Iter: A METAR Browsing & Flight Charting Mobile Application

Isaac Braun
School of Computing
Southern Adventist University
Collegedale, Tennessee 37315
Email: isaacbraun@southern.edu
Telephone: (419) 706-0480

***Abstract*—Small business and private pilots do not have a efficient and accessible method of viewing both current and forecasted METAR weather data alongside flight path charting and comparison on a mobile device. The lack of this resource hinders a pilot's capability to plan the most efficient and safe flight. We propose and execute the development of a cross-platform mobile application, *Iter*, to fulfill the desired need in the market. Features include current and forecasted graphical METAR browsing, viewing of decoded METAR data, dual flight path charting, and automated path comparison. The product required the development of many modules including METAR Station Model, Search, Flight Path Selection, and Flight Path Comparison. Development was be done using React Native and split into four sprints of three weeks each. Unit/Module testing was done with *Jest* and acceptance testing was done through a survey completed by a user test group.**

I. Introduction

A. Motivation

Without sufficient knowledge of the weather, pilots can unknowingly put themselves or their passengers in danger. With knowledge of current and forecasted weather conditions, pilots can choose the best flight path to allow for the most safe flight possible. This decision would be greatly aided by the ability to lay out two flight paths and see current and forecasted weather data along those paths. In addition, it would be greatly beneficial for an application to compare user-selected flight paths based off weather data and provide information on which is the most favorable. Unfortunately, an accessible application that offers weather data browsing plus flight path charting does not exist.

A common form that the aviation field uses for viewing weather data is called a METAR. It is one of the most common formats in the world for communication observable weather data and is highly standardized by the *International Civil Aviation Organization*. Unfortunately, what the METAR acronym stands for is not as standardized. The *Center for Environmental Data Archive* defines it as *Meteorological Terminal Aviation Routine Weather Report* while the the United States Federal Aviation Administration shortens its meaning to just *Aviation Routine Weather Report*. METARs are more commonly displayed in a text format but can also be displayed in a graphical format, which allows for easy and quick information gathering when they are displayed on a map. Fig. 1 and 2 give a excellent interpretation of graphical METARs.

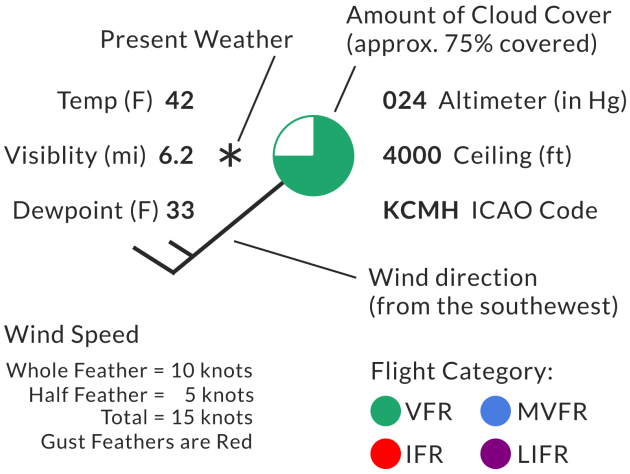

Fig. 1. Interpretation key for graphical METARs using the Station Model.

While there are accessible mobile applications that feature METAR browsing, such as *Weekend Flyer* [1] and *AeroWeather Lite* [2], they do not provide forecasted data or flight planning to the desired extent. *AeroWeather Lite* provides simple METAR browsing but no flight planning, while *Weekend Flyer* attempts to do both but does not fulfill the path planning desires to satisfaction.

The goal is to have a simple and accessible, as well as visually pleasing, cross-platform mobile application that provides these features. The proposed app will display METARs on a map for simple browsing and deciphered weather data when a specific METAR report is selected. It will also allow two flight paths to be marked out, displays METARs along those paths, and compare the specified alternate flight paths to determine which one is more suitable based on the weather conditions.

B. Problem Statement

Small business and private pilots do not have a efficient and accessible method of viewing current and forecasted METAR data alongside flight path planning based on that weather data. The lack of this resource hinders a pilot's capability to plan

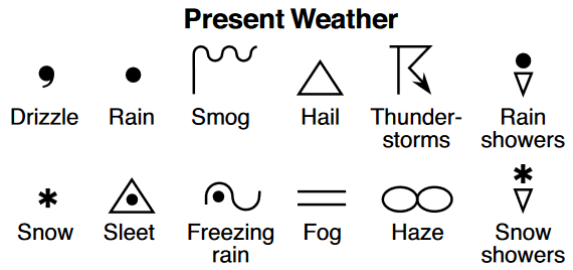

Fig. 2. Interpretation key for *Present Weather* symbol used in graphical METARs. Figure sourced from the NYSED Reference Table for Physical Setting/Earth Science [3].

the most efficient and safe flight. The proposed mobile application, *Iter*, will satisfy these desires in an efficient, visually pleasing, and accessible way. It will provide aviation weather data browsing, simple flight path planning with weather data displayed along the routes, and flight path comparison based of weather conditions.

II. Setting

A. Existing Applications

There are multiple mobile applications that provide aviation weather data browsing and a few that provide some form of flight path planning. Some are too simplistic while others are extremely complex and meant for much more than weather browsing and planning. Table I gives a quick overview of what features the prominent existing applications offer.

1) Weather Data Browsing: There are a few mobile applications that provide simple METAR browsing. Both *AeroWeather Lite* [2] and *METARs Aviation Weather* [4]. Both applications provide browsing of current METAR data in a text and map format and can be be set up to update the user when METAR data changes regarding specified criteria. *AeroWeather Lite* also provides forecasted weather conditions in TAF form and is free, while *METARs Aviation Weather* costs $6.99 USD. They both fulfill the METAR browsing desire, but not the support of flight path planning.

2) Flight Path Planning: The two best mobile applications found that provide flight planning are *WeekendFlyer* [1] and *Garmin Pilot* [5]. While both applications are free they are not very similar. *WeekendFlyer* is a very simple application that provides very simple weather data viewing and flight path planning. The usage of the app is not extremely clear or easy and the flight planning is very basic, so while it comes close it does not fulfill the desired feature set. *Garmin Pilot* is a highly rated and complex application that provides much more than the desired feature set. It provides weather data viewing and flight planning, but does not provide alternate flight paths or path decisions. These features may fulfill some of the specified desires but not to the full capacity.

3) Influencing Applications: Even though it is not an weather application, *Google Maps* [6] has a well designed UI/UX specifically in regard to the routes. Evaluation and inspiration from it will be helpful in the development of the proposed application.

TABLE I
Existing Application Evaluation

	Browsing	Charting	Evaluation
AeroWeather Lite	✓	×	×
METARs Aviation Weather	✓	×	×
WeekendFlyer	✓	✓	×
Garmin Pilot	×	✓	×

B. Literature Review

1) Cross-Platform Development: With many options for mobile application development available, exploring the field was useful in the decision of what development platform to choose. Support for the two major mobile platforms, iOS and Android, is essential for the proposed application. Therefore the options available for development are limited to using a cross-platform development environment or developing the application twice on separate native platforms [7]. In regards to simplicity and time, a cross-platform approach reduces the amount of learning required [8] since code can be reused for both platforms and in turn reduces time spent in development as well. With development being handled by a singular person, reducing the time and learning required would be greatly beneficial, making cross-platform development a perfect choice.

This lead to exploration being turned towards general cross-platform development guidance and specific platform information. Many useful bits of information were gathered through this exploration:

1) Navigation Differences: iOS and Android handle page navigation differently, so be aware of forcing navigation incorrectly [9].
2) Test Often: things can break easily when developing for multiple platforms, so test often [9].
3) User Roles: put yourself in the position of the user and ask “Why would I use this app?” [9]. This seems very useful to keep in mind while developing in order to make sure what is being created is actually useful to the user.
4) Don’t Fear Modules: Use the resources available. Do not make life unnecessarily hard [9]. Sometimes developers are tempted to do everything themselves, this is a reminder not to do that and make use of the great development already done by others.
5) In a study done by Biørn-Hansen et al. regarding animations in mobile applications React Native was found, in most cases, to require the least amount of device hardware to perform animations [10].

2) UI and UX: Since many of the existing aviation weather applications do not have good UI/UX design, this is a major focus for the product. Therefore, exploring was done in this area with many valuable points of information being found.

1) Visual/Functional Consistency: it is extremely important to make sure that the UI design is consistent, both visually and in behavior. This makes for easier navigation and understanding of the program [11].

2) Apply Fitt's Law: design larger targets for more important items and reduce the total amount of targets needed to complete a task [11].
3) Postel's Law: "Be liberal in what you accept, and conservative in what you send." [12]
4) Jakob's Law: UX should work similarly to the way other websites/apps do since users are used to them [12].
5) Engineer for Errors: useful error messages that help guide the user are extremely important and can help to retain the user rather than turn them away when there is an error [11].

3) Decision Algorithm: The product includes a system that can decide the better of two flight paths depending on weather data criteria. A couple areas were explored to gain the needed for the development of this system.

1) Some form of comparison or decision algorithm will be the method of making this decision. Purdila and Pentiuc published a paper discussing the building of a fast decision tree algorithm that can be used against a large dataset with minimal loss of accuracy while using as little memory as possible [13]. The paper discusses their proposed algorithm in depth and concludes that it is successful in usage against large datasets. This information may be useful in the development of the product and therefore is mentioned.
2) The algorithm decides based on weather data, meaning the knowledge of what conditions or values are more preferred for flying is needed. The majority of information gathered in this area was found from a chapter in a FAA handbook titled *Weather Theory* [14]. It covers the areas of "temperature (heat or cold), moisture (wetness or dryness), wind velocity (calm or storm), visibility (clearness or cloudiness), and barometric pressure (high or low) [14]." The information found in this handbook will be used to make the decision algorithm work as well as it can.

III. Proposed Product

A. Product Description

Our proposed product to fulfil the need is a cross-platform mobile application titled *Iter*, which will be easy to use and aesthetically pleasing. It will provide many features including, but not limited to, graphical METAR browsing, viewing of decoded METAR data, multiple flight path charting, and flight path comparison.

B. Requirements

A simple Use Case Diagram for the proposed *Iter* application is displayed in Fig. 3. This shows what a user can do with the application and how the process flows. For more information about the application requirements, see Appendix A.

C. Major Tasks

This product will require quite a few major development tasks which are outlined in the following sections.

1) Design: Designing the application UI and UX is a highly important task because it effects how the development is done and has such an impact on user's reception of it. If the UI or UX is terrible, the application will likely be passed by users searching for applications in this category. For these reasons the design will be completed first. A full design of the application screens and UX will be completed using InVision Studio.

2) Map Unit: The map unit will handle the display and user manipulation of the aviation focused map. This unit will likely employ one or two premade modules/plugins to handle the display of the map.

3) Weather API Calling Unit: This unit will handle calls to the weather API to retrieve data. Other modules that require weather data will call this unit with parameters to obtain it.

4) METAR Station Model Unit: In order to display the METAR data on a map efficiently, the data is organized into a Station Model as shown in Fig. 1. The creation of each separate METAR Station Model will be handled in this unit. It will take weather data as parameters and return an Station Model object or view.

5) METAR Placer Unit: METAR Station Models are placed on the map at the stations that the data was pulled from. The placement of the METARs and interaction with the map will be handled by this unit.

6) METAR Detailed View / Search Module: This unit handles the display of METAR data in a more detailed and explained manner when a singular METAR is selected from the map. This also includes the functionality of the search bar used to find locations or METARs.

7) Flight Path Module: Input, calculation, and map integration will be handled by the Flight Path Module. This module will integrate multiple units not only to handle path input and computing/linking of the path, but to interact with the Map unit and others to correctly display the path and update what METARs are displayed. This module will be relatively large and take significant development time.

8) Flight Path Comparison Module: The Flight Path Comparison Module will compare the user's selected paths and make a decision on which is more optimal. This includes the decision algorithm unit along with any other units required to interact with the application view to display comparison results to the user.

9) Forecast/Timeline Unit: This unit will handle input from the Forecast Timeline UI element and change what data is displayed determined by the time selected on the Forecast Timeline.

10) Integration: All of the modules will be integrated together to make the app cohesive and completely functional. This may require the development of a few units to achieve the desired result.

D. Development Sprints

Development will be completed in sprints of sections. The sections will consist of units or modules that work together or are prerequisites for other sections. An overview of this

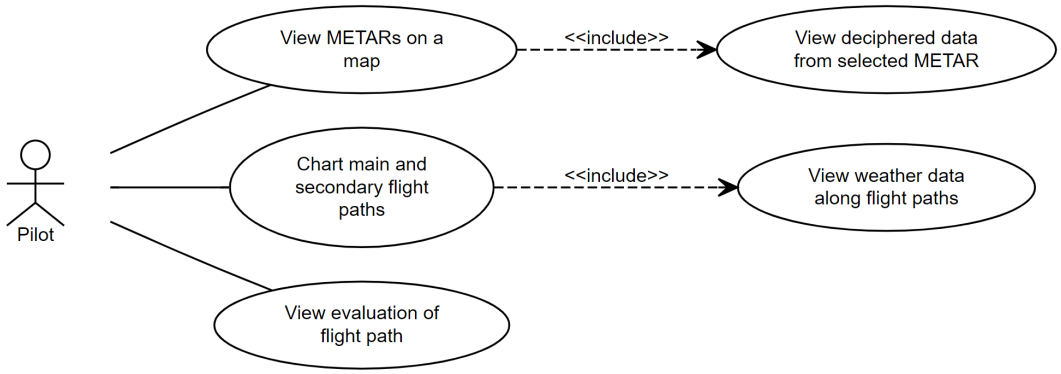

Fig. 3. Simple Use Case Diagram for the proposed *Iter* application.

TABLE II
DEVELOPMENT SPRINTS

Order	Included Tasks	Est. Time
1	Design, Map Unit, Weather API Calling Unit	20 hrs.
2	METAR Station Model Unit, METAR Placer Unit, Forecast/Timeline Unit	20 hrs.
3	Detailed View / Search Module, Begin Flight Path Module	20 hrs.
4	Flight Path Module, Flight Path Comparison Unit	20 hrs.

process along with the estimate time for each section is shown in Table II.

E. Deliverables

Deployable mobile *Iter* application supporting iOS 14+ and Android 11+. The application will be available as an Android APK or as a preview on iOS through the *React Native* development environment tool *Expo*.

F. Tools / Resources

Basic list of the tools and resources used in development of the product. Since iOS application deployment requires MacOS, design and development will be done on Microsoft Windows and Visual Studio Code but then transferred over to MacOS and XCode to deploy the iOS version of the product..

- *Microsoft Windows*: Operating system used to design and develop the product.
- *Apple MacOS*: Operating system used to do minor adjustments and deployment of the iOS product variant.
- *Microsoft Visual Studio Code*: IDE used for code editing and development.
- *Apple XCode*: IDE used for minor adjustments and deployment of the iOS product variant.
- *React Native*: Cross-Platform Environment of choice.
- *weatherapi.com API*: Weather data API used to obtain the needed weather data.
- *InVision Studio*: Design software used to design the UI and UX.

IV. TESTING PLAN

In order to ensure the application will provide the required features and perform smoothly, multiple testing methods will be implemented. This will include testing before and during development along with feature and UX testing by a group of test users. The testing methods are detailed in the following sections, with an overview displayed in Table III.

A. Test-Driven Development

The application will be developed using a test-driven method, meaning that as the application is being developed tests will be written first before unit or module code. As the application will be written in *React Native*, *Jest* (an included testing package) will be used to write tests that the following code must pass.

Tests will be written for all units and modules, as well as for the integration of modules. System testing will be done by consulting sessions with other developers to assure quality of the code base along with end-to-end testing through *Detox*, a test library used to emulate user interaction with an application and test user reactions. Unit testing will require a 98% pass rate, module testing 95%, while integration and system tests will require a 90% percent pass rate.

B. Acceptance and UX Testing

The test group will be asked to test the application by attempting to complete a set of tasks that fulfill the requirements of the acceptance tests. For each task, the users will be asked to complete a set of steps, answer if it was able to be completed, and compare the process with regards to ease and efficiency, Ease and efficiency will be graded on a number scale from 1 to 10, with 10 being the best. Lastly the users will be asked to write any other notes they thought of while going through the

TABLE III
TESTING METHODS AND PASS RATES

	Method	Min. Requirement
Unit	*Jest*	98% Pass Rate
Module	*Jest*	95% Pass Rate
Integration	*Jest*	90% Pass Rate
System	*Detox*	90% Pass Rate
Acceptance	User Testing	100% Completion & $\geq$ 7 in Scoring Sections

process. The acceptance tests are listed below along with the specific steps and compared result detailed for one of them. Specifics for each acceptance test are detailed in Appendix B. All acceptance tests must be reported as completed by the users and receive scores of at least 7 in the other two categories in order to pass.

The acceptance tests that will be compared are:

1) View and browse METAR data on a map, both current and forecasted data.
2) View more detailed data from a specific METAR when selected.
    a) Acceptance Test Steps
        1) Open application
        2) Browse map for any METAR
        3) Select METAR from map
    b) Expected Result: new view opened with decoded data from the selected METAR.
3) Chart a single flight path and be able to easily see weather data along the path.
4) Chart an alternate flight path and be able to easily see weather data along the path.
5) Chart multiple points for one flight path and be able to easily see weather data along the path.
6) Have flight path compared to determine which path is more suitable based on weather conditions.

V. DISCLAIMER

Everything in this paper until this point has been the proposal for *Iter*. The following sections document what actually happened in implementation, the results, reflection, and *Iter* documentation.

VI. IMPLEMENTATION

Development of *Iter* was successful and enjoyable but was not without its troubles. The planned sprints went almost as planned but with some minor changes. Some details of development also had to be changed and there is more work to be done in the future, both of which are outline below.

A. Sprint 1

The first sprint took around 21 hours and was focused on UI/UX design, researching and connecting with the *Aviation Weather Center* API, and setting up the Map Unit.

B. Sprint 2

The second sprint was a heavier sprint with about 31 hours spent with the majority of time spent on the creation of the METAR Station Model Unit. Following that the Detailed View Module and Forecast/Timeline Unit were completed.

C. Sprint 3

The third sprint took up approximately 26 hours and was spend heavily on just two major modules. The Search Module took a long time to get working correctly and as it was required for the Flight Path Module, needed to be completed before finally beginning and completing the Flight Path Module.

D. Sprint 4

The last sprint had around 31 hours put into it. It primarily was focused on completing the Flight Path Comparison Module, but was also used to polish the application more and complete some final testing.

E. Issues/Changes

1) Viewing Flight Paths: METAR Station Models are not limited to those near the flight path since the amount displayed depending on the zoom level was sufficient enough to see weather data relevant to the path.

2) Test Driven Development: since *Jest* and *React Native* were relatively new to the developer it was not possible to do true test driven development as writing tests for UI components that had not been figured out yet did not make sense. *Jest* was still used for testing many of the functions inside the project and a survey was used with a test group, but *Detox* was not used for end-to-end testing as it would have taken too much time to figure out compared to the benefit it would provide.

3) Forecasted Data: The original thought was to gather data from weather APIs and create forecasted data in a METAR formatted system, but this proved to not be worth the time or resources for the small benefit it would provide. Instead, a different format of weather data, TAFs, were used as they provide the forecasted data for each airport station. The forecasted data in these TAFs are displayed in the station models and highlighted in green to make sure it is known that it is forecasted data.

F. Android Storage Issues

The application uses a storage system that was working extremely well, but on Android it runs into a file size limit while using Expo. If it is built as a standalone app there are ways to solve this issue but currently it will not store any data from the API when run on an Android system.

G. Future Work

1) Deployment: Continued development and eventual deployment of *Iter* to both the *Google Play Store* and *Apple App Store* is desired.

2) Flight Path Features: Adding separate indicators for each flight path on the map view that displays what the path is and the estimated travel time is along with showing the estimated aircraft progress along the path when scrubbing the timeline is also desired.

VII. Results

All of the functional requirements where fulfilled and a cross-platform deployable application was created. The time estimates for the sprints in Table II did not turn out to be accurate. An average of 27 hours were spent on each sprint as everything took longer than expected, for example the search module that took many hours just to find and create a solution that would begin to function. With that, *Iter* is a beautiful and full-functioning application that fills the niche spot in the market it set out to.

VIII. Documentation

A. Building/Running

As *Iter* is not published to the app store, it can currently only be run by cloning the *GitHub* repository and running three commands.

1) cd iter-app
2) npm install
3) expo start

After this the *Expo* server will be running and it can be run by scanning the QR code with the *Expo Go* app on either *iOS* or *Android*.

B. Testing

Automated *Jest* tests can be run from inside the ”iter-app” directory with the command: npm test.

IX. Testing

As mentioned, the developer struggled with testing functions or units that updated views or states. So, many functions do not have automated tests but were tested through manual acceptance testing and the survey group. Outside of those types of functions, most functions and units were tested extensively using *Jest*.

X. Reflection

This project was very intriguing and fulfilling, even though there were things that should have been done differently. Some of the knowledge gained from early research about cross-platform development was not used, for example to test on both platforms often, and thus made development a bit more difficult when it came to debugging issues on *Android*. As mentioned test-driven development was attempted but not actually accomplished, and so it is desired to attempt this method again in a future project as it seems to have interesting benefits. So much was learned over the process of the project, not only from a technical standpoint but from aspects of design, client relations, and good development practices.

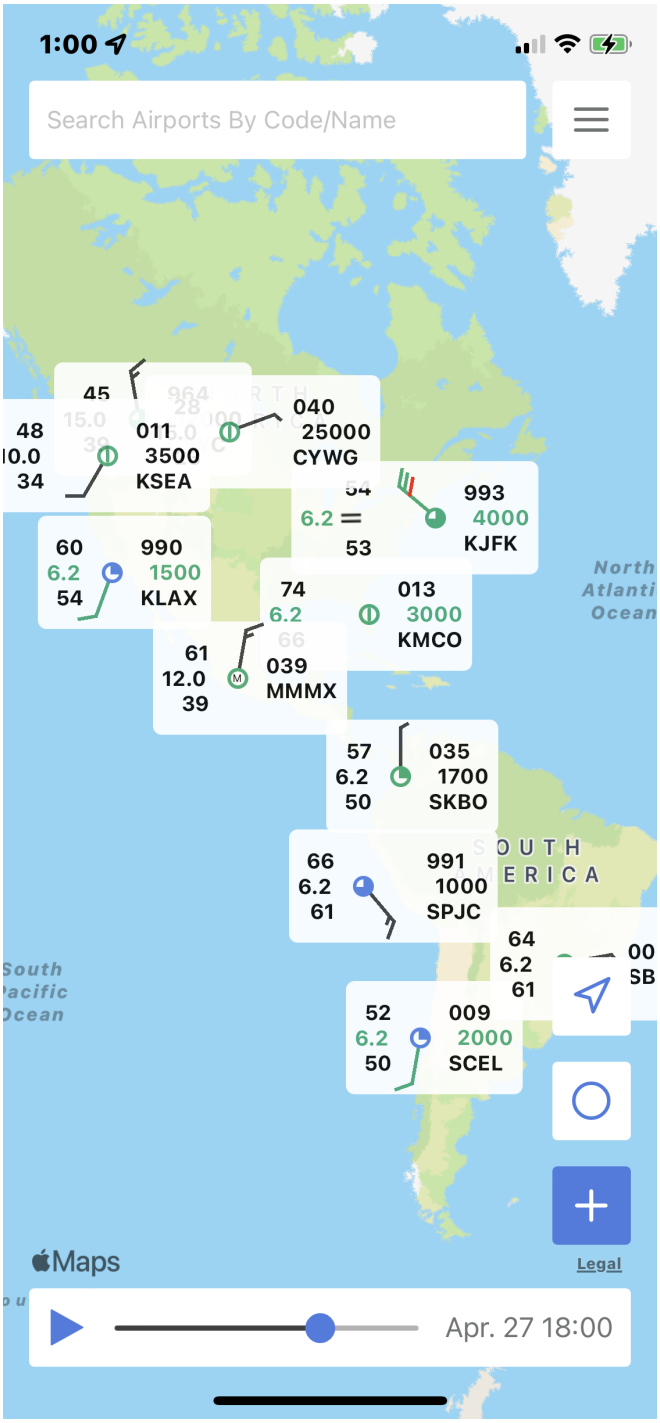

Fig. 4. *Iter* Home Screen displaying current and forecasted data.

XI. Conclusion

With small business and private pilots not having an efficient and accessible method of viewing both current and forecasted METAR weather data alongside flight path charting and comparison on a mobile device, their ability to plan the best and most safe flight is hindered. With the development of our cross-platform mobile application, *Iter*, this specific but

highly important section in the market is filled.

APPENDIX A
REQUIREMENTS

1) Role:
   a) Pilot
2) Requirements
   a) Functional
      i) View aviation specific weather data in graphical format (METAR), both current and forecast, on a map. (Est. 20 hrs.)
      ii) View more detailed data deciphered from a selected METAR. (Est. 10 hrs.)
      iii) Chart main and alternate flight paths and view weather data along those paths. (Est. 20 hrs.)
      iv) Chart multiple points for one flight path and see weather data along the path. (Est. 5 hrs.)
      v) Have flight paths compared by system to determine which path is more suitable based on weather conditions. (Est. 20 hrs.)
   b) Nonfunctional
      i) Price: Free
      ii) Application must be efficient and simplistic, complex only if sought out by user.
      iii) Application must be visually pleasing.
      iv) Supported Platforms (Minimum Versions)
         A) iOS 14
         B) Android 11

APPENDIX B
ACCEPTANCE TESTS

1) View and browse METAR data on a map, both current and forecasted data.
   a) Steps
      1) Open application
      2) Browse map for any METAR
      3) Move time slider to change time of displayed data
   b) Expected Result(s):
      i) Displays map with METARs accurately placed on it
      ii) Data clearly displays forecasted weather changes with time slider movement
      iii) Data displayed for each METAR matches data pulled from Weather API service for that METAR
2) View more detailed data from a specific METAR when selected.
   a) Steps
      1) Open application
      2) Browse map for any METAR
      3) Select METAR from map
      4) Press ”Decoded Details”
   b) Expected Result(s): new view opened with decoded data from the selected METAR.
3) Chart a single flight path and be able to easily see weather data along the path.
   a) Steps
      1) Open application
      2) Press ”+” button
      3) Input origin
      4) Input destination
      5) Press ”View”
   b) Expected Result(s):
      i) Map view updates to display path
      ii) Map view zooms to display full path
4) Chart an alternate flight path and be able to easily see weather data along the path.
   a) Steps
      1) Open application
      2) Press ”+” button
      3) Press ”Alternate Path” table
      4) Input origin
      5) Press ”+” to add a waypoint
      6) Input waypoint
      7) Input destination
      8) Press ”View”
   b) Expected Result(s):
      i) Alternate Path inputs are separate/different than Main Path
      ii) Map view updates to display path
      iii) Map view zooms to display full path
5) Chart multiple points for one flight path and be able to easily see weather data along the path.
   a) Steps
      1) Open application
      2) Press ”+” button
      3) Press ”Alternate Path” table
      4) Input origin
      5) Press ”+” to add a waypoint
      6) Input waypoint
      7) Repeat for as many waypoints as desired
      8) Input destination
      9) Press ”View”
   b) Expected Result(s):
      i) Alternate Path inputs are separate/different than Main Path
      ii) Map view updates to display flight path following all points inputted.
      iii) Map view zooms to display full path
6) Have flight paths compared to determine which path is more suitable based on weather conditions.
   a) Steps
      1) Open application
      2) Press ”+” button
      3) Input ”Main” and ”Alternate” Paths that are different but have the same origin and destination.

4) Press ”Compare”

b) Expected Result(s): app view updates to display which path has more favorable weather along its route

REFERENCES

[1] Thomas Court Software Development LLC, “Weekend Flyer,” [Accessed: Sept. 28, 2021]. [Online]. Available: https://apps.apple.com/us/app/weekend-flyer/id1056694984

[2] Lakehorn AG, “AeroWeather Lite,” [Accessed: Sept. 28, 2021]. [Online]. Available: https://apps.apple.com/us/app/aeroweather-lite/id288286079

[3] New York State Education Department, “Reference Tables for Physical Setting/EARTH SCIENCE,” [Accessed: Oct. 3, 2021]. [Online]. Available: http://www.nysed.gov/common/nysed/files/programs/state-assessment/2011-earth-science-reference-tables-english.pdf

[4] Aviation Mobile Apps LLC, “METARs Aviation Weather,” [Accessed: Sept. 28, 2021]. [Online]. Available: https://apps.apple.com/us/app/metars-aviation-weather/id1000246864

[5] Garmin DCI, “Garmin Pilot,” [Accessed: Sept. 28, 2021]. [Online]. Available: https://apps.apple.com/us/app/garmin-pilot/id340917615

[6] Google LLC, “Google Maps,” [Accessed: Sept. 28, 2021]. [Online]. Available: https://apps.apple.com/us/app/google-maps/id585027354

[7] A. Biørn-Hansen, T.-M. Grønli, and G. Ghinea, “A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development.” *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–34, Aug. 2018. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=bsu&AN=147795686&site=ehost-live&scope=site

[8] M. Huynh, P. Ghimire, and D. Truong, “Hybrid app approach: could it mark the end of native app domination?” *Issues in Informing Science and Information Technology*, vol. 14, pp. 49+, 2017. [Online]. Available: https://link.gale.com/apps/doc/A498734445/AONE?u=tel_a_sau&sid=bookmark-AONE&xid=ca9e8ffc

[9] C. Preimesberger, “10 Tips for Successful Cross-Platform Mobile App Development.” *eWeek*, pp. 1–1, Aug. 2015, [Accessed: Sept. 28, 2021]. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=109350268&site=ehost-live&scope=site

[10] A. Biørn-Hansen, T.-M. Grønli, and G. Ghinea, “Animations in Cross-Platform Mobile Applications: An Evaluation of Tools, Metrics and Performance,” *Sensors (Basel, Switzerland)*, vol. 19, May 2019. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=136449290&site=ehost-live&scope=site

[11] N. Babich, “The 4 Golden Rules of UI Design,” Oct. 2019, [Accessed: Sept. 28, 2021]. [Online]. Available: https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/

[12] J. Yablonski, “Laws of UX,” 2021, [Accessed: Sept. 28, 2021]. [Online]. Available: https://lawsofux.com/

[13] V. Purdila and S.-G. Pentiuc, “Fast Decision Tree Algorithm,” *Advances in Electrical and Computer Engineering*, vol. 14, no. 1, pp. 65+, 2014. [Online]. Available: https://link.gale.com/apps/doc/A585800921/AONE?u=tel_a_sau&sid=bookmark-AONE&xid=1bd1aed0

[14] FAA, “Weather Theory,” [Accessed: Sept. 28, 2021]. [Online]. Available: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/phak/media/14_phak_ch12.pdf