Final Report Project: Virtual Closet Platform: A Cloud-Based, Data-Driven Solution for Sustainable Fashion

Jose De Leon

Northeastern University

College of Professional Studies

ITC6000: Database Management Systems

Professor: Andy Chen

March 30, 2025

ITC6000 Final Project Report

Title: Virtual Closet Platform: A Cloud-Based, Data-Driven Solution for Sustainable Fashion

1. Introduction

The virtual closet platform is a mobile and web-based application designed to promote sustainable fashion and reshape how users manage their wardrobe. It allows individuals to upload and organize clothing, build outfits, try on garments virtually using AR, and engage with a community through shared styles and AI-driven recommendations (Google, n.d). Users can also buy, sell, or swap second-hand items, encouraging reuse and reducing fashion waste.

The platform primarily attracts environmentally conscious users, fashion enthusiasts, and budget-conscious shoppers. It helps users extend the lifecycle of clothing while making sustainable fashion interactive, personalized, and socially engaging.

All core features are free to use, including virtual closet management, outfit sharing, and AI suggestions. Revenue is generated through small commissions on peer-to-peer transactions and optional logistics support, keeping the app accessible while supporting its operational and sustainability goals.

This project is personally inspired by my girlfriend's shift away from fast fashion during the pandemic. Witnessing her embrace second-hand clothing and promote conscious consumption sparked the idea for an app that makes sustainable fashion easy, fun, and community-driven.

2. Business Analysis

User Personas

Taylor – Fashion Influencer (Age 31)

Taylor uses the app to share daily outfit content, plan looks for events, and experiment with vintage styles. She relies on AR try-on and social features to engage her audience and discover trends.

Ashley – Budget-Conscious Student (Age 19)
Ashley browses the swap section and uses AI recommendations to build stylish looks without overspending. The virtual try-on helps her avoid ill-fitting purchases.

Jaimy – Eco-Warrior (Age 38)
Jaimy focuses on sustainability. She buys second-hand, resells rarely worn items, and donates her proceeds. Reviews and ratings help her connect with like-minded users.

Business Rules

1. Each user can upload multiple clothing items, but each item belongs to only one user.
2. Users can create multiple Looks, and each Look can include several clothing items.
3. A clothing item can be reused across many Looks and appear in multiple Transactions.
4. Transactions must involve at least one clothing item and two users (buyer and seller).
5. Users can interact with content by liking, commenting, and reviewing other users' outfits.

3. Table Design (ER Diagram)

The database consists of the following key entities:

1. **Users** (user_id as Primary Key)
    - **Foreign key**: N/A
    - **Attributes:** username, first_name, last_name, email, password, profile_picture, description, style_preferences, body_measurements, user_type.
    - **Relationships:**
        - One-to-many with Clothing_Items and Looks

        - Many-to-many with Transactions via Users_Transactions

2. **Clothing Items** (Clothing_ID as Primary Key)
    - **Foreign Key:** user_id (references Users)
    - **Attributes:** category, brand, color, size, name, season.
    - **Relationships:**
        - Belongs to one **User** but can be part of multiple Looks due to the bridge table **Clothing_Look**.
        - Can be included in multiple **Transactions** due to the bridge table **Clothing_Transaction**.
3. **Looks** (Look_ID as Primary Key)
    - **Foreign Key:** user_id (references Users)
    - **Attributes:** title, description, category.
    - **Relationships:**
        - One-to-many with Users
        - Many-to-many with Clothing_Items via Clothing_Look.
4. **Transactions** (Transaction_ID as Primary Key)
    - **Foreign key**: N/A
    - **Attributes:** payment_method, transaction_date, total_amount, transaction_type, payment_status, status
    - **Relationships:**
        - Many-to-many with Users and Clothing_Items via respective bridge tables(Users Transactions and Clothing Transaction)
5. **Users_Transactions** (Bridge Table)

- **Keys:** user_transactions_id (PK); buyer_id (FK), seller_id (FK), transaction_id (FK)
- **Attributes:** timestamp, role
- **Relationships:**
  - Has a many-to-One with **Users** and **Transactions**

6. **Clothing_Look** (Bridge Table)
   - **Keys:** look_clothing_id (PK)**;** look_id (FK), clothing_id (FK)
   - **Attributes:** date_added
   - **Relationships:**
     - Has a many-to-One with **Looks** and with **Clothing Items**
7. **Clothing_Transaction** (Bridge Table)
   - **Keys:** clothing_transaction_id (PK)**;** clothing_id (FK), transaction_id (FK)
   - **Attributes:** price, condition_at_sale, timestamp
   - **Relationships:**
     - Has a many-to-One with **Transactions** and with **Clothing Items**

Entity-Relationship Diagram: Virtual Closet Platform Database Schema

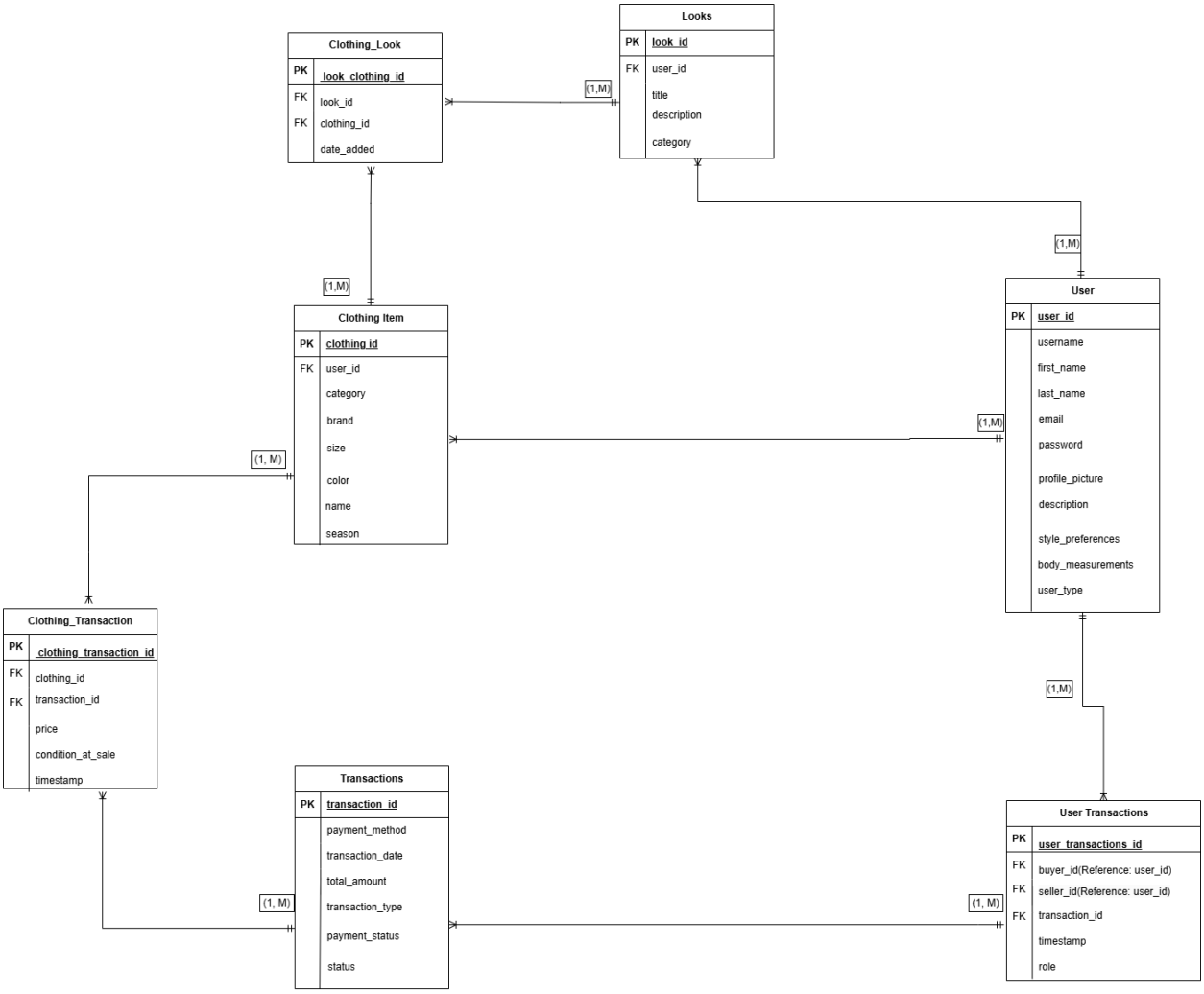

Figure 1: Entity-Relationship Diagram: Virtual Closet Platform Database Schema

4. Database Implementation (SQL + User Story)

This section demonstrates core platform functionality through SQL queries that reflect real user behavior. The following user story shows how a clothing sale is processed and analyzed within the system.

User Story: Taylor Sells an Item

Step 1: Buyer and Seller are Linked in a Transaction

Records roles and transaction data through the User_Transactions table.

Link Buyer and Seller in a Transaction

```
SELECT
    Users.user_id,
    Users.username,
    User_Transactions.transaction_id,
    User_Transactions.role,
    Transactions.total_amount
FROM Users
JOIN User_Transactions ON Users.user_id = User_Transactions.user_id
JOIN Transactions ON User_Transactions.transaction_id = Transactions.transaction_id
WHERE User_Transactions.role = 'seller';
```

Figure 2: Link Buyer and Seller in a Transaction

Step 2: Track Clothing Item Sold in the Transaction

Tracks the item sold, sale condition, and transaction date.

Identify Clothing Sold and Its Details

```
SELECT
    Clothing_Item.name,
    Clothing_Item.category,
    Clothing_Transaction.price,
    Clothing_Transaction.condition_at_sale,
    Transactions.transaction_date
FROM Clothing_Item
JOIN Clothing_Transaction ON Clothing_Item.clothing_id = Clothing_Transaction.clothing_id
JOIN Transactions ON Clothing_Transaction.transaction_id = Transactions.transaction_id
WHERE Transactions.payment_status = 'completed';
```

Figure 3: Identify Clothing Sold and Its Details

Step 3: Buyer Views Item History Before Purchasing

A buyer wants to see how many times an item has been involved in past transactions.

Check Item Resale History

```
SELECT
    Clothing_Item.clothing_id,
    Clothing_Item.name,
    COUNT(Clothing_Transaction.clothing_transaction_id) AS resale_count
FROM Clothing_Item
JOIN Clothing_Transaction
    ON Clothing_Item.clothing_id = Clothing_Transaction.clothing_id
GROUP BY Clothing_Item.clothing_id, Clothing_Item.name;
```

Figure 4: Check Item Resale History

Step 4: Seller Reviews Revenue Generated from Sales

This helps sellers track their total earnings.

View Seller's Total Revenue

```
SELECT
    Users.user_id,
    Users.username,
    SUM(Transactions.total_amount) AS total_revenue
FROM Users
JOIN User_Transactions
    ON Users.user_id = User_Transactions.user_id
JOIN Transactions
    ON User_Transactions.transaction_id = Transactions.transaction_id
WHERE User_Transactions.role = 'seller'
  AND Transactions.payment_status = 'completed'
GROUP BY Users.user_id, Users.username
ORDER BY total_revenue DESC;
```

Figure 5: View Seller's Total Revenue

These queries support backend APIs and analytics tools for resale tracking, revenue summaries, and seller dashboards. The story reflects the business rule that users can act as buyers or sellers, and that every transaction involves at least one clothing item.

Seller Dashboard Powered by SQL Queries

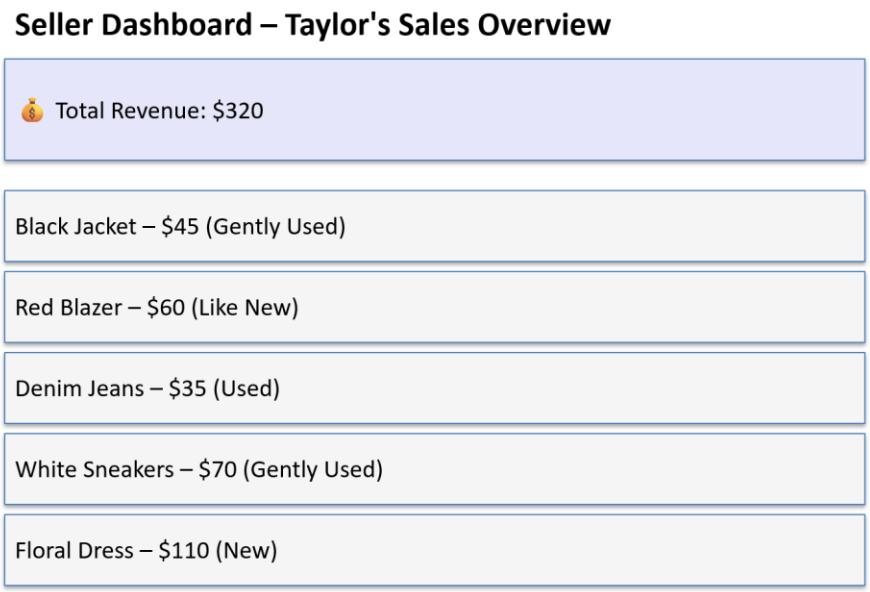

Figure 6: Seller Dashboard Powered by SQL Queries

This mockup illustrates the frontend view powered by the SQL logic in Step 4, which aggregates total revenue from completed transactions per seller.

5. Analytics, Reports, and Metrics

The platform collects valuable data from user behavior, transactions, and outfit creation. This section presents key SQL-driven reports for stakeholders such as business teams, analysts, and recommendation engines. Each query includes sample output and is designed to support data-driven decisions and engagement strategies.

Example 1: "Top Revenue-Generating Sellers"

Stakeholder: Business/Operations team
Use Case: Identify top-performing sellers for reward programs, partnerships, or promotional support.

SQL Action: This query calculates the top 10 users who have earned the most from completed sales as Sellers.

Top Revenue-Generating Sellers

```
SELECT
    Users.user_id,
    Users.username,
    SUM(Transactions.total_amount) AS total_revenue
FROM Users
JOIN User_Transactions ON Users.user_id = User_Transactions.user_id
JOIN Transactions ON User_Transactions.transaction_id = Transactions.transaction_id
WHERE User_Transactions.role = 'seller'
  AND Transactions.payment_status = 'completed'
GROUP BY Users.user_id, Users.username
ORDER BY total_revenue DESC
LIMIT 10;
```

Figure 7: Top Revenue-Generating Sellers

Sample Table: Top Revenue-Generating Sellers

user_id	username	total_revenue
101	taylorstyle	320
102	ashlee123	275
103	eco_chic	250

Table 1: Top Revenue-Generating Sellers

Business Rule Connection:

This query supports the business rule that *users can sell clothing items through transactions*, enabling the platform to aggregate total revenue per seller for analytics and performance tracking.

Example 2: "Most Resold Clothing Categories (Last 6 Months)"

Stakeholder: Data/Trends Analyst

Use Case: Spot fashion trends and category popularity for resale and recommendation algorithms.

SQL Action: This query counts how many times each clothing category was resold in the last six months.

Most Resold Clothing Categories (Last 6 Months)

```
SELECT
    Clothing_Item.category,
    COUNT(Clothing_Transaction.clothing_transaction_id) AS resale_count
FROM Clothing_Item
JOIN Clothing_Transaction ON Clothing_Item.clothing_id = Clothing_Transaction.clothing_id
JOIN Transactions ON Clothing_Transaction.transaction_id = Transactions.transaction_id
WHERE Transactions.payment_status = 'completed'
  AND Transactions.transaction_date >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY Clothing_Item.category
ORDER BY resale_count DESC;
```

Figure 8: Most Resold Clothing Categories (Last 6 Months)

Sample Table: Top Most Resold Clothing Categories

category	resale_count
Dresses	45
Jackets	40
Jeans	38

Table 2: Top Most Resold Clothing Categories

Business Rule Connection:

This aligns with the rule that *clothing items can be involved in multiple transactions*, enabling resale tracking and trend analysis across categories.

Example 3: “Average Number of Items Per Transaction by Buyer”

Stakeholder: Marketing/Engagement Team

Use Case: Identify and reward power buyers or test bundling promotions.

SQL Action: This query computes the average number of clothing items purchased per transaction by each buyer.

Average Number of Items Per Transaction by Buyer

```
SELECT
    Users.user_id,
    Users.username,
    COUNT(Clothing_Transaction.clothing_transaction_id) * 1.0 / COUNT(DISTINCT User_Transactions.transaction_id) AS avg_items_per_transaction
FROM Users
JOIN User_Transactions ON Users.user_id = User_Transactions.user_id
JOIN Clothing_Transaction ON User_Transactions.transaction_id = Clothing_Transaction.transaction_id
WHERE User_Transactions.role = 'buyer'
GROUP BY Users.user_id, Users.username
ORDER BY avg_items_per_transaction DESC;
```

Figure 9: Average Number of Items Per Transaction by Buyer

Sample Table: Average Number of Items Per Transaction by Buyer

user_id	Username	avg_items_per_transaction
102	ashlee123	3.2
104	bulkbuyer87	2.9
105	casualshopper	2.5

Table 3: Average Number of Items Per Transaction by Buyer

Business Rule Connection:

This supports the business rule that *each transaction must include at least one clothing item* and helps analyze purchasing patterns across multi-item transactions.

Example 4: “Most Versatile Clothing Items (Used in Multiple Looks)”

Stakeholder: Recommendation Engine / AI Team

Use Case: Identify items to promote as “style-it-multiple-ways” pieces.

SQL Action: This query identifies clothing items used in the highest number of different Looks.

Most Versatile Clothing Items (Used in Multiple Looks)

```sql
SELECT
    Clothing_Item.clothing_id,
    Clothing_Item.name,
    COUNT(DISTINCT Clothing_Look.look_id) AS look_count
FROM Clothing_Item
JOIN Clothing_Look ON Clothing_Item.clothing_id = Clothing_Look.clothing_id
GROUP BY Clothing_Item.clothing_id, Clothing_Item.name
ORDER BY look_count DESC;
```

Figure 10: Most Versatile Clothing Items (Used in Multiple Looks)

Sample Table: Most Versatile Clothing Items (Used in Multiple Looks)

clothing_id	name	look_count
201	Black Blazer	7
202	White Tee	6
203	High-Waisted Jeans	6

Table 4: Most Versatile Clothing Items (Used in Multiple Looks)

Business Rule Connection:

This reflects the rule that *a clothing item can be part of many Looks*, allowing the platform to identify versatile, high-utility items.

Example 5: "Average Sale Price by Item Condition"

Stakeholder: Sellers + Pricing Optimization

Use Case: Help sellers understand ideal pricing strategies based on item condition.

SQL Action: This query calculates the average price for items sold, grouped by their condition at the time of sale.

Average Sale Price by Item Condition

```
SELECT
    Clothing_Transaction.condition_at_sale,
    AVG(Clothing_Transaction.price) AS avg_price
FROM Clothing_Transaction
JOIN Transactions ON Clothing_Transaction.transaction_id = Transactions.transaction_id
WHERE Transactions.payment_status = 'completed'
GROUP BY Clothing_Transaction.condition_at_sale
ORDER BY avg_price DESC;
```

Figure 11: Average Sale Price by Item Condition

Sample Table: Average Sale Price by Item Condition

condition_at_sale	avg_price
New	60
Like New	50
Gently Used	40

Table 5: Average Sale Price by Item Condition

Business Rule Connection:

This query supports the business rule that *clothing items can be involved in multiple transactions*, and captures *condition-specific pricing data* through the condition_at_sale attribute.

These analytics empower the platform's stakeholders to make data-driven decisions, optimize user experiences, and support the app's broader mission of sustainability, personalization, and community engagement.

6. Security Concerns

The virtual closet platform handles several types of sensitive data that require thoughtful privacy and security controls. As the data expert, it's critical to identify risks and inform legal, compliance, and security teams to ensure proper protection.

Sensitive or Potentially Sensitive Data

- **Personally Identifiable Information (PII):** Fields like name, email, and profile description must be protected to prevent identity theft.

- **Style Preferences & Body Measurements:** These may reveal sensitive lifestyle or physical details and require cautious handling.
- **Transactional Metadata:** Even without processing payments directly, stored data (roles, totals, timestamps) can reveal spending behavior.
- **Item History:** Clothing_Transaction data can expose personal patterns (e.g., frequent selling), which could be misused if taken out of context.

Relevant Security Considerations

- **People-Related Risks:** Weak passwords or unattended accounts can be exploited. Mitigate via strong password policies, education, and session management.
- **Data Access:** Overly broad data access or misconfigured remote entry can lead to breaches. Limit permissions and encrypt databases.
- **Application-Level Risks:** SQL injection and unvalidated inputs are major threats. Use secure coding and input sanitization.
- **Compliance:** Regulations like GDPR may apply. Fields such as email or measurements might require user consent and privacy disclaimers.

7. Architecture

The virtual closet platform is designed using a cloud-based, n-tier architecture to ensure scalability, modularity, and high availability. Given the interactive features, such as clothing uploads, social engagement, transaction processing, and virtual try-on experiences, a cloud deployment allows the system to support a growing user base and millions of concurrent transactions per day.

Architecture Model

The virtual closet platform uses a cloud-based, 3-tier architecture designed for scalability, modularity, and high availability. It supports high user activity, such as clothing uploads, social engagement, AI recommendations, and AR try-on, by separating logic across three layers:

- **Presentation Tier**: Web and mobile front-end built with HTML, CSS, JavaScript, and mobile-native tools.
- **Application Tier**: Handles business logic, recommendations, and API requests using scalable technologies like Node.js or Python.
- **Data Tier**: Stores user, item, and transaction data in a secure relational database (e.g., PostgreSQL or MySQL) (PostgreSQL Global Development Group, n.d.).

The app is hosted on cloud infrastructure (e.g., AWS, Azure, or Google Cloud), which offers elastic scalability, global availability, automated backups, and built-in support for load balancing and compliance (Chen, 2025; Amazon Web Services, n.d.).

8. Wrap-Up and Future Considerations

This project taught me how to build a full database-driven application—from defining user needs to writing SQL that powers real features like resale tracking and dashboards. I gained experience modeling many-to-many relationships with bridge tables and creating queries that support backend logic.
I also learned how to plan for growth using a cloud-based, three-tier architecture and saw how data structure must align with both user experience and business goals. Most importantly, this project showed how tech can support impactful goals like sustainability and community engagement.

Reference

Chen, A. (2025). ITC 6000: Module 5 – Common Database Architectures [PDF file]. Department of College of Professional Studies, Northeastern University.

Amazon Web Services. (n.d.). *AWS Well-Architected Framework*. Amazon Web Services. https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html

Google. (n.d.). *How augmented reality (AR) works*. https://arvr.google.com

PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. https://www.postgresql.org/docs/