2. Introduction

The chosen system for this assignment is an **Event Ticket Booking Application**, similar in scope and complexity to platforms such as BookMyShow. The system enables users to discover events, select venues and showtimes, choose seats, complete payments, receive tickets, and manage bookings after purchase. It also supports backend processes such as inventory control, pricing management, payment verification, fraud detection, and automated notifications.

Functional decomposition is an important technique for understanding complex systems because it breaks a large system into smaller, logically related components. Instead of focusing on screens or interface elements, functional decomposition focuses on **what the system does** and **how responsibilities are distributed** across layers. This approach helps designers and engineers identify hidden logic, system dependencies, failure points, and automation requirements early in the design process.

The scope of this decomposition is limited to the **digital ticket booking platform**, including user-facing features and backend system logic required to support them. Physical venue operations, hardware scanners, and third-party infrastructure beyond system integration points are excluded. The analysis focuses on functional behavior rather than visual design or implementation details.

---

3. Methodology

The functional decomposition was approached by first identifying the primary goals a user attempts to accomplish within the system, such as discovering events, booking tickets, making payments, and managing bookings. These high-level goals were translated into major functional domains, which form the first level of the decomposition.

Each primary function was then broken down progressively into more detailed sub-functions. The decomposition was carried out across five distinct layers. **Level 1** represents primary functional buckets that define broad system responsibilities. **Level 2** captures user-visible sub-functions within each bucket. **Level 3** represents detailed functional capabilities that support those sub-functions. **Level 4** contains micro-components and operational steps required to execute features. **Level 5** consists of conditional logic and system-only operations that handle automation, validation, error recovery, and reliability.

Features were categorized based on responsibility rather than screen placement. User-facing capabilities were placed in higher layers, while automation, validation, and background processes were pushed to deeper layers. Shared logic such as inventory synchronization, fraud detection, and payment verification was centralized to avoid duplication and ensure consistency across the system.

Throughout the decomposition, special attention was given to concurrency, failure scenarios, edge cases, and backend automation. This ensured that the system model reflects real-world operating conditions such as high traffic, payment ambiguity, and partial user actions.