Table of Work Packages, Time Estimates, and Assignments

Task Description	Time Estimates (hours)	Actuals (hours)	Responsible
Firebase Database	**3**		**TBD**
Web Application	**10**		**TBD**
Mobile Application	**10**		**TBD**
GPS location services (API)	**5**		**TBD**
Software Requirements	**10**		**Whole Group**
Software Design	**20**		**Whole Group**
Software and System Test Plan	**10**		**Whole Group**
User Manual	**2**		**Whole Group**

GANNT Chart



Technical Progress Metrics

For technical progress we chose to track by section for the Concept of Operations and the Project Management Plan documents. Each section was assigned to a member of the software engineering team and completed on time as of October 5, 2017. The requirements phase which includes the Software Requirements Specification will be tracked by the total requirements completed against the total requirements outstanding. This type of metric is most useful to our team as it has worked with success with our previous deliverables and ensured total participation.

As each major software requirement will have an associated UML diagram we will track the progress in a similar manner by comparing the total UML diagrams completed against the UML diagrams outstanding. This will ensure we have a easily reportable and verifiable metric to show timely progress is being made towards project completion. For the Software Design Description there are three major components to our project to be designed and implemented. They are the backend database, the frontend mobile application and web application.

To ensure each team member can play to their strengths we will split the group into sub teams assigned to these tasks. Each sub team will report to and work with the entire team towards completion of their assigned task.

As the project moves to the detailed code portion of the project we will track integration of the various Google Maps APIs, number of completed packages, classes and methods against those that are outstanding. The number of database operations including corporate account creation and deletion, admin and employee logins including creation and deletion, coordinate data storage amongst others which are specific to our project will be tracked.

For software operation metrics we will track query execution time on the database which will have direct consequences for the web and mobile application speed. Memory usage, CPU usage and other profiling metrics on mobile will be tracked which are very integral to having a smooth application on devices with limited resources. We will make sure to limit the number of API calls to certain services such as Google Maps to ensure end users are not experiencing excessive battery drain on their devices.

We envision that a project of this scope will have complex code that must utilize wrappers for numerous services and we will mitigate code complexity risks by thoroughly commenting and writing understandable code.

Plan for tracking, control, and reporting of progress

Each team member will submit their weekly individual progress report through the report channel on Slack which will link to a Google Docs document. The individual weekly report should include time spent on assigned tasks and include a breakdown of the scheduled tasks and the progress made on them. The weekly report should include any time constraints the team member faces and a tentative completion date of the assigned task.

Each week each team member will read each of the individual weekly progress reports and posit any questions through the numerous communication channels. The team members will examine the completed tasks, outlined progress metrics and reassess any project risks.