Task 1

Implement Ticket management application that provides different abilities for users (Venue manager [1], event manager [2], authorized user [3] and anonymous [4]);

User can have multiple roles (1 – 4). All user roles can act as anonymous user;

User roles:

1. Anonymous:
    a. View all published events and theirs tickets availability;
    b. Register and login into system;
2. User:
    a. Purchase tickets;
    b. See purchase history;
    c. See cart;
3. Event manager:
    a. CRUD operations on event entity;
4. Venue manager:
    a. Manage users;
    b. Setup venue;

User roles and user management **should not be implemented** during this task. This is just a specification for the future tasks. See **Requirements** section for what you should implement.

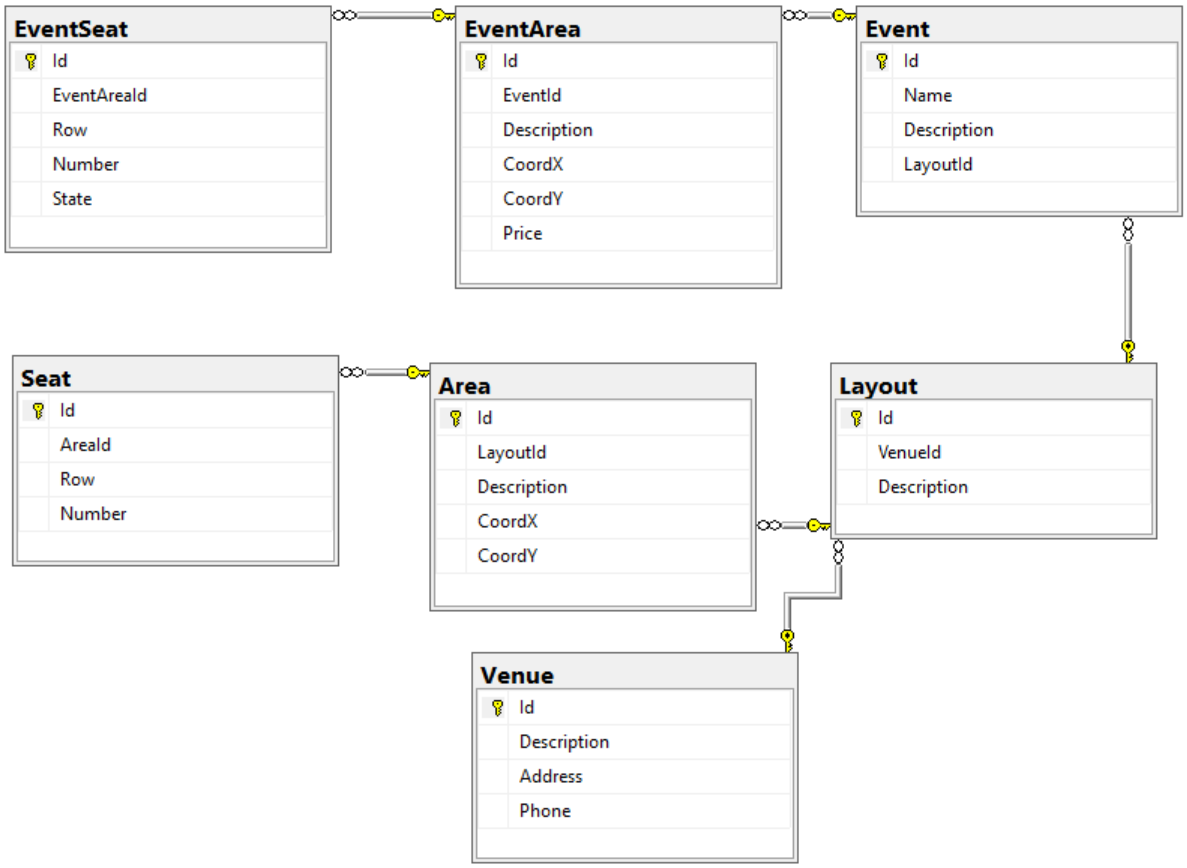

Figure 1 – DB Schema

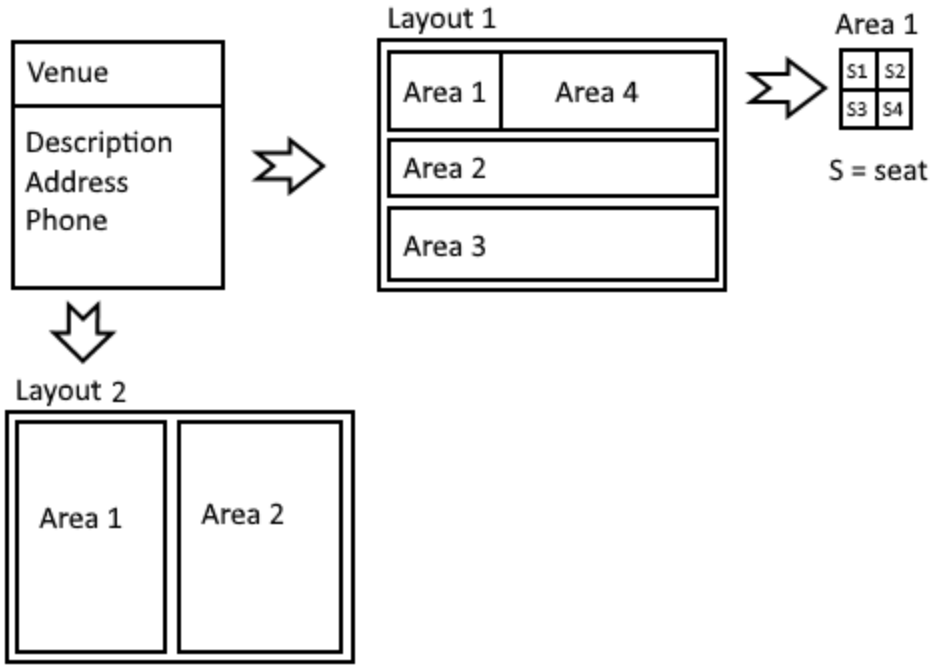

Figure 2 - Venue map

Requirements:

1. Use **.NET 5** for test projects and **NETStandard 2.1** for class libraries and **ADO.NET** for the access to the database, **NUnit** as a test framework**.**
2. Use suggested solution structure from the template in your private repository.
3. Ability to create Venues:
    a. **Venue** consists of layouts; **Layout** is set of Areas. **Area** consists of **seats (fig. 2)**;
4. Ability to create Events;
5. Create solution with 4 class libraries projects.
    a. 1st - Data Access layer with classes and interfaces that allows to manipulate core entities using ADO.NET. Use **Repository** pattern to give access for all entities. Only repository interfaces and domain entities classes should be public. You should Create, Delete and Update Event via store procedure.
    b. 2nd - Business logic layer should reference to DAL and **proxy all repository calls** with validation logic.
        i. Validation: adding business logic validation, for example: check that we do not create event for the same venue in the same time. That we don't create event in the past.
        ii. **Event**: event can't be created without any seats;
        iii. **Venue**: unique name;
        iv. **Layout**: layout name should me unique in venue;
        v. **'Area**: area description should be unique in layout;
        vi. **Seat**: row and number should be unique for area;

c. 3rd - Create unit test project (should test validation logic on business logic). Include positive and negative use cases. It should work multiple times in any order (be independent and repeatable);
d. 4th - create unit test project with integration tests. Integration tests should cover event management API (create event, update event and delete event); Test data should include at least: one venue, two layouts, at least one layout should have two areas, each area should have 5+ seats;

FAQ:

1. Q. Can use EntityFramework instead of ADO.NET?
   A. No.

2. Q. What type of project is required for test.
   A. Class library. Please do not use *test project* template in visual studio. You have to create class library project and install NUnit nuget package.

3. Q. Why business logic layer with only validation in it is incorrect approach?
   A. Validation is not a main feature of your project. Notice the “proxy all repository calls” part. You should provide ability to create all entities in your application and validation should be in front of the write operation.