Smart Fashion Wardrobe & Outfit Generator - Project Plan

1. Project Overview

The Smart Fashion Wardrobe & Outfit Generator is a personal utility and demo project. It aims to help users manage their wardrobe digitally by uploading photos of clothes, which are automatically tagged with attributes (color, category, fabric, etc.). The core feature is an AI-driven outfit generator that suggests clothing combinations based on the user's wardrobe, occasion, and optionally, weather conditions.

2. Requirements

2.1 Functional Requirements (User Stories)

- As a user, I want to upload clothing photos so I can digitize my wardrobe.
- As a user, I want AI to automatically tag my clothes with attributes (color, style, fabric).
- As a user, I want to generate outfit suggestions for specific occasions.
- As a user, I want to view all the clothes in my wardrobe in a list or grid view.
- As a user, I want to delete items from my wardrobe.
- As a user, I want to save past outfits in my history.

2.2 Non-Functional Requirements

- Usability: Clean, minimal UI. Initially desktop-focused, mobile optimization optional later.
- Security: Password hashing, user data protection.
- Performance: App should feel fast and responsive with smooth user interaction.
- AI: Pretrained models for clothing attribute tagging and outfit suggestion.

3. Technology Stack

- Frontend: React (modern, component-based UI framework).

- Backend: Django (Python-based framework, fast for prototyping and scalable).

- Database: SQLite (lightweight, easy to set up for MVP).

- AI: Pretrained computer vision + recommendation models integrated via Python.

4. System Architecture

The system follows a monolithic architecture with clear separation between frontend, backend, and database. AI modules are integrated into backend endpoints for tagging and outfit generation.

```
[React Frontend] <--> [Django Backend + AI Integration] <--> [SQLite Database]
```

5. API Design

- POST /api/clothes/add – Upload clothing item (with image + optional tags).
- GET /api/clothes/list/{userId} – Fetch all clothes of a user.
- DELETE /api/clothes/{id} – Remove clothing item.
- POST /api/outfits/generate – Generate outfit suggestions for a given occasion.
- POST /api/outfits/save – Save an outfit user actually wore.
- GET /api/outfits/history/{userId} – Retrieve past outfits.

6. Team Roles

- Frontend Developer(s): Build React UI (upload clothes, display wardrobe, show outfits).

- Backend Developer: Set up Django models, APIs, and connect with SQLite.

- AI Developer (You): Integrate pretrained AI for tagging and outfit generation.

- Designer: Create clean, minimal UI mockups + assist with frontend components.

7. Project Milestones

- Milestone 1: Project setup + user authentication + basic clothes upload/list.
- Milestone 2: AI tagging integration + outfit generation endpoint.
- Milestone 3: Frontend polish + outfit history + occasion-based filtering.
- Milestone 4: Final integration + testing + documentation.