Fit of the Day

by

Liam Palmer, Tyler Blair, Daniel Le

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Liam Palmer	4/26/21
Liam Palmer	Date
Tyler Blair	4/26/21
Tyler Blair	Date
Daniel Le	4/26/21
Daniel Le	Date
Abdou Fall	4/26/21
Abdou Fall, Faculty Advisor	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services
April 2021

Contents

List of Illustrations

Abstract

As a team of indecisive people, our goal is to make choosing what to wear a little easier. Fit of the Day is an innovative application that gives you new ways to style your clothes when you are having trouble deciding for yourself. Using local weather data as well as your daily calendar events, Fit of the Day generates outfits tailored to fit your life. If you are feeling adventurous, you also have the option to generate entirely random outfits. We decided to build Fit of the Day because there was not a provider for a service like this. Fit of the Day not only helps you manage your closet but will also save you time by automating your outfit choices.

Introduction

Some of the most successful people in the U.S., Mark Zuckerberg, Steve Jobs and Barack Obama wear nearly identical outfits every day. While this appears strange to some, it is not without reason. Their daily uniform helps them streamline their decision making and reduce something called *decision fatigue.*

Problem Statement

Decision fatigue can be described as an inverse relationship between decision quality and quantity. As a person makes more consecutive decisions, their decision quality decreases. Some sources suggest that people make a staggering thirty-five thousand decisions every day (Hoomans, 2015). Furthermore, researchers from Columbia and Stanford University state "as the complexity of making choices rises, people tend to simplify their decision-making processes by relying on simple heuristic," which can decrease the relative quality of a decision (Iyengar and Lepper, 996). Everyone has room to improve and reduce the impact of decision fatigue on their lives.

On the other hand, some people might just want to find the perfect outfit but do not have the time or energy to choose one. Choosing an outfit every day can be a tedious process, especially considering other life events and the unpredictability of the weather. Other people may not have the desire to make these outfit decisions at all. So, in an age of IoT devices and automation, why not let a computer choose your outfit for you?

Solution

Fit of the Day takes the decision making out of daily clothing choices. It is a Web application that analyzes local weather data, personal calendar events, and a user's clothing to

automate outfit creation. Regardless of the day's occasion or weather, Fit of the Day automatically generates and recommends suitable outfits for the user to wear.

While there are many apps out there that let people organize their closets, not many will create outfits on their own. With the configurable options Fit of the Day provides, finding great outfits can become as simple as clicking a button. This creates less mental fatigue throughout day that allows users to be more productive in their day-to-day workflow.

Project Goals

The initial goals of the project were as follows:

- Streamline outfit creations into one app/Web site
- Help users that struggle to decide what to wear
- Track user laundry, as in which items are clean or dirty

These were later updated to include more specific goals to narrow the scope, including the following:

- Display local weather data as provided by OpenWeather and based on GPS location or a given ZIP Code
- Provide a calendar or calendar integration to track user events
- Generate outfits based on calendar events, weather data, and user favorites.

Overview

This report covers how the project was completed. The sections to follow include: Project Concept, Budget, User Profiles, Use Case Diagram, Testing, Timelines, and Problems Encountered.

Discussion

Project Concept

This concept for this project originated from one of the team member's personal struggles with decision making and outfit creation. This team member often wore similar or the same clothes in specific combinations that were simple and easy. He felt as though his outfits were stagnant yet was put off by the time and effort required to explore more possibilities. The basic idea was initialized off the following statement: A Web site that aggregates weather data, personal calendar events, and clothes to automatically generate an outfit for any day.

Design Objectives

The goals for the project were as follows:

1. Streamline outfit creation and generation into one application.
2. Help those who struggle to decide what to wear.
3. Track which items are clean and dirty and provide a reset for laundry days.
4. Display accurate local weather data based on GPS location or a provided ZIP Code.
5. Provide an integrated calendar to track and show user events.
6. Generate outfits based on calendar events, weather data, and user favorites.

Goals 1-3 were the initial goals the project was based on. They provide the general idea of what the projects purpose should be. Goals 4-6 were added later as a way of refining the project to meet specific functional and technological requirements. No other goals were dropped from the initial plan for the project.

Methodology & Technical Approach

For the methodology of the project, the team took an agile iterative approach based loosely on Scrum. The iterations were a week in length and tasks were managed through the issue tracking provided by GitHub, where the codebase was also hosted. Team members worked largely independent of one another and virtual check-in meetings were only required twice per week. The daily standup meetings commonly used in Scrum were not required, in order to accommodate differences in schedules and availability. Retrospective meetings, where progress and expectations could be examined and adjusted, were performed on the last day of each development iteration. No automated pipelines for code changes were used, but all changes entering the main branch required passing unit tests and approval by the lead developer.

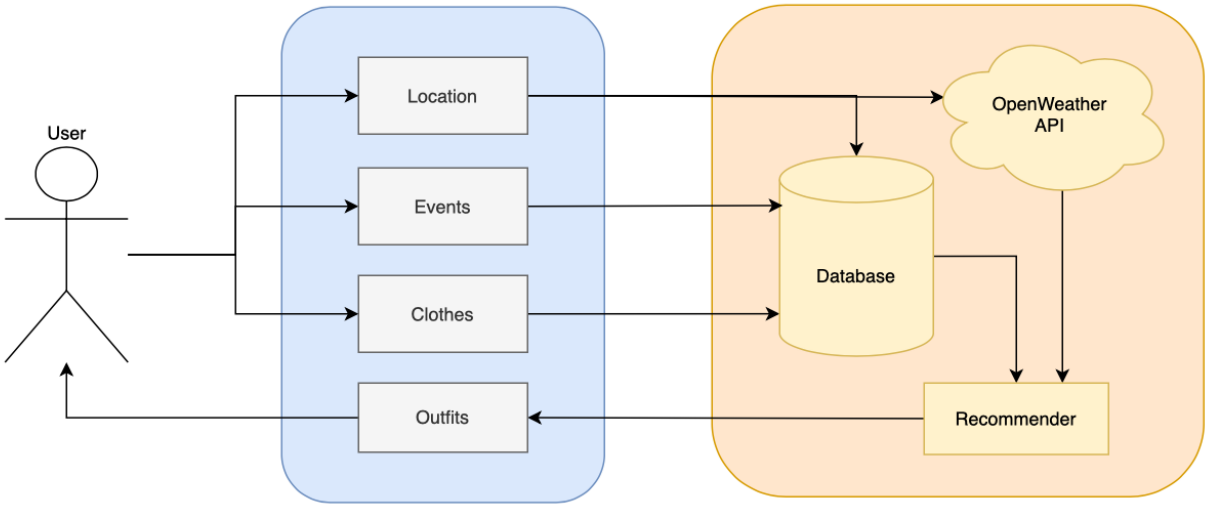
Technical Architecture

Application

Fit of the Day is an interactive Web application build with Node.js and a Web server framework called Express. All application code was therefore written in JavaScript, which was advantageous because developers only had to work with a single language across both the front and back end. Persistence was handled with a PostgreSQL database and an Object Relational Mapping (ORM) tool called Sequelize to handle querying.

The two main functions of the site are closet management and outfit generation. Figure 1 shows the technical diagram depicting the flow of data for outfit recommendation. Closet, location, and event data originates from the user and is input into the site. When this data is combined with current weather data it is used by the recommendation algorithm to generate outfits for a particular day.

Figure 1 Application Data Flow

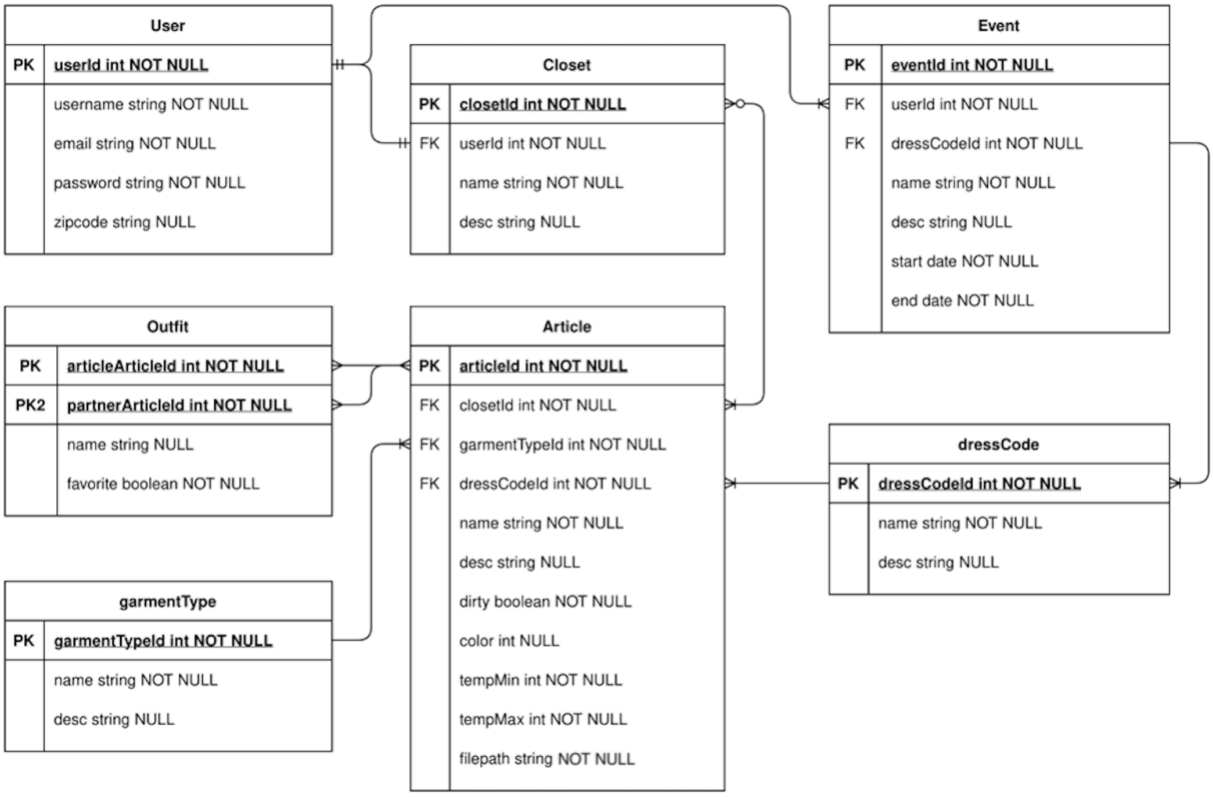


The recommendation algorithm consists of basic data processing and a number of database queries. The weather data enters the system largely unchanged. A saved location or ZIP Code is used to retrieve weather data from OpenWeather. The average temperature of the day is used as the target temperature that an outfit must match in order to be recommended. For the event data, the database is queried to retrieve the next upcoming event. This event's associated dress code, or how formal it is, is used to set the bounds of the generated outfit dress codes. If no event is found, then all dress codes are included. Finally, the user's closet is analyzed to find all matching outfits. If a user has marked any outfits as a favorite, then those outfits have a higher priority and are more likely to be recommended. Furthermore, if any article in an outfit is marked as being dirty, the recommendation system ignores the outfit entirely.

Database Schema

Figure 2 show the database schema used in Fit of the Day and the relationships between the different tables.

Figure 2 Entity Relationship Diagram



Testing

Overview

This section will be focused on the testing of the project. This includes the methodology chosen, the results of the testing, and the impacts of the test results.

Methodology

The two main testing methods chosen for the project were user acceptance testing (UAT) and unit testing. UAT was used because the project needed to be easily understood by any given end user. It also helped to verify correct functionality and find bugs unanticipated by the developers. All of the main features were tested to ensure that the site interface was user friendly and the concepts were understandable by new users. Unit testing was used to ensure only reliable

and consistent code was built. It tested isolated functionality at the most basic level of the project. The majority of all code committed included associated unit tests to test both normal and edge case scenarios.

Testing Scope

The main features tested for UAT were:

- Account creation
- Adjusting account settings
- Adding articles of clothing
- Editing articles of clothing
- Removing articles of clothing
- Create a saved outfit

These features were chosen for testing as they were deemed the most crucial features of the project, and as such need to be fully accepted by the end user.

Objectives

The primary strategy used for the UAT testing was to start the tester at a given point and ask them to perform certain tasks within the application with little to no guidance. This allowed the test supervisor to record the difference between the general user experience expected for Web sites and the actual experience provided by Fit of the Day. With this information the site could be improved to be more user friendly by matching expectations with reality. For unit testing, the strategy was to write tests for the most critical pieces of code first, then branch out to the less important areas.

Logs and Procedures

UAT

The initial phase of testing results is shown below. Each test case shows the expected result, actual result, a brief explanation of the pass or fail, and other relevant information.

Table 1 UAT Test 1.1

User Acceptance Test	
Item #	1
Test Case #	1
Date of Test	2/7/2021
Name of Tester	Brian
Role of Tester	User
Expected Output	Tester creates an account and is redirected to the site dashboard.
Actual Output	Tester was unable to create an account and get to the dashboard.
Pass / Fail	Fail
Reason for Pass / Fail	User made an account but was given failure error. The account was made, but user was not redirected to sign in screen. User was unable to sign into the account. User was attempting to stress test the site by using email formats that while technically correct, did not end in ".com". For example, endings using: "@no" and ".nit"
Further Testing	1. Password with numbers: Pass 2. Username with all capital letters: Fail 3. Email: jack@jack.com, Username: jack, Password: jack: Pass

Table 2 UAT Test 2.1

User Acceptance Test	
Item #	2
Test Case #	1
Date of Test	2/7/2021
Name of Tester	Brian
Role of Tester	User
Expected Output	Tester navigates to the account page and updates the saved ZIP Code for the account.
Actual Output	User got to account page with no issue. When they tried to set a new ZIP Code, they got a failure error. Attempted ZIP Code Input: 42069 Database entry: ObjectHTMLInputElement
Pass / Fail	Fail
Reason for Pass / Fail	Database shows the updated ZIP Code as an ObjectHTMLInputElement.

Table 3 UAT Test 3.1

User Acceptance Test	
Item #	3
Test Case #	1
Date of Test	2/7/2021

Name of Tester	Brian
Role of Tester	User
Expected Output	Tester navigates to the closet page and adds a new article of clothing, a top, a bottom, and a one piece.
Actual Output	User made each of the requested pieces of clothing. User did not initially recognize “New Article” button as the way creating a new item.
Pass / Fail	Pass
Reason for Pass / Fail	User completed the requested task with little to no issues.

Table 4 UAT Test 4.1

User Acceptance Test	
Item #	4
Test Case #	1
Date of Test	2/7/2021
Name of Tester	Brian
Role of Tester	User
Expected Output	Tester navigates to the closet page and edit the color and formality setting of an article of clothing.
Actual Output	User was able to edit an article with no issue.
Pass / Fail	Pass

Reason for Pass / Fail	User completed the requested task with little to no issues.

Table 5 UAT Test 5.1

User Acceptance Test	
Item #	5
Test Case #	1
Date of Test	2/7/2021
Name of Tester	Brian
Role of Tester	User
Expected Output	Tester navigates to the closet page and removes an article of clothing.
Actual Output	User was able to delete an article with no issues.
Pass / Fail	Pass
Reason for Pass / Fail	User completed the requested task with little to no issues.

Table 6 UAT Test 6.1

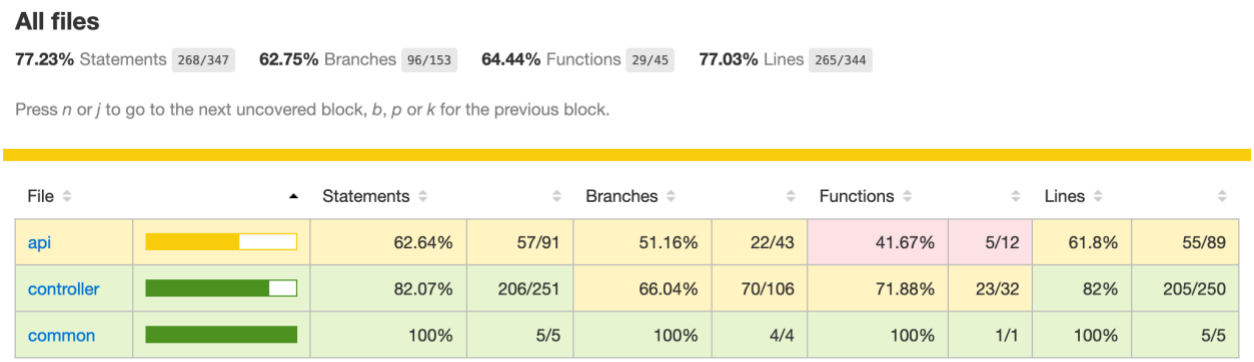
User Acceptance Test	
Item #	6
Test Case #	1
Date of Test	2/7/2021
Name of Tester	Brian

Role of Tester	User
Expected Output	Tester adds new articles for a top and a bottom, then creates a saved outfit with the newly created articles.
Actual Output	User was able to create an outfit with no issues.
Pass / Fail	Pass
Reason for Pass / Fail	User completed the requested task with little to no issues.

Unit Tests

The common JavaScript test runner Mocha was used to run all unit tests. A test coverage tool called Istanbul was used to monitor and report unit test coverage across all project files. The reports generated by Istanbul are displayed below in Figures 3-6.

Figure 3 All Unit Test Coverage



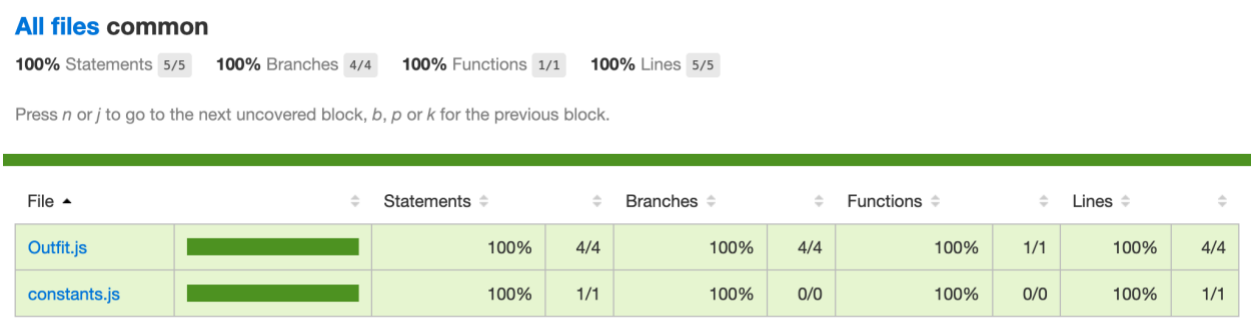
All files

77.23% Statements 268/347 **62.75%** Branches 96/153 **64.44%** Functions 29/45 **77.03%** Lines 265/344

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
api	62.64%	57/91	51.16%	22/43	41.67%	5/12	61.8%	55/89
controller	82.07%	206/251	66.04%	70/106	71.88%	23/32	82%	205/250
common	100%	5/5	100%	4/4	100%	1/1	100%	5/5

Figure 4 Common Unit Test Coverage



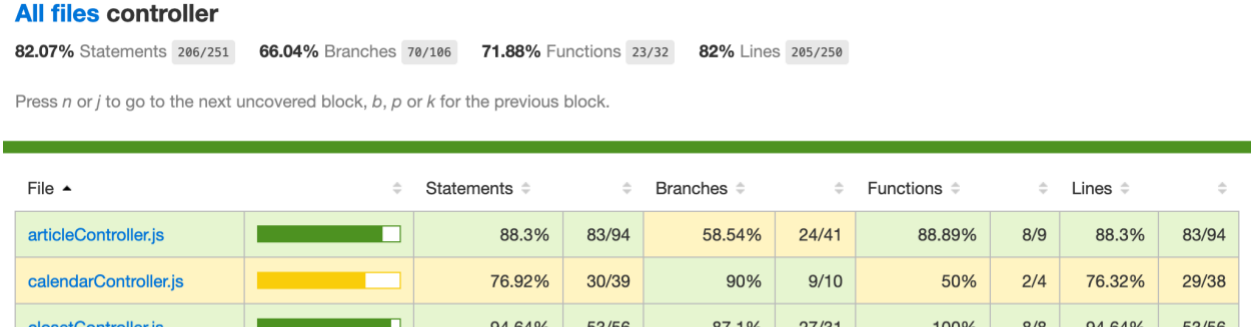
All files common

100% Statements 5/5 **100%** Branches 4/4 **100%** Functions 1/1 **100%** Lines 5/5

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
Outfit.js	100%	4/4	100%	4/4	100%	1/1	100%	4/4
constants.js	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Figure 5 Controller Unit Test Coverage



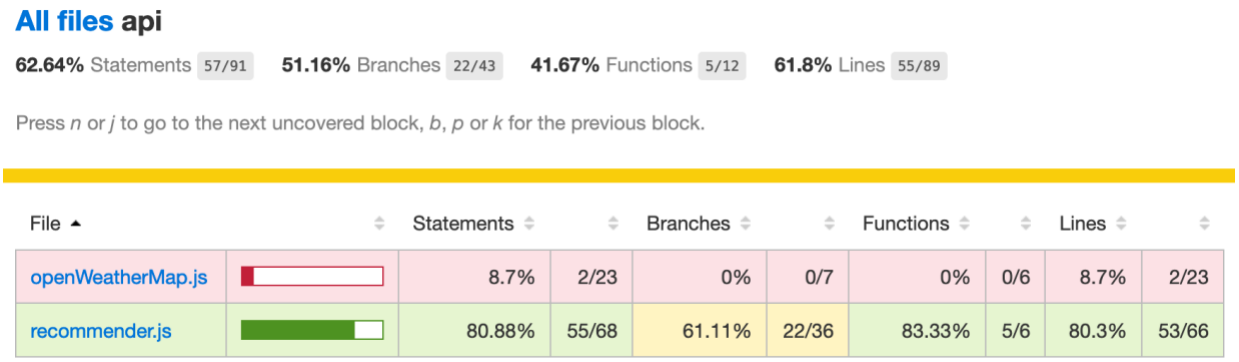
All files controller

82.07% Statements 206/251 **66.04%** Branches 70/106 **71.88%** Functions 23/32 **82%** Lines 205/250

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
articleController.js	88.3%	83/94	58.54%	24/41	88.89%	8/9	88.3%	83/94
calendarController.js	76.92%	30/39	90%	9/10	50%	2/4	76.32%	29/38
closetController.js	94.64%	53/56	87.1%	27/31	100%	8/8	94.64%	53/56
dashboardController.js	64.52%	40/62	41.67%	10/24	45.45%	5/11	64.52%	40/62

Figure 6 API Unit Test Coverage



Testing Review

The results from testing were extremely valuable to the project as a whole. They uncovered many bugs that were overlooked or missed. UAT was most useful to find issues with user interaction while the unit tests helped find and prevent issues with specific code functionality

During UAT, some major flaws and weaknesses in the existing form validation and user input processing were revealed. Table 1 depicts one of the tests that helped uncover this problem. Additionally, UAT showed that the system for associating ZIP Codes to user accounts was bugged and saved data in an incorrect format, as shown in Table 2. As a result of these tests, resolving all existing bugs was prioritized before adding any new functionality. Beyond finding bugs, UAT highlighted that the wording and terminology used on the site was not immediately understandable to new users. Examination of the struggles new users faced also emphasized the need for instructions or visual hints for empty site components on sign up.

Unit testing helped to identify edge and corner cases as well as breaking changes that affected multiple modules. All commits entering the main branch of the project were required to have all passing unit tests. The one outlier to the generally consistent and acceptable level test coverage is the module that worked with the OpenWeather Application Programming Interface

(API). Mocking this API was decided to be too much effort for the small returns, given the time constraints of the project and the relative isolation of this module's functionality.

Budget

The project budget was built from two main expense areas: labor cost and infrastructure cost. No additional fees for software licensing were factored in as no paid third-party software was used, nor was any intended to be used at a later point in time. Figure 7 shows the final project budget.

Figure 7 Project budget

	Rate Per/Hr	Work Effort (Hours)	1 X Costs	Ongoing Annual		
				Rate Per/Hr	Work Effort (Hours)	1 X Support Cost
Labor - IT	20	450	$ 9,000.00	20	10	$ 200.00
Labor - External			$ -		0	$ -
Software - External						8964.12
Hardware - External						
Misc.						
TOTAL			$ 9,000.00			$ 9,164.12

The project budget is based on developer labor, server and hosting costs. The labor cost was calculated with the assumed number of work hours spent at the time of project completion. The infrastructure costs were based on running one Web server and one database through Amazon Web Services (AWS) for one year, while not applying any free trials or discounts. The servers used in this calculation were not of the same level as those being used during project development. The development servers were exclusively using free tier services while the projected budget was based on larger and more robust servers, though not the maximum size available through AWS.

User Profiles

Two distinct user profiles are listed below. These tables list user roles and what knowledge someone would need to know to fulfill the roles. Table 7 shows the user profile for developers, maintainers, and any other type of site administrators. This includes anything from security analyst to database administrator or front-end developer. Table 8 describes the user profile for Fit of the Day's end user or customer.

Table 7 Developer User Profile

Developer User Profile
Application: GitHub, JavaScript, CSS, Pug, AWS
Potential Users: Developers
Software and Interface Experience: This user should have experience with software development or full stack development. They should understand JavaScript code, client/server communication, relational SQL databases.
Experience with Similar Applications: This individual would have experience with web servers.
Task Experience: This user should be fluent in the applications above to keep maintenance going.
Frequency of Use: After the application is created, developers will only gradually maintain it with bug fixes and updates to make sure it is still functioning.
Key Interface Design Requirements that the Profile Suggests: The developer will need to understand web server structure to be able to maintain it if patches are needed.

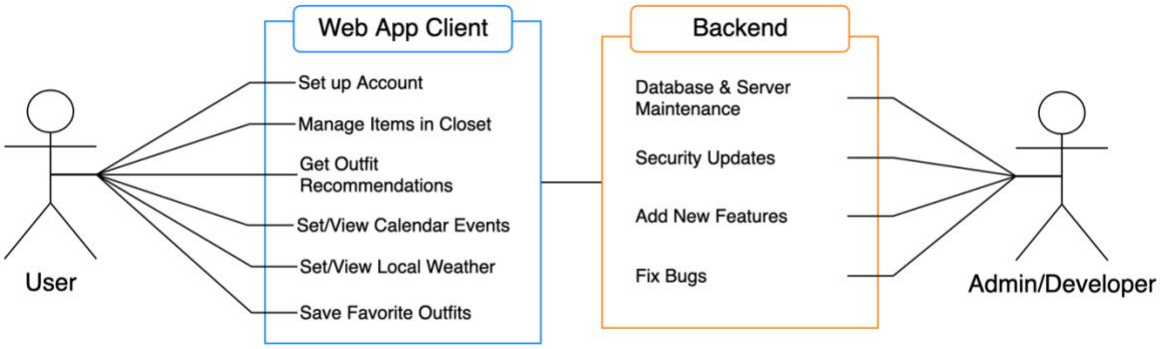
Table 8 Customer User Profile

Customer User Profile
Application: Fit of the Day Web App
Potential Users: General public
Software and Interface Experience: This user would need to be able to navigate a typical Web site.
Experience with Similar Applications: No additional experience should be needed to use the site.
Task Experience: The everyday user should be able to view the current forecast and next calendar event. They should also be able to generate outfits, save outfits, and add/remove articles of clothes to/from their virtual closet
Frequency of Use: Daily/Occasional
Key Interface Design Requirements that the Profile Suggests: They should be able to view forecasts, check calendar events, generate outfits, save outfits, add/remove articles of clothes to/from their "closet" and access the Web site with no issues.

Use Case Diagram

Figure 8 depicts a use case diagram which shows how the different user roles will interact with each other and the project as a whole. It shows the project's two main points of access and the possible interactions a given user role could have with them.

Figure 8 Use Case Diagram



Project Timeline

The timelines below are split into two tables. Figure 9 depicts the initial timeline that the project started out with. Its purpose was to give a general overview of where the project was expected to be at different points in development. Figure 10 shows an updated version made around halfway through the project's development. Dates of past milestones were updated to reflect reality and upcoming deadlines were adjusted to fit the projects velocity at the time. This adjustment helped to set more realistic expectations and goals for progress over the second half of development.

Figure 9 Initial Project Timeline

Task #	Task Name	Duration	Start Date	End Date
Task 1	Team Contract	1 Day	8/31/2020	8/31/2020
Task 2	Initial Server Setup	1 Month	9/15/2020	9/30/2020
Task 3	Initial Site Code	1 Month	10/1/2020	10/31/2020
Task 4	Site Database Access	1 Month	11/1/2020	11/30/2020
Task 5	Initial Features	2 Weeks	12/1/2020	12/14/2020
Task 6	Design Prototype	3.5 Months	8/31/2020	12/14/2020
Task 7	Solid Initial Feature Wrap-up? Addition Features	1 Month	1/1/2021	1/31/2021
Task 8	Additional Features?	1 Month	2/1/2021	2/27/2021
Task 9	Completion	----	----	March 1 2021
Task 10	IT Expo Prep	1.5 Months	March 1 2021	April 13 2021

Figure 10 Updated Project Timeline

Task #	Task Name	Duration	Start Date	End Date
Task 1	Team Contract	1 Day	8/31/2020	8/31/2020
Task 2	Initial Server Setup	1.5 Months	9/15/2020	10/15/2020
Task 3	Initial Site Code	1 Month	9/1/2020	10/7/2020
Task 4	Site Database Access	10 Days	10/1/2020	10/11/2020
Task 5	Initial Features	2 Months	10/1/2020	12/14/2020
Task 6	Design Prototype	3.5 Months	8/31/2020	12/14/2020
Task 7	Recommender System (Machine Learning)	4 Months	10/1/2020	2/28/2021
Task 8	Solid Initial Feature Wrap-up. Additional Features	1 Month	1/1/2021	1/31/2021
Task 9	Additional Features	1 Month	2/1/2021	2/27/2021
Task 10	Completion	----	----	3/1/2021
Task 11	IT Expo Prep	1.5 Months	3/1/2021	4/13/2021

Problems Encountered

The problems encountered during development came mainly from a lack of technical experience working with Node.js, live server hosting on AWS, and figuring out an effective recommender system. The coding skills of the development team varied widely between members. This caused initial development to be slow as the learning curve was at its highest, especially for the less experienced team members. The issue with the live server hosting resulted from the unforeseen expiration of a free student account. The AWS account that was being used to host the site initially had previously been used and only had partial student credit. When the account balance was depleted, the team was charged for the hosting before the mistake was discovered and corrected. Finally, in project planning there were many ideas for implementing the recommendation system. The initial idea was to create a machine learning algorithm that could learn and grow from each individual user's preferences and taste. As development progressed, the idea to incorporate machine learning became less feasible due to time constraints and lack of data science experience.

Recommendations for Improvement

To improve upon the project, a more robust recommendation system would be the primary goal. A machine learning algorithm could be added, as well as custom tags for the system to categorize clothes, for example those that follow a certain fashion trend. Other features that would add value to the project are support for external calendar providers as well as a mobile version of the application. External calendars such as Google Calendar would make it so the user would not need to manually add all their events in that are already stored elsewhere. A mobile version of the site would be more convenient to use and may see a higher usage than a Web application.

Conclusion

Over this project's lifecycle, the development team has grown tremendously. Managing the complexities of building a Web application from the ground up was a huge undertaking. One of the hardest problems to deal with initially was pacing. Since development was slow in the beginning, it had a ripple effect and crunched the deadlines together later on in the project.

The team also had to learn and develop some new skills. When starting out, no one on the team had much direct experience with the frameworks being used. Over the project's development, the team's main network tech, Tyler, learned to code again from a basic level after not programming for years. The lead developer, Liam, improved his skills as a full stack developer by learning about database ORMs, the Express framework, and managing a complex project. Daniel, the front-end designer, furthered his skill as a UX/UI developer and, like Tyler, learned more about coding. Both Tyler and Daniel learned how to make use of Git, GitHub, and its issue tracking and project management features. While the team had a fair share of

communication issues and disagreements, its members ultimately built mutual respect for one another and worked hard to make this project a success.

This project has undergone a variety of changes since its inception. At the end of the first half of development, the application only had basic functionality and a simple user interface. Throughout the latter half of development, the user interface was overhauled and core features were both added and improved. The two most substantial features included the improved recommendation system and the built-in calendar. The calendar was one of the three core requirements for the recommendation system data. With the calendar completed, the recommendation system could become fully formed, incorporating all streams of data: weather, events, and clothes.

Bibliography

Hoomans, Dr. Joel. 2015. "35,000 Decisions: The Great Choices of Strategic Leaders." Accessed October 2020. https://go.roberts.edu/leadingedge/the-great-choices-of-strategic-leaders.

Iyengar, Sheena S. and Mark R. Lepper. 2000. "When Choice is Demotivating: Can One Desire Too Much of a Good Thing?" *Journal of Personality and Social Pyschology.* https://doi.org/10.1037/0022-3514.79.6.995.

Appendix

Appendix A: IT Expo 2021 Poster