

QUICKTICK

(EVENT TICKET BOOKING APP)

Main Project Report

Submitted in partial fulfillment of the requirements for the award

of the degree of

BACHELOR OF COMPUTER APPLICATIONS

BASELIOS POULOSE II CATHOLIC COLLEGE

PIRAVOM

BASELIOS MOUNT, PIRAVOM

Re-accredited with 'A' Grade by NAAC

(Affiliated to Mahatma Gandhi University)

2024-25



Submitted by:

ANAGHA S NAIR (Reg No: 220021090191)

**QUICKTICK
(EVENT TICKET BOOKING APP)**

Main Project Report

**Submitted in partial fulfillment of the requirements for the award
of the degree of**

BACHELOR OF COMPUTER APPLICATIONS

**BASELIOS POULOSE II CATHOLIC COLLEGE
PIRAVOM
BASELIOS MOUNT, PIRAVOM**

**Re-accredited with 'A' Grade by NAAC
(Affiliated to Mahatma Gandhi University)**

2024-25



Guided by:

Dr. Nimmol P John

Submitted by:

**ANAGHA S NAIR
(Reg No: 220021090191)**

BASELIOS POULOSE II CATHOLICOS COLLEGE

Affiliated to Mahatma Gandhi University

(Re-accredited with 'A' Grade by NAAC)

PIRAVOM

2024-25

DEPARTMENT OF COMPUTER APPLICATIONS



Certificate

This is to certify that the project entitled "QUICKTICK" submitted in partial fulfillment for the award of the degree of BACHELOR OF COMPUTER APPLICATION is a bonafide report of the project done by Anagha S Nair (Reg no: 220021090191) during the year 2024-25.

Internal Guide:

Dr. Nimmol P John

Head of the Department

Dr. Anu Paul

Examiner: 1

College Seal

Department Seal

DECLARATION

*I hereby declare that this project work entitled “QUICKTICK” is a record of original work done by me under the guidance of **Dr. Nimmol P John**, Assistant Professor, Department of Computer Applications and the work has not formed the basis for the award of any degree or diploma or similar title to any candidate of any university subject.*

Internal Guide

Signature of Student

Dr. Nimmol P John

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

At the outset, I thank God Almighty for making endeavor a success.

I express my gratitude to **Dr. Baby Paul**, Principal, Baslios Poulose II Catholicos College, for providing me with adequate facilities, ways and means by which I was able to complete the project work. I express my sincere thanks to my internal guide **Dr. Nimmol P John** who guide me properly from the beginning to the end of my project and examining the draft of this project, suggestions and modifications. With immense pleasure I take this opportunity to record out sincere thanks to Head of the Department **Dr. Anu Paul**, Associate Professor, Department of Computer Applications for her motivation throughout this project.

Last but not the least, I also express my gratitude to all other members of the faculty and well-wishers who assisted me in various occasions during the project work.

- **Anagha S Nair**

ABSTRACT

ABSTRACT

The Event Ticket Booking System is a digital platform designed to simplify the process of event management and ticket booking. Built using Flutter, it provides a seamless cross-platform experience, ensuring accessibility on both mobile and web applications. The system aims to bridge the gap between event organisers, stall manager, and users by offering an efficient and organized event management solution.

The system comprises four main modules: Admin, Event Organisers, Stall Manager, and Users. The Admin Module acts as the central authority, overseeing the entire system by managing user registrations, approving events, and ensuring smooth operations. Event Organisers can create, update, and manage event details such as venue, date, ticket pricing, and availability, making it easy for them to reach a wider audience.

The Stall Manager Module is specifically designed to handle stall bookings, vendor allocations, and event logistics. This feature is beneficial for large-scale events where multiple stalls are set up for exhibitions, food courts, or merchandise sales. Meanwhile, the Users Module provides attendees with a user-friendly interface to explore events, book tickets, and make secure payments through an integrated payment gateway.

Key features of the system include QR-based ticket verification, which enhances security and prevents fraudulent entries, and real-time event updates, ensuring that users receive the latest event information. By offering a centralized, automated, and efficient ticket booking solution, the Event Ticket Booking System enhances event planning, improves ticketing efficiency, and provides a smooth experience for all stakeholders involved.

TABLE OF CONTENTS

1.Introduction.....	11
1.1. Background and motivation	12
1.2. The proposed system	12
1.3. Project Scope.....	12
2. System Analysis.....	14
2.1. Introduction	15
2.2. Stake holders of this project.....	16
2.2.1. Admin.....	16
2.2.2. Event Organiser.....	16
2.2.3. Stall Manager.....	16
2.2.4. User.....	17
2.3Software requirement specifications	17
2.3.1. System Features.....	17
2.3.1.1. Admin.....	17
2.3.1.2. Event Organiser.....	17
2.3.1.3. Stall Manager.....	18
2.3.1.4. User.....	18
2.3.2. Non-functional Requirements.....	19
2.4. Feasibility study	19
2.5. Software Development lifecycle model	21
2.6. Hardware and software requirements	22
2.6.1. Software specifications	22
2.6.1.1.Flutter.....	22
2.6.1.2Dart	22
2.6.1.3Supabase.....	22
2.6.1.4.Windows 11	23
2.6.2. Hardware requirements.....	23

3. System Design	24
3.1. System Architecture.....	25
3.2. Module Design.....	25
3.3. Database Design.....	27
3.3.1. Normalization.....	27
3.3.2. Table structure.....	29
3.3.3. Data Flow Diagram.....	34
3.3.3.1. Introduction to data flow diagrams.....	34
3.3.3.2. Data flow diagram.....	38
3.4. Interface Design.....	45
3.4.1. User interface screen design.....	45
3.4.2. Output design.....	46
4.Implementation.....	47
4.1. Coding Standards.....	47
5. System Testing.....	49
5.1.1 Unit testing.....	51
5.1.2 Integration testing.....	51
5.1.3 Black box testing.....	52
5.1.4 White box testing.....	52
5.1.5 Validation Testing	52
5.1.6 User acceptance testing.....	53
6. Conclusion.....	54
6.1.Future Enhancement.....	57
7.Bibliography.....	58
8.Appendix.....	59
8.1. Screenshots.....	59

INTRODUCTION

1. INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

The event industry has seen rapid growth, with concerts, conferences, and exhibitions becoming increasingly popular. Traditional event management and ticket booking methods often rely on manual processes, leading to inefficiencies such as long queues, delayed updates, and difficulties in managing stalls and vendors. These challenges create a need for a centralized and automated system that simplifies ticket booking and event organization while enhancing user experience.

With the rise of digital platforms, online ticket booking has become the preferred choice for users due to its convenience and speed. However, many existing systems fail to integrate event organisers, stall manager, and users within a single platform. This gap in the market has motivated the development of the Event Ticket Booking System, which aims to provide a seamless and secure solution for event management, stall allocation, and ticket purchases.

By leveraging Flutter for cross-platform accessibility, the system ensures ease of use across different devices. Features like QR-based ticket verification, real-time event updates, and secure online payments enhance security and user convenience. This project is a step toward digitizing event management, offering a more organized and hassle-free solution for both event organisers and attendees.

1.2 PROPOSED SYSTEM

The Event Ticket Booking System is a digital platform designed to streamline event management by integrating four key modules: Admin, Event Organisers, Stall Manager, and Users. The system enables event organisers to create and manage events, stall manager to oversee vendor allocations, and users to browse and book tickets conveniently. The admin module ensures smooth operations by managing approvals and monitoring the system. Built using Flutter for cross-platform accessibility, the system includes features like QR-based ticket verification, real-time event updates, and secure online payments to enhance security and user experience. By automating ticket booking and stall management, the system reduces manual errors, improves efficiency, and provides a seamless event experience for all stakeholders.

1.3 PROJECT SCOPE

Limitations of existing system

- Manual ticket booking process leading to long queues, delays, and errors in managing event entries.

- Lack of a centralized platform for event organisers, stall manager, and users, causing coordination issues.
- No real-time event updates, causing delays in informing users about ticket availability, event changes, or cancellations.
- Higher operational costs due to manual processes, increasing administrative workload for event organisers.
- Poor user experience as traditional systems lack an intuitive and automated interface for easy navigation and booking.

Advantages of proposed system

- Automated ticket booking process reducing manual efforts, queues, and errors, ensuring a smooth user experience.
- Centralized platform integrating event organisers, stall manager, and users for seamless coordination and management.
- Cross-platform accessibility with Flutter, allowing users to book tickets and manage events on both mobile and web applications.
- Real-time event updates keeping users informed about ticket availability, event changes, and cancellations.
- User-friendly interface ensuring easy navigation, event browsing, and hassle-free ticket booking.

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1. INTRODUCTION

Software Engineering is the analysis, design, construction, verification and management of technical or social entities. To engineer software accurately, a software engineering process must be defined. System analysis is a detailed study of the various operations performed by the system and their relationship within and module of the system. It is a structured method for solving the problems related to the development of a new system. The detailed investigation of the present system is the focal point of system analysis. This phase involves the study of parent system and identification of system objectives. Information has to be collected from all people who are affected by or who use the system. During analysis, data are collected on the variable files, decision point and transactions handled by the present system. The main aim of system is to provide the efficient and user friendly automation. So the system analysis process should be performed with extreme precision, so that an accurate picture of existing system, its disadvantages and the requirements of the new system can be obtained.

System analysis involves gathering the necessary information and using the structured tool for analysis. This includes the studying existing system and its drawback, designing a new system and conducting cost benefit analysis. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. The system is studied to the minute detail and analyzed. The system is viewed as a whole and the inputs to the system are identified. The outputs from the organization are traced through various phases of processing of inputs.

There are a number of different approaches to system analysis. When a computer-based information system is developed, systems analysis (according to the Waterfall model) would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, technologically and operationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

Techniques such as interviews, questionnaires etc. can be used for the detailed study of these processes. The data collected by these sources must be scrutinized to arrive at a conclusion.

The conclusion is an understanding of how the system functions. This system is called the Existing System. The Existing system is then subjected to close observation and the problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as a proposal which is the Proposed System. The proposal is

then weighed with the existing system analytically and the best one is selected. The proposal is then presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is a loop that ends as soon as the user is satisfied with the proposal.

2.2. STAKE HOLDERS OF THIS PROJECT

2.2.1. ADMIN

The Admin plays a crucial role in managing and overseeing the entire system to ensure smooth operations. They are responsible for approving event organisers, verifying stall manager, and monitoring user activities to maintain a secure and well-organized platform. The admin also manages system settings, including event categories, and user access controls, ensuring that all stakeholders adhere to the platform's guidelines.

Additionally, the Admin monitors event performance, handles dispute resolution, and generates reports on ticket sales and user engagement. They also ensure secure payment processing, prevent fraudulent activities, and provide technical support when needed. By acting as the central authority, the Admin ensures efficiency, security, and a seamless experience for event organisers, stall manager, and users within the system.

2.2.2. EVENT ORGANISER

Event organisers are responsible for creating, managing, and promoting events on the platform. They can add event details such as venue, date, ticket pricing, and seating availability. Organisers also have control over ticket categories to attract more attendees. Their role ensures that users have access to accurate and up-to-date event information.

Additionally, event organisers can track ticket sales, manage bookings, and handle stall requests in real time. They also coordinate with stall manager to allocate vendor spaces efficiently. By using the system's analytics and reporting features, organisers can monitor event performance and make data-driven decisions to improve future events. Their role is essential in ensuring successful event execution and a smooth experience for both users and stalls.

2.2.3. STALL MANAGER

The Stall Manager module is essential for efficiently handling stall bookings at events, ensuring seamless organization and vendor coordination. The Registration page enables stall manager to sign up and access the platform. Through the My Account section, managers can update their profiles and credentials. The View Events page allows them to browse available events and assess stall allocations, while the Send Request page facilitates stall booking submissions. The View My Request page helps track the status of

these requests, ensuring transparency in approvals and rejections. By streamlining stall reservations, managing vendor details, and coordinating with event organisers, the Stall Manager module enhances event efficiency, optimizes space distribution, and ensures a hassle-free experience for vendors and attendees.

2.2.4. USER

Users are the primary audience of the system, responsible for browsing events, booking tickets, and making payments through the platform. They can search for upcoming events based on categories, name. The system provides a user-friendly interface that allows them to view event details, select seats, and purchase tickets conveniently.

Additionally, users receive real-time updates on event schedules, ticket availability, and cancellations. They can access their tickets digitally, use QR-based verification for entry, and manage bookings easily. By offering a seamless ticket booking experience, the system enhances user convenience and engagement with various events.

2.3. SOFTWARE REQUIREMENTS SPECIFICATIONS

2.3.1. SYSTEM FEATURES

2.3.1.1. ADMIN

- The Admin has the highest authority in the system and manages all operations.
- The Admin can log in to the system using a valid username and password; otherwise, an error message is displayed.
- The Admin verifies and approves or rejects event organisers and stall manager during registration.
- The Admin can view and manage details of event organisers, stall manager, and users.
- The Admin oversees the entire system, ensuring smooth functionality and resolving any technical or operational issues.
- The Admin has the authority to approve or remove events based on platform policies and guidelines.
- The Admin monitors ticket sales and stall bookings, ensuring transparency and efficiency.
- The Admin can generate reports on ticket sales, most booked events, and stall performance.
- The Admin handles user complaints and feedback, taking necessary actions to improve the system.
- The Admin can manage content pages, such as About Us, Terms & Conditions, and FAQs, to provide users with relevant information.

2.3.1.3. EVENT ORGANISER

- Event organisers can log in to the system using a valid username and password; otherwise, an error message is displayed.

- Event organisers can create and manage events by adding event details such as name, date, time, venue, and ticket prices.
- Event organisers can set ticket categories and manage availability for different seating sections.
- Event organisers can update or cancel events, notifying users about any changes in real time.
- Event organisers can track ticket sales and monitor booking status for their events.
- Event organisers can collaborate with stall manager to allocate and manage vendor stalls for their events.
- Event organisers can view user details related to ticket bookings and provide customer support if needed.
- Event organisers receive notifications when all tickets for an event are sold out.
- Event organisers can analyze event performance using sales reports and user engagement data.
- Event organisers can respond to user queries and feedback to improve event experiences.

2.3.1.4. STALL MANAGER

- Stall manager can log in to the system using a valid username and password; otherwise, an error message is displayed.
- Stall manager can view an overview of available stalls, upcoming events, and pending booking requests.
- Stall manager can update their profile details, manage credentials, and configure notification preferences.
- Stall manager can browse events, check stall availability, and assess vendor participation.
- Stall manager can allocate stalls to vendors based on availability and event requirements while also submitting booking requests for stalls.
- Stall manager can track stall reservations, monitor booking statuses, and receive notifications when stalls are fully booked.

2.3.1.5. USER

- Users can register and log in to the system using a valid username and password; otherwise, an error message is displayed.
- Users can browse and search for events based on categories, location, and date.
- Users can view event details such as name, venue, date, time, ticket price, and available seats.
- Users can book tickets by selecting the desired event.
- Users can make secure online payments using multiple payment options to complete ticket purchases.
- Users receive digital tickets with QR codes for easy verification at the event entry.
- Users get real-time notifications about ticket confirmation, event updates, or cancellations.
- Users can view and manage their bookings, including canceling tickets if allowed by the event policy.
- Users can provide complaints and ratings for events to help organisers improve future experiences.

2.3.2. NON-FUNCTIONAL REQUIREMENTS

Usability: The user interface of QickTick should be intuitive, responsive, and provide a seamless shopping experience.

Performance: The system should handle a large number of concurrent users and provide fast response times for browsing, searching, and booking seats.

Security: The system should implement robust security measures to protect user data, including secure payment processing and data encryption.

Reliability: QickTick should be highly available and reliable, with minimal downtime and effective error handling mechanisms.

Scalability: The system should be designed to handle high traffic volumes, supporting a large number of schedules, BSPs, and Users.

Compatibility: QuickTick should be compatible with major web browsers and mobile devices to ensure broad accessibility.

2.4. FEASIBILITY STUDY

Feasibility is defined as the practical extent to which a project can be performed successfully. To evaluate feasibility, a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards.

Various other objectives of feasibility study are listed below.

- To analyse whether the software will meet organizational requirements.
- To determine whether the software can be implemented using the current technology and within the specified budget and schedule.
- To determine whether the software can be integrated with other existing software.

Various types of feasibility that we checked include technical feasibility, operational feasibility, and economic feasibility.

Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and

technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyses the technical skills and capabilities of the software development team members.
- Determines whether the relevant technology is stable and established.
- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

From our perspective there are two languages PHP, HTML and database MySQL which are used to develop this web based applications. PHP is used in the front end and MySQL is used in the back end. The Word to the Wise is web based and thus can be accessed through any browsers. As we are using these latest technologies which are currently trending and used by a number of developers across the globe, we can say that our project is technically feasible.

Operational Feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority.
- Determines whether the solution suggested by the software development team is acceptable.
- Analyses whether users will adapt to a new software.
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

We found that our project will be satisfied for the client since we were discussing every detail about the software with the client at every step. The most important part of operational feasibility study is the input from client. So the software is built completely according to the requirements of the client. We have used the current industry standards for the software. Hence we can say that this software is operationally feasible.

Economic Feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by

developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis).
- Cost of hardware, software, development team, and training.

It is estimated that our project is economically feasible as development cost is very minimal since the tools and technologies used are available online. It's a group student project so there are no personnel costs. Development time is well planned and will not affect other operations and activities of the individuals. Once the system has been developed, the companies purchasing the system will be providing with a manual for training purposes. There is no need to purchase new hardware since the existing computers can still be used to implement the new system.

2.5. SOFTWARE DEVELOPMENT LIFECYCLE MODEL

One of the basic notions of the software development process is SDLC models which stand for Software Development Life Cycle models. SDLC – is a continuous process, which starts from the moment, when it's made a decision to launch the project, and it ends at the moment of its full remove from the exploitation. Software development lifecycle (SDLC) is a framework that defines the steps involved in the development of software. It covers the detailed plan for building, deploying and maintaining the software. SDLC defines the complete cycle of development i.e. all the tasks involved in gathering a requirement for the maintenance of a Product.

Some of the common SDLC models are Waterfall Model, V-Shaped Model, Prototype Model, Spiral Model, Iterative Incremental Model, Big Bang Model, Agile Model. We used Agile Model for our Project.

Agile Model

Agile Model is a combination of the Iterative and incremental model. This model focuses more on flexibility while developing a product rather than on the requirement. In the agile methodology after every development iteration, the client is able to see the result and understand if he is satisfied with it or he is not. Extreme programming is one of the practical use of the agile model.

The basis of this model consists of short meetings where we can review our project. In Agile, a product is broken into small incremental builds. It is not developed as a complete product in one go. At the end of each sprint, the project guide verifies the product and after his approval, it is finalized. Client feedback is taken for improvement and his suggestions and enhancement are worked on in the next sprint. Testing is done in each sprint to minimize the risk of any failures.

Advantages of Agile Model:

- It allows more flexibility to adapt to the changes.

- The new feature can be added easily.
- Customer satisfaction as the feedback and suggestions are taken at every stage.
- Risks are minimized thanks to the flexible change process.

Disadvantages:

- Lack of documentation.
- If a customer is not clear about how exactly they want the product to be, then the project would fail.
- With all the corrections and changes there is possibility that the project will exceed expected time .

2.6. HARDWARE AND SOFTWARE REQUIREMENTS.**2.6.1. SOFTWARE SPECIFICATION**

This project is developed using modern cross-platform technologies to ensure efficiency, scalability, and a seamless user experience across both web and mobile applications.

Front end : Flutter

Development tool : Flutter

Database : Supabase

Programming Language : Dart

Operating System : Windows 11

2.6.1.1.FLUTTER

Flutter, developed by Google, is an open-source UI toolkit designed for building high performance, natively compiled applications for mobile, web, and desktop from a single codebase. It provides a reactive framework with customizable widgets, enabling developers to create visually appealing and responsive user interfaces. Flutter's hot reload feature allows instant updates during development, improving productivity. It eliminates the need to maintain separate codebases for different platforms, ensuring faster development cycles and consistent UI/UX across devices.

2.6.1.2. DART

Dart is the programming language used in Flutter development. Created by Google, Dart is an object-oriented, class-based language designed for building fast and scalable applications. Key features of Dart:

- JIT (Just-in-Time) and AOT (Ahead-of-Time) Compilation: Ensures fast development cycles and high-performance execution.
- Null Safety: Reduces runtime errors by preventing null reference issues.
- Expressive and Concise Syntax: Simplifies development while maintaining readability and maintainability.

2.6.1.3.SUPABASE

Supabase is an open-source backend-as-a-service (BaaS) that provides scalable database management and authentication solutions. Built on PostgreSQL, it offers:

- Real-time Database: Enables live synchronization of data across clients.
- Authentication: Supports multiple login methods, including email/password and OAuth.
- Storage Services: Secure cloud storage for files, images, and media.

2.6.1.4.Windows 11

Windows 11 is the development environment for this project, offering improved performance, security, and support for modern development tools. Its compatibility with Flutter and Supabase ensures a smooth development process.

2.6.2. Hardware requirements

The selection of hardware configuring is a very task related to the software development, particularly inefficient RAM may affect adversely on the speed and corresponding on the efficiency of the entire system. The processor should be powerful to handle all the operations.

The hard disk should have the sufficient to solve the database and the application.

Hardware used for development:

CPU : Intel i5

Processor Memory : 16 GB

Cache : 12 MB

SSD : 512 TB

Monitor : 14.6"

Monitor Keyboard : Standard 108 keys Enhanced Keyboard

Mouse : Optical Mouse

Minimum Hardware Required For Implementation:

CPU : Pentium IV Processor

Memory : 256MB Above

Cache : 512 KB Above

Hard Disk : 20 GB Above

Monitor : Any

Keyboard : Any

Mouse : Any

SYSTEM DESIGN

3. SYSTEM DESIGN

3.1. SYSTEM ARCHITECTURE

A system architecture or system's architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).

The system architecture can best be thought of as a set of representations of an existing (or to be created) system. It is used to convey the informational content of the elements comprising a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that architecture describes may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations or people.

System architecture is primarily concerned with the internal interfaces among the system's components or subsystems, and the interface between the system and its external environment, especially the user.

The structural design reduces complexity, facilitates change and result in easier implementation by encouraging parallel development of different parts of the system. The procedural design transforms structural elements of program architecture into a procedural description of software components. The architectural design considers architecture as the most important functional requirement. The system is based on the three-tier architecture.

The first level is the user interface (presentation logic), which displays controls, receives and validates user input. The second level is the business layer (business logic) where the application specific logic takes place. The third level is the data layer where the application information is stored in files or database. It contains logic about to retrieve and update data. The important feature about the three-tier design is that information only travels from one level to an adjacent level.

3.2. MODULE DESIGN

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality. Conceptually, modules represent a separation of concerns, and improve maintainability by enforcing logical boundaries between components. Different modules of this project include.

1. User Authentication

This module enables stakeholders to authenticate themselves within the event ticket booking system, enhancing security and access control. The Admin logs in using their credentials to manage the platform and perform administrative tasks. Event Organisers log in using the credentials provided during registration to create and manage events, track bookings, and respond to user feedback. Stall Manager access their accounts to manage stall-related activities. Users log in with their registered credentials to browse events, book tickets, make payments, and interact with event organisers. Authentication ensures that each stakeholder can securely access their respective functionalities within the system.

2. Registration

This module contains the all registration process in the system. There is much registration in the system. All registration specified in the system is included for the smooth running of the system. This module includes the registrations that can be performed by all stakeholders. Admin can register the details like name, email. The bus service providers can register in the system. They should provide details like name, email, contact, address, profile photo, proof. This registered information is helpful to provide meaning full knowledge to other stakeholders. The customer / users also should register to use the site the customer should provide details like name, email, address, contact etc.

3. Activities

This module defines the activities performed by different stakeholders in the event ticket booking system. The Admin verifies event organisers, manages event categories, and oversees platform operations. The Event Organiser creates and manages events, sets ticket prices, handles bookings, and addresses user complaints and feedback. The Stall Manager oversees assigned stalls, view all requests. The User can edit their profile, browse events, book multiple tickets, make payments, view bookings, review attended events, and submit complaints or feedback to the admin.

4. Reports

This module enables stakeholders to generate and view various reports, including pie charts, bar charts, PDFs, and tabular data, providing valuable insights from the system. The Admin can generate reports such as the most booked events, most popular event categories, events with the highest ticket sales, and a list of all bookings, helping them efficiently manage the platform. Event Organisers can generate reports on ticket sales, booking trends, and attendee insights for their events. Stall Manager can access reports on stall bookings and visitor engagement. These reports eliminate manual paperwork, ensuring accurate data analysis and enhancing decision-making within the system.

3.3. DATABASE DESIGN

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for the users. The general theme behind a database is to integrate all information. Database design is recognized as a standard of management information system and is available virtually for every computer system. In database design several specific objectives are considered:

- Ease of learning and use
- Controlled redundancy
- Data independence
- More information at low cost
- Accuracy and integrity
- Recovery from failure
- Privacy and security
- Performance

A database is an integrated collection of data and provides centralized access to the data. Usually the centralized data managing the software is called RDBMS. The main significant difference between RDBMS and other DBMS is the separation of data as seen by the program and data has in direct access to stores device. This is the difference between logical and physical data.

3.3.1. Normalization

Designing a database is complete task and the normalization theory is a useful aid in the design process. The process of normalization is concerned with transformation of conceptual schema into computer representation form. There will be need for most databases to grow by adding new attributes and new relations. The data will be used in new ways. Tuples will be added and deleted. Information stored may undergo updating also. New association may also be added. In such situations the performance of a database is entirely depend upon its design. A bad database design may lead to certain undesirable things like:

- Repetition of information
- Inability to represent certain information
- Loss of information

To minimize these anomalies, Normalization may be used. If the database is in a normalized form, the data can be growing without, in most cases, forcing the rewriting application programs.

This is important because of the excessive and growing cost of maintaining an organization's application programs and its data from the disrupting effects of database growth. As the quality of application programs increases, the cost of maintaining the without normalization will rise to prohibitive levels. A normalized database can also encompass many related activities of an organization thereby minimizing the need for rewriting the applications of programs. Thus, normalization helps one attain a good database design and there by ensures continued efficiency of database.

Normalization theory is built around the concept of normal forms. A relation is said to be in normal form if it satisfies a certain specified set of constraints. For example, a relation is said to be in first normal form (1NF) if it satisfies the constraint that it contains atomic values only. Thus every normalized relation is in 1NF. Numerous normal forms have been defined. Codd defined the first three normal forms.

All normalized relations are in 1NF, some 1NF relations are also in 2NF and some 2NF relations are also in 3NF. 2NF relations are more desirable than 1NF and 3NF are more desirable than 2NF. That is, the database designer should prefer 3NF than 1NF or 2NF.

Normalization procedure states that a relation that is in some given normal form can be converted into a set of relations in a more desirable form. We can define this procedure as the successive reduction of a given collection of relations to some more desirable form. This procedure is reversible. That is, it is always possible to take the output from the procedure and convert them back into input. In this process, no information is lost. So it is also called "no loss decomposition".

First Normal Form

A relation is in first normal form (1NF) if and all its attributes are based on single domain. The objective of normalizing a table is to remove its repeating groups and ensure that all entries of the resulting table have at most single value.

Second Normal Form

A table is said to be second Normal Form (2NF), when it is in 1NF and every attribute in record is functionally dependent upon the whole key, and not just a part of the key.

Third Normal Form

A table is in third Normal Form (3NF), when it is in 2NF and every non-key attribute is functionally dependent on just the primary key.

3.3.2. Table Structure

Table is a collection of complete details about a particular subject. These data are saved in rows and Columns. The data of each Row are different units. Hence, rows are called RECORDS and Columns of each row are called FIELDS.

Data is stored in tables, which is available in the backend the items and data, which are entered in the input, form id directly stored in this table using linking of database. We can link more than one table to input forms. We can collect the details from the different tables to display on the output.

There are mainly 16 tables in the project. They are

1. tbl_admin
2. tbl_district
3. tbl_place
4. tbl_eventtype
5. tbl_stalltype
6. tbl_eventorganisers
7. tbl_event
8. tbl_stallmanagers
9. tbl_stallrequest
10. tbl_user
11. tbl_eventbooking
12. tbl_rating
13. tbl_complaint

1. Table : tbl_admin

Primary Key: admin_id

Foreign Key: Nill

Description: This table is used to store details of admin.

Field name	Data Type	Constraints	Description
admin_id	UUID	Primary key	Unique id of admin
ad_name	TEXT	Not NULL	Name of the admin
ad_email	TEXT	Not NULL	Email of the admin
ad_password	TEXT	Not NULL	Password set by the admin

2. Table : tbl_district

Primary Key: district_id Foreign key:

Nill

Description: This table is used to store districts.

Field name	Data Type	Constraints	Description
district_id	INT(8)	Primary key	Unique id of district
district_name	TEXT	Not NULL	Name of the district

3. Table : tbl_place

Primary Key: place_id

Foreign key: district_id

Description: This table is used to store places and corresponding pincode.

Field name	Data Type	Constraints	Description
place_id	INT(8)	Primary key	Unique id of place
place_name	TEXT	Not NULL	Name of the place
district_id	INT(8)	Foreign key	Unique id of district

4. Table: tbl_eventtype

Primary Key: eventtype_id

Foreign key: Null

Description: This table stores information pertaining to the eventtype of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	eventtype_id	INT (8)	Primary key	Unique ID of Eventtype
2	eventtype_name	TEXT	Not Null	Name of Eventtype

5. Table: tbl_stalltype

Primary Key: stalltype_id

Description: This table stores information pertaining to the stalltype of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	stalltype_id	INT (8)	Primary key	Unique ID of stalltype
2	stalltype_name	TEXT	Not Null	Name of stalltype

6. Table: tbl_eventorganisers

Primary Key:organisers_id

Foreign key: place_id

Description: This table stores information pertaining to the eventorganisers of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	organisers_id	UUID	Primary key	Unique ID of organisers
2	organisers_name	TEXT	Not Null	Name of organisers
3	organisers_gender	TEXT	Not Null	Gender of organisers
4	organisers_email	TEXT	Not Null	Email of organisers
5	organisers_contact	TEXT	Not Null	Contact of organisers
6	organisers_password	TEXT	Not Null	Password of organisers
7	organiser_photo	TEXT	Not Null	Photo of organisers
8	organiser_proof	TEXT	Not Null	Proof of organisers
9	organisers_status	INT(8)	Defalut value= 0	Status of organisers
10	place_id	INT (8)	Foreign key	Unique ID of Place

7. Table : tbl_event

Primary Key: event_id

Foreign key: organiser_id, place_id

Description: This table stores information pertaining to the event of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	event_id	INT (8)	Primary key	Unique ID of event
2	event_name	TEXT	Not Null	Name of event
3	event_details	TEXT	Not Null	Details of event
4	event_photo	TEXT	Not Null	Photo of event
5	event_date	TIMESTAMPZ	Not Null	Photo of event
6	event_ticketprice	TEXT	Not Null	Photo of event
7	event_count	TEXT	Not Null	Photo of event
8	event_status	INT(8)	Not Null	Photo of event
9	organiser_id	UUID	Foreign key	Unique ID of Organisers
10	place_id	INT (8)	Foreign key	Unique ID of Place

8. Table: tbl_stallmanagers

Primary Key: stallmanger_id

Description: This table stores information pertaining to the stallmanager of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	stallmanger_id	UUID	Primary key	Unique ID of stallmanger

2	stallmanger_name	TEXT	VARCHAR(100)	Name of stallmanger
3	stallmanger_email	TEXT	VARCHAR(100)	Email of stallmanger
4	stallmanger_contact	TEXT	VARCHAR(100)	Contact of stallmanger
5	stallmanger_password	TEXT	VARCHAR(100)	Password of stallmanger
6	stallmanger_photo	TEXT	VARCHAR(100)	Photo of stallmanger
7	stallmanger_proof	TEXT	VARCHAR(100)	Proof of stallmanger
8	stallmanger_status	INT(8)	Default value=0	Status of stallmanger

9. Table : tbl_stallrequest

Primary Key: request_id

Foreign key: event_id, stallmanager_id, stalltype_id

Description: This table stores information pertaining to the stallrequest of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	request_id	INT (8)	Primary key	Unique ID of request
2	request_date	TIMESTAMPZ	Not Null	Date of request
3	request_status	INT(8)	Not Null	Status of request
4	request_message	TEXT	Not Null	Message of request
5	event_id	INT(8)	Not Null	Unique ID of event
6	stallmanager_id	UUID	Not Null	Unique ID of stallmanager
7	stalltype_id	INT(8)	Not Null	Unique ID of stalltype

10. Table : tbl_user

Primary Key: user_id

Foreign key: place_id

Description: This table stores information pertaining to the user of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	user_id	UUID	Primary key	Unique ID of user
2	user_name	TEXT	Not Null	Name of user
4	user_email	TEXT	Not Null	Email of user
5	user_contact	TEXT	Not Null	Contact of user
6	user_password	TEXT	Not Null	Password of user
7	user_photo	TEXT	Not Null	Photo of user

11. Table : tbl_booking

Primary Key: eventbooking_id

Foreign key: event_id,user_id

Description: This table stores information pertaining to the eventbooking of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	eventbooking_id	INT (8)	Primary key	Unique ID of eventbooking
2	eventbooking_amount	TEXT	Not Null	Amount of eventbooking
3	eventbooking_status	INT (8)	Default value=0	Status of eventbooking
4	eventbooking_date	TIMESTAMPZ	Not Null	Date of eventbooking
5	eventbooking_ticketcount	TEXT	Not null	Ticketcount of eventbooking
6	user_id	UUID	Foreign key	Unique ID of user
7	event_id	INT (8)	Foreign key	Unique ID of event

12. Table : tbl_rating

Primary Key: rating_id

Foreign key: user_id, organiser_id

Description: This table stores information pertaining to the rating of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	rating_id	INT (8)	Primary key	Unique ID of Rating
2	rating_date	TIMESTAMPZ	Not Null	Date of Rating
3	rating_value	INT(8)	Default value=0	Description of Rating
4	rating_content	TEXT	Not Null	Content of Rating
5	user_id	UUID	Foreign key	Unique ID of user
6	organiser_id	UUID	Foreign key	Unique ID of organiser

13. Table : tbl_complaint

Primary Key: complaint_id

Foreign key: user_id, event_id

Description: This table stores information pertaining to the complaint of the project system.

SLNO	NAME	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	complaint_id	INT (8)	Primary key	Unique ID of Complaint
2	complaint_title	TEXT	Primary key	Title of Complaint
3	complaint_date	TIMESTAMPZ	Not Null	Date of Complaint
4	complaint_content	TEXT	Not Null	Content of Complaint
5	complaint_status	INT (8)	Default value=0	Status of Complaint
6	user_id	UUID	Foreign key	Unique ID of user
7	event_id	INT (8)	Foreign key	Unique ID of event

3.3.3. Data Flow Diagram

3.3.3.1. Introduction to Data Flow Diagrams

Data Flow Diagram is a network that describes the flow of data and processes that change, or transform, data throughout the system. This network is constructed by use a set of symbols that do not imply a physical implementation. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as inputs.

There are various symbols used in a DFD. Bubbles represent the processes. Named arrows indicate the data flow. External entities are represented by rectangles. Entities supplying data are known as sources and those that consume data are called sinks. Data are stored in a data store by a process in the system. Each component in a DFD is labelled with a descriptive name. Process names are further identified with a number.


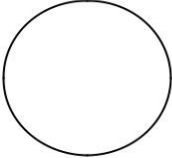


The Data Flow Diagram shows the logical flow of a system and defines the boundaries of the system. For a candidate system, it describes the input (source), outputs (destination), database (files) and procedures (data flow), all in a format that meet the user's requirements.

The main merit of DFD is that it can provide an overview of system requirements, what data a system would process, what transformations of data are done, what files are used, and where the results flow. This network is constructed by use a set of symbols that do not imply a physical implementation. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. External entities are represented by rectangles. Entities supplying data are known as sources and those that consume data are called sinks. Data are stored in a data store by a process in the system. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as inputs.

Rules for constructing a Data Flow Diagram

1. Arrows should not cross each other
2. Squares, circles and files must bear names.
3. Decomposed data flow squares and circles can have same time
4. Choose meaningful names for data flow
5. Draw all data flows around the outside of the diagram.

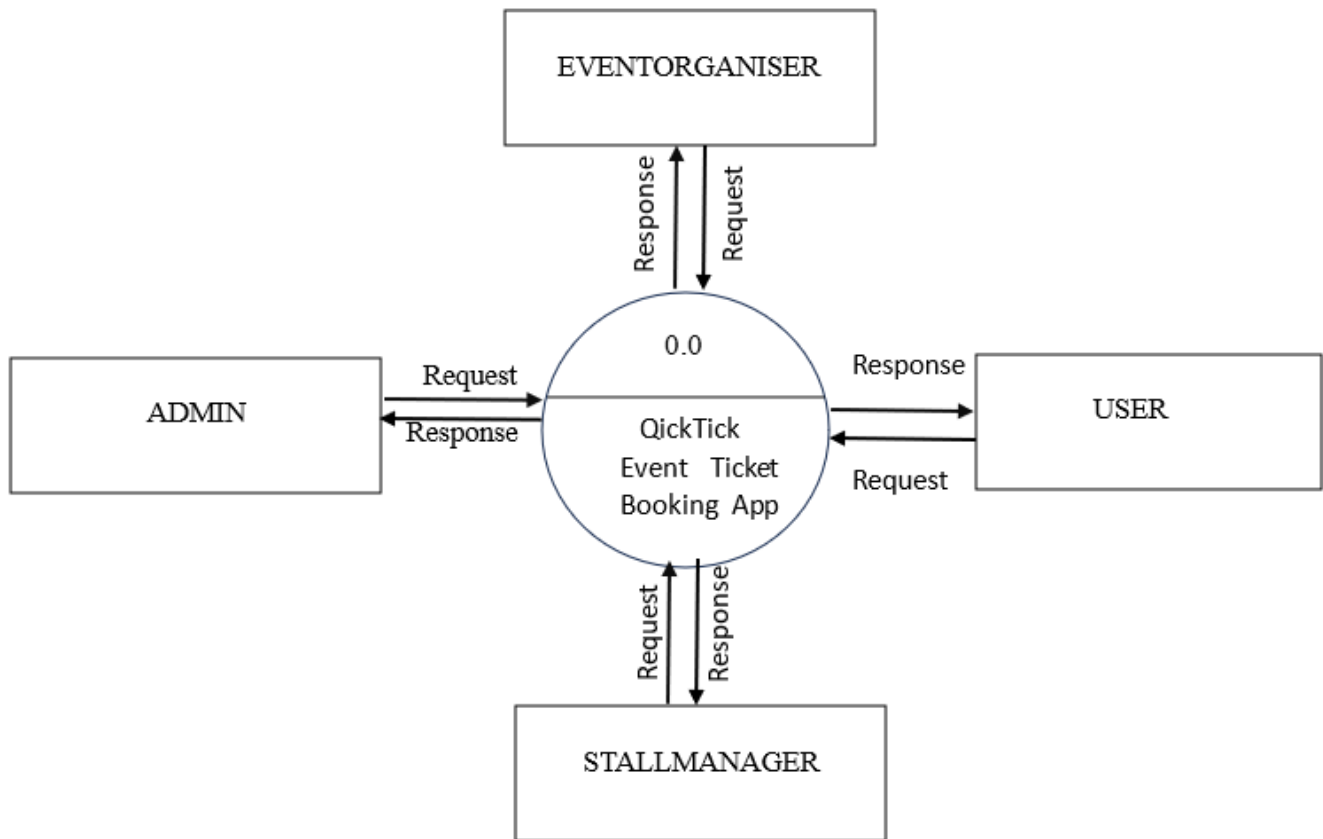
Basic Data Flow Diagram Symbols

	<p>A data flow is a route, which enables packets of data to travel from one point to another. Data may flow from a source to a process and from data store or process. An arrow line depicts the flow, with arrow head pointing in the direction of the flow.</p>
	<p>Circles stands for process that converts data into information. A process represents transformation where incoming data flows are changed into outgoing data flows.</p>
	<p>A data store is a repository of data that is to be stored for use by a one or more process may be as simple as buffer or queue or sophisticated as relational database. They should have clear names. If a process merely uses the content of store and does not alter it, the arrowhead goes only from the store to the process. If a process alters the details in the store then a double-headed arrow is used.</p>
	<p>A source or sink is a person or part of an organization, which enters or receives information from the system, but is considered to be outside the contest of data flow model.</p>

3.3.3.2. Data Flow Diagram

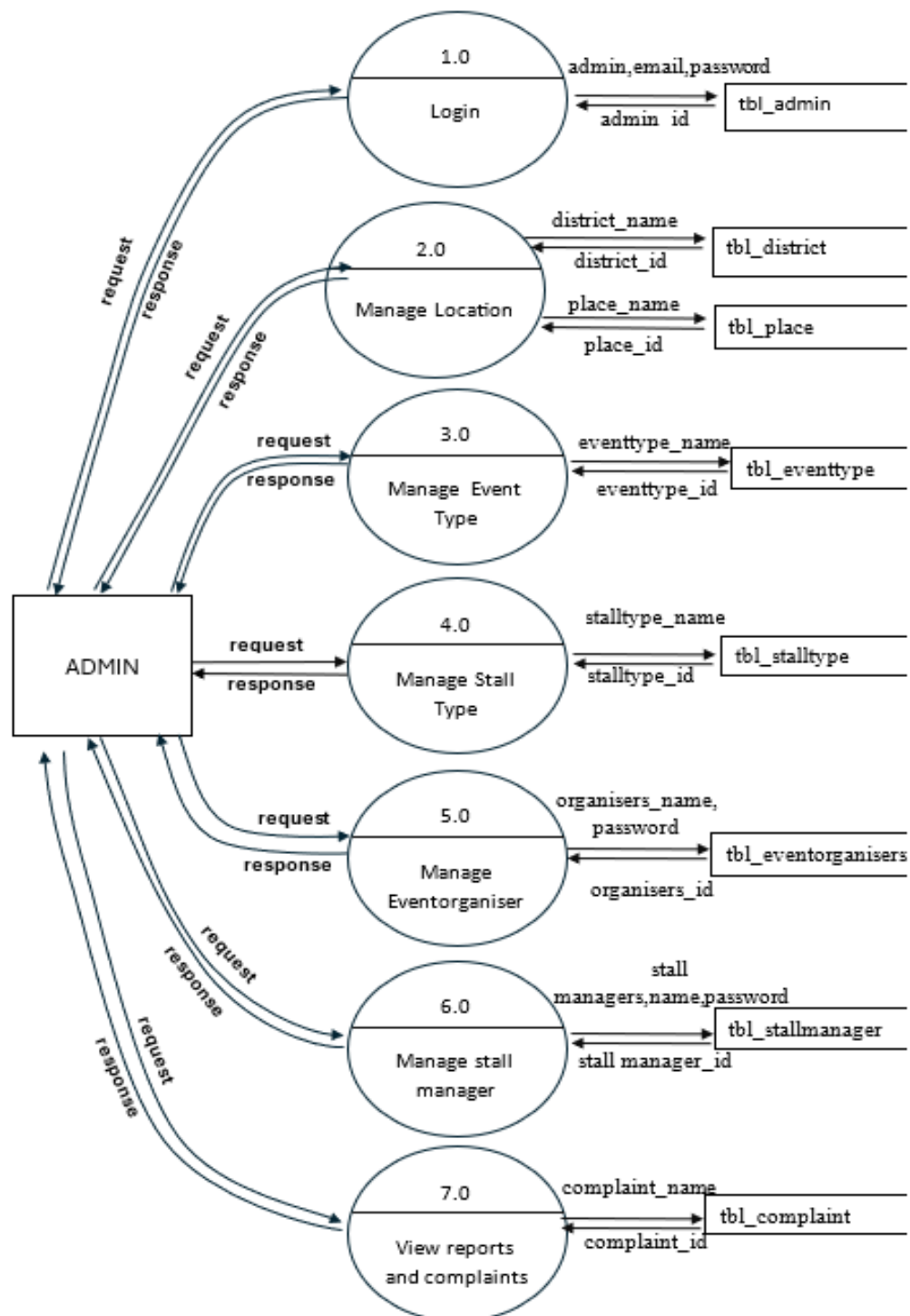
Each component in a DFD is labelled with a descriptive name. Process name are further identified with number. Context level DFD is draw first. Then the process is decomposed into several elementary levels and is represented in the order of importance. A DFD describes what data flow (logical) rather than how they are processed, so it does not depend on hardware, software, and data structure or file organization.

A DFD methodology is quite effective; especially when the required design.

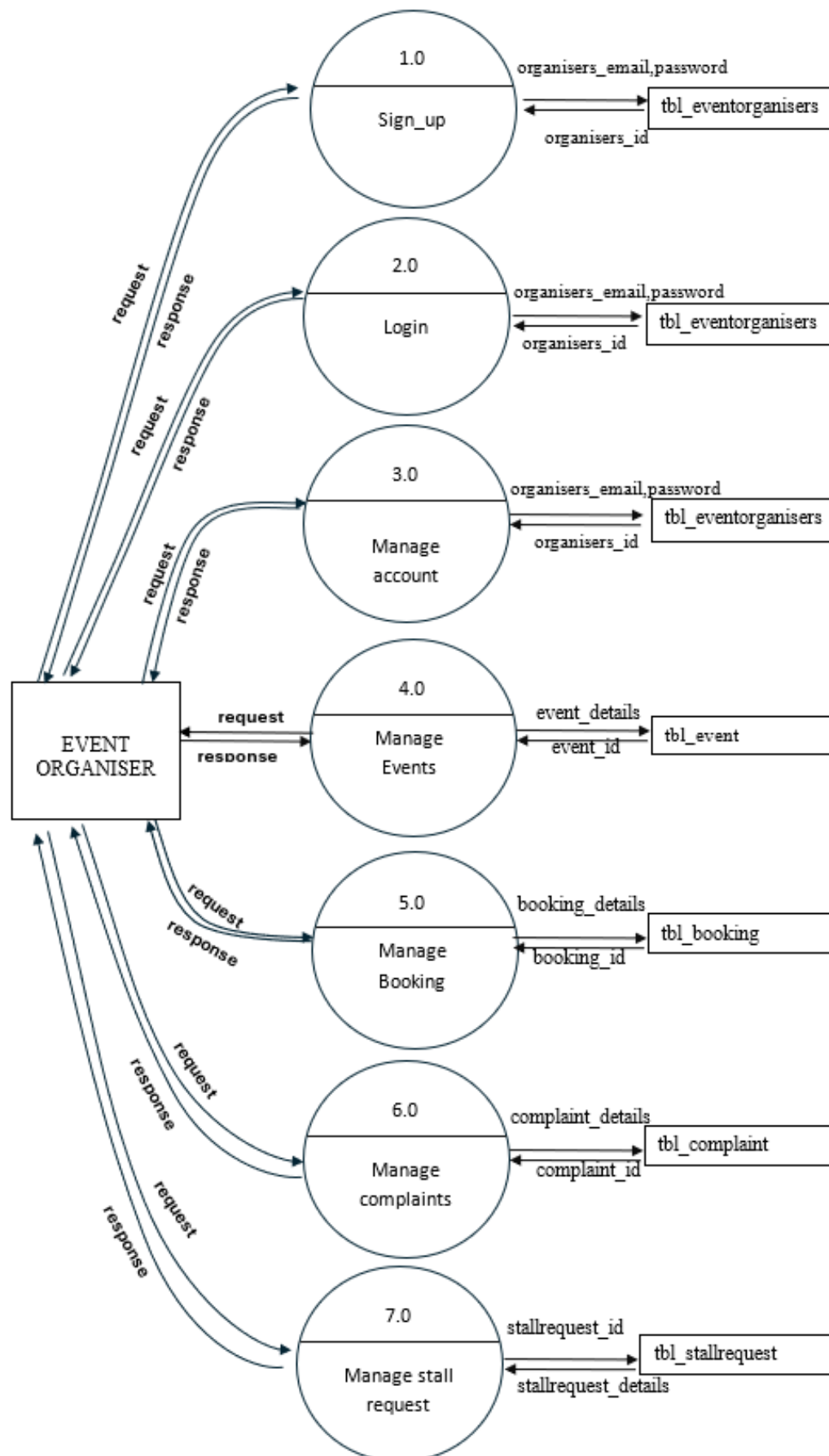
LEVEL 0

LEVEL 1

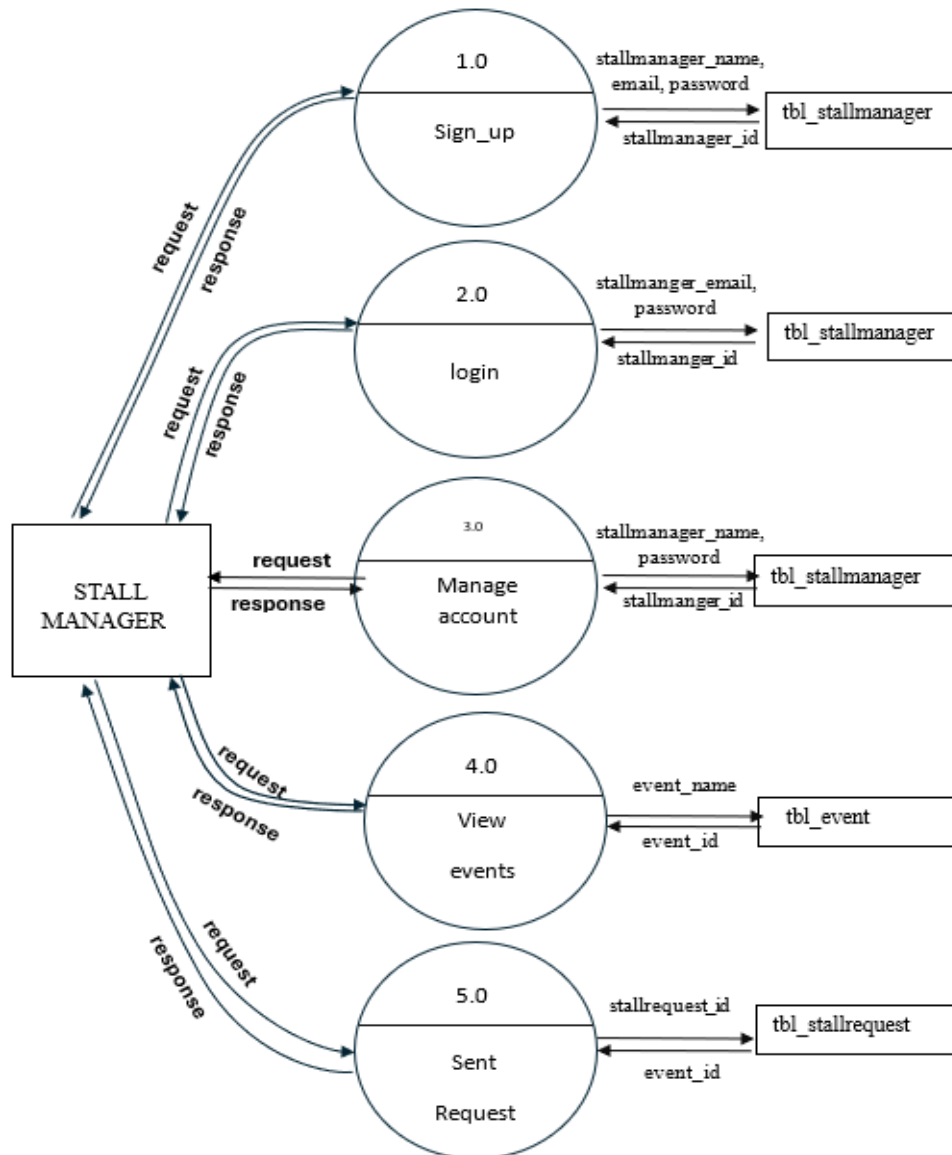
1) ADMIN



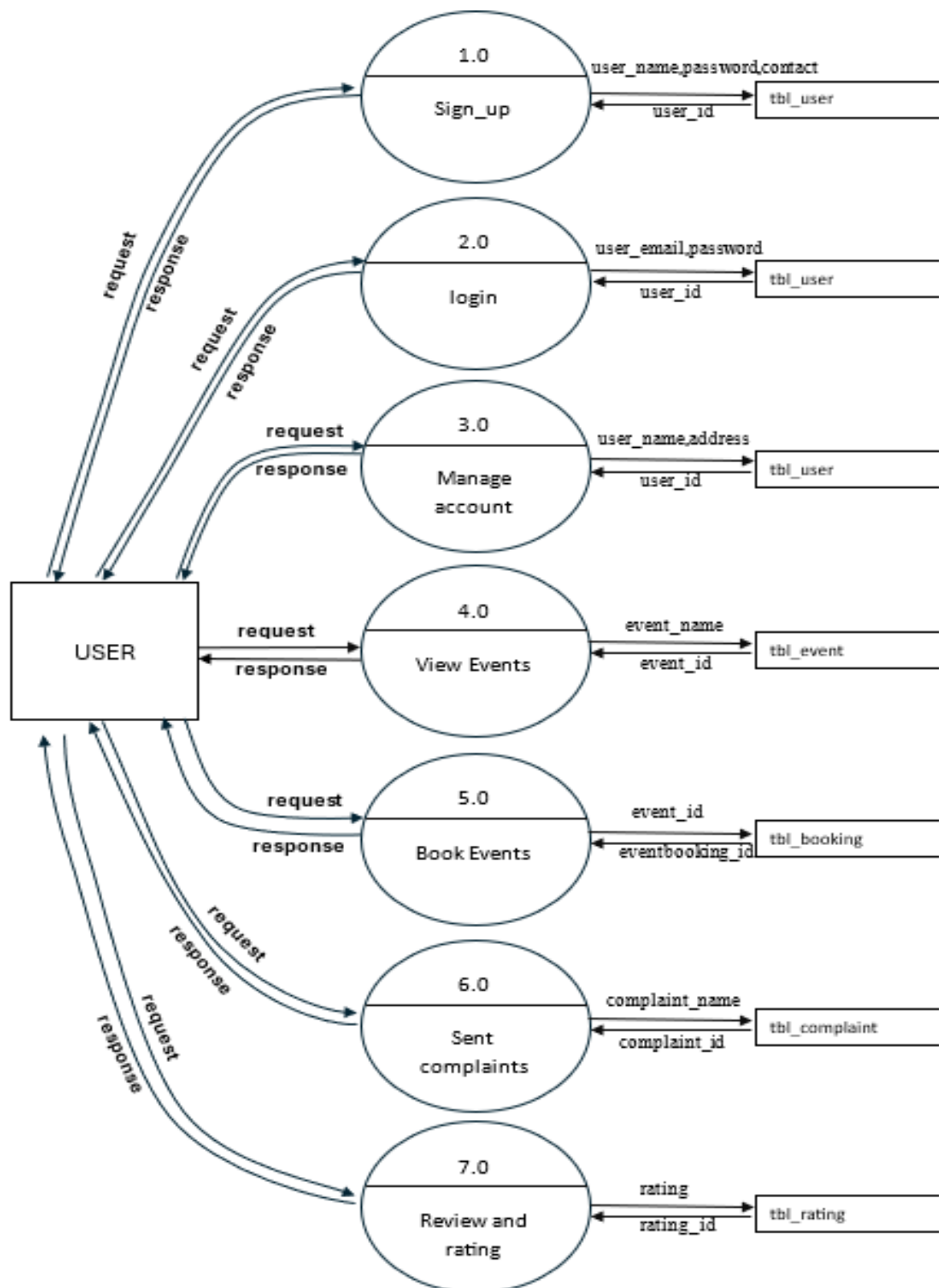
2) EVENT ORGANISER



3) STALL MANAGER

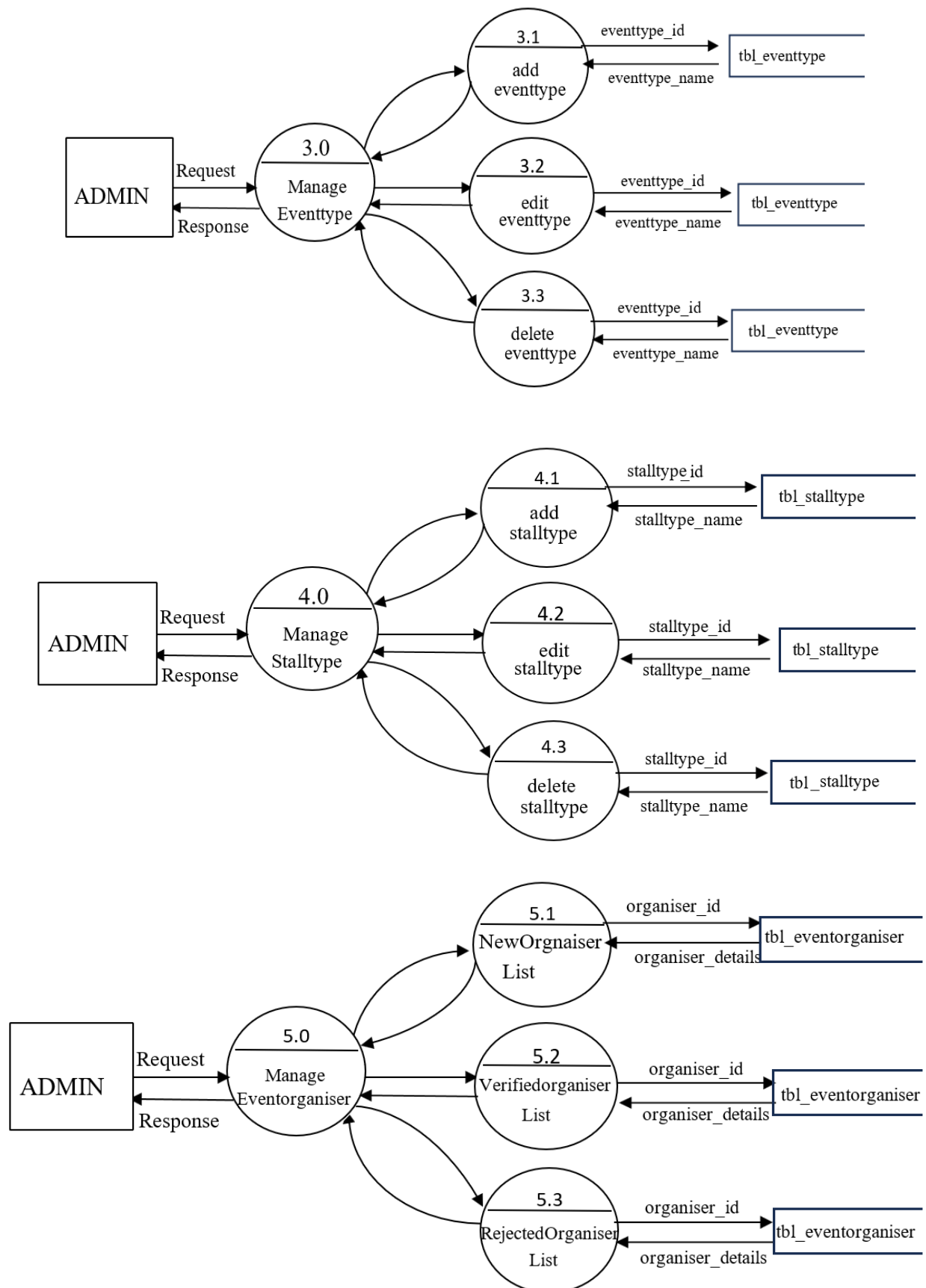


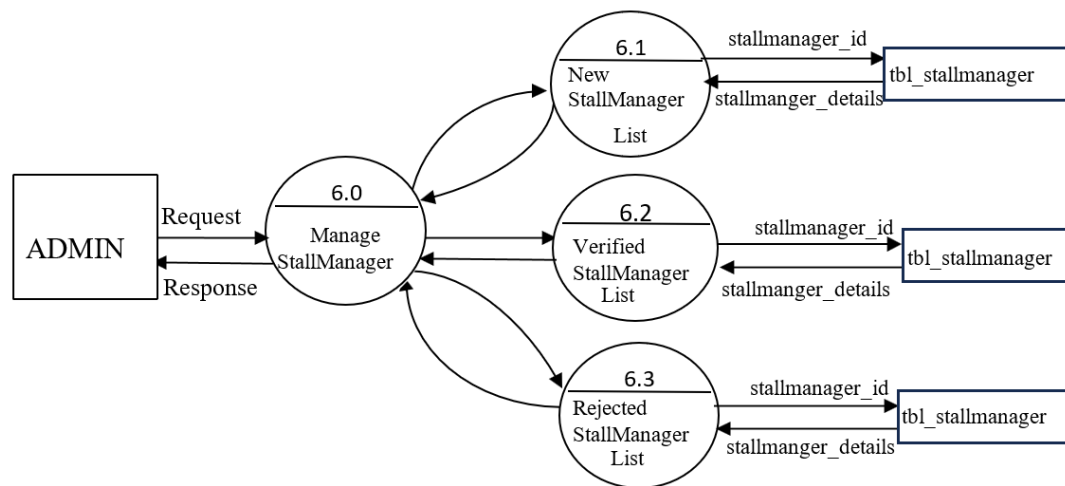
4) USER



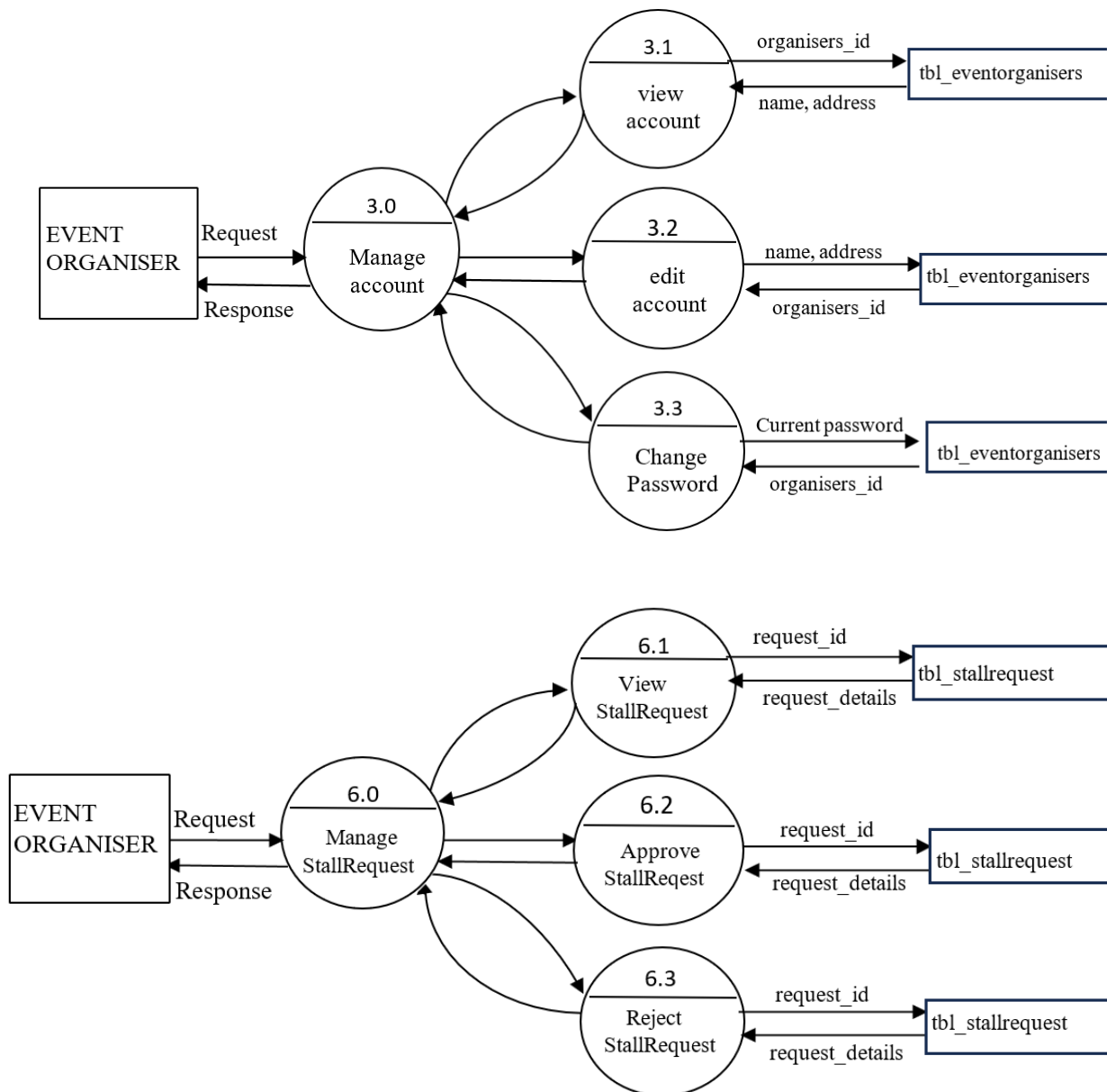
LEVEL 2

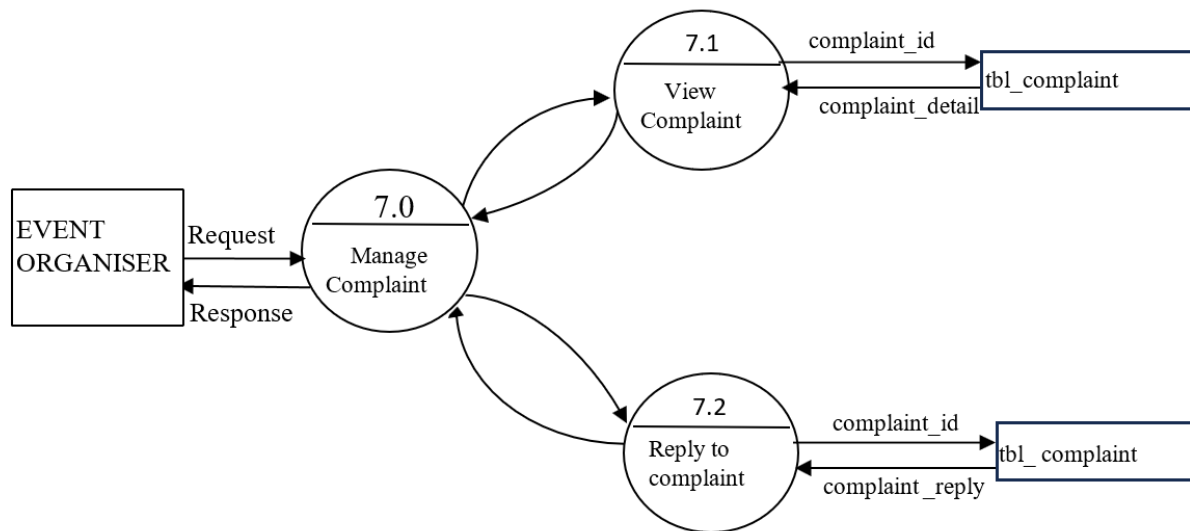
1) ADMIN



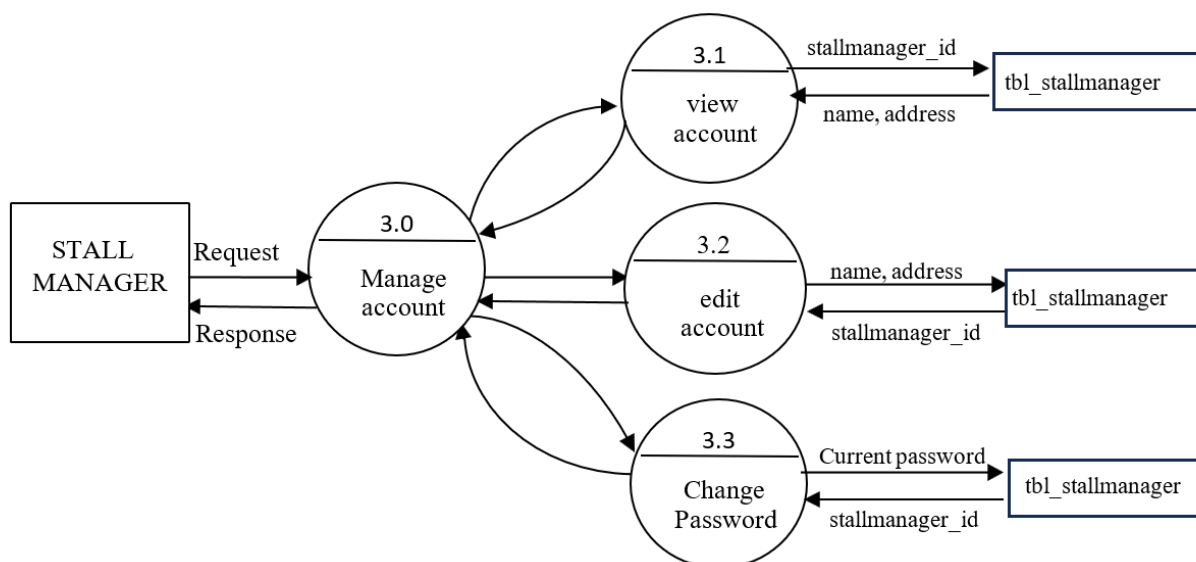


2) EVENT ORGANISER

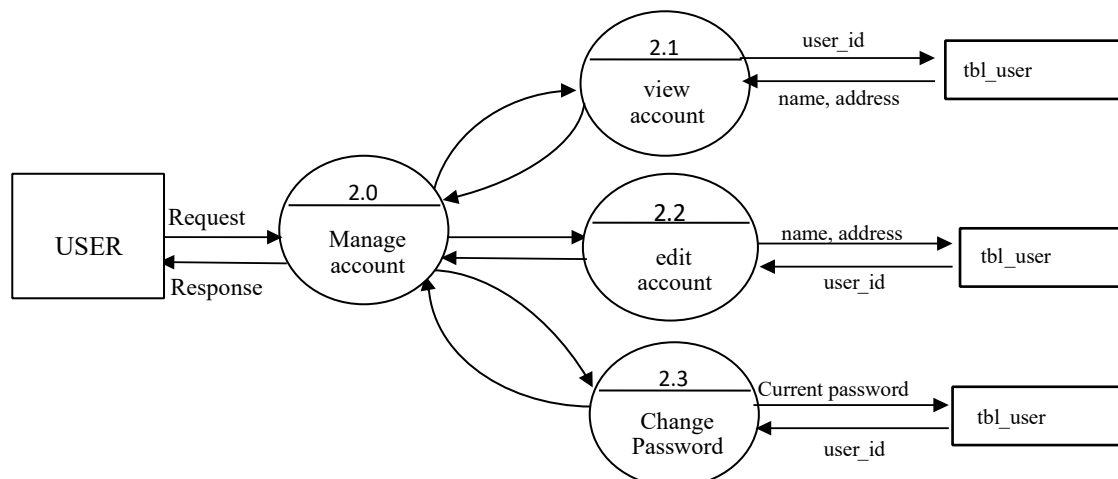


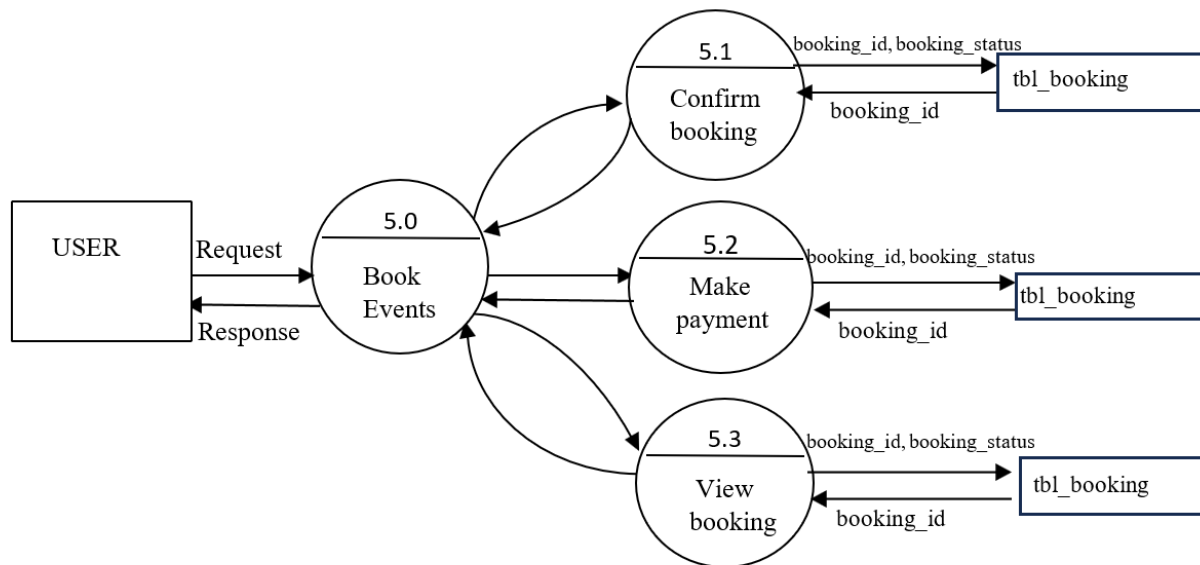


3) STALL MANAGER



4) USER





3.4. INTERFACE DESIGN

These modules can apply to hardware, software or the interface between a user and a machine. An example of a user interface could include a GUI, a control panel for a nuclear power plant, or even the cockpit of an aircraft. In systems engineering, all the inputs and outputs of a system, subsystem, and its components are listed in an interface control document often as part of the requirements of the engineering project. The development of a user interface is a unique field.

3.4.1. User Interface Screen Design

The user interface design is very important for any application. The interface design describes how the software communicates within itself, to system that interpreted with it and with humans who use it. The input design is the process of converting the user-oriented inputs into the computer-based format. The data is fed into the system using simple inactive forms. The forms have been supplied with messages so that the user can enter data without facing any difficulty. They data is validated wherever it requires in the project. This ensures that only the correct data have been incorporated into system. The goal of designing input data is to make the automation as easy and free from errors as possible. For providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right messages and help for the user at right are also considered for development for this project.

Input Design is a part of the overall design. The input methods can be broadly classified into batch and online. Internal controls must be established for monitoring the number of inputs and for ensuring that the data are valid. The basic steps involved in input design are:

- Review input requirements.

- Decide how the input data flow will be implemented.
- Decide the source document.
- Prototype on line input screens.
- Design the input screens.

The quality of the system input determines the quality of the system output. Input specifications describe the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce results from accurate data. The input design also determines whether the user can interact efficiently with the system.

3.4.2. Output Design

A quality output is one, which meets the requirements of end user and presents the information clearly. In any system result of processing are communicated to the user and to the other system through outputs. In the output design it is determined how the information is to be displayed for immediate need.

It is the most important and direct source information is to the user. Efficient and intelligent output design improves the system's relationships with the user and helps in decision -making. The objective of the output design is to convey the information of all the past activities, current status and to emphasis important events. The output generally refers to the results and information that is generated from the system. Outputs from computers are required primarily to communicate the results of processing to the users.

Output also provides a means of storage by copying the results for later reference in consultation. There is a chance that some of the end users will not actually operate the input data or information through workstations, but will see the output from the system.

Two phases of the output design are:

1. Output Definition
2. Output Specification

Output Definition takes into account the type of output contents, its frequency and its volume, the appropriate output media is determined for output. Once the media is chosen, the detail specification of output documents are carried out. The nature of output required from the proposed system is determined during logical design stage. It takes the outline of the output from the logical design and produces output as specified during the logical design phase. In a project, when designing the output, the system analyst must accomplish the following:

- Determine the information to present.

- Decide whether to display, print, speak the information and select the output medium.
- Arrange the information in acceptable format.
- Decide how to distribute the output to the intended receipt.
- Thus by following the above specifications, a high quality output can be generated.

4. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned into a working system. The implementation stage is a systems project in its own right. It includes careful planning, investigation of current system and its constraints on implementation, design of methods to achieve the changeover, training of the staff in the changeover procedure and evaluation of changeover method.

4.1. CODING STANDARDS

Writing an efficient software code requires a thorough knowledge of programming. This knowledge can be implemented by following a coding style which comprises several guidelines that help in writing the software code efficiently and with minimum errors. These guidelines, known as coding guidelines, are used to implement individual programming language constructs, comments, formatting, and so on. These guidelines, if followed, help in preventing errors, controlling the complexity of the program, and increasing the readability and understandability of the program.

A set of comprehensive coding guidelines encompasses all aspects of code development. To ensure that all developers work in a harmonized manner (the source code should reflect a harmonized style as a single developer had written the entire code in one session), the developers should be aware of the coding guidelines before starting a software project. Moreover, coding guidelines should state how to deal with the existing code when the software incorporates it or when maintenance is performed.

Since there are numerous programming languages for writing software codes, each having different features and capabilities, coding style guidelines differ from one language to another. However, there are some basic guidelines which are followed in all programming languages. These include naming conventions, commenting conventions, and formatting conventions.

1. **File header comments** are useful in providing information related to a file as a whole and comprise identification information such as date of creation, Name of the creator, and a brief description of the software code.

2. **Trailing comments** are used to provide explanation of a single line of code. These comments are used to clarify the complex code. These also specify the function of the abbreviated variable names that are not clear. In some languages, trailing comments are used with the help of a double slash (//).

3. **Indentation:** This refers to one or more spaces left at the beginning of statements in the program. Indentation is useful in making the code easily readable. However, the spaces used for indentation should be followed in the entire program.

4. **Implementing coding guidelines:** If coding guidelines are used in a proper manner, errors can be detected at the time of writing the software code. Such detection in early stages helps in increasing the performance of the software as well as reducing the additional and unplanned costs of correcting and removing errors. Moreover, if a well-defined coding guideline is applied, the program yields a software system that is easy to comprehend and maintain.

TESTING

5. TESTING

Coding conventions are a set of guidelines for a specific programming language that recommend programming style, practices and methods for each aspect of a piece program written in this language. These conventions usually cover file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, etc. These are guidelines for software structural quality. Software programmers are highly recommended to follow these guidelines to help improve the readability of their source code and make software maintenance easier.

5.1. TEST CASES

The objective of system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and to ensure that the computer system and the associated clerical and other procedures work together. The initial phase of system testing is the responsibility of the analyst who determines what conditions are to be tested, generates test data, produced a schedule of expected results, runs the tests and compares the computer produced results with the expected results with the expected results. The analyst may also be involved in procedures testing. When the analyst is satisfied that the system is working properly, he hands it over to the users for testing. The importance of system testing by the user must be stressed. Ultimately it is the user must verify the system and give the go-ahead.

During testing, the system is used experimentally to ensure that the software does not fail, i.e., that it will run according to its specifications and in the way users expect it to. Special test data is input for processing (test plan) and the results are examined to locate unexpected results. A limited number of users may also be allowed to use the system so analysts can see whether they try to use it in unexpected ways. It is preferably to find these surprises before the organization implements the system and depends on it. In many organizations, testing is performed by person other than those who write the original programs. Using persons who do not know how certain parts were designed or programmed ensures more complete and unbiased testing and more reliable software.

Parallel running is often regarded as the final phase of system testing. Since he parallel operation of two systems is very demanding in terms of user resources it should be embarked on only if the user is satisfied with the results of testing -- it should not be started if problems are known to exist. Testing is the major quality control measure during software development.

Its basic function is to detect errors in the software.

Thus the goal of testing is to uncover requirement design and coding errors in the program. Testing is the process of correcting a program with intends of finding an error. Different types of testing are,

1. Unit Testing

2. Integrated Testing
3. Black Box Testing
4. White Box Testing
5. Validation Testing
6. User Acceptance Testing

5.1.1. Unit Testing

In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In this testing we test each module individual and integrated the overall system. Unit testing focuses verification efforts on the smaller unit of software design in the module. This is also known as module testing. The modules of the system are tested separately. The testing is carried out during programming stage itself. In this testing step each module is found to working satisfactory as regard to the expected output from the module. There are some validation checks for verifying the data input given by the user which both the formal and validity of the entered. It is very easy to find error debug the system.

5.1.2. Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Data can be lost across an interface; one module can have an adverse effect on the other sub functions when combined by, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. This testing was done with sample data. The developed system has run success full for this sample data. The need for integrated test is to find the overall system performance.

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. Integration testing identifies problems that occur when units are combined. By using a test plan that requires you to test each unit and ensure the viability of each before combining units, you know that any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis. Progressively larger groups of tested software

components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

5.1.3. Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well. In black box testing the structure of the program is not considered. Test cases are decided solely on the basis of the requirements or the specification of the program or module, and the internals of the module or program are not considered for selection of the test cases.

In the Black Box testing tester only knows the input that can be given to the system and what output the system should give. In other words, the basis of deciding test cases in functional testing is requirements or specifications of the system or module. This form of testing is also called functional or behavioural testing. One advantage of the black box technique is that no programming knowledge is required.

5.1.4. White Box Testing

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements. White Box testing is concerned with testing the implementation of the program. The intent of this testing is not to exercise all the different input or output conditions but to exercise the different programming structures and data structures used in the program.

5.1.5. Validation Testing

At the culmination of Black Box testing, software is completely assembled as a package, interface errors have been uncovered and corrected and final series of software tests, Validation tests begins. Validation testing can be defined many ways but a simple definition is that validation succeeds when

the software functions in a manner that can be reasonably accepted by the customer. After validation test has been conducted one of the two possible conditions exists.

1. The function or performance characteristics confirm to specification and are accepted.
2. A derivation from specification uncovered and a deficiency list is created.

We have given various validations in our forms so that there will be a neat format for the data's that are entered on to the website. We have also given an already existing validation so that the data redundancy is reduced; same data is not entered twice.

5.1.6. User Acceptance Testing

Acceptance Testing is a level of the software testing process where a system is tested for acceptability. User Acceptance testing is the software testing process where system tested for acceptability & validates the end to end business flow. Such type of testing executed by client in separate environment & confirms whether system meets the requirements as per requirement specification or not.

UAT is performed after System Testing is done and all or most of the major defects have been fixed. This testing is to be conducted in the final stage of Software Development Life Cycle (SDLC) prior to system being delivered to a live environment. UAT users or end users are concentrating on end to end scenarios & typically involves running a suite of tests on the completed system.

User Acceptance testing also known as Customer Acceptance testing (CAT), if the system is being built or developed by an external supplier. The CAT or UAT are the final confirmation from the client before the system is ready for production. The business customers are the primary owners of these UAT tests. These tests are created by business customers and articulated in business domain languages. So ideally it is collaboration between business customers, business analysts, testers and developers. It consists of test suites which involve multiple test cases & each test case contains input data (if required) as well as the expected output. The result of test case is either a pass or fail.

CONCLUSION

6. CONCLUSION

The project entitled “**QUICKTICK**” was completed on time. QuickTick is a comprehensive event ticket booking platform designed to streamline the ticketing process for organisers, vendors, and attendees. With its user-friendly interface and robust management features, QuickTick ensures a seamless experience for booking event tickets and managing stall reservations. The Stall Manager module plays a vital role in optimizing event space distribution by allowing stall manager to oversee stall availability, allocate spaces to vendors, and coordinate with event organisers. By integrating features like real-time booking updates, vendor communication, and detailed reporting, QuickTick enhances event efficiency and ensures smooth operations. Whether managing ticket sales, stall bookings, or vendor queries, QuickTick provides a reliable solution for hassle-free event planning, making it the go-to platform for event organisers and participants alike.

FUTURE ENHANCEMENTS

6.1. FUTURE ENHANCEMENTS

Future enhancements for your event ticket booking system could include implementing a dynamic recommendation engine that suggests events based on user preferences and past bookings. Adding a loyalty program with reward points for frequent users can improve engagement. Integrating real-time chat support for customer queries and assistance would enhance user experience. Expanding payment options to include digital wallets and UPI for smoother transactions can increase convenience. Enhancing security features such as biometric login or two-factor authentication would ensure safer user data protection. Additionally, incorporating an analytics dashboard for organisers to track ticket sales, attendee demographics, and event performance can provide valuable insights. These enhancements would improve functionality, security, and user satisfaction, making your system more robust and efficient.

BIBLIOGRAPHY

7. BIBLIOGRAPHY

1. SYSTEM ANALYSIS AND DESIGN
Elias M Award,
Second Edition, 1999,
Galgotica Publications, Delhi
2. DATABASE SYSTEM CONCEPT
Silberschatz, Korth, Sudarshan, Fourth Edition.
3. https://en.wikipedia.org/wiki/Operating_system
4. <http://ecomputernotes.com/software-engineering/feasibilitystudy>
5. https://en.wikipedia.org/wiki/Third_normal_form
6. <https://www.guru99.com/user-acceptance-testing.html>
7. https://www.tutorialspoint.com/system_analysis_and_design.html
8. <http://softwaretestingfundamentals.com/test-case/>
9. <http://softwaretestingfundamentals.com/black-box-testing/>
10. <https://www.geeksforgeeks.org/asp.net-coding-standards/>
11. <https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/>

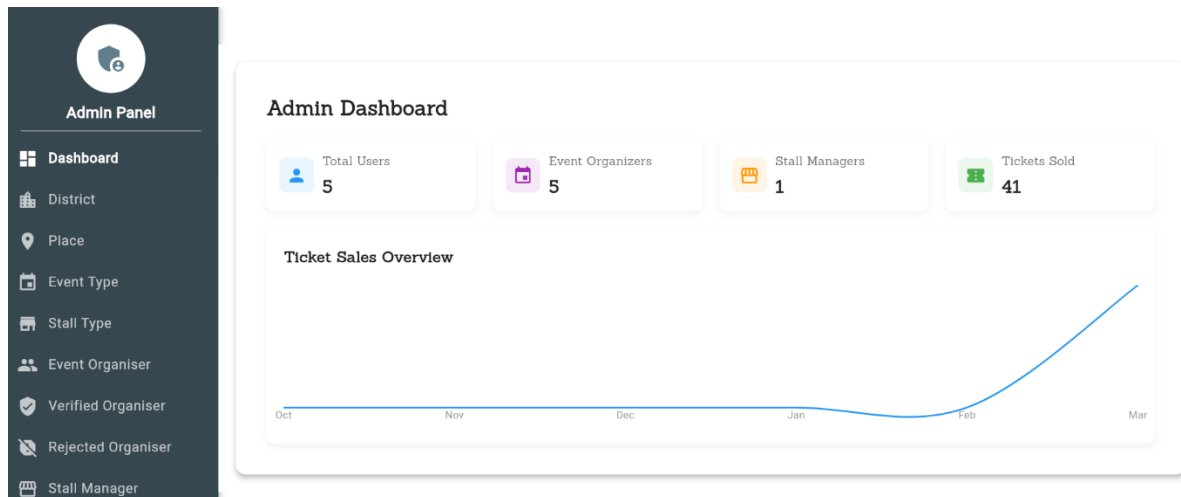
APPENDIX

8. APPENDIX

8.1. SCREENSHOTS

Website UI

FORM 1- ADMIN DASHBOARD



FORM 2- ADDING DISTRICT

Add District

District:

SUBMIT

Added Districts


1	Ernakulam		
2	Thrissur		
3	Kannur		
4	Thiruvananthapuram		
5	Kollam		

FORM 3- MANAGE EVENT ORGANISER

Manage Event Organizers

Name	Email	District	Address	Contact	Photo	Proof	Act
Anagha S	anagha123@gmail.com	Kannur	Thirumarady	9874563210			
Manish	manish123@gmail.com	Kollam	Kollam	9876543210			
Sunita	sunita123@gmail.com	Pathanamthitta	Pathanamthitta	7896541230			
Vivek	vivek@gmail.com	Alappuzha	Alappuzha	8976543210			



FORM 4- VERIFIED EVENT ORGANISER




Admin Panel

- Dashboard
- District
- Place
- Event Type
- Stall Type

Verified Event Organizers

Name	Email	District	Address	Contact	Photo	Proof	Action
anagha	anagha@gmail.com	Ernakulam	thirumarady	2369857120			Reject



FORM 5 – REJECTED EVENT ORGANISER




Admin Panel

- Place
- Event Type
- Stall Type
- Event Organiser
- Verified Organiser
- Rejected Organiser

Rejected Event Organizers


Name	Email	District	Address	Contact	Photo	Proof	Action
Anjana	anjana@gmail.com	Thrissur	Thrissur	9876543210			Approve


FORM 6 – LOGIN PAGE




Login

Sign in to your account

 Email Address

 Password




[LOGIN](#)

FORM 7 – REGISTRATION PAGE

Register as Organizer

Fill in the details below to get started.



Full Name

Email Address

District

Place

Contact Number

Address

Upload Proof






Password

Confirm Password

[REGISTER](#)

FORM 8 – EDIT PROFILE


Profile Details


	Name Anagha S
	Email anagha123@gmail.com
	Contact 9874563210
	Address Thirumarady
	Place Thalassery


[Edit Profile](#)[Change Password](#)

FORM 9 – CREATE EVENT

Create New Event


Upload Event Image
Supports JPG, PNG (Max 5MB)




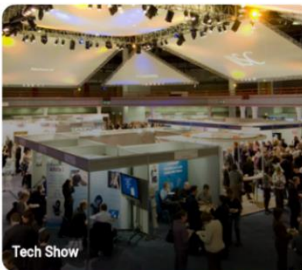



[Create Event](#)


FORM 10 – MY EVENTS


[←](#) **My Events** [+ Create](#)



Library Expo


Tech Show


FutureConf


MegaConf


Confexus


Music Fest

FORM 11 – USER BOOKING

User Booking

Anju

Tech Show
 Date: 24-03-25
 Tickets: 3

Anuu

MegaConf
 Date: 24-03-25
 Tickets: 1

Anuu

Library Expo
 Date: 21-03-25
 Tickets: 19

FORM 12 –EVENTS DETAILS

Event Details

Date: 28-02-25
 Time: 13:15:00
 Duration: 1 hour
 Tickets: Available Tickets: 17

Event Overview

Join us for an insightful conference featuring industry experts, engaging discussions, and networking opportunities.

Stall Requests

Ratings

Complaints

FORM 13 – STALL REQUESTS

Stall Manager Requests

Anjana Ginesh

Book
 sbgswsgqus

Approved

Anjana Ginesh

Merchandise Stall
 I am requesting for this stall

Approve

Reject

FORM 14 – USER REVIEWS

User Reviews

Anuu

great

★★★★★

Anuu


good

★★★★★

FORM 15 – USER COMPLAINTS

←

User Complaints



good

By: Anuu

Event: Library Expo

2025-03-16

Details:

Good

Contact: 9807654321

New

↑

Reply

FORM 16 – STALL REQUEST

←

Event Details

Library Expo

Date: 01-03-25

Time: 13:30:00

Duration: 2 hours

Send Request

Additional Notes

Submit Request

Event Overview


Explore a wide collection of books, meet authors, and enjoy literary discussions at the book fair.

Apply Request

FORM 17 – MY REQUESTS

←

My Requests



Music Fest


Stall Type: Book

Event Venue: Karunagappally, Kollam

Stall Details: sbgswsqqs

Event Details: Get ready for an unforgettable night filled with electrifying performances, mesmerizing lights, and an incredible atmosphere. Feel the energy of live music as talented artists take the stage to create an experience like no other!

Approved



FutureConf

Stall Type: Merchandise Stall

Event Venue: Kunnankulam, Thrissur

Stall Details: I am requesting for this stall

Event Details: Join us for an insightful conference featuring industry experts, engaging discussions, and networking opportunities.

Pending

App UI

FORM 18- APP LOGIN

LOGIN



LOG IN

Don't have an account? [Register](#)

FORM 19-APP REGISTRATION



REGISTRATION



REGISTER

Have an account? [Sign In](#)

FORM 20- APP HOMEPAGE



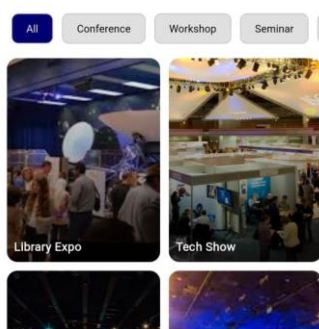
New Events

[VIEW ALL](#)



Popular Events

[VIEW ALL](#)



Home



FORM 21 -APP SEARCHPAGE

All

Conference

Workshop

Seminar



Library Expo



Tech Show



FutureConf



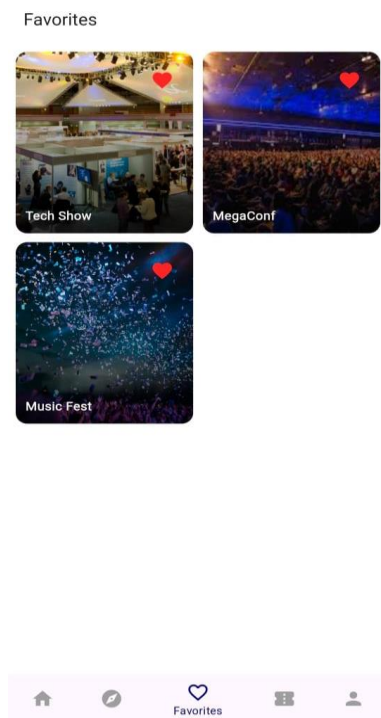
MegaConf



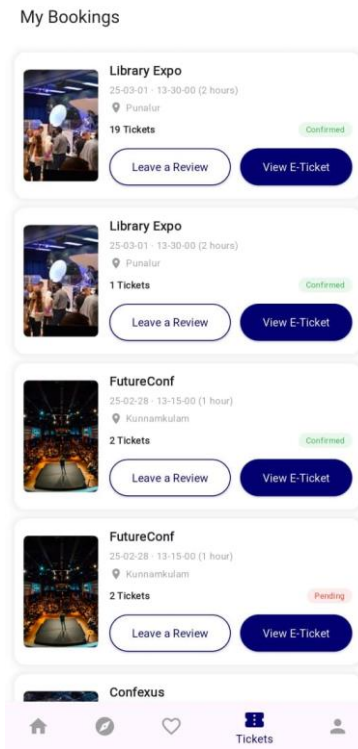
Explore



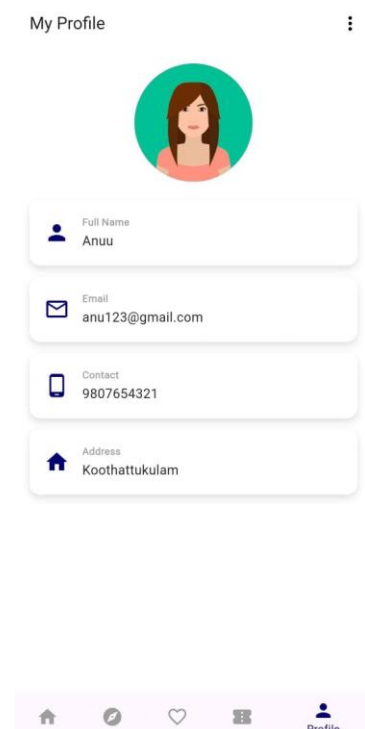
FORM 22 - APP FAVORITES



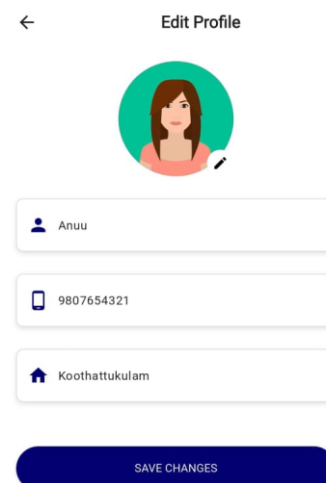
FORM 23 – APP MYBOOKINGS

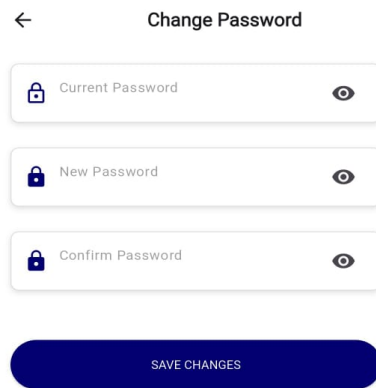


FORM 24 - APP MYACCOUNT



FORM 25 - APP EDITPROFILE



FORM 26 - APP CHANGE PASSWORD

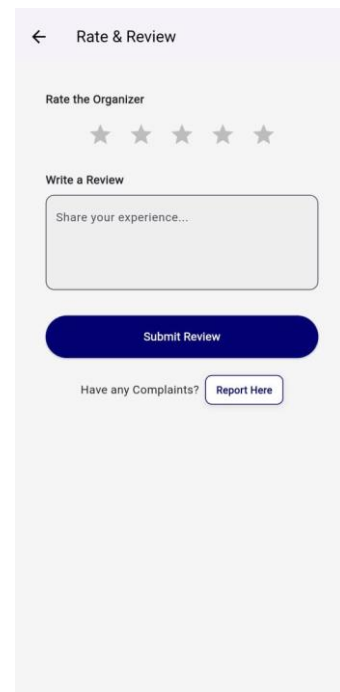
Change Password

Current Password

New Password

Confirm Password

SAVE CHANGES

FORM 27 - APP REVIEW

Rate & Review

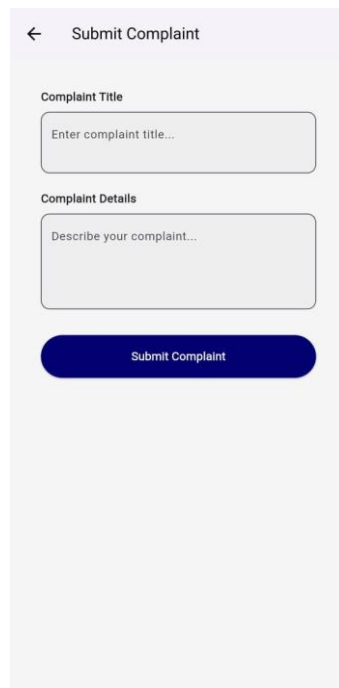
Rate the Organizer

Write a Review

Share your experience...

Submit Review

Have any Complaints? [Report Here](#)

FORM 28 - APP COMPLAINT

Submit Complaint

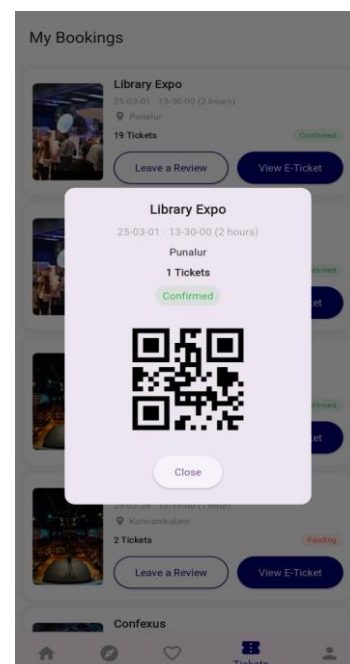
Complaint Title

Enter complaint title...

Complaint Details

Describe your complaint...

Submit Complaint

FORM 29 - APP VIEW TICKET

My Bookings

Library Expo

25-03-01 - 13-30-00 (2 hours)

Punalur

19 Tickets

Confirmed

Leave a Review

View E-Ticket

Library Expo

25-03-01 - 13-30-00 (2 hours)

Punalur

1 Tickets

Confirmed

Close

Confexus

Tickets