Software Requirements Specification Document

ManageTheFans Portal

System Design

- **Multi-tier architecture** with separate frontend, backend, and database layers
- **Responsive design** for desktop and mobile devices
- **Progressive Web App (PWA)** capabilities for mobile installation
- **Role-based access control** (Admin vs Client users)
- **Secure file storage** using Supabase Storage
- **Asynchronous processing** for notifications and email delivery

Architecture Pattern

- **Frontend**: Component-based architecture using React
- **Backend**: RESTful API with Node.js
- **Serverless approach** leveraging Supabase for database and storage
- **Microservices approach** for:
  - User authentication (Clerk)
  - Payment processing
  - Content management
  - Notification delivery
  - Appointment scheduling

State Management

- **Redux** for global state management
- **Context API** for component-specific state
- **Local storage** for persisting user preferences and session data
- **Form state management** with React Hook Form
- **Persistent state** for multi-step onboarding process

Data Flow

- **Client-side rendering** with server-side data fetching

- **RESTful API calls** between frontend and backend
- **Supabase Realtime** for real-time notifications and messaging
- **Secure file upload** flow with progress indicators and validation
- **Event-driven architecture** for notification system

Technical Stack

- **Frontend**:
  - React.js with TypeScript
  - Tailwind CSS for styling (matches the dark theme requirements)
  - Redux for state management
  - Axios for API requests
  - React Router for navigation
  - React Hook Form for form management
- **Backend**:
  - Node.js with Express
  - TypeScript for type safety
  - Clerk for authentication
  - Supabase Client for database operations
- **Database & Storage**:
  - Supabase PostgreSQL for relational data
  - Supabase Storage for secure file storage
- **Infrastructure**:
  - Vercel for hosting
  - Supabase for backend services and messaging
  - Stripe API for payment processing
  - Resend API for email notifications

Authentication Process

- **Clerk authentication** with customizable UI
- **JWT handling** managed by Clerk
- **OAuth 2.0** for third-party platform integration
- **Multi-factor authentication** option for admin accounts
- **Role-based permissions** for admin vs client access
- **User management dashboard** through Clerk

Route Design

- **Public routes**:
  - `/login` (handled by Clerk)
  - `/register` (handled by Clerk)
  - `/forgot-password` (handled by Clerk)
- **Protected client routes**:
  - `/dashboard`
  - `/onboarding/*` (multi-step process)
  - `/profile`
  - `/brand-strategy`
  - `/content-upload`
  - `/billing`
  - `/appointments`
  - `/messaging`
  - `/rent-men` (conditional)
- **Admin routes**:
  - `/admin/dashboard`
  - `/admin/clients`
  - `/admin/content-review`
  - `/admin/appointments`
  - `/admin/messaging`
  - `/admin/billing`

API Design

- **Authentication endpoints**: Handled by Clerk

- **User endpoints**:
  - `GET /api/users/:id`
  - `PUT /api/users/:id`
  - `PATCH /api/users/:id/verification`

- **Onboarding endpoints**:
  - `POST /api/onboarding/step/:stepNumber`
  - `GET /api/onboarding/progress`

- **Content endpoints**:
  - `POST /api/content/upload` (integrates with Supabase Storage)
  - `GET /api/content/user/:userId`
  - `DELETE /api/content/:id`
- **Payment endpoints**:
  - `POST /api/payments/create-subscription`
  - `GET /api/payments/invoices`
  - `PUT /api/payments/update-method`
- **Appointment endpoints**:
  - `POST /api/appointments`
  - `GET /api/appointments/user/:userId`
  - `PUT /api/appointments/:id`
- **Notification endpoints**:
  - `POST /api/notifications/send` (integrates with Resend API)
  - `GET /api/notifications/user/:userId`
  - `PATCH /api/notifications/:id/read`
- **Messaging endpoints**:
  - `POST /api/messages`
  - `GET /api/messages/conversation/:conversationId`
  - `PATCH /api/messages/:id/read`

Database Design ERD (Supabase Tables)

- **Users Table**: (Managed by Clerk, with additional fields in Supabase)
  - `id` (PK, synced from Clerk)
  - `email` (synced from Clerk)
  - `role` (admin/client)
  - `created_at`
  - `updated_at`
  - `verification_status`
- **Profiles Table**:
  - `id` (PK)
  - `user_id` (FK)

  - `full_name`
  - `phone`
  - `preferred_contact_method`
  - `preferred_check_in_time`
  - `timezone`
  - `brand_description`
  - `voice_tone`
  - `do_not_say_terms`
- **Platform_Accounts Table**:
  - `id` (PK)
  - `user_id` (FK)
  - `platform_type` (OnlyFans, Rent.Men, etc.)
  - `username`
  - `needs_creation` (boolean)
  - `credentials` (encrypted)
- **Content_Strategy Table**:
  - `id` (PK)
  - `user_id` (FK)
  - `growth_goals` (JSON)
  - `content_types` (JSON)
  - `upload_frequency`
- **Media_Files Table**: (Links to Supabase Storage)
  - `id` (PK)
  - `user_id` (FK)
  - `storage_path`
  - `file_type`
  - `upload_date`
  - `status` (pending/approved/rejected)
- **Verification_Documents Table**: (Links to Supabase Storage)
  - `id` (PK)
  - `user_id` (FK)
  - `document_type` (ID front, ID back, selfie)
  - `storage_path`
  - `upload_date`
  - `verification_status`
- **Subscriptions Table**:

    - `id` (PK)
    - `user_id` (FK)
    - `plan_type`
    - `stripe_subscription_id`
    - `status`
    - `start_date`
    - `end_date`
- **Appointments Table**:
    - `id` (PK)
    - `user_id` (FK)
    - `client_name`
    - `appointment_date`
    - `location`
    - `details`
    - `status`
- **Messages Table**: (Using Supabase Realtime)
    - `id` (PK)
    - `conversation_id` (FK)
    - `sender_id` (FK)
    - `recipient_id` (FK)
    - `content`
    - `created_at`
    - `read_at`
    - `attachments` (JSON)
- **Conversations Table**:
    - `id` (PK)
    - `title`
    - `created_at`
    - `updated_at`
    - `last_message_preview`
- **Conversation_Participants Table**:
    - `id` (PK)
    - `conversation_id` (FK)
    - `user_id` (FK)
    - `joined_at`
- **Notifications Table**:

- `id` (PK)
- `user_id` (FK)
- `type`
- `content`
- `created_at`
- `read_at`
- `delivery_method` (email/in-app)

Security Requirements

- **Data Encryption**: All sensitive data encrypted at rest and in transit
- **Authentication**: Secure authentication via Clerk with MFA options
- **Authorization**: Role-based access control for all routes and data
- **File Validation**: Server-side validation of all uploaded files
- **Rate Limiting**: API rate limiting to prevent abuse
- **Input Sanitization**: All user inputs sanitized to prevent injection attacks
- **Audit Logging**: Comprehensive logging of all system access and changes
- **Compliance**: GDPR and CCPA compliance for user data handling

Performance Requirements

- **Page Load Time**: < 2 seconds initial load, < 500ms for subsequent interactions
- **API Response Time**: < 200ms for standard requests
- **Scalability**: Support for up to 10,000 concurrent users
- **File Upload**: Support for files up to 100MB with progress indicators
- **Mobile Optimization**: Optimized assets for mobile data usage
- **Offline Support**: Basic offline functionality for critical features
- **Real-time Messaging**: < 100ms delivery time for messages

Testing Strategy

- **Unit Testing**: Jest for frontend and backend components
- **Integration Testing**: Cypress for end-to-end testing
- **Performance Testing**: Lighthouse for performance metrics
- **Security Testing**: Regular penetration testing and vulnerability scanning
- **Usability Testing**: User testing sessions for key workflows

- **Cross-browser Testing**: Support for Chrome, Firefox, Safari, Edge