GITGOOD

Help gitGood, gitBetter

REQUIREMENTS SPECIFICATIONS

DEMO 1

WEATHER TO WEAR

EPI-USE

KYLE LIEBENBERG DIYA BUDHIA ALISHA PERUMAL	IBRAHIM SAID TAYLOR SERGEL

Table of Contents

INTRODUCTION

Weather to Wear is a mobile application that aims to simplify weather forecasts into a personalized wardrobe consultant. By analysing real-time weather forecasts with the combination of a user's personalised styling preferences, it uses AI-driven outfit suggestions to provide appropriate fashion recommendations from a user's personal clothing collection. The application seeks to streamline the daily challenge of choosing outfits that are weather-appropriate, ensuring users step out confidently prepared for any weather condition while expressing their unique fashion sense.

VISION AND OBJECTIVES

Weather to Wear aims to transform and simplify the daily act of choosing outfits into a personalized, engaging experience by using weather data, AI-driven outfit recommendations, and social interactions. This enables users to dress confidently and stylishly for any weather condition. The primary objective is to develop a system that integrates weather forecasts with users wardrobe to suggest outfits that are tailored to weather conditions and personal style preferences. The objective is to deliver a system that personalizes outfit recommendations, enhances user engagement and utilizes innovative visualization through the means of virtual try-ons.

BUSINESS NEEDS

Weather to Wear offers a unique, time-saving solution for outfit-planning. By combining weather-driven AI recommendations, AR visualizations, and social collaboration, Weather to Wear will stand out in the fashion-tech industry. It delivers an innovative, impactful solution that supports a growing user base with a scalable PWA, ensuring that data is handled in a secure manner and communication is encrypted for user privacy.

PROJECT SCOPE:

The project scope for Weather to Wear outlines the key features and deliverables for the mobile application. The project scope, based on the requirements are:

- Clothing Catalog System:
    - Develop a user-managed wardrobe database with detailed clothing profiles and options to add, edit, or remove items.
- Weather API Integration:
    - Implement a flexible model to integrate multiple weather APIs for reliable, real-time forecast data.
- Intelligent Recommendation Engine:
    - Create an AI-driven system using neural networks to suggest personalized, weather-appropriate outfits with a user rating system for learning preferences.
- Social Platform:
    - Build a real-time feed for sharing outfits, enabling comments and collaborative feedback, with content filtering for safety.
- Construct a Virtual Closet Management System:
    - Develop tools to organize a user’s wardrobe by categories such as occasion or season, enabling easy navigation and selection.
- Create an Outfit Emergency Response Platform:
    - Design a system to deliver quick outfit recommendations in response to sudden weather changes, ensuring users can adapt their clothing choices efficiently.
    - Provide practical adaptation instructions for modifying current outfits to suit unexpected weather conditions, minimizing disruption.
- Predictive Wardrobe Planning:
    - Develop a system that generates outfit plans for upcoming trips or events by integrating precise weather forecasts for the destination and dates, enabling users to pack and prepare efficiently.
- Visual Calendar:
    - Create a visual calendar displaying complete outfit recommendations for the upcoming week, with adaptive modifications that update automatically as weather forecasts change, ensuring users are always prepared for current conditions.

USER STORIES

1. As a user, I want to add my clothing items to the closet catalog so that the system can recommend appropriate outfits based on my wardrobe.
2. As a user, I want to add detailed tags to clothing items, such as color and temperature suitability
3. As a user, I want to edit details of clothing items from my virtual closet so that my closet catalog stays up-to-date with my current physical wardrobe
4. As a user, I want to remove clothing items from my virtual closet so that my closet catalog stays up-to-date with my current physical wardrobe
5. As a user, I want to sort my clothing items into different categories so that the system can quickly identify items for recommendations
6. As a user, I want the system to pull real-time weather data from the weather API so that I receive accurate outfit recommendations
7. As a user, I expect constant polling of real-time weather data from multiple weather APIs so that I receive accurate outfit recommendations even when one API fails
8. As a user, I want outfit suggestions based on the day's weather forecast so that I can dress appropriately without manual planning.
9. As a user, I want to rate recommended outfits so that the system learns my style preferences over time.
10. As a user, I want layering suggestions for temperature changes throughout the day so that I'm prepared for varying conditions.
11. As a user, I want to share my outfit combinations on a real-time feed
12. As a user, I want to comment on friends' outfit posts so that I can engage in collaborative style discussions
13. As a user, I want inappropriate content to be filtered from the social feed so that the platform remains safe and enjoyable
14. As a user, I want tools to organize my wardrobe by season so that I can easily find items for specific events.
15. As a user, I want to categorize outfits by occasion so that I can quickly select appropriate clothing for events like work or parties.
16. As a user, I want quick outfit recommendations for sudden weather changes so that I can adapt my clothing without stress.
17. As a user, I want instructions for adapting my current outfit to unexpected weather so that I can make practical adjustments on the go.
18. As a user, I want to use augmented reality to virtually try on outfit combinations so that I can see how they look before dressing.

19. As a user, I want 360° visualizations of outfits in realistic lighting so that I can evaluate them from all angles.
20. As a user, I want outfit plans for upcoming trips based on destination weather forecasts so that I can pack and prepare efficiently.
21. As a user, I want a visual calendar of outfit recommendations for the week so that I can plan my wardrobe with adaptive updates as forecasts change.
22. As a user, I want to see recommended outfits displayed on a customizable 3D avatar so that I can visualize how my clothes look together.
23. As a user, I want the 3D visualization to work seamlessly across devices so that I can use it on my phone or tablet.

USER CHARACTERISTICS

Weather To Wear users are tech-savvy and seek efficient, personalized solutions to streamline outfit planning. They value style, desire weather-appropriate clothing recommendations, and enjoy social engagement through sharing outfits and receiving feedback. They often face decision fatigue when matching outfits to weather and need intuitive, accessible tools like AR try-ons and 3D visualizations to enhance their experience.

1. Demographics
    - Age: Primarily 18–40 years old, spanning young adults to early middle-aged users who are fashion-conscious and tech-savvy.
    - Gender: All genders, as the system supports diverse clothing styles and personal preferences.
    - Occupation: Students, young professionals, and creatives who value style and practicality in their daily routines.
2. Technology Usage
    - Device Preference: Regular users of smartphones and tablets, comfortable with mobile apps and progressive web apps (PWAs).
    - Tech Literacy: Moderate to high, capable of navigating intuitive interfaces, uploading clothing items, and engaging with augmented reality features.
    - Internet Access: Frequent access to reliable internet, though they may experience intermittent connectivity (supported by offline caching).
3. Fashion and Style Interests
    - Fashion Awareness: Interested in personal style, seasonal trends, and possibly celebrity/influencer fashion, seeking to express individuality.
    - Decision Fatigue: Finds morning outfit selection time-consuming or challenging, especially when balancing weather and style.

FUNCTIONAL REQUIREMENTS

FR-1: Clothing-Catalog Management

- **FR-1.1:** The system shall allow a user to add a wardrobe item by taking a photo and completing a short form (name, category, material, weather-appropriateness, tags).
- **FR-1.2:** The system shall let the user edit or delete any wardrobe item they own.
- **FR-1.3:** The system shall provide search, filter and sort capabilities over the wardrobe (category, colour, material).
- **FR-1.4:** Wardrobe data and images shall be persisted so that a user's catalog is available across devices.
- **FR-1.5:** Each wardrobe item shall support tagging for weather conditions (hot, rainy, sunny) and occasions (work, casual).
- **FR-1.6:** The system shall enforce a maximum file size and accepted image types (JPEG, PNG) during wardrobe uploads.

FR-2: Weather-Data Integration

- **FR-2.1:** The system shall retrieve a location-specific weather forecast (min/max temperature, precipitation, wind, humidity, condition) from an external API.
- **FR-2.2:** If the primary provider fails, the system shall automatically fallback to a secondary API without user intervention.
- **FR-2.3:** Weather data shall be refreshed at a configurable interval (default: every 6 hours) so that recommendations stay current.
- **FR-2.4:** The system shall attempt to automatically detect the user's location using IP-based geolocation.
- **FR-2.5:** If automatic detection fails, the system shall allow manual location input from the user.
- **FR-2.6:** The weather module shall log each API call's success/failure and selected provider for debugging and analytics.

FR-3: Wardrobe-Recommendation Engine

- **FR-3.1:** Given a forecast and the user's wardrobe, the system shall produce at least one ranked outfit recommendation for a selected day.
  - **FR-3.1.1:** The ranking shall consider temperature range, weather condition, and user preferences.
- **FR-3.2:** The engine shall learn from user feedback captured through a 5 star rating system.
- **FR-3.3:** Recommendations shall include layering advice when forecasts indicate significant temperature variation.
- **FR-3.4:** The engine shall exclude items marked unsuitable for the predicted conditions.
- **FR-3.5:** The system shall avoid recommending identical outfits across consecutive days, where alternatives exist.
- **FR-3.6:** The engine shall take time-of-day into account (e.g., colder mornings vs. warm afternoons) if available in the forecast.

FR-4: Social-Sharing Platform

- **FR-4.1:** Users shall be able to publish an outfit post (image + caption + list of items) to a social feed.
- **FR-4.2:** Followers shall be able to like and comment on a post in real time.
  - **FR-4.2.1:** Likes and comments shall update asynchronously without page reload.
- **FR-4.3:** The system shall filter or block offensive text or images before a post becomes visible.
- **FR-4.4:** Privacy controls shall let the author restrict visibility to:
  - **FR-4.4.1:** Public
  - **FR-4.4.2:** Followers only
  - **FR-4.4.3:** Private (visible only to user)
- **FR-4.5:** The system shall allow users to follow and unfollow others.
  - **FR-4.5.1:** A users feed shall show posts of those who they follow

FR-5: User-Feedback & Learning

- **FR-5.1:** The system shall record each outfit rating action against the corresponding outfit version.
- **FR-5.2:** The user's preferences shall be updated after each outfit rating action.
  - **FR-5.2.1:** The recommendation engine shall incorporate user's preference and weather conditions into future ranking within 24 hours or upon user refresh.
- **FR-5.3:** The system shall allow users to view and manage their past feedback history.
- **FR-5.4:** The engine shall increase outfit diversity if consistent negative feedback is received for similar combinations.

FR-6: Virtual-Closet Analytics

- **FR-6.1:** The system shall display usage statistics (e.g., most-worn items, cost-per-wear).
- **FR-6.2:** Users shall be able to tag clothing by occasion and filter their closet on those tags.
- **FR-6.3:** The system shall visualize wardrobe insights using graphs (e.g., pie chart for usage distribution).

FR-7: Emergency-Outfit Response

- **FR-7.1:** When the forecast changes beyond a configurable threshold (e.g., ≥ 5 °C drop or rain chance > 60%), the system shall:
  - **FR-7.1.1:** Notify the user
  - **FR-7.1.2:** Suggest a new outfit suitable for the new forecast
- **FR-7.2:** Notifications shall be sent via in-app alert or push notification if enabled.

FR-8: Fashion Time-Machine Planner

- **FR-8.1:** The system shall allow users to attach events or trips to future calendar dates.
- **FR-8.2:** For each future date with an event, the system shall:
  - **FR-8.2.1:** Pre-compute an outfit based on long-range forecast
  - **FR-8.2.2:** Update the outfit automatically if the forecast changes

FR-9: Virtual Try-On

- **FR-9.1:** The app shall provide an AR preview that overlays the recommended outfit on:
  - **FR-9.1.1:** A live camera view
  - **FR-9.1.2:** A customizable 3D avatar
- **FR-9.2:** The try-on experience shall function in modern browsers that support camera access.

FR-10: Cross-Cutting Functionalities

- **FR-10.1:** Authentication & Authorisation
  - **FR-10.1.1:** All user-modifiable endpoints shall require authentication.
  - **FR-10.1.2:** Each user shall only be able to access and modify their own wardrobe, preferences, and posts.
- **FR-10.2: Offline Mode**
  - **FR-10.2.1:** The PWA shall cache the latest wardrobe and recommendations.
  - **FR-10.2.2:** The app shall queue user actions (e.g., ratings) and sync them once back online.
  - **FR-10.2.3:** Weather forecasts shall be cached to allow the recommendation engine to generate outfits while offline
- **FR-10.3:** Multiplatform Delivery
  - **FR-10.3.1:** The app shall work in modern browsers (Chrome, Safari, Firefox).
  - **FR-10.3.2:** The app shall support installation as a home screen shortcut on desktop.

- **FR-10.4:** Logging & Monitoring
  - **FR-10.4.1:** The backend shall log API failures, user actions, and system warnings.
  - **FR-10.4.2:** Admins shall be able to review logs via the dashboard.
- **FR-10.5:** Testability
  - **FR-10.5.1:** Each module (auth, weather, recommendation) shall include unit and integration tests.
  - **FR-10.5.2:** The system shall be CI/CD integrated, running tests on each publish.

TECHNOLOGY REQUIREMENTS

FRONTEND

CONSIDERATION 1 - REACT NATIVE

OVERVIEW

React Native is a JavaScript-based framework for building cross-platform mobile applications, leveraging React's component-based architecture. It compiles to native code, enabling near-native performance, and supports PWA development through web-based React for browser compatibility.

ADVANTAGES

- Rich Ecosystem: Extensive libraries and tools (e.g., React Navigation, Redux) support complex features like social feeds, API integration, and state management.
- AR and 3D Support: Libraries like ViroReact or Three.js enable AR and 3D visualization for virtual try-ons and avatar rendering, critical for the wow factor (Fashion Time Machine, 3D avatars).
- PWA Compatibility: React's web capabilities make it ideal for PWA deployment with service workers for offline caching, meeting delivery requirements.

DISADVANTAGES

- Performance: Slightly slower than Flutter for complex animations or heavy 3D rendering due to JavaScript bridge overhead, which may impact AR/3D visualization performance.
- Native Integration: Requires native modules for some advanced features (e.g., camera for AR), increasing complexity.

SUITABILITY

React Native is highly suitable due to its strong PWA support, which aligns with the project's requirement for flexible deployment and offline caching. Its ecosystem supports rapid development of social features (real-time feed, comments) and integrates well with weather APIs. AR and 3D visualization libraries meet the wow factor needs (e.g., Fashion Time Machine, 3D avatars), though performance tuning may be needed for smooth rendering. The budget-friendly ecosystem and community support make it viable within the R5000 constraint, especially for a team familiar with JavaScript.

CONSIDERATION 2 - FLUTTER

OVERVIEW

Flutter is a Dart-based framework by Google for building cross-platform mobile applications with a single codebase. It uses the Skia graphics engine to render UI, offering high-performance, native-like experiences and supporting web deployment for PWAs.

ADVANTAGES

- AR and 3D Support: Libraries like ARKit/ARCore plugins and Flutter’s custom rendering capabilities support advanced features like virtual try-ons and 3D visualization.
- Rapid Development: Hot reload speeds up iteration, aiding development within the project timeline.
- PWA Support: Flutter’s web support enables PWA deployment with service workers for offline caching, meeting delivery requirements.

DISADVANTAGES

- Ecosystem Size: Smaller ecosystem compared to React Native, with fewer libraries for niche features like social feeds or complex API integrations, potentially increasing development effort.
- Learning Curve: Dart is less common than JavaScript, which may require additional learning for teams unfamiliar with it.
- Web Maturity: Flutter’s web support is less mature than React’s, potentially causing challenges for PWA-specific optimizations like SEO or browser compatibility.

SUITABILITY

Flutter suits "Weather to Wear" for its high-performance AR and 3D visualization, ideal for the Fashion Time Machine and 3D avatars, and consistent UI across platforms, but its smaller ecosystem and Dart’s learning curve may complicate social features and slow development. It supports PWA deployment

FRONTEND FINAL PICK - REACT NATIVE

React Native is the better choice for the "Weather to Wear" project due to its robust support for Progressive Web App (PWA) deployment, which aligns with the project’s requirement for flexible, accessible delivery across devices with offline caching. Its extensive JavaScript ecosystem and libraries, such as ViroReact for AR and Three.js for 3D avatars, facilitate rapid development of social features (e.g., real-time feed, comments) and weather API integrations.

TECHNOLOGY REQUIREMENTS

BACKEND

CONSIDERATION 1 - NODE.JS WITH JAVASCRIPT

OVERVIEW

Node.js is a JavaScript runtime built on Chrome's V8 engine, designed for scalable, event-driven backend applications. It supports asynchronous I/O, making it ideal for real-time features and API integrations, running on a single-threaded, non-blocking model.

ADVANTAGES

- Performance: Asynchronous architecture handles multiple API calls (e.g., weather APIs) and real-time social feed updates efficiently, meeting the project's performance requirements.
- Ecosystem: Vast npm ecosystem provides libraries like Express for API development, TensorFlow.js for AI recommendations, and Socket.IO for real-time social features, reducing development time.
- PWA Support: Seamless integration with frontend React Native, facilitating PWA deployment and API-driven features.

DISADVANTAGES

- Dynamic Typing: JavaScript's lack of static typing can lead to runtime errors, potentially complicating maintenance for complex features like the recommendation engine.
- Scalability Complexity: While scalable, large-scale applications may require careful management of asynchronous code to avoid callback hell or performance bottlenecks.

SUITABILITY

Node.js with JavaScript is highly suitable due to its performance for real-time features like the social feed and weather API integrations, aligning with the project's need for quick outfit generation and scalability. Its extensive ecosystem supports AI libraries for the recommendation engine and secure communication (TLS) for user data privacy. The budget-friendly tools and compatibility with React Native make it ideal, though dynamic typing may require extra testing to ensure reliability for features like the Fashion Time Machine.

CONSIDERATION 2 - NODE.JS WITH TYPESCRIPT

OVERVIEW

TypeScript is a superset of JavaScript that adds static typing, running on Node.js for backend development. It compiles to JavaScript, offering the same runtime capabilities as Node.js but with enhanced type safety and developer tooling.

ADVANTAGES

- Type Safety: Static typing reduces runtime errors, improving reliability for complex systems like the wardrobe database and AI recommendation engine, critical for accurate outfit suggestions.
- Developer Experience: Enhanced IDE support (e.g., IntelliSense) and type checking improve code maintainability and team collaboration, especially for features like 3D avatar visualization and social platform APIs.
- Ecosystem Access: Inherits Node.js's npm ecosystem, supporting libraries for AI (TensorFlow.js), real-time features (Socket.IO), and API frameworks (Express).
- Scalability: Leverages Node.js's asynchronous model, ensuring performance for API calls and social features, with TypeScript's structure aiding large-scale codebases.

DISADVANTAGES

- Build Overhead: Compiling TypeScript to JavaScript adds a build step, slightly increasing CI/CD pipeline complexity compared to plain JavaScript.

SUITABILITY

Node.js with TypeScript is well-suited for "Weather to Wear" due to its type safety, which enhances reliability for the AI recommendation engine and wardrobe database, ensuring accurate outfit suggestions and secure data handling. It supports all project requirements, including real-time social features and API integrations, while aligning with React Native for PWA deployment.

BACKEND FINAL PICK - NODE.JS WITH TYPESCRIPT

Node.js with TypeScript offers better reliability and maintainability through type safety, making it preferable for complex systems like the recommendation engine and wardrobe management, especially for larger teams or long-term maintenance. The trade-off is a slight increase in complexity.

TECHNOLOGY REQUIREMENTS

AI FRAMEWORK

CONSIDERATION 1 - TENSORFLOW.JS

OVERVIEW

TensorFlow.js is a JavaScript library for training and deploying machine learning models in the browser or Node.js environments. It supports neural networks, deep learning, and collaborative filtering, enabling complex AI tasks like personalized recommendations directly in the app.

ADVANTAGES

- Versatility: Supports neural networks and ensemble methods, ideal for building a sophisticated recommendation engine that learns from user ratings and handles layering suggestions for temperature variations.
- On-Device Training: Enables on-device model training
- Browser Compatibility: Runs in React Native and Node.js environments, ensuring seamless integration with the PWA and backend for real-time outfit suggestions.
- Pre-trained Models: Offers pre-trained models and transfer learning, reducing development time for features like style affinity or weather-based recommendations.

DISADVANTAGES

- Performance Overhead: Neural network training on devices can be resource-intensive, potentially impacting performance on low-end smartphones, requiring optimization for quick outfit generation.
- Complexity: Building and tuning deep learning models demands expertise, which may challenge teams new to ML, increasing development effort.

SUITABILITY

TensorFlow.js is ideal for "Weather to Wear" as it supports complex AI tasks like personalized outfit and layering recommendations, with on-device training reducing costs within the R5000 budget. It integrates seamlessly with React Native and Node.js for PWA and social features, and its neural network support suits the Fashion Time Machine, though performance optimization is needed for low-end devices.

CONSIDERATION 2 - SUPRISE

OVERVIEW

Surprise (Simple Python Recommendation System Engine) is a Python library designed for building and evaluating recommender systems, particularly collaborative filtering algorithms like SVD or KNN. It is typically used in server-side environments but can be integrated with Node.js via Python microservices.

ADVANTAGES

- Specialized for Recommendations: Excels at collaborative filtering, making it ideal for the project's requirement to learn user preferences via a simple rating system for outfit suggestions.
- Ease of Use: Simple API and pre-built algorithms (e.g., SVD) reduce development time, suitable for quick prototyping within the project timeline.

DISADVANTAGES

- Server-Side Limitation: Designed for Python environments, requiring a microservice architecture (e.g., Flask or FastAPI) to integrate with Node.js, adding complexity and potential latency for real-time features.
- Limited Flexibility: Focused on collaborative filtering, less suited for neural networks or advanced features like layering recommendations or 3D visualization integration, limiting support for wow factors like Fashion Time Machine.

SUITABILITY

Surprise is moderately suitable for "Weather to Wear" due to its effective collaborative filtering for outfit recommendations based on user ratings, but its server-side Python nature limits flexibility for advanced features like layering or the Fashion Time Machine, increases integration complexity with Node.js/TypeScript and React Native

AI FINAL PICK - TENSORFLOW.JS

TensorFlow.js is more versatile, supporting neural networks, on-device training, and integration with the project's JavaScript-based stack (React Native, Node.js). It excels for complex features like layering, predictive planning, and potential AR/3D visualization support, fitting the project's core and wow factor requirements.

TECHNOLOGY REQUIREMENTS

DATABASE STORAGE

CONSIDERATION 1 - MONGODB

OVERVIEW

MongoDB is a NoSQL database that stores data in flexible, JSON-like BSON documents, ideal for handling unstructured or semi-structured data. It's designed for scalability and works well with Node.js due to its JavaScript compatibility.

ADVANTAGES

- Flexible Schema: Easily accommodates diverse data types (e.g., clothing profiles with varying attributes like materials, weather suitability) and evolving requirements like social feed posts or user ratings, supporting the clothing catalog and social platform.
- Scalability: Horizontal scaling via sharding supports growing user bases and large datasets, aligning with the project's scalability requirements.
- Node.js Integration: Native compatibility with Node.js (via drivers like Mongoose) simplifies development for the TypeScript backend, streamlining API development for weather and outfit data.

DISADVANTAGES

- Complex Queries: Less suited for complex relational queries (e.g., joining user ratings with weather data for recommendations), potentially complicating AI analytics.
- Security Setup: Requires explicit configuration for encryption and partitioning (e.g., TLS, role-based access), critical for the project's user data privacy requirement.

SUITABILITY

MongoDB is highly suitable for "Weather to Wear" due to its flexible schema, which supports the dynamic clothing catalog and social platform features like outfit posts and comments. Its scalability and Node.js integration align with the project's performance and PWA requirements, and the free Atlas tier fits the budget. It supports the Fashion Time Machine's data needs (e.g., storing outfit plans) but may require additional effort for complex AI queries or ensuring data consistency for real-time features.

CONSIDERATION 2 - POSTGRESQL

OVERVIEW

PostgreSQL is a relational database management system (RDBMS) that uses structured tables with SQL for querying. It's known for strong data consistency, advanced querying capabilities, and robustness, suitable for structured data and complex relationships.

ADVANTAGES

- Advanced Queries: Powerful SQL capabilities support complex joins and aggregations, ideal for AI-driven recommendations combining weather, user preferences, and ratings (e.g., for the recommendation engine and Fashion Time Machine).
- Security: Built-in support for encryption (TLS) and fine-grained access control aligns with the project's requirement for secure data partitioning and user privacy.
- Data Consistency: ACID compliance ensures reliable transactions for user wardrobe updates, ratings, and social interactions, meeting the project's reliability requirement.

DISADVANTAGES

- Schema Rigidity: Fixed schemas require upfront design for clothing catalog attributes, potentially complicating rapid iteration for evolving features like social posts or trend integration.
- Scalability: Vertical scaling is simpler, but horizontal scaling is more complex than MongoDB, potentially challenging for large-scale growth.

SUITABILITY

PostgreSQL suits "Weather to Wear" for its strong consistency and advanced querying, ideal for the AI recommendation engine and Fashion Time Machine, with JSONB and security features meeting data and privacy needs

DATABASE STORAGE FINAL PICK - POSTGRESQL

PostgreSQL is superior for data consistency and complex queries, supporting the recommendation engine and Fashion Time Machine's analytics needs, but its schema rigidity and scaling complexity may hinder agility for evolving features.

TECHNOLOGY REQUIREMENTS

PRIMARY API FRAMEWORK

CONSIDERATION 1 - OPENWEATHERMAP

OVERVIEW

OpenWeatherMap is a widely-used weather API providing real-time, forecasted, and historical weather data from over 40,000 weather stations and global meteorological sources. It supports JSON/XML formats and offers a freemium model with extensive documentation.

ADVANTAGES

- Comprehensive Data: Provides temperature, humidity, wind speed, precipitation, and more, sufficient for outfit recommendations and layering suggestions based on weather conditions.
- Free Tier: Offers 1,000 API calls/day (60/minute limit) for free
- Global Coverage: Supports worldwide locations, ideal for the project's broad user base and the Fashion Time Machine's trip planning feature.
- Ease of Integration: Well-documented with Node.js libraries, ensuring seamless integration with the TypeScript backend and React Native frontend.

DISADVANTAGES

- Rate Limits: Free tier's 60 calls/minute limit may restrict real-time updates for high-traffic scenarios, potentially affecting emergency outfit responses.
- Accuracy: Crowdsourced data from weather stations may be less precise than AI-driven models for hyperlocal forecasts, which could impact recommendation accuracy

SUITABILITY

OpenWeatherMap is highly suitable for "Weather to Wear" due to its comprehensive data, free tier, and easy integration with Node.js/TypeScript and React Native, supporting core features like outfit recommendations and the Fashion Time Machine's predictive planning. Its global coverage and budget-friendly model align with the project's extensibility and cost requirements. However, the rate limit and potential accuracy concerns may require caching strategies (as specified in the proposal) to ensure reliability for real-time features.

CONSIDERATION 2 - STORMGLASS.IO

OVERVIEW

StormGlass.io is a global weather API specializing in high-resolution forecasts and historical data, with a focus on maritime, green energy, and agricultural applications. It aggregates data from trusted meteorological institutions and uses AI to select the best source, offering JSON responses.

ADVANTAGES

- High-Resolution Data: Provides detailed parameters (e.g., temperature, precipitation, wind, humidity) and maritime data, supporting precise outfit recommendations and layering for varying weather conditions
- AI-Driven Source Selection: Automatically chooses the best data source for accuracy, enhancing reliability for the recommendation engine and Fashion Time Machine.

DISADVANTAGES

- Maritime Focus: Emphasis on marine data may include unnecessary parameters for the project, increasing complexity and potentially affecting performance.
- Limited Free Tier: Fewer free API calls compared to OpenWeatherMap, which may limit testing or require earlier investment in paid plans, straining the budget.

SUITABILITY

StormGlass.io is moderately suitable for "Weather to Wear" due to its high-resolution, AI-driven data, which supports accurate outfit recommendations and predictive planning for the Fashion Time Machine. Its integration with Node.js/TypeScript is straightforward, and the free tier fits the budget, though it’s more limited than OpenWeatherMap’s. The maritime focus and fewer free calls make it less optimal unless marine-specific data (e.g., coastal weather) is prioritized, and caching is needed to mitigate rate limits and ensure performance.

PRIMARY API FINAL PICK - OPENWEATHERMAP

OpenWeatherMap offers broader coverage, a generous free tier, and mature integration with Node.js, making it ideal for the project’s global user base, budget constraints, and rapid development needs. Its rate limits are a concern but can be mitigated with caching, as required by the proposal.

SERVICE CONTRACTS

Outlines all the Database Design, API endpoints, request and response format