PROJECT REPORT

Project Title: Aura Platform – Intelligent Movie Streaming Web Application
Client/Organization: Academic Project / Independent Development
Prepared By: Abhinav Mishra, Kumar Mayank, Akhand Pratap Singh
Date: *20 February 2026*

1. Executive Summary

Aura Platform is a full-stack, API-driven movie streaming web application built using **Django 4.2.7 (Python)**. The platform provides users with a Netflix-inspired interface to browse, search, watch trailers, manage personal watchlists, and submit ratings & reviews.

The system integrates with **TMDB API** for real-time movie metadata and **YouTube Data API** for trailer embedding. Additionally, it includes a **Supervisor Portal** that enables analytics tracking, traffic monitoring, and content moderation.

The project demonstrates modern web architecture principles including role-based authentication, API abstraction, retry mechanisms, data caching, and analytics dashboards. The expected outcome is a scalable and secure streaming platform prototype suitable for academic and enterprise demonstration.

2. Project Overview

Business Problem Statement

Most academic streaming projects lack:

- Real-time movie data integration
- Structured content moderation
- User engagement tracking
- Analytics dashboards for decision-making

There is a need for a scalable, API-driven streaming application that integrates modern OTT architecture concepts while maintaining security and performance.

Project Objectives

- Develop a full-stack movie streaming web application
- Integrate real-time movie data using TMDB API
- Embed YouTube trailers with embeddability validation
- Implement user authentication and role-based access
- Build analytics dashboard for supervisors
- Implement content moderation (hide/unhide movies)
- Ensure secure and scalable architecture

Scope

In-Scope

- User Registration & Login
- Movie Browsing (Trending, Popular, Top Rated, Upcoming)
- Movie Search
- Movie Detail Page with Trailer
- Watchlist Management
- Rating & Review System (1–10 scale)
- Supervisor Analytics Dashboard
- Content Moderation
- View Tracking with IP logging

Out-of-Scope

- Video hosting or actual movie streaming
- Payment gateway integration
- Subscription model implementation
- Mobile application

Key Deliverables

- Fully functional Django web application
- 2 Django apps (movies, accounts)
- 4 database models
- 14+ view functions
- Supervisor analytics dashboard
- Production-ready configuration setup
- Technical documentation

Success Criteria

- Successful API integration without rate-limit failure
- Accurate tracking of movie views
- Role-based access control functioning properly
- Secure user authentication
- Smooth UI/UX across devices

3. Solution Architecture & Design

System Architecture Overview

Aura Platform follows Django's **MVT (Model-View-Template)** architecture:

- **Model Layer:** Handles database operations (Movie, Watchlist, Rating, MovieView)
- **View Layer:** Business logic & API integration
- **Template Layer:** Frontend rendering with dynamic data

The architecture includes a dedicated **Service Layer** for external API communication with retry logic using exponential backoff.

Technology Stack

Frontend

- HTML5
- CSS3 (Glassmorphism + Dark Theme)
- JavaScript (AJAX interactions)
- Google Fonts (Inter)
- Font Awesome Icons

Backend

- Django 4.2.7
- Python
- requests library
- tenacity (retry logic)
- python-dotenv

Database

- SQLite (Development)
- Easily upgradeable to PostgreSQL (Production)

Server

- Gunicorn WSGI Server

Integration Points

- **TMDB API**
    - Popular Movies
    - Trending Movies
    - Top Rated Movies
    - Search API
    - Movie Metadata
- **YouTube Data API**
    - Trailer Search
    - Embeddability Validation
    - Cached Trailer Key Storage

Security & Compliance Considerations

- CSRF Protection
- Password validation (Django validators)
- Staff-only access decorators

- Login-required decorators
- Environment variable-based secret management
- XSS & Clickjacking protection
- IP tracking with proxy support

Scalability & Performance Strategy

- Service-layer API abstraction
- Retry logic with exponential backoff
- Pagination (max 500 pages TMDB limit)
- Trailer key caching in database
- Modular Django apps for separation of concerns
- Production-ready Gunicorn configuration

4. Implementation Plan

Project Phases

1. Requirement Analysis & Planning
2. Database Design & Model Creation
3. API Integration Development
4. Authentication & Authorization Setup
5. Frontend UI Implementation
6. Supervisor Dashboard Development
7. Testing & Debugging

8. Deployment Preparation

Timeline & Milestones

- Week 1: Project setup & architecture design
- Week 2: API integration & core backend
- Week 3: Frontend UI & authentication
- Week 4: Analytics dashboard & moderation
- Week 5: Testing & optimization

Resource Allocation

- 1 Full-Stack Developer (Project Owner)
- APIs (Free-tier usage)
- Local Development Environment

Risk Assessment & Mitigation

Risk	Mitigation Strategy
API Rate Limit	Retry logic + caching
API Downtime	Graceful error handling
Unauthorized Access	Role-based decorators
Data Inconsistency	Unique constraints
Security Vulnerabilities	Django security middleware

Testing & Quality Assurance Strategy

- Unit testing of forms and models
- Manual UI testing
- Role-based access testing
- API error handling validation
- Security testing (CSRF & authentication)

5. Development Team Introduction

Role	Name	Experience	Responsibilities
Project Manager	Abhinav Mishra, Kumar Mayank, Akhand Pratap Singh	1+ Years Academic	Planning & Architecture
Backend Developer	Kumar Mayank, Akhand Pratap Singh	1+ Years	Django Backend & Services
Frontend Developer	Abhinav Mishra, Kumar Mayank, Akhand Pratap Singh	1+ Years	UI/UX & Responsive Design

QA Engineer	Abhinav Mishra,Kumar Mayank, Akhand Pratap Singh	1+ Years	Testing & Debugging

(Since this is an individual academic project, all roles were handled by the same developer.)

7. Conclusion & Next Steps

Aura Platform successfully demonstrates a modern full-stack streaming application integrating real-time APIs, analytics tracking, secure authentication, and role-based moderation.

The project is scalable to production with PostgreSQL, Redis caching, and cloud deployment.

Next Steps

- Deploy on cloud infrastructure
- Replace SQLite with PostgreSQL
- Add Redis caching
- Implement recommendation engine
- Add subscription/payment model