RainCheck: Final Report

Group G, Section 1: Kyle Goodwin, Alec Rydeen, Jacob Mitchell (Liam Costello not present for most of semester)

http://raincheck.hebab7bne5f3cfe6.eastus.azurecontainer.io:5000/index

Introduction:

The main idea for our app was to make the process of getting instant weather updates to plan outdoor activities easier and more reliable. Our whole group enjoys outdoor activities like skiing, sailing, and hiking, and weather has a large impact on whether these activities are possible or enjoyable. Our app can be used by anyone, anywhere. More specifically designed for personal use, so people can jump on before going outside and get the exact weather in the place they are planning to go.

Everyone knows when you wake up early, with a few hours of sleep, ready to ski, hike, surf, sail, whatever it may be, only to check the forecast, and see that it no longer looks as good as it did yesterday. With RainCheck, our goal was to allow a way for people who are already on location, to post “condition reports” for people at home to see and decide whether it is still worth it for them to make the trip. Every surfer knows the excitement of seeing a forecasted amazing day, only to feel dread when you arrive and see flat water. The problem we are trying to solve with our app is the unreliability and tedious process of getting the weather for places on weather apps. They are not always accurate, and if you want to look at multiple places, it can take a while to get the updates on it. RainCheck makes it easy and reliable to get weather updates on specific places, because what better way to get the exact conditions other than someone being there already?

RainCheck allows users to post the current weather conditions of any place to the app, giving two simple parameters: The location, and a description of the conditions. After posting, or not posting at all, users can view all the reports other users have made, and get the current weather conditions at a place of their choice. The app also allows

users to delete messages if they are now old/inaccurate, or do not provide any helpful insight.

Requirements:

The requirements for our product generally came from our user stories that we created, based on our user personas. After creating our three user personas and many user stories to go along with them, we came up with our feature requirements and quality attributes to be implemented later on. Here are a few examples of user stories that drove our development:

- “As an outdoor activist, I want to be able to keep track of multiple locations at once, as I may be planning to hit several different places that day”
- “As someone who travels and surfs a lot, I am constantly surfing at different locations that vary greatly. I want to be able to easily check locations that are not close to me, to know if I should travel there”
- “As a high school principal, I want the ability for users to remove their old posts so I can get the most accurate information to ensure the safety of my students.”
- “As a non-tech savvy hiker, I am easily overwhelmed by complicated webpages and applications. I would like a weather app that is easy to understand and navigate, to plan my hikes accordingly.”
- As a ski club coordinator, I want to be able to see the current conditions on multiple ski mountains so I can make an informed decision on where the club should meet.

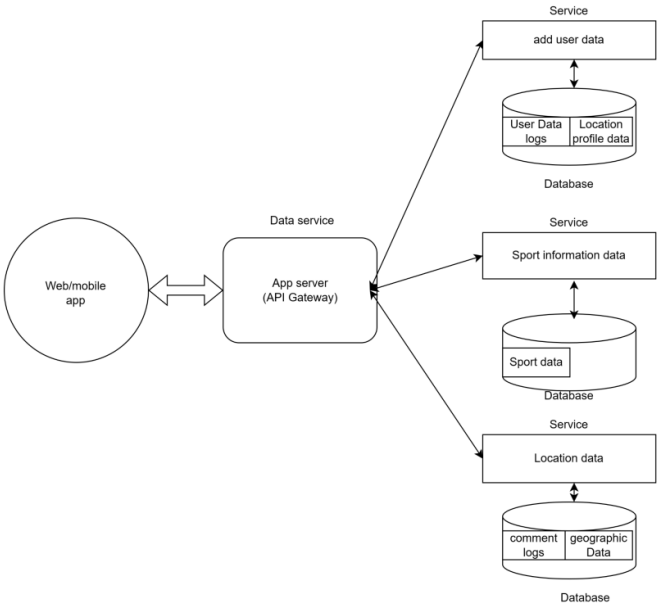
These are just a few of the user stories we created that helped shape our vision of what we wanted our app to become. When we made our user stories, we realized how different outdoor sports people can be, and we tried to capture that in our personas. Our old surf bum, Broth McGraw, is the reason we wanted to use a simple, easy to navigate, and nice looking user interface. We imagined people like Broth to be confused by more complicated apps, and just want a simple way to check the surf before he goes to shred

the waves. One of our other user stories, Chad Thunder, is a multi sport athlete interested in anything he can get to. Because of this he wants the most up to date information, as he may be doing multiple activities in one day. He is what gave us the idea to implement the delete function. With delete being implemented, it allows us to remove old, out of date data, which helps people like Chad Thunder out. Our final user story, Lee Smith, is a pretty average joe, but he is a trip planner for the local ski club. Because of this, he is interested in more resources than just the condition reports on our app. This inspired us to add a page with links to live webcams for some locations.Taking these stories into consideration, we had a few specific quality attributes that came to mind when brainstorming the final vision of the app:

- **Scalability**
    - If our app grows to a large scale, we want to be able to support many posts for a single place, as well as many posts for any place around the world, to make the app usable on a global scale
- **Usability**
    - Because our app is supposed to make finding weather conditions much easier for the user, we want our app to be very easy to use and streamlined so it is a quick process to find exactly what the user is looking for.
- **Reliability**
    - With weather being a very important part of people's lives, we want to make sure our app is running at the maximum efficiency at all times so people can get updates in case of a severe weather event.

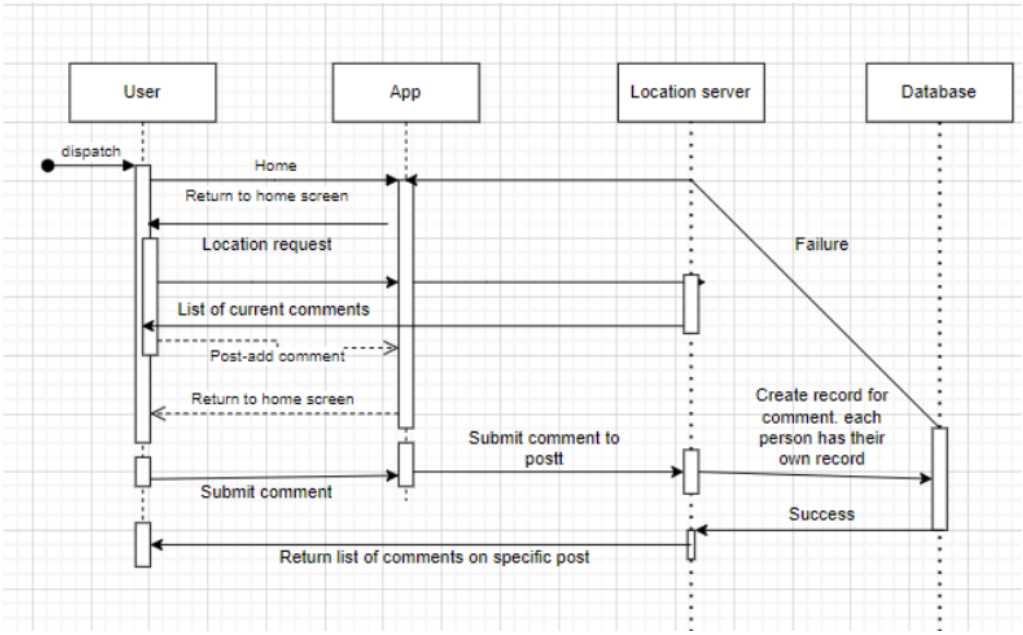
Designs:

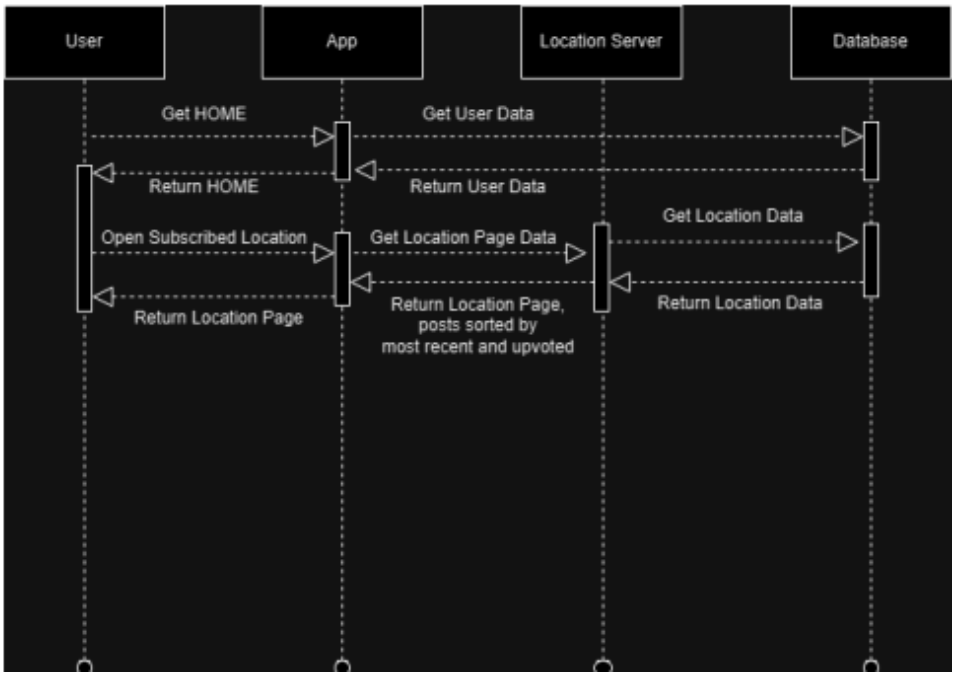
At the beginning of planning our app we had very ambitions goals of what we wanted to have done at the end of phase 5, which we added to our architecture and sequence diagrams:



Architecture:

Sequence:





Our final product has five main routes, some with parameters and all with returns:

- **“/”, “/index”**
    - These routes are access to the homepage of our app with a small info page, and links to other important features. This route does not use any methods.

- **“/create”**
    - This route is used for direction to our create a record page, which uses methods POST and GET
- **“/delete”**
    - This route is used for direction to our delete records page which used methods DELETE and GET
- **“/records”**
    - This route is used for direction to our view records page, which displays all the records in post order. This route does not use any methods.
- **“/weathercams”**
    - This route is used for direction to our Weather cam link page. This route does not use any methods.

Development and Testing

- When we got to phase 5 of our product, we had just gotten out of solving tons of issues with azure functions and docker. It took us weeks just to get our create and view records working locally on any one of our machines, and this really hindered our progress on the final product of the web app. By the time we got these working, we had about a week to try and create our two features and get them deployed before we presented. This leads into our goals for this phase:
- Bold/larger names mean they did more work than others on that specific task
    - Main goal: Implement two distinct features (**Alec Rydeen**, Kyle Goodwin)
    - Create and share presentation w/group members (**Kyle Goodwin)**
    - Encourage and keep constant communication (**Jacob Mitchell,** Kyle Goodwin, Alec Rydeen)
    - Create final report, documenting the whole process for project (**Kyle Goodwin, Alec Rydeen)**
- For our final product, we have implemented the following two features:
    - Delete function using JavaScript
    - Bootstrap for UI additions

- Bootstrap did not require much testing, but all testing was done locally. Multiple different iterations were created, but the final one decided on was used. When implementing bootstrap, our users' stories were kept in mind, to create a user interface, which is both easy to use, and efficient.
- When Implementing the delete function, we ran into a roadblock where the use of a form restricted us to only using the POST and GET methods for our delete function. To overcome this, a JavaScript function was created to use an input field with a separate action to actually use the DELETE method. Once this issue was overcome, it helped to avoid conflicts with other methods, and made our application more consistent.
- The delete function required testing for deleting different names from the database. We had trouble at first getting it to delete words with spaces and caps. Once we got it to a place where we were happy and deployed, it worked as intended.
    - Delete function removes all the data for a region, which is used to remove old data, as our apps functionality relies on having up to date, and accurate data.
- We had to do tons of testing for our data manager and function app early on in the project because of issues getting it to print correctly to the view route and add a record to the database. Most of this testing was done in azure developer tools, and knowledge gained from that was used to fix these issues as they arose. Most issues came from using different kinds of input fields, such as trying to input data in queries vs a body. This problem was identified and solved using Azure developer tools.

Reflections

- Overall we all learned a lot from this project, despite the roadblocks we faced throughout it. We learned to persevere when things weren’t going our way and that communication was one of the most important aspects in order to get all the

work done on time. We learned tons of new skills not only in keeping a constant stream of communication, but also technical skills as well. Learning backend database management and coding as well as front-end site development using tools like flask, azure, bootstrap, and HTML were all very valuable learning experiences.

- Future plans and implementations:
    - Search/Filter function for the viewing page, so you can look at just a specific place
    - Profile feature so you can save locations you view a lot
    - Comment on specific posts, goes along with profiles
    - Upvote/downvote feature to be able to boost better posts to the top
    - Update delete function after profiles are introduced, so you can only delete posts that you made