

Requirement & Design Specification

**EventHub System**

**Subject: PRN222**

**– QuyNhon, March 2025 –**

**Record of Changes**

| **Version** | **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\*A - Added M - Modified D - Deleted

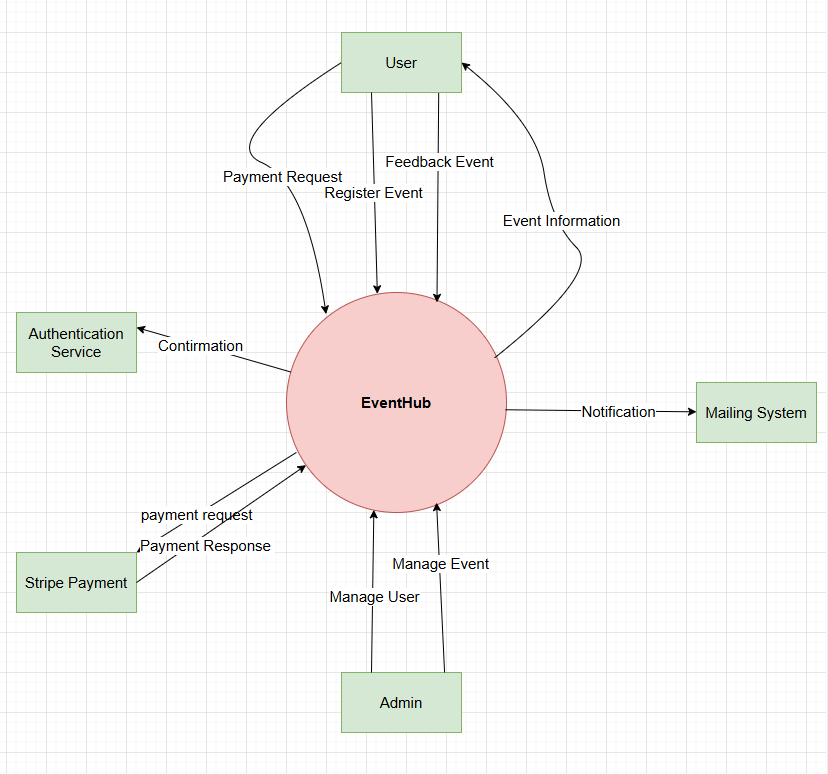
# Requirement Specification

## Problem description

**Purpose:** toBuild a web-based Event Management System named EventHub to facilitate the organization and participation of events at the university. The platform allows students to browse and register for events, make payments through Stripe, and submit feedback. Administrative users can manage events, monitor capacity, and view registered participants.

This system helps streamline event coordination and enhances user experience by integrating secure authentication, real-time event updates, and third-party services like Stripe and Email notifications.. The system requirements are described as follows:

## System context:



*Figure I-1 System context is modeled by a class diagram of the EventHub Web Application*

The diagram in Figure 1-1 illustrates the system context of the EventHub Web Application, outlining how the core system interacts with various external users and external systems.

### External Users

1. **Users (Students or Attendees):**
   * These are end-users who access the system to:  
     + Register or log in using the external identity/authentication service.
     + Browse available events and view detailed information.
     + Register for events and pay for tickets online.
     + Update their personal profiles.
     + Submit feedback after attending events.
2. **Admin:**
   * Admins are responsible for:  
     + Managing events (creating, updating, and deleting).
     + Viewing lists of registered users.
     + Monitoring event capacities to prevent overbooking.

### External Systems

1. **Identity / Auth Service:**
   * Provides authentication functionality for both users and admins.
   * Ensures secure login using credentials or an identity provider (e.g., ASP.NET Identity).
2. **Email Service:**
   * Sends transactional emails such as:  
     + Event registration confirmations.
     + Profile update acknowledgments.
     + Admin notifications or capacity alerts.
3. **Stripe Payment Gateway:**
   * A third-party online payment processor used for event ticket purchases.
   * The system creates a secure **checkout session** using Stripe's API.
   * Handles payment redirection and returns a result (success/cancel) to EventHub.

### System Core

* **EventHub Web Application:**
  + This is the central software system that integrates all business logic and external service communication.
  + It serves as the main interface for both users and admins.
  + Key internal responsibilities include:  
    - Managing events and users.
    - Orchestrating Stripe payment sessions for ticket purchases.
    - Sending notifications via the email service.

## I.3: Functional requirements

## I.3.1 Use case diagrams

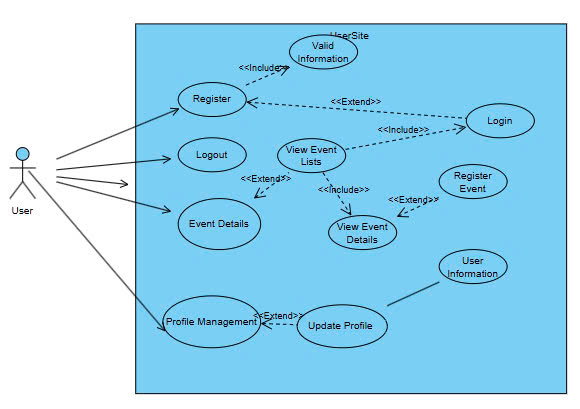
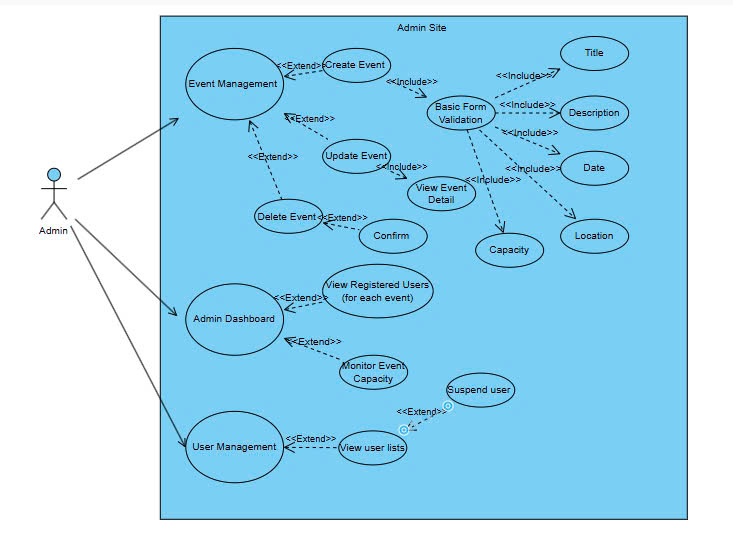
Based on the described requirements in Section I.1 and the system context in I.2, the **EventHub Web Application** is designed as a web platform serving two main user roles: **Students (Users)** and **Administrators**.

Figure 1-2 presents the **Overview Use Case Diagram** for the User role, illustrating key functions such as: Registering/Login via the Identity Service, Browsing and viewing event details, Registering for an event, Viewing tickets, Submitting feedback, Managing personal profile.

Figure 1-3 showcases the Admin Use Case Diagram, covering administrative functionalities such as: Creating, updating, and deleting events, Monitoring event capacity, Viewing a list of registered users, Managing ticket types.

*Figure I-2 Overview Use Case Diagram of User Site*

**

*Figure I-3 Overview Use Case Diagram of Admin Sit3*

## I.3.2 Use case descriptions

Tables 1-1 to 1-6 describe the use cases: Register, Login, Booking Management, Service Management, News Management, User Management.

*Table I-1 Use case description for Registration*

| ID and Name: | **REG01, Register** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/3/2025 |
| Primary Actor: | Student | Secondary  Actors: |  |
| Description: | This use case allows Users to create new account to log in and get access into the system | | |
| Trigger: | The user select the register button below the login form and enter their information for completing the process of registration | | |
| Preconditions: | * The system must be online and accessible. * The register page is available, and the user has access to valid login credentials. | | |
| Postconditions: | 1. Success: The user receives a confirmation email or message indicating successful registration. They navigate to their email to complete the act of registering. 2. Failure: The user is notified of the registering failure and must enter the valid email with the format | | |
| Normal Flow: | 1. The user navigates to the login page. 2. The user enters their username, email, password, and confirm password. 3. The system validate the email with the right format:  * If the email is authenticated, the system then sends a confirmation letter to the student's email. * The user complete the act of confirmation  1. The system redirect user to login page | | |
| Alternative  Flows: |  | | |
| Exceptions: | Invalid Information:   * If the user enters invalid information ( email already in use, weak password), the system displays appropriate error messages and prompts the user to correct the information.   System Error:   * If there is a system error during registration, the system displays an error message and prompts the user to try again later. | | |
| Priority: | High | | |
| Business Rules: | Each email address must be unique in the system.  Password must meet specified security requirements | | |

*Table I-2 Use case description for Log In*

| ID and Name: | **UC-02 Login** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/03/2025 |
| Primary Actor: | Admin | Secondary  Actors: | Student |
| Description: | This use case describes the process of a user (user, admin) logging into the System to access their respective dashboard and functions. | | |
| Trigger: | The user initiates the login process by navigating to the login page and entering their credentials. | | |
| Preconditions: | 1. The user must already have an account in the system.  2. The system must be online and accessible. | | |
| Postconditions: | **1. Success:** The user is authenticated and redirected to the appropriate dashboard based on their role (user, or admin).  **2. Failure:** The user is notified of the login failure and given the option to re-enter credentials. | | |
| Normal Flow: | 1. The user navigates to the login page. 2. The user enters their registered email and password. 3. The user submits the login form. 4. The system validates the entered credentials. 5. If validation passes, the system authenticates the user. 6. The user is redirected to the homepage or dashboard | | |
| Alternative  Flows: | User quits the system | | |
| Exceptions: | **Invalid Credentials:**   * If the user enters invalid credentials (e.g., incorrect email or password), the system displays an error message and prompts the user to correct the information.   **Account Locked:**   * If the user's account is locked due to multiple failed login attempts, the system displays a message indicating the account is locked | | |
| Priority | High | | |
| Business Rules: | 1. Users must enter valid credentials to access the system.  2. The system must validate the role of the user (user, or admin) and redirect them accordingly. | | |

*Table I-3 Use case description for User Management*

| ID and Name: | **UC-03 User Management** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/03/2025 |
| Primary Actor: | Admin | Secondary Actor | None |
| Description: | This use case describes the process of admin manage the user in the system | | |
| Trigger: | The admin selects the "User Management" option from the admin dashboard. | | |
| Preconditions: | 1. The admin must be logged into the system with appropriate permissions.  2. User’s list must already exist in the system. | | |
| Postconditions: | The admin is able to view list of users in the system and able to view the detail of their information | | |
| Normal Flow: | 1. User click on “User Management” on the dashboard of admin 2. The system displays a list of users with their basic information. 3. Admin click on specific user to view their detail of information 4. The system displays a modal containing the user’s details. | | |
| Alternative  Flows: | Suspend User:   * Admin can have an option to suspend a user. Suspended users can not login into the system | | |
| Exceptions: | **System Error**   * If the system encounters an error while retrieving user data, it displays an error message and prompts the admin to try again later. | | |
| Priority | High | | |
| Business Rules: | Only authorized admins can access and view user information.  The system must maintain data privacy and prevent unauthorized access to user details. | | |

*Table I-4 Use case description for Event Management*

| ID and Name: | **UC-04 Event Management** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/03/2024 |
| Primary Actor: | Admin | Secondary Actor | None |
| Description: | This use case describes the process of admin manage the event with the following act: Create, Update, Delete | | |
| Trigger: | The admin selects the “Event Management" option from the admin dashboard. | | |
| Preconditions: | The admin must be logged into the system with appropriate permissions. | | |
| Postconditions: | **1. Success:** The service is created, updated, or deleted successfully, and the system reflects these changes immediately.  **2. Failure:** If any validation or data integrity issue occurs, the system displays an error message and the changes are not applied. | | |
| Normal Flow: | 1. Create new Event:  * The admin selects the "Create new Event" option. * The system prompts the admin to enter event details including title, description, date, location, capacity * The admin submits the event information. * The system validates the data, saves the new service, and displays a confirmation message.  1. Update Event:  * The admin selects the “Update Event” option on the specific service they want to. * The system prompts the admin to enter new Event details including title, description, date, location, capacity * The admin submit the new service information. * The system pop up a confirmation box “ Confirm to change”. If the admin clicks Confirm. The system validates the data, saves the new service * Services will immediately display new information  1. Delete Event:  * The admin selects the “Delete Event” option on the specific event they want to. * The system pop up a confirmation box “ Confirm to Delete” * If the admin clicks “Yes”. The system then immediately disable the event | | |
| Alternative  Flows: | 1. Update Event:  * The admin selects the “Update Event” option on the specific service they want to. * The system prompts the admin to enter new event details including title, description, date, location, capacity * The admin submit the new event information. * The system pop up a confirmation box “ Confirm to change”. * Admin click “Cancel”, and no changes are applied.  1. Delete Service:  * The admin selects the “Delete Event” option on the specific event they want to. * The system pop up a confirmation box “ Confirm to Delete” * If the admin clicks “No”, and the event remains unchanged. | | |
| Exceptions: | **System Error**   * If a system issue occurs while processing the request, an error message is displayed, and the admin is asked to try again later.   **Validation Error**   * If required fields are missing or incorrect, the system displays an error message and prevents the action. | | |
| Priority | High | | |
| Business Rules: | * The system must ensure data consistency and prevent duplicate service entries. * Deleted services should not be permanently removed but marked as inactive to maintain historical records. | | |

*Table I-5 Use case description for Event Management*

| ID and Name: | **UC-05 Event Details** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/03/2025 |
| Primary Actor: | Users | Secondary Actor | None |
| Description: | This use case describes the process of how an user can view and register events in the system. | | |
| Trigger: | The user selects the "Event Detail" option from the dashboard. | | |
| Preconditions: | 1. The user must be logged into the system with appropriate permissions. | | |
| Postconditions: | * The user can **view event list, event details, register** for new event | | |
| Normal Flow: | **View Event List:**   1. User selects the view event option. 2. The system prompts the user to view list of all event available   **View Event Details:**   1. User selects the view event details option 2. The system displays the current events details title, description, date, location, capacity   **Register Event:**   1. User chose option “Register new event” in event detail 2. The system prompts the user to fill in all event details for register | | |
| Exceptions: | **System Error**   * If the system encounters an error while retrieving user data, it displays an error message and prompts the admin to try again later.   **Validation Error**   * If required fields (e.g., title or content) are missing, the system displays an error message and prevents submission. | | |
| Priority | High | | |

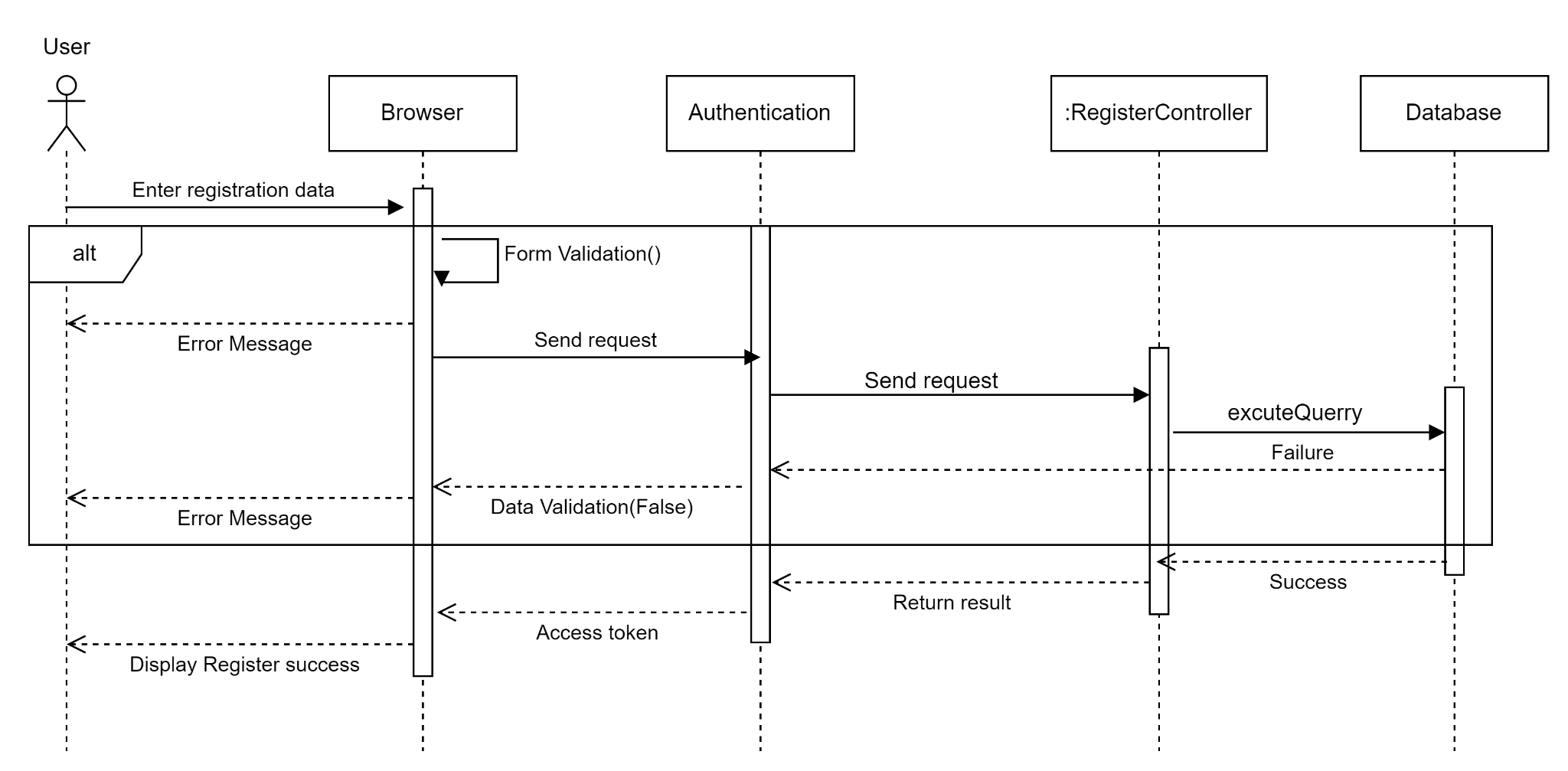
| ID and Name: | **UC-06 Update Information** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/03/2025 |
| Primary Actor: | Users | Secondary Actor | None |
| Description: | This use case allow user to change and update their information | | |
| Trigger: | The user selects the “Edit Information" option from the User information. | | |
| Preconditions: | The user must be logged into the system with appropriate permissions. | | |
| Postconditions: | The user can **update their information** | | |
| Normal Flow: | * The user selects the “Edit Information” option * The system prompts the user to enter changed information * The user save the new information. * The system pop up a confirmation box “ Confirm to change”. If the user clicks Confirm. The system validates the data, saves the new information * Services will immediately display new information | | |
| Exceptions: | **System Error**   * If the system encounters an error while retrieving user data, it displays an error message and prompts the admin to try again later.   **Validation Error**   * If required fields (e.g., title or content) are missing, the system displays an error message and prevents submission. | | |
| Priority | High | | |
| Business Rules: |  | | |

| ID and Name: | **UC-07 Pay with Stripe** | | |
| --- | --- | --- | --- |
| Created By: | NamPN | Date Created: | 23/03/2025 |
| Primary Actor: | Users | Secondary Actor | None |
| Description: | To initiate a payment session via Stripe for purchasing an event ticket | | |
| Trigger: | User clicks the "Pay Now" or "Checkout" button | | |
| Preconditions: | - User has selected an event and ticket type  - Required payment details (amount, URLs) are available  - Stripe API keys are configured | | |
| Postconditions: | - A Stripe Checkout Session is successfully created  - The user is redirected to the Stripe payment interface | | |
| Normal Flow: | * User selects ticket and submits payment * Backend invokes CreateCheckoutSessionAsync with required parameters * Stripe returns a session object * Frontend redirects user to Stripe's hosted payment page | | |
| Exceptions: | * **- Stripe API fails or returns an error →** user sees an error message | | |
| Priority | High | | |
| Business Rules: |  | | |

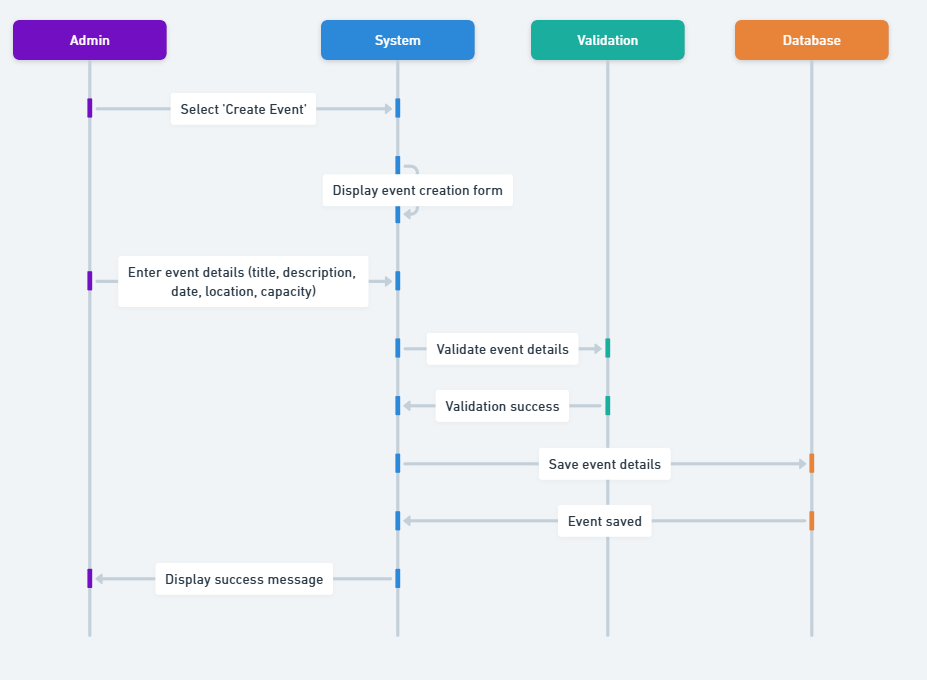
## II. Analysis Model

## II.1.Sequence diagram:

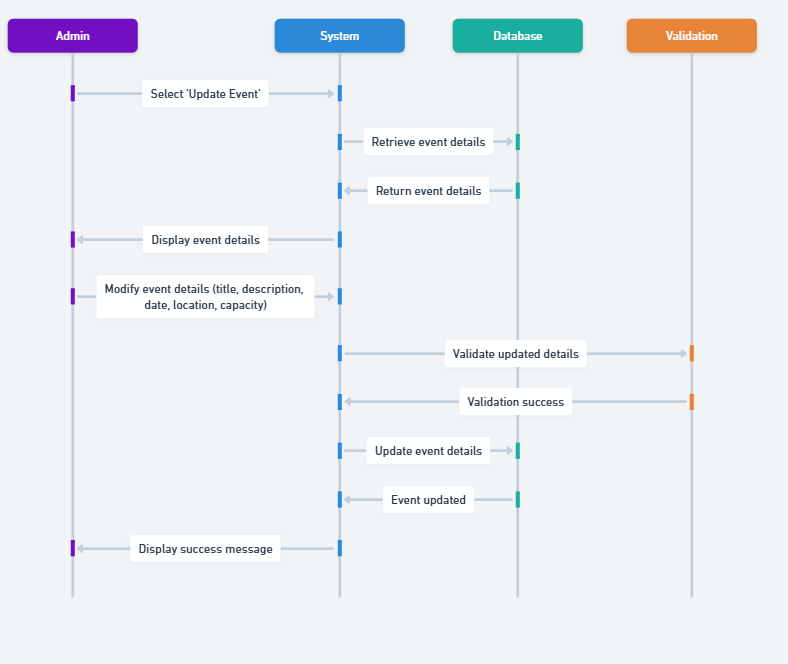
1. Authentication/Login/Logout



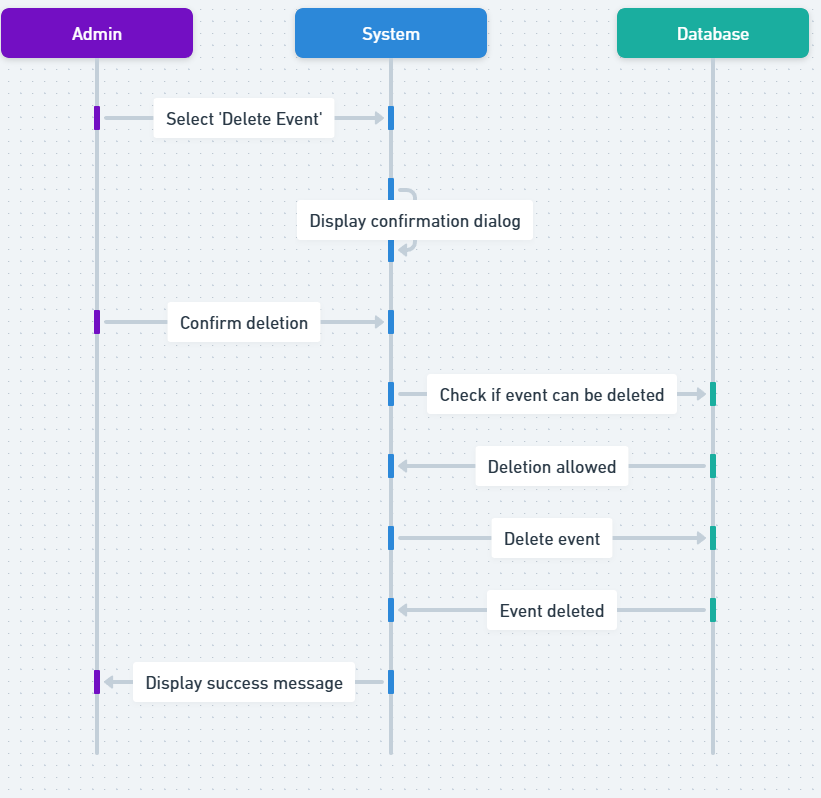
2. Admin Sequence Diagram:



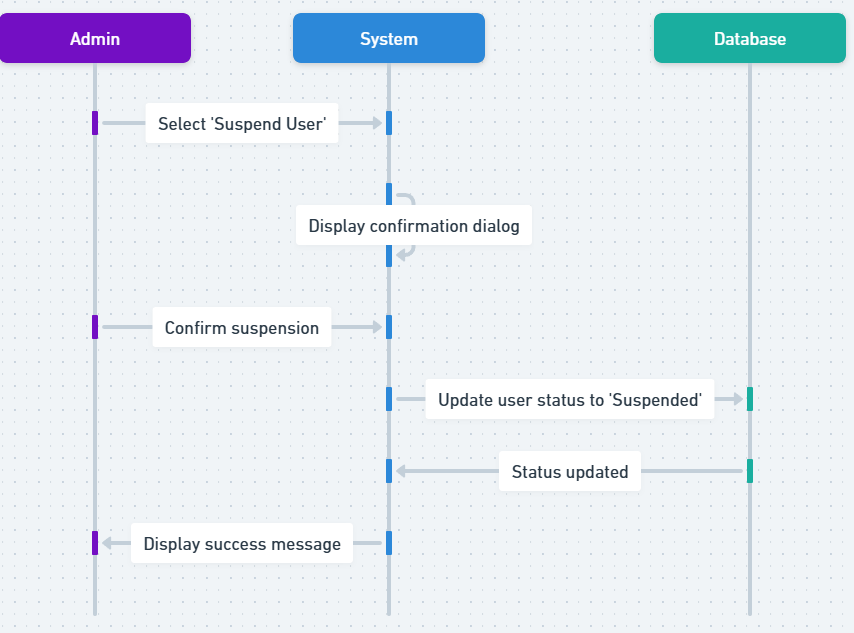
**Create Event Sequence Diagram:**

****

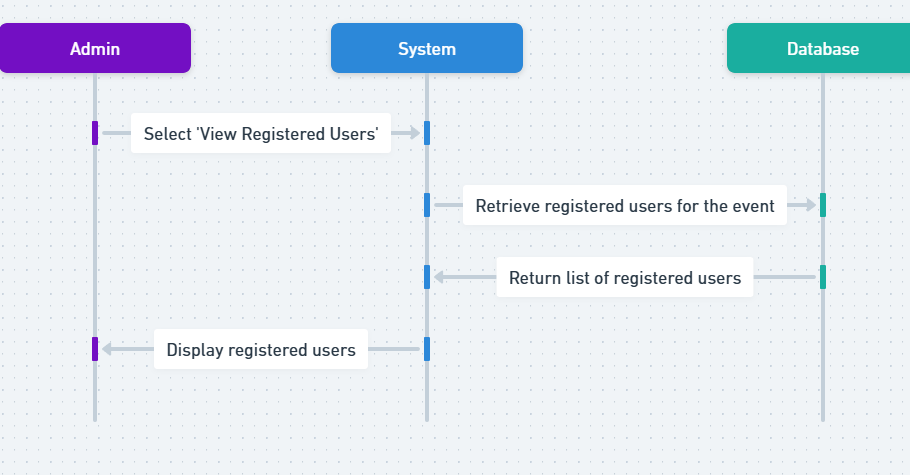
***Update Event Sequence Diagram***

******

***Delete Event Sequence Diagram***

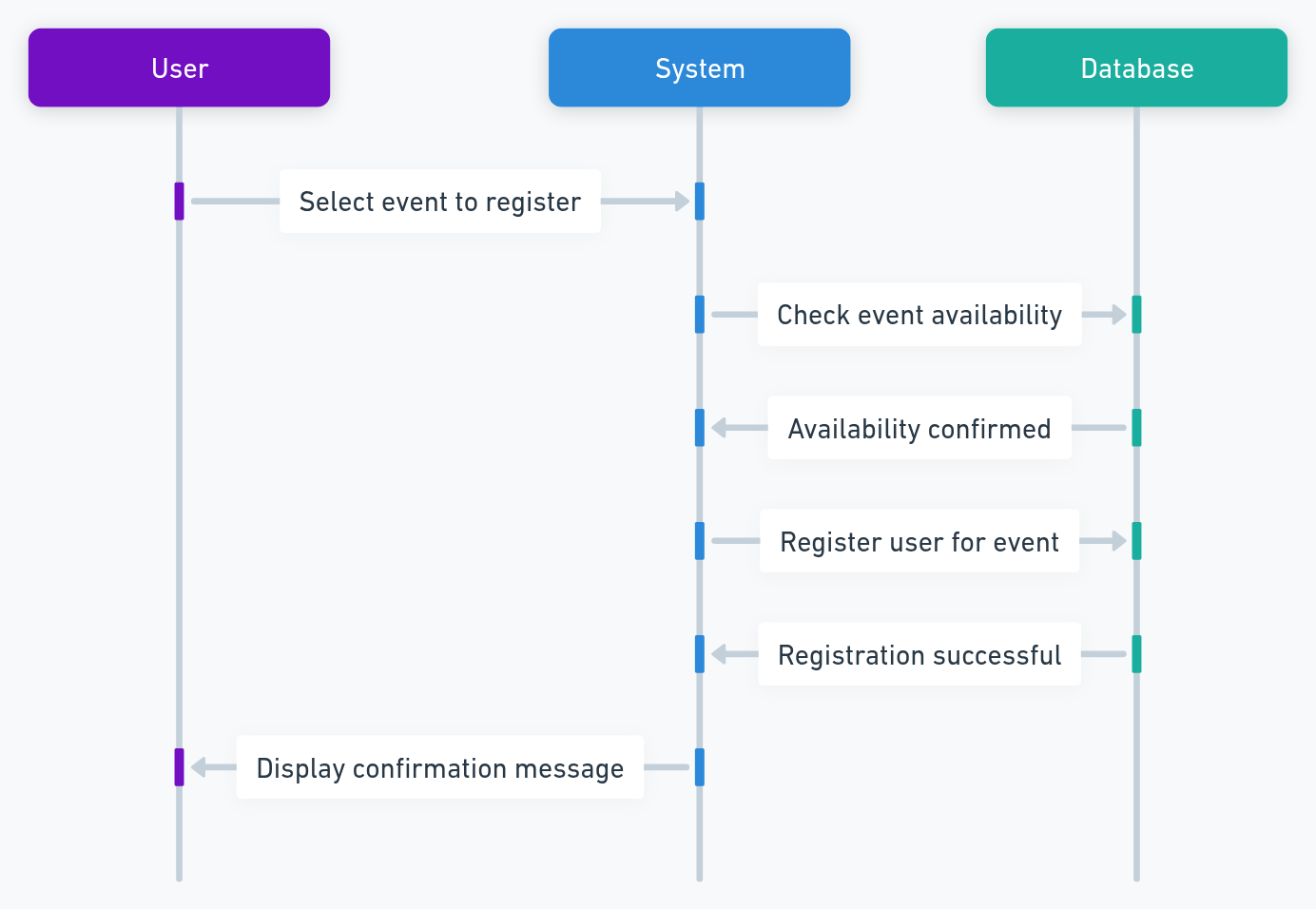
******

***Suspend a User Sequence Diagram***

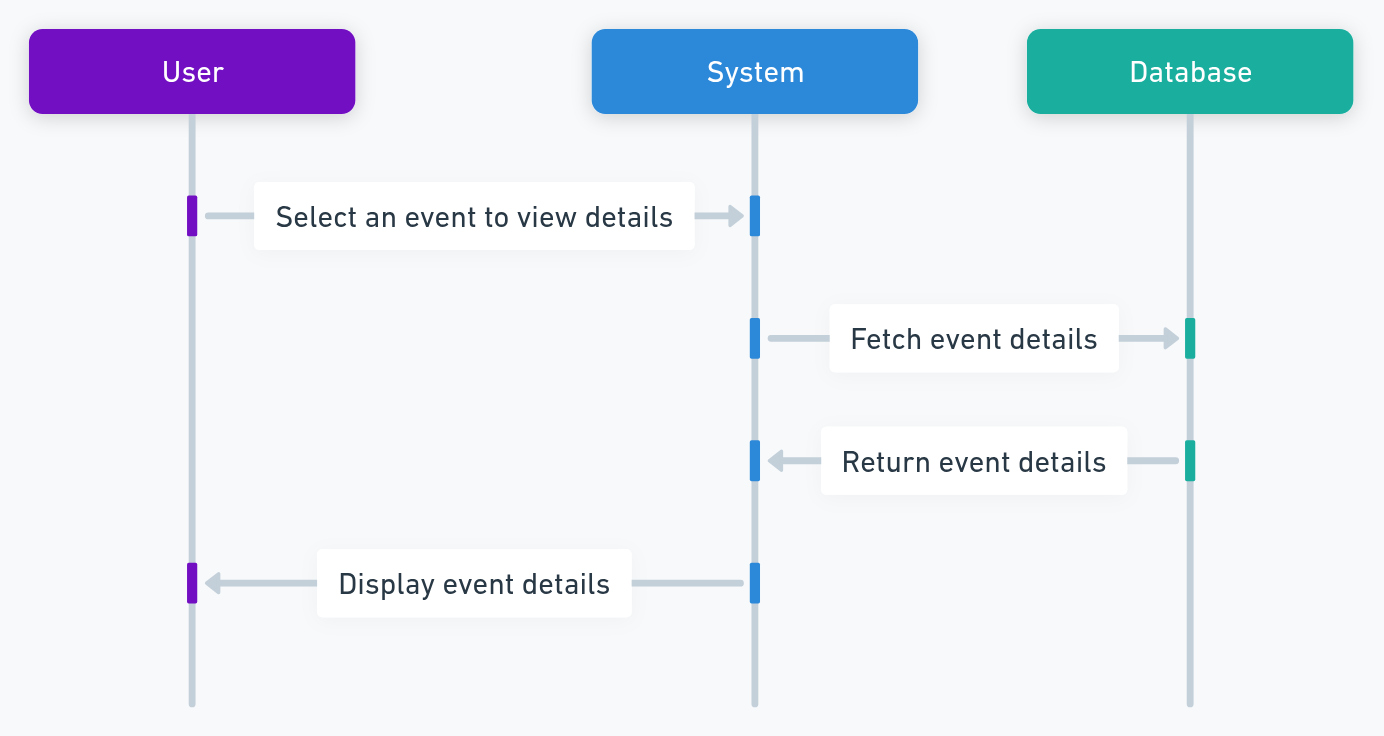
******

***View Registered User Sequence Diagram***

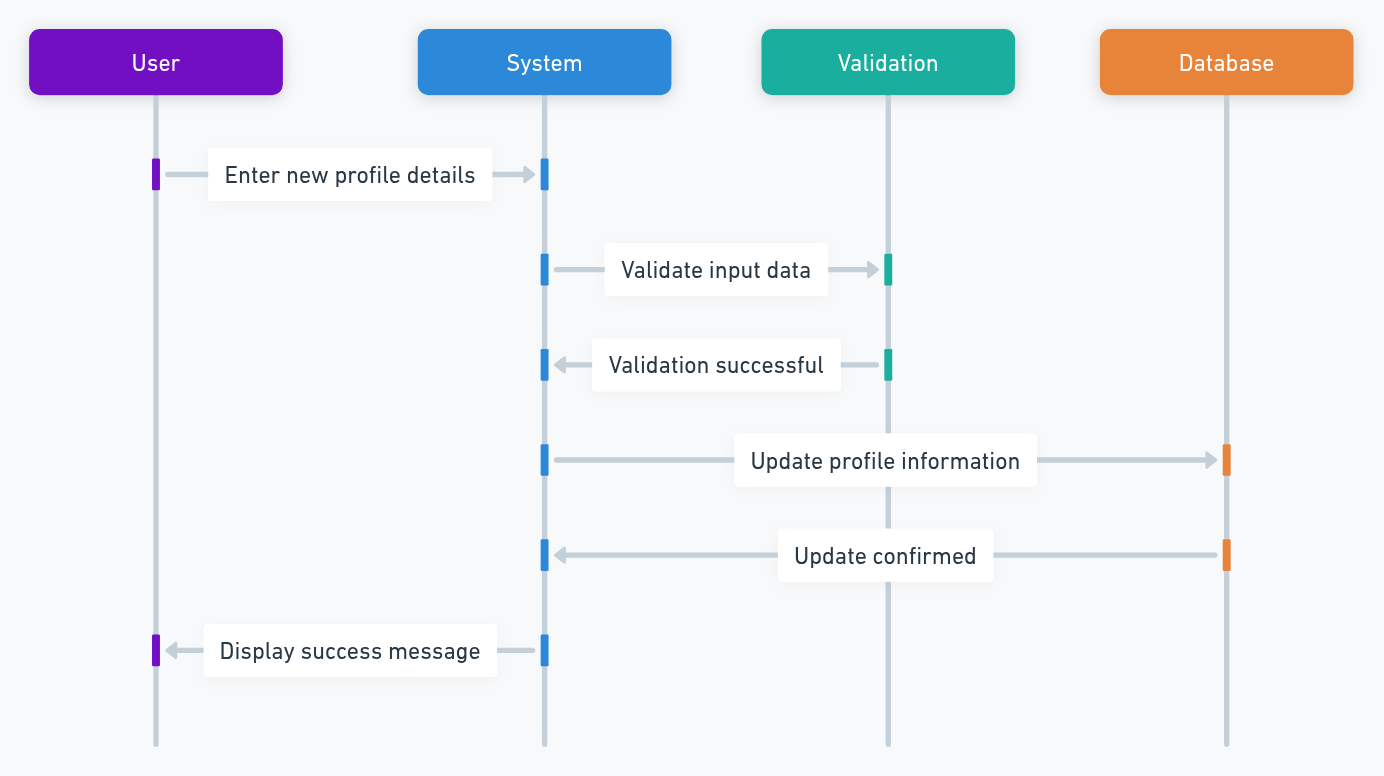
***3.User Sequence Diagram:***

******

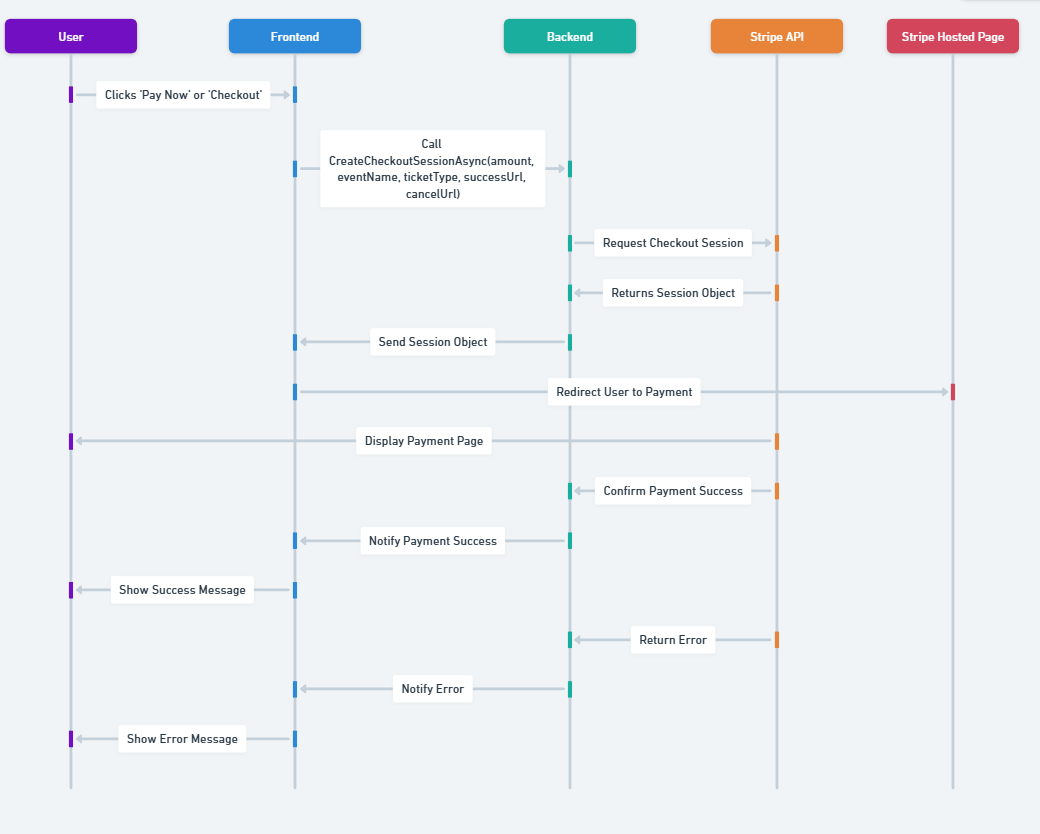
***Register Event for User Sequence Diagram***

******

***View Event Details sequence diagram***

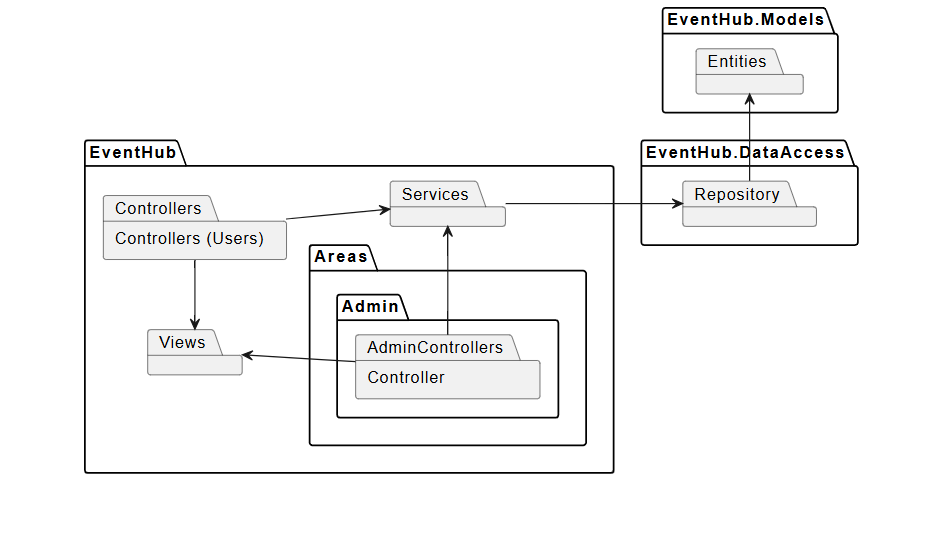
******

***Update Profile Sequence Diagram***

******

***Stripe Payment Sequence Diagram***

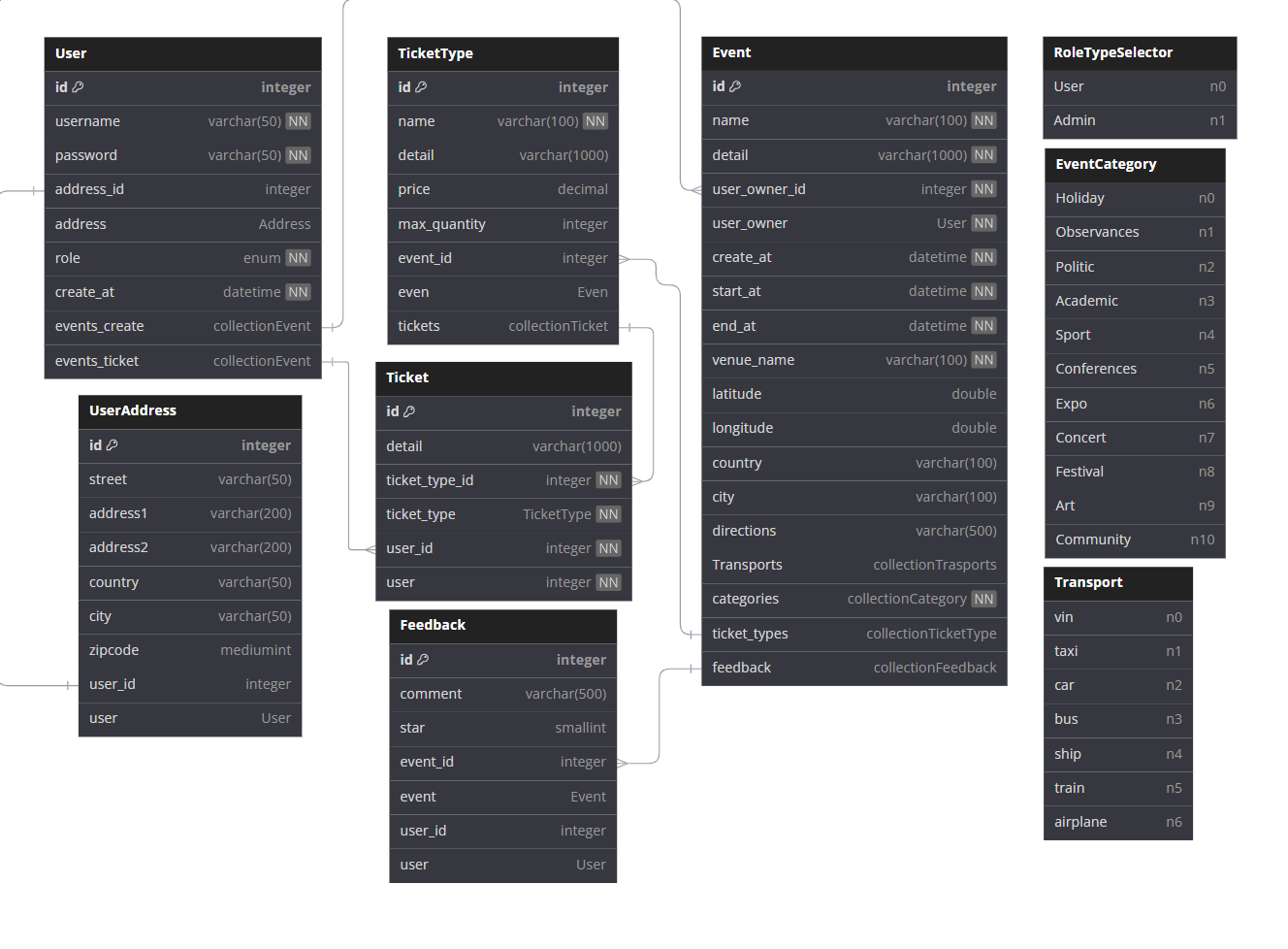
## II.2.Package diagram:



Package diagram for EventHub

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 1 | EventHub.Areas.Admin.Controllers | Dedicated to backend administrative functions such as event management, user moderation, and platform monitoring. |
| 2 | EventHub.Controllers (Users) | Handles user-facing requests like browsing events, submitting feedback, and booking tickets. It passes data to services and renders appropriate views. |
| 3 | EventHub.Services | Central business logic layer. Processes requests from both admin and user controllers and manages orchestration with the data access layer. |
| 4 | EventHub.DataAccess.Repository | Acts as a data gateway, handling communication between the services layer and the database. Encapsulates Entity Framework logic. |
| 5 | EventHub.Models | Defines core data entities used across the system, such as User, Event, Ticket, and Feedback. Used by repositories and exposed to services. |
| 6 | EventHub.Views | Contains UI components (Razor .cshtml views). Displays data passed from controllers via ViewModels. |

## II.3.Database Diagram:



1. **User Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | Integer | Primary Key, Auto Increment | Unique identifier for the user |
| username | Varchar(50) | Not Null, Unique | User’s login name |
| password | Varchar(50) | Not Null | User’s password (hashed) |
| address\_id | Integer | Foreign Key → UserAddress(id) | Links to user's address |
| role | Enum (User, Admin) | Not Null | Role of the user in the system |
| create\_at | Datetime | Not Null | Account creation date |
| events\_create | Collection |  | Events created by the user |
| events\_ticket | Collection |  | Events the user has tickets for |

1. **UserAddress Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | Integer | Primary Key, Auto Increment | Unique identifier for the address |
| street | Varchar(50) |  | Street name |
| address1 | Varchar(200) |  | Address line 1 |
| address2 | Varchar(200) |  | Address line 2 |
| country | Varchar(50) |  | Country name |
| city | Varchar(50) |  | City name |
| zipcode | Mediumint |  | Postal code |
| user\_id | Integer | Foreign Key → User(id) | Links address to a user |

### Event Table

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | Integer | Primary Key, Auto Increment | Unique identifier for the event |
| name | Varchar(100) | Not Null | Event name |
| detail | Varchar(1000) |  | Event description |
| user\_owner\_id | Integer | Foreign Key → User(id) | User who created the event |
| create\_at | Datetime | Not Null | Event creation date |
| start\_at | Datetime | Not Null | Event start date |
| end\_at | Datetime | Not Null | Event end date |
| venue\_name | Varchar(100) | Not Null | Venue where the event is held |
| latitude | Double |  | Latitude of the venue |
| longitude | Double |  | Longitude of the venue |
| country | Varchar(100) |  | Country where the event is held |
| city | Varchar(100) |  | City where the event is held |
| directions | Varchar(500) |  | Directions to the venue |
| Transports | Collection |  | Available transport options |
| categories | Collection |  | Categories the event belongs to |
| ticket\_types | Collection |  | Ticket types for the event |
| feedback | Collection |  | User feedback for the event |

1. **TicketType Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | Integer | Primary Key, Auto Increment | Unique identifier for the ticket type |
| name | Varchar(100) | Not Null | Ticket type name |
| detail | Varchar(1000) |  | Ticket type description |
| price | Decimal |  | Ticket price |
| max\_quantity | Integer |  | Maximum number of tickets available |
| event\_id | Integer | Foreign Key → Event(id) | Event associated with the ticket type |
| even | Reference |  | Reference to Event |
| tickets | Collection |  | Collection of issued tickets |

### Ticket Table

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | Integer | Primary Key, Auto Increment | Unique identifier for the ticket |
| detail | Varchar(1000) |  | Ticket details |
| ticket\_type\_id | Integer | Foreign Key → TicketType(id), Not Null | Links ticket to a ticket type |
| ticket\_type | Reference |  | Reference to TicketType |
| user\_id | Integer | Foreign Key → User(id), Not Null | Links ticket to a user |
| user | Reference |  | Reference to User |

1. **Feedback Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | Integer | Primary Key, Auto Increment | Unique identifier for feedback |
| comment | Varchar(500) |  | User's comment on the event |
| star | Smallint |  | Rating given by the user (e.g., 1-5 stars) |
| event\_id | Integer | Foreign Key → Event(id) | Event being reviewed |
| event | Reference |  | Reference to Event |
| user\_id | Integer | Foreign Key → User(id) | User who submitted the feedback |
| user | Reference |  | Reference to User |

1. **RoleTypeSelector Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| User | Enum(n0) |  | Regular user role |
| Admin | Enum(n1) |  | Admin role |

1. **EventCategory Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Holiday | Enum(n0) |  | Holiday-related events |
| Observances | Enum(n1) |  | National or local observances |
| Politic | Enum(n2) |  | Political events |
| Academic | Enum(n3) |  | Academic conferences, workshops |
| Sport | Enum(n4) |  | Sports-related events |
| Conferences | Enum(n5) |  | Professional conferences |
| Expo | Enum(n6) |  | Trade fairs, exhibitions |
| Concert | Enum(n7) |  | Music concerts |
| Festival | Enum(n8) |  | Cultural festivals |
| Art | Enum(n9) |  | Art exhibitions, performances |
| Community | Enum(n10) |  | Community events |

1. **Transport Table**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| vin | Enum(n0) |  | VIN-based transportation |
| taxi | Enum(n1) |  | Taxi services |
| car | Enum(n2) |  | Private cars |
| bus | Enum(n3) |  | Bus transport |
| ship | Enum(n4) |  | Ship or ferry transport |
| train | Enum(n5) |  | Train transport |
| airplane | Enum(n6) |  | Airplane transport |