2019

Hofstra Campus Map Application

PROJECT PLANNING

J ABRAHAM; F. CHEN; X. CHEN; X. HU; W. SANTOS

Contents

1. Introduction

This document gives an outline of the overall project plan, introducing a general workflow based on the design specifications and module system created for the application. It is divided into 5 sections: Risk Analysis, Resources Requirements, Work Breakdown, Project Scheduling, and Project Monitoring. It is heavily derived from the Requirements Specification Document. Please refer to the newest version of that document for further information.

2. Risk Analysis

1a. Risk Metrics

Assessing risk will be critical to the overall success of the project, allowing the development team to be made aware of potential areas of interest. It will be quantified in three parts:

- Impact (**I** : 1-3)
    - Negligible(1) - Considerable(2) - Critical(3)
- Likelihood (**L** : 1-3)
    - Remote(1) - Possible(2) - Likely(3)
- Risk (**R** : Impact x Likelihood)
    - Low(1-3.9) - Moderate(4-6.9) - High(7-9)

1b. Analysis of Major Risks

Risk (I | L | R)	Risk Type	Analysis
Leadership Change 2 | 1 | 2	Organizational	The project leader changing, though highly unlikely, will lead to uncertainty for the team as a new management style takes over. This can lead to disorganization as the team reorients itself around a new leadership. Although serious in the short-term, team cohesion will most likely stabilize in the long term.

Team Members Incapacitated
3 | 1 | 3
	Personnel	Although the impact is critical, the probability that team members are incapacitated is low. The following key personnel have been identified as especially critical to project success:
		• Application Architect
		  • As the person in charge of organizing the structure of the application, this individual is critical to the successful development of the system.
		• Project Manager
		  • As the person leading the team, driving efficient workflow, and ensuring that tasks are met, the loss of this individual will hurt overall team productivity.

Project Complications Force Design Changes 3 | 2 | 6	Requirements	If complications to the project arise, either from module delays or from Technology problems, it is entirely possible that Requirements will have to be reevaluated and altered. Such changes will impact workflow on a critical level since time will need to be alotted to reorganize, reorient, and replan for these changes.
Modules Are Delayed 3 | 2.5 | 7.5	Estimation	Due to team inexperience with scheduling, it is highly likely that modules will be delayed. This becomes more serious for critical modules.
GUI 3 | 3 | 9	Technology	Due to inexperience with developing a UI for the GPS system, displaying the route on a map will prove to be a very difficult challenge to overcome.

1c. Risk Mitigation

Risk can be properly mitigated given proper warning and preparation. The above low-moderate risks can be mitigated given time and proper focus. As such, delays will occur, which is the main critical risk identified. As such, a buffer time of approximately 15 days has been alotted to account for module delays and unexpected events.

2. Resource Requirements

This section highlights the resources needed to complete the software. They are organized into two sections, Software & Hardware.

2a. Software Resource Requirement:

- **Android Studio: 2019 (3.3.2):** This will be the primary editor that will be used among the development team to eliminate the probability of errors occurring between different editors.
- **Java Programming Language:** The application will use Java as the primary programming language.
- **GitHub:** GitHub will be the primary collaborative tool for the development team, with its main functions being version control and integration.
- **Google Drive:** Google Drive will serve as a repository for all documents written by the development team due to its immediate interconnectivity through Google Docs.
- **Trello:** For Project Monitoring and Tracking, Trello will be used to assign tasks and ensure that everyone is aware of their respective tasks.

2b. Hardware Resource Requirements

- **Android Device:** To ensure that the application is optimized for Android devices, there must be at least one Android Device available for proper testing.

3. Work Breakdown

The application will be divided into five modules and one submodule. The Module Development Days (MDDs) assume an idealistic 8 hours of development time per day, ignoring realistic working schedules. Furthermore, each module will be given one day after completion for simple testing which is NOT factored in for MDDs. Please see Section 4 for the realistic Project Scheduling that considers the aforementioned factors.

3a. Breakdown

Phase I: Initial Development

Module 1: Nodes in Map (2 days) **Conrad/Franklin/William**

- Structure nodes(building info)
- Path node(pathing)

Module 2: Map System (4 days) **Conrad/Franklin**

- Connect .json to Map
  - Parse files
  - Build local copy of graph

Module 2.5: Simple GUI (2 days) **Conrad**

- Visual representation of map

- Design

Module 3: Pathfinding (1 day) **Conrad / Franklin**

- Develop Search Algorithm
- Determine distance between nodes
- Sort/search for nodes

Module 5: GUI (4 days) **Jijin, Xin**

- Upgrade UI
- Event listener

Phase II: QA Testing

3b. Module Dependencies

The figure below shows the modules, their duration, and their dependencies. As there is very little branching, the Critical Path is clear. The Critical Path, M1→M2→M3→M5 will take 13 Module Development Days to complete or 104 hours of development at minimum. Remember that Module Development Days does **NOT** account for realistic scheduling as stated above. That will be addressed in the next section.

Activities	Duration(Days)	Dependency
M1	2	None
M2	4	M1
M2.5	2	M1
M3	1	M2, M2.5

M5	4	M2.5

4. Project Scheduling

The application will be scheduled with Schedule Development Days (SDD) as its unit. In general, SDDs are calculated by multiplying Module Development Days by two. This is because SDDs assume a more realistic development timeframe of 1½ - 4 hours per day. Note that some modules are given extra time due to their complexity and that each module is given 1 extra day after completion for debugging and testing. When taken to account, the Critical Path, M1→M2→M3→M5, will take at approximately 30 SDDs to complete or 120 hours.

4a. Timeline

Note that there will be weekly meetings on Saturday at 3PM.

Phase Ω: Research, Planning, and Preparation

Feb 20th - 22nd: Study the notes and language required

23rd: Weekly meeting

25-28th: Setup the development environment for application

Phase I: Initial Development

1th: Module 1 development start (4.1)

March 4st: Module 1 complete

5nd: Test/debug Module 1; Module 2 start (3.3, 4.1)

6rd: Module 2.5 start

8th: Module 2.5 complete

9th: Test/debug Module 2.5

11th: Module 2 complete

12th: Test/debug Module 2; Module 3 start (3.3, 4.2)

14th: Module 3 complete

15th: Test/debug Module 3; 1st integration start

4th: Module 5 start (3.1, 3.2, 3.4, 3.5, 4.3)

13th: Module 5 complete

14th: Test/debug Module 5, 2nd integration start

Phase II: QA Testing

19th → 29th: Test/debug Build 1.0

5. Project Monitoring

. Overall, progress in development of the application will be monitored via Trello and GitHub. Trello allows the development team to assign work, and mark completed tasks, while GitHub commits are expected to have version information and a changelog describing changes to functionality, as well as any fixes applied. Additionally, regular meetings will be held for team members to provide progress updates and propose new changes or discuss any currently under consideration.

5a. Traceability

Each module corresponds to sections of the Requirements Document as follows:

Module 1: Section 4.1

Module 2: Sections 3.3, 4.1

Module 3: Sections 3.3, 4.2

Module 5: Sections 3.1, 3.2, 3.4, 3.5, 4.3

Additionally, the overall design of the application is related to Section 3.8, while testing follows section 3.7.

All requirements change, and change control follows sections 6.1 and 6.2.

5b. Testing / Quality Assurance

Testing will be divided into three distinct categories:

1. Unit testing will be conducted during development as the modules are being built, to insure the individual components of the application work as intended. One to two people will be designated as primary testers during this time in order to conduct white-box tests. Typically, this will be the developer of the module, as well as a helper if deemed necessary. Where possible, automated tests will be constructed to assist with regression testing across distinct versions of each module.
2. Migration testing will be conducted whenever merges occur, in order to ensure each component works as intended when integrated with the rest of the system. Automated tests constructed prior will be used to ensure each module's individual functionality, while a mix of automated and manual tests will be conducted to ensure cross-module

functionality. Testing will largely consist of white box testing of cross-module functions and dependencies.

3. Phase testing will occur at the end of each major phase of development. This will largely consist of black box tests to ensure that the application functions as expected by the end user. At this point, most of the development team will exhaustively test from an end-user point-of-view, and the team may request the aid of third-party individuals to provide feedback as well.