**CHAPTER THREE**

**SYSTEM ANALYSIS AND DESIGN**

**3.0 Introduction**

This chapter highlighted the software development model use in the project, the requirement engineering and its processes and also the system design.

**3.1 Software Development Model**

A Software or system process model is a description of the sequence activities carried out in a software development project, and the relative order of these activities. There are various software development approaches which are employed during the development process of software. Each process model follows a particular life cycle in order to ensure success in the process of software development. For this study, the software development model adopted is described below.

**3.1.1 Agile software development method**

Agile software development is a modern and iterative approach to software development that prioritizes flexibility, collaboration, and customer satisfaction. It contrasts with traditional, sequential methods (like the Waterfall model) by breaking the development process into smaller, manageable increments called iterations or sprints (López-Alcarria *et al.,* 2019).



Fig 3.1 Agile methodology (techcabal.com)

Agile software development is a set of principles and practices that emphasize adaptability, collaboration, and customer-centricity throughout the software development life cycle (SDLC). Key characteristics of Agile include:

1. Iterative Development: The project is divided into small increments or iterations. At the end of each iteration, a potentially shippable product increment is delivered.
2. Customer Collaboration: Continuous feedback from customers, stakeholders, and end-users is actively sought and integrated into the development process.
3. Cross-functional Teams: Agile teams are typically small, cross-functional groups that include developers, testers, designers, and other necessary roles. These teams are self-organizing and work closely together.
4. Prioritization and Flexibility: Requirements and priorities can change throughout the project, and Agile embraces these changes, allowing for flexible adaptation to evolving customer needs and market conditions.
5. Emphasis on Individuals and Interactions: Agile values individuals and their interactions over processes and tools, fostering open communication and collaboration.
6. Working Software as a Measure of Progress: The primary measure of progress is working, potentially shippable software rather than comprehensive documentation or plans.

Agile software development is a customer-centric, flexible, and collaborative approach that is well-suited for projects where requirements may change, and rapid adaptation is necessary. It promotes faster delivery, improved quality, and reduced risk, ultimately leading to more successful software projects.

**3.2 REQUIREMENT ENGINEERING**

Requirement engineering refers to the process of defining, documentation, and maintaining requirements in the engineering design process, it provides the appropriate mechanism to understand what the customer desires, analyzing the need and accessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating specification and managing the requirements as they are transformed into a working system.

Requirement engineering it is a four process, which are requirements elicitation, requirements analysis, requirements validation and requirement management.

1. Requirement elicitation is the process of communication and collaborating with key stakeholders to assemble the insight and identify the project’s need.
2. Requirement analysis is the process of defining the expectation of the users for an application that is to be built or modified.
3. Requirement validation is the process of confirming that the system meets its objectives and functions as intended.
4. Requirements management is the process of managing changing requirements during the requirements engineering process and system development.

Below is a report of each of the requirement engineering process for the project;

**3.2.1 REQUIREMENT ELICITATION**

Although there are different requirement elicitation techniques such as interview, questionnaires, brainstorming and prototyping, but interview was used for the project.

**3.2.2 REQUIREMENTS ANALYSIS**

Based on the analysis and review on the overall requirement, the functional and non-functional requirement of the system was discovered and identified.

Functional requirement is the requirement that describes what the software system should do or those. It is a must to have these requirements. Below are some of the functional requirement;

1. The system should be able to be viewed on any browser.
2. The system should allow user and admin to login
3. The system should allow user to register in an event
4. The system should allow user to book an event venue
5. The system should allow users to view event venues
6. The system should allow admin to manage event venues
7. The system should allow admin to manage bookings

Non-functional requirement is the requirement that define the software system attribute such as security, performance, maintenance, and usability. Below are some of the non-functional requirement.

1. Ability to take input fast
2. Ability to be deployed in any environment
3. The software should be portable i.e. one OS to another
4. The system should be user friendly
5. The system should be responsive

**3.2.3 REQUIREMENTS VALIDATION**

The requirements of the project are validated, the requirements are verifiable, comprehensible, traceable and also adaptable.

**3.2.4 REQUIREMENTS MANAGEMENT**

In requirements management of the project, changing requirements, new ideas and feedback are collected from customers and other stakeholders, there are refined to align with the overall business and product strategy and then translate the ideas into features and requirements.

**3.2.5 FEASIBILITY REPORT**

This feasibility report evaluates the viability of developing the platform, considering technical, economic, and operational aspects.

1. **Technical Feasibility:**

The platform technical feasibility assesses the platform's infrastructure, development requirements, and compatibility with existing technologies. Key considerations include:

* Platform Development: it requires skilled developers proficient in web development, database management, and user interface design.
* Infrastructure: Adequate server capacity, storage, and bandwidth are needed to support user interactions and data management.
* Compatibility: The platform should be compatible with different devices (desktops, tablets, mobile devices) and web browsers to ensure accessibility for a wide user base.

1. **Economic Feasibility:**

The economic feasibility of the platform examines the financial viability of the project. Key factors to consider include:

* Development Costs: The costs associated with platform development, including hiring developers, UI/UX designers, and database management specialists.
* Operational Costs: Ongoing expenses related to server hosting, maintenance, security measures, and content moderation.

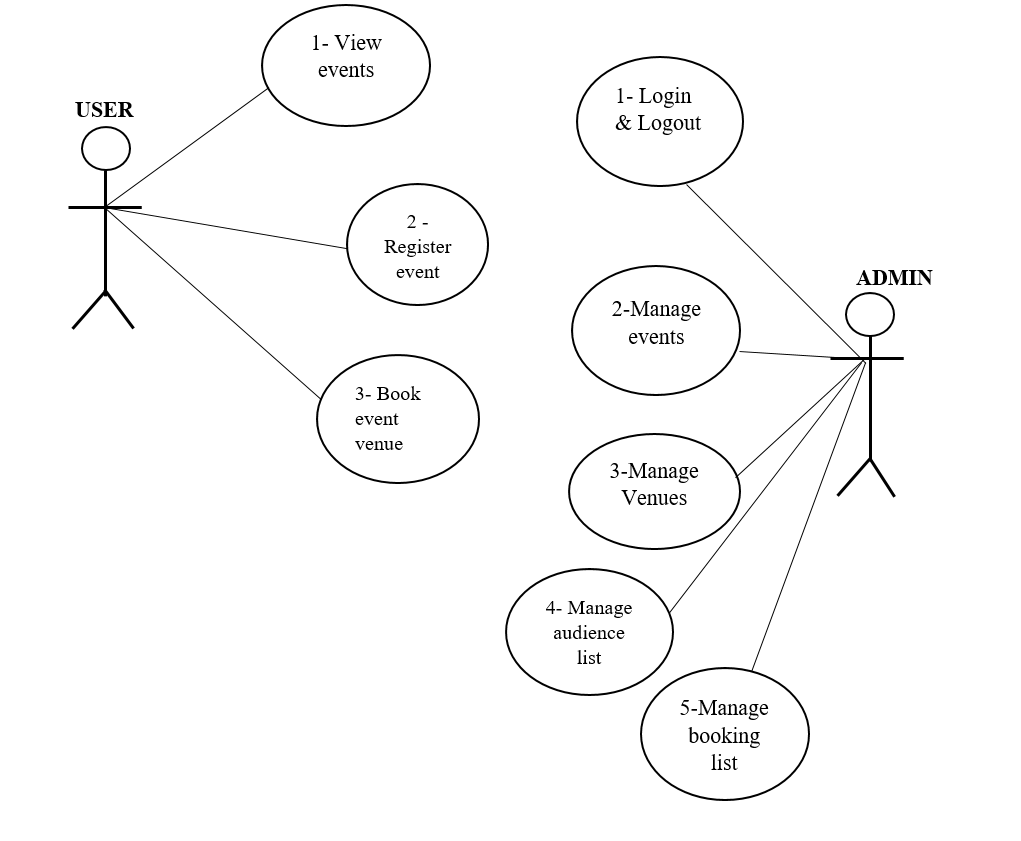
1. **Operational Feasibility:**

Operational feasibility assesses the practicality of running the platform smoothly and efficiently. Key considerations include:

* Resource Availability: Ensuring access to necessary resources, such as event data, user engagement, and technical support.
* Content Moderation: Implementing efficient content moderation mechanisms to maintain a positive user experience and ensure compliance with community guidelines.
* Scalability: Planning for future growth and the ability to handle an increasing number of users.

Based on the assessment of technical, economic, and operational feasibility, the platform demonstrates strong potential for successful implementation. The platform requires a skilled development team, adequate infrastructure. The revenue generation potential and user adoption factors contribute to the overall viability of the platform

**3.3 SYSTEM DESIGN**



***Fig 3.1 Use case diagram of the system.***

**3.3.1 USE CASE**

A use case diagram is a visual representation of the interactions between actors (users or external systems) and a system, showcasing the various use cases (functionalities) of the system. Below is the use case diagram of the platform

**User:**

1. View Events: Browse and view event details.

2. Register Event: Sign up for events.

3. Book Event Venue: Select and book a venue.

**Admin:**

1. Login & Logout: Access and exit the system securely.

2. Manage Events: Create, update, or delete events.

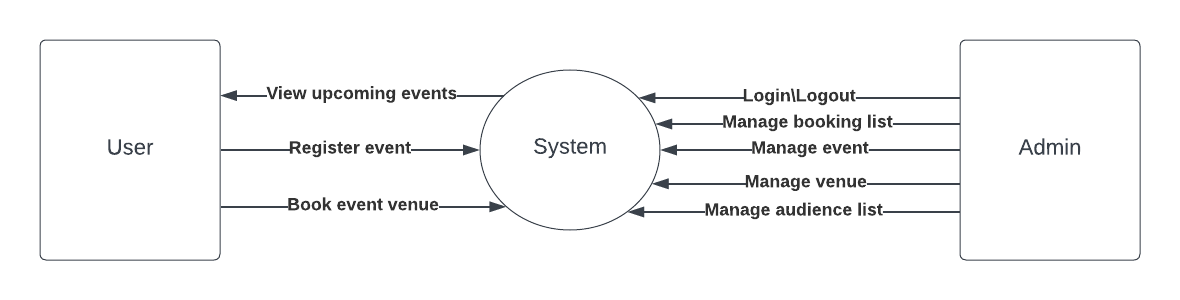
3. Manage Venues: Add, update, or remove venues.

4. Manage Audience List: Oversee attendee lists.

5. Manage Booking List: Handle and manage venue bookings.

**3.3.2 DATA FLOW DIGRAM**

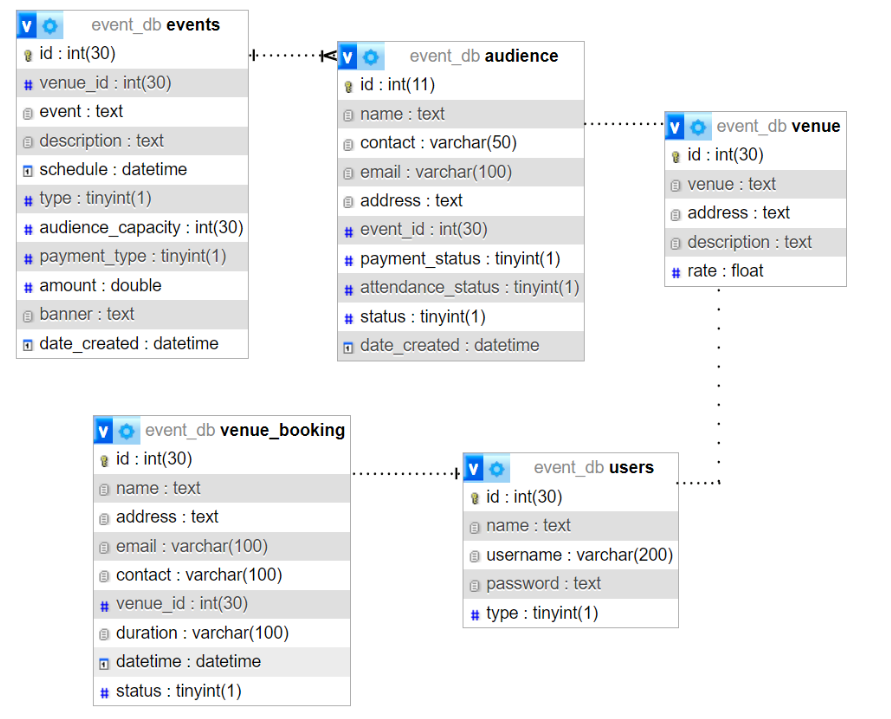
A dataflow diagram (DFD) is a graphical representation of the flow of data through a system. It illustrates how data is processed by a system in terms of inputs and outputs.



***Fig 3.2 Dataflow diagram of the system***

**3.3.3 ENTITY RELATIONSHIP DIAGRAM**

An Entity-Relationship (ER) diagram is a visual representation of the entities (objects), attributes (properties), and relationships between entities in a database system. It helps to model the structure and organization of data within a system and illustrates how different entities are related to each other. Below is the entity relationship diagram of the system



***Fig 3.3 entity relationship diagram of the system***

Key

Primary key

Relationship**-------------**

One 

Many 

**3.4 CHAPTER SUMMARY­**

The main focus of this chapter was the software development model use in the project and how it works, requirement engineering of the project and making report on each of its processes and also the design tool use for the system design.