Basketball Analysis Assessment App - Comprehensive Deployment Analysis

Repository: https://github.com/baller70/BasketballAnalysisAssessmentApp.git
Analysis Date: December 26, 2025
Cloned Location: /home/ubuntu/basketball_app

📋 Executive Summary

The Basketball Analysis Assessment App is a full-stack Next.js application that provides AI-powered basketball shooting form analysis. It uses a hybrid architecture combining:
- **Frontend:** Next.js 14 (TypeScript, React 18)
- **Backend:** Python FastAPI + Flask (multiple services)
- **Database:** PostgreSQL (Prisma ORM)
- **AI/ML:** Multiple providers (OpenAI, Replicate, RoboFlow, MediaPipe, HuggingFace)
- **Storage:** AWS S3
- **Authentication:** NextAuth.js

🏗️ Application Architecture

Primary Components

1. **nextjs_space/** - Main Next.js frontend application
2. **basketball-analysis/** - Alternative Next.js frontend (more feature-complete)
3. **python-backend/** - FastAPI service for pose detection
4. **huggingface-backend/** - Flask service for advanced pose analysis
5. **dual_tier_analysis/** - Comparison logic for FREE vs PROFESSIONAL tiers
6. **python-scraper/** - Data collection tools
7. **image_collection/** - Image dataset management

Active Main Applications

- **PRIMARY FRONTEND:** `basketball-analysis/` (most feature-complete)
- **PRIMARY BACKEND:** `python-backend/` (FastAPI - pose detection with MediaPipe)
- **SECONDARY BACKEND:** `huggingface-backend/` (Flask - advanced hybrid analysis with YOLOv8)

📦 Technology Stack

Frontend (Next.js)

Framework & Core:
- Next.js 14.2.28/14.2.33
- React 18.2.0

- TypeScript 5.x
- TailwindCSS 3.3.3/3.4.1

Key Libraries:
- **Database:** Prisma 6.7.0/7.1.0 with @prisma/client
- **Authentication:** NextAuth 4.24.13
- **State Management:** Zustand 5.0.9
- **Data Fetching:** TanStack React Query 5.0.0/5.90.11
- **AI/ML:**
- TensorFlow.js 4.22.0
- MediaPipe Pose 0.5.1675469404
- @tensorflow-models/pose-detection 2.1.3
- @tensorflow-models/body-pix 2.2.1
- **UI Components:**
- Radix UI (various components)
- Framer Motion 12.23.25
- Lucide React 0.555.0
- Recharts 2.15.3/3.5.1
- Plotly.js 2.35.3/3.3.0
- **Storage:** AWS SDK (@aws-sdk/client-s3, @aws-sdk/s3-request-presigner)
- **Forms:** React Hook Form 7.67.0 + Zod 4.1.13
- **HuggingFace:** @gradio/client 2.0.1
- **Utilities:**
- uuid 13.0.0
- html-to-image 1.11.13
- axios 1.13.2
- bcryptjs 3.0.3

Backend (Python)

python-backend (FastAPI):

```
- FastAPI 0.109.2
- Uvicorn 0.27.1
- MediaPipe >=0.10.13
- OpenCV 4.9.0.80
- Pillow 10.2.0
- NumPy 1.26.4
- Replicate 0.25.1
- Pydantic 2.6.1
```

huggingface-backend (Flask):

```
- Flask 3.0.3
- Flask-CORS 4.0.1
- Gunicorn 22.0.0
- PyTorch 2.2.0
- Torchvision 0.17.0
- Ultralytics 8.2.0 (YOLOv8)
- MediaPipe 0.10.14
- OpenCV Headless 4.9.0.80
- Pillow 10.3.0
- NumPy 1.26.4
```

dual_tier_analysis:

```
- MediaPipe >=0.10.0
- RoboFlow SDK (inference-sdk >=0.9.0, roboflow >=1.1.0)
- OpenCV >=4.8.0
- Pandas >=2.0.0
- SciPy >=1.11.0
- Matplotlib >=3.7.0
- Seaborn >=0.12.0
- Requests/HTTPX for API calls
```

🔑 Required Environment Variables

Frontend (Next.js)

Database:

```
DATABASE_URL="postgresql://user:password@host:5432/basketball_shooting_db"
```

AI/ML APIs:

```
# OpenAI (Vision AI analysis)
OPENAI_API_KEY="sk-your-openai-api-key"

# RoboFlow (Basketball detection)
ROBOFLOW_API_KEY="your-roboflow-api-key"

# Abacus AI (Optional)
ABACUS_API_KEY="your-abacus-api-key"
```

AWS S3 Storage:

```
AWS_ACCESS_KEY_ID="your-aws-access-key"
AWS_SECRET_ACCESS_KEY="your-aws-secret-key"
AWS_REGION="us-east-1"
S3_BUCKET_NAME="basketball-shooters-db"
```

NextAuth (Authentication):

```
NEXTAUTH_URL="http://localhost:3000"  # Or production URL
NEXTAUTH_SECRET="your-nextauth-secret-key"

# OAuth Providers (Optional)
GOOGLE_CLIENT_ID=""
GOOGLE_CLIENT_SECRET=""
GITHUB_CLIENT_ID=""
GITHUB_CLIENT_SECRET=""
```

Python Backend Connection:

```
NEXT_PUBLIC_PYTHON_API_URL="http://localhost:8000"  # Or deployed URL
```

HuggingFace (Optional - for image enhancement):

```
NEXT_PUBLIC_REALESRGAN_SPACE_URL="https://your-huggingface-space-url"
```

ShotStack (Video rendering - Optional):

```
SHOTSTACK_SANDBOX_API_KEY="your-shotstack-sandbox-key"
SHOTSTACK_PRODUCTION_API_KEY="your-shotstack-production-key"
SHOTSTACK_ENV="sandbox"  # or "production"
```

Backend (Python)

python-backend:

```
# Replicate API (for AI-powered skeleton detection)
REPLICATE_API_TOKEN="your_replicate_api_token"

# Server Configuration
HOST="0.0.0.0"
PORT="8000"

# CORS Configuration
ALLOWED_ORIGINS="http://localhost:3000,https://your-frontend-url.com"

# Optional: MediaPipe Model Complexity (0=Lite, 1=Full, 2=Heavy)
MEDIAPIPE_MODEL_COMPLEXITY="2"
```

huggingface-backend:

```
# CORS Configuration
ALLOWED_ORIGINS="*"  # Or specific origins
```

dual_tier_analysis:

```
# RoboFlow API
ROBOFLOW_API_KEY="your_roboflow_api_key"

# ShotStack API
SHOTSTACK_SANDBOX_API_KEY="your_shotstack_sandbox_key"
SHOTSTACK_PRODUCTION_API_KEY="your_shotstack_production_key"
SHOTSTACK_ENV="sandbox"

# Abacus AI
ABACUS_API_KEY="your_abacus_api_key"

# Vision API Configuration
VISION_PRIMARY_PROVIDER="anthropic"  # or "openai"
VISION_FALLBACK_PROVIDER="openai"
VISION_TIMEOUT="30"

# Performance
PARALLEL_PROCESSING="false"
MAX_WORKERS="4"

# Logging
LOG_LEVEL="INFO"
LOG_FILE="phase4_pipeline.log"
```

🗄️ Database Schema

Database Type: PostgreSQL
ORM: Prisma

Core Models:

1. **User** - Authentication and user accounts
   - id, email, password (hashed), name
   - Relations: profile, analyses

2. **UserProfile** - Player physical information
   - heightInches, weightLbs, wingspanInches
   - age, experienceLevel, bodyType, athleticAbility
   - dominantHand, shootingStyle
   - Relations: user, analyses, drillVideos

3. **UserAnalysis** - Analysis sessions
   - imageUrl, s3Path
   - roboflowPoseData, roboflowDetection, shootingPhase
   - Biomechanical angles: elbowAngle, kneeAngle, wristAngle, shoulderAngle, hipAngle, releaseAngle
   - visionAnalysis (JSON), bodyPositions (JSON)
   - Scores: overallScore, formScore, balanceScore, releaseScore, consistencyScore
   - strengths, improvements, drills (JSON)
   - matchedShooterId, matchConfidence, similarShooters
   - processingStatus, processingError
   - Relations: userProfile, historyEntries

4. **AnalysisHistory** - Track progress over time
   - Snapshot of scores and key metrics
   - scoreChange, improvementAreas, regressionAreas
   - Relations: analysis

5. **Shooter** (Elite Shooter Database)
   - name, position, heightInches, weightLbs, wingspanInches
   - bodyType, dominantHand, shootingStyle
   - careerFgPercentage, career3ptPercentage, careerFtPercentage
   - skillLevel, era
   - Relations: biomechanics, images, stats, strengths, weaknesses, habitualMechanics

6. **ShootingBiomechanics** - Elite shooter form data
   - elbowAngle, shoulderAngle, hipAngle, kneeAngle, ankleAngle
   - releaseHeight, releaseAngle, entryAngle
   - followThroughExtension, balanceScore, arcConsistency

7. **ShooterImage** - Elite shooter image library
   - imageCategory, imageUrl, s3Path
   - capturePhase, shootingAngle
   - isPrimary

8. **ShootingStats** - Season statistics
   - season, gamesPlayed
   - fgAttempts, fgMade, threePtAttempts, threePtMade
   - ftAttempts, ftMade, pointsPerGame

9. **ShootingStrength** / **ShootingWeakness** - Analysis points
   - strengthCategory/weaknessCategory
   - description, confidenceScore/severityScore

10. **HabitualMechanics** - Shooting habits

    - habitName, habitType, frequency
    - impactOnPerformance

11. **DrillVideoSubmission** - User drill videos

    - drillId, drillName, focusArea
    - mediaType, mediaUrl, thumbnailUrl
    - analyzed, analyzedAt, analysisType
    - Coach analysis results: overallGrade, gradeDescription, coachAnalysis (JSON)
    - Relations: userProfile

12. **NextAuth Models:**

    - Account, Session (for OAuth)

🤖 AI/ML Integrations

1. HuggingFace

Usage:
- Optional image enhancement via Gradio client
- Custom HuggingFace Spaces deployment available in `huggingface-backend/`
- Not currently deployed but can be hosted on HuggingFace Spaces

Integration Point:

```
// basketball-analysis/src/services/imageEnhancement.ts
const HUGGINGFACE_SPACE_URL = process.env.NEXT_PUBLIC_REALESRGAN_SPACE_URL || ''
```

Models Used:
- YOLOv8x-pose (person detection and pose estimation)
- MediaPipe (secondary pose estimation)
- RealESRGAN (image enhancement - optional)

2. OpenAI

API: GPT-4 Vision
Usage:
- Vision-based shooting form analysis
- Natural language coaching feedback
- Biomechanical assessment

Cost: ~$0.01 per image (FREE tier)

3. Anthropic Claude

API: Claude 3.5 Sonnet
Usage:
- Advanced vision analysis (PROFESSIONAL tier)
- Elite shooter comparison
- Detailed coaching recommendations

Cost: ~$0.03 per image (PROFESSIONAL tier)

4. Replicate

Usage: AI-powered skeleton detection
Required: Yes (for python-backend)
Token: Pre-configured in deployment docs

5. RoboFlow

API: Custom trained basketball pose detection
Usage:
- 18 basketball-specific keypoints
- Ball detection
- Shooting phase classification
- 95%+ accuracy on basketball poses

Training: Custom trained on 19,562 basketball images
Cost: ~$0.20 per 1000 predictions

6. MediaPipe

Type: Open-source (Google)
Usage:
- FREE tier pose detection
- 33 full-body landmarks
- Client-side and server-side
- 85-90% accuracy

Cost: FREE

7. TensorFlow.js

Usage:
- Client-side pose detection
- Body segmentation
- Real-time analysis

Models:
- @tensorflow-models/pose-detection
- @tensorflow-models/body-pix

8. Ultralytics YOLOv8

Usage:
- Person detection
- Pose estimation (17 keypoints)
- Basketball detection

Location: huggingface-backend

🌐 External API Services

1. AWS S3

- Image/video storage
- Pre-signed URLs for secure access
- Bucket: basketball-shooters-db

2. ShotStack (Optional)

- Video rendering and editing
- Annotation overlays
- Professional video output
- Sandbox and production environments

3. NextAuth Providers (Optional)

- Google OAuth
- GitHub OAuth
- Email/password authentication

📊 Dual-Tier Architecture

The app supports two analysis tiers:

FREE Tier

- **Cost:** ~$0.01 per analysis
- **Pose Detection:** MediaPipe (open-source)
- **Vision AI:** OpenAI GPT-4 Vision
- **Accuracy:** 85-90%
- **Keypoints:** 33 full-body
- **Best For:** Casual players, practice tracking

PROFESSIONAL Tier

- **Cost:** ~$0.50-1.00 per analysis
- **Pose Detection:** RoboFlow (custom trained)
- **Vision AI:** Anthropic Claude 3.5 Sonnet
- **Accuracy:** 95%+
- **Keypoints:** 18 basketball-specific
- **Best For:** Elite athletes, professional coaching
- **Additional:** ShotStack video rendering

🚀 Deployment Requirements

1. Frontend Deployment (Next.js)

Options:
- Vercel (recommended for Next.js)
- AWS Amplify
- Netlify
- Abacus AI hosting

Requirements:
- Node.js 20+
- PostgreSQL database (can be cloud-hosted)
- All environment variables configured
- Prisma migrations run

Build Command:

```
npm install
npx prisma generate
npm run build
```

Start Command:

```
npm start
```

2. Backend Deployment (Python)

Recommended Platform: Railway (from docs)

Alternative Platforms:
- Render
- Fly.io
- AWS Elastic Beanstalk
- Google Cloud Run
- Heroku

Requirements:
- Python 3.9+
- Dockerfile included in python-backend/
- Set root directory to `python-backend`
- Configure CORS for frontend URL

Start Command:

```
uvicorn app.main:app --host 0.0.0.0 --port $PORT
```

3. HuggingFace Backend (Optional)

Platform: HuggingFace Spaces

Requirements:
- Gradio/Flask Space
- GPU instance recommended for YOLOv8
- Configure ALLOWED_ORIGINS

Start Command:

```
gunicorn -w 4 -b 0.0.0.0:7860 app:app
```

4. Database

Required: PostgreSQL 14+

Options:
- Supabase (free tier available)
- Railway PostgreSQL
- AWS RDS
- Heroku Postgres
- Neon
- PlanetScale (with PostgreSQL adapter)

Setup:

```
# Install Prisma CLI
npm install -g prisma

# Generate Prisma client
npx prisma generate

# Run migrations
npx prisma migrate deploy

# Seed database (optional)
npx prisma db seed
```

🔐 Security Considerations

1. **API Keys:** Store in environment variables, never commit to Git
2. **Database:** Use connection pooling for production
3. **Authentication:** NextAuth handles OAuth securely
4. **CORS:** Restrict to specific frontend domains in production
5. **S3:** Use pre-signed URLs with expiration
6. **Passwords:** Bcrypt hashing for user passwords
7. **Rate Limiting:** Implement for API endpoints

📝 Configuration Checklist

Essential (Must Have):

- [ ] DATABASE_URL (PostgreSQL connection string)
- [ ] OPENAI_API_KEY (for vision analysis)
- [ ] AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, S3_BUCKET_NAME (for storage)
- [ ] NEXTAUTH_URL, NEXTAUTH_SECRET (for authentication)
- [ ] NEXT_PUBLIC_PYTHON_API_URL (backend connection)
- [ ] REPLICATE_API_TOKEN (for python-backend)
- [ ] ALLOWED_ORIGINS (for python-backend CORS)

Recommended (Enhanced Features):

- [ ] ROBOFLOW_API_KEY (for professional tier)
- [ ] GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET (OAuth)
- [ ] GITHUB_CLIENT_ID, GITHUB_CLIENT_SECRET (OAuth)
- [ ] NEXT_PUBLIC_REALESRGAN_SPACE_URL (image enhancement)

Optional (Premium Features):

- [ ] SHOTSTACK_SANDBOX_API_KEY (video rendering)
- [ ] SHOTSTACK_PRODUCTION_API_KEY (video rendering)
- [ ] ABACUS_API_KEY (Abacus AI integration)

🏃 Quick Start Guide

Local Development:

1. **Clone Repository:**

```
git clone https://github.com/baller70/BasketballAnalysisAssessmentApp.git
cd BasketballAnalysisAssessmentApp
```

1. **Setup Frontend:**

```
cd basketball-analysis  # or nextjs_space
npm install
cp .env.example .env
# Edit .env with your keys
npx prisma generate
npx prisma migrate dev
npm run dev
```

1. **Setup Python Backend:**

```
cd python-backend
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r requirements.txt
cp .env.example .env
# Edit .env with your keys
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

1. **Access:**
   - Frontend: http://localhost:3000
   - Backend: http://localhost:8000
   - API Docs: http://localhost:8000/docs

📚 Documentation Files

The repository includes extensive documentation:

- **QUICK_DEPLOY.md** - Fast Railway deployment guide (10-15 min)
- **START_BACKEND.md** - Backend startup instructions
- **DEPLOYMENT_QUICK_REFERENCE.md** - Essential deployment info
- **PYTHON_BACKEND_DEPLOYMENT_GUIDE.md** - Comprehensive backend deployment
- **TIER_COMPARISON_REPORT.md** - FREE vs PROFESSIONAL tier comparison
- **DATABASE_CONNECTION_ANALYSIS.md** - Database setup guide
- **PHASE4_INTEGRATION_GUIDE.md** - Advanced features integration
- **SHOTSTACK_INDEX.md** - Video rendering setup
- **ROBOFLOW_QUICK_START.md** - RoboFlow integration guide

🐛 Known Issues & Troubleshooting

Common Issues:

1. **CORS Errors:**
   - Ensure ALLOWED_ORIGINS includes your frontend URL
   - No trailing slashes in URLs

2. **Database Connection:**
   - Verify DATABASE_URL format
   - Ensure PostgreSQL is running
   - Run prisma migrations

3. **MediaPipe Not Available:**
   - Install system dependencies
   - Check python-backend Dockerfile

4. **Missing Environment Variables:**
   - Copy .env.example to .env in each directory
   - Verify all required keys are set

5. **Port Already in Use:**
   - Change PORT in environment variables
   - Update frontend NEXT_PUBLIC_PYTHON_API_URL

💰 Estimated Costs

Development/Testing:

- **Database:** Free (Supabase free tier)
- **Backend:** $5/month (Railway free tier credit)
- **Frontend:** Free (Vercel hobby tier)
- **S3 Storage:** ~$1-5/month
- **AI APIs:** ~$5-10/month (low usage)
- **Total:** ~$10-20/month

Production (100 users/day):

- **Database:** $5-10/month
- **Backend:** $10-20/month
- **Frontend:** Free-$20/month
- **S3 Storage:** ~$10-20/month
- **AI APIs:** ~$20-50/month
- **Total:** ~$50-120/month

High Traffic (1000+ users/day):

- **Database:** $20-50/month
- **Backend:** $50-100/month
- **Frontend:** $20-50/month
- **S3 Storage:** ~$50-100/month

- **AI APIs:** ~$200-500/month
- **Total:** ~$340-800/month

🎯 Next Steps for Deployment

1. **Choose hosting platforms:**
   - Frontend: Vercel/Netlify/AWS
   - Backend: Railway/Render/Fly.io
   - Database: Supabase/Railway/AWS RDS

2. **Obtain API keys:**
   - OpenAI API key
   - AWS credentials
   - Replicate API token
   - (Optional) RoboFlow API key

3. **Setup database:**
   - Create PostgreSQL instance
   - Run Prisma migrations
   - Optionally seed with elite shooters data

4. **Deploy backend first:**
   - Deploy python-backend to Railway
   - Test /health endpoint
   - Note the deployed URL

5. **Deploy frontend:**
   - Configure all environment variables
   - Set NEXT_PUBLIC_PYTHON_API_URL to backend URL
   - Deploy to Vercel

6. **Test integration:**
   - Upload test image
   - Verify pose detection works
   - Check database entries
   - Monitor logs for errors

7. **Optional enhancements:**
   - Deploy HuggingFace backend for advanced analysis
   - Setup ShotStack for video rendering
   - Configure OAuth providers
   - Enable RoboFlow for professional tier

📞 Support & Resources

- **Repository:** https://github.com/baller70/BasketballAnalysisAssessmentApp.git
- **Documentation:** See /docs folder in repository
- **Backend API Docs:** https://your-backend-url/docs (auto-generated by FastAPI)

- **Railway Support:** https://railway.app/help
- **Vercel Support:** https://vercel.com/support

Analysis Completed: December 26, 2025
Analyst: DeepAgent AI
Version: 1.0.0