Context-Aware Clothing Recommendation

Project Plan by:
William Parr
Tyler Witte
Ethan Wieczorek
Nickolaus Eaton
Christian Ehlen
Nicus Hicks

List of Figures

List of Tables

1 Introductory Material

1.1 Acknowledgement

Professor Goce Trajcevski will be assisting in the technical advice and equipment retrieval for this project.

1.2 Problem Statement

Sometimes it is difficult to predict what the weather will be later in the day, and someone could properly prepared for an earlier part of the day with the clothes they have on currently, but unseen changes in the weather will bring problems for this person. There is also the case of someone travelling for an extended period of time. Not being native to the area, a traveler will not have the same idea of what to wear at a location as the people who live there, so this project attempts to create a solution to these problems presented.

This project aims at providing a comprehensive solution for a clothes recommendation system. The main idea is to couple data from multiple sources and integrate it in a manner that will enable the users to select outfit and/or plan which clothes to pack for an expected trip. As a motivational example-scenario: upon detecting that the user has woken up (e.g., Fitbit), a trigger is fired that connects to a Weather Channel API. Given the weather prediction and the tasks and places that the user has entered into their calendar for that particular day, the application will provide a recommendation for selecting the items from the closet to be worn/carried for the day. The application will select clothing from the user's database of clothing items. Each clothing item will be uploaded by the user and tagged with a specific temperature rating. The recommendations will choose the user's clothes based on these recommendations.

1.3 Operating Environment

This product will mostly be used on smartphones with some users potentially using a web version of the application. Operating environments will not really affect our application any more than the normal operating environments of the user's smartphone.

If we do end up moving to include a hardware aspect of the project, our hardware aspect will consist of a single screen hooked up to a computing device (a screen with a raspberry pi, or a cheap android tablet, etc.) that will live in a singular location in the user's closet. The operating conditions of the closet will be easily manageable for any computing device we decide upon as the device will be stationary and in a dark, dry location.

1.4 Intended Users and Intended Uses

Our first primary intended user is those who travel frequently for business. These individuals may be traveling to different countries in the duration of just one or two days. Our application is intended to allow these users not to worry about checking the weather, but rather just have it done for them. Our application will also take their schedule into account as well, so that it will give smart predictions based on the weather at that time of day and also where they will be.

Another primary user are those who live in an area where there is easy access to public transportation. These individuals may have a longer commute and may not be able to go home during the day. Therefore, our application needs to be able to give recommendations for an entire day based on the weather.

Our secondary user is those who would like to make their day more efficient. These individuals would like to have an outfit prediction based on the weather, and may not have a busy schedule. Picking out an outfit takes time, and our application will make their day more efficient.

1.5 Assumptions and Limitations

Assumptions:

- System will run on a single-user basis, so should be able to run on many concurrent systems.
- That the user will be able to have internet connection a majority of their time, and always when they are getting ready for the day.

Limitations:

- The system will need to be connected to the internet in order function.
- Users will need a smartphone in order to run the end product.
- The integrated development environment won't be available to us on school computers so work will take place off campus

1.6 Expected End Product and Other Deliverables

The only currently planned deliverable is the android, iOS, and web application. The application will have the ability to do all available functions across all 3 versions of the application. The client will be able to first input all of their owned pieces of clothing and catalog them in their digital "closet." The user will then set a daily time they would like to be initially told an outfit for the day (this will normally be the same time as their alarm). The user can also request an outfit update at any point of the day inside the application. The application will provide an outfit based on the current and possible weather for the day at the specified time every day. The application will suggest clothing that the user already owns and has catalogued in the application. The user will be able to see alternative options for the clothing items suggested in case the top suggestion is dirty or not currently usable.The application will also periodically check in throughout the day to see if the weather predictions have changed and suggest items like snow boots, raincoats, and umbrellas. The application will notify the user if any updates that require a sudden change in attire do occur.

The application will reach the alpha stage by december 2018. The alpha stage will consist of an initial product we will use to retrieve feedback from testers and potential users. The beta stage will be when we have mostly finalized the product and are looking for improvements will be reached by May 2019, and the finalized application will be released three weeks before the end of the Spring 2019 semester.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The goal of this task is to produce a context aware application that will connect to a weather API and the user's personal calendar to produce a clothing recommendation from the user's wardrobe based on the data retrieved from the two sources.

2.2 Functional Requirements

The functional requirements of this project include the ability for the application written in ReactJS Native to communicate with an external weather API and the user's calendar. Once the data is retrieved from these sources, the application will contact the database that contains elements representing the user's wardrobe, and based on what clothing is available to wear at the time, the weather, and the events of the user's day, the application will display a list of all the different pieces of clothing that would be appropriate to wear for that day.

- Log in and out
- View available clothing
- Select clothing to be worn that day
- Add/Remove clothing from wardrobe
- Refresh wardrobe when wardrobe is low or out

2.3 Constraints Considerations

Non-Functional Requirements

- **Scalability** - The database of the application must be scalable to ensure many users will be able to access the application and their wardrobes.
- **Availability** - The application needs to be available 24/7 outside of maintenance for when the users require context updates for their wardrobe.
- **Reliability** - Application must be able to recover loss of database and preserve user accounts and wardrobes.
- **Maintainability-** Database must be maintained to ensure proper updates via the weather and calendar.
- **Security** - Database must be secure to protect information about the users and their clothing.
- **Data Integrity** - The clothing items stored in the database should only be able to be modified by the users or an administrator.
- **Usability** - Will be accessible by all users and administrators, so all users can receive updates for clothing recommendations. Users will connect to the application via a mobile device and recommended clothing will be displayed as the weather and calendar are checked.
- **Performance -** The application should generate a recommendation based off the weather and calendar events in less than 3 seconds.

Standards

All development will take place remotely and outside of the lab environments, and we do not think there will be any practices within the development of this project that would be deemed unethical, but we will still adhere to the standards and protocols brought forth by IEEE/ABET. For our requests to firebase we will be using the AJAX standard that the Firebase sdk uses in React. All data we transfer between our database and our front end will be sent using JSON objects, and will follow the same JSON standards on both the front and back end. If we have any concerns about this, we will be sure to consult our faculty advisor about possible breaches in the standards and protocols.

2.4 Previous Work And Literature

There is a web application similar to this project called “Daily Dress Me”, and this application looks at the weekly forecast and makes clothing recommendations on the temperature and overall weather. The difference between “Daily Dress Me” and our application is that our project is going to be a mobile application rather than a web application. Another addition to our project that will separate us from “Daily Dress Me” is our project will also be making recommendations based on the user’s calendar, and the clothing recommended will be pieces contained in the user's personal wardrobe rather than just a general outfit like provided on "Daily Dress Me".

Another project that has a similar description to ours is a previous is an application designed by our client. This is an application that creates recommendations for public transportation for users based on the weather. This will be similar to ours in the sense that we will both be generating the best solutions for the user based off the weather situations, but they differ in the aspect that this application deals with the most optimal routes to take with public transportation based on the weather, while our project deals with preparing the user for the weather at the beginning of the day.

The pros of using "Daily Dress Me" over this project would be that this web service recommends entire outfits over a single piece of clothing at a time. This works because it gives a general outfit to the user that could be matched to any normal person's wardrobe. A second pro would be a proper outfit would be recommended, when in this application, just individual pieces of clothing would be recommended, and the user could wear an outfit that may be suited for the weather but not exactly stylish. The con of using this web service over our project is the fact that the clothing recommended in this project are the user's personal clothes rather than a general outfit, so the users will know exactly what clothing would be good to wear in their wardrobe.

Source: http://dailydressme.com

2.5 Proposed Design

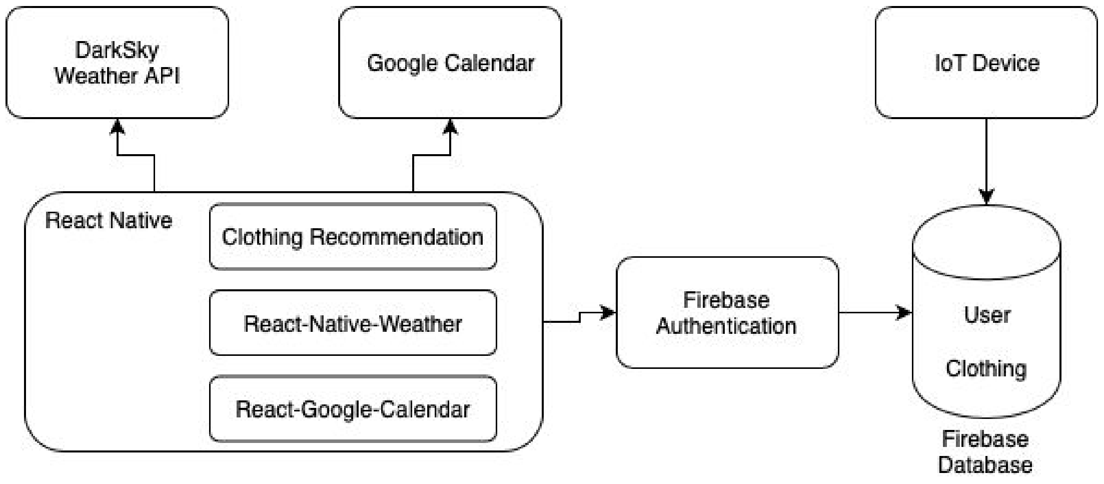


Figure 1: *Software Architecture*

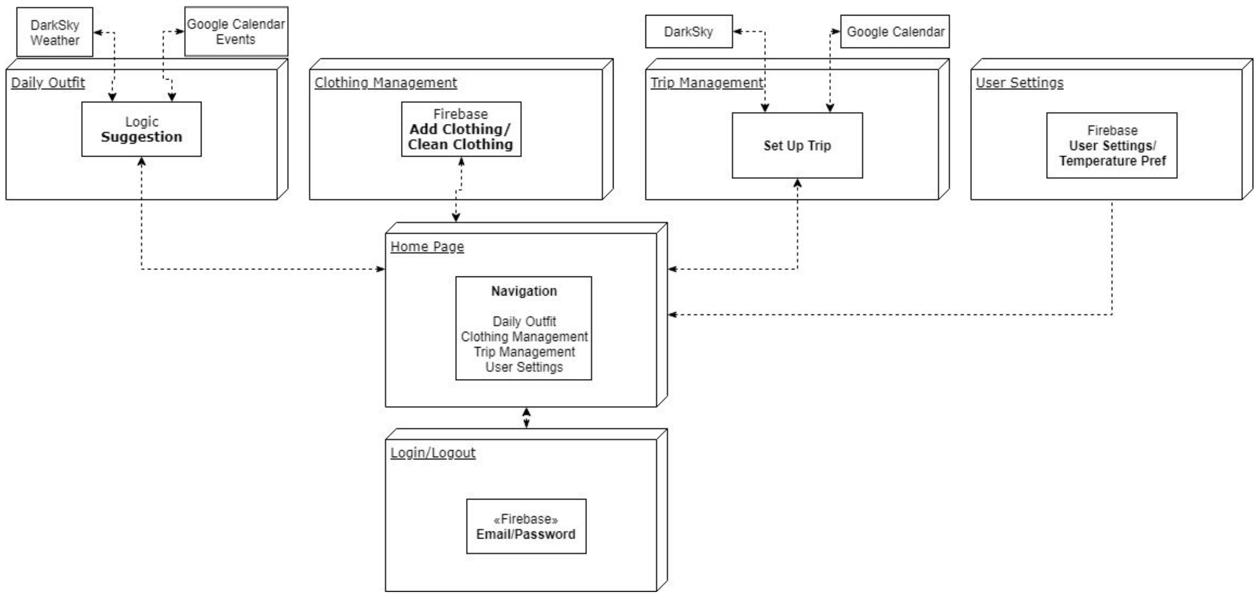


Figure 2: *System Block Diagram*

This project aims at providing a comprehensive solution for a clothes recommendation system. The main idea is to couple data from multiple sources and integrate it in a manner that will enable the users to select outfit and/or plan which clothes to pack for an expected trip. As a motivational example-scenario: upon detecting that the user has woken up (e.g., Fitbit), a trigger is fired that connects to the DarkSky API. Given the weather prediction and the tasks and places that the user has entered in the calendar for that particular day, an app will provide a recommendation for selecting the items from the closet to be worn/carried for the day. If the proposed outfit is in the wrong temperature range for the user, the user can look at a suggestion in a higher or lower temperature. They can permanently set this as an option on their account if they are a person who tends to be cold or warm more often than the average we find.

2.6 Technology Considerations

The strengths of the technology we will be using are mostly attributed to the connectivity of the application and its strength in terms of available APIs for input sources. Specifically, we will be using APIs for both weather and calendar inputs to the application backend. Another strength is to implement a hardware display connected to the application for easier interactions. We have thought of a few possible designs for both the mobile part and the hardware part. The in-closet hardware we have considered are: using LEDs on hangers to display which clothes should be worn on a certain day, making a “smart mirror” which is a mirror with a screen behind it that will display live updates from our application, and using a stationary tablet that would be next to the clothing and receive constant updates and report findings.

Some foreseen weaknesses and trade-offs of the technology is the complications from adding connected devices, such as an mobile application linked to a clothing visual representation in one’s closet. Implementing a strong IOT network will be necessary for this project’s success.

Putting LEDs on the hangers has a few notable weaknesses. The main weakness would be the user. We would have to trust the use to put the same items of clothing back on the same hangers. Not many users would be willing to take even more time when doing laundry in order to ensure the success of an application like this so we ruled this option out.

The smart mirror has a few strengths and weaknesses. Everyone has a mirror in their closet or bathroom so this would just be an augment of a pre-existing household item. If the mirror were to be placed in a bathroom, our electronics would have to be waterproof in order to remain safe. This was a big concern of ours, as well as the fact that the biggest downside would be the time making the electronics would take away from the necessary development time we have in order to make the application work to the best of our ability.

We decided to go with the stationary tablet option. The biggest strength is that we are using pre-existing hardware and can focus on the software part of our project. The application can have a “tablet mode” and a smartphone mode. This would allow us to release only one version of the application but still have it be useful for both the stationary in-closet version and the on-the-go mobile phone version of the application. All of our data will be synced over any devices the user connects to their account and we will not have to worry about there being a difference between the clothing attached to the hardware and the clothing in their database.

2.7 Task Approach

We will be using the Agile methodology for our team. Every two weeks we will evaluate what is the most blocking part of the project to work on at that time and assign programming stories to each team member. Each story will have one name attached to it but multiple people are expected to pair program on each task. The pair programming will ensure that no one member will be solely responsible for a single part of the project and we will all learn as much as we can about the entire technology stack our project uses.

The first screen seen upon entry to the app is the login page, which is used for security and account management. This will work with Google's login service, and is paired with our Firebase backend. The login page will load whenever a user has not previously logged in, as well as whenever their firebase token expires. If the user is currently logged in with a token validated by Firebase, then the home screen will load and they will be able to use all of the functions of the application.

Upon loading the application we will poll the device for the user's current location and save that in the global state for the application. We will need to receive weather and schedule information for the user. Both of these will be taken care of with respective API calls, with DarkSky as the weather API and Google's Calendar API for schedule details. The DarkSky request will use the user's current location to find the current weather and the predicted weather for the day. We will be perform ajax requests to each of the APIs and getting back the JSON objects that they return.

The main focus of the app is the suggestion for what clothing to wear. We will parse the JSON objects for the information we are looking for and use that information to complete the clothing recommendation algorithms. We will take this information and check against what clothing is still in the database as being clean. We will need to write an algorithm that takes user temperature preferences as well as the information returned in those JSON objects and compares it to the temperature ratings of clothes in the user's virtual closet.

The secondary focus of the application is to provide trip planning for extended trips. For this we will use the calendar information to determine where the user will be and look up the predicted weather in that area during the time they will be there. We will then go through the same process we use for daily recommendations marking clothes as used for every day that we provide an outfit for.

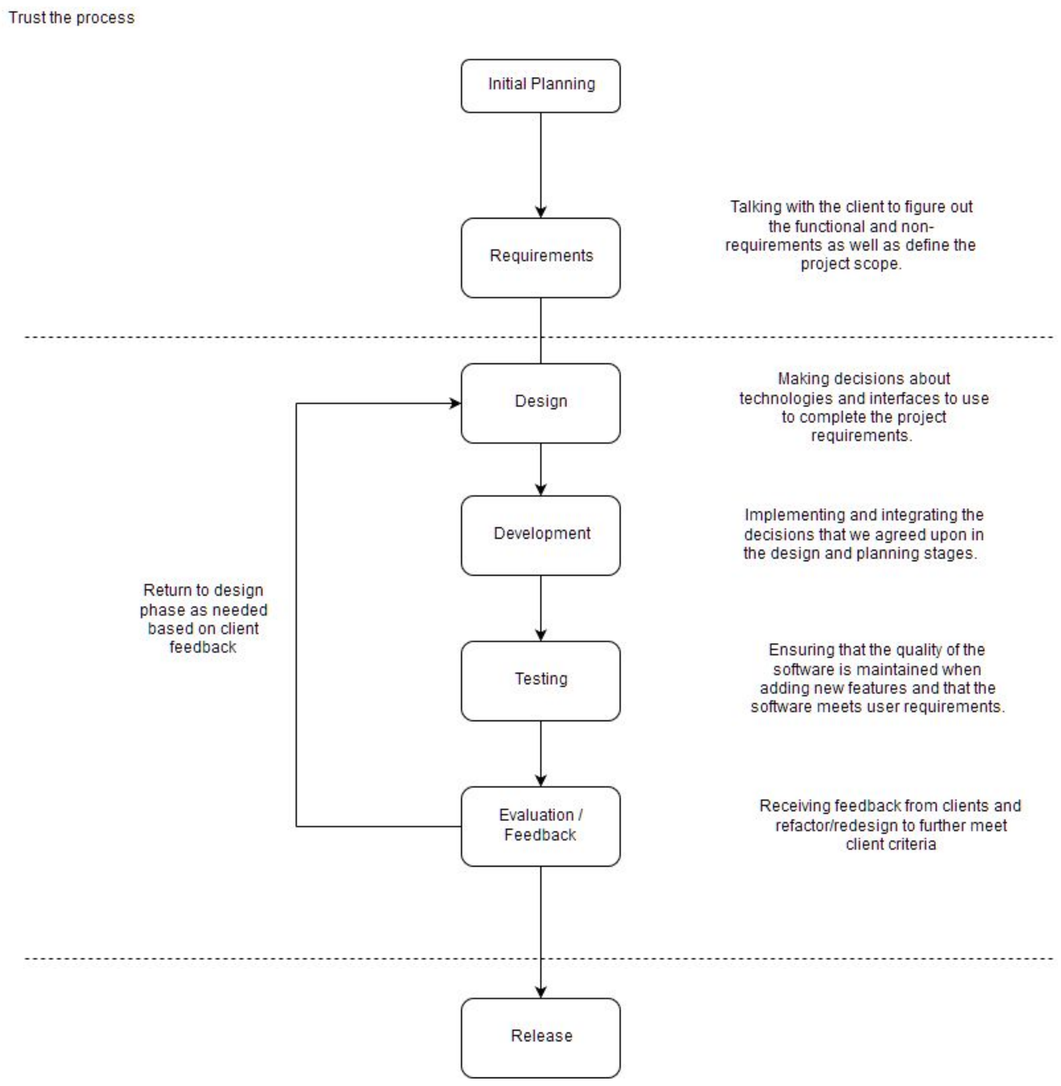

Trust the process
Initial Planning
Requirements
Talking with the client to figure out the functional and non-requirements as well as define the project scope.
Design
Making decisions about technologies and interfaces to use to complete the project requirements.
Development
Implementing and integrating the decisions that we agreed upon in the design and planning stages.
Return to design phase as needed based on client feedback
Testing
Ensuring that the quality of the software is maintained when adding new features and that the software meets user requirements.
Evaluation / Feedback
Receiving feedback from clients and refactor/redesign to further meet client criteria
Release

Figure 3: Task Approach Breakdown

2.8 Possible Risks And Risk Management

One of the areas of knowledge we do not currently understand and will be learning as a group is the firebase API service. The initial cost of the firebase license will slow us down a little bit at first, but once we have the license the biggest risk will be everyone learning a new API framework. Only two of us have ever written ReactJS code before and we will have to work as a group to help the others catch up to our skill level and help us all become better ReactJS programmers.

2.9 Project Proposed Milestones and Evaluation Criteria

Phase 1

- Planning

    - Confirm project scope
        - Implementation Plan: Once the client has confirmed our understanding of the scope it is finished.
    - Architecture Design / Tech Stack Decision
        - Implementation Plan: The team completed a session for designing the architecture design and documentation for tech stack research. The stack was then decided based on discussion and research.

Evaluation Criteria: Once documentation is completed for the tech stack decision and the client informs that our scope and requirements are correct, we will have completed this phase.

Phase 2

- UI Creation
    - UI / Wireframe created
        - **Implementation Plan**: The team created wireframes and UI mockups using draw.io.
    - Mobile UI created including splash screen
        - Home page with weather display is available
        - Data entry page where users can enter their clothing is available
        - Model to view clothing suggestions is available
            - **Implementation Plan:** The UI framework will be created by having a tab view with a splash screen implemented
- User Login and Storage
    - Registration/Login
        - User is able to log in through the google account login service
            - **Implementation Plan**: The team implemented Firebase authentication.
    - Store users securely
        - Keeps a copy of the users google account id in the state
        - Makes a global copy in the redux state of the user’s information
        - Uses that account id as a reference to any data that user requests of creates
            - **Implementation Plan**: The team implemented a Firebase database to store the users securely.
    - Profile Settings
        - User can update settings for their account that are replicated on the database relative to their account id
            - **Implementation Plan**: The team will implement functionality on the front end to change user preferences on the backend.

        - Data entry
            - The user is able to submit their clothes to the database under a table related to their account id
                - **Implementation Plan:** This will be done through a tab on the front end.
            - We are able to recall a list of all clothes related to any user's account id
                - **Implementation Plan:** This will be completed through a query that will run from the front end to get the payload of the clothes
- Clothing prediction
    - Consistently retrieve weather data from mobile app
        - **Implementation Plan:** Displayed on the homepage and useable in the API calls for the suggestion algorithm.
    - Display outfit based on weather data
        - The user is able to view suggested clothing items based on the current weather
        - The suggested clothing items are items out of the users digital wardrobe
            - **Implementation Plan:** This will be accomplished through an algorithm that will display the correct clothing based on the temperature.
    - Working alpha
        - Everything up until this point has been tested and is in working order so that we are able to demonstrate the usefulness of the app
- Integration of external devices
    - Efficient clothing inventory
        - We have an efficient way to enter clothing on the react native app across all devices
            - **Implementation Plan:** This will be accomplished with a tabview that the user will take a picture of their clothing and enter it to their virtual wardrobe.
- Final Touches
    - Push Notifications
        - The app is capable of pushing notifications to the user's phone whenever weather changes are detected
        - The application runs in the background sometimes in order to check for weather changes and calendar changes
        - The phone will push any newly required clothing items to the user's notification bar to ensure the user sees it in a timely manner
            - **Implementation Plan:** This will be done by writing native code or Xamarin code to implement push notifications.
    - Working beta
        - Everything up until this point has been tested and is in working order so that we are able to demonstrate the usefulness of the app

Evaluation Criteria: Once we have completed the features and they are stable in the production environment the feature will be done. Once we have completed all the features, the phase will be considered completed.

Phase 3

- Final Documentation
  - **Implementation Plan:** Write, edit, and review the documentation for the application.
- Releasing final product
  - **Implementation Plan:** Possibly release the application to the public with a developer license.

Evaluation Criteria: One the final documentation is accurate then we will release the final product. At that time, we will have completed all the phases.

2.10 Project Tracking Procedures

Trello

To track individual stories, we will use Trello. Trello is a project management software that lets you create a scrum board for tracking stories. Our team will assign stories to individuals and Trello will be used to track what the story is, how difficult the story is, who is working on it. This will keep our team organized and help track progress throughout the semester.

Agile Development

In our project, we will be using the agile development cycle. We will make use of pointing meetings as well as retro meetings to track our progress throughout the year. By using pointing meetings we can see how much we can accomplish during a given sprint. This will be done by calculating velocity for both the team and individuals. Also, we will make use of retrospective meetings to discuss what needs to be changed in order to make us more efficient and to evaluate our progress on the project.

Individual velocity

Individual velocity will be tracked by taking the average number of story points completed per sprint. This will be used to estimate progress in future sprints and also check to make sure the individual is on par with team progress.

Team velocity

Team velocity will be tracked the same way as individual velocity, but as a team. This will give us an accurate estimation what we can expect to complete in future sprints and if we are on schedule for completing milestones.

Reference to Gantt Chart

All of these techniques will be used with reference to our gantt chart. We can calculate these estimations and see if the estimation puts us on schedule with the gantt chart. This allows us to keep the big picture in mind and to make sure that we are making progress continually throughout the year. If we are not on schedule according to the gantt chart, we can make adjustments as needed.

Gitlab

Gitlab gives a detailed report for analytics and allows us to track how long it takes to go from receiving the story to committing code, to completing the story. This will give us an accurate estimation for completing stories as well as resolving issues. This will give great insight for the details once stories are assigned. The analytic “code” will be a great tool to use for a team as well.

2.11 Expected Results and Validation

Our end goal of this project is to have a fully function mobile/web application where users can plan their wardrobe for the day based on the weather and calendar events for the day or an extended trip. All of this information for the recommended clothing for the day will be presented to the user upon login, and an ability to search for a location where the application will display the clothing recommendations for the searched destination.

Our efforts are quite evenly split with some developers working connecting and integrating the APIs being used to obtain the weather and calendar information, and some working with the backend and database with authentication for the users to access the wardrobe that they upload to the application.

Testing for the project will be done constantly because of the convenience and ease of running React Native code. Scanning a QR code on a mobile device with the application Expo will load the project code into the phone, where all additions and refactorings will be evaluated. Tests that will have to be ran will be the functionality of the GUI and the connections to the database and external APIs. The GUI will have to tested to ensure that all buttons, menus, and displays are fully responsive and functional. Then the connections will be tested by ensure the data is being properly fetched from the APIs and sent to the database to report the proper clothing based on the day. The testing of the app needs to be done regularly in order to ensure that it functions correctly throughout development. In particular, the constant testing will ensure that no functionality in the app is lost in successive iterations.

2.12 Test Plan

For each stage of development, we will test new implemented features, as well as performing regression testing in order to ensure that the overall quality of the project stays consistent. This will require us to perform the following tests often, in addition to other tests as needed.
Tentative Test Cases:

Application collects weather data.

Test Case:

For this FR, we want ensure that the application correctly retrieves information from the weather reporting service API, DarkSky.

Test Steps:

a. Collect weather data from the DarkSky API.
b. Verify that the weather data in the app matches the 3 popular weather forecasts within 2 degrees.

Success:

Success in this test is measured in accuracy of reported weather compared to manually collected data. Weather responses matching 99.99% of the time within 2 degrees of three popular weather forecasts* are considered a success, matching the approximate accuracy of weather predictions.

**3 popular forecasts are weather.io, The Weather Channel, The National Weather Service*

Reasonable recommendations are given.

Test Case:

For this FR, we want to ensure that the system can give the needed recommendations and that the recommendations are reasonable and correct. The sample wardrobe will include a set of clothing for warm weather and a set for cold weather, and provided a cold or warm weather pattern.

Test Steps:

a. Give program sample wardrobe.
b. Review returned clothing suggestions compared to weather.

Success:

In a wardrobe of warm clothing and cold clothing, any suggestion including the off kind of clothes is a failure. Otherwise, it’s successful, with all returned clothing suggestions being appropriate to the weather conditions. Due to the polarized nature of this test case, suggestions should be accurate 100% of the time, or else some logic is incorrect.

User data is saved correctly and retrievable.

Test Case:

For this FR, we want to make sure that user information is correctly stored in the database and retrieved from the database. A mock user account will have its clothing edited, saved, and re-accessed.

Test Steps:

a. Upload user information.
b. Load and edit user information through the application.
c. Check that the information was correctly saved.

Success:

A successful test here occurs when data is saved as intended. A new shirt, pants, socks, and underwear being entered and remaining altered after re-accessing is a success. Trivial edits should be successful 99% of the time, leaving room for connection errors.

Clothing is correctly categorized and displayed.

Test Case:

For this FR, we want to ensure that when a user inputs clothing into the system, it is put into the correct place in the databases and can be accessed again.

Test Steps:

a. Input clothing items through the application.
b. Check the databases to ensure that the items were categorized correctly.

Success:

For each clothing option entered, checking that the database has the new clothing under the appropriate location will confirm a success or failure. Again, success should occur 99% of the time.

Application runs on Android and iOS.

Test Case:

For this FR, we want to ensure that the application runs on both Android and iOS at comparable qualities.

Test Steps:

a. Start the application on an Android and iOS device.
b. Run all previous tests for both systems.

Success:

Success for this case would be represented by no fails in the previous test cases, when replicated on both systems.

Performance

The user should receive a recommendation based on the weather in under 3 seconds.

Test Case:

The user will logout and login making sure that nothing will be saved in the cache, and then request a recommendation for their clothing. This should render on their phone in under 3 seconds.

Success:

Success in this test case is determined by the time between the app being opened to the responsible page, and the time taken for a suggestion to be displayed on the screen. As long as the time between request and completion of logic is under 3 seconds, it’s a success.

The user should be able to be authenticated in less than 1 seconds.

Test Case:

When a user has logged out of the application or is creating a new account, the authentication service should authenticate them in less than 1 second and be displayed in the database.

Success:

If authentication response time is 1 second or less, it's a success.

Security

Users will not have access to administrative functions

Test Case:

Create a new user account within the database and test whether or not their privileges are escalated enough to make any malicious changes within the application and the database.

Test Steps:

a. Login as superuser
b. Ensure that superuser receives all admin functions
c. Logout
d. Log back in as regular user
e. Ensure admin functions are no longer visible

Success:

Success criteria is measured by a regular user account sending administrative requests, and if they fail due to privilege reasons, the test is a success.

Usability

Efficiency of New Clothes

Test Case:

When the user is adding new items of clothes, they should be able to do this in an efficient manner. The user should be able to select what temperature range this should be worn at and the category of the article of clothing under 5 seconds based on user experience with technology. The user in the future may be able to add their clothing from a csv file making the web interface extremely efficient.

Success

Success in this area entails clothing items being added in less than 5 seconds in the plans current state. There may be updates to this criteria later based on future design plan changes.

Performance Tracking

Test Case:

Any process that takes more than 2 seconds should display a window saying "Please Wait" this allows the user to not overload the application, but also understand that the app is working and that it is not user error. This case should force a process to take extra time, upwards of two seconds.

Success:

If the app displays a "please wait" notice for a process taking longer than 2 seconds, the test is a success.

Compatibility

External APIs

Test Case:

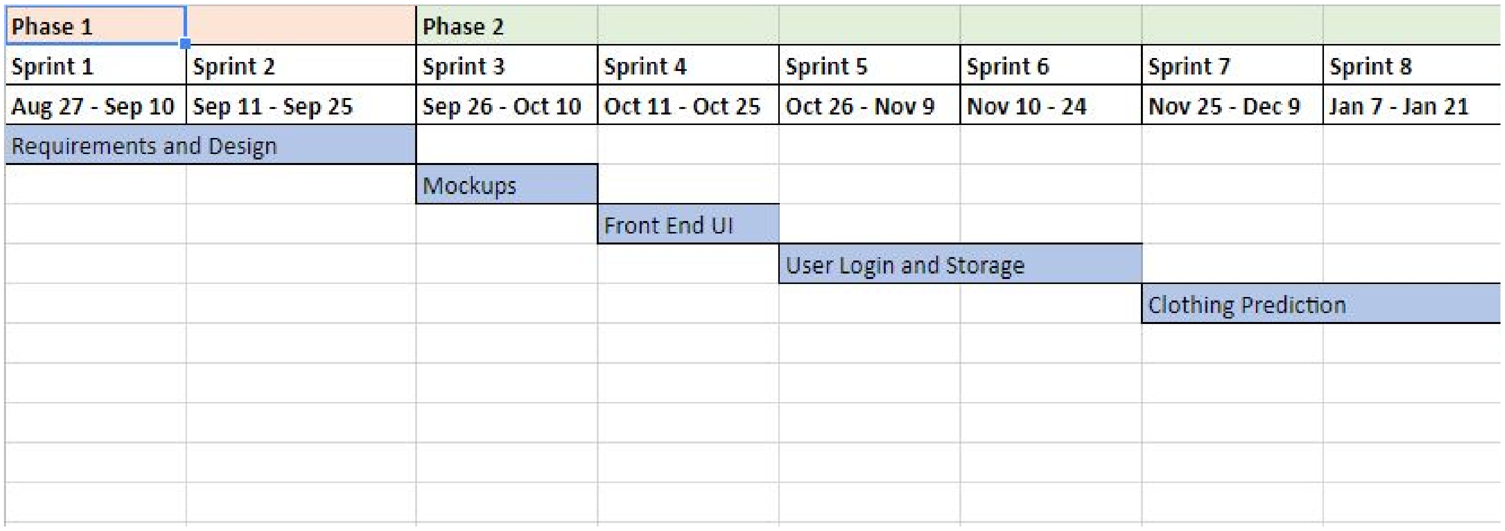
Test various ways the APIs are used for the application and how they can be plugged into the environment. This way the application can get the best use out of these APIs.

Success:

A success for this test is a response to API calls containing the requested information.

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline



Phase 1		Phase 2					
Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Aug 27 - Sep 10	Sep 11 - Sep 25	Sep 26 - Oct 10	Oct 11 - Oct 25	Oct 26 - Nov 9	Nov 10 - 24	Nov 25 - Dec 9	Jan 7 - Jan 21
Requirements and Design							
		Mockups					
			Front End UI				
				User Login and Storage			
						Clothing Prediction	

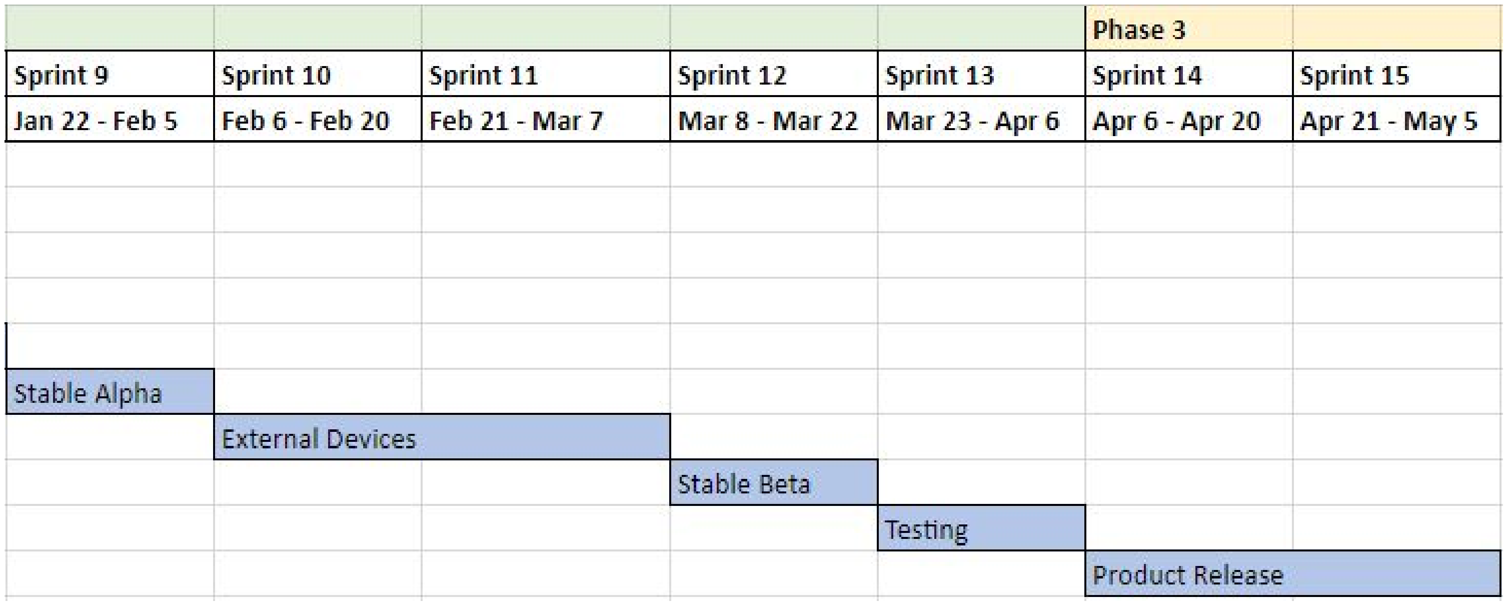


Figure 5: Gantt Chart

As seen above, our team feels that this is the most appropriate timeline given our time constraints. Each sprint represents a two week period from the start of the first semester to the end of the second semester. Roughly each feature should take around two sprints, which allows us to make sure that we have completed the feature given the acceptance criteria and also time to test it in the production environment.

Phase Number	Description	Estimated Time	Notes
1	This phase will be the requirements and design phase. This phase will include determining the scope of the project as well as completing the architectural design of the project. Deliverables: ● Scope of project ● Architecture Diagram of system ● Technology stack choices and reports	2 Sprints	
2	This phase will be the part where there is actual development and will take up the majority of the project. This will include development, wireframes, and further design decisions. Deliverables: ● Wireframes/Mockups ● Alpha version of application ● Documentation regarding design choices ● Beta version of application	11 Sprints	There will be testing done here as well while doing development.
3	This phase will be the final transition from beta to the released product. This is where testing will be finalized to ensure a proper release of the product. This will also be the preparation stage to finalize our project as a whole for final presentations. We will also add additional functionality in this phase once the team ensures the product is stable. Deliverables: ● Testing reports ● Final Product and Presentation	4 Sprints	

Table 6: Timeline Description

As shown in our phase breakdown, we have three different stages. This allows us to see a larger goal when completing the subtasks and what scope we should focus on during that sprint. This allows us to keep the goal in mind and not get sidetracked by non-functional requirements. We felt like the breakdown of initial planning, development, and finalization was the most logical choice because it lines up with the class schedule.

3.2 Feasibility Assessment

Our project, as it stands, is populated with tasks ranging from trivial to complex. Receiving weather information has been solved using the DarkSky API, and will be a feasible solution even after the app is released, due to DarkSky's scaling pricing. Other hardware implementations we may add include: smart mirror compatibility, LED hangers, and an always on tablet. These hardware implementations are within the scope of our stretch goals. However, despite that fact, we are quite sure we'll be capable of creating a fully functional, presentable app with our core functionality by the end of next semester.

We are using React Native to produce our front end application because it will reduce some of the complexity for cross-platform mobile development. Some of that complexity will return when we need to perform hardware specific tasks such as push notifications and notification sounds. We foresee an increased completion time in our sprints when we reach that stage of the project. We have planned for a little leeway farther down the line in case we reach any breaking bugs with our development when it comes to native code that differs between iOS and Android.

We see a small cost estimate for this project. Until our app is released, our calls to the DarkSky API will be well within the free use limits. We will require a developers license for the Apple app store in order to post our application in the public domain for iPhone users. We may need to rent an android tablet in order to test out the tablet mode on our application for users to leave on display in their closet. In total, the cost for this project should be well within affordable amounts.

Foreseen Challenges

The major issues we foresee having for this project:

- Integrate multiple third party services
- Create reasonable multi-day travel functionality
- Research and implement smart mirror
- Create system for LED hangers
- Develop realistic method of entering large amounts of clothing
- Hardware specific programming (Native Code and Xamarin components)

3.3 Personnel Effort Requirements

Task	**Reference**	**Estimated development time (hours)**	**Estimated documentation and testing time (hours)**	**Total Time (hours)**
Create back end API structure	Create the back end API structure for data to be collected from our database	40	4	44

	table			
Create database table for storing user clothes	Create the relational database tables for user clothes lookups	25	5	30
Create database table for storing logins and passwords	Create the relational database tables for username and password hash lookups	30	4	34
Create react front end entry landing page and login page	We will need a login page to determine what user is in the application and what user we will need to look up the clothes for	25	5	30
Create react front end page for displaying what clothes to wear	We will need a front end page that will display the clothing choices made	20	2	22
Create the API lookup for the weather API that will get the data we will test against.	We will need to retrieve the weather data whenever the phone performs a lookup to get the latest weather data from our weather service	20	5	25
Create an “alarm” function that will allow the user to set a daily time they want to be notified of an outfit.	This will be the main chunk of the application what we foresee as the most used function.	30	5	35
Create the ability to send notifications to devices	This will be necessary for any weather updates and for the daily clothes alarm	20	5	25
Create a function to periodically check for weather updates throughout the day and notify the user	This will notify the user any time the weather has significantly changed to necessitate the need for an outfit or attire change such as an umbrella or snow boots.	40	3	40
Create a way to interface with the user’s google calendar to get their daily events	This will be to select clothes for the user based on a tiered system of clothes for the day. Business > business casual > casual. The clothing choices for the day will be clothes in the category of the highest tier of clothing for the day based on tags in calendar events.	80	10	90
Create a database table for storing clothing items	This will store the type of clothing, the weather for the clothing, and casualness of the clothing (business, business casual, casual).	20	5	25
Create lookup queries for the database for each of the table	The API queries will be what we call from the react side in order to get the information we need from the query	50	10	60

Table 7: Personnel Effort Breakdown

3.4 Other Resource Requirements

We have both Android and iOS devices to test the application on. Google's Firebase will serve as our server backend and hosts the database for all of our user-specific data. It is assumed that the user will have both an internet connection and a closet with enough variety of clothing needed to fulfill all of the weather types for the area the user lives in. If we include some of our optional project extensions we may require more hardware, such as:

- Smart Mirror
  - Raspberry pi
  - Mirror
  - LCD screen
- Android tablet
- LED Hangers
  - Hangers
  - Programmable LEDs
  - Raspberry Pi for LED control

3.5 Financial Requirements

The costs of this application will depend on the size of the active user base. The free version of firebase and of the dark sky API we are using have a query limit per day. After our application goes over 1000 queries in a single day, DarkSky has a small financial cost of $0.0001/call. We are using the free license for firebase currently. The free version allows us to have up to 100 concurrent users connected to the database. We can store up to 1 gigabyte of data for free and can access up to 10 gigabytes of data per month. If we go over these amounts the cost will increase to $25/month for the next tier of usage. We will also require a Apple developer license in order to upload our application to the Apple app store if we decide to take the application to market. An Apple developers license costs $99 annually.

4 Closure Materials

4.1 Conclusion

We recognize that it can be hard to find a proper outfit for the day when you are clouded by the fog of waking up early in the morning. That's why we are bringing the best of technology and possibly IOT to bring our users a better way to find an outfit for the day. We plan to implement an application platform for mobile devices, and move towards a physical representation which will point our users to the available clothing options.

Our context-aware clothing recommendation system will benefit our users with smart suggestions on what to wear; rain or shine, hot or cold, formal or casual. This will be done by interfacing with the Google location system and an external weather database to get up-to-date weather information in order to give accurate recommendations. The application will also interface with the user's Google Calendar, allowing the user to define events and trips in advance.

4.2 References

Hussain, Muhammed Mas-ud, et al. “Incorporating Weather Updates for Public Transportation Users of Recommendation Systems - IEEE Conference Publication.” *An Introduction to Biometric Recognition - IEEE Journals & Magazine*, ieeexplore.ieee.org/document/7517814.

Npm, (2018). React-google-calendar-api. [online] Available at: https://www.npmjs.com/package/react-google-calendar-api [Accessed 12 Oct. 2018].

Npm. (2018). *react-native-weather*. [online] Available at: https://www.npmjs.com/package/react-native-weather [Accessed 12 Oct. 2018].

Google, (2018). Firebase Database [online] Available at: https://firebase.google.com/ [Accessed 20 Oct. 2018].

Dark Sky API, (2018). [online] Available at: https://darksky.net/dev [Accessed 20 Oct. 2018].

Getting Started – React. (2018). [online] Available at: https://reactjs.org/docs/getting-started.html [Accessed 20 Oct. 2018]

Pricing Plans - Firebase. [online] Available at: https://firebase.google.com/pricing/ [Accessed 20 Oct 2018]

4.3 Appendices

Iowa State University and Senior Design Team 34 do not discriminate on the basis of genetic information, pregnancy, physical or mental disability, race, ethnicity, sex, color, religion, national origin, age, marital status, sexual orientation, gender identity, or status as a U.S Veteran.