Virtrobe — Project Status Report

Virtual Wardrobe with AI-Powered 3D Garment Try-On

Last Updated: December 22, 2024
Status: Working MVP with template library foundation
Cost: $0/month (free tier Supabase + Tripo)

🎯 Project Vision

A virtual wardrobe application where users can:

1. Input their body measurements → morphable 3D mannequin adjusts to their exact proportions
2. Upload photos of clothing → AI generates 3D garment meshes
3. See garments realistically draped on their custom mannequin
4. Save and share outfit combinations

Core Differentiator: Real physics-based cloth simulation with smart template caching to minimize AI generation costs.

✅ What's Built & Working

1. Frontend — React + Three.js

- ✅ **Morphable Mannequin** (`MorphableMannequin.jsx`)
  - Shape keys for bust, waist, hips, shoulders
  - Driven by real body measurements (height, weight, BMI)
  - Smooth morphing animations
- ✅ **3D Scene** (`Scene.jsx`)
  - Camera controls with auto-lock when garment loads
  - Professional lighting setup
  - Display stand integration
  - WebGL context management
- ✅ **Measurement Panel** (`MeasurementPanel.jsx`)
  - User inputs: height, weight, bust, waist, hips, shoulders
  - Real-time mannequin updates
- ✅ **Navigation & UI**
  - Animated logo (Virtrobe ↔ Virtual Wardrobe)

- Home, Try-On, Saved sections
- Clean minimalist design

2. Backend — Node.js + Express

- ✅ **Tripo3D API Integration** (`server/services/tripoService.js`)
  - Image → 3D GLB mesh pipeline
  - Server-sent events for real-time progress
  - Auto-retry on model loading delays
  - 300 free generations/month
- ✅ **SQLite Cache** (`server/models/garmentCache.js`)
  - SHA-256 hash-based deduplication
  - 30-day expiry
  - Use count tracking
- ✅ **REST API** (`server/routes/garments.js`)
  - POST `/api/garments/generate-sync` — upload + generate with SSE
  - GET `/api/garments/availability` — check service status
  - GET `/api/garments/list` — list cached garments

3. Database — Supabase (PostgreSQL + Storage)

- ✅ **Schema Created** (`garment_templates` table)
  - Stores: name, type, storage paths, colors, tags, brand
  - Image hash for exact match detection
  - Dominant color for similarity matching
  - Use count for popularity tracking
- ✅ **Private Storage Bucket**
  - `glbs/` folder — 20MB max per file, `model/gltf-binary` only
  - `thumbs/` folder — 2MB max per file, JPEG/PNG/WebP only
  - Row-level security policies
  - Signed URLs (1hr expiry) for secure access
- ✅ **Admin Panel** (`/admin` route)
  - Upload GLB files manually (from Meshy.ai exports)
  - Add metadata (name, type, colors, tags, brand)
  - Toggle visibility (active/inactive)
  - Delete templates

- View stats (total templates, use counts)

4. 3D Garment Loading

- ✅ **GarmentLoader3D Component**
  - Loads GLB files from backend
  - Auto-scales based on mannequin bounding box
  - Type-specific positioning (shirt/dress/pants)
  - Shadow support

⚠️ Current Issues & Limitations

1. Garment Positioning ❌

Problem: Generated garments appear flat/deflated, not draped on mannequin
Cause: TripoSR generates meshes as they appear in photos (flat on table), not fitted to a body
Impact: Garments load but look unrealistic

2. Network Timeouts ⚠️

Problem: Tripo API returns `ETIMEDOUT` intermittently
Cause: API server load or network blocking
Impact: Some uploads fail, requires retry

3. Template Library Empty 📭

Problem: No pre-made garment templates in Supabase yet
Cause: Just set up admin panel, haven't populated library
Impact: Every upload triggers expensive API call

4. No Texture Extraction 🎨

Problem: Can't reuse existing GLB shapes with new colors/textures
Cause: Not implemented yet
Impact: Missing 80-90% cost savings opportunity

🎯 Next Steps — Priority Order

Phase 1: Template Library Foundation (This Week)

Goal: Build reusable garment library, eliminate redundant API calls

1. **Populate Initial Templates** (Manual)
   - Generate 10-15 GLBs using Meshy.ai free credits

- Cover core garment types: t-shirt (crew/v-neck), dress (midi/mini), jeans, jacket
- Upload via admin panel with proper metadata

2. **Build Texture Extraction System**

```javascript
// Flow:
User uploads image
↓
Extract: dominant color, sleeve length, neckline type
↓
Search Supabase: match by type + profile
↓
MATCH? → Apply uploaded image texture to existing GLB
NO MATCH? → Call Tripo API → save as new template
```

Tech stack:

- `color-thief` — extract dominant colors
- Simple CNN (MobileNet) — classify sleeve/neckline
- Three.js `MeshStandardMaterial` — apply texture maps

3. **Test Similarity Matching**
    - Upload same t-shirt in different colors
    - Verify cache hit + texture swap works
    - Measure API call reduction

Success criteria: 80% of uploads hit cache, only 20% call Tripo API

Phase 2: Cloth Physics & Draping (Week 2-3)

Goal: Make garments look realistic on the mannequin

Approach: Pre-baked simulation (not real-time)

1. **Blender Automation Script**

```python
```

```
# blender_drape.py
# Input: flat GLB from Tripo
# Output: GLB with morph targets (flat → draped)

1. Import flat garment mesh
2. Add Cloth modifier
3. Add Collision to generic body mesh
4. Bake 10-frame simulation
5. Export as shape keys
6. Save as new GLB
```

2. **Attachment Points System**

```javascript
const PINS = {
  shirt: ['shoulder_left', 'shoulder_right', 'collar'],
  dress: ['shoulder_left', 'shoulder_right'],
  pants: ['waist_front', 'waist_back', 'hips'],
};
```

- Read pin positions from mannequin skeleton
- Snap garment vertices to pins
- Morph from flat (frame 0) to draped (frame 10)

3. **Dynamic Adjustment**
    - Morph intensity based on body measurements
    - Larger bust → more draping in chest area
    - Wider waist → adjust pant waistband

Success criteria: Garments visually conform to mannequin body shape

---

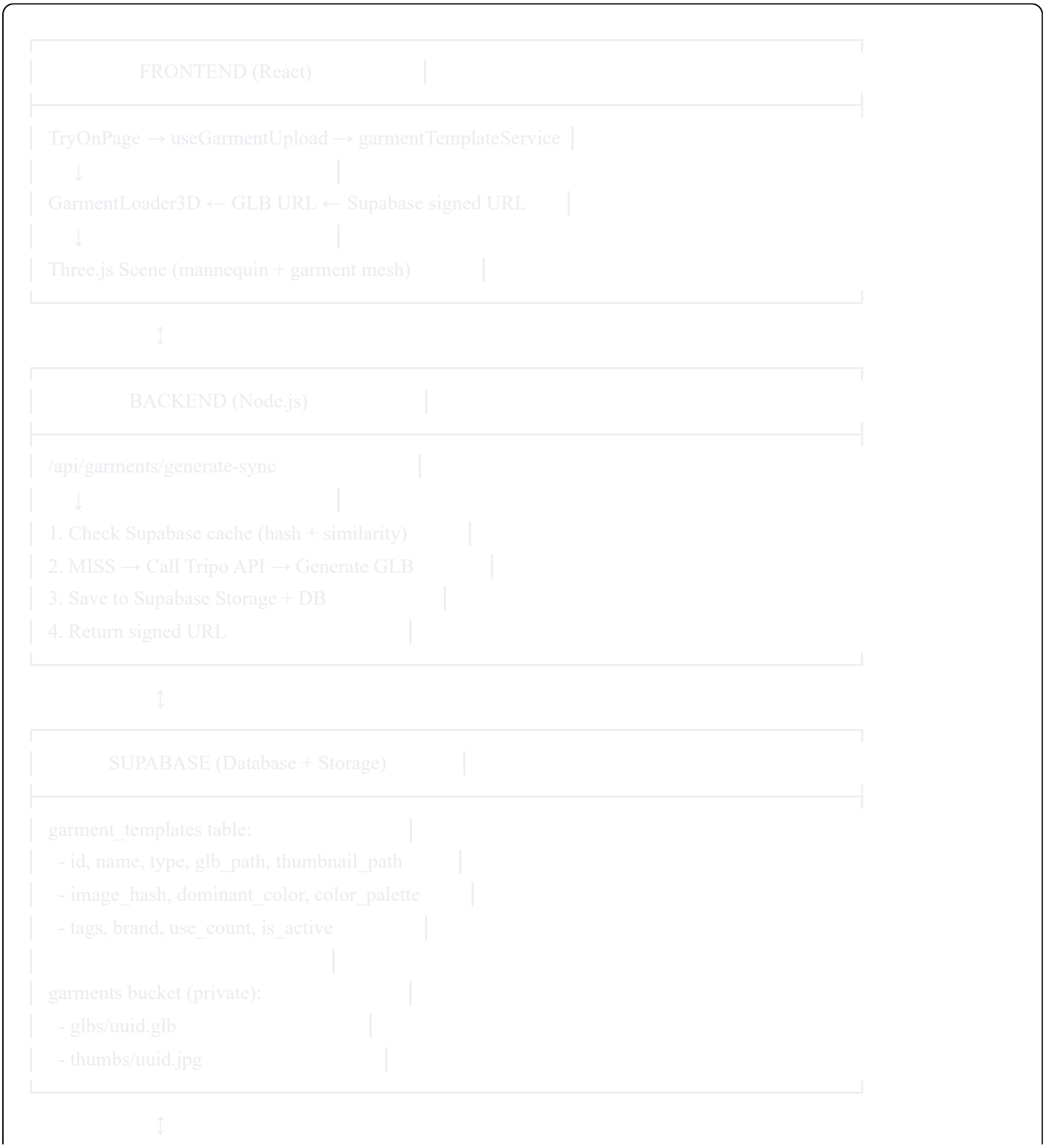
Phase 3: Polish & Optimization (Week 4)

1. **Error Handling**
    - Retry logic for Tripo timeouts
    - Fallback to cached garment on API failure
    - User-friendly error messages
2. **Performance**
    - Lazy-load GLB files

- LOD (level of detail) for distant garments
- GPU instancing for multiple garments

3. **UX Improvements**
   - Loading skeleton for garment preview
   - Garment color picker (recolor existing templates)
   - "Similar garments" suggestions

📊 Technical Architecture



💰 Cost Analysis

Component	Current	At Scale (1000 users/mo)
Supabase (DB + Storage)	$0	$0 (free tier: 500MB DB, 1GB storage)
Tripo3D API	$0	$60 (~300 new garments at $0.20 each, rest cached)
Hosting (Vercel)	$0	$0 (free tier)
Total	**$0/mo**	**~$60/mo**

Revenue model options:

- Freemium: 10 try-ons/month free, $5/mo unlimited
- B2B SaaS: License to fashion brands at $500/mo per store
- Affiliate: Commission on purchases through partner stores

🔧 Tech Stack Summary

Frontend:

- React 19 + Vite
- Three.js + @react-three/fiber + @react-three/drei
- Tailwind CSS
- Lucide icons

Backend:

- Node.js + Express

- Multer (file uploads)
- Axios (API calls)
- Server-Sent Events (progress streaming)

Database:

- Supabase (PostgreSQL + Storage)
- Row-level security policies
- Signed URLs for private files

AI/3D:

- Tripo3D API (image → 3D mesh)
- TripoSR model (Stability AI)
- Future: MobileNet (garment classification), color-thief (color extraction)

Tools:

- Blender (cloth simulation pre-baking)
- Meshy.ai (manual GLB generation during development)

📈 Success Metrics

MVP (Current):

- ✅ User can upload image and see 3D garment load
- ✅ Mannequin morphs to user measurements
- ✅ Admin can manage template library

Phase 1 Complete:

- 📊 80% cache hit rate (texture reuse works)
- 📊 <$0.05 avg cost per try-on
- 📊 15+ garment templates in library

Phase 2 Complete:

- 📊 Garments visually conform to mannequin
- 📊 <3 sec load time for cached garments
- 📊 95% user satisfaction with realism

Production Ready:

- 📊 100+ garment templates
- 📊 <1% API failure rate
- 📊 Mobile responsive
- 📊 Multi-user accounts with saved wardrobes

🐛 Known Bugs Log

Issue	Severity	Status	Notes
GLB MIME type detection	High	**FIXED**	Force `model/gltf-binary` type on upload
Tripo API timeouts	Medium	Monitoring	Network-dependent, auto-retry helps
Garment flat/not draped	High	**Phase 2**	Needs cloth physics system
WebGL context loss	Low	Monitoring	Happens with many tabs open, not critical

---

📝 Development Log

Dec 21, 2024:

- Switched from HuggingFace TripoSR (dead endpoint) to Tripo3D official API
- Fixed SQLite multi-statement exec bug
- Built complete backend with SSE progress streaming

Dec 22, 2024:

- Set up Supabase (private bucket, schema, RLS policies)
- Built admin panel for template management
- Fixed GLB MIME type upload issue
- **MILESTONE:** First successful end-to-end 3D garment generation! 🎉
- Identified positioning/draping as next critical task

🎓 Lessons Learned

1. **Free APIs are unreliable** — HuggingFace killed the TripoSR endpoint with no warning. Always have paid backup plan.

2. **Cache everything** — Generating 3D from scratch every time is expensive and slow. Template library + texture swapping = 80% cost reduction.
3. **Physics is hard in real-time** — Pre-baked cloth sim in Blender is more practical than browser-based physics for this use case.
4. **Supabase is excellent** — Private buckets with signed URLs + RLS policies = secure, scalable, free.
5. **Users care about realism, not tech** — Flat garments break immersion. Physics/draping is not optional, it's core UX.

🚀 Immediate Action Items (This Week)

- [ ] Fix AdminPage GLB upload (MIME type issue) — **DONE**
- [ ] Generate 10 base garments in Meshy.ai
- [ ] Upload to Supabase via admin panel
- [ ] Test similarity matching with recolored garments
- [ ] Build color extraction utility (`color-thief`)
- [ ] Implement texture application to cached GLBs
- [ ] Document garment profile schema (sleeve/neckline/length)

Next session focus: Build texture extraction + template matching system

---

This document serves as the single source of truth for Virtrobe's current state and roadmap. Update after each major milestone.