Movie Ticket Management System - Problem Statement

Intern: Shashank

Duration: 1 Month

Project Type: Full-Stack Web Application with AI Integration

Project Overview

Design and develop a comprehensive Movie Ticket Management System that serves two primary user types: Theater Owner/Manager and Customers. This project will test your ability to make architectural decisions, implement full-stack features, integrate AI capabilities, and document your technical choices through Architecture Decision Records (ADRs).

Phase I: Planning & Foundation

1.1 Architectural Decision Records (ADR)

Before beginning implementation, you must create detailed ADRs documenting your technology choices. Each ADR should include:

- **Context:** Why this decision is needed
- **Options Considered:** Alternative technologies evaluated
- **Decision:** The chosen technology
- **Rationale:** Detailed reasoning behind the choice
- **Consequences:** Benefits and trade-offs

Required ADR Topics:

- **Backend Framework Selection:** FastAPI vs Node.js (Express/NestJS). Consider: Performance, type safety, async capabilities, ecosystem maturity, documentation quality

• **Programming Language for Backend:** Python vs JavaScript/TypeScript. Consider: Development speed, type safety, library ecosystem, team expertise, scalability, AI integration capabilities

• **Database Selection:** MongoDB vs SQL (PostgreSQL/MySQL). Consider: Data structure, relationships, scalability, ACID properties, query complexity, booking system requirements

• **Frontend Framework:** React vs Vue vs Angular vs Svelte. Consider: Component architecture, state management, learning curve, ecosystem maturity, ticket booking UI/UX requirements

• **Frontend Language:** TypeScript vs JavaScript. Consider: Type safety, developer experience, code maintainability, error detection, IDE support, team expertise

• **State Management Solution:** Redux vs Zustand vs Context API vs Recoil vs Jotai. Consider: Complexity, boilerplate, learning curve, DevTools, scalability, performance

• **UI Component Library:** Material-UI (MUI) vs Ant Design vs Chakra UI vs shadcn/ui vs Tailwind CSS + Headless UI. Consider: Design consistency, customization, component variety, bundle size, documentation

• **CSS/Styling Approach:** Tailwind CSS vs CSS Modules vs Styled Components vs Emotion vs SASS. Consider: Developer experience, bundle size, reusability, maintainability, design system compatibility

• **Frontend Folder Structure:** Feature-based vs Layer-based vs Domain-driven. Consider: Scalability, maintainability, team collaboration, code organization, separation of concerns

• **Authentication Strategy:** JWT vs Session-based vs OAuth2. Consider: Stateless vs stateful, security, scalability, token management, session handling, Google OAuth implementation requirements

• **AI Model Selection:** OpenAI vs Anthropic vs Google Gemini vs Open-source LLMs. Consider: Cost, latency, capabilities, privacy, rate limits, prompt engineering requirements

• **Voice AI Framework Selection:** PipeCat vs LiveKit vs Vapi vs Retell AI vs Custom implementation. Consider: Real-time performance, ease of integration, cost, documentation, latency requirements

• **STT Provider Selection:** ElevenLabs vs Deepgram vs OpenAI vs Cartesia vs Google Speech. Consider: Speech recognition accuracy, Indian English accent support, code-switching (English-Hindi), latency, cost per minute

• **Voice Architecture:** Speech-to-Speech (Direct) vs Cascaded (STT → LLM → TTS). Consider: End-to-end latency, debugging complexity, control flexibility, cost implications, conversation naturalness

1.2 System Architecture Design

Create detailed architecture diagrams showing:

• System components and their interactions

- Data flow between frontend, backend, database, and AI services
- Authentication and authorization flow
- Payment processing flow
- Voice AI integration architecture

1.3 Database Schema Design

Design comprehensive schema/collections for:

- Users (Theater Owners and Customers)
- Theaters and Screens
- Movies and Showtimes
- Bookings and Seats
- Payments and Transactions
- AI conversation logs

1.4 Frontend Architecture Design

Create detailed frontend architecture documentation including:

Tech Stack Documentation:

- Chosen frontend framework and version
- Programming language (TypeScript/JavaScript)
- State management solution
- UI component library
- CSS/styling approach
- Build tool and bundler configuration

Folder Structure Design:

Document your chosen folder structure with justification. Examples to consider:

Feature-based Structure:

```
src/
■■■ features/
■ ■■■ auth/
■ ■ ■■■ components/
```

■ ■ ■■■ hooks/

■ ■ ■■■ services/

■ ■ ■■■ types/

■ ■■■ theaters/

■ ■■■ bookings/

■ ■■■ ai-chat/

■■■ shared/

■ ■■■ components/

■ ■■■ utils/

■ ■■■ types/

■■■ store/

```

**Layer-based Structure:**

```

src/

■■■ components/

■■■ pages/

■■■ services/

■■■ hooks/

■■■ store/

■■■ utils/

■■■ types/

```

**State Management Architecture:**

- Global state structure
- Local vs global state decisions
- State persistence strategy
- API state management (React Query, SWR, Redux Toolkit Query)

**Component Architecture:**

- Component composition patterns
- Reusable component library
- Page components vs feature components
```

- Smart vs presentational components

Routing Strategy:

- Route structure and organization
- Protected routes implementation
- Role-based route access
- Nested routing for complex features

Phase II: Core Application Development

2.1 User Authentication & Authorization

Requirements:

- User registration and login for both user types (Theater Owner and Customer)
- **Google OAuth integration (mandatory)**
- Traditional email/password authentication
- Email verification with OTP
- Password reset functionality
- JWT-based authentication
- Role-based access control (RBAC)
- Protected routes and API endpoints
- Session management

Authentication Methods:

- Google OAuth integration
- Email/password authentication with OTP verification

Security Considerations:

- Password hashing and salting
- JWT token management
- OAuth 2.0 implementation
- Role-based access control (RBAC)
- Input validation and sanitization

User Roles:

- **Theater Owner/Manager**: Can add theaters, movies, showtimes, manage bookings
- **Customer**: Can browse movies, book tickets, view booking history

2.2 Theater Management Module (Theater Owner)

Requirements:

- Add/Edit/Delete theaters
- Add/Edit/Delete screens within theaters
- Set seating arrangements (rows, columns, seat types)
- Add movies to theaters
- Create and manage showtimes
- View booking analytics
- Manage pricing (base price, seat type multipliers)

2.3 Customer Booking Module

Requirements:

- Browse available movies
- Search and filter movies (genre, language, date, location)
- View theater details and showtimes
- Select seats with real-time availability
- View seat map with different seat types (Regular, Premium, Recliner)
- Booking cart functionality
- Booking confirmation and ticket generation

2.4 Server-Side Implementation

Requirements:

- RESTful API design with proper HTTP methods
- Input validation and sanitization
- Error handling and logging
- Rate limiting
- CORS configuration
- API documentation (Swagger/OpenAPI)
- Database connection pooling

- Transaction management for bookings

Phase III: Payment Integration & Notifications

3.1 Payment Gateway Simulation

Requirements:

- Simulate payment flow (no actual payment gateway integration)
- OTP-based payment verification
- Multiple payment method options (Card, UPI, Wallet - UI only)
- Payment status tracking (Pending, Success, Failed)
- Transaction history
- Booking confirmation only after successful payment
- Automatic seat release on payment failure/timeout

3.2 Email Notification System

Requirements:

- Email service integration
- OTP emails for:
- Registration verification
- Payment verification
- Password reset
- Booking confirmation emails with ticket details
- Booking cancellation emails
- Email templates with branding

3.3 Booking Management

Requirements:

- View booking history
- Cancel bookings (with refund simulation)
- Download/Print ticket

- QR code generation for tickets
- Booking status tracking

Phase IV: AI Co-Pilot Integration

4.1 Theater Management AI Co-Pilot

Requirements:

- Natural language interface for theater owners
- AI-powered assistance for:
- Adding new movies and showtimes
- Analyzing booking patterns
- Suggesting optimal pricing
- Predicting popular showtimes
- Generating reports and insights
- Answering questions about bookings and revenue

Capabilities to Implement:

- Chat interface integrated into theater owner dashboard
- Context awareness of current theater data
- Ability to execute actions (with confirmation)
- Data visualization recommendations
- Conversational analytics queries

4.2 Customer Support AI Assistant

Requirements:

- AI chatbot for customer queries
- Capabilities:
- Movie recommendations based on preferences
- Booking assistance
- FAQ handling
- Booking status queries
- Refund and cancellation support

Phase V: Voice AI Integration

5.1 Voice AI Framework Research & Selection

Research Requirements:

- Evaluate available voice AI frameworks and architectures
- Consider two primary approaches:
- **Speech-to-Speech (Direct)**: Real-time audio input to audio output
- **Cascaded (STT → LLM → TTS)**: Traditional pipeline with separate components
- Evaluation criteria:
- Real-time speech-to-text capabilities
- Text-to-speech quality and latency
- End-to-end latency comparison
- Multi-language support (especially English and Hindi)
- Integration complexity
- Cost implications
- Scalability
- Frameworks to research: PipeCat, Vapi, Bland AI, Twilio Voice, custom WebRTC solutions, OpenAI Realtime API

Create ADR documenting:

- Speech-to-Speech vs Cascaded approach comparison
- Frameworks evaluated for each approach
- Latency benchmarks and performance comparison
- Selected framework/architecture and rationale
- Integration architecture diagram
- Expected challenges and mitigation strategies

5.2 Speech-to-Text for Text Chat Voice Announcement

Core Requirement:

Integrate speech-to-text capabilities to enable voice announcement of text chat messages, providing an accessible and enhanced chat experience.

Requirements:

- **Text Chat Voice Reading**: Convert AI co-pilot text responses to speech for audio playback
- **Customer Support Chat**: Enable voice output for all chat interactions
- **User Controls**:
- Toggle between text-only and voice-announced modes
- Play, pause, stop, and replay controls for each message
- Adjust speech rate and voice selection
- Volume control
- **Visual Feedback**:
- Speaking indicator showing which text is currently being read
- Highlight text being spoken in real-time (karaoke-style)
- **Queue Management**: Sequential playback of multiple messages
- **Background Reading**: Allow voice reading while user continues browsing

5.3 Voice AI Integration

Core Requirement:

Integrate voice AI capabilities to enable customers to interact with the system through voice commands in addition to text-based chat. The system should support real-time voice conversations with natural speech interaction.

Voice AI Architecture Components:

Your chosen Voice AI framework will integrate the following components:

- **Speech-to-Text (STT)**: Convert customer voice input to text for processing
- **Text-to-Speech (TTS)**: Convert AI responses back to natural-sounding speech
- **Chat Model**: Process the transcribed text and generate intelligent responses
- **Voice Framework**: Orchestrate the entire voice conversation flow

Technology Stack Selection:

Research and select the best Voice AI framework that integrates STT, TTS, and conversational AI. Consider frameworks like:

- **Recommended Options**: PipeCat, LiveKit, Vapi, Retell AI, or similar voice AI orchestration frameworks
- **Evaluation Criteria**: Real-time latency, ease of integration, API reliability, cost, documentation quality, Indian English accent support
- **ADR Required**: Document why you chose a specific voice framework in your ADR documentation

Voice AI Implementation Requirements:

- **Real-time Voice Conversation**: Support live, bidirectional voice communication with minimal latency (<1s target)
- **Speech Recognition Accuracy**: High-quality STT with support for Indian English accents and code-switching (English-Hindi)
- **Natural Voice Output**: Use high-quality TTS with natural-sounding voices and appropriate emotional tone
- **Conversation Flow**: Handle interruptions, clarifications, and multi-turn voice dialogues
- **Voice Commands**: Support voice-based:
- Movie search and browsing
- Showtime selection
- Seat selection assistance
- Booking confirmation
- Theater management commands (for accessibility)
- **Fallback Handling**: Gracefully handle speech recognition errors with retry mechanisms

Integration Points:

- **Chat Model Integration**: Connect your voice framework with your chosen AI model (GPT-4/Claude/Gemini)
- **Context Preservation**: Maintain conversation context across voice interactions
- **Multi-modal Support**: Allow users to seamlessly switch between text chat and voice
- **Session Management**: Handle voice session timeouts and reconnections

Voice User Experience Design:

- Clear audio cues for when AI is listening vs speaking
- Visual indicators showing voice transcription in real-time
- Push-to-talk or always-on listening mode options
- Background noise handling and echo cancellation
- Voice conversation history alongside text chat

Architecture Considerations:

- **Speech-to-Speech**: Lower latency, more natural conversations, unified processing (but less control, harder debugging)
- **Cascaded (STT $\rightarrow$ LLM $\rightarrow$ TTS)**: More flexibility, easier debugging, established patterns (but higher latency, multiple API calls)
- Document trade-offs and justify chosen approach in ADR

Note on Framework Selection:

If you're unsure which Voice AI framework to use, you will be provided with a recommended easy-to-use framework during the implementation phase. However, you should still research options and document your reasoning in your ADR.

Technical Requirements Summary

Mandatory Features:

1. Dual user authentication (Theater Owner & Customer)
2. **Google OAuth integration (mandatory)**
3. Email/password authentication with OTP verification
4. Theater and movie management (Owner)
5. Showtime and seat management
6. Customer booking system with cart management
7. Simulated payment processing with OTP
8. Email notification system
9. AI-powered co-pilot for theater management
10. AI-powered customer support assistant
11. Voice AI integration with STT and framework selection
12. Text chat voice announcement feature
13. Comprehensive ADR documentation (including Voice AI framework and STT provider selection)

Optional Enhancements:

- Image upload for movies and theaters
- Rating and review system
- Advanced seat selection with different seat types (Regular, Premium, Recliner)
- Real-time seat availability updates using WebSockets
- Mobile-responsive design and PWA features
- Multi-language support for the entire application
- Advanced analytics dashboard for theater owners
- Voice biometrics for authentication
- Code-switching support (English-Hindi) in voice AI
- Dynamic pricing based on demand
- Loyalty program and discount system

Technical Requirements

Performance

- API response time < 500ms for non-AI endpoints
- AI response latency < 2s for text interactions
- Voice AI response latency < 1s
- Support concurrent bookings with race condition handling
- Database query optimization

Security

- Password hashing with salt
- SQL/NoSQL injection prevention
- XSS and CSRF protection
- Rate limiting on authentication endpoints
- Secure session management
- Environment variable management for secrets
- HTTPS enforcement (in production setup)

Code Quality

- Clean code principles
- Proper error handling
- Logging and monitoring
- Unit tests for critical functions
- Integration tests for API endpoints
- Code documentation
- Git version control with meaningful commits

Deployment Preparation

- Docker containerization
- Environment-based configuration
- Database migrations/seeding scripts
- README with setup instructions
- API documentation

Deliverables

Phase I Deliverables

1. All ADRs documented (Backend, Database, Frontend Framework, Frontend Language, State Management, UI Library, Styling, Folder Structure, Auth, AI Model, Voice AI Framework, STT Provider, Voice Architecture)
2. System architecture diagrams with complete data flow
3. Database schema design with relationships
4. **Frontend architecture documentation:**
    - Tech stack selection with justification
    - Folder structure design
    - State management architecture
    - Component architecture patterns
    - Routing strategy
5. Project setup with basic structure and folder organization
6. Initial README documentation with setup instructions

Phase II Deliverables

1. Complete authentication system with Google OAuth and JWT
2. Email/password authentication with OTP verification
3. Theater management module with full CRUD operations
4. Customer booking module (without payment integration)
5. Comprehensive API documentation (Swagger/Postman)
6. Functional frontend for both Theater Owner and Customer roles

Phase III Deliverables

1. Payment simulation integration with OTP verification
2. Email notification system for all user actions
3. Booking management features (view, cancel, download tickets)
4. Transaction history and refund simulation
5. QR code generation for tickets

Phase IV Deliverables

1. AI co-pilot for theater management with conversational interface
2. Customer support AI assistant with context awareness
3. Natural language query processing for analytics
4. AI-powered recommendations and insights
5. Integration testing of AI features

Phase V Deliverables

1. Voice AI framework ADR with Speech-to-Speech vs Cascaded comparison
2. STT provider ADR with accent support and latency analysis
3. Speech-to-text integration for text chat voice announcement
4. Complete Voice AI integration (STT + TTS + Chat Model orchestration)
5. Voice-enabled booking flow (end-to-end)
6. Text chat with voice announcement feature (play, pause, replay controls)
7. Multi-modal experience (seamless text and voice switching)
8. Final documentation, testing, and demo preparation

Evaluation Criteria

Technical Excellence (35%)

- Code quality and organization
- Architecture design decisions (backend + frontend)
- Frontend tech stack selection and justification
- Folder structure and code organization
- Security implementation
- Database design
- API design
- Performance optimization

ADR Quality (20%)

- Thoroughness of analysis for all tech stack decisions
- Clear reasoning and justification for Frontend Framework, Language, State Management, UI Library, and Styling choices
- Clear reasoning for Voice AI framework and STT provider selection
- Consideration of trade-offs across all decisions
- Professional documentation

Feature Completeness (20%)

- All mandatory features implemented
- Working end-to-end flows
- Error handling
- Edge case management

AI & Voice Integration (15%)

- Effectiveness of conversational interface (text)
- Voice AI quality and responsiveness
- STT accuracy for text chat voice announcement
- Framework integration quality
- Multi-modal user experience

Innovation & Polish (10%)

- User interface quality
- Additional features
- Performance optimization (voice latency)
- Deployment readiness

Submission Guidelines

Final Submission Must Include:

1. Complete source code repository

2. ADR documentation folder (including all backend and frontend tech stack decisions, Voice AI framework and STT provider selection)

3. Frontend architecture documentation (tech stack, folder structure, state management, component patterns)

4. README with setup instructions

5. API documentation

6. Database schema documentation

7. Environment configuration guide

8. Demo video (10-15 minutes) showing text chat voice announcement AND voice AI integration

9. Deployment instructions

Code Quality Standards:

- Proper code comments
- Consistent naming conventions
- Error handling and logging
- Input validation
- Unit tests (bonus)
- Git commit message standards

Support & Guidance

- Regular check-in meetings for progress review
- Access to technical mentorship
- Code review sessions at phase completions
- Clarification on requirements as needed
- Recommended Voice AI framework will be provided if needed

Important Notes

- **Google OAuth is mandatory:** Must implement Google OAuth integration for user authentication.
- **Frontend tech stack decisions are critical:** Document all frontend choices (Framework, Language, State Management, UI Library, Styling, Folder Structure) in ADRs with clear justification.
- **DO NOT integrate external payment gateways.** Create internal payment simulation only.

- **Voice AI is mandatory:** Must implement voice AI integration with proper framework selection.
- **Text chat voice announcement:** STT integration for reading chat messages aloud is required.
- **Document your Voice AI choice:** Include Voice AI framework and STT provider selection in your ADR documentation with clear reasoning.
- **Test voice edge cases:** Background noise, accent variations, interruptions, multi-turn conversations.
- **AI integration:** Ensure AI can interact through both text and voice modalities.
- **Optimize voice latency:** Target sub-1s end-to-end response time for voice interactions.
- **Folder structure consistency:** Maintain consistent folder structure throughout the project as documented in Phase I.

Good luck with your project! Focus on creating a well-architected, secure system with high-quality Voice AI integration and user-friendly interface while documenting your decisions thoroughly.

End of Problem Statement