ACCESSIBLE WARDROBE

Project Summary

The **Accessible Wardrobe app** is a platform designed to help disabled individuals and busy professionals seamlessly shop, rent, and virtually try on clothes. It focuses on accessibility, user-friendly design, and leverages AR/VR for an enhanced try-on experience.

Methodology

Stage	Methodology Used	Our Updates
Requirement Gathering	Traditional Requirements Analysis	Defined features: login, profile, dress catalog, virtual try-on, rentals.
Design Phase	Prototyping	Created UI/UX mockups using Figma, Adobe XD.
Development	Agile Development	2-week sprints; built React Native app, Spring Boot backend.
Testing	Test-Driven Development (TDD)	Unit testing (JUnit, Jest), integration testing, UAT testing.
Deployment	DevOps Practices	Deployed backend on VPS with secured environment and mobile builds ready.

Literature Survey

Paper/Topic	Methodology Used	Our Improvements
Traditional E-commerce Apps	Monolithic systems with basic web apps	Introduced AR/VR for virtual try-on, enhanced accessibility for disabled users
Mobile Rental Apps	Simple catalog browsing	Added secure virtual trials and rental booking per day
Admin Portals	Basic CRUD operations	Advanced admin panel with booking and rental reports

Existing System vs Proposed System

Feature	Existing System	Proposed System (Accessible Wardrobe)
Virtual Try-On	Not Available	Available with secure image handling
Renting Clothes	Limited Options	Day-wise rental with

		delivery, return
User Experience (UX)	Moderate	Highly accessible and user-friendly
Admin Control	Basic	Full wardrobe and rental management
Payment Methods	Online only	Secure Cash on Delivery, payment gateway soon

Hardware Requirements

Component	**Minimum Requirement**
Developer Machine	8+ GB RAM, Intel i5/Ryzen 5, SSD
Testing Devices	Android & iOS Phones/Emulators
Server	2+ vCPU, 4+ GB RAM VPS

Software Requirements

Component	**Tool/Technology**
Frontend	React Native, Expo CLI
Backend	Java 17+, Spring Boot
Database	MySQL
API Testing	Postman
IDE	IntelliJ IDEA, VS Code
Version Control	Git + GitHub/GitLab
Deployment	VPS, Docker (optional)

Database Tables

users
- id (PK)
- email
- password (encrypted)
- name
- phone_number
- gender
- profile_pic (encrypted)

user_roles
- id (PK)
- user_id (FK)
- role

wardrobe
- id (PK)
- gender
- wear_type (top, bottom)
- cloth_type
- color
- size
- product_name
- rating_count
- rating_value
- price
- scratch_price
- rent_amt
- image

orders
- id (PK)
- wardrobe_id (FK)
- address
- paid_amt
- payment_status
- created_by (FK)

rentals
- id (PK)
- wardrobe_id (FK)
- address
- days
- paid_amt
- payment_status
- created_by (FK)

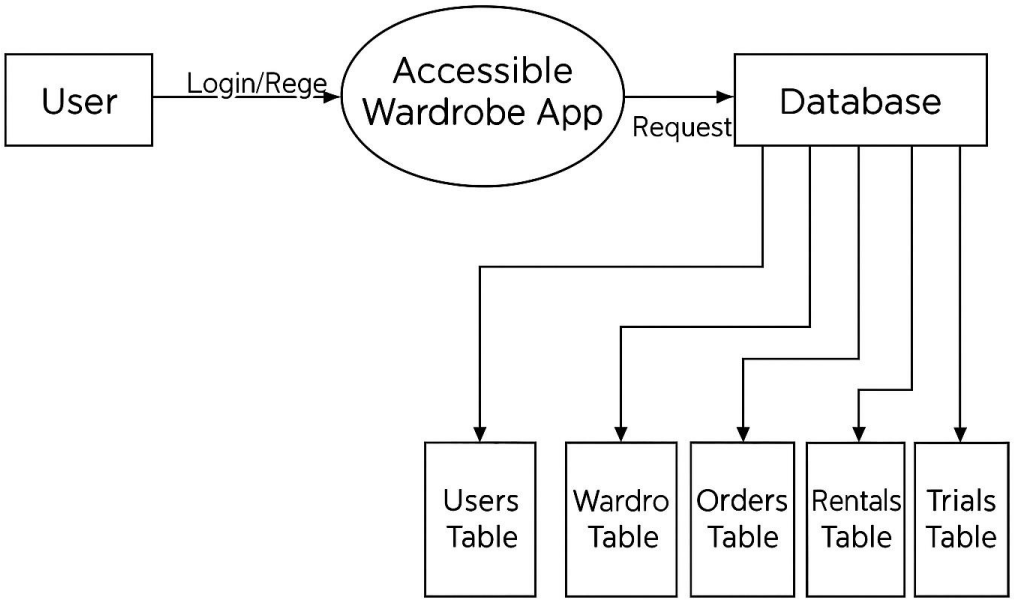
trials
- id (PK)
- profile_image (encrypted)
- dress_image
- category
- output_image (encrypted)
- created_by (FK)

Testing

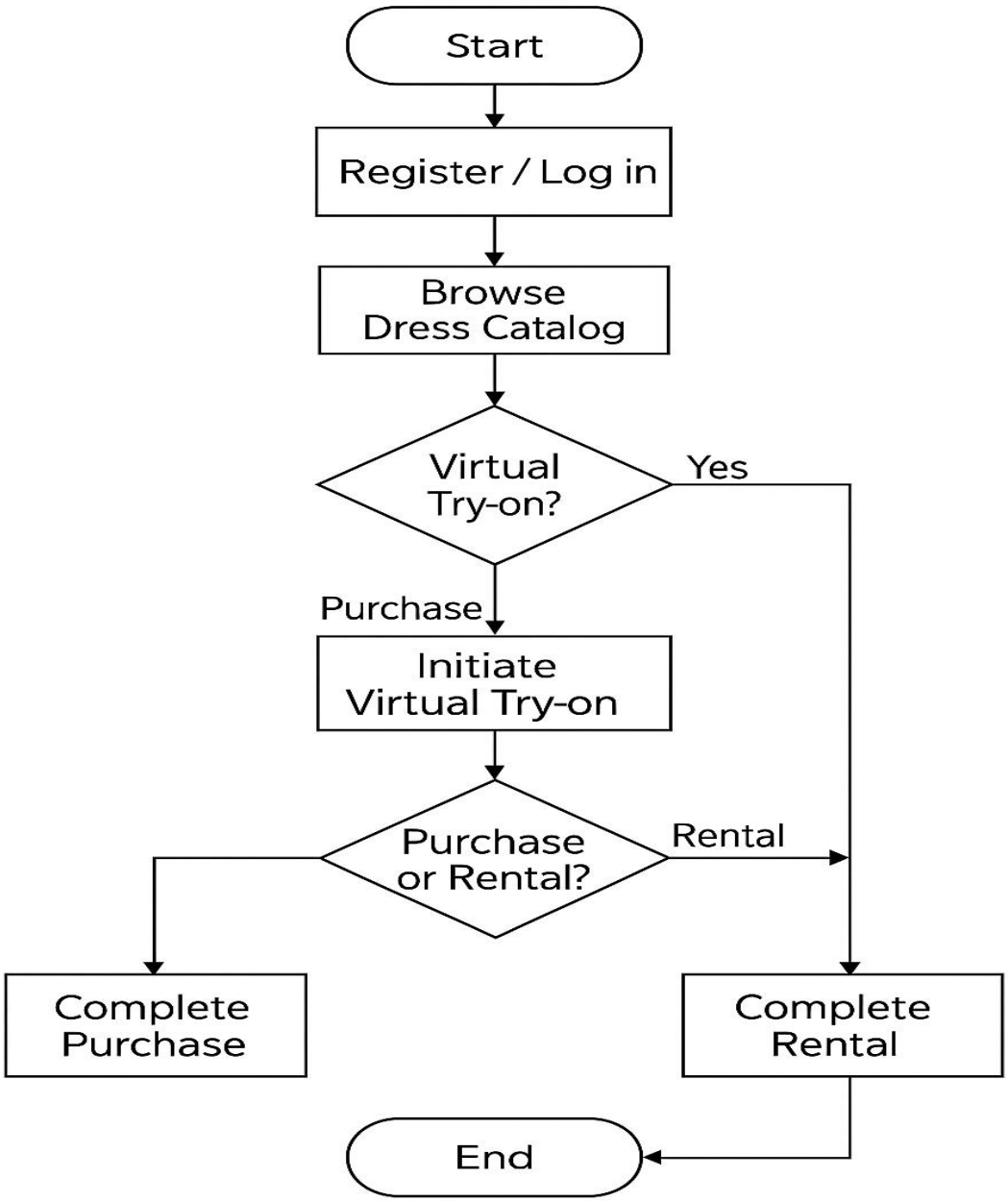
Type	Tools Used	Description
Unit Testing	JUnit (Java), Jest (React Native)	Tested each function/module

		individually
Integration Testing	Postman, Manual Testing	End-to-end API and flow testing
User Acceptance Testing (UAT)	Real device testing	Verified app usability, performance

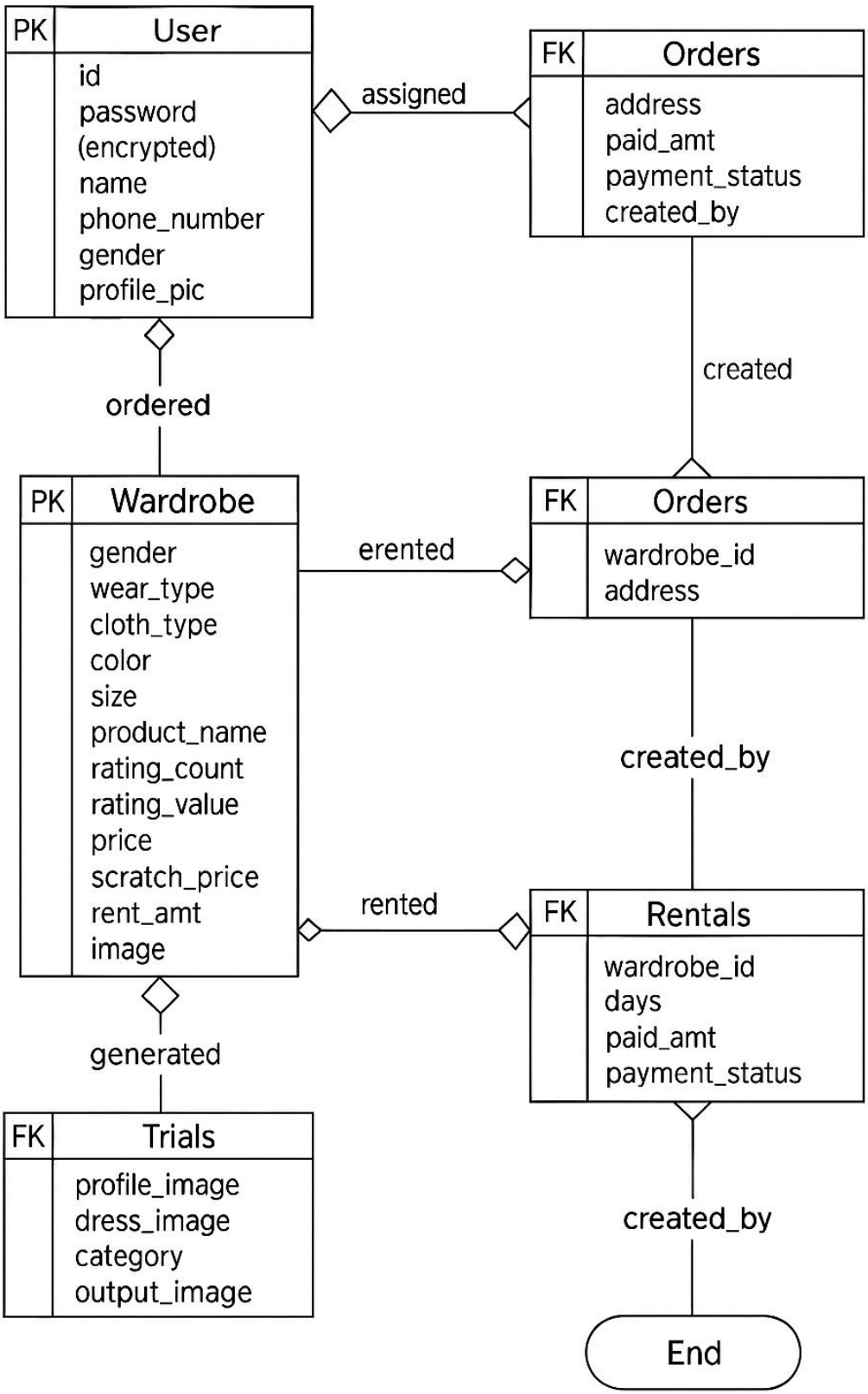
Data Flow Diagram



Workflow Diagram


Start
Register / Log in
Browse
Dress Catalog
Virtual
Try-on?
Yes
Purchase
Initiate
Virtual Try-on
Purchase
or Rental?
Rental
Complete
Purchase
Complete
Rental
End

ER Diagram


PK
User
id
password
(encrypted)
name
phone_number
gender
profile_pic
assigned
FK
Orders
address
paid_amt
payment_status
created_by
created
ordered
PK
Wardrobe
gender
wear_type
cloth_type
color
size
product_name
rating_count
rating_value
price
scratch_price
rent_amt
image
erented
FK
Orders
wardrobe_id
address
created_by
rented
FK
Rentals
wardrobe_id
days
paid_amt
payment_status
generated
FK
Trials
profile_image
dress_image
category
output_image
created_by
End

Notes:

• Secure Cash on Delivery currently available.

• Payment gateway integration (Razorpay/Stripe) planned for future release. (for report purpose only)

• End-to-end encryption on virtual trial images.

• Option for users to delete their trial images any time.