Context-Aware Clothing Recommendations

DESIGN DOCUMENT

Team 34

Client/Advisers
Goce Trajcevski

Team Members/Roles
Ethan Wieczorek - Lead Developer
Nickolaus Eaton - Product Manager
Will Parr - Communications/Meeting Facilitator
Nicus Hicks - Head of Report Development
Christian Ehlen - Lead Quality Assurance Engineer
Tyler Witte - Software Project Manager

Team Email
sdmay19-34@iastate.edu

Team Website

https://sdmay19-34.sd.ece.iastate.edu/

Table of Contents

List of Figures:

1 Introduction

In this section, we describe an overview the project.

1.1 Acknowledgement

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

1.2 Problem and Project Statement

It's common to see people wearing clothes that are inappropriate for the weather, because they did not check the weather forecast. The problem is straightforward, it's easy to get dressed while unaware of what you should actually be putting on or packing. Our goal is to implement a system that suggests clothing according to the weather and events that you have scheduled for the day.

This project aims at providing a comprehensive solution for a clothing recommendation system. The main idea is to collect 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 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.

Extending this example would correspond to two kinds of scenarios: (1) a user needs to plan a business trip in multiple places and different geo/climate zones and providing a recommendation for which clothes to pack; (2) an abrupt change in certain parameters (e.g., weather; change of meeting place) necessitates obtaining another clothing item (e.g., a sweater or an umbrella) -- in which case the system should recommend the nearest store or a store requiring the smallest deviation from a planned route.

1.3 Operational 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 or around 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 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 or tablet 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 Deliverables

The current 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 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 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. Specifications and Analysis

Now, we discuss the proposed design and an analysis of it.

2.1 Proposed Design

After discussion and research, we decided to use React Native for our implementation of this project. We came to the decision on this framework because it allows us to develop for both Android and iOS simultaneously. For our external APIs, we chose DarkSky for weather data because it is well supported and has libraries that streamlines its implementation and use within React Native. For the project database, we chose to go with Google's Firebase Real-Time Database because it is lightweight and is compatible with React Native. Firebase services also offer Google Authentication, so instead of us handling all the accounts that would need to be created for the use of the app, we can use this authentication service and have the users sign into the app using their own google account. We are taking an agile development approach to completing this problem by assigning smaller subproblems to each member of our team and completing the entire project in parts.

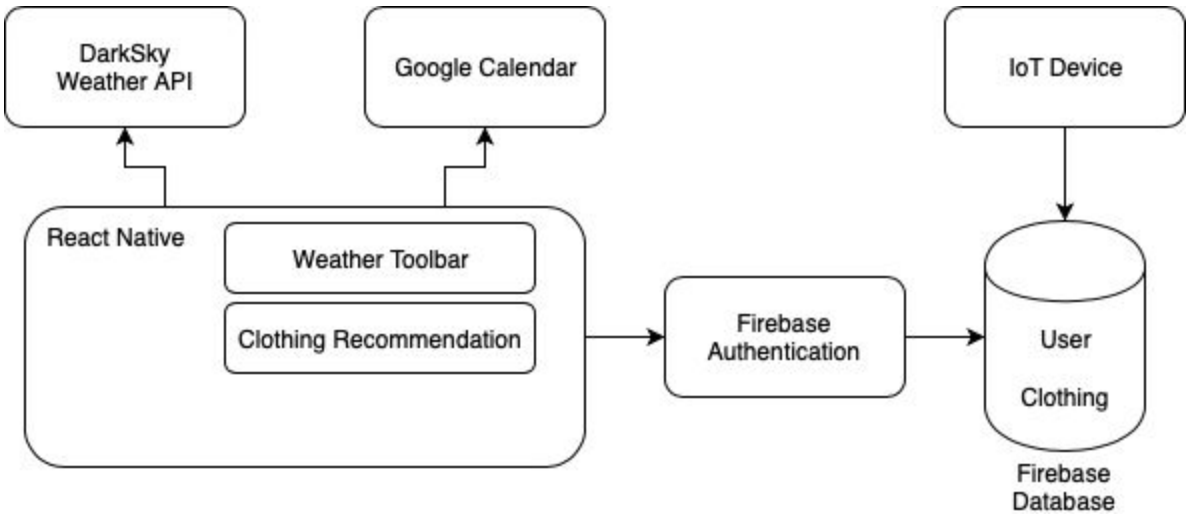


Figure 1: System Block Diagram Depicting Proposed Design

To meet the specifications of the client, we have determined both functional and non-functional requirements. The functional requirements are the minimum requirements that we need to complete in order to have a functioning app that meet the specifications of the client.

Functional Requirements:

- Ability for the user to login and logout of the app using their Google account
- The user should be able to see available clothing that is not dirty
- The app should display the clothing to be worn that day when the user requests to see a recommendation

- The user should be able to add and remove clothing items to their inventory for the app to use for recommendations
- Once the user marks items as clean, the wardrobe should refresh and update possible clothing options

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 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.

2.2 Design Analysis

So far we have determined the login page, as well as the weather api and display we will be using.

Login Page: For the login page we had a few different designs we were deciding between. Option 1 was a basic username/password login page. Option 2 was an email/password login page. Option 3 was a firebase enabled login through google.

We chose the firebase enable google login because of a few key strengths. The login authentication is handled entirely by google and we don't need to authenticate usernames or emails with hashed passwords. This adds a layer of security because our database doesn't actually store any password hashes so there's no security vulnerability if our database is compromised.

Weather API: We are using the Dark Sky weather API. The dark sky api provides us with an easy call to make for all of the weather information necessary for daily use. More importantly, our application is allowed to make 1000 requests to dark sky for free every day. Beyond the base 1000 requests the price per request is considerably low. Out of all of the APIs we researched this is the best price per request we found for our use. There are no limitations to dark sky that we have found.

3 Testing and Implementation

Since the app we are creating will ideally be used by a large number of users at the same time, it's important that we ensure the resulting app is tested for successful user interface implementation and simultaneous activity.

For building and testing our project, we are using a set of tools, libraries, and services called Expo, which allow us to view changes to our app each time it is saved. These live changes allow for quick refactorization for visual components.

Entering the app brings the user to a login page, which also allows the user to create a new account. Once they have entered valid account information, they are brought to a home screen, with navigation to each of the individual major aspects of the app. Selecting daily outfit displays the suggested clothing for the day, taking information provided by API calls with DarkSky and Google Calendar. Selecting clothing management instead allows the user to add, remove, and tell the system that clothing has been cleaned. These changes require interaction with the Firebase, specifically dealing with the clothing associated with them. Choosing trip management will bring the user to a screen prompting the user for information about a trip, specifically requiring the dates and locations for the trip. This will allow the system to suggest clothing for the trip itself ahead of time. Lastly, selecting user settings brings up a screen where the user can set preferences relating to temperature preferences pertaining to clothing.

3.1 Interface Specifications

Our system runs primarily on smartphones and tablets. In every case, it's important that the application has a familiar, if not identical, appearance and interface. Additionally, we will port the app to a web app, to allow users to use a computer instead. This will be especially beneficial for adding multiple pieces of clothing.

3.2 Hardware and software

As of now, we are focused on creating the app to meet the functional requirements for the client. Software that we are using includes Fiddler to be able to debug web requests and how the requests interact with the database. This allows us to isolate the front end from HTTP requests for further modularization of development.

Hardware that we are using includes both Android and iOS devices so that we are not limited to testing virtually on emulators. We test on the real devices through the use of Expo, which builds the app live so that developers can see their changes in real time. Through the use of all of these tools we are developing an efficient way of testing and developing the app.

3.3 Functional Testing

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. Manually collect weather data.
b. Check weather data collected by app against manually collected data.

Success:

Success in this test is measured in accuracy of reported weather compared to manually collected data. If the app weather and current actual weather agree, it's successful. Weather responses matching 90% of the time are considered a success, matching the approximate accuracy of weather predictions.

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:

c. Give program sample wardrobe.
d. 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:

e. Upload user information.
f. Load and edit user information through the application.
g. 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. 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:

h. Input clothing items through the application.
i. 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:

j. Start the application on an Android and iOS device.
k. 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.

3.4 Non-Functional Testing

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 less than 1 second and be displayed in the database.

Success:

This case measures success in time between a request being created for authentication and getting a positive or negative response. As long as the time between request and response 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 in this case 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. This functionality should execute 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 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:

The app should display "please wait" when a process takes more than 2 seconds 100% of the time.

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:

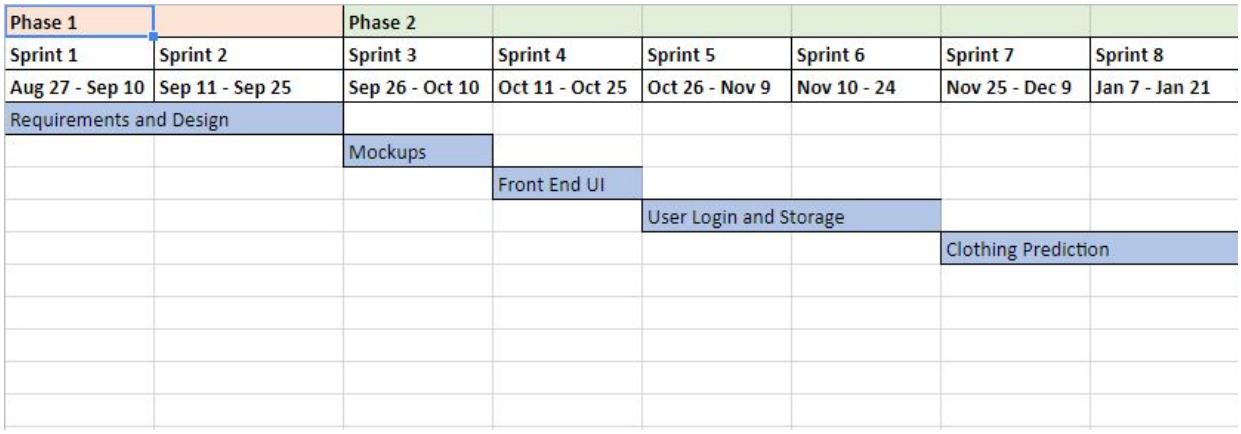
A success for each API is a response to API calls containing requested information- DarkSky giving weather feedback, for example.

3.5 Process

Before any actual development we made sure that our requirements and scope were correct by pitching our idea to our client. If the client was satisfied by the product and felt that it met all of their requirements, we could move on to development.

We first began development testing by testing the overall functionality of the mobile app and its interaction with the database. Once the mobile app is working and displays the placeholders for all of the screens without any major issues, we are moving to the testing of displaying the data. If the data is displayed correctly and tests well with users we will move to the web interface of the application. We plan on moving to porting the mobile app to having a web interface as well for large wardrobe inputs. This will need to correctly be stored in the database and reflect what the user has entered from the web interface. Once this is a stable feature, we will move on to adding external devices.

Moving on, we will likely be moving to adding IoT devices such as smart hangers for more accurate recommendations for the user that will take into account colors, style, and thickness. This will enable the user to have far more accurate predictions for their requested days.



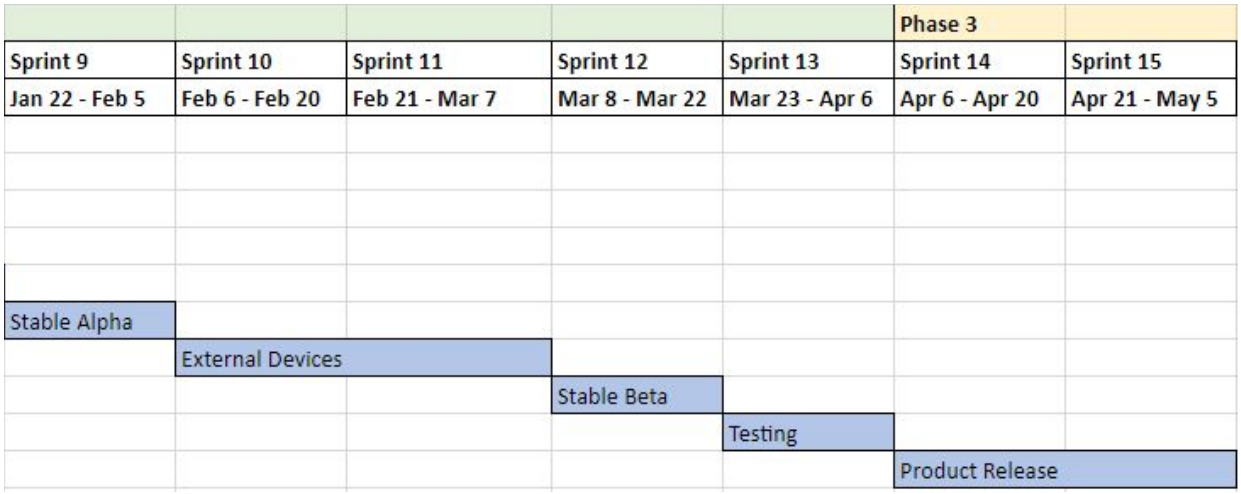


Figure 2: Gantt Chart for Project Development Timeline

As shown in the diagram above, we will continually be testing new features and receiving feedback from our client. Our testing process will take place throughout the entire development lifecycle allowing us to pivot if we need to refine requirements. By continually designing and testing, it allows us to take steps back and view where the project should be headed and not get tunnel vision on the development side of new features.

3.6 Results

There have been no system tests performed at this time, as basic implementation is still in process. However, we have successfully retrieved information in experiments. Some issues which have been encountered include; getting the app running under different firewall constraints on secure networks and implementing interaction between different services. In particular, we had issues connecting our React Native App to the Firebase Database we have set up. One of the major causes for this issue was our lack of experience with the two systems.

4 Closing Material

You're almost done reading. You got this, I believe in you!

4.1 Conclusion

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. The intent of the system is to provide a user with day to day updates on what to wear as well as a function to help a user pack for a trip. The system will intelligently select qualifying items the user should wear or pack based on the user's calendar and the upcoming weather.

Thus far, we have begun implementing a design for a universal-type react native application that is to be functional on both android and ios devices. Our implementation has reached the early app development stages and is has a functional react dashboard with a weather API call. Our goal is to complete additional calendar api calls and to complete a functional user interface so that we can begin to add clothing to the database.

Our best plan of action is to continue with the application development in sprints. If we successfully complete each development cycle, we expect to have a working prototype of the application within the next couple weeks.

4.2 References

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. (n.d.). Available at:
https://darksky.net/dev [Accessed 20 Oct. 2018].

Getting Started – React. (n.d.). Available at:
https://reactjs.org/docs/getting-started.html [Accessed 28 Nov. 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.