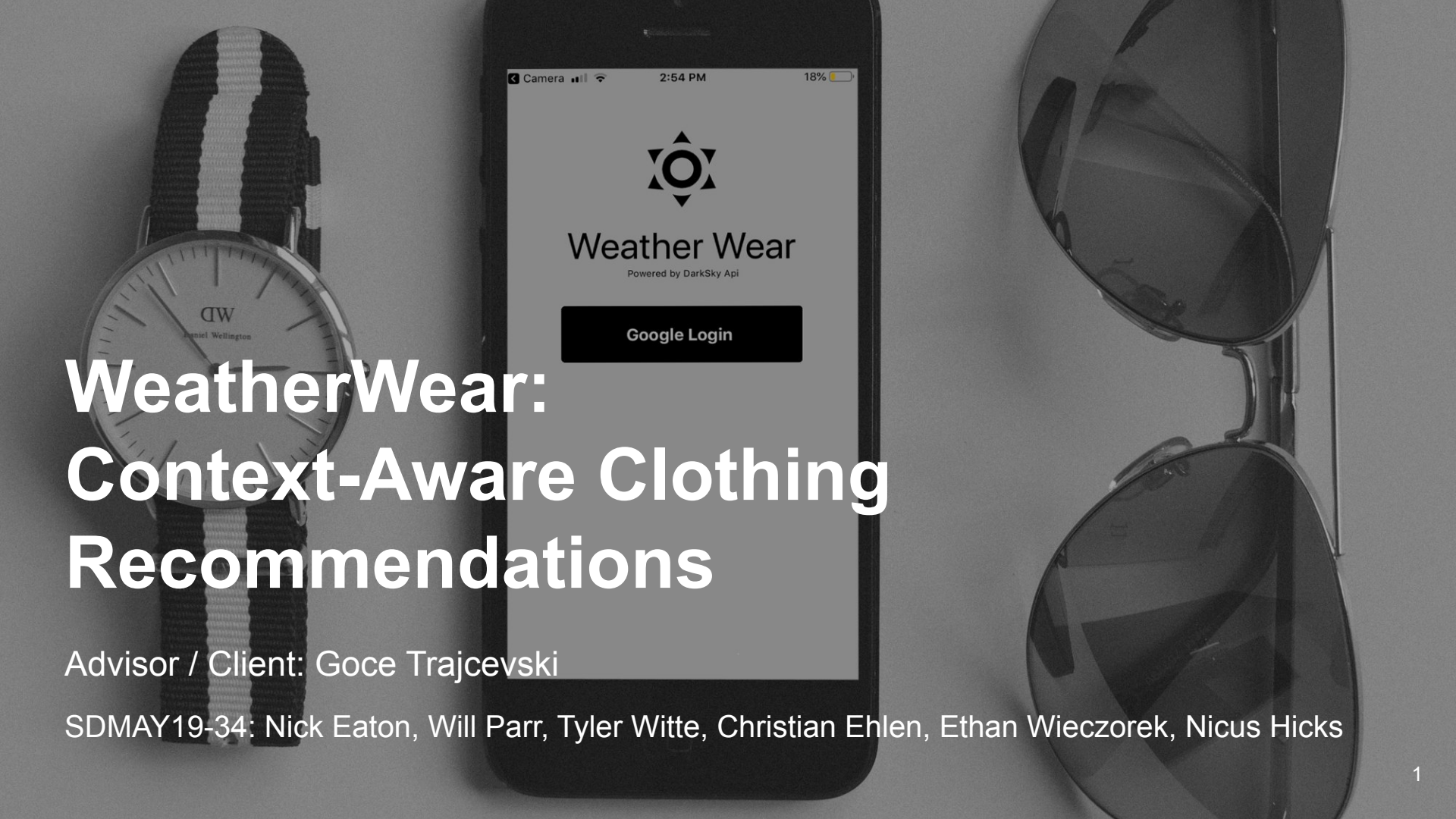


WeatherWear: Context-Aware Clothing Recommendations

Advisor / Client: Goce Trajcevski

SDMAY19-34: Nick Eaton, Will Parr, Tyler Witte, Christian Ehlen, Ethan Wieczorek, Nicus Hicks

Team Member Positions

Ethan Wieczorek: Lead Backend Developer

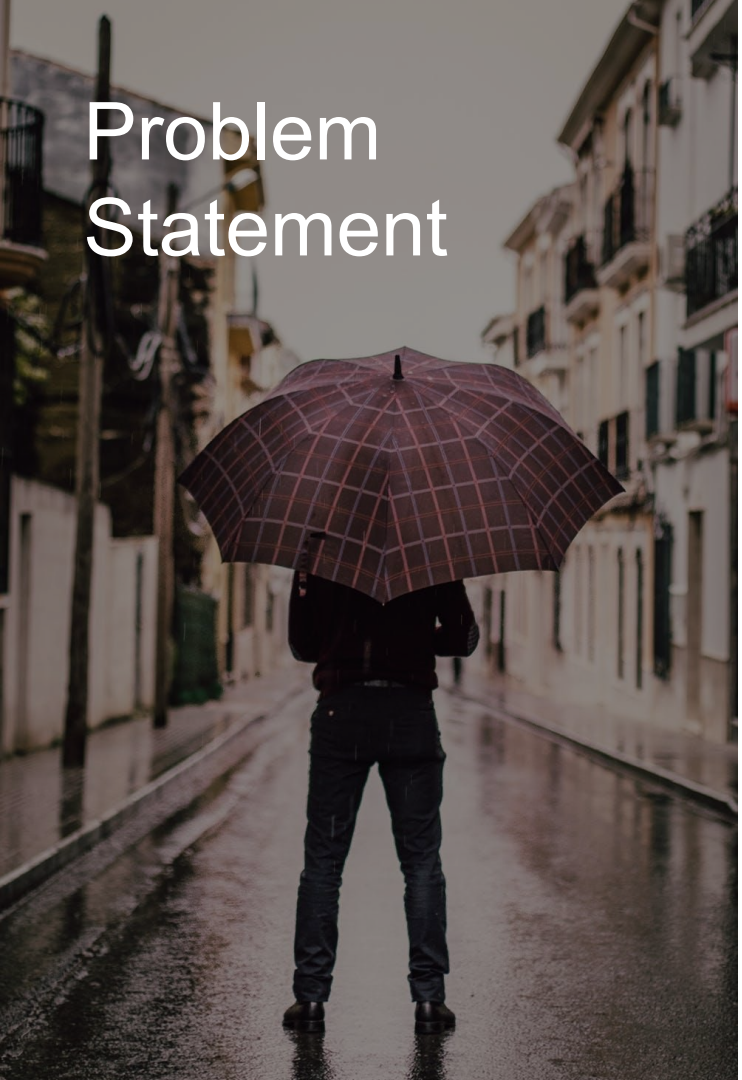
Nickolaus Eaton: Product Manager

Nicus Hicks: Director of Documentation and Reporting

Will Parr: Lead Frontend Developer

Tyler Witte: Lead Software Architect

Christian Ehlen: UI/UX and Quality Assurance Developer



Problem Statement

Problem

- Ambiguous and inefficient clothing selection for weather
- Selecting clothing to pack for trip
- Clothing appropriate for event types

Solution

- Individual wardrobe tracking
- Daily clothing suggestion
- Weather, Calendar, and Location tracker

Summary

The WeatherWear mobile app, for iOS and Android, leverages React Native, the DarkSky API, and an Azure SQL DB alongside a clothing-recommendation system to give users outfit ideas for the current day or packing lists for upcoming trips.

Market Survey - What makes this project *unique?*

- Mobile Application
  - Cross-device compatible
- Graphical User Interface (GUI)
  - Intuitive
  - Fast
- Personalized to You
  - Wardrobe
  - Schedule

★ Many other applications focus on style or manual choice of outfits.

★ Our application chooses from **your** clothes based on weather and event data

Functional Requirements

- Google Authentication / Login
- User Request for Clothing Items
- Clothing Managed in Wardrobe
- Context Matching for Weather
- Cross-Device Compatible

Non-functional Requirements

- Scalability
- Security
- Usability
- Performance
- Maintainability

Design Plan and Objectives

Assumptions:

- System supports multiple users
- System supports multiple devices per user
- User has consistent internet connection

Limitations:

- System must be connected to internet to function
- Users must have a smartphone or tablet

Project Milestones & Management

Phase 1: Planning
Aug 27 -Sep 25

Milestones:
Requirements
Solution Design
Use-cases & market survey

Phase 2: Development
Sep 26 - Apr 20

Milestones:
Technical Design
Mockups
Front End UI
User Login and Storage
Clothing Prediction
Stable Alpha
Stable Beta
Testing

Phase 3: Release
Apr 21 - May 5

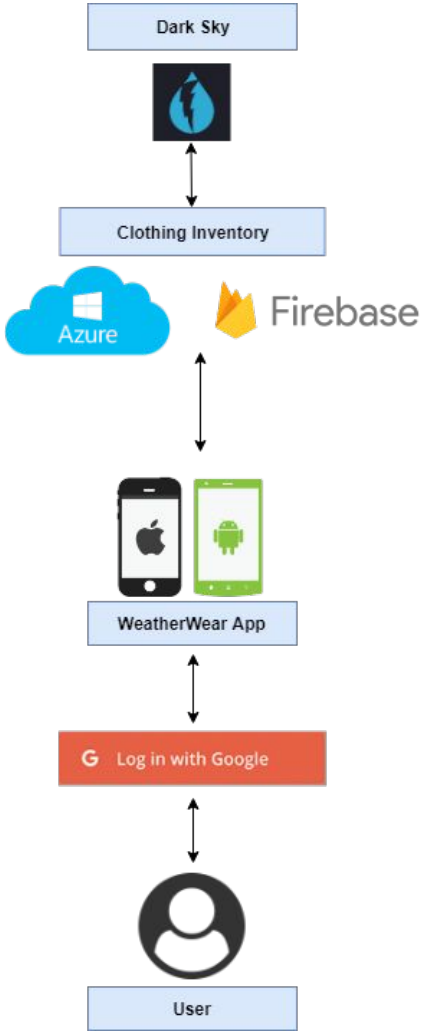
Milestones:
Product Release
Improvements

Technology Stack and Development Tools

- DarkSky API
- Azure Cloud Services
- Google Firebase
- Google Authentication
- React Native
- ExpressJS
- Axios

High Level Architecture

- User Login with Google
- WeatherWear mobile app
- App interacts with Node.js backend
- Azure SQL DB and Firebase authentication for user and clothing data
- DarkSky API for detailed weather data



Functional Decomposition

WeatherWear app:

- Trip Planning Request
- Clothing Recommendation Request

Node.js Back-end:

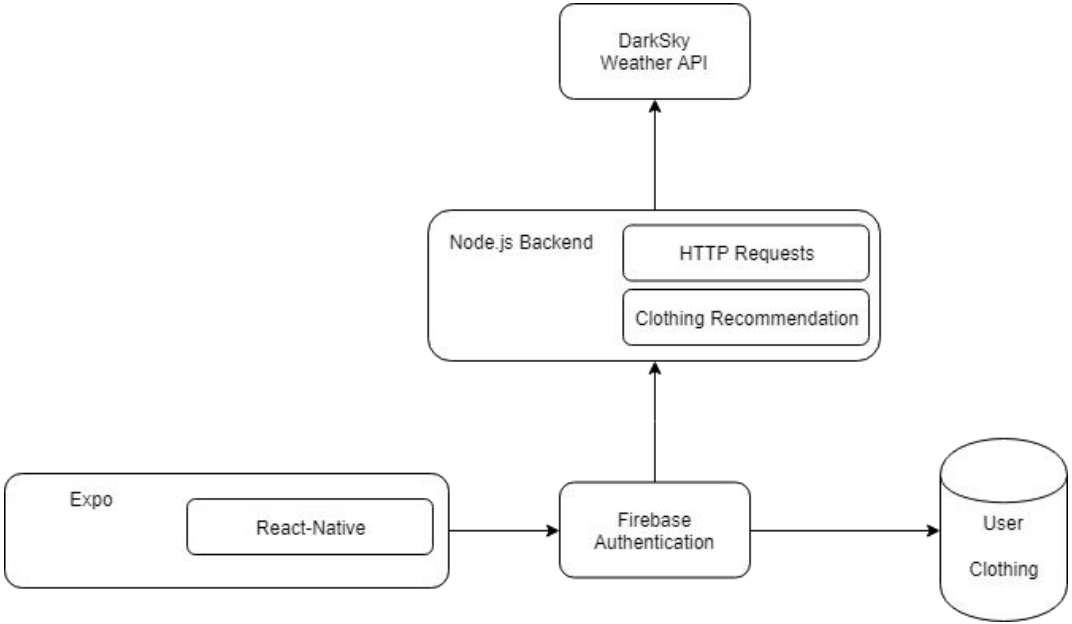
- RESTful API
- Recommendation Algorithms

Weather:

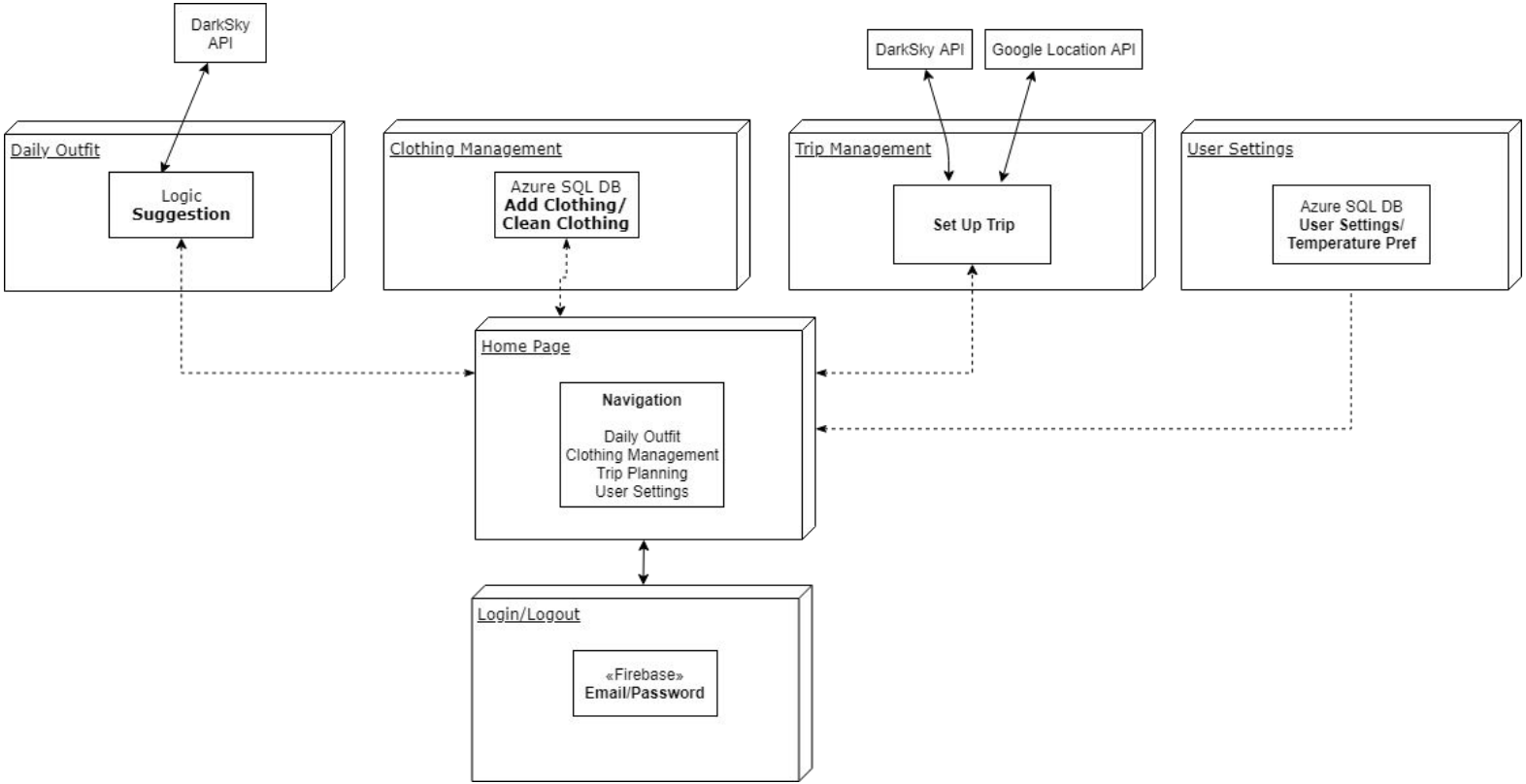
- Dark Sky API
- React-Native-Weather

Clothing:

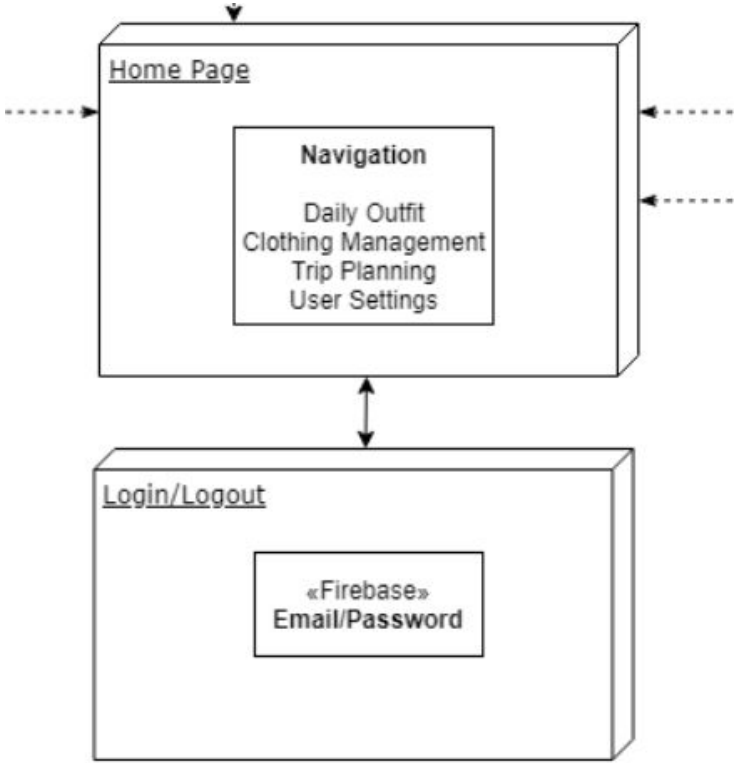
- Azure SQL Database



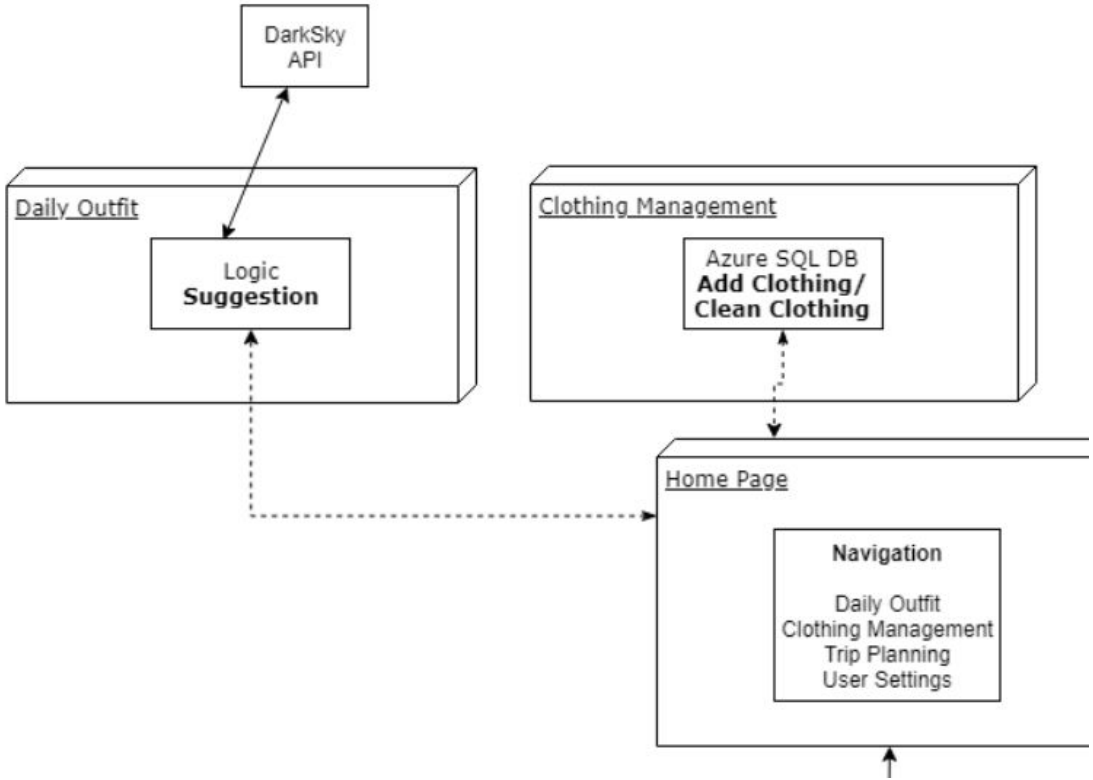
Detailed Design



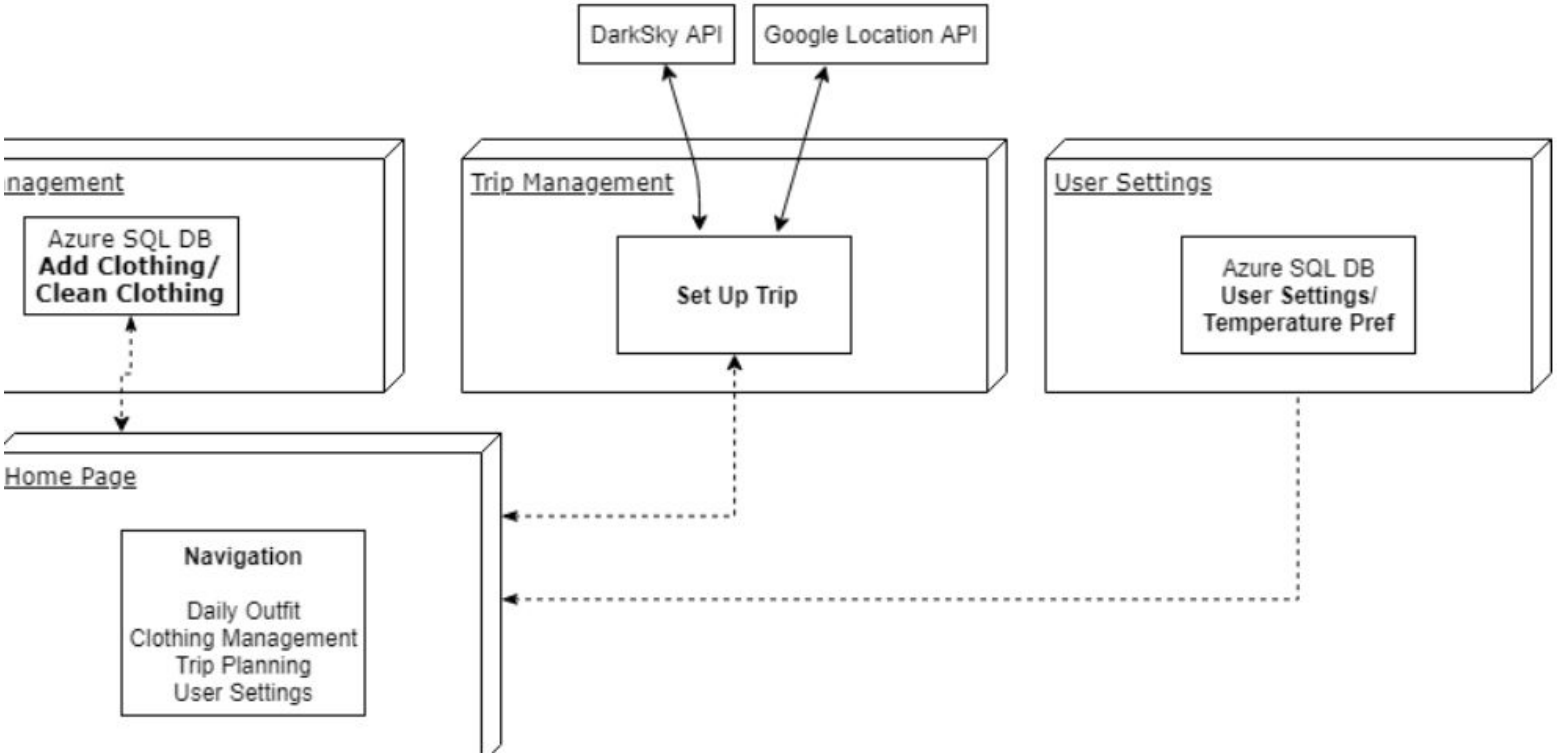
Login and Home Page (Front-End)



Daily Outfit and Wardrobe Management



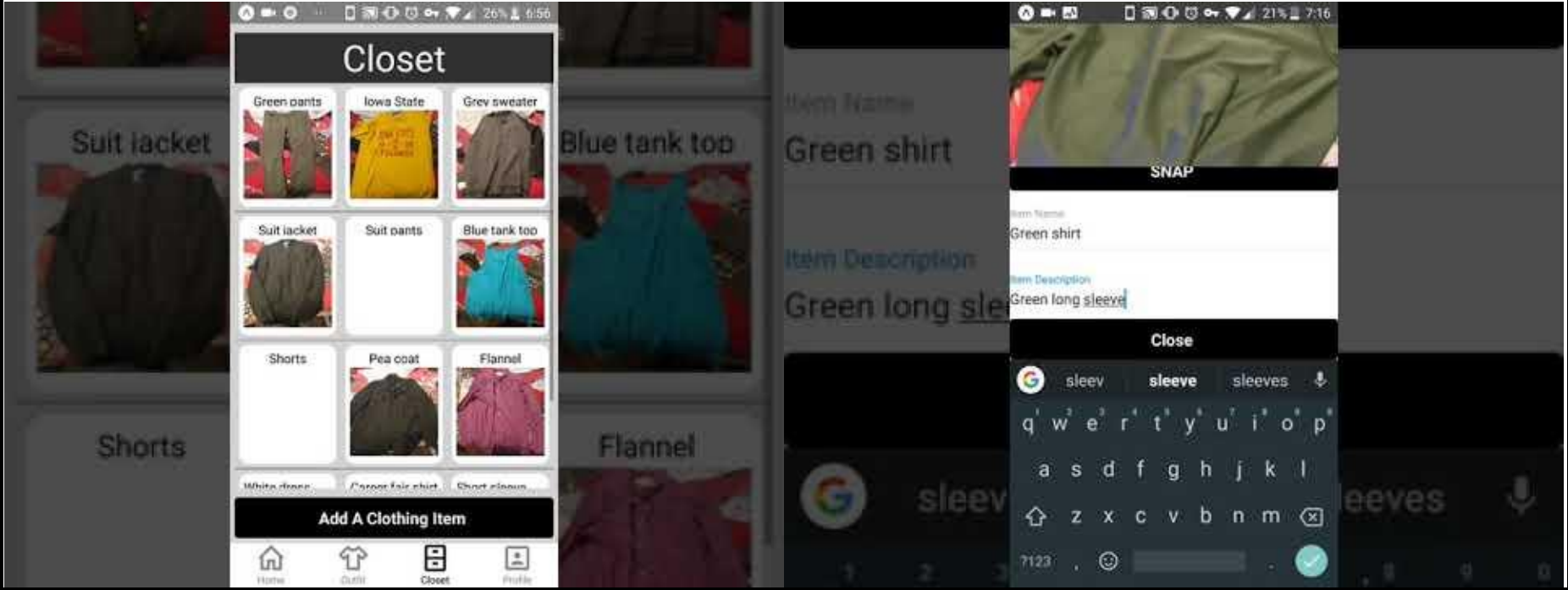
Trip Management and User Settings



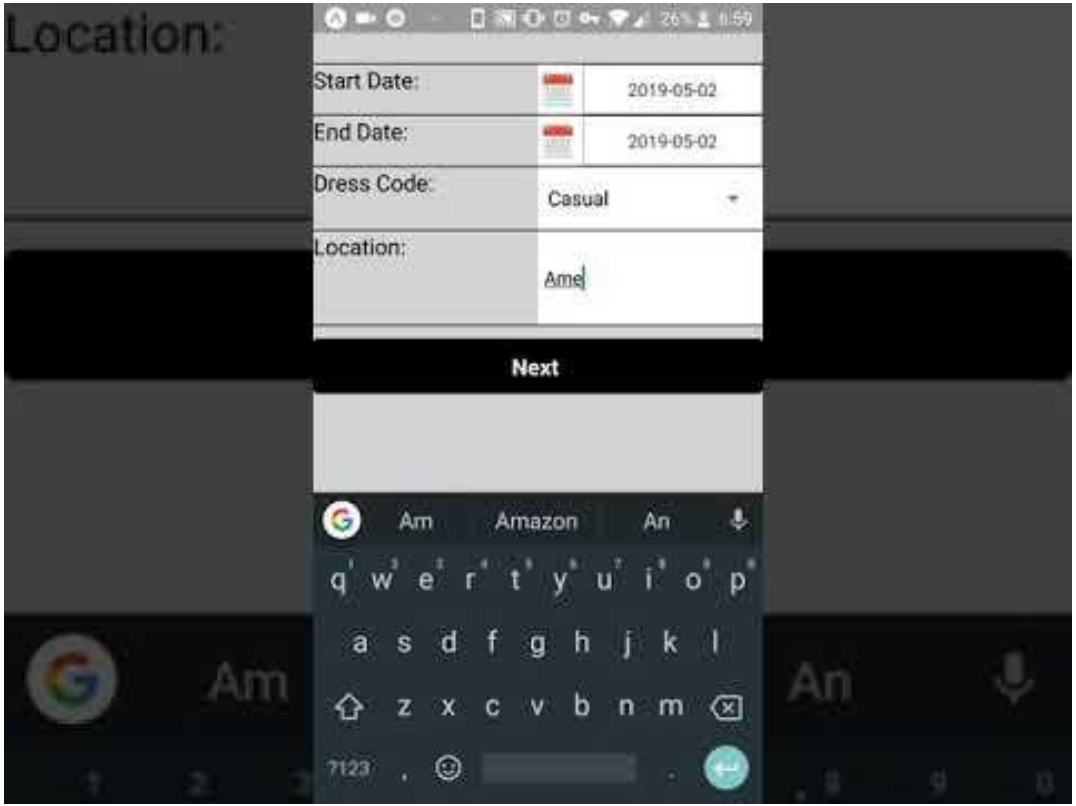
Demonstration - Login, Closet, Profile

Login, Closet, Profile

Add Clothing



Demonstration - Outfit and Trip Planning



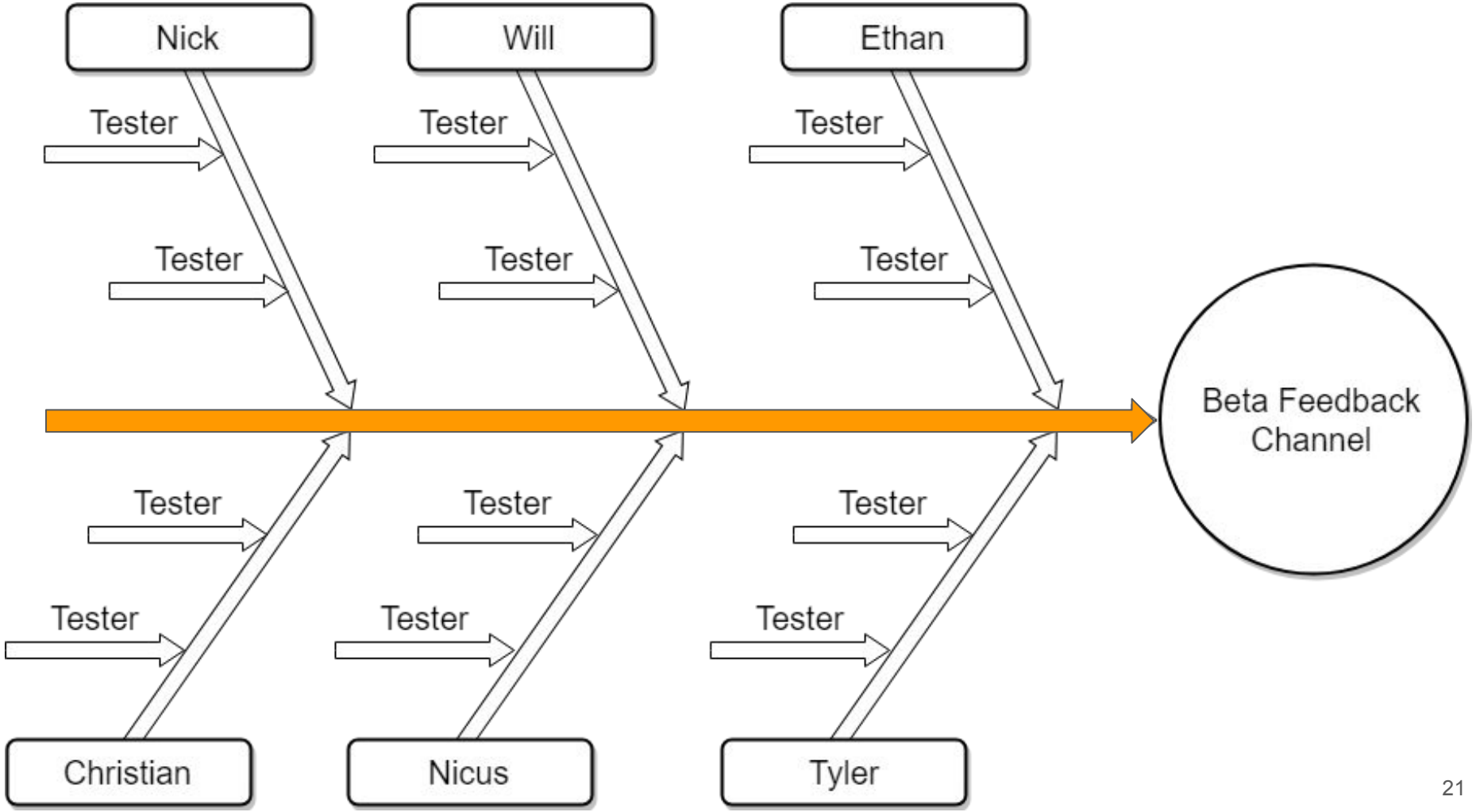
Applicable Standards & Best Practices

- Agile Development Methodology : Efficient task scheduling per team member
- Testing using Enzyme and Jest, Node.js unit testing
- Git Monorepo with Dev branch
- Team Members own unique positions & user stories
- Merge request code reviews

Testing Plan

Testing Classifications:

- Unit Testing
- System and Integration Testing
- Performance and Stress Testing
- User Acceptance Testing
- Beta Testing


Nick
Will
Ethan
Tester
Tester
Tester
Tester
Tester
Tester
Beta Feedback Channel
Tester
Tester
Tester
Tester
Tester
Tester
Christian
Nicus
Tyler

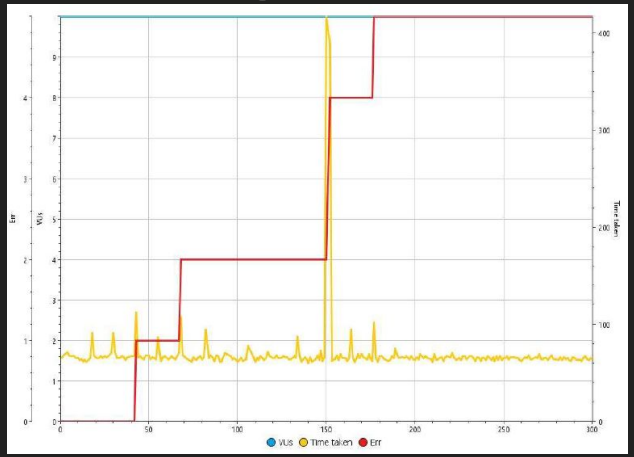
Test Plan - Functional

Integration	Validation
Location and Weather Data	• Correct Location • Correct Weather
Database Relations	• Correct Clothing per user • Update User Settings
Device cross-compatibility	• iOS and Android both function the same
Clothing Categorization	• Correct Categorization

Test Plan - Non-Functional

Integration	Validation
Performance	• Authentication • Recommendations
Security	• Administrative functions • Passwords • Clothing access
Usability	• Temperature settings
Compatibility	• APIs

Results of Testing




```
Test Suites: 10 passed, 10 total
Tests:       10 passed, 10 total
Snapshots:   10 passed, 10 total
Time:        10.005s
Ran all test suites.
```

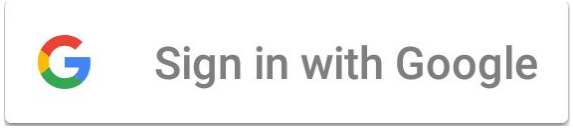
System Load Testing using SMARTBEAR LoadUI
Tests run: 2741
Avg. Response Time: 72 ms

```
POST /users/CreateUser 200 362.503 ms - 21
    √ should create a SINGLE user on /CreateUser POST (381ms)
{ host: '127.0.0.1:50836',
  'accept-encoding': 'gzip, deflate',
  'user-agent': 'node-superagent/3.8.3',
  id: '1234567890',
  connection: 'close' }
GET /users/GetUser 200 49.866 ms - 174
    √ should get a single user on /GetUser GET (56ms)
PUT /users/UpdateUser 200 50.797 ms - -
    √ should update a single user's name and preferences on /UpdateUser PUT (55ms)
PUT /users/UpdateUserLocation 200 40.611 ms - -
    √ should update a single user's location on /UpdateUserLocation PUT (44ms)
{ host: '127.0.0.1:50842',
  'accept-encoding': 'gzip, deflate',
  'user-agent': 'node-superagent/3.8.3',
  id: '1234567890',
  connection: 'close' }
GET /users/GetUser 200 39.396 ms - 162
    √ should get the updated user on /GetUser GET (43ms)
DELETE /users/DeleteUser 200 41.036 ms - -
    √ should delete a single user on /DeleteUser DELETE (44ms)

  Clothing
Connected to SQL Server
HEADERS: {"host":"127.0.0.1:50847","accept-encoding":"gzip, deflate","user-agent":"node-superagent/3.8.3","id":"12345
67890","content-type":"application/json","content-length":"44","connection":"close"}
BODY: {"firstname":"Tester","lastname":"McTester"}
Results: { recordsets: [ [ [Object] ] ],
  recordset: [ { id: '1234567890' } ],
  output: {},
  rowsAffected: [ 1 ] }
POST /users/CreateUser 200 292.617 ms - 21
    √ should create a test user on /CreateUser POST (296ms)
POST /clothing/AddClothing 200 42.435 ms - 11
    √ should create a SINGLE clothing item on /AddClothing POST (46ms)
GET /clothing/GetAllClothing 200 43.378 ms - 172
    √ should get a single clothing item on /GetAllClothing GET (47ms)
GET /clothing/GetAllClothingNames 200 36.780 ms - 21
    √ should get a single clothing item name on /GetAllClothingNames GET (40ms)
GET /clothing/GetItemInformation 200 39.921 ms - 172
    √ should get the correct clothing information name on /GetItemInformation GET (44ms)
GET /clothing/GetClothingByCategory 200 42.944 ms - 172
    √ should get the correct clothing item name on /GetClothingByCategory GET (47ms)
PUT /clothing/UpdateClothing 200 44.339 ms - -
    √ should update a single clothing item on /UpdateClothing PUT (49ms)
GET /clothing/GetItemInformation 200 36.708 ms - 177
    √ should get the correct UPDATED clothing information name on /GetItemInformation GET (41ms)
DELETE /clothing/DeleteClothing 200 47.412 ms - 11
    √ should delete a single clothing item on /Clothing DELETE (50ms)
GET /clothing/GetClothingRecommendation 200 502.142 ms - 228
    √ should get the correct object type on /GetClothingRecommendation GET (507ms)
POST /clothing/GetTripRecommendation 200 386.584 ms - 633
    √ should get the correct object type on /GetTripRecommendation GET (391ms)
DELETE /users/DeleteUser 200 41.356 ms - -
    √ should delete the test user on /Users DELETE (45ms)
```

Risks & Mitigation

- Account information
  - Mitigation: Google Login
- Loss of local data
  - Mitigation: Data stored on Azure database
- Secure access to data
  - Mitigation: Google Security/Authentication



Resource & Cost Estimate

Resources Required:

- Deployment Platforms
- Azure SQL database
- Server system

Costs:

- API query limits when exceeded
- Database costs for security & size
- Apple Developer License

Lessons Learned

- Team Organization
- Project Scope
- System Development
- Task Sharing
- Risk Mitigation

Future: Potential Directions

- Style Recommendations
- Integration with more diverse clothing types
- E-Commerce/Advertising integration
- Camera AI Clothing Recognition
- Social Media Integration
- Distributed Database

Q&A