North Star Senior Project Plan

Paul Mittan, Lockheed Martin

Benjin Dubishar

Chris Carey

Josh Osgood

JP Morgan

*Professor John Loser*

**Table of Contents**

[Overview](#h.t08il7kvk58l)

[Goals and Scope](#h.xynzcww3s726)

[Client](#h.5et1cbm56fxp)

[Server](#h.qz664iuyggdr)

[Services](#h.1f3ui8e8p1d8)

[System](#h.ujweqlrxg1l)

[Deliverables](#h.wjwfjh1xwx6)

[Exploration and Design](#h.t4fe2gtv0dx8)

[Implementation](#h.o8fvhlsr8jl6)

[Client](#h.8t9nrsm1s9xk)

[Control](#h.ef1r1m1m8j3a)

[Complete System Integration](#h.aw2j6bwcmdqv)

[Risk Management](#h.fk8q6j3ves6)

[Schedule Risks](#h.xse3czinork7)

[Time Estimation Inaccuracy](#h.vslrhq1n1beq)

[Scope Creep](#h.kes51h1o1jo5)

[Budget Risks](#h.j51esb8nu3kh)

[Limited Budget](#h.5l9l4knts018)

Operational [Risks](#h.ntfitpwarp2r)

Cannot Hardware Upgrade to Latest OS

[Technical Risks](#h.ntfitpwarp2r)

[Limited Android Experience](#h.o678r7tu0394)

[No Platform-Independent Solution Exists](#h.uh8o3asnell)

[Incapable Hardware](#h.s33btppsipyk)

[Scheduling and Estimates](#h.t6i86ycmkvbv)

[Measurements and Metrics](#h.xvbqfnhxrfvy)

Progress [and Effort Metrics](#h.kmtrxtxef7de)

Defect [Metrics](#h.khrokw8in7go)

Product [Metrics](#h.khrokw8in7go)

[Technical Process](#h.1fdg4odpohil)

[Tools](#h.slhahosvsa51)

[Code Control](#h.sekpvwg4mpbi)

[Development Environments](#h.z7t0fwaexj51)

[Techniques](#h.luewhqjskpkl)

[Client](#h.x42vzu6vtdqt)

[Control](#h.l65wanw44my1)

[Artifacts](#h.hd2vs7blgsrm)

[Testing Strategy](#h.bmxewknkoelf)

[Unit Testing](#h.x2ua7lkb3bsi)

[Integration Testing](#h.cuqcgvtlm45)

[System Testing](#h.8clgwckgl7mi)

# Overview

Our project involves the creation of a system that will allow users to track their location throughout a large, open, indoors space. Since GPS isn't quite accurate enough for tracking on such a small scale, an alternative method is necessary to get accurate tracking information. A large part of this project will deal with researching different technologies for device tracking and what user devices have the best tracking capabilities in order to find the best platforms to work with. After that has been completed, our focus will shift to the client interface and server functionality.

Our product will operate on a couple of different platforms, since it is split up into different sections. The main app that the client will use to track their location will operate as either a web app or a native android app, with research being done into how much work it would be to also develop an ios app for apple devices. The server side of our product will operate on either a virtual or physical unix-based server. And our map and itinerary creation programs will either be web apps or programs.

Our sponsor for this project is Paul Mitton and the Lockheed-Martin corporation. Paul came to us with this project because he had a need for a small indoor map system for judges to navigate project fairs. The previous team that worked on this system was able to develop a system, however it gather location data too small and was not very precise. Lockheed-Martin wants this to be an open-source project, so that other developers can benefit from and improve upon this project.

Our schedule will be divided up, as there are a couple sections that can be done in parallel. The majority of our work starting out will be hardware research, both for what devices we want to support for the client and what system we will use for location tracking. Once we have that down, we can work on the client application in tandem with host side applications like the server, map, and itinerary applications.

We will be responsible for delivering the hardware specifications for the system we want to implement. All the code for the client, server, map, and itinerary code needs to be finished and uploaded on an open repository. In addition, all project documentation, project deliverables, and sponsor requested action items need to be completed.

# Goals and Scope

## Client

The team will develop a client application on the Android platform targeting a specific device (TBD). The ultimate goal is complete platform neutrality, though this is initially beyond the scope of the project. The client application will be able to communicate with the server application for the following purposes: retrieving a configured floorplan, notifying the server of its location, and retrieving a set of waypoints (if applicable). It would be desirable for the client application to interface with an external judging system, but this is beyond the initial scope of the project.

The main function of the client application is to determine its position relative to each positional anchor, which would then be translated into an absolute position based on a floorplan. This process must be, at a minimum, reasonably accurate. If possible, this process must also be efficient enough to allow for live updating.

At the time of this project plan, all other client functionality is beyond the scope of this effort.

## Server

The team will develop a server application for deployment on a specific Windows laptop. Platform neutrality would be desirable, but is initially beyond the scope of this effort. The server application will be able to accept map configurations and event itineraries from either an internal or external service. The server will then communicate with one or more client devices for the following purposes: distributing map configurations, retrieving client locations, and distributing event itineraries (if applicable). The ability of the server to distribute client locations to each client might also be desirable, but is initially beyond the scope of this effort. The server must be capable of handling (TBD) concurrent users, with support for additional users desirable.

## Services

The system will provide administrators with the ability to configure floorplans, including areas of interest and the location of positioning anchors. These configurations can be saved in a format readable by the client, which would receive them after connecting to a server. It would be desirable for these floorplans to be configurable on-the-fly to allow for unplanned changes to the event layout. The ability to define complex features, such as wall thickness/materials or map items that fall beneath the minimum resolution of the floorplan, is beyond the scope of this effort.

The system will also provide administrative users with the ability to define a list of map waypoints, as well as a means of associating these itineraries with users or user groups. These will be saved in a format that is readable by the client, allowing users to make note of these locations on their own display. The ability to determine any path to these waypoints is beyond the scope of the application.

These services may exist internal or external to the server, but ought to be accessible to the event administrators and redistributable during an actual event. These applications need not be developed by the team if accessible solutions already exist.

## System

The complete system will be able to be deployed in no more than 30 minutes (excluding the time to create a detailed floorplan). The system will usually be deployed alongside a web-based judging system, but any integration is beyond the initial scope of the project.

# Deliverables

## Exploration and Design

The initial phase of the project focuses on three primary tasks: researching a variety of potential hardware and software solutions for wireless triangulation, evaluating hardware platforms upon which these solutions must be able to run, and determining an effective software architecture and language for the server and its accompanying control utilities.

## Implementation

The implementation phase of the project branches into two primary sections. The Client branch focuses on the tablet’s functionalities themselves, whereas the Control branch contains the server itself and the utilities for creating supporting constructs.

### Client

*Wireless Triangulation Research*

Research into how location tracking via wireless technologies such as WiFi, Bluetooth, and RFID can be used implemented, compatibility with current hardware, and its feasibility within reasonable budget constraints.

*Prototyping*

Implementation of a highly-specialized prototyped app for determining 2D position relative to a set of anchored “beacon” points.

*App Construction*

Elaboration of a fully-realized app around the completed prototype from the previous stage.

### Control

*Map Formatting*

Creation of a utility for creating a metadata-based map of the venue, including placement of tables/objects within. This map will be distributed to the client devices through the server runtime. This forms the basis of the visual location platform, to be used in the Itinerary Maker, Server, and Client-side app.

*Itinerary Integration*

A second utility for creating a number of separate itineraries for each judge or team of judges, also distributed to client devices. This utility operates on top of the output from the mapping utility, and therefore must be completed afterward.

*Information Distribution*

The final piece of the Control suite is the server, which handles information collection and distribution of maps, itineraries, and client locations. The basic construction of the server can happen in parallel to the utility pipeline and client app, but must be integrated with the rest of the Control suite before the client apps.

## Complete System Integration

Lastly, the control and client sides of the project need to be integrated together and have higher-level tests performed.

# Risk Management

## Schedule Risks

### Time Estimation Inaccuracy

Description:

The team may inaccurately estimate the time required to complete various phases of the project. This may cause the team to miss committed deadlines to the clients.

Mitigation Strategy:

The team will adjust time completion estimates for phases as the team begins them. Completion times for later phases will be overestimated, so that the team has the necessary room to adjust completion times so as to stay on schedule. In the event that the team is ahead of schedule, the iterative nature of the development phase will allow the team to add an extra schedule to fill the extra time.

Management Strategy:

Time measurements and metrics will be used to assess team velocity in order to make the most accurate completion time estimates.

### Scope Creep

Description:

The project may have additional requirements added throughout its duration. If the team commits to implementing all additional requirements, the team is at risk of not completing the project.

Mitigation Strategy:

After the team has specified the selected requirements for implementation, any additional requirements would be formally assessed, categorized, and prioritized. Even though additional requirements will be assessed on their level of importance and ease of implementation, in general most additional requirements will not be committed to by the team.

Management Strategy:

The team will maintain a document of additional requirements which contain various attributes such as the level of importance and ease of implementation, from which the team can make a decision on which requirements to commit or not commit to.

## Budget Risks

### Limited Budget

Description:

The team has a limited or nonexistent budget allocated for the purchase of required hardware components. With a limited budget, the team may be forced to purchase low-quality hardware that will fail to meet the project requirements. The team may also be unable to purchase new hardware after an iteration shows that previously purchased hardware is ineffective; the hardware is not upgradable if necessary.

With no budget, the team will not be able to meet the project requirements to specifications.

Mitigation Strategy:

The team will compare potential hardware options against a set of criteria to justify the need for certain hardware purchases. With no budget, the team will fallback on already owned devices which may or may not meet project requirements.

Management Strategy:

The team will rely on various measurements and metrics during acceptance testing to quantify the quality of allocated hardware in the project.

## Operational Risks

**Cannot Hardware Upgrade to Required OS**

Description:

The team has implemented a solution that works on later OS versions that the hardware devices cannot be upgraded to.

Mitigation Strategy:

The team will explore the most likely solutions and what OS versions they require. The team will also select and target hardware that can be upgraded to the latest OS version if possible.

Management Strategy:

The team will note the required OS version for all implemented software. The team will also note that latest OS version that hardware can be updated to. This will also be used as part of the hardware selection criteria.

## Technical Risks

### Limited Android Experience

Description:

The team has limited experience with Android development. None of the team members have taken a formal Android development course (none are offered at RIT).

Mitigation Strategy:

The team will spend one week self-learning Android through online tutorials and projects before attempting to implement an Android solution, should an Android solution be chosen.

Management Strategy:

If a native Android solution is selected, the team will allot one week directly prior to the first iteration of implementation in the schedule.

### No Platform-Independent Solution Exists

Description:

Given that Apple and Google have historically diverged from using standards that support both platforms (e.g. NFC vs. Airplay, Micro USB vs. Lightning, etc.), it is likely that an effective solution that works for one platform will not be compatible with the other.

Mitigation Strategy:

The team will spend the first few weeks of the project investigating various solutions and identifying which platforms it can be implemented on. The team will assign priority to cross-platform solutions when selecting an implementation strategy.

Management Strategy:

If forced to make a decision between Android and iOS, the team will most likely implement the solution for Android first due to its low-cost entry barrier, use of open standards, access to hardware for development, and higher overall expertise

### Incapable Hardware

Description:
There may not exist hardware that is powerful enough to meet the project requirements (or it may exist but is out of budget).

Mitigation Strategy:

The team will research similar projects to determine what hardware was used and what accuracy was obtained to first determine if this risk is legitimate. The team will exhaust their exploration time investigating software solutions to improve the accuracy of the implementation.

Management Strategy:

The team will use measurements and metrics to determine whether the failure to meet requirements is a result of the hardware or the software.

# Scheduling and Estimates

This project consists of two separate efforts: the development of the client application, and the development of the server/utilities. Each of these solutions has unique characteristics that must be considered in our schedule. The client application, for instance, has fairly rigid requirements: show the user their current position on the map. With that said, a disproportionate amount of risk comes from this simple requirement. Conversely, there isn’t much risk involved with the server application, but the requirements are still open to interpretation.

To address these concerns, we’ve decided to develop both solutions concurrently while utilizing different schedules and methodologies. The schedule for the client application will be based on the spiral model - an iterative risk-oriented approach that involves a cyclical process of determining the goals, investigating the solutions, executing the solutions, and verifying the results. By following this basic schedule, we can continue to explore, evolve and refine our solution to achieving location tracking by unconventional means.

The remainder of the project is risky in the sense that the requirements are not completely known/defined, so we decided that a requirements-oriented agile process would suit it well. We have yet to decide on a particular process, but the general idea is that we will prioritize features and pursue them incrementally. (I’ll expand more on this later).

The simplest way of pursuing both these efforts concurrently is to divide our total available time between them. Because multitasking has been shown to cause some inefficiency due to the overhead of switching tasks, it would be ideal to split the team for these efforts. With that said, we would all benefit from having a part in each project, so it might be preferable to either rotate team members or simply divide our team along a certain percentage.

Milestones

End of Week 8 - Technologies Identified

End of Week 14 - Working Prototype

# Measurements and Metrics

## Progress and Effort Metrics

**Hours Worked:**

The number of hours worked by each team member will be recorded. Additionally, the number of hours worked will be organized into categories such as documentation, development, testing, etc. This will aid the team in determining where team resources are most needed.

**Team Velocity:**

During the development phase, tasks will be estimated into units of work (hours of work required). This will allow the team to calculate the individual and team velocities over time intervals. From this, the team can determine whether the team needs to speed up work to make deadlines, or can afford to slow down velocity to prevent bugs.

**Defect Metrics**

**Bugs Reported/Fixed:**

During the development phase, the team will report bugs along with their severity for tasks that have been implemented and tested. This will assist the team in determining if the team needs slow down development velocity in order to decrease the number of bugs created. The team will keep track of how many bugs are fixed.

**Average Bug Fix Time:**

The team will track the average time needed to fix a bug. This will assist the team in adjusting velocity as a slower velocity might not be necessary to minimize bugs if the team is quick to fix bugs.

## Product Metrics

**Range of Error:**

Before the testing phase of project, the team will determine a specific set of measurements that can be used to measure and evaluate the accuracy of the project and determine the range of error. This will most likely be a distance-based accuracy measurement, but a final decision depends on the design of the product.

**Test Coverage:**

The team will also record an estimate of the percentage of code that is unit tested and additionally a percentage of tasks and features that are integration and acceptance tested. The specifics of these metrics depend on the testing strategy that the team adopts during the product planning phase.

**Device Coverage:**

If appropriate, the team may determine a rough percentage of additional tablet devices that the product is compatible for. This will depend on the minimum required OS for the product and current tablet OS usage statistics. This measurement may not be needed if the team specifically targets only a single type of device.

# Technical Process

## Tools

### Code Control

To maintain our project, we will use Git to control the code, testing suite, and any necessary public-facing documentation (architecture and class diagrams, setup instructions, etc.). Git provides more finely-tuned control over alternatives like Mercurial, SVN, and CVS. To host our project, we will use a student GitHub account, fulfilling the “open-source” requirement from the sponsor, and providing easy access to anyone wishing to fork the project.

### Development Environments

*Statement of Tentativeness*

This section is not resolute, due to the fact that we have not yet investigated the full abilities of each language. The sponsor has expressed a wish for tools to be as cross-platform as possible, but we are keen to avoid a project based fully upon Java (save us from Swing!). Although we will investigate the possibilities of a web-based system “head”, we are tentatively selecting C#.

Additionally, we are still working with the sponsor to determine the ideal client/mobile platform. Again, he prefers a platform-agnostic solution, but we believe neither a mobile-site (Javascript) nor a cross-platform API would provide the sensor access necessary to complete the project. The group is performing investigations to solidify our intuitions, and we’ve already found that iOS doesn’t provide access to the depth of data.

*Control*

The computer-side suite of apps comprises three parts, the map builder, the itinerary creator, and - most importantly - server. Because these three programs will need to work with each other on a Windows machine, we’ve chosen C# as our language-of-choice due to its clean libraries and native runtime, and Visual Studio as the accompanying IDE, over Java (notoriously obnoxious package management) and Eclipse.

*Client*

The client-side app will be a native Android app written for 4.X and above, designed to run on 7+ inch tablets. To create this, we will be using the customized version of Eclipse packaged with the standard Android Development Kit (ADK), which necessitates the app to be written in Java.

If time, priorities, and APIs permit, we will also create an accompanying iOS app for iOS 7. This is, however, lower priority than the Android app and server-side program and companion tools.

## Techniques

### Client

This project has several parts which must be developed both serially and in parallel. Because the exploratory/research-oriented aspect of the hardware side of the project, we looked for an iterative approach with a greater degree of planning and preparation at the beginning, due to the additional financial expenses if our initial path ended up incorrect.

### Control

The sponsor has stated that he’d like us to use Agile development processes if possible, so that he can learn about it via proxy. We believe that the set of programs which make up the Control aspect of the project are an excellent opportunity to employ a variety of Agile development techniques, though we haven’t nailed out specifically which at this time.

Development sprints will be hugely beneficial in maintaining a dynamic nature of priorities and integration as the different components of the Control suite evolve. Although the map maker music be reasonably complete before itinerary maker can begin, the two can be developed in parallel with the server itself, before direct integration into the complete workflow. For example, code sharing via libraries between each individual program is a necessity for map interaction.

### Artifacts

At the beginning of each significant development cycle, all diagrams pertaining to the architectural organization, use of the system, and interaction between components will be updated to reflect the latest understandings of requirements and feasibility. Additionally, a testing document will be maintained, which will explain the techniques we plan to use to ensure that the code behind the system is working correctly; this is elaborated upon more in the Testing Strategy section of this document.

# Testing Strategy

Our testing strategy will differ based on parts of the project. We do not have any defined testing methodology or technologies at this stage; it will be decided on after we have completed more of the research and decided what platforms, technologies, and languages we will be developing with. We do have high-level goals of what testing we want to see done in regards to testing on our project.

## Unit Testing

The client, server, map, and itinerary code will all be put under unit testing. The assertion and test driving frameworks we will use for unit testing may differ between sections, depending on factors like what language and environment those code sections use. We will track metrics relating to how much of the code base is under test, and will strive to getting as much of our testable code as possible under test.

## Integration Testing

We will have a heavy focus on integration testing, as all of these subsystems will be interacting with each other and will rely heavily on information provided by each other. Our focus will primarily be on the client-server relation, as the other two sections - being primarily graphical - won't require as much work. While it would help identify integration problems a lot faster, we will probably not set up automated testing for this project, as we don't want to spend time dealing with setting up, configuring, and maintaining a server for that.

## System Testing

Once the system is set up, we will perform black box testing on the whole system to see how well it meets the requirements we've established. Primarily, this will be done to test how accurately we can track locations of devices and how our system handles multiple devices relying on it at once. The connection between our client based solution and the server based solution will also be tested to make sure it is up to our standards.

**\* Need to determine where to put “Open Source Licensing” and “Landing Page” - perhaps deliverables?**