Real-time Virtual Try-On Mobile Application: Technical Specifications

1. System Architecture
- Mobile application (iOS and Android)
- Backend server (RESTful API)
- Database (PostgreSQL)
- Cloud storage for images
- Content Delivery Network (CDN) for fast image delivery

2. Technology Stack
- Mobile: React Native
- Backend: Node.js with Express.js
- Database: PostgreSQL
- ORM: Sequelize
- Authentication: JWT, OAuth 2.0 for SSO
- Image Processing: OpenCV or TensorFlow for virtual try-ons
- Cloud Services: AWS (S3 for storage, EC2 for hosting, CloudFront for CDN)

3. Database Schema
```
-- Users table (for both individual users and businesses/affiliates)
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    user_type ENUM('individual', 'business', 'affiliate') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- User profiles
CREATE TABLE UserProfiles (
    profile_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    full_name VARCHAR(255),
    profile_picture_url VARCHAR(255),
    bio TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Business profiles
CREATE TABLE BusinessProfiles (
```

```sql
    business_profile_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    business_name VARCHAR(255) NOT NULL,
    description TEXT,
    website_url VARCHAR(255),
    subscription_plan ENUM('basic', 'premium', 'enterprise') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Products
CREATE TABLE Products (
    product_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    category ENUM('clothing', 'hair', 'footwear', 'accessories') NOT NULL,
    image_url VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Product variations
CREATE TABLE ProductVariations (
    variation_id SERIAL PRIMARY KEY,
    product_id INTEGER REFERENCES Products(product_id),
    color VARCHAR(50),
    size VARCHAR(20),
    stock_quantity INTEGER,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Try-ons
CREATE TABLE TryOns (
    tryon_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    product_id INTEGER REFERENCES Products(product_id),
    image_url VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Social shares
CREATE TABLE SocialShares (
    share_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    tryon_id INTEGER REFERENCES TryOns(tryon_id),
    platform VARCHAR(50) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Notifications
CREATE TABLE Notifications (
    notification_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    content TEXT NOT NULL,
    is_read BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Analytics
CREATE TABLE Analytics (
    analytics_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    product_id INTEGER REFERENCES Products(product_id),
    event_type ENUM('try_on', 'click_through', 'purchase') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Ecommerce orders (for Phase 2)
CREATE TABLE Orders (
    order_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    total_amount DECIMAL(10, 2) NOT NULL,
    status ENUM('pending', 'processing', 'shipped', 'delivered',
'cancelled') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```sql
-- Order items (for Phase 2)
CREATE TABLE OrderItems (
    order_item_id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES Orders(order_id),
    product_id INTEGER REFERENCES Products(product_id),
    variation_id INTEGER REFERENCES ProductVariations(variation_id),
    quantity INTEGER NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Existing tables remain the same

-- Password Reset Tokens
CREATE TABLE PasswordResetTokens (
    token_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    token VARCHAR(255) UNIQUE NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- User Sessions
CREATE TABLE UserSessions (
    session_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    token VARCHAR(255) UNIQUE NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Two-Factor Authentication
CREATE TABLE TwoFactorAuth (
    twofa_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    secret_key VARCHAR(255) NOT NULL,
    is_enabled BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```sql
);

-- Likes
CREATE TABLE Likes (
    like_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    content_type ENUM('product', 'tryon') NOT NULL,
    content_id INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Comments
CREATE TABLE Comments (
    comment_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    tryon_id INTEGER REFERENCES TryOns(tryon_id),
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Followers
CREATE TABLE Followers (
    follower_id INTEGER REFERENCES Users(user_id),
    followed_id INTEGER REFERENCES Users(user_id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (follower_id, followed_id)
);

-- Content Reports
CREATE TABLE ContentReports (
    report_id SERIAL PRIMARY KEY,
    reporter_id INTEGER REFERENCES Users(user_id),
    content_type ENUM('product', 'tryon', 'comment') NOT NULL,
    content_id INTEGER NOT NULL,
    reason TEXT NOT NULL,
    status ENUM('pending', 'reviewed', 'actioned', 'dismissed') DEFAULT
'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```sql
);

-- Tags
CREATE TABLE Tags (
    tag_id SERIAL PRIMARY KEY,
    name VARCHAR(50) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Content Tags (for both products and try-ons)
CREATE TABLE ContentTags (
    content_tag_id SERIAL PRIMARY KEY,
    tag_id INTEGER REFERENCES Tags(tag_id),
    content_type ENUM('product', 'tryon') NOT NULL,
    content_id INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Search History
CREATE TABLE SearchHistory (
    search_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    query TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- User Preferences
CREATE TABLE UserPreferences (
    preference_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    notification_settings JSONB,
    privacy_settings JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Product Reviews
CREATE TABLE ProductReviews (
    review_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    product_id INTEGER REFERENCES Products(product_id),
    rating INTEGER CHECK (rating >= 1 AND rating <= 5),
    content TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Existing tables remain the same

-- Social Media Authentication
CREATE TABLE SocialAuth (
    social_auth_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES Users(user_id),
    provider ENUM('google', 'facebook', 'apple', 'twitter') NOT NULL,
    provider_user_id VARCHAR(255) NOT NULL,
    access_token TEXT,
    refresh_token TEXT,
    token_expires_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(provider, provider_user_id)
);

-- Add index for performance
CREATE INDEX idx_social_auth_user ON SocialAuth(user_id);
CREATE INDEX idx_tryon_user ON TryOns(user_id);
CREATE INDEX idx_tryon_product ON TryOns(product_id);
CREATE INDEX idx_likes_user ON Likes(user_id);
CREATE INDEX idx_likes_content ON Likes(content_type, content_id);
CREATE INDEX idx_comments_tryon ON Comments(tryon_id);
CREATE INDEX idx_followers_follower ON Followers(follower_id);
CREATE INDEX idx_followers_followed ON Followers(followed_id);
CREATE INDEX idx_content_tags_tag ON ContentTags(tag_id);
CREATE INDEX idx_content_tags_content ON ContentTags(content_type,
content_id);
CREATE INDEX idx_search_history_user ON SearchHistory(user_id);
CREATE INDEX idx_product_reviews_product ON ProductReviews(product_id);
```

4. API Endpoints

- User Management: /api/users/
- Authentication: /api/auth/
- Products: /api/products/
- Try-ons: /api/tryons/
- Social Features: /api/social/
- Analytics: /api/analytics/

…

5. Security Measures
- HTTPS for all communications
- JWT for session management
- OAuth 2.0 for SSO
- Password hashing using bcrypt
- Input validation and sanitization
- Rate limiting to prevent abuse
- Regular security audits

6. Scalability Considerations
- Horizontal scaling of backend servers
- Database sharding for large datasets
- Caching layer (e.g., Redis) for frequently accessed data
- Asynchronous processing for time-consuming tasks (e.g., image processing)

7. Third-party Integrations
- Social media platforms for SSO and sharing
- Payment gateways for e-commerce features
- Analytics tools for business insights
- Cloud services for infrastructure

8. Testing Strategy
- Unit testing for individual components
- Integration testing for API endpoints
- End-to-end testing for critical user flows
- Performance testing to ensure responsiveness under load
- Security testing to identify vulnerabilities

9. Deployment and DevOps
- CI/CD pipeline for automated testing and deployment
- Containerization using Docker for consistent environments
- Kubernetes for orchestration and scaling
- Monitoring and logging (e.g., ELK stack, Prometheus)

10. Future Considerations
- AI/ML integration for personalized recommendations

- AR capabilities for enhanced try-on experience
- Internationalization and localization
- Accessibility features for users with disabilities

This document serves as a living guide for the technical implementation of the Virtual Try-On Mobile Application. It should be regularly updated as the project evolves and new technical decisions are made.