**Infosys Limited**

**Movie Production Management System**

**Requirements Specification Document**

24-03-2025

**REVISION LIST**

| Ver.Rev | Date | Authors | Description |
| --- | --- | --- | --- |
| V1.0 | 24-03-2025 | soorya.xavier | User stories and requirements for the Movie Production Management System |

**TABLE OF CONTENTS**

1. PROJECT DESCRIPTION 3

2. USER STORIES 3

3. FRONTEND COMPONENTS 7

4. ENVIRONMENT REQUIREMENTS 11

# **PROJECT DESCRIPTION**

This internship project aims at developing a centralized platform for film production companies. The system will help manage projects, coordinate cast and crew, track budgets, schedule shoots, manage assets, and facilitate team communication. By addressing challenges like resource allocation, scheduling conflicts, and budget tracking, the system will improve efficiency and collaboration throughout the production process.

# **USER STORIES**

|  |  |
| --- | --- |
| **US 01** | Priority: Must Have |
| **User Story Description** | |
| As a production manager, I want to create a new movie project with basic details (title, genre, budget, timeline) so that I can establish the foundation for production planning. | |
| **Acceptance Criteria** | |
| * System allows creation of new project with required fields * Project dashboard is generated upon creation * Ability to assign key team members to the project * Option to import project template or start from scratch | |
|
|
|
|
|

|  |  |
| --- | --- |
| **US 02** | Priority: Must Have |
| **User Story Description** | |
| As a production manager, I want to create, track, and manage the production budget so that I can ensure the project stays financially on track. | |
| **Acceptance Criteria** | |
| * Detailed budget creation with categories and line items * Real-time expense tracking against budget * Variance reporting and alerts for over-budget items * Budget revision history and comparison features * Export functionality for financial reports | |
|
|
|

|  |  |
| --- | --- |
| **US 03** | Priority: Must Have |
| **User Story Description** | |
| As a production coordinator, I want to manage cast and crew information, contracts, and availability so that I can efficiently organize the production team. | |
| **Acceptance Criteria** | |
| * Database of cast and crew with contact information and roles * Contract status tracking and notification system * Availability calendar for each team member * Document storage for signed agreements * Search and filter functionality for quick access | |
|
|

|  |  |
| --- | --- |
| **US 04** | Priority: Must Have |
| **User Story Description** | |
| As an assistant director, I want to create and manage a detailed shooting schedule based on script breakdown so that I can optimize shooting days and resources. | |
| **Acceptance Criteria** | |
| * Integration with script breakdown elements * Drag-and-drop schedule builder with day/night designation * Resource allocation visualization * Weather forecast integration * Automatic shooting day calculations * Schedule conflict detection | |
|
|
|

|  |  |
| --- | --- |
| **US 05** | Priority: Must Have |
| **User Story Description** | |
| As an assistant director, I want to upload, manage versions, and break down scripts into scenes, locations, and elements so that other departments can plan accordingly. | |
| **Acceptance Criteria** | |
| * Script upload and version control * Scene numbering and categorization * Element tagging (props, wardrobe, cast, special effects) * Revision tracking with highlighted changes * Export functionality for breakdown reports | |
|
|
|

|  |  |
| --- | --- |
| **US 06** | Priority: Must Have |
| **User Story Description** | |
| As a production coordinator, I want to catalog potential filming locations with details, permits, and availability so that I can efficiently plan the shooting schedule. | |
| **Acceptance Criteria** | |
| * Location database with photos, address, and contact information * Permit status tracking and renewal alerts * Location availability calendar * Cost tracking per location * Map integration for geographic visualization | |
|
|

|  |  |
| --- | --- |
| **US 07** | Priority: Must Have |
| **User Story Description** | |
| As a production coordinator, I want to track all physical assets (props, costumes, equipment) so that I can ensure availability when needed for shooting. | |
| **Acceptance Criteria** | |
| * Inventory management system for all assets * Check-in/check-out functionality * Asset assignment to scenes and shooting days * Barcode/QR code scanning support * Maintenance scheduling for equipment | |
|
|
|

|  |  |
| --- | --- |
| **US 08** | Priority: Must Have |
| **User Story Description** | |
| As a production manager, I want to generate and distribute daily progress reports so that stakeholders are informed about shooting progress and issues. | |
| **Acceptance Criteria** | |
| * Automated report templates with customizable fields * Scene completion tracking * Issue/problem logging * Distribution list management * Historical report access and search | |
|
|
|

|  |  |
| --- | --- |
| **US 09** | Priority: Must Have |
| **User Story Description** | |
| As an assistant director, I want to generate and distribute call sheets so that cast and crew know when and where to report for work. | |
| **Acceptance Criteria** | |
| * Automated call sheet creation from schedule * Individualized call times for cast/crew * Digital distribution via email/SMS * Confirmation tracking for receipt * Weather and special instructions inclusion | |
|
|
|

|  |  |
| --- | --- |
| **US 10** | Priority: Must Have |
| **User Story Description** | |
| As a production manager, I want to create, assign, and track tasks for my team members so that I can ensure all departmental responsibilities are fulfilled on time. | |
| **Acceptance Criteria** | |
| * Task creation with deadlines and priorities * Assignment to individuals or groups * Status tracking and progress updates * Dependency linking between tasks * Notification system for upcoming deadlines | |
|
|
|

|  |  |
| --- | --- |
| **US 11** | Priority: Must Have |
| **User Story Description** | |
| As a team member, I want a centralized communication platform so that I can easily collaborate with other departments without information getting lost in emails. | |
| **Acceptance Criteria** | |
| * Departmental chat channels * Direct messaging functionality * File sharing capabilities * @mention and notification system * Search functionality for past communications | |
|
|
|

|  |  |
| --- | --- |
| **US 12** | Priority: Must Have |
| **User Story Description** | |
| As a production coordinator, I want to store and manage all production documents so that relevant team members can access them when needed. | |
| **Acceptance Criteria** | |
| * Hierarchical folder structure for document organization * Version control for all documents * Permission-based access controls * Search functionality with metadata filtering * Document preview without download | |
|
|
|

|  |  |
| --- | --- |
| **US 13** | Priority: Must Have |
| **User Story Description** | |
| As a production manager, I want to track editing progress, visual effects delivery, and sound mix status so that I can ensure the film completes post-production on schedule. | |
| **Acceptance Criteria** | |
| * Milestone tracking for post-production phases * Delivery status for VFX shots and sound elements * Version control for edits and cuts * Feedback and approval workflow * Integration with common post-production software | |
|
|
|

|  |  |
| --- | --- |
| **US 14** | Priority: Must Have |
| **User Story Description** | |
| As a production manager, I want access to production analytics and reports so that I can assess efficiency and identify potential issues early. | |
| **Acceptance Criteria** | |
| * Customizable dashboard with key performance indicators * Budget vs. actual spending reports * Schedule adherence metrics * Resource utilization statistics * Export functionality for presentations | |
|
|
|

|  |  |
| --- | --- |
| **US 15** | Priority: Must Have |
| **User Story Description** | |
| As a team member, I want to access the system via mobile devices so that I can update and retrieve information while on set or location. | |
| **Acceptance Criteria** | |
| * Responsive design for various device sizes * Offline mode with synchronization upon connectivity * Camera integration for photo/document upload * Push notifications for critical updates * Reduced data usage mode for limited connectivity areas | |
|
|
|

# **FRONTEND COMPONENTS**

**Dashboard**

**Description**: The central hub for project overview, providing quick access to key information and actions.

**Features**:

* Project overview with progress indicators (e.g., percentage completion, visual charts).
* Upcoming deadlines and milestones (calendar view, list).
* Recent activity feed (logs of actions taken).
* Quick access to frequent actions (shortcuts to common tasks).
* Alerts and notifications panel (system messages, warnings).
* Weather forecast for shooting locations (integrated weather API).

**Project Management Module**

**Description**: Financial tools for budgeting, expense tracking, and reporting all aspects of a production project.

**Features**:

* Project creation wizard (step-by-step setup).
* Project details view/edit (comprehensive project information).
* Team assignment interface (drag-and-drop assignment).
* Production timeline visualization (Gantt chart, timeline).
* Milestone tracking board (Kanban board, progress tracking).
* Document repository (file storage and management).
* Budget Management Interface (quick access to budget tools)

**Budget Management Interface**

**Description**: Tools for creating, tracking, and managing the project's financial aspects.

**Features**:

* Budget creation templates (pre-defined budget structures).
* Line item entry and categorization (detailed expense tracking).
* Real-time expense tracking (updated financial data).
* Budget vs. actual visualization (charts, graphs).
* Approval workflow for expenses (digital approval process).
* Financial reporting tools (customizable reports).

**Production Schedule Builder**

**Description**: A visual calendar for creating and managing the shooting schedule.

**Features**:

* Script breakdown integration.
* Calendar view with day/night indication.
* Resource allocation visualization.
* Drag-and-drop schedule management.
* Conflict detection highlighting.
* Schedule version comparison.

**Cast & Crew Portal**

**Description**: Personalized access to schedules, contacts, and documents for team members.

**Features**:

* Contact database with search.
* Availability calendar.
* Contract status tracking.
* Call time notifications.
* Personalized schedule view.
* Role-based document access.

**Asset Management System:**

**Description**: Inventory tracking and management for production equipment and props.

**Features**:

* Inventory database with search (asset catalog).
* Asset categorization and tagging (asset organization).
* Check-in/check-out interface (asset tracking).
* Asset assignment to scenes (scene allocation).
* Maintenance scheduling (asset maintenance).
* Barcode/QR scanning interface (quick identification).

**Location Management**

**Description**: A database for managing shooting locations, permits, and logistics.

**Features**:

* Location database with media gallery (location info).
* Map integration with location pins (visual mapping).
* Permit tracking interface (permit management).
* Location scheduling calendar (location usage).
* Cost management for each location (location expenses).
* Weather integration for outdoor locations (real-time weather).

**Communication Center:**

**Description**: A real-time platform for team collaboration and information sharing.

**Features**:

* Department channels (group communication).
* Direct messaging (private communication).
* File sharing interface (document sharing).
* Notification management (notification settings).
* Search functionality (message search).
* @mention system (tagging users).

**Reporting Tools:**

**Description**: Features for generating and distributing production reports and summaries.

**Features**:

* Daily progress report generator (automated reports).
* Call sheet builder (call sheet creation).
* Customizable report templates (report customization).
* Distribution list management (report distribution).
* Report archive with search (report storage).
* Export options (PDF, Excel, Email).

**Mobile Application**

**Description**: A field-optimized mobile interface for on-the-go production management.

**Features:**

* Simplified interface for field use (mobile optimization).
* Camera integration for documentation (photo/video capture).
* GPS integration for location services (location tracking).
* Push notification system (real-time alerts).
* Offline functionality (offline data access).
* Sync status indicator (data synchronization).

# **ENVIRONMENT REQUIREMENTS**

**Technology Stack:**

* **Backend:** Spring Boot, Spring Data, Spring REST, Spring Cloud Consul
* **Frontend**: Angular/React, Typescript/JavaScript HTML5, CSS3, Bootstrap
* **Database**: MySQL/NOSQL
* **Security** - OAuth/Spring Security

**Non-Functional Requirements:**

* **Performance**: The application should be fast and responsive, with quick load times and minimal lag or delay.
* **Scalability**: The application should be able to handle large amounts of traffic and scale as needed to accommodate growth.
* **Security**: The application should be secure and protect user data from unauthorized access, with measures such as encryption and secure authentication. Secure password storage using hashing. Protect against common web vulnerabilities (e.g., SQL injection, XSS).
* **Reliability**: The application should be reliable and available, with minimal downtime or outages.
* **Usability**: The application should be easy to use and intuitive, with a clear and consistent user interface.
* **Maintainability**: The application should be easy to maintain and update, with clear and well-organized code that is easy to understand and modify.
* **Compatibility**: The application should be compatible with a wide range of devices and browsers, with support for different screen sizes and resolutions.
* **Implement** caching mechanisms to improve performance.
* **Session Management**: Implement secure session management techniques to prevent unauthorized access. Consider using short-lived sessions with automatic timeouts.
* **Lazy Loading**: Implement lazy loading techniques for images and other content, ensuring faster initial page loads and smoother user experience.
* **User Interface (UI) and User Experience (UX) Design**: Design an intuitive user interface that is easy to navigate and provides a seamless user experience. Prioritize responsiveness and clarity across all devices (desktop, mobile, tablets).

**Final Deliverables:**

* Application archive (.jar ) with source code
* Database DDL Script
* Complete Source code
* Sample screenshots of important screens

**Implementation Guidelines:**

* Utilize Lombok for model class creation and logging.
* Implement logic using Lambdas and Streams.
* Organize code into logical layers (controller, service, DTO, entity, etc).
* Use meaningful package names reflecting the domain.
* Employ Spring dependency injection for effective component management.

**General Guidelines:**  
**Error Handling:**

* Handle cases where the user attempts to access without authorization.
* Implement a custom exception to throw user-defined messages.
* Centralized exception handling mechanism should be used to capture exceptions and translate them into HTTP response.
* Implement proper error handling mechanisms to catch any issues and log all the service exceptions using LoggingAspect
* Handle success and error responses appropriately with proper messages.

**DTOs and Mapping:**

* Data Transfer Objects (DTOs) should be used for API request and response.
* Entities should not be exposed directly to APIs.
* Use ModelMapper to convert from entities to dto and vice versa.

**Database Interaction:**

* Spring Data Repository should be used for database operation.
* Add appropriate properties specific to the application like database properties and table generation strategies.

**API Design:**

* All the api should be mapped with the base URI
* Set server port number of your choice.
* Use appropriate HTTP methods and status code for all rest end points.
* Use Swagger for generating the API documentation.

**Microservice Communication:**

* Circuit Breaker pattern should be implemented for all the critical services with a timeout of 3 seconds and error threshold of 50%. The circuit breaker should open after 10 consecutive failures and remain open for 60 seconds. Upon circuit breaker opening, a fallback mechanism should return a generic message to the client.
* Fallback behaviors should be defined for all critical operations.
* Create two instances for any Microservice and implement load balancing.

**Validation:**

* Bean validation should be used for all inputs throughout the process.
* Custom Validators should be used for complex validation.
* All request parameters must be validated for null or empty values. For any such invalid values, ‘Please provide a valid <attribute name>’ should be the error message.
* Any date and time value should not start with zero.

**Testing:**

* Write JUnit test cases for all service methods using Mockito.
* 80% code coverage should be achieved.

**Code Quality:** To ensure adherence to coding standards, the project should be analyzed using SonarQube. The following minimum acceptable values must be met:

|  |  |
| --- | --- |
| **SonarQube Metrics** | **Minimum Acceptable Value** |
| Security | A |
| Reliability | A |
| Issues | <= 5 |
| Coverage | >= 80% |
| Duplications | <=3% |
| Security Hotspots | A |

**Postman:** Once done with implemen­­ting the requirements, use Postman or Swagger to test whether the REST endpoints are working fine