
**University of Arkansas – CSCE Department**
**Capstone I Preliminary Proposal – Fall 2019**

# Weather Ways: Tracking Weather Along a Route

**William Henness, Zachary Cantrell, Audrey Timmerman,**
**Nicholas Brinkley, Madison Galloway, David Turnbough**

## Abstract

It is often quite difficult to consistently find updated information regarding weather conditions while traveling for business or leisure. One can easily find this information for specific locations, however planning routes for travel and updating the routes' information is a task which many find difficult to accomplish. This application will provide a simple user interface for the users to plan a trip to and from any destination for which weather information is available. Currently a widely accessible application of this nature which works effectively and provides a high ease of use for the user does not seem to exist. Thus, this service could be provided to potential users with the creation of this application. The user base itself is potentially quite massive due to trips and vacations being quite common, and with almost every individual owning a phone.

The main objective of the application is to provide constant weather information for the user's current location, along the user's route, and at the user's final destination. Following this general idea, the application created and may utilize information provided by available software. The purpose of the app will be providing the user with a route planning tool using Google Maps software and its information. An in-app map will aid the user in viewing the layout of the potential route which is being created and guiding them along it. Following a chosen route, information will be provided to the user regarding the conditions of the areas which will be traveled to by the user.

## 1.0 Problem

According to a AAA travel survey in 2019, approximately 100 million U.S. adults are planning on taking a family vacation. Of these 100 million (around 4 out of every 10 U.S. adults), around 53% plan on taking a road trip (Hall). This means around 50 million U.S. adults in 2019 plan on taking a road trip within the year. Business trips also provide a large portion of yearly travelers which number far higher. The potential users, then, would consist of millions of U.S. adults which will be traveling to any destination.

Organizing a road trip can be quite difficult because vast areas will be covered with many differing environmental conditions, and coordinating events to take place along these routes

requires specific timing. Once the road trip is in motion, unexpected weather changes may occur which then dampen the enjoyment of the vacation, as well as potentially ruining the plans of the trip which could harm the goers' finances and further delay the remainder of the trip. The desire is to alleviate the surprise of weather changes for the user to enhance their experience.

Weather conditions can change and turn hazardous very quickly, and these updates can be difficult for people to be aware of while driving. Certain conditions, such as a tornado, pose a very high risk of danger throughout a large area, and without consistent updates drivers may be unaware. Lighter weather conditions may also prove to be dangerous for travelers. Unexpected rain or snow can be stressful and downright dangerous. Using this application, users would be able to mitigate the danger and make informed decisions based on the information provided. Safety is a major concern when travelling. Therefore, anything which can improve it should be attempted.

## 2.0 Objective

The objective of this project is to create an application which will provide information regarding weather and route tracking for the user while traveling. This application aims to provide relevant weather information in a concise and helpful way. The goal is to allow users to make informed decisions about weather conditions so that long road trips are safer and less hectic.

## 3.0 Background

### 3.1 Key Concepts

The technologies necessary for completing this problem are the Google Maps API and the Open Weather API.

Google Map API:

Google Maps API allows you to "Build customized, agile experiences that bring the real world to your users with static and dynamic maps, Street View imagery, and 360° views" ("Google Maps Platform Documentation"). The features we care most about are the routes and places which allow the users to find the best way to get from any start location to any final destination while also being provided with high-quality directions and real-time traffic updates. The places features help users discover new locations and includes built in features such as current locations, found place location, autocomplete geocoding, geolocation, and time zone. Because we are going to be creating an android based application, we will be taking advantage of Google's SDK for android.

Open Weather API:

Open weather's API allows access to current weather data for any location and the current weather is frequently updated based on global models and data from over 40,000 weather

stations. This API also provides hourly forecast and is geographically accurate. For our project we would care about accessing current weather data and and applying over time forecast data along a route.

### 3.2 Related Work

Some developers who have done work in this area are Morecast, who have created a browser and phone application that allows the user to see the weather on their designated route. However, their mobile application is not optimized and does not allow you to zoom out as much as on the web application. The application does not offer any other alternative routes; it only offers one route from each location. Additionally, it provides no routing assistance, so it cannot be used while getting directions to the destination. This makes it either useless or very dangerous for people traveling alone. Our implementation would fix this problem and allow users to type in their specific home addresses and get unique directions and weather advisories for their specific route. These advisories would layer on top of the directions with no input needed from the user ("Route Weather").

## 4.0 Design

### 4.1 Requirements and Design Goals

For this project there is one central use case. Therefore, the app will be streamlined to the application, making it easy to find relevant information quickly. Using the app, users should be able to make a route from their location to their destination and the app will track them along this route. This route information will be used to define points along the route. These points will be matched to locations where information from the national weather service is being collected. This information will then be used to predict weather along the user's route. The app should be streamlined to make it easy to start a new route, delete an old route and have continuously updating weather information as predictions change.

There are five requirements for this app to be considered functioning. The first is that it must be able to design a route from the user to their destination. If the user gets off route, the route should be recalculated, and the weather updated. Second, it must be able to show an hourly forecast based on the route's timing and location predictions. Third, the information in the forecast must be updated regularly along the route to keep weather up to date. Fourth, this app should use the Google Maps API to give users directions while a weather icon and brief weather information is displayed in a header. This header will usually show the next forecasted weather location and the time when the change will occur. However, if there is a weather emergency the emergency weather alert will replace the normal header. Finally, this app should be functional on common Android devices.

If the previous design goals are readily met, there are further developments that could improve performance. For example, other features could be importing routes from other Google maps applications, showing live radar of the area with the route predictions, or adding a text to speech function to read weather warnings to the driver.

### 4.2 High Level Architecture

This application will be designed in Android Studio, and thus its core frontend language will be Java. Our initial application, when opened, will look very similar to how Google maps, or any other map application looks. Where our application will be different, however, is that it will take the information on the route and send it to a separate fragment that holds weather information. This separate part of our application will tell the user the weather conditions at their current location, as well as the weather conditions at different locations along their route. By including these features, we are allowing the user to manage their trip and avoid getting into potentially dangerous weather conditions.

Google Maps API is the premiere maps API and the obvious choice for our project. It offers several key pieces of functionality required by our project. These include a routing feature, the ability to set waypoints or markers along the route, keeping track of user's current location, an overlay of current directions, and the ability to grab geographical data from the markers. Google maps is incredibly intuitive and even includes an introductory base project in Android Studio. The user will first need to enter an initial and final location, though the initial location will likely be the user's current location. Using the Maps API, we can easily create a route with these destinations. This route will feature some of Google's different warnings along the route, indicating to users that their current route may be slow. An overlay will be required so the user can pay as much attention to the road as possible. This overlay should look similar to the one that is in the Google Maps application, indicating the next place the user needs to turn. Along with the route overlay will be a weather overlay that displays information from our weather API. This weather overlay will have to be created and is further discussed in the next section. Its overall purpose is the same as the route overlay. Because we require the user's current location, we will need to activate location services on our application. This is done in the Android Manifest. For the purpose of this project, we should only need to access the users Fine Location. However, we must ask permission before using their location to avoid any privacy issues. We will find the weather along the route by adding several different markers along our route. These markers will contain geographic coordinates. These coordinates will be used to grab weather conditions along that route. The user's current location will be used to query current weather conditions. Google Maps will work hand in hand with the Open Weather API, as they are both required in each design phase of our application. How this will look, and exactly what is required to be shared between the two will be discussed later.

There are several free weather API's that we could have used for this project. The one we chose, which is also one of the more well known, is Open Weather. Most of the weather API's provided different tiers of functionality at different price points. We felt that Open Weather provided the best functionality for us of any of the free plans at our disposal. The free version of Open Weather gives us access to several tools that are relevant to the functionality of our application. These include a current weather API, a 5-day outlook with 3-hour intervals, weather alerts, and several maps relating to the current weather. Our application is designed to have both upcoming and current weather conditions. Because of this, access to these different functionalities is vital to completing some of our more important design goals. Additionally,

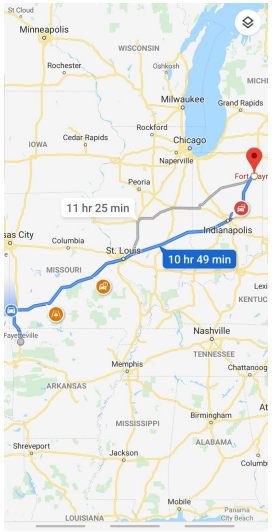
weather data can be accessed using longitude and latitude values, which works well with the Google Maps API. Because we will be saving locations on the route using geographical data, this can easily be passed to our weather toolsets. The current weather API that Open Weather offers will be used both in the route view of the weather and the location specific view of the weather. This will regularly be updated to match with the user's current location, which is simple to keep track of thanks to the Maps API. Some of the information we get from the Open Weather API includes the current temperature, humidity, wind speeds, cloud cover, sunrise and sunset, and overall weather conditions. The 3-hour interval view gives us access to less information but over a wider range of time. It will also be used in both the route view and the location specific view. Though it lacks as much information, it contains enough for an overtime look at the weather. The data we are given access to includes temperature, humidity, and overall weather conditions. These are updated every few hours, which will allow us to also update our application every few hours. The weather alert functionality is also an important aspect of our project. Since the user will be driving while using this app, even if they have a passenger, it is a necessity that overall interaction with our app is minimized. Alerts allow the user to safely travel with the knowledge that any potentially dangerous situations can be avoided with a simple message. This will be used both for locations on the route and the user's current location, so that they can make the best decisions possible. This API works with triggers to set the alerts and display them when necessary. As mentioned in the Google Maps portion, Maps API will also be used to create an overlay on the Maps view, so the user can pay more attention to the road, without having to change the screen to look at weather conditions. This will likely be current weather conditions, though alerts will also be displayed in this overlay. The overall layout of this is shown below, we are attempting to integrate these two API's as seamlessly as possible.

For storing our users' data and any other necessary data, we have decided to go with a client server system using Heroku. This is a free software that we all have experience with. The Heroku server will be able to hold all of the data relating to user locations, route locations, weather conditions throughout the route, and any other potential data that the user will need saved. While this will be more difficult than implementing something like a content provider, it will offer much more specialized uses as our application begins to take shape. If any part of our program needs something saved inside of our database, say the weather conditions at a certain point on your route, our client will contact the server and get that necessary information. The client can then relay that information to the proper activity on the app. The map and weather APIs will need to be constantly checked for updates to the weather. Because of this, we need to ensure that the different aspects of our project are actively checking if new or altered information is present in the corresponding database. Luckily, Google Maps API naturally does this, constantly maintaining an updated map. We intend that, for whenever Google updates info, it sends a check to the weather API for the new information on the server.

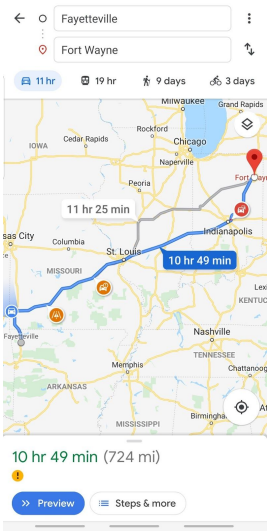
Heroku's free tier allows the storage of up to 10K rows of information. It also will allow up to 20 connections at a rate of 25 MB of RAM. Heroku allows a plethora of programming language for the server. However, we are going to be using Python for the server function calls. These function calls will allow us to access all information stored on the server and perform whatever calculations or interpretations necessary before we send the data to the client.

The user's interaction with the application's navigation as well as the application's ability to provide easy to understand information is a key goal of the application's front-end. Since the application deals with users who may be operating a vehicle, the ease of use is a high concern when implementing the user's interaction with the application. The ability to quickly move from screen to screen, as well as gathering information from the application at a glance is imperative. Since the application is built upon the Google Maps API, many users will be comfortable with the application from their initial use. The application will consist of three screens to provide information to the user about their route's forecasted weather conditions: the routing screen, a list view of forecasted weather conditions along their route, and an in-depth look at the weather conditions at a particular point along their route. The transition from screen to screen is meant to be quick and efficient, at most the user will need to tap the screen twice to move anywhere in the application. However, the user will for the most part remain on the routing screen for their trip once it is initially set.

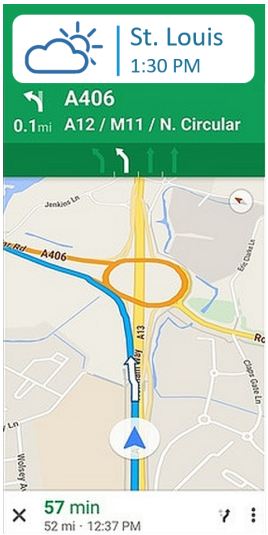
Example Map Image 1




Example Map Image 2




Example Map Image 3



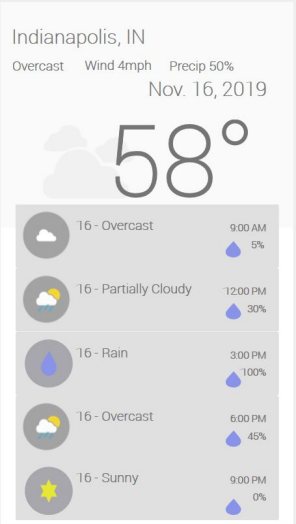

The first screen the user will be presented with when they open the application will look very similar to the Google Maps application. The screen will consist primarily of a map, which will utilize the Google Maps API. Across the top portion of the screen will be ribbon containing an area for the user to enter a desired destination, as well as their current location or starting position, as shown above in Example Map Image 1. Once the user enters the required information into the fields regarding their route, as shown above in Example Map Image 2, the weather conditions the user will encounter will be determined utilizing information retrieved from the Open Weather API. The forecasted weather information the user will experience will be presented to the user along the top of the application in a banner as shown in Example Map Image 3. The application will notify the user of the weather conditions they will face, as well as when they will arrive at the given weather conditions. The banner will update as the user reaches the current forecasted weather location, to the next forecasted weather location along their route.

The information in the banner will consist of a location, temperature, a time at which the user will reach the weather condition, and an icon displaying the weather conditions in an easily recognizable format to allow the user to focus on the road. In addition to the forecasted weather conditions, the application will provide the user with a secondary banner, beneath the weather banner, providing the user with turn by turn instructions for their route, similar to Google Maps. The turn-by-turn banner will also allow the user to change their route if needed. Clicking on the weather banner will take the user to the application's second information screen, consisting of the forecasted weather list view.

Hourly Weather Example


In addition to the user's routing screen, the user will be able to access a secondary screen by clicking on the forecasted weather banner. Upon clicking on the screen, the user will be brought to a list view containing weather conditions the user will encounter at hourly time intervals along their route, as shown in the Hourly Weather Example above. Each item in the list will consist of the time at which the user will arrive at the given location, the weather condition, the temperature, and an icon representing the conditions the user will encounter. The weather conditions the user will face will be displayed in easy-to-understand icons, so the user does not need to take their eyes off of the road for long periods of time. The application's easily readable information will allow the user to make informed decisions about their route based on the weather conditions they will meet. Additionally, if the user wants more information regarding a weather condition about one of the locations on the list view, they will be able to tap on the weather condition and the final information screen will display, containing detailed information about the weather conditions.

Single Location Over Time Example




The application's final information screen provides detailed information for the user regarding the forecasted weather conditions about a particular location along the user's route. When the user taps on one of the forecasted weather conditions from the list view items, they will be taken to a screen displaying detailed information regarding the weather conditions of that location, as shown in the image above. The information this screen will provide consists of the location, temperature, wind, precipitation, and weather conditions for the coming hours at the given location. This information will allow the user to make an informed decision regarding the time they stay in a given location. That is, if the user determines they would like to make a stop along their route, they will be able to use this information to determine when it is best to do so.

There are a number of features that our team has considered including into our app that have not been discussed. These different features will be discussed here. Because the different tiers of functionality on the weather API's we explored are so expensive, there are only so many pieces of functionality that we have access to. While it is enough to give a good look at our applications potential, there are areas we must consider leaving out. One that we are likely to implement, but may not, is a radar. Open Weather gives us access to their basic weather maps which include current weather data. This includes several things including wind speeds, precipitation, temperature, etc. However, because it provides us access to only current weather conditions, we cannot implement a full radar. Further weather map functionality is unlocked at 180 dollars a month, which is well above our budget. This would give us access to past and future predictions as well, opening up a wide range of possibilities. This plan would also give us access to a wider array of weather API's. It includes everything from the free plan, as well as a 4 day/hourly API, 16 day/daily API, and a 30-day forecast API. This would give the user more options for the route view of upcoming weather conditions. While all these functionalities would be a welcome addition to our app, they are unnecessary for the purpose of this project. The biggest piece of functionality we will likely leave out is a rerouting feature. This feature would

ask the user to change routes based off the weather or any other reason the user gives. This is likely too tall of a task for us in the amount of time we have. We would have to somehow communicate with the routing behavior of Google Maps and change it to work with the changing weather conditions. If we were to fully implement this application with unlimited resources, then this feature would be an obvious addition. Lastly, Open Weather features one last API that we have not discussed. This API is a UV Index. While this is a useful feature, it is unclear if it belongs in our application. There are more important areas of our design, and while this will remain a possibility, it will probably not find a place in our application, though it could be included in the specific location view of weather. This application easily allows us to add new pieces of information because the API's we are working with are relatively straightforward and very reliable. Because of this, we can easily remove and add components. So, while some of these items are in our future work section, some of them could find a place in the final version of the project.

### 4.3 Risks

| Risk | Risk Reduction |
|---|---|
| Tied to a single platform for development | Research options and choose a platform that will allow growth and support |
| Depend on service to provide weather data | Choose a weather API with a long history and good reputation |

### 4.4 Tasks

1. Research:
    a. Determine if the idea has already been created by someone else.
        i. Look for similar products, and what they did differently.
    b. Target Platform.
        i. Decide on target device.
    c. Programming Language
        i. Programming language should be used to create the application.
    d. Frameworks
        i. Open source vs. proprietary frameworks available.
        ii. Locate a mapping framework for the basis of the application.
        iii. Locate a weather forecasting framework that will provide a database of forecasted weather conditions.
    e. Database
        i. Determine the need for a database
    f. Source Control
        i. Determine how source code will be maintained.
    g. Determine if the idea is conceivable.
        i. Determine if it can be produced in the timeline.
        ii. Possess the needed skills to produce the application.
    h. Target Audience

        i. Desired users.
2. Software Development Approach:
    a. Decide on software process model for the application (Agile, waterfall, etc.).
    b. Assign roles for each member of the team.
    c. Determine meeting times and dates.
3. Design:
    a. User Interface.
        i. Greeting screen.
        ii. Loading Screen.
        iii. Menu Design.
        iv. Main Screen.
    b. Determine application architecture.
4. Implementation:
    a. Divide work between team members.
    b. Construct the application.
        i. Create the smaller components of the application
        ii. Merge smaller components, ensuring there is not issues.
5. Testing:
    a. Determine the application works as intended.
    b. Obtain user feedback.
    c. Determine need for improvements.
6. Documentation:
    a. Create a report outlining the application
        i. The goal of the application
        ii. The process of creating the application
        iii. The desired end user

### 4.5 Schedule

| Task | Timeframe |
|---|---|
| Research:<br>• Discuss the overall goal of the project<br>• Determine the application does not already exist<br>• Look into Frameworks needed<br>• Decide on programming language<br>• Determine target device for application<br>• Decide if the application is conceivable, in the given time frame and with the team's current skill set | 01/13/2020 - 01/27/2020<br>01/13/2020 – 01/20/2020<br><br><br>01/20/2020 – 01/27/2020 |
| Software Development Approach:<br>• Determine the software model for the project<br>• Assign team member roles for the project<br>• Discuss scheduling for future meetings | 01/27/2020 – 02/03/2020<br>01/27/2020 – 02/03/2020 |
| Design:<br>• Determine how the application should look when complete<br>• User interface<br>• User interaction with the application<br>• Navigation within the application<br>• Decide how the application architecture should interact | 02/03/2020 – 02/10/2020<br>02/03/2020 – 02/10/2020 |
| Implementation:<br>• Meetings to discuss progress with individual parts<br>• Decide how the work will be divided<br>• Construct the application<br>• Converge completed modules of the application as applicable | 02/10/2020 – 03/09/2020<br>02/10/2020 – 02/17/2020<br><br>02/17/2020 – 03/09/2020 |
| Testing:<br>• Test application for bugs<br>• Perform real world testing<br>• Get user feedback from testers | 03/09/2020 – 03/16/2020<br>03/09/2020 – 03/16/2020 |
| Documentation:<br>• Document the application<br>• Describe its purpose<br>• Explain how the application was designed and created<br>• Illustrate the need for the application | 03/16/2020 – 04/03/2020<br>03/16/2020 – 04/03/2020 |

### 4.6 Deliverables

- Android Studio Java code
- Heroku server
- Final report
- Website code

## 5.0 Key Personnel

**Madison Galloway** – Galloway is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas and will also be completing a minor in Mathematics through the University of Arkansas. She has completed relevant courses including Software Engineering, Database Management, Information Security, and Big Data Analytics and Management. She has experience in Java and C. This experience comes from both school projects and personal projects. Galloway will be the backend lead, writing code for the server and working as the dedicated liaison with the frontend team.

**William Henness -** Henness is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses including Software Engineering, Database Management, Programming Paradigms, Information Security, and Mobile Programming. He has experience with java programming, having completed an internship with Cerner on the FetaLink desktop team and several school projects. Additionally, he has experience with C++ and Python, mostly from personal and course projects. Henness will be the frontend lead, focusing on the Google Maps API and working as the dedicated liaison with the backend team.

**Audrey Timmerman** – Timmerman is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed Software Engineering along with Programming Paradigms and has experience in Java and writing simple apps that interface with a server. Timmerman will be the team lead and be responsible for overall organization as well as being on the backend team.

**David Turnbough** - Turnbough is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses including Software Engineering, Database Management, Cloud Computing, and Information Security. Turnbough will be on the frontend team, focusing on the user interface and testing as well as supporting other aspects of the frontend.

**Zachary Cantrell** — Cantrell is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas and has completed a minor in Mathematics at the University of Arkansas. He has completed Software Engineering, Programming Paradigms, and Computer Networks. He has experienced socket programming, mobile programming, and database programming. Cantrell will be on the frontend team,

focusing on getting and sending information with the server to ensure that the weather data is up to date, along with facilitating and other frontend projects.

**Nicholas Brinkley** - Brinkley is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses including Software Engineering, Database Management, and Big Data Programming. Brinkley will be on the backend team with a focus on setting up the Heroku server. He will the account holder for the Heroku free tier and help support any other backend functions.

## 6.0 Facilities and Equipment

The only equipment we will need for this is our own personal computers and Android Studio code development software. We may also need a Heroku server to store user data if we decide to go that route.

## 7.0 References

"Google Maps Platform Documentation." *Google Maps Platform | Google Developers*, Google, 2019, developers.google.com/maps/documentation.

Hall, Julie. "AAA: Nearly 100 Million Americans Will Embark on Family Vacations This Year." *AAA NewsRoom*, 20 Mar. 2019, newsroom.aaa.com/2019/03/100-million-americans-will-embark-on-family-vacations/.

"Route Weather." *Morecast*, Morecast, 2019, morecast.com/en/plan-your-route.