AI VIRTUAL TRY-ON PLATFORM

Production-Ready Technical Specification & Implementation Guide

Complete System Design for Android + Web Platform

Document Version:	1.0
Target Platform:	Android APK + Web Application
AI Model:	IDM-VTON (Diffusion-based)
Date Generated:	February 16, 2026
Status:	Ready for Implementation

TABLE OF CONTENTS

1. SYSTEM OVERVIEW

1.1 Vision

A production-ready virtual try-on platform delivering Kolors-level realism with diffusion-based image synthesis, sub-30 second processing, mobile and web cross-platform support, OAuth authentication, and async job architecture.

1.2 Core Features

• Users sign up/sign in using Google OAuth

• Upload front-facing photo of themselves

• Upload image of shirt/top garment

• System generates realistic composite image (10-30 seconds)

• Output is downloadable and saved to user account

• Gallery view of all generated try-ons

• Rate limiting and quota management

• Privacy-first image handling with auto-deletion

1.3 Supported Platforms

Platform	Technology	Priority
Android Mobile	Flutter APK	High
Web Application	Next.js 14	High
Backend API	FastAPI	Critical
GPU Inference	PyTorch on CUDA	Critical

2. AI MODEL STACK (PRODUCTION-LOCKED)

2.1 Complete Pipeline Architecture

The AI pipeline consists of 5 sequential stages, each handling a specific aspect of the virtual try-on process:

• Stage 1: Human Parsing (SCHP) - Segment body parts from user photo

• Stage 2: Pose Estimation (OpenPose) - Extract 18 keypoints for body alignment

• Stage 3: Garment Warping (TPS) - Align garment to user's pose

• Stage 4: Diffusion Try-On (IDM-VTON) - Generate realistic composite (8-15 seconds)

• Stage 5: Refinement (RealESRGAN) - Upscale to HD quality (1-2 seconds)

2.2 Model Specifications

A. Human Parsing: Self Correction Human Parsing (SCHP)

Property	Value
Repository	github.com/GoGoDuck912/Self-Correction-Human-Parsing
Purpose	20-class body part segmentation
Input Resolution	512 x 512 RGB
Output	Segmentation mask (body, arms, torso, etc.)
GPU Memory	~2GB
Inference Time	~200ms
Why Chosen	Production-stable, handles occlusions well

B. Pose Estimation: OpenPose Body Module

Property	Value
Repository	github.com/CMU-Perceptual-Computing-Lab/openpose
Purpose	18-keypoint body pose detection
Input Resolution	368 x 368 RGB
Output	18 (x, y, confidence) keypoints
GPU Memory	~1.5GB
Inference Time	~150ms
Why Chosen	Industry standard, reliable, no hands/face needed

C. Virtual Try-On Core: IDM-VTON (Critical Component)

Property	Value
Repository	github.com/yisol/IDM-VTON
Purpose	Diffusion-based virtual try-on synthesis
Architecture	Stable Diffusion 1.5 + ControlNet + Garment Encoder
Input Resolution	512 x 768 RGB (person + garment)
Output	512 x 768 try-on result
GPU Memory	~12GB (fp16 precision)
Inference Time	8-15 seconds (20 diffusion steps)
Why Chosen	Best open-source diffusion try-on, handles complex patterns

D. Refinement: RealESRGAN

Property	Value
Repository	github.com/xinntao/Real-ESRGAN
Purpose	HD upscaling and artifact removal
Model Variant	RealESRGAN_x2plus
Input	512 x 768 try-on output
Output	1024 x 1536 final result
GPU Memory	~3GB
Inference Time	1-2 seconds
Why Chosen	Industry-standard, pre-trained on faces/bodies

3. SYSTEM ARCHITECTURE

3.1 High-Level Component Diagram

The system follows a microservices architecture with clear separation of concerns:

• CLIENT TIER: Android APK (Flutter) + Web App (Next.js)

• API GATEWAY: FastAPI with rate limiting and JWT validation

• BACKEND SERVICES: Auth API, Job API, Result API

• DATABASE: PostgreSQL (users, jobs, results, quotas)

• JOB QUEUE: Redis for async processing

• GPU INFERENCE SERVICE: Isolated FastAPI service running AI pipeline

• OBJECT STORAGE: AWS S3 / Cloudflare R2 for images

3.2 Database Schema

Users Table:

• id (UUID, primary key)

• google_id (VARCHAR, unique, indexed)

• email (VARCHAR, indexed)

• name (VARCHAR)

• profile_picture_url (TEXT)

• plan (VARCHAR: free/pro/enterprise)

• credits_remaining (INT, default 5)

• created_at, last_login (TIMESTAMP)

Jobs Table:

• id (UUID, primary key)

• user_id (UUID, foreign key to users)

• status (VARCHAR: pending/processing/completed/failed)

• user_image_url, garment_image_url (TEXT)

• result_image_url (TEXT, nullable)

• error_message (TEXT, nullable)

• processing_time_ms (INT)

• created_at, started_at, completed_at (TIMESTAMP)

Results Table:

• id (UUID, primary key)

• job_id, user_id (UUID, foreign keys)

• image_url, thumbnail_url (TEXT)

• is_favorite (BOOLEAN, default false)

• deleted_at (TIMESTAMP, for soft delete)

• created_at (TIMESTAMP)

3.3 API Endpoints

Method	Endpoint	Purpose
POST	/api/v1/auth/google/login	Google OAuth login
POST	/api/v1/auth/refresh	Refresh JWT token
POST	/api/v1/jobs/create	Submit try-on job
GET	/api/v1/jobs/{id}/status	Poll job status
GET	/api/v1/jobs/{id}/result	Fetch completed result
DELETE	/api/v1/jobs/{id}	Cancel/delete job
GET	/api/v1/results	List user's gallery
POST	/api/v1/results/{id}/favorite	Mark as favorite
DELETE	/api/v1/results/{id}	Delete result
GET	/api/v1/user/profile	Get user info
GET	/api/v1/user/quota	Check usage limits

4. MVP DEVELOPMENT ROADMAP

Phase 1: Core MVP (Weeks 1-8)

Goal: Ship image-only try-on with shirts, focusing on core functionality and user experience.

Week	Focus Area	Time	Deliverable
1-2	Infrastructure Setup	10-15h	GPU server, DB, S3, domain ready
3-4	AI Pipeline	40-50h	End-to-end pipeline script working
5	Backend API	25-30h	FastAPI with OAuth, job management
6	GPU Service	20-25h	Queue consumer, pipeline orchestration
7	Web App	30-35h	Next.js UI with upload and results
8	Android App	35-40h	Flutter APK with full functionality

Phase 2: Quality & Optimization (Weeks 9-12)

• Week 9-10: Realism Improvements - Fine-tune IDM-VTON, add face preservation

• Week 11: Performance Optimization - Model quantization, caching, target <15s

• Week 12: UX Polish - Tutorial, quality validation, social sharing, dark mode

Phase 3: Launch Prep (Weeks 13-16)

• Week 13-14: Testing & QA - Unit tests, load testing, security audit

• Week 15: Play Store Submission - Privacy policy, app listing, content rating

• Week 16: Soft Launch - 100 beta users, collect feedback, fix critical bugs

5. TECH STACK SUMMARY

5.1 Frontend Technologies

Component	Technology	Rationale
Web Framework	Next.js 14 (App Router)	Modern React, SSR, TypeScript support
Web Styling	Tailwind CSS	Rapid development, responsive design
Web State	Zustand / React Query	Lightweight, API caching
Mobile Framework	Flutter 3.x	Cross-platform, native performance
Mobile Language	Dart	Type-safe, productive
Mobile State	Riverpod / Provider	Reactive state management
Auth (Both)	Google Sign-In	Trusted, one-click authentication

5.2 Backend Technologies

Component	Technology	Rationale
API Framework	FastAPI 0.108+	High performance, auto docs, async
Language	Python 3.10	AI ecosystem, productivity
ORM	SQLAlchemy 2.0	Robust, type-safe database access
Validation	Pydantic V2	Data validation, serialization
Primary DB	PostgreSQL 15	Reliable, ACID compliant
Cache/Queue	Redis 7	Fast in-memory data structure store
Object Storage	AWS S3 / Cloudflare R2	Scalable, cost-effective
CDN	CloudFront / Cloudflare	Fast content delivery

5.3 AI/ML Stack

Component	Technology	Notes
Deep Learning	PyTorch 2.0+	Primary framework
Diffusion Models	Diffusers (HuggingFace)	IDM-VTON implementation
Computer Vision	OpenCV, Pillow	Image preprocessing
Compute	CUDA 11.8, cuDNN 8.9	GPU acceleration
Optimization	Mixed precision (fp16)	40% memory reduction
Models	SCHP, OpenPose, IDM-VTON, RealESRGAN	Open-source stack

6. COST BREAKDOWN & INFRASTRUCTURE

6.1 GPU Server Options

Provider	GPU	Cost/Hour	8hr/day	24/7	Recommendation
Vast.ai	RTX 3090	$0.30-0.40	$96/mo	$288/mo	Best for MVP
RunPod	RTX 3090	$0.50-0.60	$144/mo	$432/mo	Good reliability
AWS EC2	G5.xlarge	$1.00-1.50	$300/mo	$900/mo	Enterprise scale

6.2 Monthly Cost Estimate (MVP Phase)

Service	Option	Monthly Cost
GPU Server	Vast.ai RTX 3090 (8hr/day)	$96 - $144
Backend Server	EC2 t3.medium / DigitalOcean	$24 - $30
Database	RDS t3.micro + Redis	$30
Or Managed DB	Supabase (PostgreSQL + Auth)	$25
Object Storage	S3 (1TB) / Cloudflare R2	$15 - $40
CDN	CloudFront / Cloudflare Free	$0 - $20
Monitoring	Sentry Free / DataDog	$0 - $15
Domain + SSL	Domain + Let's Encrypt	$1
TOTAL (8hr GPU)		**$170 - $260/mo**
TOTAL (24/7 GPU)		**$350 - $450/mo**

6.3 Cost Optimization Strategies

• Run GPU only during peak hours (8-12 hours/day initially)

• Auto-delete user uploads after 7 days, results after 30 days

• Use Cloudflare CDN (free tier) instead of paid options

• Compress images to WebP format (30-50% size reduction)

• Implement aggressive rate limiting for free users

• Batch similar jobs together to maximize GPU utilization

• Cache common garment images to avoid re-processing

7. SECURITY, PRIVACY & LEGAL COMPLIANCE

7.1 Data Privacy (GDPR/CCPA Compliant)

• All images encrypted at rest (S3 SSE-KMS) and in transit (HTTPS only)

• User uploads auto-deleted after 7 days (configurable)

• Generated results stored for 30 days by default

• Users can delete their data anytime via UI

• No third-party data sharing without explicit consent

• No model training on user data (initially)

• CDN uses signed URLs with expiration

• Access restricted to authenticated user only

7.2 Content Moderation (Critical for Safety)

• Use AWS Rekognition or Google Vision API for content detection

• Reject explicit content, violence, hate symbols automatically

• Special protection: Reject images of minors (under 18)

• Maximum image size: 10MB

• Allowed formats: JPEG, PNG, WebP only

• Minimum resolution: 512x512 pixels

• Maximum resolution: 2048x2048 pixels

7.3 Rate Limiting Strategy

User Type	Per Minute	Per Hour	Per Day	Per Month
Free User	1 request	5 requests	5 requests	20 requests
Pro User	3 requests	30 requests	50 requests	500 requests
Enterprise	10 requests	200 requests	Unlimited	Unlimited

7.4 Google Play Store Compliance Checklist

✓ Privacy Policy URL (must be publicly accessible)

✓ Data Safety form completed (declare what data you collect)

✓ Target API level 33+ (Android 13 or higher)

✓ Content rating questionnaire (ESRB/PEGI)

✓ Minimal permissions (INTERNET, READ/WRITE_EXTERNAL_STORAGE, CAMERA)

✓ In-app disclosure before requesting sensitive permissions

✓ No deceptive behavior or misleading content

✓ No sexual or inappropriate content

✓ Age gate: 18+ only (Terms of Service)

7.5 Terms of Service - Key Points

Acceptable Use:

✓ Personal use and experimentation

✓ E-commerce product visualization

✓ Fashion design iteration

✗ Deepfakes or impersonation

✗ Content involving minors

✗ Illegal or harmful content

✗ Harassment or abuse

✗ Commercial resale without proper licensing

Intellectual Property:

• User retains rights to uploaded images

• Generated images are user's property

• User cannot claim AI model ownership

• Must credit app if used commercially (optional requirement)

• Platform reserves right to use anonymized data for improvements

8. RISKS & MITIGATION STRATEGIES

8.1 Technical Risks

Risk	Impact	Probability	Mitigation Strategy
Slow inference (>30s)	High	Medium	Use fp16, reduce steps, show progress, manage expectations
GPU server downtime	Critical	Medium	Managed service (RunPod), queue retry, backup GPU
Poor quality output	High	Medium	Extensive testing, input guidelines, quality validation
S3 cost explosion	Medium	Low	Auto-delete old files, use R2, compress images, monitor spend
Database overload	Medium	Low	Proper indexing, archive old jobs, read replicas if needed

8.2 Business Risks

Risk	Impact	Mitigation Strategy
High GPU costs eat margins	Critical	Use Vast.ai, peak-hours only, charge Pro users, strict limits
Google Play rejection	High	Follow policies strictly, beta test first, have privacy policy
User abuse (spam/bots)	Medium	Email verification, rate limiting, CAPTCHA, content moderation
Model copyright issues	Medium	Use Apache 2.0/MIT models, check IDM-VTON license, attribute
Competition from big players	Medium	Focus on niche, better UX, faster iteration, community

8.3 Ethical Risks

Risk	Mitigation Strategy
Deepfake misuse	Watermark outputs, prohibit impersonation in ToS, reject celebrity faces
Body image issues	No body shape editing (MVP), positive messaging, mental health resources
Protection of minors	18+ age gate, face detection to reject young faces, clear ToS
Copyright infringement	Watermark brand garments, 'Personal use' disclaimer, DMCA process

9. CONCRETE IMPLEMENTATION STEPS

Week 1: Day 1 - Environment Setup (8 hours)

- Hour 1-2: Create Vast.ai account, launch RTX 3090 instance, SSH setup
- Hour 3-4: Clone repositories (SCHP, OpenPose, IDM-VTON, RealESRGAN)
- Hour 5-6: Install dependencies (PyTorch, CUDA 11.8, diffusers, OpenCV)
- Hour 7-8: Download pre-trained model weights from HuggingFace/GitHub

Week 1: Day 2 - Model Testing (8 hours)

- Hour 1-2: Test SCHP on sample images, verify segmentation masks
- Hour 3-4: Test OpenPose, ensure 18 keypoints detected correctly
- Hour 5-6: Test IDM-VTON with sample person + garment images
- Hour 7-8: Test RealESRGAN upscaling, verify quality improvement

Week 1: Day 3 - Pipeline Integration (8 hours)

- Create VirtualTryonPipeline class that chains all 5 stages
- Implement TPS warping module for garment alignment
- Add error handling and logging at each stage
- Test end-to-end on 10 diverse sample images
- Optimize for memory usage (fp16, model offloading)
- Benchmark inference time (target: <20 seconds)

Week 1: Days 4-5 - Backend API (16 hours)

- Set up FastAPI project structure with routers and services
- Implement Google OAuth 2.0 login flow with JWT tokens
- Create PostgreSQL database schema and migrations
- Set up Redis for job queue and caching
- Implement image upload to S3 with presigned URLs
- Build job creation, status polling, and result endpoints
- Add rate limiting middleware using Redis
- Write unit tests for critical endpoints

9.1 Critical Code Components

Backend Project Structure:

backend/app/ contains main.py (FastAPI app), models/ (DB schemas), routers/ (API endpoints), services/ (business logic), utils/ (JWT, rate limiting). Separate gpu_inference/ directory contains pipeline.py and worker.py for GPU processing.

Mobile App Flow:

1. User opens app → Google Sign-In screen
2. After auth → Upload screen (take photo or select from gallery)
3. Upload garment image → Submit job button
4. Show progress indicator (poll /api/v1/jobs/{id}/status every 2 seconds)
5. Display result with download and share options
6. Gallery tab shows all previous try-ons with delete option

10. SUCCESS METRICS & MONITORING

10.1 MVP Launch Targets (Month 1)

Metric	Target	Measurement Method
User Signups	100 users	Database count
Active Users	50 users	Users with ≥1 generation
Total Generations	200 images	Jobs completed count
Avg Inference Time	<20 seconds	Job processing_time_ms field
Success Rate	95%+	(completed jobs / total jobs) × 100
Error Rate	<5%	(failed jobs / total jobs) × 100
User Satisfaction	4+ stars	In-app rating prompt
Regeneration Rate	<10%	Track duplicate user+garment pairs

10.2 Growth Targets (Month 3)

Metric	Target
Total Signups	1,000 users
Monthly Active Users	300 users
Total Generations/Month	2,000 images
Pro Subscribers	20 users
Monthly Recurring Revenue	$200
Avg Inference Time	<15 seconds
Success Rate	98%+
App Store Rating	4.5+ stars
Play Store Downloads	500+ installs
Cost Per Active User	<$2.00

10.3 Monitoring & Analytics Setup

• Error Tracking: Sentry for real-time error alerts and stack traces

• Performance: DataDog or New Relic for API latency and GPU utilization

• User Analytics: Mixpanel or Amplitude for user flows and drop-off points

• Business Metrics: Custom dashboard (Grafana) for revenue and usage

• Infrastructure: CloudWatch/Prometheus for server health and costs

• A/B Testing: Firebase Remote Config for feature experiments

FINAL RECOMMENDATIONS & NEXT STEPS

Immediate Action Items (Week 1):

✓ Create Vast.ai account and launch RTX 3090 GPU instance

✓ Clone and test IDM-VTON model locally with sample images

✓ Set up PostgreSQL and Redis databases (Docker is fine initially)

✓ Build minimal pipeline.py that runs all 5 stages end-to-end

✓ Measure actual inference time on your hardware (<20s target)

What to Avoid:

✗ Training models from scratch (use pre-trained weights)

✗ Video try-on initially (too complex, focus on images)

✗ Real-time inference (async queue is more cost-effective)

✗ Over-engineering auth (Google OAuth is sufficient)

✗ Multiple garment types in MVP (shirts/tops only initially)

✗ Perfect quality on first iteration (iterate based on user feedback)

Must-Have Before Launch:

✓ Content moderation (reject explicit/harmful content)

✓ Privacy policy page (required for Play Store)

✓ Rate limiting (prevent abuse and cost overruns)

✓ Clear error messages (help users understand what went wrong)

✓ Progress indicators (users need to see 10-20s wait time)

✓ Auto-deletion of old images (privacy and cost control)

✓ Basic analytics (understand how users interact with the app)

MVP Scope Boundaries:

In Scope (Ship This)	Out of Scope (Later)
Shirts/tops only	Pants, dresses, jackets
Front-facing photos	Side poses, back views
Single garment per image	Multiple garments, full outfits

Image output only	Video try-on, AR live camera
Android + Web	iOS app
Google OAuth only	Email/password, social logins
Basic gallery	Advanced editing, filters
Manual upload	Direct integration with Shopify/WooCommerce

Start with Day 1 tasks: Set up GPU server, clone repositories, test models.

This specification pro