// package.json

{

"name": "ai-virtual-stylist-pro",

"version": "1.0.0",

"scripts": {

"dev": "next dev",

"build": "next build",

"start": "next start"

},

"dependencies": {

"next": "^14.0.0",

"react": "^18.2.0",

"three": "^0.158.0",

"@react-three/fiber": "^8.15.0",

"@react-three/drei": "^9.88.0",

"@tensorflow/tfjs": "^4.13.0",

"@mediapipe/pose": "^0.5.0",

"framer-motion": "^10.16.0",

"zustand": "^4.4.0",

"react-dropzone": "^14.2.0",

"tailwindcss": "^3.3.0",

"axios": "^1.6.0",

"@headlessui/react": "^1.7.0",

"react-hot-toast": "^2.4.0"

}

}

ai-virtual-stylist/

├── app/

│ ├── layout.tsx

│ ├── page.tsx

│ ├── api/

│ │ ├── analyze/route.ts

│ │ ├── generate-avatar/route.ts

│ │ ├── recommendations/route.ts

│ │ └── tryon/route.ts

│ └── dashboard/

│ └── page.tsx

├── components/

│ ├── ImageUploader.tsx

│ ├── Avatar3D.tsx

│ ├── VirtualDressingRoom.tsx

│ ├── StyleRecommendations.tsx

│ └── MeasurementsDisplay.tsx

├── lib/

│ ├── ai-models/

│ │ ├── bodyAnalyzer.ts

│ │ ├── avatarGenerator.ts

│ │ └── styleAI.ts

│ └── utils/

└── public/

'use client';

import { useState } from 'react';

import { motion } from 'framer-motion';

import ImageUploader from '@/components/ImageUploader';

import Avatar3D from '@/components/Avatar3D';

import VirtualDressingRoom from '@/components/VirtualDressingRoom';

import StyleRecommendations from '@/components/StyleRecommendations';

export default function Home() {

const [userImage, setUserImage] = useState(null);

const [avatarData, setAvatarData] = useState(null);

const [measurements, setMeasurements] = useState(null);

const [selectedOutfit, setSelectedOutfit] = useState(null);

return (

<div className="min-h-screen bg-gradient-to-br from-purple-50 via-white to-pink-50">

{/\* Hero Section \*/}

<motion.section

initial={{ opacity: 0 }}

animate={{ opacity: 1 }}

className="container mx-auto px-4 py-16"

>

<div className="text-center mb-12">

<h1 className="text-5xl font-bold bg-gradient-to-r from-purple-600 to-pink-600 bg-clip-text text-transparent mb-4">

AI Virtual Stylist Pro

</h1>

<p className="text-xl text-gray-600 max-w-2xl mx-auto">

Experience the future of fashion with our AI-powered 3D virtual dressing room.

Get precise measurements, realistic avatars, and personalized style recommendations.

</p>

</div>

{/\* Main App Grid \*/}

<div className="grid lg:grid-cols-2 gap-8 max-w-7xl mx-auto">

{/\* Left Panel - Upload & Analysis \*/}

<div className="space-y-6">

<div className="bg-white rounded-2xl shadow-xl p-6">

<h2 className="text-2xl font-semibold mb-4">Step 1: Upload Your Photo</h2>

<ImageUploader

onImageUpload={handleImageUpload}

onAnalysisComplete={(data) => {

setAvatarData(data.avatar);

setMeasurements(data.measurements);

}}

/>

</div>

{measurements && (

<motion.div

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

className="bg-white rounded-2xl shadow-xl p-6"

>

<h3 className="text-xl font-semibold mb-4">Your Measurements</h3>

<MeasurementsDisplay data={measurements} />

</motion.div>

)}

</div>

{/\* Right Panel - 3D Avatar & Dressing Room \*/}

<div className="space-y-6">

{avatarData && (

<>

<div className="bg-white rounded-2xl shadow-xl p-6 h-[500px]">

<h2 className="text-2xl font-semibold mb-4">Your 3D Avatar</h2>

<Avatar3D

avatarData={avatarData}

outfit={selectedOutfit}

/>

</div>

<VirtualDressingRoom

onOutfitSelect={setSelectedOutfit}

bodyType={measurements?.bodyType}

/>

</>

)}

</div>

</div>

{/\* AI Recommendations Section \*/}

{avatarData && (

<motion.div

initial={{ opacity: 0 }}

animate={{ opacity: 1 }}

className="mt-12"

>

<StyleRecommendations

measurements={measurements}

skinTone={avatarData.skinTone}

onSelectOutfit={setSelectedOutfit}

/>

</motion.div>

)}

</motion.section>

</div>

);

}

// components/ImageUploader.tsx

import { useCallback, useState } from 'react';

import { useDropzone } from 'react-dropzone';

import { motion } from 'framer-motion';

import toast from 'react-hot-toast';

export default function ImageUploader({ onImageUpload, onAnalysisComplete }) {

const [isProcessing, setIsProcessing] = useState(false);

const [preview, setPreview] = useState(null);

const onDrop = useCallback(async (acceptedFiles) => {

const file = acceptedFiles[0];

if (!file) return;

// Create preview

const reader = new FileReader();

reader.onload = () => setPreview(reader.result);

reader.readAsDataURL(file);

// Process image

setIsProcessing(true);

const formData = new FormData();

formData.append('image', file);

try {

// Analyze body measurements

const analysisRes = await fetch('/api/analyze', {

method: 'POST',

body: formData

});

const analysisData = await analysisRes.json();

// Generate 3D avatar

const avatarRes = await fetch('/api/generate-avatar', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(analysisData)

});

const avatarData = await avatarRes.json();

onAnalysisComplete({

measurements: analysisData,

avatar: avatarData

});

toast.success('Analysis complete! Your 3D avatar is ready.');

} catch (error) {

toast.error('Failed to process image. Please try again.');

} finally {

setIsProcessing(false);

}

}, [onAnalysisComplete]);

const { getRootProps, getInputProps, isDragActive } = useDropzone({

onDrop,

accept: {

'image/\*': ['.jpeg', '.jpg', '.png', '.webp']

},

maxFiles: 1

});

return (

<div className="w-full">

<div

{...getRootProps()}

className={`

relative border-2 border-dashed rounded-xl p-8 text-center cursor-pointer

transition-all duration-300

${isDragActive ? 'border-purple-500 bg-purple-50' : 'border-gray-300 hover:border-purple-400'}

${preview ? 'bg-gray-50' : 'bg-white'}

`}

>

<input {...getInputProps()} />

{!preview ? (

<div className="space-y-4">

<div className="w-20 h-20 mx-auto bg-purple-100 rounded-full flex items-center justify-center">

<svg className="w-10 h-10 text-purple-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M7 16a4 4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M15 13l-3-3m0 0l-3 3m3-3v12" />

</svg>

</div>

<div>

<p className="text-lg font-medium text-gray-700">

{isDragActive ? 'Drop your photo here' : 'Drag & drop your full-body photo'}

</p>

<p className="text-sm text-gray-500 mt-2">or click to browse</p>

</div>

</div>

) : (

<div className="relative">

<img src={preview} alt="Preview" className="max-h-64 mx-auto rounded-lg" />

{isProcessing && (

<div className="absolute inset-0 bg-white/80 rounded-lg flex items-center justify-center">

<div className="text-center">

<div className="animate-spin rounded-full h-12 w-12 border-b-2 border-purple-600 mx-auto mb-4"></div>

<p className="text-sm font-medium">Analyzing your photo...</p>

</div>

</div>

)}

</div>

)}

</div>

</div>

);

}

// components/Avatar3D.tsx

import { Canvas } from '@react-three/fiber';

import { OrbitControls, Environment, ContactShadows } from '@react-three/drei';

import { Suspense, useRef } from 'react';

import \* as THREE from 'three';

function AvatarModel({ avatarData, outfit }) {

const meshRef = useRef();

return (

<group ref={meshRef}>

{/\* Base body mesh with user's characteristics \*/}

<mesh position={[0, 0, 0]}>

<capsuleGeometry args={[0.5, avatarData.height / 100, 8, 16]} />

<meshStandardMaterial

color={avatarData.skinTone}

roughness={0.8}

metalness={0.1}

/>

</mesh>

{/\* Dynamic clothing meshes \*/}

{outfit && (

<group>

{outfit.top && (

<mesh position={[0, 0.5, 0.01]}>

<planeGeometry args={[1, 1.2]} />

<meshStandardMaterial

map={outfit.top.texture}

transparent

opacity={0.9}

/>

</mesh>

)}

{outfit.bottom && (

<mesh position={[0, -0.5, 0.01]}>

<planeGeometry args={[0.8, 1]} />

<meshStandardMaterial

map={outfit.bottom.texture}

transparent

opacity={0.9}

/>

</mesh>

)}

</group>

)}

</group>

);

}

export default function Avatar3D({ avatarData, outfit }) {

return (

<div className="w-full h-full rounded-lg overflow-hidden bg-gradient-to-b from-gray-100 to-gray-200">

<Canvas camera={{ position: [0, 0, 3], fov: 50 }}>

<Suspense fallback={null}>

<ambientLight intensity={0.5} />

<directionalLight position={[10, 10, 5]} intensity={1} />

<AvatarModel avatarData={avatarData} outfit={outfit} />

<ContactShadows

position={[0, -1.4, 0]}

opacity={0.5}

scale={5}

blur={2.5}

/>

<Environment preset="studio" />

<OrbitControls

enablePan={false}

maxPolarAngle={Math.PI / 2}

minDistance={2}

maxDistance={5}

/>

</Suspense>

</Canvas>

</div>

);

}

// app/api/analyze/route.ts

import { NextResponse } from 'next/server';

import \* as tf from '@tensorflow/tfjs-node';

import \* as poseDetection from '@tensorflow-models/pose-detection';

export async function POST(request: Request) {

try {

const formData = await request.formData();

const image = formData.get('image') as File;

if (!image) {

return NextResponse.json({ error: 'No image provided' }, { status: 400 });

}

// Convert image to tensor

const imageBuffer = await image.arrayBuffer();

const imageTensor = tf.node.decodeImage(new Uint8Array(imageBuffer));

// Load pose detection model

const detector = await poseDetection.createDetector(

poseDetection.SupportedModels.MoveNet,

{ modelType: poseDetection.movenet.modelType.MULTIPOSE\_LIGHTNING }

);

// Detect poses

const poses = await detector.estimatePoses(imageTensor);

// Calculate body measurements from keypoints

const measurements = calculateMeasurements(poses[0].keypoints);

// Analyze skin tone

const skinTone = await analyzeSkinTone(imageTensor);

// Determine body type

const bodyType = determineBodyType(measurements);

return NextResponse.json({

measurements: {

height: measurements.height,

chest: measurements.chest,

waist: measurements.waist,

hips: measurements.hips,

shoulderWidth: measurements.shoulderWidth,

armLength: measurements.armLength,

inseam: measurements.inseam,

bodyType: bodyType

},

skinTone: skinTone,

keypoints: poses[0].keypoints

});

} catch (error) {

console.error('Analysis error:', error);

return NextResponse.json(

{ error: 'Failed to analyze image' },

{ status: 500 }

);

}

}

function calculateMeasurements(keypoints) {

// Calculate distances between keypoints to estimate measurements

const shoulderLeft = keypoints.find(kp => kp.name === 'left\_shoulder');

const shoulderRight = keypoints.find(kp => kp.name === 'right\_shoulder');

const hipLeft = keypoints.find(kp => kp.name === 'left\_hip');

const hipRight = keypoints.find(kp => kp.name === 'right\_hip');

const ankleLeft = keypoints.find(kp => kp.name === 'left\_ankle');

// Calculate measurements using keypoint distances

const shoulderWidth = calculateDistance(shoulderLeft, shoulderRight) \* 2.5; // Scale factor

const torsoLength = calculateDistance(shoulderLeft, hipLeft) \* 2.5;

const legLength = calculateDistance(hipLeft, ankleLeft) \* 2.5;

return {

height: torsoLength + legLength,

chest: shoulderWidth \* 1.2,

waist: shoulderWidth \* 0.9,

hips: shoulderWidth \* 1.1,

shoulderWidth: shoulderWidth,

armLength: torsoLength \* 0.8,

inseam: legLength \* 0.8

};

}

function calculateDistance(point1, point2) {

if (!point1 || !point2) return 0;

const dx = point1.x - point2.x;

const dy = point1.y - point2.y;

return Math.sqrt(dx \* dx + dy \* dy);

}

async function analyzeSkinTone(imageTensor) {

// Simplified skin tone detection

const pixels = await imageTensor.array();

// Process pixels to extract dominant skin color

// This would use more sophisticated color analysis in production

return '#D4A574'; // Example skin tone

}

function determineBodyType(measurements) {

const ratio = measurements.waist / measurements.hips;

if (ratio < 0.75) return 'pear';

if (ratio > 0.85) return 'apple';

if (Math.abs(measurements.chest - measurements.hips) < 5) return 'rectangle';

return 'hourglass';

}

// components/VirtualDressingRoom.tsx

import { useState } from 'react';

import { motion, AnimatePresence } from 'framer-motion';

const clothingCategories = {

tops: [

{ id: 1, name: 'Classic White Shirt', color: '#FFFFFF', price: 49.99, image: '/clothes/shirt1.jpg' },

{ id: 2, name: 'Navy Blazer', color: '#1F2937', price: 129.99, image: '/clothes/blazer1.jpg' },

{ id: 3, name: 'Casual T-Shirt', color: '#3B82F6', price: 24.99, image: '/clothes/tshirt1.jpg' },

],

bottoms: [

{ id: 4, name: 'Dark Denim Jeans', color: '#1E293B', price: 79.99, image: '/clothes/jeans1.jpg' },

{ id: 5, name: 'Khaki Chinos', color: '#D4A574', price: 69.99, image: '/clothes/chinos1.jpg' },

{ id: 6, name: 'Black Trousers', color: '#000000', price: 89.99, image: '/clothes/trousers1.jpg' },

],

shoes: [

{ id: 7, name: 'White Sneakers', color: '#FFFFFF', price: 99.99, image: '/clothes/sneakers1.jpg' },

{ id: 8, name: 'Brown Oxfords', color: '#7C2D12', price: 149.99, image: '/clothes/oxfords1.jpg' },

],

accessories: [

{ id: 9, name: 'Leather Watch', color: '#7C2D12', price: 199.99, image: '/clothes/watch1.jpg' },

{ id: 10, name: 'Silver Necklace', color: '#E5E7EB', price: 79.99, image: '/clothes/necklace1.jpg' },

]

};

export default function VirtualDressingRoom({ onOutfitSelect, bodyType }) {

const [selectedCategory, setSelectedCategory] = useState('tops');

const [selectedItems, setSelectedItems] = useState({

top: null,

bottom: null,

shoes: null,

accessories: []

});

const handleItemSelect = (item, category) => {

const updatedItems = { ...selectedItems };

if (category === 'tops') updatedItems.top = item;

else if (category === 'bottoms') updatedItems.bottom = item;

else if (category === 'shoes') updatedItems.shoes = item;

else if (category === 'accessories') {

if (updatedItems.accessories.find(a => a.id === item.id)) {

updatedItems.accessories = updatedItems.accessories.filter(a => a.id !== item.id);

} else {

updatedItems.accessories.push(item);

}

}

setSelectedItems(updatedItems);

onOutfitSelect(updatedItems);

};

return (

<div className="bg-white rounded-2xl shadow-xl p-6">

<h3 className="text-2xl font-semibold mb-6">Virtual Dressing Room</h3>

{/\* Category Tabs \*/}

<div className="flex space-x-2 mb-6 overflow-x-auto">

{Object.keys(clothingCategories).map((category) => (

<button

key={category}

onClick={() => setSelectedCategory(category)}

className={`

px-4 py-2 rounded-lg font-medium capitalize transition-all

${selectedCategory === category

? 'bg-purple-600 text-white'

: 'bg-gray-100 text-gray-700 hover:bg-gray-200'}

`}

>

{category}

</button>

))}

</div>

{/\* Clothing Grid \*/}

<div className="grid grid-cols-2 md:grid-cols-3 gap-4 max-h-96 overflow-y-auto">

<AnimatePresence mode="wait">

{clothingCategories[selectedCategory].map((item) => {

const isSelected =

selectedItems.top?.id === item.id ||

selectedItems.bottom?.id === item.id ||

selectedItems.shoes?.id === item.id ||

selectedItems.accessories.find(a => a.id === item.id);

return (

<motion.div

key={item.id}

initial={{ opacity: 0, scale: 0.9 }}

animate={{ opacity: 1, scale: 1 }}

exit={{ opacity: 0, scale: 0.9 }}

whileHover={{ scale: 1.05 }}

onClick={() => handleItemSelect(item, selectedCategory)}

className={`

relative cursor-pointer rounded-lg overflow-hidden border-2 transition-all

${isSelected ? 'border-purple-500 shadow-lg' : 'border-gray-200 hover:border-purple-300'}

`}

>

<div className="aspect-square bg-gray-100 flex items-center justify-center">

<div

className="w-20 h-20 rounded-lg"

style={{ backgroundColor: item.color }}

/>

</div>

<div className="p-3">

<p className="text-sm font-medium text-gray-900 truncate">{item.name}</p>

<p className="text-sm text-purple-600 font-semibold">${item.price}</p>

</div>

{isSelected && (

<div className="absolute top-2 right-2 bg-purple-600 text-white rounded-full p-1">

<svg className="w-4 h-4" fill="currentColor" viewBox="0 0 20 20">

<path fillRule="evenodd" d="M16.707 5.293a1 1 0 010 1.414l-8 8a1 1 0 01-1.414 0l-4-4a1 1 0 011.414-1.414L8 12.586l7.293-7.293a1 1 0 011.414 0z" clipRule="evenodd" />

</svg>

</div>

)}

</motion.div>

);

})}

</AnimatePresence>

</div>

{/\* Selected Outfit Summary \*/}

{(selectedItems.top || selectedItems.bottom) && (

<motion.div

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

className="mt-6 p-4 bg-purple-50 rounded-lg"

>

<h4 className="font-semibold text-purple-900 mb-2">Current Outfit</h4>

<div className="space-y-1 text-sm">

{selectedItems.top && <p>Top: {selectedItems.top.name}</p>}

{selectedItems.bottom && <p>Bottom: {selectedItems.bottom.name}</p>}

{selectedItems.shoes && <p>Shoes: {selectedItems.shoes.name}</p>}

{selectedItems.accessories.length > 0 && (

<p>Accessories: {selectedItems.accessories.map(a => a.name).join(', ')}</p>

)}

</div>

<p className="mt-2 font-semibold text-purple-900">

Total: ${calculateTotal(selectedItems).toFixed(2)}

</p>

</motion.div>

)}

</div>

);

}

function calculateTotal(items) {

let total = 0;

if (items.top) total += items.top.price;

if (items.bottom) total += items.bottom.price;

if (items.shoes) total += items.shoes.price;

items.accessories.forEach(a => total += a.price);

return total;

}

// components/StyleRecommendations.tsx

import { useEffect, useState } from 'react';

import { motion } from 'framer-motion';

export default function StyleRecommendations({ measurements, skinTone, onSelectOutfit }) {

const [recommendations, setRecommendations] = useState([]);

const [loading, setLoading] = useState(true);

useEffect(() => {

fetchRecommendations();

}, [measurements, skinTone]);

const fetchRecommendations = async () => {

try {

const response = await fetch('/api/recommendations', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ measurements, skinTone })

});

const data = await response.json();

setRecommendations(data.recommendations);

} catch (error) {

console.error('Failed to fetch recommendations:', error);

} finally {

setLoading(false);

}

};

return (

<div className="bg-white rounded-2xl shadow-xl p-8">

<h2 className="text-3xl font-bold mb-6 bg-gradient-to-r from-purple-600 to-pink-600 bg-clip-text text-transparent">

AI Style Recommendations

</h2>

{loading ? (

<div className="flex justify-center py-12">

<div className="animate-spin rounded-full h-12 w-12 border-b-2 border-purple-600"></div>

</div>

) : (

<div className="grid md:grid-cols-2 lg:grid-cols-3 gap-6">

{recommendations.map((outfit, index) => (

<motion.div

key={index}

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

transition={{ delay: index \* 0.1 }}

className="border rounded-xl overflow-hidden hover:shadow-lg transition-shadow cursor-pointer"

onClick={() => onSelectOutfit(outfit)}

>

<div className="aspect-square bg-gradient-to-br from-purple-100 to-pink-100 p-4">

<div className="h-full rounded-lg bg-white/50 backdrop-blur-sm flex items-center justify-center">

<span className="text-4xl">👔</span>

</div>

</div>

<div className="p-4">

<h3 className="font-semibold text-lg mb-2">{outfit.name}</h3>

<p className="text-sm text-gray-600 mb-3">{outfit.description}</p>

<div className="flex items-center justify-between">

<span className="text-sm bg-purple-100 text-purple-700 px-2 py-1 rounded">

{outfit.occasion}

</span>

<span className="text-sm font-semibold text-green-600">

{outfit.matchScore}% Match

</span>

</div>

</div>

</motion.div>

))}

</div>

)}

</div>

);

}

{

"functions": {

"app/api/analyze/route.ts": {

"maxDuration": 30

},

"app/api/generate-avatar/route.ts": {

"maxDuration": 30

}

},

"env": {

"NEXT\_PUBLIC\_API\_URL": "@api\_url",

"AI\_API\_KEY": "@ai\_api\_key",

"DATABASE\_URL": "@database\_url"

}

}

# AI Services

OPENAI\_API\_KEY=your\_openai\_key

REPLICATE\_API\_TOKEN=your\_replicate\_token

# Database

DATABASE\_URL=your\_database\_url

# Storage

AWS\_ACCESS\_KEY\_ID=your\_aws\_key

AWS\_SECRET\_ACCESS\_KEY=your\_aws\_secret

AWS\_BUCKET\_NAME=your\_bucket\_name

# Analytics

NEXT\_PUBLIC\_GA\_ID=your\_ga\_id

// lib/utils/imageOptimizer.ts

export async function optimizeImage(file: File): Promise<Blob> {

const canvas = document.createElement('canvas');

const ctx = canvas.getContext('2d');

const img = new Image();

return new Promise((resolve) => {

img.onload = () => {

const MAX\_WIDTH = 1024;

const MAX\_HEIGHT = 1024;

let width = img.width;

let height = img.height;

if (width > height) {

if (width > MAX\_WIDTH) {

height \*= MAX\_WIDTH / width;

width = MAX\_WIDTH;

}

} else {

if (height > MAX\_HEIGHT) {

width \*= MAX\_HEIGHT / height;

height = MAX\_HEIGHT;

}

}

canvas.width = width;

canvas.height = height;

ctx.drawImage(img, 0, 0, width, height);

canvas.toBlob((blob) => resolve(blob), 'image/jpeg', 0.85);

};

img.src = URL.createObjectURL(file);

});

}

// lib/cache/modelCache.ts

class ModelCache {

private cache = new Map();

async getOrLoad(key: string, loader: () => Promise<any>) {

if (this.cache.has(key)) {

return this.cache.get(key);

}

const data = await loader();

this.cache.set(key, data);

return data;

}

clear() {

this.cache.clear();

}

}

export const modelCache = new ModelCache();

// public/manifest.json

{

"name": "AI Virtual Stylist Pro",

"short\_name": "StyleAI",

"description": "AI-powered virtual dressing room",

"start\_url": "/",

"display": "standalone",

"background\_color": "#ffffff",

"theme\_color": "#7c3aed",

"icons": [

{

"src": "/icon-192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "/icon-512.png",

"sizes": "512x512",

"type": "image/png"

}

]

}

# Clone and install

git clone https://github.com/yourusername/ai-virtual-stylist.git

cd ai-virtual-stylist

npm install

# Setup environment

cp .env.example .env.local

# Edit .env.local with your API keys

# Run development

npm run dev

# Build for production

npm run build

npm run start

# Deploy to Vercel

vercel --prod