ROTATION - Technical Specification & System Architecture

Executive Summary

ROTATION is a mobile application designed to manage and gamify cannabis smoking sessions among friends. The app provides a structured rotation system with timers, notifications, and customizable features to ensure fair turn-taking during group sessions.

---

1. System Overview

1.1 Core Functionality

- **Session Management**: Create and join smoking sessions with unique codes
- **Rotation Timer**: Automated turn-based timing system
- **Master Blunt Agent (MBA)**: Session creator with administrative controls
- **Multi-Rotation Support**: Track multiple concurrent rotations
- **Real-time Synchronization**: Live updates across all participants
- **Customization**: Personalized alerts, sounds, and durations
- **Monetization**: Freemium model with premium features

1.2 Technology Stack Recommendations

Mobile Application

- **Framework**: React Native (cross-platform iOS/Android)
  - Alternative: Flutter (if team prefers Dart)
- **State Management**: Redux Toolkit or Zustand
- **Real-time**: Socket.io-client or Firebase SDK
- **Navigation**: React Navigation
- **UI Components**: React Native Paper or NativeBase
- **Audio**: react-native-sound or expo-av
- **Notifications**: React Native Push Notifications + Firebase Cloud Messaging

Backend Services

- **Runtime**: Node.js (Express.js or NestJS)
  - Alternative: Python (FastAPI) for better performance
- **API Architecture**: RESTful + WebSocket
- **Authentication**: Firebase Auth or Auth0
- **Real-time Engine**: Socket.io or Firebase Realtime Database
- **API Documentation**: Swagger/OpenAPI

Database & Storage

- **Primary Database**: PostgreSQL (relational data)
- **Cache Layer**: Redis (session state, real-time data)
- **File Storage**: AWS S3 or Cloudflare R2 (audio files, avatars)
- **Analytics Database**: MongoDB or ClickHouse

Infrastructure

- **Hosting**: AWS, Google Cloud, or DigitalOcean
- **Container Orchestration**: Docker + Kubernetes (optional for scale)
- **CDN**: Cloudflare or AWS CloudFront
- **CI/CD**: GitHub Actions or GitLab CI

Third-Party Services

- **Payment Processing**: Stripe
- **Push Notifications**: Firebase Cloud Messaging (FCM) + Apple Push Notification Service (APNS)
- **Analytics**: Mixpanel or Amplitude
- **Error Tracking**: Sentry
- **App Distribution**: TestFlight (iOS) + Google Play Console (Android)

---

2. System Architecture

2.1 High-Level Architecture

The system follows a **microservices-oriented architecture** with the following layers:

1. **Client Layer**: Mobile application (React Native)
2. **API Gateway**: Load balancing, rate limiting, authentication
3. **Service Layer**: Microservices for specific domains
4. **Real-time Layer**: WebSocket server for live updates
5. **Data Layer**: Databases, cache, and file storage
6. **External Services**: Third-party integrations

See diagram: `system-architecture.png`

2.2 Core Services

User Service

- User registration and authentication
- Profile management
- User preferences and settings
- Social features (friends, groups)

Session Service

- Create and manage smoking sessions
- Session code generation and validation
- Session lifecycle management
- Participant management

Rotation Service

- Timer logic and turn management
- Rotation state tracking
- Turn history and analytics
- Multi-rotation coordination

Notification Service

- Push notifications
- In-app alerts
- Sound/vibration triggers
- Custom notification scheduling

Payment Service

- Subscription management
- Payment processing
- Invoice generation
- Subscription tier enforcement

Content Service

- Custom sound library management
- Asset delivery (audio files)
- User-uploaded content moderation
- CDN integration

Analytics Service

- User behavior tracking
- Session statistics
- Engagement metrics
- Monetization analytics

---

3. Database Schema

3.1 Entity Relationship Diagram

See diagram: `database-schema.png`

3.2 Key Tables

USERS

- Core user authentication and identity
- Links to profiles, sessions, and subscriptions

SESSIONS

- Represents a smoking session
- Tracks Master Blunt Agent (MBA)

- Contains session settings and status

ROTATIONS

- Individual rotation within a session
- Manages timer settings and current turn
- Links to custom sounds and pass phrases

ROTATION_TURNS

- Historical record of each turn
- Tracks duration, timeouts, and skips
- Used for analytics and session summaries

SUBSCRIPTIONS

- User subscription status and tier
- Payment tracking
- Feature access control

CUSTOM_SOUNDS

- Library of alert sounds
- Categorized by free/premium
- Usage tracking for analytics

---

4. Mobile App Architecture

4.1 Application Flow

See diagram: `app-flow.png`

4.2 Key Screens

Authentication Flow

1. **Splash Screen**: App initialization
2. **Login/Signup**: Email, social, or phone authentication
3. **Onboarding**: First-time user tutorial

Main Application

1. **Home Dashboard**: Create or join sessions, view stats
2. **Session Setup**: Configure rotation settings (MBA only)
3. **Waiting Room**: Pre-session participant gathering
4. **Active Rotation**: Live timer and turn management
5. **Profile**: User stats, settings, and customization
6. **Premium Store**: Subscription plans and custom sounds

4.3 State Management

Global State (Redux/Zustand)

- User authentication state
- Current session data
- Active rotation state
- Subscription status
- App settings

Local State (Component)

- Form inputs
- UI toggles
- Temporary data

Persistent State (AsyncStorage)

- User preferences
- Cached session codes
- Offline data

---

5. Real-time Synchronization

5.1 WebSocket Events

Client → Server

- `join_session`: Join a session room

- `leave_session` : Leave a session room
- `start_rotation` : MBA starts the rotation
- `pass_turn` : User passes to next person
- `end_session` : MBA ends the session
- `update_settings` : MBA updates session settings

Server → Client

- `session_updated` : Session state changed
- `rotation_started` : Rotation has begun
- `turn_changed` : New person's turn
- `timer_alert` : Time limit reached
- `participant_joined` : New user joined
- `participant_left` : User left session
- `session_ended` : Session terminated

5.2 State Synchronization Strategy

- **Optimistic Updates**: Update UI immediately, sync in background
- **Conflict Resolution**: MBA has final authority on session state
- **Reconnection Handling**: Automatic reconnect with state recovery
- **Offline Support**: Queue actions when offline, sync when online

---

6. Key Features Implementation

6.1 Timer System

Timer Logic

Plain Text

```
1. MBA starts rotation
2. First participant's turn begins
3. Timer starts counting down from configured duration
4. At 80% of duration: Warning notification
5. At 100% of duration: Alert sound/vibration
6. User presses "Pass" button
```

```
7. Timer stops, next participant's turn begins
8. Repeat until session ends
```

Timer States

- **IDLE**: No active rotation
- **RUNNING**: Timer counting down
- **PAUSED**: Timer temporarily stopped
- **ALERT**: Time limit reached
- **COMPLETED**: Turn finished

6.2 Master Blunt Agent (MBA) Controls

- Create session and generate unique code
- Add/remove participants
- Set default duration for all turns
- Choose default alert sound
- Set pass phrase (optional)
- Start/pause/end rotation
- Skip a participant's turn
- Create multiple rotations (numbered blunts)

6.3 Customization Features

Free Tier

- Basic beep alert sound
- Standard timer durations (30s, 45s, 60s)
- Up to 10 participants per session
- Basic statistics

Premium Tier ($2.99/month or $19.99/year)

- Custom alert sounds library (50+ sounds)
- Upload personal audio files
- Custom timer durations (5s - 300s)
- Unlimited participants

- Advanced statistics and history
- Ad-free experience
- Priority support

6.4 Gamification Elements

- **Session Stats**: Total sessions, total turns, average turn time
- **Achievements**: "Speed Demon" (fastest passer), "Patient One" (longest holder)
- **Leaderboards**: Compare stats with friends
- **Badges**: Unlock badges for milestones
- **Profiles**: Customizable avatars and display names

---

7. Monetization Strategy

7.1 Revenue Streams

Subscription (Primary)

- **Free Tier**: Core functionality with ads
- **Premium Tier**: $2.99/month or $19.99/year
- **Lifetime Premium**: $49.99 one-time

In-App Purchases (Secondary)

- Custom sound packs ($0.99 - $4.99)
- Exclusive themes ($1.99)
- Avatar customization ($0.99)

Partnerships (Future)

- QR code partnerships with rolling paper brands
- Lighter manufacturer sponsorships
- Cannabis accessory affiliate marketing

7.2 QR Code Partnership Strategy

1. Partner with rolling paper, lighter, and packaging companies
2. Print QR codes on products linking to app download

3. Offer exclusive sounds or features for scanning partner QR codes
4. Track attribution and conversions
5. Revenue sharing or flat sponsorship fees

---

8. API Endpoints

8.1 Authentication

Plain Text

```
POST   /api/v1/auth/register
POST   /api/v1/auth/login
POST   /api/v1/auth/logout
POST   /api/v1/auth/refresh
POST   /api/v1/auth/forgot-password
POST   /api/v1/auth/reset-password
```

8.2 Users

Plain Text

```
GET    /api/v1/users/me
PUT    /api/v1/users/me
GET    /api/v1/users/:id
PUT    /api/v1/users/me/profile
GET    /api/v1/users/me/stats
```

8.3 Sessions

Plain Text

```
POST   /api/v1/sessions
GET    /api/v1/sessions/:id
PUT    /api/v1/sessions/:id
DELETE /api/v1/sessions/:id
POST   /api/v1/sessions/join
POST   /api/v1/sessions/:id/leave
GET    /api/v1/sessions/:id/participants
POST   /api/v1/sessions/:id/participants
DELETE /api/v1/sessions/:id/participants/:userId
```

8.4 Rotations

Plain Text

```
POST   /api/v1/sessions/:sessionId/rotations
GET    /api/v1/rotations/:id
PUT    /api/v1/rotations/:id
POST   /api/v1/rotations/:id/start
POST   /api/v1/rotations/:id/pause
POST   /api/v1/rotations/:id/end
POST   /api/v1/rotations/:id/pass
GET    /api/v1/rotations/:id/turns
GET    /api/v1/rotations/:id/history
```

8.5 Sounds

Plain Text

```
GET    /api/v1/sounds
GET    /api/v1/sounds/:id
POST   /api/v1/sounds/upload
GET    /api/v1/users/me/sounds
POST   /api/v1/users/me/sounds/:id/favorite
```

8.6 Subscriptions

Plain Text

```
GET    /api/v1/subscriptions/plans
POST   /api/v1/subscriptions/subscribe
GET    /api/v1/subscriptions/me
POST   /api/v1/subscriptions/cancel
POST   /api/v1/subscriptions/webhook
```

---

9. Security Considerations

9.1 Authentication & Authorization

- JWT-based authentication with refresh tokens
- Role-based access control (User, MBA, Admin)
- Session-level permissions (only MBA can modify session)

- Rate limiting on API endpoints

9.2 Data Protection

- Encrypt sensitive data at rest (passwords, payment info)
- HTTPS/TLS for all communications
- Secure WebSocket connections (WSS)
- Input validation and sanitization

9.3 Privacy

- GDPR/CCPA compliance
- User data deletion on request
- Anonymous analytics where possible
- Clear privacy policy and terms of service

---

10. Scalability Considerations

10.1 Performance Optimization

- Redis caching for session state
- CDN for static assets and audio files
- Database indexing on frequently queried fields
- Lazy loading and pagination

10.2 Horizontal Scaling

- Stateless API servers (scale horizontally)
- Load balancer for traffic distribution
- Separate WebSocket server cluster
- Database read replicas

10.3 Monitoring & Observability

- Application performance monitoring (APM)
- Error tracking and alerting
- Usage analytics and metrics

- Server health checks

---

11. Development Roadmap

Phase 1: MVP (Months 1-3)

- Basic authentication (email/password)
- Create and join sessions
- Single rotation with timer
- Basic alert sound (beep)
- iOS and Android apps

Phase 2: Core Features (Months 4-6)

- Master Blunt Agent controls
- Multiple rotations per session
- Custom timer durations
- Basic sound library (5-10 sounds)
- User profiles and stats

Phase 3: Monetization (Months 7-9)

- Subscription system integration
- Premium sound library
- Custom sound uploads
- Payment processing
- Ad integration for free tier

Phase 4: Growth (Months 10-12)

- QR code partnership program
- Social features (friends, groups)
- Advanced analytics
- Gamification (achievements, leaderboards)
- Marketing and user acquisition

Phase 5: Expansion (Year 2+)

- Web app version
- API for third-party integrations
- International expansion
- Advanced features based on user feedback

---

12. Testing Strategy

12.1 Unit Testing

- Jest for JavaScript/TypeScript
- Test coverage: 80%+ for critical paths
- Mock external dependencies

12.2 Integration Testing

- API endpoint testing
- Database integration tests
- WebSocket connection tests

12.3 End-to-End Testing

- Detox or Appium for mobile app testing
- Critical user flows (create session, join session, rotation)

12.4 User Acceptance Testing

- Beta testing with target users
- Feedback collection and iteration
- Performance testing under load

---

13. Deployment Strategy

13.1 Mobile App Distribution

- **iOS**: TestFlight (beta) → App Store (production)
- **Android**: Google Play Console (internal/beta/production tracks)

- App Store Optimization (ASO)

13.2 Backend Deployment

- Docker containerization
- CI/CD pipeline (GitHub Actions)
- Staging and production environments
- Blue-green deployment for zero downtime

13.3 Monitoring & Maintenance

- 24/7 uptime monitoring
- Automated alerts for critical errors
- Regular security updates
- Performance optimization

---

14. Legal & Compliance

14.1 Age Verification

- Require users to confirm 21+ age (or local legal age)
- Age gate on app launch
- Terms of service acceptance

14.2 Content Moderation

- User-uploaded content review (custom sounds)
- Reporting and blocking features
- Community guidelines

14.3 Disclaimers

- App is for legal cannabis use only
- Not responsible for illegal activities
- Use responsibly and follow local laws

---

15. Success Metrics

15.1 User Acquisition

- Downloads per month
- User registration rate
- Viral coefficient (invites sent)

15.2 Engagement

- Daily Active Users (DAU)
- Monthly Active Users (MAU)
- Average session duration
- Sessions per user per week

15.3 Monetization

- Conversion rate (free → premium)
- Monthly Recurring Revenue (MRR)
- Average Revenue Per User (ARPU)
- Churn rate

15.4 Technical

- API response time (<200ms)
- App crash rate (<1%)
- WebSocket connection stability (>99%)
- App store rating (>4.5 stars)

---

16. Cursor AI Development Guide

16.1 Project Structure

Plain Text

```
rotation-app/
├── mobile/                    # React Native mobile app
│   ├── src/
│   │   ├── components/        # Reusable UI components
│   │   ├── screens/           # App screens
│   │   ├── navigation/        # Navigation configuration
│   │   ├── services/           # API and WebSocket services
│   │   ├── store/              # State management (Redux/Zustand)
│   │   ├── utils/              # Helper functions
│   │   ├── hooks/              # Custom React hooks
│   │   └── assets/             # Images, sounds, fonts
│   ├── ios/                    # iOS native code
│   ├── android/                # Android native code
│   └── package.json
│
├── backend/                    # Node.js backend
│   ├── src/
│   │   ├── controllers/        # Request handlers
│   │   ├── services/           # Business logic
│   │   ├── models/             # Database models
│   │   ├── routes/             # API routes
│   │   ├── middleware/         # Express middleware
│   │   ├── websocket/          # WebSocket handlers
│   │   ├── utils/              # Helper functions
│   │   └── config/             # Configuration files
│   ├── tests/                  # Backend tests
│   └── package.json
│
├── database/                   # Database migrations and seeds
│   ├── migrations/
│   └── seeds/
│
├── docs/                       # Documentation
│   ├── api/                    # API documentation
│   └── architecture/           # Architecture diagrams
│
└── infrastructure/             # DevOps and infrastructure
    ├── docker/
    ├── kubernetes/
    └── terraform/
```

16.2 Development Workflow with Cursor AI

Step 1: Initialize Project

Bash

```
# Create React Native project
npx react-native init RotationApp --template react-native-template-typescript

# Create backend project
mkdir rotation-backend && cd rotation-backend
```

```
npm init -y
npm install express socket.io pg redis stripe firebase-admin
```

Step 2: Prompt Cursor AI for Component Generation

Example prompts:

- "Create a SessionSetup screen component with form inputs for duration, sound selection, and pass phrase"
- "Generate a Timer component that counts down and triggers alerts at 80% and 100%"
- "Build a WebSocket service that handles real-time session synchronization"
- "Create a PostgreSQL model for the Sessions table with Sequelize ORM"

Step 3: Implement Core Features

Use Cursor AI to generate:

1. Authentication flow (screens + API endpoints)
2. Session management (CRUD operations)
3. Real-time rotation logic (WebSocket handlers)
4. Timer system (React hooks + backend logic)
5. Payment integration (Stripe SDK)

Step 4: Testing & Debugging

- Use Cursor AI to generate unit tests
- Debug issues with AI-assisted code analysis
- Optimize performance with AI suggestions

16.3 Cursor AI Prompts for Key Features

Authentication

Plain Text

```
Create a complete authentication system for React Native with:
- Email/password login and registration screens
- Firebase Authentication integration
- JWT token management
- Secure token storage with AsyncStorage
- Auto-login on app launch
```

Session Management

Plain Text

```
Build a session management system with:
- Create session screen with session code generation
- Join session screen with code input
- Waiting room with participant list
- Real-time participant updates via WebSocket
- MBA controls for starting/ending sessions
```

Timer System

Plain Text

```
Implement a rotation timer system with:
- Countdown timer hook with start/pause/reset
- Visual timer display with progress bar
- Alert triggers at 80% and 100%
- Sound playback using react-native-sound
- Vibration using React Native Vibration API
- Automatic turn passing
```

Real-time Sync

Plain Text

```
Create a WebSocket service for real-time synchronization:
- Socket.io client integration
- Event handlers for session updates
- Automatic reconnection logic
- State synchronization with Redux
- Optimistic UI updates
```

---

17. Next Steps

Immediate Actions

1. **Set up development environment**: Install React Native, Node.js, PostgreSQL, Redis
2. **Initialize projects**: Create mobile and backend repositories
3. **Design UI mockups**: Use Figma or Adobe XD for screen designs

4. **Set up Firebase**: Create Firebase project for authentication and push notifications
5. **Database setup**: Create PostgreSQL database and run migrations

First Sprint Goals

1. Implement authentication flow
2. Create session creation and joining functionality
3. Build basic timer system
4. Set up WebSocket server for real-time updates
5. Test end-to-end flow with 2-3 users

Resources Needed

- **Developers**: 2-3 full-stack developers (React Native + Node.js)
- **Designer**: 1 UI/UX designer
- **Budget**: ~$5,000-10,000 for initial development (3 months)
- **Infrastructure**: AWS/DigitalOcean account (~$50-100/month)
- **Services**: Firebase, Stripe accounts

Contact & Support

For questions about this technical specification, contact the development team or refer to the project documentation.

Version: 1.0

Last Updated: December 11, 2025

Status: Draft for Development