Example Requirements Document

for

Converged Altitude Calculation Program (CACP)

Prepared by: Your Name Here

Updated: July 12, 2025

Overview: This document provides an example requirements document for Kansas 4-Hers in the Computer Science division. It is based on plans for a converged altitude calculation program (CACP) that allows users to calculate the height of their rockets when using multiple tracking stations. This provides users with a more accurate altitude for their rocket than using a single tracking station.

Note: Material in brackets [] and italicized is to provide information about the section and is not typically included in a requirements document.

Note: This is one example of a requirements document, other requirements document formats are fine and can be used.

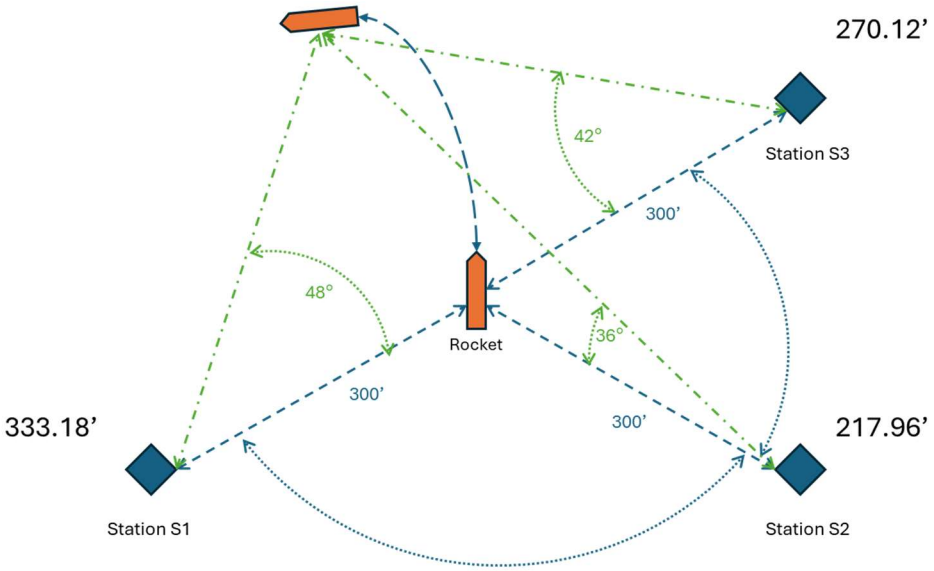
Note: This is a moderately complex requirements document. No matter how simple or complex, requirements documents should have a requirements section. They can be simpler or much more complex than this one. The goal is to share with anyone reading it, why your project matters, why you want to create it, and what needs to be done to create it.

A good requirements document will allow a developer (you) to look at it and check off things as you go. It also makes sure you don't forget something important, whether it's a reason for getting a cell phone or to make sure a button turns green when data is entered.

Overview

When someone launches a rocket, they want to know how high it went. This can be determined by using an altimeter placed in the rocket or with tracking stations placed at various distances from the rocket. The tracking stations are what interests me.

A single tracking station is okay, but if the wind blows the rocket so it doesn't fly straight up, or the tracker is on a hill the altitude may not be correct. To fix this, I want to use multiple tracking stations which will provide a more accurate altitude when I launch my rocket or anyone else who uses it.



As you can see in the diagram above each of the tracking stations have different results because the rocket wasn't straight up and down at apogee. Which of the readings is correct 333 feet, 218 feet, or 270 feet? There is more than 100 feet of difference between the tracking stations.

With multiple tracking stations figuring this out can be done manually with pencil and paper and some really hard math, but I want to make it easier and faster by turning it into an application I can use on my dad's cell phone and maybe later share it with my friends.

My objective is to build an Android app that will tell me how high my rocket went accurately using measurements from multiple tracking stations.

CACP Application Overview

[This section provides information about who the document is for, why, and how it will be used. The 'big picture.']

CACP will be an application where you can enter tracking data from multiple tracking stations to determine how high a rocket went. The app will be able to run on my dad's Android cell phone. The app will consist of a few screens that allow the user to input calculations, see the results, and view the previous history of calculations. *[← This is your objective statement in a single paragraph.]*

My app provides the following benefits:

- More accurate altitudes at rocket launches
- Faster results than doing it on paper
- Better results than just using a single tracking station
- Helps younger kids figure out how high their rockets went
- Easy to use, because it's on a cell phone

Why program CACP?

- I've never built an Android app before
- I need a project for the fair
- I don't like having to do math to find out how high my rocket went
- No one likes doing math
- Be a reason for me to get my own cell phone

If I don't build CACP, what will happen?

- I'll have to do hard math over and over again to figure out how high my rocket went
- I might not have accurate altitudes of my rocket for the fair
- My friends and younger 4-Hers won't know how high their rockets went

Who will benefit from this App?

- ME!!!
- My friends
- Other 4-Hers
- My parents

Is there something more important I should be doing?

There are other programs I'd like to create but with the fair coming up and all my other projects underway, I think this is the best thing to be working on right now. They won't help me with my other projects for the fair.

Previous Process

Before CACP, it was necessary to write down the measurements from each of the tracking stations on a sticky note, wait till after the launch, and then put them in a formula to calculate how high my rocket went.

I would use the following formula that I found on the Model Rocket Dual-Axis Altitude Tracking Calculator (https://www.translatorscafe.com/unit-converter/en-US/calculator/rocket-dual-axis-altitude/).

$$f = \\ = \sin\varepsilon_1 \sin\varepsilon_2 - \cos\varepsilon_1 \cos\varepsilon_2 \times \\ \times \left(\cos\alpha_1 \cos\alpha_2 - \sin\alpha_1 \sin\alpha_2\right)$$

This would take me about 5 minutes for each launch to figure out how high my rocket went. And sometimes we would mix up tracking station measurements and not get good results.

Process with CACP

With the CACP app, we will open the app on the cell phone, each station would call out their measurements, we would enter them in the app, and then press the calculate button. The app would then tell us how high the rocket went in a few moments. If there were an error with the measurements we could correct them prior to the next launch.

Functional Project Requirements

["This part of the requirements document states in a detailed and precise manner what the application will do." (Berezin, 12)]

[Break down the features and capabilities of your software into a list of things that need to be done to create your program. Each of the top-level items are taken from the overview section.]

*[As you build your project you can check off the items from the list below. Refer to KSF Computer Science rule 4.b. **This is the key part of a requirements document**]*

Run on my dad's Android cell phone

- ☐ Dad's phone is a Galaxy S20+
- ☐ Needs to be written in C++ language using the Android Software Development Kit (SDK)

Be something my dad can operate

- ☐ Once built, I will need dad to test it
- ☐ Make up some sample data to test with. Data needs to be both:
    - Accurate (I'll use launch data from previous launches) so that I can make sure it shows the correct numbers
    - Inaccurate (I'll make up some numbers) so that I can make sure it will show errors correctly
- ☐ Validate results with other calculators like: https://www.translatorscafe.com/unit-converter/en-US/calculator/rocket-dual-axis-altitude/

Learn to build a cell phone app

- ☐ Needs to be written in C++ language using the Android Native Development Kit (NDK)

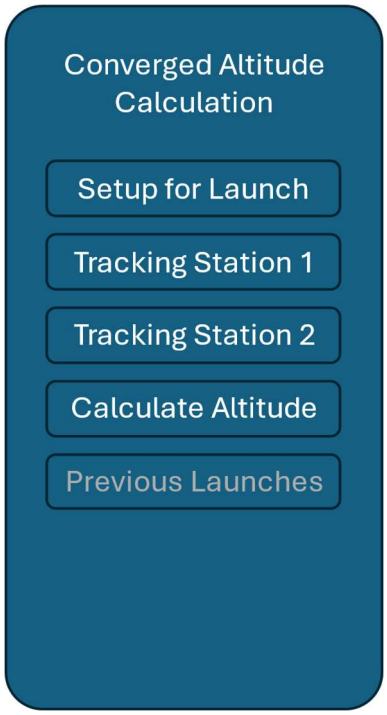
When the application loads:

- ☐ Load the main screen
- ☐ Zero out tracking station 1 and 2 values

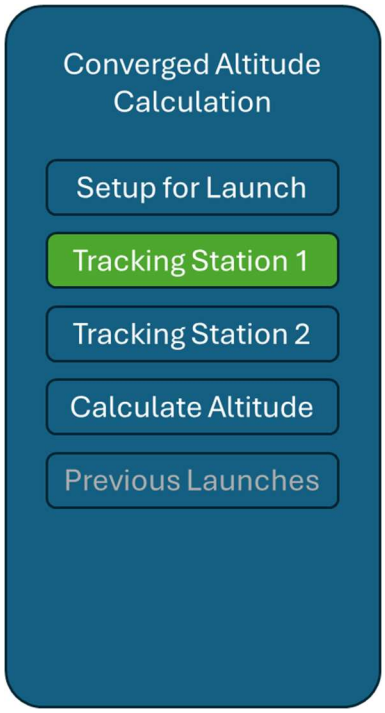
Create a main screen

- ☐ The main screen will have the following buttons:
    - Setup for Launch
    - Tracking Station 1

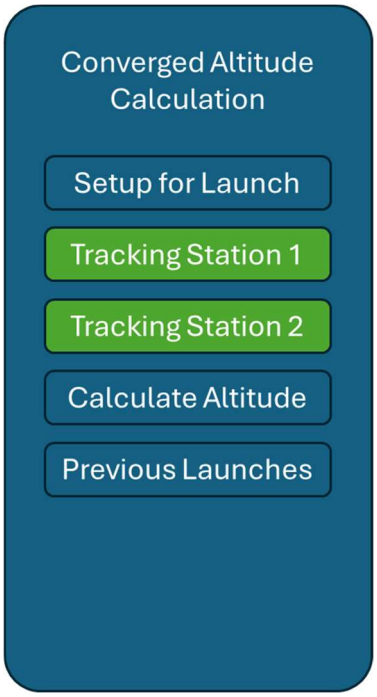
- Tracking Station 2
- Calculate Altitude
- Previous Launches



- When data is saved for tracking station 1 or 2 the buttons will change color indicating they have saved data.



- If both tracking stations have valid data (are green) activate the Calculate Altitude button.



- If either or both tracking stations do not have valid data deactivate the Calculate Altitude button.

The “setup for launch” button makes sure you have each tracking station setup correctly.

- This screen is displayed when the setup for launch button is tapped on the main screen.
- This screen will show how to place and zero out each of the tracking stations prior to launch.
- It will have a back button at the bottom of the screen.

Be able to support 2 or more separate rocket tracking stations

- Will need a screen for each tracking station to enter their values.
- The tracking screens are identical to each other.
    - Tapping on tracking station 1 takes you to the first tracking station
    - Tapping on tracking station 2 takes you to the second tracking station
- Each station will need the following inputs:
    - Distance from rocket (in feet)
        - Valid numbers: Positive numbers greater than 0 and no more than 5280
    - Distance from each other (in feet)

        - Valid numbers: Positive numbers greater than 0 and no more than 5280
    - Azimuth (in degrees)
        - Valid numbers 0 to 360
    - Elevation (in degrees)
        - Valid numbers 0 to 360
- ☐ Each input will need to be checked for correctness. For example, an 'e' is not valid for any of the inputs.
- ☐ Save this data temporarily so that I can add the information for the other tracking station
- ☐ Return to the main screen once information is entered by tapping a save button
- ☐ A clear button will clear any data set for the station.
- ☐ A back button will take the user back to the main screen without saving any changes

To tell me how high my rocket went when I enter data from my tracking stations

- ☐ Create a calculation screen that shows how high my rocket went
    - The calculate altitude button takes you to the altitude screen
    - If data isn't saved for tracking station 1 or 2 it displays an error telling you which station doesn't have data.
    - If both stations have data, the variables from each are put into the formula found at: https://www.translatorscafe.com/unit-converter/en-US/calculator/rocket-dual-axis-altitude/ this will do the math for me.
    - From the formula the following is displayed:
        - Altitude in feet
        - Closure error in %
        - If the closure error is >=10% it displays a message that the "track is not closed" in red
        - If the closure error is <10% it displays a message that the "track is closed" in green
    - At the bottom of the window is a button to save the launch
        - When pressed the values from tracking station 1 and 2, the altitude, the closure percentage, and the date and time are appended to a CSV file.
    - At the bottom of the window is a back button that returns you to the home screen

Keep old launch records for previous launches:

- ☐ On the home screen there should be a button for previous launches

- When that button is tapped it should display the previous launches (each listed by date and time) and I should be able to tap on the launch I want to view.
- This should be read from a CSV file containing the saved launch results.
    - If there are no items in the file, it should display a "no launches" message
    - Each launch will be a line in the CSV
- When I tap on a launch, it should provide the altitude and measurements from each of the stations.
- At the bottom of the launch history screen there should be a done button to exit the screen and return to the previous launches screen.
- At the bottom of the previous launches screen there should be a done button that returns you to the home screen.

Exit the program:

- ☐ When the user presses the phone's back button it will exit the application

Resources

[List any resources that you used or might use in your work. This can include places you got code from, sites that helped you troubleshoot an issue, or places where you learned how to do things or provided details you needed for your project. I used a modified APA style to create citations, use whatever style works best for you.]

Translatorcafe.com. (2025, July 10). "Model Rocket Dual-Axis Altitude Tracking Calculator" https://www.translatorscafe.com/unit-converter/en-US/calculator/rocket-dual-axis-altitude/

Developer.android.com. (2025, July 10). "Add C and C++ code to your project" https://developer.android.com/studio/projects/add-native-code

Developer.android.com. (2025, July 10). "Get started with the NDK" https://developer.android.com/ndk/guides

Droidcon.com (2025, July 10). "Using C/C++ in Android: A Comprehensive Guide For Beginners" https://www.droidcon.com/2024/01/16/using-c-c-in-android-a-comprehensive-guide-for-beginners/

Developer.android.com. (2025, July 10). "Create your first Android app" https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app

YouTube.com. Rahul Pandey. (2025, July 10). “CS194A – Android Programming Workshop” https://www.youtube.com/watch?v=xZEaFcn5WrE&list=PL7NYbSE8uaBDcLkbXsQADdvBnVbavonGn

Here are some other resources for requirements documents.

Berezin, Tanya (1999, June 14). "Writing a Software Requirements Document" https://home.adelphi.edu/~siegfried/cs480/ReqsDoc.pdf

Asana.com. (2025, July 12) "How to write a software requirement document (with template)" https://asana.com/resources/software-requirement-document-template

KrazyTech.com. Bandakkanavar, Ravi (2025, May 28) "Software Requirements Specification document with example" https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

GeeksForGeeks.org. (2023, September 20) "Software Requirement Specification (SRS) Format" https://www.geeksforgeeks.org/software-engineering/software-requirement-specification-srs-format/