# Names: Omri Portal , Yahav Dayan Project Name : SEMS Software Test Plan (STP)

## 1. Introduction

This Software Test Plan describes the strategy for testing the Event Management System backend. The goal is to ensure all endpoints related to client interaction, venue and artist creation, show handling, and ticket purchasing operate correctly and reliably.

## 2. Test Items

- Client Registration Module  
- Purchase Recording System  
- Venue and Artist Management API  
- Show Creation and Management API  
- Dynamic Pricing Recommendation Engine  
- Revenue and Sales Reporting Module

## 3. Features to be Tested

- Registration with real and system-generated IDs  
- Ticket purchase recording and total spent tracking  
- Correct computation of seat arrangements per venue  
- Artist tagging by genre and linking to shows  
- Dynamic pricing based on demand and historical data  
- Viewing show revenue and sold ticket statistics

## 4. Features Not to be Tested

- User Interface or frontend logic  
- Integration with external payment providers or APIs

## 5. Testing Strategy

We will follow a layered approach to testing:  
- Unit Testing for individual endpoints and logic (e.g., ID validation, pricing logic)  
- Integration Testing to verify data flows between routes and shared state  
- System Testing for full workflows such as registration to purchase  
- Acceptance Testing based on expected use cases

## 6. Test Environment

- Backend Framework: FastAPI (Python 3.x)  
- Test Tools: pytest, httpx, unittest  
- Operating System: Localhost (Windows/Linux)  
- Optional database simulations via Python dictionaries

## 7. Responsibilities

- Omri Portal: Backend API Testing & Unit Tests  
- Yahav Dayan: Integration & System Testing

## 8. Schedule

- Week 1–2: Unit Testing  
- Week 3: Integration Testing  
- Week 4: System Testing  
- Week 5: Final Acceptance Testing & Review

## 9. Risks and Contingencies

- Risk: Misalignment between backend behavior and frontend expectations  
- Contingency: Create API documentation and test mocks to clarify data flow