Samuel Roy
8/11/17
Final Report

Final Report – Weather Service

Byte Bistro weather app personalizer at a high level is a service that allows you to become more intimate with your local climate. Most weather services provide data (i.e. temperature, wind, etc.) but few take that data and tries to make it relevant to an individual. The goal of our weather service project is to allow people to personalize what they want to be reminded about in terms of the forecast. If a person wants to know when the temperature is above 80 degrees, they will receive an email and start putting on their beach attire. If you commute with a motorcycle, you may want to be alerted in case of upcoming rain or storm – our weather service will provide that insight.

First we start with the baseline of this project: google weather api. API stands for Application Programming Interface, within the realm of programing an API allows a developer to change parameters and access the applications assets (usually externally). This API allows us to gather data from Google's servers and serve the data to a user. During the data serve we can run checks on that data against user stored preferences on what they would like to be alerted about. The API is the backbone of this project, and is essential to writing an efficient app. Next we have the user database, this will store user session data and login information. The server backend will call from this database every night to check user preferences against weather data. There is also a mailer component of this project, which uses google secure Simple Mail Transfer Protocol (SMTP) to send secure Transport Layer Secured (TLS) emails to notify users. Email security is very important to the information technology space due to the many breaches

of user identifiable information. As a programmer that means my most important non-functional requirement is security. TLS (Transport Layer Security) is a secure protocol that allows email messages to be sent with privacy and data integrity. TLS is a subset of the Secure Sockets Layer protocol which uses encryption to secure data in transit. Simple Mail Transfer Protocol over TLS gives us exactly what we need to securely send emails to users and keep their personal information secure. Lastly we have the web frontend portion of this project. The web frontend provides a simple, easy to use layout for a user to view and change their preferences – as well as see the current weather and modify their location.

In this project we used the Agile process to stay on track and keep organized. In our team of four people I would find it hard to keep track of feature additions without pivotal tracker. This tool allows us to assign certain components of the project (big and small) to individuals on the team. I would say without a doubt that pivotal tracker was the most important agile programming concept I utilized in this project. We are constantly checking ourselves to make sure when someone is modifying project files they are testing and making frequent commits. This is so we can catch any bugs early before they burrow themselves deep into the project source. We have had to refactor the code in order to keep it organized and mainly add comments to the different snippets.

Our team weekly meetings were a good time to take stock on work completed (if any) and future ideas. I think the most important aspect of weekly meetings was really aligning everyone's expectations about the project. Given enough time individuals will deviate from a set plan – including me. A regular meeting helps shore up those deviations and allow me

personally to refocus on the task at hand. The other key aspect of these meetings is receiving constructive criticism on project ideas that have been passed around. If an idea is generally received poorly we can let that individual know that it may be out of scope. We had discussed using a Wiki solution to organize our documents but came to the conclusion for the scope of this project it was unnecessary. Instead we utilized Instant Messaging app “Google Hangouts” as I will discuss later.

Team communication is essential in this project. For this our team leveraged “Google hangouts” which allowed us to always be in chat communication with each other. For the meetings video chat is also a feature of google hangouts and gives us that face-to-face engagement not normally achieved online. In times where communication was sparse, I found sending emails reminding my team of milestones and deadlines was extremely useful in keeping that conversation alive. Though we are always in contact, not all members are available all the time – this definitely caused some issues of lack of communication. Most of these issues were resolved with email however, which is always a great resource.

There are a few things I would do differently on my next project. Mainly due to the learning curve of writing Junit test cases. My behavior in this project was to finish features, commit code, rinse and repeat until the project was built up. The mistake I made was not writing Junit test cases along the way. This is caused some headache trying to write many test cases for code that is not fresh in my mind. I would write the test cases immediately after finishing a module or component of the program. It could have just been laziness, but only after completing most of the project do I see the value in this.

Samuel Roy
8/11/17
Final Report

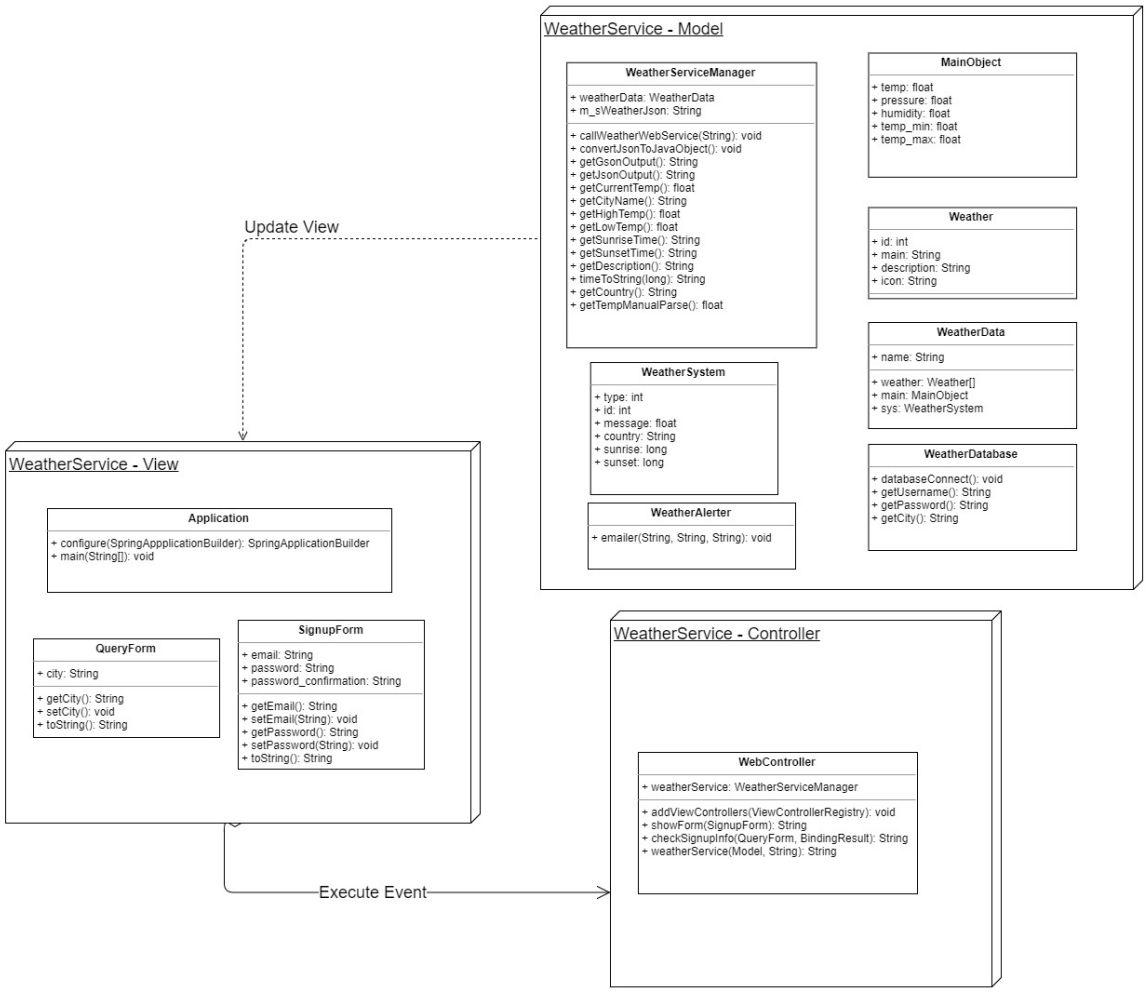
In pivotal tracker, we used a point based system to enumerate the value of different features to be added. Small additions being one, medium two, and large feature components a value of three. We had on average twelve points per iteration. If five new requirements (large) were added to the project, it would take us two to three iterations before all requirements are implemented.

As a requirement example I will use implementing the email alerting component. This requirement required finding a library that would allow us to send email addresses in a secure manner. After researching I determined the simplest way to implement this was to leverage Google's secure smtp servers. The library of choice was javax.mail, which can utilize a google mail host. The module parameters to send an alert included String variables "To address", "Message body", and "Message subject". We then take these variables create a new javax mail authenticator instance, send that instance into a new MimeMessage object – then transport using javax send.

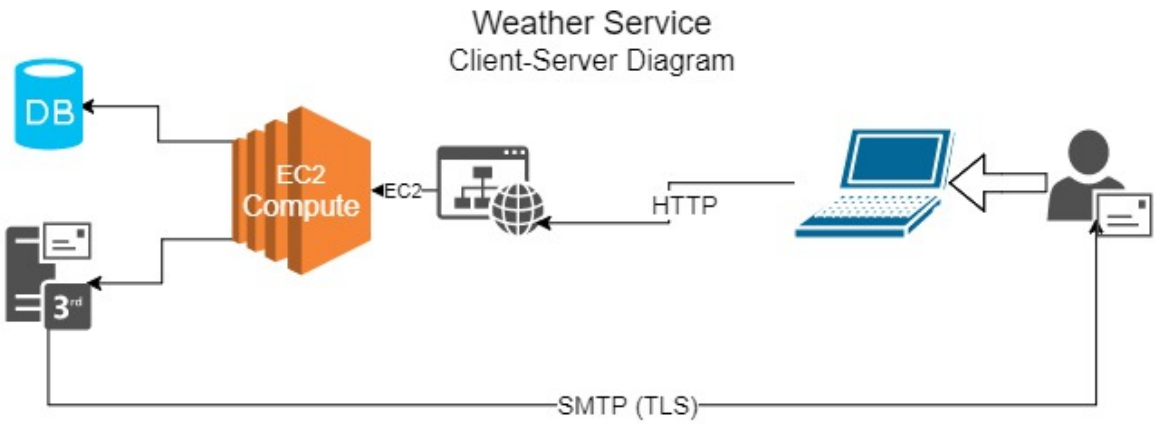
We used Model-View-Controller (MVC) which is a programming design model designed for user interfaces. We decided on this model because it really worked for our type of project, namely webserver – user – application. Simply put our Model handles all of the data and heavy lifting, the View presents the data to the user, and our Controller calls any components that the user needs. The advantages for us were separating the user interface functions with the underlying application itself – it made it easy to decide where to put functions because we knew how we wanted the data to flow.

Samuel Roy
8/11/17
Final Report

MVC UML Diagram:



Deployment diagram for server and client:



Samuel Roy
8/11/17
Final Report

During this project a few requirements changed. The biggest requirement that changed was implementing a weather map showing a handcrafted map of the region and associated temperatures (much like the TV news map). This requirement over time just revealed itself to be too time consuming and tedious for the scope of the project. And we decided to nix that feature request.

The non-functional requirements of this code I briefly touched on, but again I had security in mind from the get-go for this project. I made sure we had secure transport for the email-alerter because of this concern. We also wanted our code to be organized – so we went in and did periodic checks to make sure we didn't have any redundant or unused code that could be removed. All in all the final product was achieved to our specifications and scope of our project. Though requirements and expectations changed along the way, this is to be expected with the Agile process.

Thank you for reading this report

--Sam