ROTATION - Blunt Rotation Timer App

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.

Project Overview

ROTATION solves the common problem of people holding onto the blunt too long during group sessions by creating a structured, fair rotation system with automated timers, alerts, and real-time synchronization across all participants' devices.

Key Features

The application centers around several core features that enhance the group smoking experience. The **Session Management** system allows users to create smoking sessions with unique codes that friends can join instantly. Each session is controlled by a **Master Blunt Agent (MBA)** who creates the session and has administrative privileges to manage participants and settings.

The **Rotation Timer** is the heart of the application, providing an automated turn-based timing system that tracks how long each person has the blunt. When it's someone's turn, the timer counts down and triggers alerts at 80% and 100% of the configured duration. The app supports **Multiple Rotations**, allowing groups to track several blunts simultaneously during the same session.

Real-time Synchronization ensures that all participants see live updates across their devices through WebSocket connections. The app offers extensive **Customization** options, including personalized alert sounds, custom timer durations, and pass phrases. The business model follows a **Freemium Monetization** strategy with basic features available for free and premium features unlocked through subscription.

Documentation

This repository contains comprehensive documentation and architecture diagrams for building the ROTATION mobile app.

Architecture Diagrams

The project includes four key architecture diagrams that provide visual representations of the system design:

System Architecture (`system-architecture.png`) illustrates the complete system architecture showing the client layer (mobile app), API gateway, backend microservices, real-time layer with WebSocket server, data layer with databases and caching, and external service integrations. This high-level view demonstrates how all components interact within the system.

Database Schema (`database-schema.png`) presents the entity relationship diagram showing all database tables, their fields, relationships, and foreign key constraints. This diagram is essential for understanding the data model and implementing the database layer.

Application Flow (`app-flow.png`) maps out the complete user journey through the mobile application, from authentication through session creation, joining, active rotation, and premium features. This flowchart guides the implementation of navigation and user experience.

Rotation Sequence (`rotation-sequence.png`) provides a detailed sequence diagram showing the interaction between mobile clients, API gateway, backend services, WebSocket server, and database during a complete rotation cycle. This technical diagram is crucial for implementing the real-time rotation logic.

Technical Documentation

Technical Specification (`TECHNICAL_SPEC.md`) serves as the comprehensive technical specification document covering system overview, technology stack recommendations, architecture details, database schema, API endpoints, security considerations, scalability planning, development roadmap, testing strategy, deployment strategy, legal compliance, and success metrics. This document provides the complete blueprint for building the application.

Cursor AI Implementation Guide (`CURSOR_AI_GUIDE.md`) offers a step-by-step guide for using Cursor AI to build the ROTATION app, including detailed prompts for generating backend services, mobile app screens, API integration, WebSocket implementation, testing, and deployment. This guide accelerates development by providing ready-to-use AI prompts.

Technology Stack

Mobile Application

The mobile application is built using **React Native** with TypeScript for cross-platform iOS and Android development. State management is handled by **Redux Toolkit**, while **React Navigation** provides the navigation framework. The UI is constructed using **React Native Paper** components, with **Socket.io-client** enabling real-time communication. Audio playback utilizes **react-native-sound**, and push notifications are managed through

Firebase Cloud Messaging. Payment processing is integrated via the **Stripe React Native SDK**.

Backend Services

The backend runs on **Node.js** with **Express.js** or **NestJS** as the web framework. The API follows a **RESTful** architecture with **WebSocket** support through **Socket.io**. Authentication is provided by **Firebase Auth**, with **JWT** tokens for session management. API documentation is generated using **Swagger/OpenAPI**.

Database & Storage

PostgreSQL serves as the primary relational database for structured data. **Redis** provides caching and session state management for high-performance real-time operations. **AWS S3** or **Cloudflare R2** handles file storage for audio files and user avatars. **MongoDB** or **ClickHouse** stores analytics data for reporting and insights.

Infrastructure

The application is hosted on **AWS**, **Google Cloud**, or **DigitalOcean** with **Docker** containerization. **Kubernetes** provides optional container orchestration for scaling. **Cloudflare** or **AWS CloudFront** delivers content through a CDN. The CI/CD pipeline is implemented using **GitHub Actions** or **GitLab CI**.

Third-Party Services

Stripe handles payment processing and subscription management. **Firebase Cloud Messaging (FCM)** and **Apple Push Notification Service (APNS)** deliver push notifications. **Mixpanel** or **Amplitude** provides analytics and user behavior tracking. **Sentry** monitors errors and application health. App distribution uses **TestFlight** for iOS and **Google Play Console** for Android.

Getting Started

Prerequisites

Before beginning development, ensure you have the following tools installed: **Node.js** version 18 or higher, **npm** or **pnpm** for package management, **React Native CLI** for mobile development, **Xcode** for iOS development (macOS only), **Android Studio** for Android development, **PostgreSQL** version 14 or higher, **Redis** version 6 or higher, and **Docker** with **Docker Compose** for containerization.

Installation

To set up the development environment, first clone the repository and navigate to the project directory. For backend setup, navigate to the backend directory, install dependencies using npm, copy the environment example file to create your `.env` file, configure your environment variables, run database migrations, seed the database with initial data, and start the development server.

For mobile app setup, navigate to the mobile directory, install dependencies, copy the environment example file, configure your environment variables, install iOS dependencies using CocoaPods (iOS only), and start the Metro bundler. You can then run the app on iOS using `npm run ios` or on Android using `npm run android`.

Quick Start with Cursor AI

To accelerate development using Cursor AI, open the project in Cursor IDE and refer to the `CURSOR_AI_GUIDE.md` file. Use the provided prompts to generate backend services, mobile app components, API endpoints, WebSocket handlers, and tests. Follow the step-by-step implementation guide organized by feature area.

Development Roadmap

Phase 1: MVP (Months 1-3)

The initial MVP phase focuses on core functionality including basic authentication with email and password, the ability to create and join sessions, single rotation with timer functionality, basic alert sounds, and deployment to both iOS and Android platforms.

Phase 2: Core Features (Months 4-6)

The second phase introduces Master Blunt Agent controls, support for multiple rotations per session, custom timer durations, a basic sound library with five to ten sounds, and user profiles with statistics tracking.

Phase 3: Monetization (Months 7-9)

The monetization phase implements the subscription system integration, premium sound library, custom sound upload functionality, payment processing through Stripe, and ad integration for free tier users.

Phase 4: Growth (Months 10-12)

The growth phase launches the QR code partnership program, adds social features including friends and groups, implements advanced analytics, introduces gamification elements such as achievements and leaderboards, and executes marketing and user acquisition campaigns.

Phase 5: Expansion (Year 2+)

Future expansion includes developing a web app version, creating an API for third-party integrations, pursuing international expansion, and adding advanced features based on user feedback and market demands.

API Endpoints

Authentication Endpoints

The authentication system provides endpoints for user registration, login, logout, token refresh, password reset requests, and password resets. These endpoints handle user identity management and session control.

User Endpoints

User management endpoints allow retrieving and updating the current user's profile, viewing other user profiles, updating profile information, and accessing user statistics.

Session Endpoints

Session management includes creating new sessions, retrieving session details, updating session settings (MBA only), deleting sessions (MBA only), joining sessions with codes, leaving sessions, managing participants, adding participants (MBA only), and removing participants (MBA only).

Rotation Endpoints

Rotation management provides endpoints for creating rotations within sessions, retrieving rotation details, updating rotation settings (MBA only), starting rotations (MBA only), pausing rotations (MBA only), ending rotations (MBA only), passing turns to the next person, viewing turn history, and accessing rotation action history.

Sound Endpoints

The sound library system includes endpoints for browsing all sounds with filtering, retrieving sound details, uploading custom sounds (premium only), viewing user's personal sounds, and favoriting sounds.

Subscription Endpoints

Subscription management handles viewing available subscription plans, creating subscriptions, retrieving user subscription status, canceling subscriptions, and processing Stripe webhooks for payment events.

Database Schema

Core Tables

The database schema consists of several interconnected tables that form the foundation of the application. The **users** table stores core user authentication and identity information. The **user_profiles** table contains extended profile information including display names, avatars, preferences, and statistics.

The **sessions** table represents smoking sessions with information about the Master Blunt Agent, session codes, status, and settings. The **session_participants** table tracks which users are in which sessions and their join order.

The **rotations** table manages individual rotations within sessions, including timer settings, current turn tracking, and custom configurations. The **rotation_turns** table provides a historical record of each turn with duration tracking and timeout information. The **rotation_history** table logs all actions taken during rotations for analytics and auditing.

The **subscriptions** table tracks user subscription status, tier, payment information, and expiration dates. The **custom_sounds** table stores the library of alert sounds with categorization and usage tracking. The **user_sounds** table creates the many-to-many relationship between users and sounds, tracking favorites and personal collections.

Security & Privacy

Authentication & Authorization

The application implements JWT-based authentication with refresh tokens for secure session management. Role-based access control distinguishes between regular users, Master Blunt Agents, and administrators. Session-level permissions ensure that only the MBA can modify session settings. Rate limiting on API endpoints prevents abuse and ensures system stability.

Data Protection

All sensitive data is encrypted at rest, including passwords and payment information. All communications use HTTPS/TLS encryption for data in transit. WebSocket connections are secured using WSS protocol. Input validation and sanitization protect against injection attacks and malicious input.

Privacy Compliance

The application complies with GDPR and CCPA regulations for data protection. Users can request deletion of their data at any time. Analytics are anonymized where possible to protect user privacy. Clear privacy policies and terms of service inform users about data collection and usage.

Monetization Strategy

Subscription Tiers

The **Free Tier** provides basic functionality with a simple beep alert sound, standard timer durations of 30, 45, or 60 seconds, support for up to 10 participants per session, basic statistics, and includes advertisements.

The **Premium Tier** costs $2.99 per month or $19.99 per year and unlocks a custom alert sounds library with over 50 sounds, the ability to upload personal audio files, custom timer durations from 5 to 300 seconds, unlimited participants per session, advanced statistics and history, an ad-free experience, priority support, and early access to new features.

The **Lifetime Premium** option provides all premium features for a one-time payment of $49.99, offering the best value for long-term users.

Partnership Revenue

The QR code partnership program creates additional revenue streams by partnering with rolling paper, lighter, and packaging companies. QR codes printed on products link directly to app downloads, with exclusive sounds or features unlocked by scanning partner codes. Attribution tracking measures conversion rates, and revenue is generated through revenue sharing agreements or flat sponsorship fees.

Testing

Testing Strategy

The testing approach encompasses multiple levels of validation to ensure application quality. **Unit tests** cover individual components, services, and functions with a target of 80% code coverage. **Integration tests** verify that API endpoints, database operations, and WebSocket connections work correctly together. **End-to-end tests** validate complete user flows from authentication through rotation completion. **User acceptance testing** involves beta testing with target users to gather feedback and identify issues before launch.

Test Coverage Goals

Critical paths such as authentication, session creation, rotation timer logic, and payment processing require 100% test coverage. Overall code coverage targets 80% or higher. Edge cases and error handling scenarios are thoroughly tested to ensure robustness.

Deployment

Backend Deployment

The backend is containerized using Docker with a docker-compose configuration that includes the Node.js application, PostgreSQL database, Redis cache, and Nginx reverse proxy. The CI/CD pipeline built with GitHub Actions automatically runs tests, builds Docker images, pushes to the container registry, deploys to servers, runs database migrations, and performs health checks after deployment.

Mobile App Deployment

iOS deployment follows the standard Apple workflow through Xcode project configuration, App Store Connect setup, TestFlight beta testing, and App Store submission with required screenshots and app preview videos. Android deployment uses the Google Play Console with internal, beta, and production tracks for staged rollouts. Fastlane automates the build and deployment process for both platforms, including screenshot generation and submission workflows.

Success Metrics

User Acquisition Metrics

Success in user acquisition is measured through monthly downloads, user registration rates, and viral coefficient tracking how many invites each user sends to friends.

Engagement Metrics

User engagement is evaluated using Daily Active Users (DAU), Monthly Active Users (MAU), average session duration, and the number of sessions per user per week.

Monetization Metrics

Financial success is tracked through conversion rates from free to premium tiers, Monthly Recurring Revenue (MRR), Average Revenue Per User (ARPU), and churn rate monitoring.

Technical Metrics

Technical performance is measured by API response times targeting under 200 milliseconds, app crash rates below 1%, WebSocket connection stability above 99%, and app store ratings above 4.5 stars.

Contributing

Contributions to the ROTATION project are welcome. When contributing, please follow the coding standards established in the project, write tests for new features, update documentation to reflect changes, follow the Git workflow with feature branches, and submit pull requests for review before merging.

License

Contact

For questions, support, or partnership inquiries, please contact the development team through the project repository or official communication channels.

Acknowledgments

This project was conceived by a group of friends who identified a common problem in social smoking sessions and envisioned a technological solution. The architecture and implementation guide were created to facilitate rapid development using modern tools like Cursor AI.

Version: 1.0

Last Updated: December 11, 2025

Status: Architecture & Planning Phase