Modern Mobile Development and Integration

Raspberry PI Weather Station

Dahn, Kyle

Computer Science & Information Systems Department

Minnesota State University Moorhead, 1104 7th Avenue South, Moorhead, MN 56563

Abstract:

According to Pew Research Center, since 2011 the number of US adults who own Smartphones has doubled. With this sudden rise in smartphone ownership and usage by end users the need for swift and robust development and integration of mobile applications has also risen. To demonstrate such a need I designed, developed, tested, and deployed my own mobile application on the Android platform.

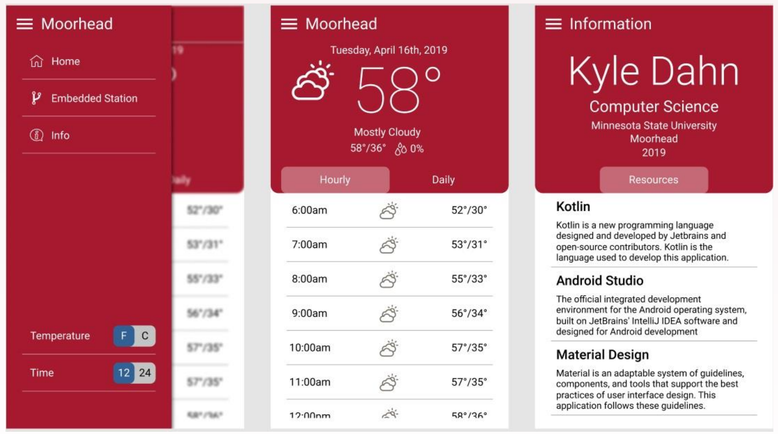
The design, development, testing, and deployment were done according to the software lifecycle following the scrum framework using modern architecture practices. This application proves the robustness of the software lifecycle, the agility of the scrum framework, and the durability of modern architectures while using contemporary programming languages. The project is significant in that it demonstrates the ability to rapidly produce an application for consumption and to connect the growing number of smartphone owners and users to local information.

This project is an extension to the projects of Ryan Anderson and Casey Vargo demonstrating the mobile front-end of a hardware to front-end progression. Ryan Anderson created a raspberry pi weather station to obtain and store data. The data collected by Ryan will be exposed by Casey through an API that my mobile application will consume.

Design:

The design of the application was created using Figma. Figma is a free online user interface (UI) tool to create, collaborate, prototype, and handoff designs.

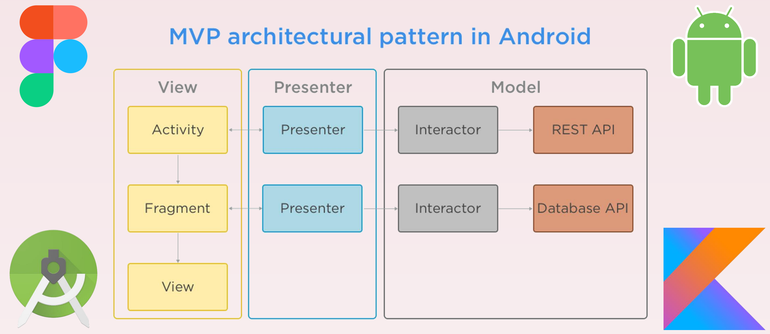
I designed the application to use the color palette of MSUM while following the guidelines and using the components of Material Design. I needed to create a simple interface that would show the local weather data retrieved from both the National Weather Service and Casey Vargo's API. Using elements of User Experience (UX) and UI I can draw the user's attention to important information by using larger or bolder text and icons as our eyes are drawn to larger or bolder items and icons first. I also grouped information depending upon importance, such as a detail of the current weather and a list of forecasted weather.

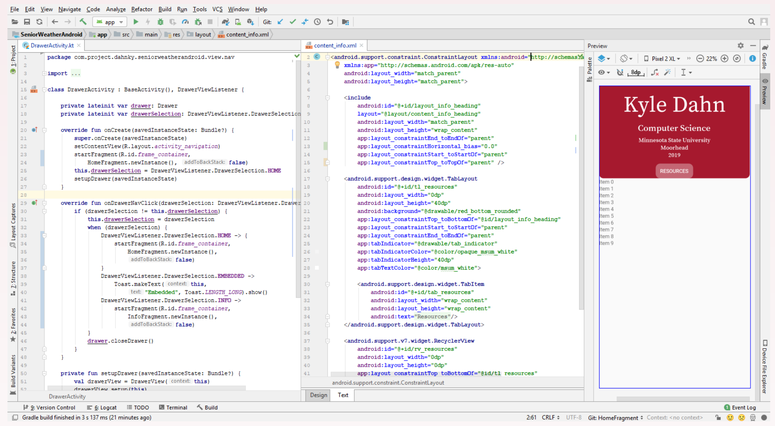


Development:

The development of the application was done in the Android Studio Integrated Development Environment using Kotlin as the base programming language. By using Android Studio I was able to supplement Kotlin with Groovy Gradle scripts to swiftly produce multiple build configurations for debugging and release. Android Studio also supports a preview of implementing designs into layouts. Finally, I can test the application across multiple Android versions and screen resolutions by using the additional emulator support included with Android Studio.

The architecture I used to build the application was Model-View-Presenter (MVP). MVP allows me to separate the application's concerns into smaller sections that increases testability, maintainability, and readability. I further split these sections into packages for the service, model, dependency injection layers, and each major screen.





Special Thanks:

Loren Johnson – *UX/UI Design Lead - Myriad Mobile*: For his guidance and knowledge of UX/UI and reviewing my design. Loren helped me in the initial phase of design as I stumbled using the darker color palette and suggested I transfer my initial design to an impactful minimalist style. I look forward to our continuous collaboration at Myriad!

Veronica Young-Cooksey – *Experience Designer - Delta Air Lines*: For her thorough and inspiring feedback after assessing the design. If it were not for Veronica there would be no date displayed! I look forward to making her designs a reality as we continue our work on a community developed mobile application.

Dr. Rhonda Ficek - *Professor - MSUM*: For her continued encouragement for me to seek the best of myself and guidance throughout my time at MSUM. Dr. Ficek's wisdom has been priceless as I navigate to the next phase in life and allowing me to share my newfound Android knowledge through guest lectures.

Resources:

In order to create this application I chose Kotlin as the programming language. Kotlin was developed by JetBrains with support from open-source contributors and made its first appearance in 2011. Kotlin is a statically-typed, cross-platform programming language with type inference.

Creating a feature-complete Android application is a daunting task for a single developer, but the greater open-source community has created various libraries to help ease the development process. Libraries range from project's created by Google and Square to a small group of people contributing on the weekend. Libraries used in this project include:

- ***Dagger***
- ***Retrofit***
- ***Material Dialogs***
- ***Material Drawer***
- ***Constraint Layout***
- ***Event Bus***

Testing & Deployment:

The final steps in providing a mobile front-end application to a user base is thorough testing followed by a well-planned deployment. The deployment and distribution planned for this application is through the Google Play Store.

In order to deploy through the Google Play Store there must be an associated Google Play Developer Account with the proper information followed by the creation of an application through the Google Play Console. After finishing development of the project the corresponding APKs and Bundles will be generated and signed. Once generated the APKs and Bundles can be uploaded through the Google Play Console through any of the predefined tracks, such as Internal Test, Closed, Open, and Production.

Before or after loading the APKs and Bundles a store listing can be drafted to inform and show users what the application will look like on their device along with the functionality and use cases. Pricing and must also be addressed before the application can be listed on the store. Once the application is uploaded a content rating questionnaire can be completed to receive a proper rating.

Once all the necessary steps are complete the application can be released on the store and made available to either a limited user-base or globally!

Google Play Developer Console

References:

1. Kotlin. (2019). Kotlin Reference Documentation. [online] Available at: https://kotlinlang.org/docs/reference/ [Accessed 5 Feb. 2019].
2. Android Developers. (2019). Android Developers Documentation. [online] Available at: https://developer.android.com/docs/[Accessed 5 Feb. 2019].
3. Material Design. (2019). Design. [online] Available at: https://material.io/design/design/ [Accessed 5 Feb. 2019].
4. Google.github.io. (2019). Dagger User's Guide. [online] Available at: https://google.github.io/dagger/users-guide [Accessed 5 Feb. 2019].
5. Retrofit 2.3.0 API. [online] Available at: https://square.github.io/retrofit/2.x/retrofit [Accessed 5 Feb. 2019].
6. Soral, R. (2019). Architectural Guidelines to follow for MVP pattern in Android. [online] AndroidPub. Available at: https://android.jlelse.eu/architectural-guidelines-to-follow-for-mvp-pattern-in-android-2374848a0157 [Accessed 5 Feb. 2019].
7. Pew Research Center. (2019). Mobile Fact Sheet. [online] Available at: https://www.pewinternet.org/fact-sheet/mobile/ [Accessed 3 Feb. 2019].