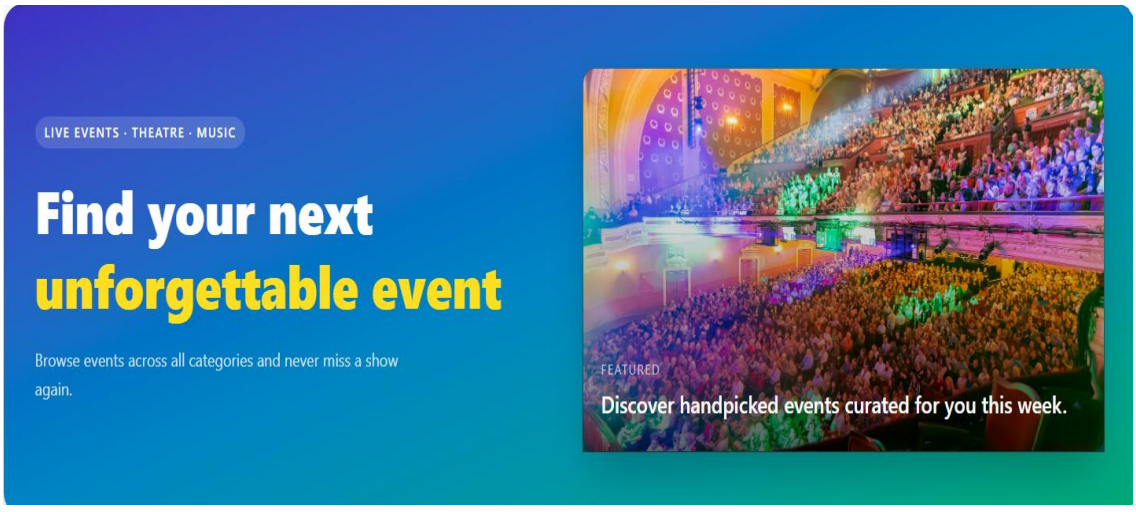
PRODUCT REPORT-DIXEN

Dixen: Event Management, Discovery & Ticketing System



DIKSHYA SINGH SHAH
CLASS:1205hf2601per

18. FEBRUAR 2026

TECHNICAL EDUCATION COPENHAGEN
Guldblommevej 5, Ballerup

Dikshya Singh Shah

Class:1205hf2601per

Project Title: Product Report of Dixen

(Event Management, Discovery & Ticketing System)

Project Name: Dixen

Name: Dikshya Singh Shah

Date: 19/01/2026 -18/02/2026

School: TEC Ballerup

Class: 1205hf2601per

Word count: 45.387 characters with space

Total Figures and Tables: 5 and 5 respectively

Supervisor: Flemming Sørensen

Signature fields

Signatures:

Participant: _____ Date: _____

DIKSHYA SINGH SHAH

Dikshya Singh Shah

Class:1205hf2601per

TABLE OF CONTENTS

List of Tables 5

List of FIGURES 5

List of Abbreviations Used in Dixen Report 6

1. READING GUIDE 7

2. Introduction 7

2.1. Purpose 7

2.2 scope 8

2.3 Stakeholders 8

2.4 Justification for a One-Person Project 8

2.5 Deviations from the Requirements Specification 9

3. General Description 9

3.1 System Description 9

3.2 System Worldview 9

3.3 System Limitations 10

3.4 Future Enhancements 11

3.5 User Profiles 11

3.6 Development Process Requirements 11

3.7 Documentation 12

3.8 Software Testing 12

4. Product Description 12

4.1 Product functionality 12

User Experience 13

Technical Perspective 13

5. Requirement Specification 14

5.1 System of Functional Requirements 14

Description of Flow: 15

5.2 Fulfilment of Functional Requirements 15

5.2.1 User Registration and Authentication 15

5.2.2 Two-Factor Authentication (2FA) 15

5.2.3 Event Discovery and Search 16

5.2.4 Event Proposal and Approval Workflow 16

5.2.5 Ticket Booking and Capacity Management 16
5.2.6 Role-Based Access Control (RBAC) 16
5.2.7 Social Media Sharing 16
5.2.8 Analytics Dashboard 16
5.2.9 google Maps Navigation Link 16
5.3 Fulfilment of Non-Functional Requirements 17
5.3.1 Performance 17
5.3.2 Security 17
5.3.3 Usability 17
5.3.4 Reliability and Maintainability 17
5.4 Constraints and Deviations 17
5.5 Traceability to Requirement Specification 18
5.6 Data Requirements 18
5.7 Use Case diagram and Descriptions 19
5.7.1 Dixen use case Diagram 19
5.7.2 Implementation use case (3 of 8 detailed below) 20
6. Estimated schedule 23
7. User manual/ Getting Started 24
7.1 Registering a New Account 24
7.2 Logging In 24
Login with Two-Factor Authentication (2FA) 24
7.3 Browsing and Registering for Events 24
Browsing Events 24
7.4 Registering for an Event 24
7.5 Submitting Event Proposals (For Users) 25
7.6 Managing Events (For Organizers) 25
Creating Events 25
7.7 Editing or Deleting Events 25
7.8 Sharing Events on Social Media 25
7.9 Admin Controls (For Admins) 25
7.10 Troubleshooting & Support 26
8. Installation and Configuration 26
8.1 Technologies Used 26

8.2 NuGet Packages 26
8.3 Database Server 27
8.4 Getting Started 27
8.4.1 Install Prerequisites 27
8.4.2 Commands 27
9. Technical Architecture and implementation 28
9.1 System/Technical Architecture Overview 28
9.1.1 Frontend 28
9.1.2 Backend 28
9.1.3 Security 28
9.2 Hardware Description 29
9.2.1 Device with Internet Access 29
9.2.2 Web Browser 29
9.2.3 Internet Connection 29
9.2.4 Accessing Dixen 29
9.3 Visual Documentation/ Diagrams 29
9.3.1 UI Screenshots: 30
9.3.2 Entity-Relationship (ER) Diagram: 30
9.3.3 System Architecture: 30
9.4 Test report 30
9.5 Implementation Highlights 31
9.6 External Interfaces 31
Frontend 31
Backend: 31
Database: 31
Email: 31
9.7 Technologies Used 31
Frontend 31
Backend 31
Authentication 31
Email Integration 31
Version Control 31
Communication 31

9.8 Authentication & Authorization Flow: 32
9.8.1 User Registration (via /api/authentication/register): 32
9.8.2 User Login (via /api/authentication/login): 32
9.8.3 Two-factor authentication (2FA) 32
9.9 System Design 32
9.9.1 Frontend Structure (Angular) 32
9.9.2 Backend Structure (ASP.NET Web application) 33
9.10 Database Design 33
9.10.1 Database Design Principles 33
10. Reflection & Future Work 36
11. Appendices 36
12. References 37

LIST OF TABLES

Table 1: Current System Limitations and Planned Future Enhancements 10
Table 2: Use Case UC-1 – User Registration with Email Confirmation 20
Table 3: Use Case UC-5 – Event Booking 21
Table 4:Use Case UC-7 Social Sharing 22
Table 5: Overview of All Defined Use Cases in the Dixen System 23

LIST OF FIGURES

Figure 1: Dixen event management system overview 10
Figure 2: High-Level Functional Flow of the Dixen Platform 14
Figure 3: Use Case Diagram of the Dixen Event Management System 19
Figure 4: Estimated schedule 23
Figure 5: Entity-Relationship Diagram (ERD) of the Dixen Event Management System 35

LIST OF ABBREVIATIONS USED IN DIXEN REPORT

Abbreviation	Full Form
2FA	Two-Factor Authentication
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
ER/ERD	Entity-Relationship / Entity-Relationship Diagram
JWT	JSON Web Token
MSSQL	Microsoft SQL Server
QA	Quality Assurance (implied in testing)
RBAC	Role-Based Access Control
SPA	Single-Page Application
SMTP	Simple Mail Transfer Protocol
URL	Uniform Resource Locator
UX	User Experience
PBKDF2	Password-Based Key Derivation Function 2
UI	User Interface
SDK	Software Development Kit
CLI	Command Line Interface
OTP	One-Time Password
EN	English (for multi-language support)
DA	Danish language code (for multi-language support)

1. READING GUIDE

This report offers a detailed overview of the Dixen Event Management System, encompassing its design, functionalities, and technical specifications. The sections are organized as follows:

- Section 2–3 introduce the project background, scope, system overview, and development approach.
- Section 4–5 describe the product functionality and how the functional and non-functional requirements are fulfilled.
- Section 6–8 present the project schedule, user guidance, and installation instructions.
- Section 9 explains technical architecture, database design, and security implementation.
- Section 10 provides reflection and future improvement considerations.
- The final sections include references and appendices.

2. INTRODUCTION

This Product Report presents the final implementation of the Dixen Event Management, Discovery & Ticketing System - a modern web-based platform for event discovery, management, and participation. It demonstrates how all functional and non-functional requirements defined in the Requirement Specification (RS) have been implemented, validated, and delivered in the working system.

The report details the system's implemented features, architecture, security implementation, database design, and technical specifications. It provides stakeholders with complete traceability from RS requirements to delivered functionality, including secure authentication, event workflows, role-based access, analytics dashboards, and multi-language support.

2.1. PURPOSE

The purpose of Dixen is to provide a secure, user-friendly, and scalable platform for discovering events, booking tickets, submitting event proposals, and managing event-related data. The scope of this report includes the functionality available to end users, organizers, and administrators, as well as system capabilities such as multilingual support, analytics, and security features.

2.2 SCOPE

Dixen is a full-featured event management platform accessible via a modern web browser. The platform provides the following capabilities:

- Secure user registration and authentication mechanisms.
- Event proposal submission, approvals workflows.
- Role-based access for different user types.
- Event categorization and venue management.
- Dashboard for Admin, User and Host.
- Social media sharing integration promotes events.
- Multi-language support, allowing users to switch languages dynamically.

Out of scope:

- payment processing and push notifications

2.3 STAKEHOLDERS

Guest User: Visitors who can browse, search and view event details without registration.

End Users: Individuals seeking to discover and attend events.

Organizers: Businesses or individuals creating and managing events.

Administrators: Platform managers responsible for approvals, user management, and overall system health.

Development Team: Engineers and designers building the system (Dikshya Singh Shah).

Support Team: Staff handling user inquiries and technical issues.

2.4 JUSTIFICATION FOR A ONE-PERSON PROJECT

Permission for a single-person project was granted for this final exam. I initially found a group in an earlier session, but I was not allowed to start or finish the project ahead of schedule. In the current session, I was unable to find a suitable group as most students preferred to work alone or had already formed groups.

As a result, I chose to complete the project independently. Working alone allowed me to take full responsibility for all aspects of the project, including planning, development, testing, and reporting. This ensured consistency and high quality in the final product while demonstrating

my ability to lead and manage a project independently- an important skill in both academic and professional contexts.

2.5 DEVIATIONS FROM THE REQUIREMENTS SPECIFICATION

Any changes to this requirement specification must be approved by the supervisor, Flemming Sørensen. Approved changes will be documented, justified, and included in an updated version of the specification, which will be submitted along with the final project deliverables.

3. GENERAL DESCRIPTION

3.1 SYSTEM DESCRIPTION

Dixen is a web-based platform aimed at end-users, organizers, and administrators. It allows users to discover events, register for them, and interact with organizers. Admin can create, edit, and categorize events, manage performers and venues, and submit approve or reject proposal and manage user roles, and ensure system health, organizer can send request to create event.

Key points supported by your Dixen report:

End-user focus: Users can browse and register for events without requiring technical expertise.

Platform accessibility: The system is web-based and responsive, usable on desktops, tablet, and mobile browsers (*see Hardware Description in your report*).

Data input and storage: Users and organizers input event-related data (title, description, date, venue, categories), which is stored in a MSSQL database (*see Product report -Database Design).*

Search functionality**:** Users can search events by category, date, or location, *(see Event Discovery in Product Functionality*).

3.2 SYSTEM WORLDVIEW

The Dixen system worldview presents the platform as a hub connecting users, organizers, and administrators within the event management ecosystem.

Users can discover and book events, organizers can create and manage event proposals, and administrators oversee the system and approve events. The platform interacts with a database for storing events and user data, an email service for account verification and 2FA, and social media for event sharing.

This ecosystem highlights Dixen as a central, interactive system that facilitates smooth communication, secure transactions, and efficient event management see Dixen Use case Diagram.

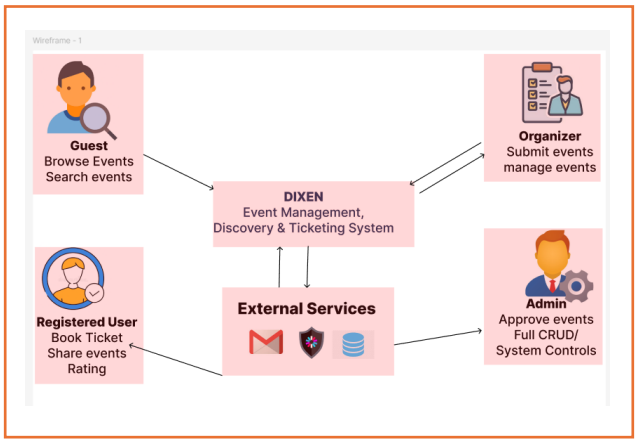


Figure 1: Dixen event management system overview (https://github.com/Dikshya-2/Dixen/wiki)

3.3 SYSTEM LIMITATIONS

Feature	Status	Planned Future Enhancement
Payment processing	Not implemented	Integrate Stripe/PayPal for ticket payments
Push notifications	Not implemented	Real-time notifications for event updates
Calendar Integration	Not implemented	Google Calendar/iCal sync for registered events
Advanced Analytics for organizers	Basic Dashboard	Detailed attendances and engagement reports
Multi-language support	Implemented(EN/DA)	Add more languages for global users

Table 1: Current System Limitations and Planned Future Enhancements

3.4 FUTURE ENHANCEMENTS

Future extensions could include:

- Payment integration to allow ticket purchases directly through the platform.
- Push notifications for upcoming events or reminders.
- Advanced analytics for organizers to track attendance and engagement.
- Calendar integration for users to add events to personal calendars.

3.5 USER PROFILES

Dixen supports several user roles, each with different permissions (*see Role-Based Access Control*):

Guest (Anonymous User)**:** Can browse events; favorites stored locally only.

Registered User (Attendee): Can browse, register, submit event proposals, and share events.

Host (Organizer): can submit event proposal for admin approval.

Admin: Full control over events, users, and system operations.

System Admin (Conceptual): Ensures system uptime and backend maintenance (not implemented in current version).

3.6 DEVELOPMENT PROCESS REQUIREMENTS

As the project was completed individually, most work was carried out from home, with occasional days at TEC, Guldblommevej 5, Ballerup. A kanban board was implemented using Github projects to manage tasks and keep an overview of priorities *see appendix 8*.

Development started with the backend, which was completed approximately 70%, before shifting focus to frontend development. Work on the report was done simultaneously, as most of the priority requirements were completed early. Daily task tracking, progress logs, and reflections on completed work are documented in appendix 5 (Logbook), supporting this incremental development process (*see Appendix 7 workplan*).

3.7 DOCUMENTATION

Documentation produced includes:

- System architecture and worldview diagrams
- Use Case descriptions and diagrams
- ER diagrams and class diagrams
- Mockup of the UI (see Appendix- 2)
- Technical implementation on GitHub

3.8 SOFTWARE TESTING

Testing was conducted throughout development:

Use Case Testing: Validates functionality as described in Use Cases.

API Testing: Conducted via Swagger to verify GET, POST, PUT, DELETE endpoints.

UI Functional Testing: Ensures frontend screens perform expected actions.

Unit testing: Implemented for backend using xUnit, Moq, and In-Memory databases (*see appendix-7 Test Report*).

4. PRODUCT DESCRIPTION

Dixen is a web-based event management, discovery, and ticketing platform designed for individuals and organizers to easily find, create, and manage events. The platform is accessible via modern web browsers on desktops, tablets, and mobile devices, offering a responsive, user-friendly experience.

4.1 PRODUCT FUNCTIONALITY

When users access Dixen, they can:

Discover Events: Browse and filter upcoming events by category, location, or date. Each event displays key details such as title, description, performers, venue, and date. Clicking on an event shows full details.

Register and Manage Attendance: Users can register for events and view their booking history. And book events and buy tickets.

Submit Event Proposals: Organizers can create, edit, and submit event proposals for approval. Each event can include performers, venues, schedules, and category assignments.

Social Sharing: Users and organizers can share events on social media platforms like Facebook, Twitter, and WhatsApp.

Analytics and Management: Organizers and admins can access dashboards to track attendance, registrations, and engagement, as well as manage events, performers, and venues.

Security and Access Control: The system supports secure email-based registration, two-factor authentication (2FA), and role-based access for Users, Organizers, and Admins.

USER EXPERIENCE

Dixen is designed for simplicity and accessibility:

- Users can explore events without registering; however, an account is required to book events and view booking history.
- Organizers have a dedicated interface to manage events efficiently.
- Admins oversee the platform and can create, view, edit, and delete all events and users, as well as approve or reject event proposals.
- The interface is mobile-first, responsive, and supports multiple languages for a global audience.

TECHNICAL PERSPECTIVE

Dixen is a single-page web application (SPA) using:

Frontend: Angular with Tailwind CSS for responsive UI and dynamic interactivity.

Backend: ASP.NET Core Web API with Entity Framework Core and Identity, secured via JWT-based authentication and 2FA.

Database: MSSQL stores users, events, tickets, categories, and other entities securely.

In essence, Dixen provides a complete end-to-end event management solution, making it easy for users to discover events, and for organizers to create, manage, and promote their events efficiently and securely.

5. REQUIREMENT SPECIFICATION

This section provides an overview of the Requirement Specification (RS) that formed the foundation for the design and development of the Dixen Event Management, Discovery & Ticketing System. The RS defined both functional and non-functional requirements and served as a contractual and technical reference throughout the development process.

All high-priority requirements defined in the RS have been implemented in the final system. Medium- and low-priority requirements were evaluated based on project scope and time constraints and are documented under future enhancements where applicable. This product has been developed as part of the final project (svendeprøve), H6.

5.1 SYSTEM OF FUNCTIONAL REQUIREMENTS

The below figure illustrates the high-level functional flow of the Dixen platform, covering the interactions of Guests, Registered Users, Organizers, and Administrators. This flowchart summarizes the main operations and decision points, linking them to the functional requirements described in Sections 5.2.1–5.2.9. More detailed flowcharts covering specific processes such as registration, email verification, login with 2FA, and code generation are provided in the Requirements Specification and for reference.

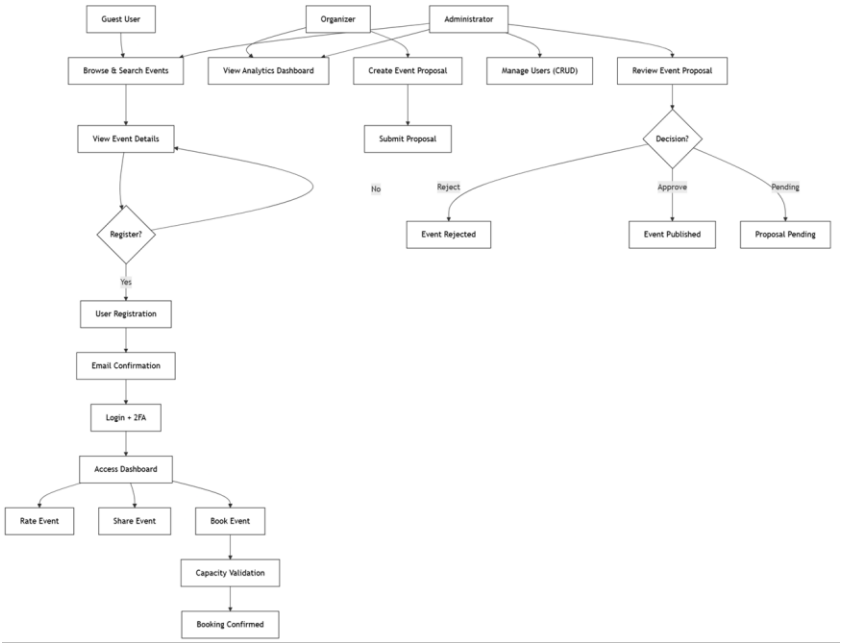


Figure 2: High-Level Functional Flow of the Dixen Platform (https://github.com/Dikshya-2/Dixen/wiki)

DESCRIPTION OF FLOW:

Guest & Registered User Flow:

Guests can browse and search for events. If they choose to register, the system performs user registration, email confirmation, and login with 2FA. Once logged in, the user can access the dashboard, book events, and view booking confirmation after capacity validation. Registered users have additional options to share events on social media and rate them.

Organizer Flow:

Organizers can create and submit event proposals. Submitted proposals are reviewed by an Administrator, who can approve, reject, or leave them pending.

Administrator Flow:

Administrators have full CRUD access to manage users and events, as well as access to analytics dashboards. Organizers also have access to analytics to review event statistics.

5.2 FULFILMENT OF FUNCTIONAL REQUIREMENTS

The Dixen platform fulfills the core functional requirements defined in the Requirement Specification. The implemented functionality supports all primary user roles: Guests, Registered Users, Organizers, and Administrators.

5.2.1 USER REGISTRATION AND AUTHENTICATION

The system provides secure user registration using email and password. Email confirmation is mandatory before account activation. Authentication is handled using JWT tokens, ensuring secure session management across the application.

5.2.2 TWO-FACTOR AUTHENTICATION (2FA)

Two-factor authentication mechanism has been implemented. Users who enable 2FA must enter a 6-digit verification code sent to their registered email address during login, providing an additional layer of security.

5.2.3 EVENT DISCOVERY AND SEARCH

Users can browse and search events by category, date, and location. Events display detailed information including title, description, venue, performers, and schedule. Guests can explore events without registration.

5.2.4 EVENT PROPOSAL AND APPROVAL WORKFLOW

Organizers can create and submit event proposals. Submitted proposals require administrator approval before becoming publicly visible. Admins can approve, reject events as defined in the RS.

5.2.5 TICKET BOOKING AND CAPACITY MANAGEMENT

Registered users can register for events and manage their booking history. The system validates event capacity to prevent overbooking and stores booking data securely.

5.2.6 ROLE-BASED ACCESS CONTROL (RBAC)

The system implements role-based access control to ensure users only access features appropriate to their role:

- Users can browse, register, and share events.
- Organizers can manage events, performers, and venues.
- Administrators have full system access, including approvals and user management.

5.2.7 SOCIAL MEDIA SHARING

Events can be shared on supported social media platforms, increasing visibility and engagement.

5.2.8 ANALYTICS DASHBOARD

Organizers and administrators have access to dashboards displaying event statistics, registrations, and engagement metrics.

5.2.9 GOOGLE MAPS NAVIGATION LINK

The system provides a direct Google Maps navigation link on each event page, allowing user to quickly locate. The link is automatically generated using the event's name, address, and city to ensure accurate location results.

5.3 FULFILMENT OF NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements defined in the Requirement Specification have been addressed as follows:

5.3.1 PERFORMANCE

The system is designed to support concurrent users with responsive API performance. Under normal conditions, API responses remain within acceptable response times.

5.3.2 SECURITY

Passwords are stored securely using ASP.NET Identity's built-in hashing algorithm, PBKDF2 (Password-Based Key Derivation Function 2).

Each password is salted and hashed, making it computationally infeasible for attackers to recover plain-text passwords, even if the database is compromised. JWT tokens are used for stateless authentication and secure session management. HTTPS is enforced for all communication, and 2FA adds an extra layer of security.

5.3.3 USABILITY

The user interface is mobile-first and responsive, ensuring usability across desktops, tablet, and mobile devices. The interface supports multiple languages, allowing users to switch languages dynamically.

5.3.4 RELIABILITY AND MAINTAINABILITY

The system follows a modular architecture using Angular and ASP.NET Core, supporting maintainability and future scalability. Database integrity and structured logging support stable operation.

5.4 CONSTRAINTS AND DEVIATIONS

Certain features defined as optional or future enhancements in the Requirement Specification were not implemented due to scope and time limitations:

- Payment processing
- Push notifications
- Calendar integration

These features are documented under Future Development and Integrations and do not affect the fulfillment of core system requirements.

5.5 TRACEABILITY TO REQUIREMENT SPECIFICATION

The Requirement Specification served as a continuous reference throughout development. Each implemented feature can be traced back to one or more requirements defined in the RS. Detailed requirement descriptions, use cases, diagrams, and acceptance criteria are documented in the Requirement Specification and referenced where necessary.

5.6 DATA REQUIREMENTS

The system must store structured data to support authentication, event management, booking, and analytics. The following logical data entities are required:

- Users (including roles and authentication data)
- Organizers
- Events
- Categories
- Venues and Halls
- Bookings and Tickets
- Event Submissions
- Social Shares
- Event Reviews

Each entity supports one or more functional requirements such as user registration, event creation, role-based access control, capacity-controlled booking, and administrative approval workflows.

The detailed relational schema, entity relationships, constraints, and normalization strategy are described in Section 9.10 (Database Design)

5.7 USE CASE DIAGRAM AND DESCRIPTIONS

This section describes three key implemented use cases out of a total of eight. These use cases represent the core functionalities of the system across different user roles: Users, Organizers, and Administrators. Each case includes the actor, use case ID, description, preconditions, inputs, outputs, expected results, main flow, and alternative flows.

The complete set of use case descriptions can be found in *Appendix 1 section: Case Description*).

5.7.1 DIXEN USE CASE DIAGRAM

The diagram illustrates how individual users interact with the Dixen platform, showing the complete system functionality across the three main actors: Guest, Registered User, and Administrator. For further details, *refer to the System worldview , User profile, RBAC and Appendix 1 section 12.2 Use case Diagram.*

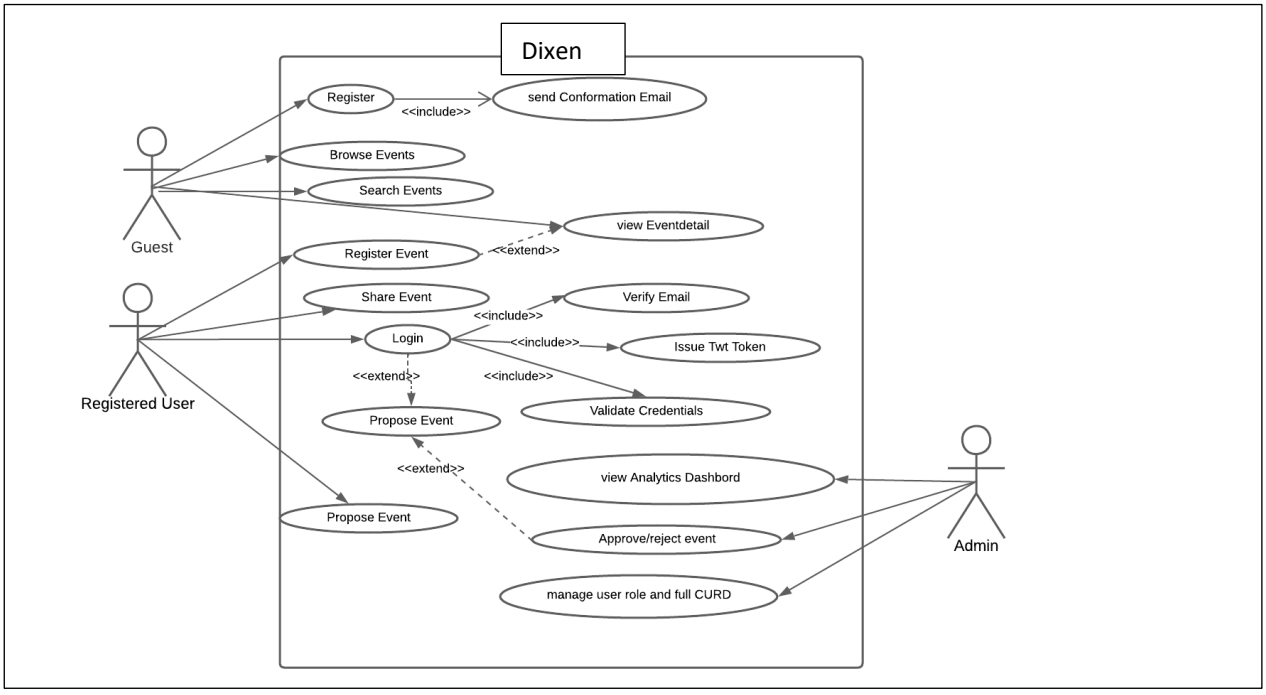


Figure 3: Use Case Diagram of the Dixen Event Management System
(https://github.com/Dikshya-2/Dixen/wiki)

5.7.2 IMPLEMENTATION USE CASE (3 OF 8 DETAILED BELOW)

5.7.2.1 USER REGISTRATION WITH EMAIL CONFIRMATION

Actor: User	**Id:** UC- 1
Description: The user registers for the application, and the system sends an email confirmation link.	
Precondition: The actor has a valid email address.	
Input: Full Name, Email, Password, Age and Gender	
Output: ➢ User account created in the database. ➢ Confirmation email sent to the provided email address. ➢ Users receive a message to check their email for confirmation.	
Expected Result: The user account is created in the system in an unconfirmed state, and a confirmation email with a link is sent to the user's email address.	
Main Flow: ➢ The actor enters the required registration information (e.g., name, email, password). ➢ The system validates the information and creates the user's account in the database in an unconfirmed state. ➢ The system sends an email with a confirmation link to the user's email address. ➢ The system informs the user to check their email for the confirmation link.	
Alternative Flows: ➢ The user enters invalid registration information (System shows validation errors). ➢ The confirmation email fails to send (system informs the user to try again).	

Table 2: Use Case UC-1 – User Registration with Email Confirmation

5.7.2.2 EVENT BOOKING

Actor: User	**ID**: UC-5
Description: The registered user books a ticket for a selected event. The system assigns the user to a hall with available capacity and sends a booking confirmation email.	
Precondition: ➢ The user is logged in to the application. ➢ Event must exist and have at least one hall with available capacity.	
Input: EventId	
Output: ➢ Booking records are created in the database. ➢ Hall Assigned ➢ Confirmation email sent to the user.	
Expected Result: The user successfully books a ticket for the selected event and receives a confirmation email with booking details.	
Main Flow: ➢ The user selects an event to book. ➢ The system verifies the user's identity from the authentication token. ➢ The system retrieves event details and hall information. ➢ The system checks hall capacities. ➢ The system assigns the user to the first available hall. ➢ The system creates a booking record. ➢ The system sends a confirmation email to the user. ➢ The user sees a booking confirmation message.	
Alternative Flows: ➢ No Available Seats: The system informs the user that the event is fully booked. ➢ User Not Authenticated: The system returns an unauthorized error. ➢ Event Not Found: The system returns an error indicating the event does not exist.	

Table 3: Use Case UC-5 – Event Booking

5.7.2.3 SOCIAL SHARING (SHARE EVENT ON PLATFORM LIKE TWITTER, FACEBOOK)

Actor: User	**ID:** UC-7
Description: The user can share an event on social media platforms like Twitter and Facebook.	
Precondition: The actor is logged in to the application.	
Input: Select event, Chosen social media platform	
Output: ➢ Event details are shared on the selected platform. ➢ SocialShares record created in the database. ➢ A confirmation message is displayed to the user.	
Expected Result: The actor can share the event on the selected social media platform.	
Main Flow: ➢ The user clicks "Share Event". ➢ The system displays options to share the event on social platforms such as Twitter, Facebook, or others. ➢ Select a platform (e.g., Twitter). ➢ If the user is not logged in to the selected social media platform, the system prompts the user to log in before proceeding. ➢ The user logs in to the selected social media platform (if necessary). ➢ The system shares the event details on the chosen platform, including the event title, description, and a link. ➢ The system logs the share in SocialShares.	
Alternative Flows: ➢ Sharing fails due to a network error. ➢ Unsupported platform is selected.	

Table 4:Use Case UC-7 Social Sharing

The table below provides an overview of all eight defined use cases in the Dixen system, including the associated actors and a summary of each functionality. Detail description of all eight use case are available in *appendix 1* Requirement specification

Use Case Id	Actor	Summary
UC-2	User	Confirm email address
UC-3	User	Login with 2FA (QR Code and Google Authentication)
UC-4	Admin	Event categorization (events can belong to multiple categories)
UC-6	User	Submit event proposal (event-submission)
UC-8	Admin	Event management (create, edit, and delete events

Table 5: Overview of All Defined Use Cases in the Dixen System

6. ESTIMATED SCHEDULE

5-week Agile timeline delivered on schedule (19-Jan to 18-Feb 2026)

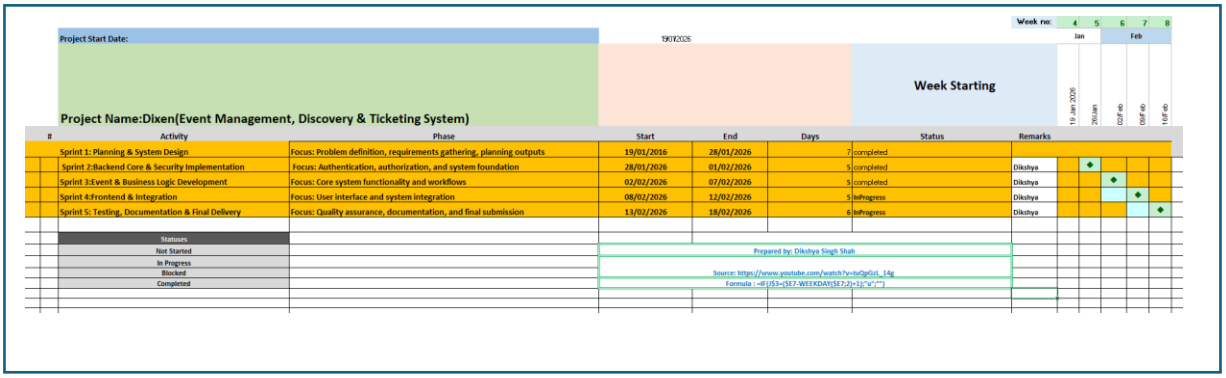


Figure 4: Estimated schedule

Progress updates and task completion records -Appendix 5, Detailed work plan *appendix-4* and process methodology - *process report section 6.5*

7. USER MANUAL/ GETTING STARTED

7.1 REGISTERING A NEW ACCOUNT

- Open the registration page.
- Enter your Full Name, Email, and Password.
- Click "Register".
- Check your email inbox for a confirmation link.
- Click the link to activate your account.

7.2 LOGGING IN

LOGIN WITH TWO-FACTOR AUTHENTICATION (2FA)

- Go to the login page.
- Enter your registered Email and Password.
- If 2FA is enabled, scan the QR code with your Google Authenticator app.
- Enter the generated code.
- Click "Submit" to log in.

7.3 BROWSING AND REGISTERING FOR EVENTS

BROWSING EVENTS

- After logging in, navigate to the Events page.
- Browse through the list or use search filters.
- Click on an event for details.

7.4 REGISTERING FOR AN EVENT

- Select the event you want to attend.
- Choose the desired hall/location.
- Confirm your registration.
- Receive a confirmation message/email.
- Note: If the hall is full or event has passed, you will be notified.

7.5 SUBMITTING EVENT PROPOSALS (FOR USERS)

- Log in to your account.
- Navigate to "Submit Proposal".
- Fill out the form with topic and description.
- Submit your proposal for admin review.
- Wait for confirmation or feedback.

7.6 MANAGING EVENTS (FOR ORGANIZERS)

CREATING EVENTS

- Log in as Organizer.
- Go to "Create Event".
- Enter event details: title, description, date/time.
- Select categories and assign halls/performers.
- Submit the event for admin approval.

7.7 EDITING OR DELETING EVENTS

- Go to "Events".
- Select the event you want to edit or delete.
- Make changes and save or confirm deletion.

7.8 SHARING EVENTS ON SOCIAL MEDIA

- On any event page, click "Share".
- Choose the social platform (Twitter, Facebook, etc.).
- Log in to the platform if prompted.
- Confirm to share the event details.

7.9 ADMIN CONTROLS (FOR ADMINS)

- Log in with Admin credentials.
- Manage user roles and permissions.
- Approve or reject submitted event proposals.
- Edit or delete any event in the system.

7.10 TROUBLESHOOTING & SUPPORT

- Common issues and solutions.
- How to reset your password.
- Contact info for technical support.

8. INSTALLATION AND CONFIGURATION

Dixen is a web management application that combines ASP.NET Core with Entity Framework on the backend and Angular 20.3.16 with Tailwind CSS on the front-end.

8.1 TECHNOLOGIES USED

- ASP.NET Core
- Entity Framework
- Angular 20.3.16
- Tailwind CSS v,4.1.18
- Node 22.21.0

8.2 NUGET PACKAGES

In this project, I use NuGet packages to add external libraries and frameworks that provide essential functionality, helping us build the application more efficiently without reinventing the wheel.

Below is a summary of the key NuGet packages included and their purpose:

- Microsoft.AspNetCore.Authentication.JwtBearer (v8.0.1)
- Microsoft.AspNetCore.Identity.EntityFrameworkCore (v8.0.20)
- Microsoft.EntityFrameworkCore (v8.0.20) and related packages
- Moq v4.20.72
- QRCoder (v1.6.0)
- Swashbuckle.AspNetCore (v6.6.2)
- Xunit 2.5.3

8.3 DATABASE SERVER

SQL Server 2019

8.4 GETTING STARTED

To run Dixen locally, follow these steps:

8.4.1 INSTALL PREREQUISITES

- .NET Core SDK (version 8)
- Node.js (version 22.21.0)
- Angular CLI: 20.3.15
- Package Manager: npm 10.9.4

8.4.2 COMMANDS

These commands install the specified packages and ensure that they can be used in web application -Dixen (*See appendix-3 for more detail*)

- ng new DixenUI
- ng serve
- ng generate component component-name
- ng g service service GenericService (name)
- npm install

9. TECHNICAL ARCHITECTURE AND IMPLEMENTATION

This section provides an overview of the technical structure of the Dixen web application, including its core components, technologies used, and system behavior. The aim is to give a high-level understanding of how the product functions from a technical perspective.

For in-depth technical details, architecture diagrams, technology justifications, and implementation processes, please refer to the Process Report, particularly the section titled "*Architecture and Structure, Method & Technologies Choice and Use Case Description in Appendix1*.

9.1 SYSTEM/TECHNICAL ARCHITECTURE OVERVIEW

9.1.1 FRONTEND

Framework: Angular 20.3.16, TypeScript

UI: Tailwind CSS for responsive design

Key Components: Registration/Login, Event List, Category Filter, Organizer Dashboard

9.1.2 BACKEND

Framework: ASP.NET Core 8 Web API

Database: Microsoft SQL Server, accessed via Entity Framework Core (code-first)

Authentication: ASP.NET Core Identity, JWT tokens, 2FA, email confirmation

APIs: RESTful endpoints for user, event, and category management

9.1.3 SECURITY

- JWT-based authentication for stateless, secure sessions
- 2FA with QR code generation and OTP verification
- Password hashing using ASP.Net Core Identity (PBKDF2 with salted hashes)
- Role-based access control for different user types
- Repository Pattern: EF Core services isolate data access
- Angular Route Guards: Protect admin routes (CanActivateFn)
- secure API communication, CORS policy.

9.2 HARDWARE DESCRIPTION

To access and use the Dixen web event management system, users must meet the following hardware and connectivity requirements:

9.2.1 DEVICE WITH INTERNET ACCESS

Users need a device capable of connecting to the Internet. This can include:

- Desktop or laptop computers
- Smartphones
- Tablets

9.2.2 WEB BROWSER

Users must have a modern, up-to-date web browser installed on their device. Supported browsers include, but are not limited to:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

9.2.3 INTERNET CONNECTION

A stable internet connection is essential for loading and interacting with the Dixen platform. This connection can be:

- Wired (Ethernet)
- Wireless (Wi-Fi)
- Mobile data (3G, 4G, 5G)

9.2.4 ACCESSING DIXEN

Once the above requirements are met, users can access Dixen by opening their preferred web browser and entering the Dixen URL in the address bar. This will direct them to the Dixen login page, where they can securely access event management features.

9.3 VISUAL DOCUMENTATION/ DIAGRAMS

This section provides an overview of the visual documentation and diagrams related to the Dixen web event management system. For detailed visuals, please refer to the specified appendices and sections.

9.3.1 UI SCREENSHOTS:

Screenshots of the Dixen web application are provided for every implemented function, demonstrating fulfillment of the functional requirements defined in Section 5.2. Each screenshot corresponds to a specific feature, including user registration, 2FA login, event discovery and filtering, event proposal submission and approval, ticket booking, social sharing, analytics dashboards, and Google Maps navigation. Detailed screenshots are included in Appendix 6.

9.3.2 ENTITY-RELATIONSHIP (ER) DIAGRAM:

The ER diagram, which outlines the data relationships within the system, can be found in Section 9.10.7.

9.3.3 SYSTEM ARCHITECTURE:

For a detailed description of the system's architecture and structure, please refer to the Process Report under the section titled "Architecture and Structure.

9.4 TEST REPORT

To ensure the reliability and correctness of the Dixen Event Management System, I implemented unit testing using the xUnit framework in ASP.NET Core 8. Unit tests were designed to validate individual methods within the backend application, confirming that each function behaves as expected under various conditions.

I chose xUnit because it is a free, open-source unit testing framework for .NET development, offering a rich set of features for writing clean and effective test cases. In my tests, I used the [Fact] attribute to denote test methods that follow the Arrange-Act-Assert pattern, promoting clarity and consistency in our testing approach.

To isolate the units under test and eliminate external dependencies, I incorporated the Moq library. Moq allows mock implementations of interfaces and dependencies, enabling focused testing of business logic without requiring real database or service interactions. This approach ensured that each test remains independent and deterministic.

Additionally, I employed in-memory databases during testing of the generic repository pattern, enabling realistic simulation of data operations without needing a physical database. This provided a lightweight and efficient environment to verify data access functionality.

The combination of xUnit, Moq, and in-memory data stores facilitated a robust testing strategy, allowing early detection of issues and contributing significantly to the stability and maintainability of the application (*see appendix -7 for more details*).

9.5 IMPLEMENTATION HIGHLIGHTS

All core features delivered successfully:

- Many-to-Many Event Categorization: Junction table enables flexible filtering.
- Dynamic Filtering: Angular real time category updates
- Secure authentication (JWT and 2FA)
- Event CRUD, submission event request and approval workflow
- Real-time analytics dashboard
- Multi-language support (EN/DA)

Detailed development process in Process Report Section.

9.6 EXTERNAL INTERFACES

FRONTEND: Angular using REST API calls to backend

BACKEND: ASP.NET Core Web API with Entity Framework

DATABASE: MSSQL for all persistent storage

EMAIL: SMTP server for sending confirmation and 2FA codes

9.7 TECHNOLOGIES USED

FRONTEND: Angular with TypeScript, HTML, CSS, Bootstrap.

BACKEND: ASP.NET Core Web API (.NET 8), Entity Framework Core (EF Core), MSSQL.

AUTHENTICATION: ASP.NET Identity with JWT-based session management.

EMAIL INTEGRATION: SMTP for sending confirmation emails and 2FA codes.

VERSION CONTROL: GitHub, Email.

COMMUNICATION: Email (for sending project-related updates, reports, etc.)

9.8 AUTHENTICATION & AUTHORIZATION FLOW:

9.8.1 USER REGISTRATION (VIA /API/AUTHENTICATION/REGISTER):

- The user provides their email, password, and full name.
- The system validates the email format and checks for duplicates.
- Upon successful registration, an email confirmation token is generated and sent to the user's email address.
- The user must confirm their email to activate the account.
- Email Confirmation (via /API/authentication/confirm-email):
- The user clicks the confirmation link in the email.
- The system decodes and validates the confirmation token.
- Once validated, the user's email status is marked as confirmed, and the account becomes active.

9.8.2 USER LOGIN (VIA /API/AUTHENTICATION/LOGIN):

- The user provides their email and password to log in.
- The system verifies the credentials and checks if the email is confirmed.
- If Two-Factor Authentication (2FA) is enabled, a one-time code is sent to the user's email.
- The user must enter the 2FA code to complete the login process.
- Upon successful login, a JWT token is issued for secure session management.

9.8.3 TWO-FACTOR AUTHENTICATION (2FA)

- Dixen implements email-based 2FA to enhance account security.
- After successful credential verification, a 6-digit code is sent to the user's email.
- This second layer of verification ensures that only the email owner can access the account, even if login credentials are compromised.

9.9 SYSTEM DESIGN

9.9.1 FRONTEND STRUCTURE (ANGULAR)

COMPONENTS: Login, Register, EventList, EventDetails, AdminPanel.

GENERIC SERVICES: A single reusable Angular service is implemented to handle all backend API communication across components. This generic service abstracts HTTP operations (GET,

POST, PUT, DELETE) and allows components to interact with different REST endpoints by specifying the endpoint URL and data payload dynamically.

9.9.2 BACKEND STRUCTURE (ASP.NET WEB APPLICATION)

CONTROLLERS: AuthController, EventController, EventSubmittionController.

SERVICES: JWTService, EmailService, 2FAService.

IDENTITY: ASP.NET Core Identity for user management.

DATA ACCESS: Generic Repository, ApplicationDbContext.

Data Transfer Objects or Mappers, for clean architecture.

9.10 DATABASE DESIGN

The Dixen Event Management System requires a structured relational database to manage users, events, bookings, venues, reviews, categories, and related entities. To fulfill these requirements, a normalized relational database schema was designed and implemented using Entity Framework Core (Code-First approach) together with ASP.NET Core Identity for authentication and authorization.

9.10.1 DATABASE DESIGN PRINCIPLES

The database schema was designed according to the following principles:

9.10.2 NORMALIZATION (3NF)

The schema follows Third Normal Form (3NF):

- No repeating groups
- No partial dependencies
- No transitive dependencies
- Separation of concerns between entities

For example:

- Venues and Halls are separated to allow multiple halls per venue.
- Bookings and Tickets are separated to support multiple ticket types per booking.
- EventCategories is used as a junction table to support many-to-many relationships.

9.10.3 IDENTITY INTEGRATION

User authentication and authorization are handled using ASP.NET Identity.

The AspNetUsers table stores:

- User credentials
- Security information
- Email confirmation
- 2FA configuration
- Role assignment

This ensures:

- Secure password hashing
- Token-based authentication (JWT)
- Role-based access control (RBAC)

9.10.4 RELATIONSHIP DESIGN

The system includes the following key relationships:

- One Organizer - Many Events
- One Event - Many Bookings
- One Booking - Many Tickets
- One Venue - Many Halls
- One Event - Many Performers
- One Event - Many Reviews
- Many Events - Many Categories (via EventCategories)

All relationships are enforced using foreign-key constraints to ensure referential integrity.

9.10.5 SOFT DELETION STRATEGY

Several entities include an IsDeleted boolean field. Instead of permanently removing records from the database, an IsDeleted boolean flag marks them as deleted.

Example: When a booking is deleted, the system sets IsDeleted = true. The record remains in the database, preserving historical data for analytics or auditing purposes, but it is hidden from standard user views.

9.10.6 INTEGRITY AND CONSTRAINTS

The database enforces:

- Primary keys (GUID-based identifiers)
- Foreign key constraints
- Required fields (e.g., Title, Email)
- Unique constraints (Email address)
- Capacity validation at application level

This ensures data consistency and prevents orphaned records.

9.10.7 ENTITY-RELATIONSHIP DIAGRAM (ERD)

The Entity-Relationship Diagram (ERD) presented below visually represents the complete database structure of the system. It illustrates all entities, their attributes, primary and foreign key relationships, and the cardinality between tables. The diagram demonstrates how the database supports the system's functional requirements.

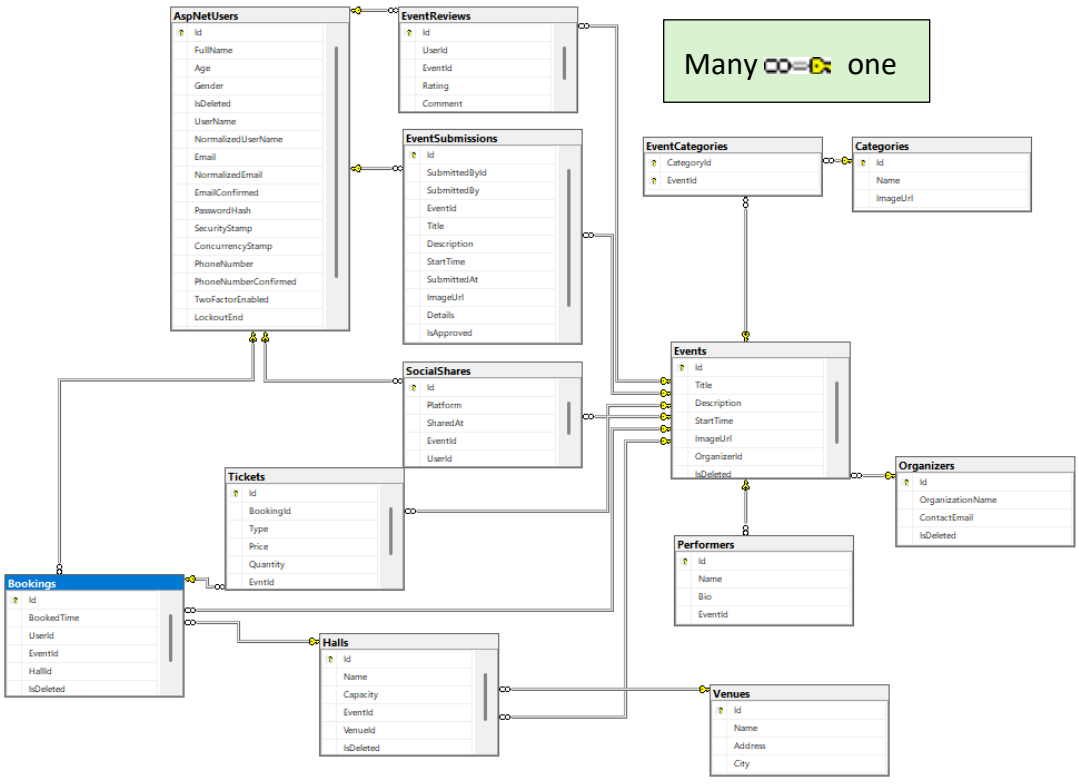


Figure 5: Entity-Relationship Diagram (ERD) of the Dixen Event Management System (https://github.com/Dikshya-2/Dixen/wiki)

The diagram demonstrates how the database supports:

- User registration and authentication
- Event management
- Capacity-controlled booking
- Role-based authorization
- Analytics and reporting

10. REFLECTION & FUTURE WORK

Developing Dixen was both challenging and rewarding. One significant challenge was managing the workload as a single-person project. To meet deadlines, I dedicated extended hours- including evenings and weekends-to ensure all functional and non-functional requirements were implemented and tested. This experience taught me the importance of time management, prioritization, and focus in full-stack development projects.

From a technical perspective, I strengthened my skills in frontend and backend development, including Angular, ASP.NET Core, JWT authentication, and two-factor security. I also gained hands-on experience in database design, role-based access control, and implementing robust testing strategies using xUnit and Moq.

Working independently emphasized the value of planning, documenting progress, and systematically tracking requirements. For future projects, I will focus on better task estimation, balancing workload, and incremental delivery, to maintain productivity without excessive hours while ensuring high-quality results. Detailed activity logs and time management reflections are available in appendix 5.

11. APPENDICES

Appendix 1: Requirements Specification

Appendix 2: Mockup

Appendix:3 Command and installation

Appendix 4: Workplan

Appendix 5: Logbook

Appendix 6: UI Screenshots

Appendix 7: UnitTest

Appendix 8: Source Code (GitHub link)

12. REFERENCES

1. ASP.NET Core 8 Documentation: https://learn.microsoft.com/en-us/aspnet/core/
2. Angular Documentation: https://angular.io/docs
3. SMTP: Simple Mail Transfer Protocol: https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
4. OWASP Password Storage Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
5. QRCoder. (2023). GitHub Repository. https://github.com/codebude/QRCoder
6. Tailwind CSS: https://tailwindcss.com/docs/installation/using-vite
7. xUnit.net: https://xunit.net/?tabs=cs
8. Jwt.io: https://www.jwt.io/introduction#what-is-json-web-token
9. Microsoft. (2023) ASP.NET Core Identity: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-10.0&tabs=visual-studio
10. Node.js: https://nodejs.org/en/docs/
11. GitHub: https://docs.github.com/
12. Angular CLI: https://angular.io/cli
13. ngx-translate: https://v17.angular.io/guide/i18n-overview
14. AGChart: https://www.ag-grid.com/charts/angular/quick-start/