



Digital Egypt Pioneers Initiative
Full stack .Net Web Developer - CAI2_SWD5_S7

Event planning and ticketing system

Aya Khaled	21092131
Alexandra Raees	21082208

Contents

Project planning and Management	3
Project Overview	3
Objectives	3
Project scope	4
Requirements gathering	6
Stakeholder Analysis	6
User stories and use case	7
Functional requirements	8
Non-Functional requirements	9
System Analysis and design	11
Problem definition	11
System actors and interactions	11
System Capabilities (What the system can do)	11
System Constraints (Limitations)	12
Software Architecture	12
Technologies used	13
Database Design (SQL Server)	14
Schema Diagram	15
Implementation	16
Website interface	16
Conclusion	19
Future Work	19

Project planning and Management

Project Overview

The Event Management System is a web-based application designed to facilitate the creation, organization, and management of events. Built using the ASP.NET Identity framework, it incorporates user authentication, role-based access control, and event booking functionalities.

The system supports multiple stakeholders, including users (who can browse and book events), admins (who manage roles, users, and event categories), and optionally event managers (who monitor event statuses and logistics).

Each event belongs to a specific category, defined by parameters like name, pricing, and capacity. The application maintains secure access using claims and roles, enabling a scalable and maintainable solution for managing various event types and users.

Objectives

- **Efficient Event Management:** Enable organizers to seamlessly create, manage, and track events, ensuring a streamlined planning process.
- **User-Friendly Attendee Registration:** Allow attendees to browse, register, and purchase tickets effortlessly, enhancing user experience.
- **Vendor Management System:** Facilitate vendor applications, approvals, and service management, ensuring smooth collaboration with event organizers.

- **Secure & Scalable Payment Integration:** Ensure fast, secure, and seamless transactions using Stripe & PayPal, with support for high-traffic demand.
- **Automated Ticketing System with QR Code Validation:** Generate QR-coded digital tickets for secure, automated verification and access control.
- **Real-Time Event Analytics & Reporting:** Provide live insights into ticket sales, revenue, and attendee engagement for data-driven decision-making.
- **High Performance & Scalability:** Optimize the system to handle large-scale events while ensuring fast response times and reliability.

Project scope

- **In-Scope Features:**

The Event Management System will include the following key features and modules:

- 1. User Management:**

User registration, login, and profile management.

Role-based access (User, Admin, Vendor, Event Manager).

Two-factor authentication for enhanced security.

- 2. Event Management:**

Event creation, editing, and scheduling by organizers.

Category-based event organization with pricing and capacity settings.

Event status tracking (e.g., Pending, Approved, Cancelled).

3. Attendee Features:

Event browsing and filtering.

Ticket booking and payment through Stripe and PayPal.

QR code-based ticket generation and entry validation.

4. Vendor Management:

Vendor registration and application.

Approval workflows and service assignment.

Communication channels between vendors and organizers.

5. System Administration:

Role and claims management.

Event category and pricing management.

User activity and audit logs (optional for advanced scope).

6. Performance & Scalability:

Optimized backend and database for high-volume traffic.

Scalable architecture to support large-scale events.

- **Out of Scope (Future Enhancements):**

The following items are not included in the current phase of the project:

1. Native mobile applications (iOS/Android).
2. Payment integration
3. Analytics and Reporting
4. In-app live chat or video conferencing.
5. Offline/physical kiosk ticket printing.
6. Integration with third-party event promotion tools (e.g., Eventbrite, Meetup).
7. Custom hardware solutions for venue check-in (e.g., scanners, RFID readers).

Requirements gathering

Stakeholder Analysis

1. Users

Description: Regular users who register, log in, and book events.

Responsibilities:

- Register and manage their profile.
- Browse and book events.
- View booking history.
- Use secure login features like two-factor authentication.

2. Admins

Description: System administrators who manage users, roles, event categories, and claims.

Responsibilities:

- Assign roles and permissions.
- Manage user claims.
- Create and maintain event categories.
- Monitor system access and data integrity.

3. Developers

Description: Technical team responsible for implementing and maintaining the system.

Responsibilities:

- Design and update the database.
- Ensure security and performance.
- Implement new features and fix bugs.

4. System Owners

Description: The organization or individuals who own the platform.

Responsibilities:

- Define business requirements.
- Ensure ROI (Return on Investment).
- Oversee compliance, legal, and financial aspects.

User stories and use case

1. User Stories

- **As a User...**

- **User Registration and Login:**

- As a user, I want to register and log in so that I can access the event management system.

- **Event Booking and Management:**

- As a user, I want to browse and book available events so that I can reserve a venue for my activity.

- **View User Profile and History:**

- As a user, I want to view my profile and booking history so that I can keep track of my past and upcoming events.

- **Two-Factor Authentication:**

- As a user, I want to enable two-factor authentication so that my account is more secure.

- **As an Admin...**

- **Role-Based Authorization:**

- As an admin, I want to assign roles to users so that I can control who can perform specific actions in the system.

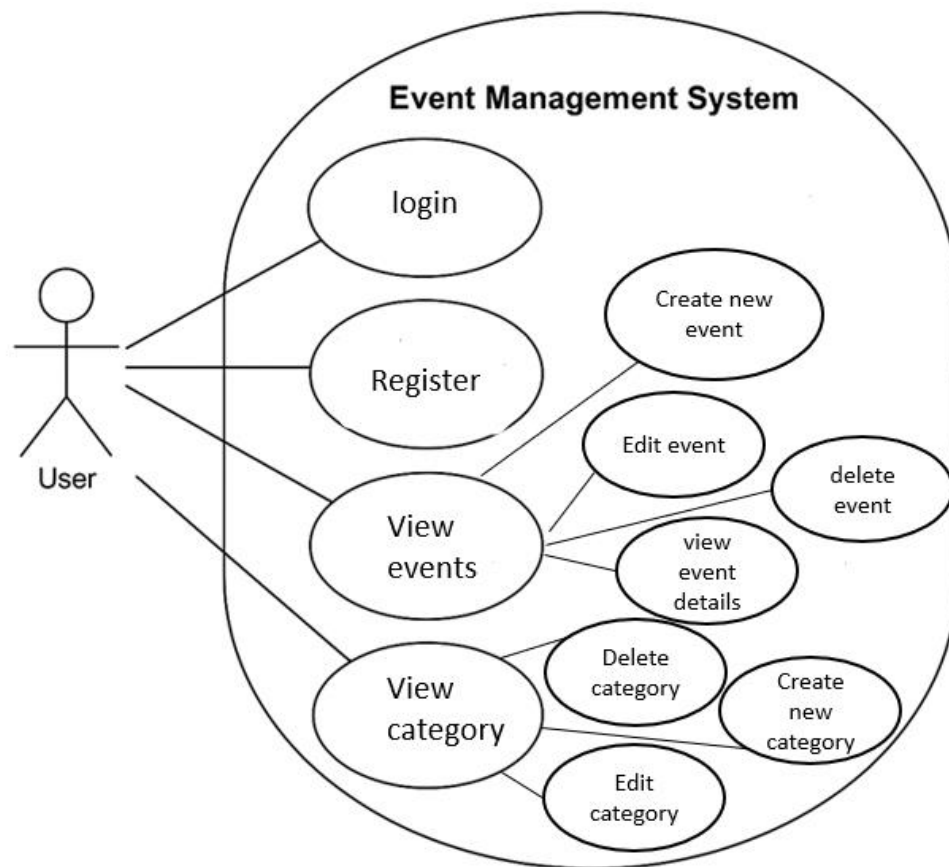
- **Event Category Management:**

- As an admin, I want to create and manage event categories so that users can book events under appropriate types with defined pricing and capacity.

- **User Claims Management:**

- As an admin, I want to manage user claims so that I can grant or restrict access to specific features based on permissions.

2. Use Case Diagram



Functional requirements

These define what the system should do

1. User Management

- The system must allow users to register, log in, and log out.
- Users must be able to view and update their profiles.
- The system must support password reset and two-factor authentication.

2. Role and Permission Management

- Admins must be able to assign roles (e.g., Admin, User, Event Manager) to users.
- Role-based access control should determine which features each role can access.
- Claims must be assignable at both the user and role level.

3. Event Category Management

- Admins must be able to add, update, delete, and activate/deactivate event categories.
- Each category must include details like name, description, hourly/daily price, and capacity.

4. Event Booking

- Users must be able to browse available event categories.
- Users must be able to create and view their event bookings.
- Event booking must capture date, venue, time, category, and user information.

5. Authentication and Security

- The system must authenticate users before granting access to protected features.
- Support for external login providers.

Non-Functional requirements

These describe **how** the system performs, including constraints and qualities

1. Performance

- The system should respond to user actions within 2 seconds under normal load.
- Database queries should be optimized for fast retrieval of bookings and categories.

2. Scalability

- The system should support an increasing number of users and bookings without performance degradation.

3. Security

- Passwords must be stored using hashing algorithms (e.g., ASP.NET Identity default).
- Data transmission should be secured using HTTPS.
- Role and claim-based authorization should restrict access to protected resources.

4. Availability

- The system should be available 99.5% of the time, excluding scheduled maintenance.

5. Maintainability

- The system should be built with a modular and clean architecture.
- Code should be documented and follow naming conventions.

6. Usability

- The user interface should be intuitive and responsive.
- Users should receive meaningful error messages and feedback.

7. Backup and Recovery

- Daily database backups should be scheduled.
- Recovery procedures should allow restoration within a 4-hour window in case of failure.

8. Compatibility

- The system should work on modern browsers.
- The backend should be compatible with .NET Core/.NET 6+ environments.

System Analysis and design

Problem definition

Managing events, ticket sales, and vendor coordination is often complex and inefficient. Many existing event management platforms lack seamless integration of attendee registration, vendor participation, and real-time analytics. Additionally, issues like fraudulent tickets, slow payment processing, and limited scalability create challenges for event organizers.

System actors and interactions

1. Users

- Can browse event categories and events.
- Can book events.
- Manage their account.

2. Event Managers / Administrators

- Create/Edit/Delete Event Categories.
- Create/Edit/Delete Events.
- View and manage bookings.
- Assign roles to users.

3. Event Participants

- Browse events.
- Book and cancel events.
- View booking history.

4. System

- Applies schema changes.
- Ensures data consistency and versioning.

System Capabilities (What the system can do)

1. User Accounts

- Users can register, log in, and manage their profile.
- Users can have roles like “Admin” or “Customer”.

2. Event Management

- Admins can add, edit, and delete events.
- Events have a title, description, date, location, and category.

3. Category Management

- Admins can manage categories like “Workshop”, “Seminar”, etc.

4. Event Booking

- Users can browse events and book them.
- Each booking is saved with user details.

5. View Bookings

- Users can see their own bookings.
- Admins can view all bookings and users.

6. Secure Access

- Admin-only pages are protected.
- Users must log in to book events.

System Constraints (Limitations)

1. Technology

- Built using ASP.NET Core and SQL Server.
- Works on web browsers only.

2. User Access

- Only admins can manage events and categories.
- Regular users can only view and book events.

3. Booking Limits

- No payment feature is included.
- No real-time updates.

4. Security

- Needs HTTPS for safe data transmission.
- User roles and permissions must be managed correctly.

Software Architecture

The system follows the MVC design pattern, commonly used in ASP.NET Core applications. It separates the application logic into:

Model: Represents data and business logic.

View: The user interface.

Controller: Handles user requests, processes data, and returns views.

1- User Interface (View Layer)

- Built using Razor pages and HTML/CSS/JavaScript.
- Allows users to:
 - Register/Login
 - Browse and book events
 - View their bookings
 - Admins: manage events & categories

2- Controllers (Controller Layer)

- Handles routing and requests:
 - AccountController – Manages user registration and login.
 - EventController – Displays events, handles bookings.
 - AdminController – Manages categories and event CRUD operations.

3- Models (Model Layer)

- Represents application data and logic:
 - User
 - Event
 - Category

Technologies used



HTML: The standard language for creating web pages using HTML strict compliance. It is used for web programming and is responsible for the structural content that makes up the web.



CSS: A Cascading stylesheet is used to show how a site would look written in HTML. This will be in every case added to the same tag as the content for styling the web pages in a proper way.



JavaScript: A programming language designed to run in browsers to create dynamic and interactive effects on the website. It resembles Java which is used for bringing up the page to the front.



SQL Server: A database management system by Microsoft used for storing and managing data with strict structure. It is used in software development and is responsible for handling and organizing the data that powers applications and websites.



.NET: A software development framework by Microsoft used to build web, desktop, and mobile applications. It supports multiple programming languages like C#, F#, and VB.NET, and provides a structured environment for building and running apps efficiently across different platforms.



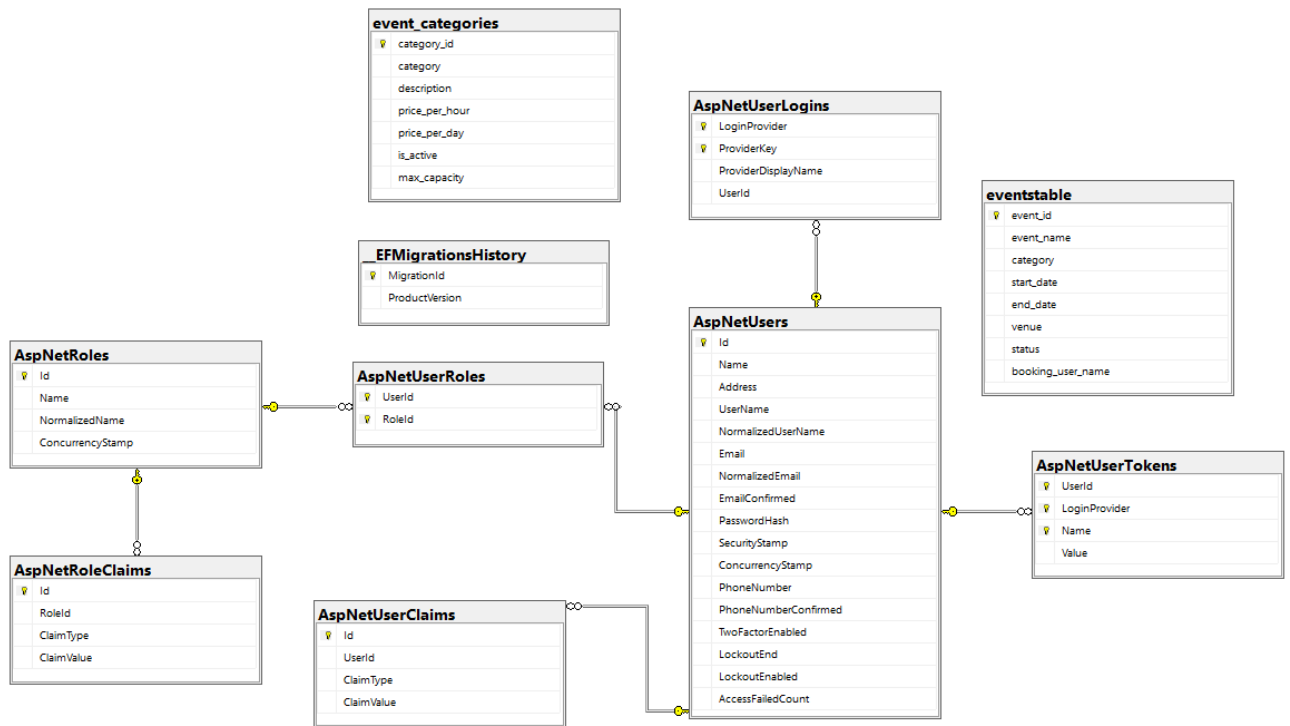
C#: A modern programming language developed by Microsoft, used for building applications on the .NET platform. It is object-oriented and designed to be simple, powerful, and easy to use for creating web, desktop, mobile, and game applications.

Database Design (SQL Server)

Uses Microsoft SQL Server with tables such as:

- AspNetUsers: Registered users
- AspNetRoles, AspNetUserRoles: For role-based access
- Eventstable: Event details
- event_categories: Categories like Workshop, Conference

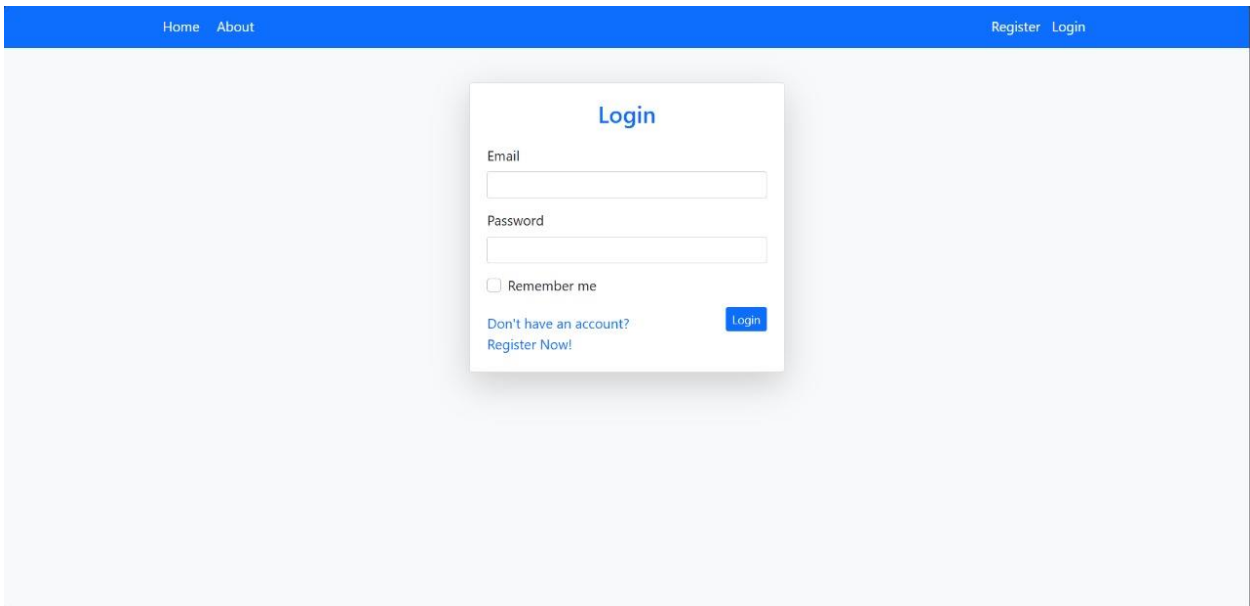
Schema Diagram



Implementation

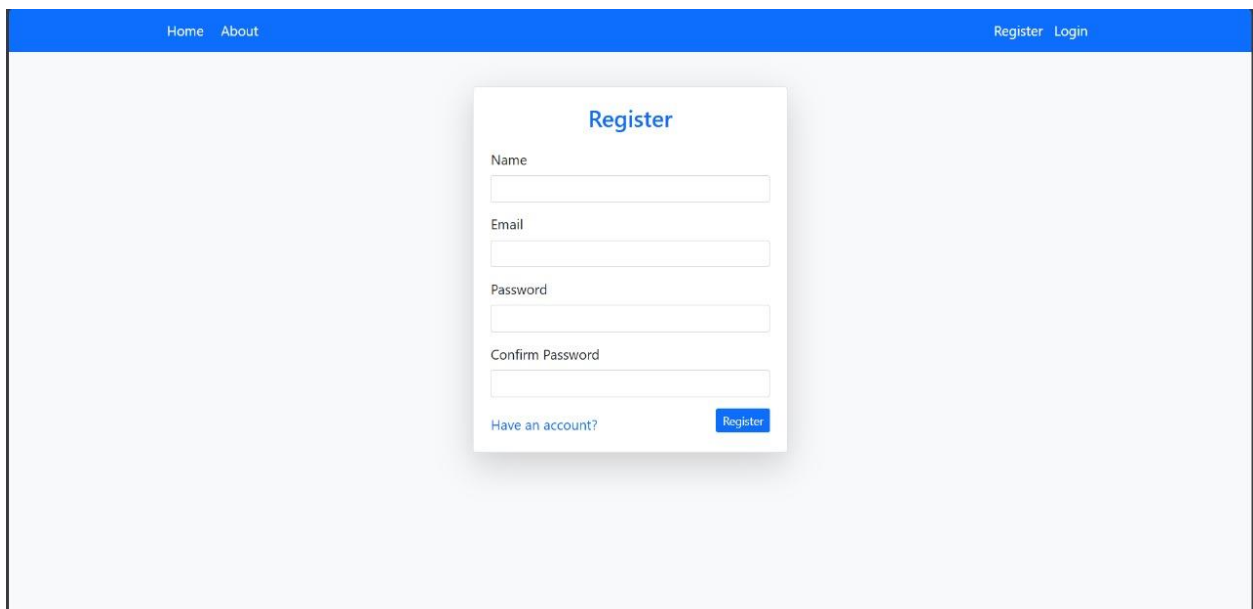
Website interface

- **Login**



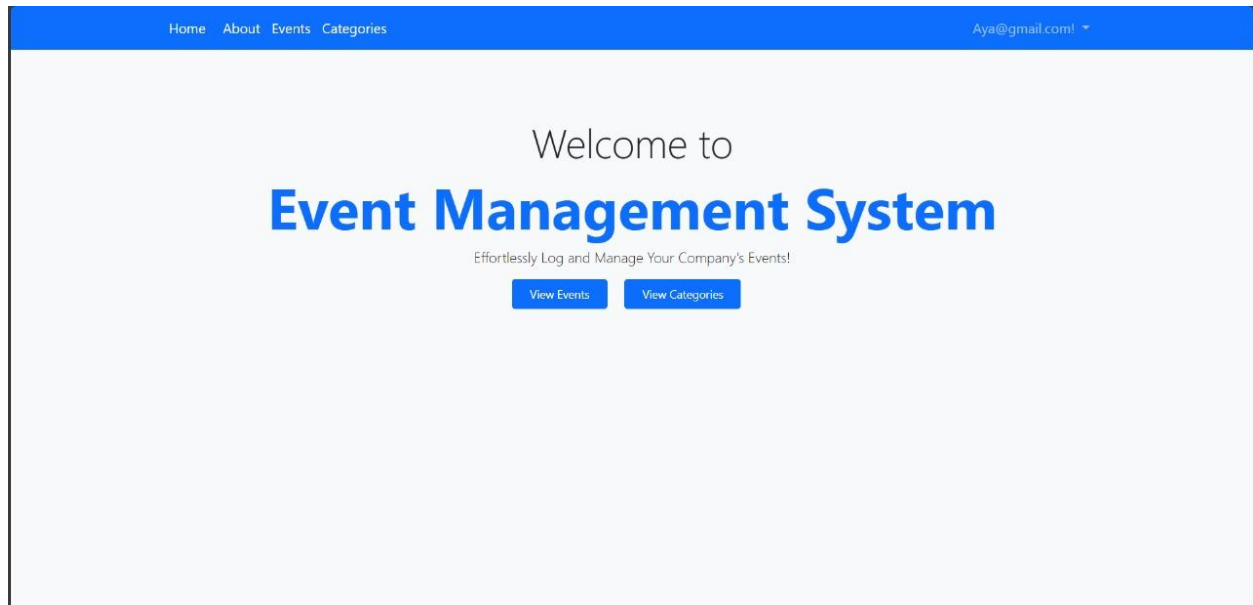
The image shows a web application interface for a login form. At the top, there is a blue navigation bar with the links "Home" and "About" on the left, and "Register" and "Login" on the right. The main content area has a light gray background. In the center, there is a white login form with a blue title "Login". The form contains two input fields: "Email" and "Password". Below the "Password" field, there is a checkbox labeled "Remember me". At the bottom of the form, there is a link "Don't have an account? Register Now!" and a blue "Login" button.

- **Register**

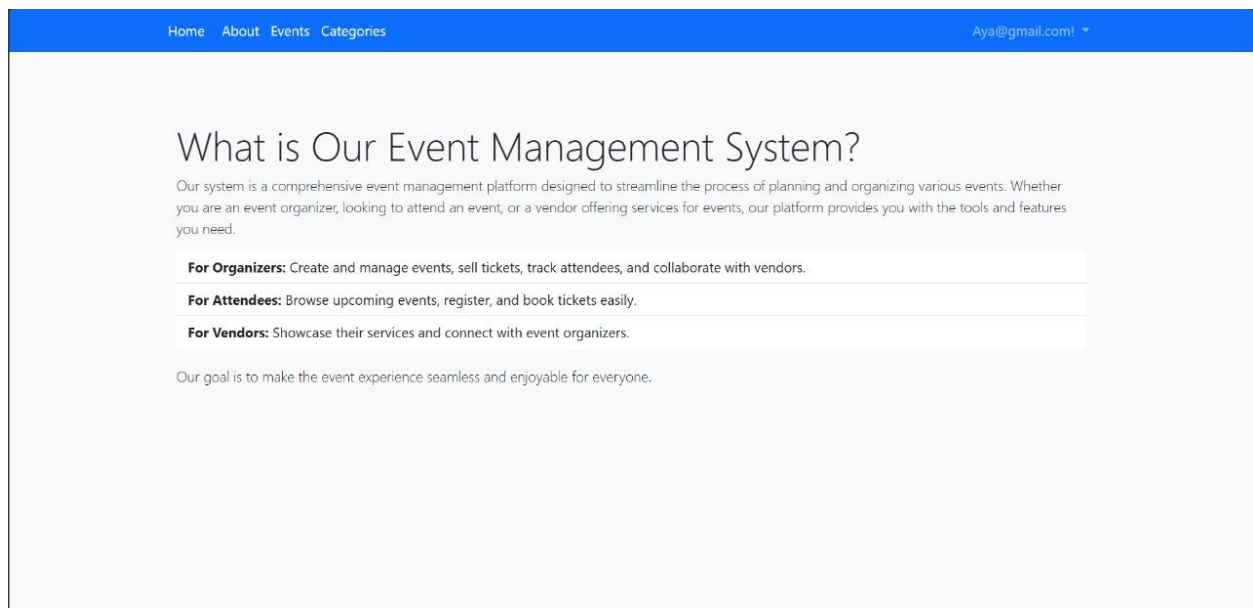


The image shows a web application interface for a register form. At the top, there is a blue navigation bar with the links "Home" and "About" on the left, and "Register" and "Login" on the right. The main content area has a light gray background. In the center, there is a white register form with a blue title "Register". The form contains four input fields: "Name", "Email", "Password", and "Confirm Password". At the bottom of the form, there is a link "Have an account?" and a blue "Register" button.

- **Home**



- **About**



- Category

HomeAboutEventsCategories

Aya@gmail.com!

Categories

Create New

Search events

Search

Category	Description	Price Per Hour (\$)	Price Per Day (\$)	Is Active	Max capacity			
Azure Category	Azure is a Cloud Platform	10.00	40.00	True	20	Edit	Details	Delete

- Events

HomeAboutEventsCategories

Aya@gmail.com!

Events

Create New

Search events

Search

Event Name	Category	Start Date	End Date	Venue	Status	Booking User			
Azure Fabric Egyptian Event	Azure Category	31/05/2025 11:30:00 ص	31/05/2025 06:00:00 م	Nasr City		Aya Khalid	Edit	Details	Delete
AI Event		11/06/2025 02:30:00 م	30/06/2025 05:30:00 م	Cairo		Alexandra Raees	Edit	Details	Delete

Conclusion

Future Work

1 **Booking of Tickets:**

In future iterations, the system can be enhanced by integrating a comprehensive ticket booking module. This module would allow users to reserve tickets for specific events directly through the platform.

Features may include:

- Real-time availability checking.
- Ticket quantity selection.
- Automated ticket confirmation via email or SMS.
- Unique ticket IDs or QR codes for verification at event entry.

2 **Payment Services Integration:**

To support seamless event registration and booking, a secure payment gateway will be integrated. This feature will allow users to:

- Pay for event bookings using various methods (credit/debit cards, mobile wallets, etc.).
- Receive automated invoices and payment confirmations.
- Benefit from secure transaction handling using services like Stripe, PayPal, or Razorpay.
- Admins will be able to track financial records for each event and generate reports.