# Backend operations with media files

January 23, 2026

You are required to work with the code provided with the task and make the necessary modifications. Basic code structure and files are provided. You can add extra functionalities if needed, **but not allowed to modify the existing code.** For each schema, insert sample input records to demonstrate the 2 features. After completing the implementation, export all Postman API collections as a JSON file.

Organize all required project files (along with the exported json file) into a single folder, excluding the `node_modules` directory. Create a zip folder with the last 3 digits of your ID and submit it in the classroom.

## 1 StyleSync

Choosing what to wear every day shouldn't feel harder than choosing what to eat! This project addresses the everyday struggle by turning a personal wardrobe into a smart system that tracks clothing usage, understands weather conditions, and suggests outfits that actually make sense—while also helping users identify which clothes deserve a second chance and which are ready to be donated.

## 2 Schema Design

### 2.1 User Model (userModel.js)

**Fields:**

- name (String, required)
- location (String, required)
- stylePreferences (Array of Strings)["minimalist", "comfortable","formal"]
- Profile picture (single image, 10MB)

### 2.2 Clothes Model (clothesModel.js)

Represents individual clothing items.

**Fields:**

- name (String, required)

- category (String, enum: top, bottom, dress, suit etc)
- color (String, required)
- season (Array of Strings)
- occasion (Array of Strings)
- images (Array of objects: url, public_id). Max 5 images; Limit: 10MB
- wearCount (Number, default 0)
- lastWorn (Date)
- status (String, enum: active, donated, sold)

## 2.3 Accessories Model (accessoriesModel.js)

**Fields:**

- name (String, required)
- type (String, enum: jewelry, bag, shoes, hat, belt)
- color (String, required)
- compatibleWith (Array of Strings) [clothes category, can be hard coded]
- image (single object: url, public_id). Limit: 10MB
- wearCount (Number, default 0)
- lastWorn (Date)
- status (String, enum: active, donated)

## 2.4 Outfit Model (outfitModel.js)

Represents generated or saved outfit combinations.

**Fields:**

- name (String)
- clothingItems (Array of ObjectIds, ref: Clothes)
- accessories (Array of ObjectIds, ref: Accessories)
- weatherCondition (String)
- outfitImages (single object: url, public_id). Max 5 images; Limit: 10MB
- wearCount (Number, default 0)
- createdAt (Date)

## 2.5 Laundry Model (laundryModel.js)

Tracks clothing cleaning status.

**Fields:**

- items (Array of ObjectIds, ref: Clothes)
- status (String, enum: pending, washing, drying, done)
- scheduledDate (Date)

## 2.6 Weather Model (weatherModel.js)

Stores weather data for outfit recommendations.

**Fields:**

- location (String, required)
- conditions (String, enum: sunny, rainy, cloudy, cold, hot)
- date (Date)

# 3 Core Features

## 3.1 Smart Outfit Generation

The system retrieves current weather information based on the user's location and filters clothing items by season, occasion, and color compatibility. Clothing items that are marked as donated or currently in laundry [Laundry.status !== "done"] are excluded. Accessories are selected based on compatibility rules defined in the accessories schema. The generated outfit is stored with references to selected clothing items and accessories, along with optional outfit imagery. Each generation updates wear counts for the outfit itself; individual clothing items and accessories.

## 3.2 Wardrobe Analytics

Wardrobe analytics evaluates clothing usage over time using wear count and last worn data. Frequently worn items (wearCount >= 7) are identified as most-used wardrobe pieces, while items with minimal or no usage are flagged as least-used (wearCount <= 2 or lastWorn >= 1 year). Based on these insights, the system suggests donation for underutilized items without deleting them from the database.