

**Software Engineering and Testing. BSC Year 2, 2020/2021**

**(Assignment 3 - 20%)**

**Assessment 3: Design and Draft Implementation**

**Submitted by: B00149860 - Ibrahim Camara B00155039 - Hayden Carr**

**B00143012 – Ibrahim Abushaban**

**Declaration**

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ordinary Degree in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated.

Author: B00149860 - Ibrahim Camara Dated:16/03/24

Author: B00155039 - Hayden Carr Dated:16/03/24

Author: B00157525 - Joshua Hand Dated:16/03/24

Author: B00143012 – Ibrahim Abushaban Dated:16/03/24

**Table of Contents**

1. Project Definitions..............…..… Page 3
2. Methodology.........................….… page 5
3. Requirements ........................……. page 6

3.1 Use Cases ......................… …..page 6

3.2 Use Case Specifications..........… page 7

1. Case Diagrams...........…………….. page 7
2. Conclusion………………………... page 9

# Title: Hii Events

# *Abstract / Executive Summary (200 Words max)*

# Project Definitions

This document is a comprehensive guide and reference for the development and use of the "Hii Events," an innovative platform developed with the purpose of changing the way in which people handle and manage events. Mainly, this document addresses software developers, project managers, and the stakeholders involved in the project to provide a general understanding among all concerned parties towards the objectives, specifications, and technical framework under which "Hii Events" shall operate. It describes the functional specification, software components, and goals of the project that would allow joint efforts for delivering the solution that will improve the access to purchase tickets for events and booking venues by taking into account the needs of event organizers, venue owners, and attendees.

**The primary objectives of "Hii Events" include:**

**Enhancing Event Accessibility**: Making it easier for attendees to find and participate in events.

**Streamlining Ticket Purchasing**: Simplifying the process of buying tickets, from selection to payment and issuance.

**Facilitating Venue Booking**: Offering an efficient system for event organisers to find and book venues.

**Improving User Experience**: Ensuring a user-friendly interface for all stakeholders.Functional Specifications

**User Roles and Permissions**

* **Admin**: Full system access, including user management and content approval.
* **Event Organiser**: Can create and manage event listings, view booking requests, etc.
* **Customer**: Can browse events, purchase tickets, and submit venue rental enquiries.

**Event Listing and Management**

Which means an organiser can easily list his events, manage the events' details, and potential participants' search and filter of the events by location, date, and other criteria.

**Ticket Purchasing Process**

A seamless process from selection to purchase includes robust payment processing and issuing digital tickets.

**Venue Rental Quotation System**

It solicits and generates quotations for the renting out of venues with a system aiming to simplify the booking process for them.

**User Accounts and Profiles**

Detail account creation and profile management features emphasise the privacy and security in handling the way user data is availed.

**Notifications and Alerts**

A comprehensive notification system to keep users informed about important updates related to their events or accounts. Document Revision

# 3. Methodology

**OOAD**

OOAD is essential for designing and developing complex software systems. It helps in breaking down the system into manageable components. OOAD facilitates modularity, reusability, and scalability.

**CLASSES**

Classes encapsulate data and methods related to a concept or entity in the system. Classes help in organizing code, promoting reusability, and modular structures. They represent the blueprint for creating objects.

**CLASS DIAGRAMS**

A class diagram is represents the structure of a system by showing the classes in the system, their attributes, methods, and relationships with other classes. Class diagrams provide a visual representation of the static aspects of the system's design.

Class diagrams depict the static structure of a system, showing classes and their relationships, Dynamic diagrams such as sequence diagrams or activity diagrams, represent the dynamic behavior of the system, showing how objects interact over time.

**ERD**

An ERD is a visual representation of the entities and their relationships in a database. It illustrates the logical structure of a database.

**VOLATILE VS PERSISTENT**

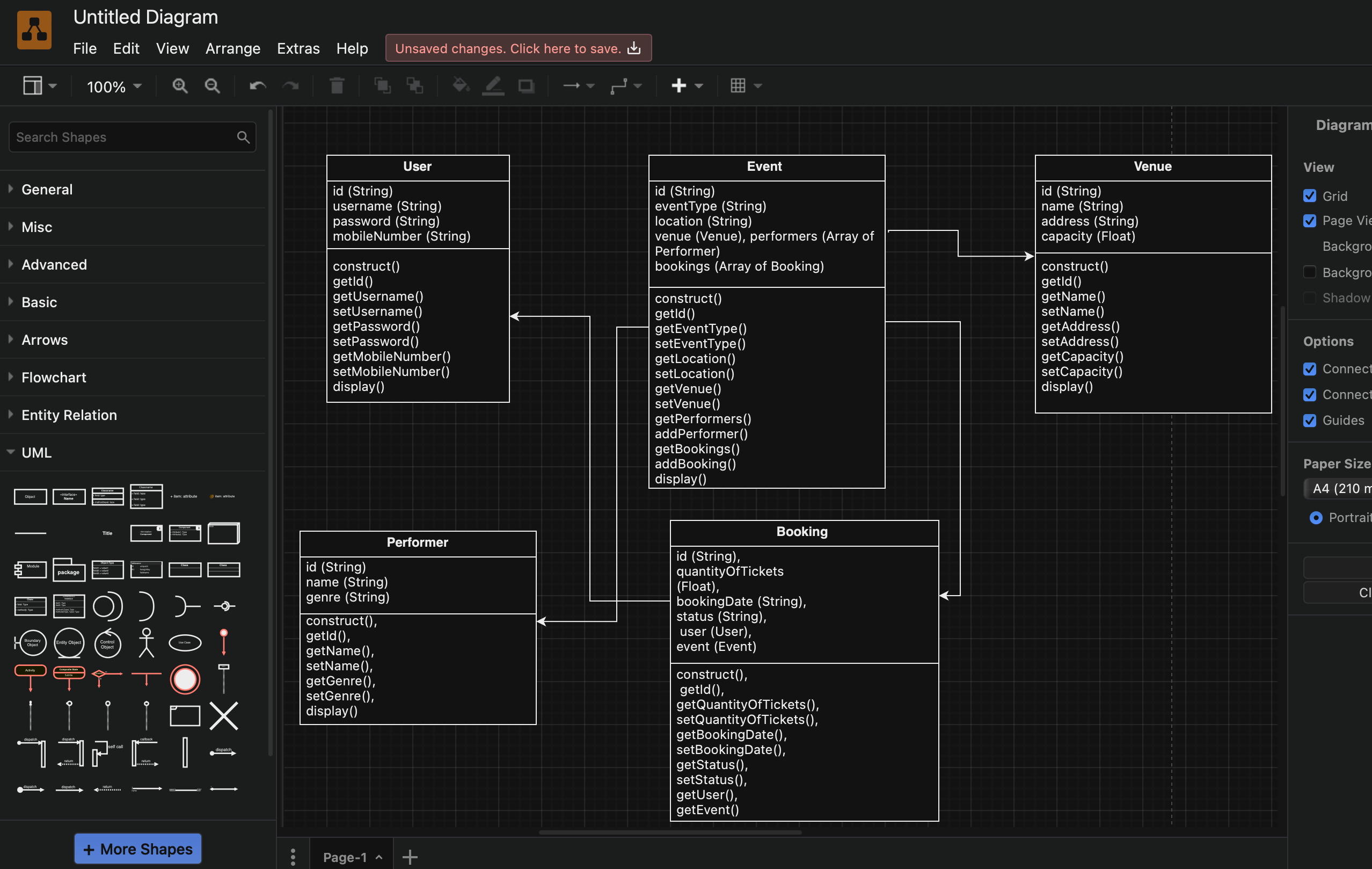
Volatile storage refers to temporary storage that loses its contents when the browser is shut down or restarted. Persistent storage retains its contents even when the system is shut down or restarted.

**4. Requirements**

4.1 Use Cases

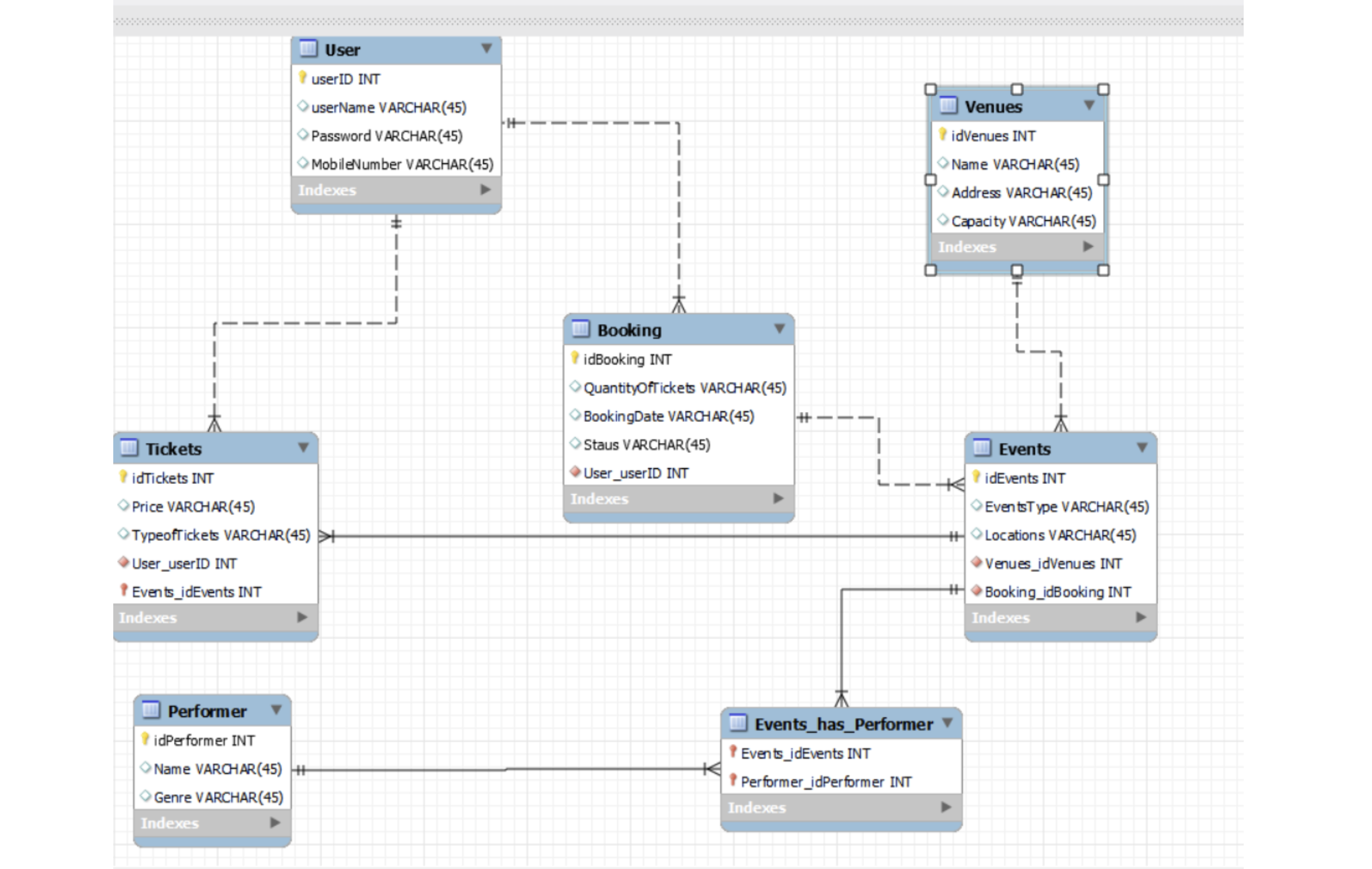
4.2 Use Case Specifications

(Specifically – how use case specifications have been used as a means to develop the ***classes/attributes/methods*** and database ***tables***)

**5. Case Diagrams**

* **Classes**: **User**, **Booking**, **Venue**, **Event**, and **Performer** are the primary classes.
* **Composition (Aggregation)**: **Event** class has a composite aggregation relationship with **Venue** and **Performer** indicating that an event is composed of these elements. This is represented by a filled diamond connector from **Event** to **Venue** and **Performer**.
* **Association**: **User** class has a one-to-many association with **Booking** because a single user can have multiple bookings.

The purpose of class diagram design is to summarise the core functionality and relationships of an event management system. The User class encapsulates all the attributes needed for a user's identity and interaction within the system. A reservation is associated with a user through a one-to-many relationship, reflecting the concept that a user can make multiple reservations for an event. Events have a complex relationship between venue and performer, meaning that an event is defined by its location and the performers in attendance. This structure emphasises that the event lifecycle is dependent on the venue and performers. Finally, the mapping is intentionally kept simple to make the system easier to understand and future extensibility



* **Entities**: Include **User**, **Booking**, **Venue**, **Event**, and **Performer**.
* **Relationships**: **User** and **Booking** have a one-to-many relationship. **Event** and **Venue** also have a one-to-many relationship, but between **Event** and **Performer**, it's a many-to-many relationship.

ERD design decisions reflect practical considerations regarding database normalization and efficient data management. Each entity corresponds to a potential table in the database, and the attributes are used as fields within those tables. The one-to-many relationship between users and bookings allows you to centralize user information, reduce duplication, and ensure reservations are traced back to individual users. Event to Venue one-to-many approach from event to venue, reflecting the reality that venues can accommodate multiple events over time. Many-to-many relationships between events and performers are resolved by link tables, making it easy to assign multiple performers to multiple events, a common scenario when hosting events

# 6.Conclusions

In conclusion the design and draft implementation of HII Events have allowed for clearer planning due to the visualization it has provided. Through the use of Object-Oriented Analysis and Design methodology we are wroking on a system that caters to the needs of event organisers,vanue owners and attendees.

Our class diagrams and ERD provide a clear blueprint for the structure of the website ensuring a comprehensive understanding of its design. The project contains functionalities crucial for event management.

The project encompasses functionalities essential for efficient event management, laying a strong foundation for further development and refinement. As we continue to improve our goals of developing a user-friendly platform remain the same.

Your conclusions and recommendations as to how far the project has progressed.

Your changes to the original proposal that the design has revealed and necessitated.

Additional sections: Table of Contents, Table of Figures, References, Index

Checklist: Is your document complete and correct?

*Content:*

* Does the design include all requirements from the customers’ needs
* Are you satisfied with all parts of the document?
* Do you believe all parts have been implemented?
* Have you explained your methodology and design choices?
* Have you clearly articulated your understanding of the purpose of all diagrams created ?
* What are these diagrams? Why you need them? How were they developed?
* Is each part of the document in agreement with all other parts?
* Does the design create a solution for the initial proposal?

*Completeness*:

* Are all the necessary components specified?
* Are the design specifications precise enough?
* Are all sections from the document template included – if changed, why?

*Clarity*:

* Is the design reasonable?
* Is the level of details for each design section appropriate?
* Is the design written in a language appropriate to the intended audience of software engineering teams?
* Are all items clear and unambiguous?