# 1 System architecture

Choosing the right architecture for developing the event management system is very important for this project. For this we intend to use object-oriented architecture where by we will follow the Model-View-Controller (MVC) design pattern. The selected programming language for developing the application is ASP.NET which is a good choice because ASP.NET provides an MVC framework that is easy to follow and implement. This design pattern has the following components:

* Model – In this component, classes representing core entities defining the application like Event, user and notifications will be created. The classes will consist of methods and attributes that define each entity. The goals are to model each entity as a class with attributes and methods specific to it. Operations necessary for the data access will also be defined at this level.
* View – In this level, user interfaces are designed and created. The user interfaces will make it possible for the end users to interact with the system for example to create and manage events among other functions defined in the functional requirements.
* Controller- In this level, the business logic is implemented based on the functional requirements this the business logic for activities like creating events, updating, and deleting will be defined. User inputs will be handled at this level to make sure the business logic is implemented as defined for each functional requirements and also the nonfunctional requirement.

The application will also use a database where for this we will use SQL server. The database tables will be created based on the entities defined in the model part of the application.

# 2. Project planning

The project will be released based on the following milestones.

1. **Milestone 1: Initialization**

To start off, we will focus on implementing the basic event creation functionality. Our goal is to design user-friendly interfaces for both event creation and viewing. It's important to conduct unit testing to ensure that these basic functionalities are working smoothly.

1. **Milestone 2: Event Management Subsystem**

Moving forward, our next release will involve implementing advanced event management features such as filtering, searching, updating, and deleting. We aim to develop comprehensive user interfaces that make event management a breeze. As always, thorough unit testing is crucial in this phase.

1. **Milestone 3: Notification Subsystem**

In this release, we will be focusing on implementing event notification functionalities. Users should have the ability to customize their notification settings according to their preferences. Testing these notification features thoroughly is an essential part of this phase.

1. **Milestone 4: Reporting Subsystem**

Next up, we will be developing time usage report generation features. Our aim is to create user-friendly interfaces for generating and viewing reports related to the events. Unit testing plays a vital role in ensuring that all reporting functionalities are working as intended.

1. **Milestone 5: Complete System**

In the final release of our project plan, we will conduct integration testing for the entire system. This step is crucial in ensuring that all components work seamlessly together. Additionally, we will fine-tune user interfaces based on feedback received throughout the development process while also focusing on performance optimization and bug fixes.

## Lifecycle Model

**1. Agile Development Sprint Planning**

During each sprint planning meeting, our main objective is to define tasks for the upcoming iteration. We prioritize these tasks based on user feedback and requirements gathered so far.

**2. Sprint Development**

As we progress through each sprint, our focus lies in implementing features and functionalities within time-boxed sprints. We strictly follow Agile development practices which include continuous integration and automated testing.

**3. Sprint Review**

Once the features for a sprint are completed, we demonstrate them to stakeholders for their valuable feedback. This feedback is then incorporated into the planning and development of the next sprint.

**4. Sprint Retrospective**

After each sprint, we hold discussions focused on continuous improvement. We carefully analyze the feedback and lessons learned during the previous sprint. This helps us identify areas for process improvement and make necessary adjustments in subsequent sprints.