5-2018

Cross Platform Web Application Development Using ASP.NET Core

Sanjina Shakya
St. Cloud State University, sshakya@stcloudstate.edu

Cross Platform Web Application Development Using ASP.NET Core

by

Sanjina Shakya

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Computer Science

May, 2018

Abstract

Cross platform software development is a practice of developing an application which can be run on more than one OS platform such as Windows, Linux, MacOS, etc. Today's web application development trend is moving towards cross platform frameworks as we can see that the types of web application we build today are very different from the ones we used to build a decade ago. With the rise of cloud-based systems, mobile technologies, and extensive web interactive environments, web applications today are becoming more dynamic and consequently the web application development process have made tremendous advances. Web applications are expected to be highly performant and scalable. To address the rapidly changing web development landscape and performance scalability requirements there is a need of a powerful framework. ASP.NET Core is one such framework which incorporates modern approach for building web applications. It provides cross-platform capabilities i.e. having a single code base for the application and running it on different platforms. It also provides high performance and modular infrastructure.

This report presents a software development process of cross platform web application in which it explains design, development and deployment of ASP.NET Core web application. A timesheet management system which is developed as a part of this project, helps to manage timesheet operations of an organization and provides efficient way to track time used by employees in various projects and their time offs. The application is called MyTime. It is built utilizing ASP.NET Core MVC Framework and Entity Framework Core which is deployed on two platforms; Windows and Linux. This report describes the detail steps of deployment for both platforms. It provides an overall idea of how ASP.NET Core framework can be utilized to build a web application with cross platform capabilities.

Table of Contents

List of Tables

List of Figures

Chapter I: Introduction

Motivation of the Work

The web application development process today is mostly monolithic because the development process of an application targets a specific operating system and is tied to a specific application framework (Aroraa, 2017). For example, application developed using .NET frameworks is targeted to Windows OS and application developed using iOS frameworks is targeted to iOS. The benefit of using monolithic approach is that the framework itself provides a complete infrastructure to develop an application but it is not the best way because components of such applications are highly coupled which makes it hard to change one part without affecting other. Also, developing OS specific applications makes it less compatible with other operating platforms and hence they are expensive to develop. So, we need modern web development framework with cross platform capabilities.

Further, with the rapid progress in both hardware and software technologies, web applications have made tremendous advancements and have become extensively interactive. The user interface of most web application includes dynamic functionalities and complex backend system which incorporates distributed infrastructures. Web applications are embracing cloud-hosted shared computing platforms such as Azure and Amazon EC2 where application run from hosted platforms rather than from dedicated servers. On top of that, web pages need to be lightweight to support mobile and IoT devices and load quickly (Aroraa, 2017). That is why we need a web development framework that supports modular and cloud-optimized capabilities.

These challenges helped in identifying a need of a lean web framework with a new approach of web application development. ASP.NET Core as a modern framework, addresses all the scenarios mentioned above by providing modular, extensible, cross-platform and leaner framework to develop every kind of web application that can be accessed with browsers, tablets, mobile phones and IoT devices (Charbeneau, Bristowe, & Basu, 2017). The motivation behind this paper is to design and develop a web application utilizing cross platform web development approach as an alternative to native monolithic development approach. To research about most recent cross platform web technologies and learn/understand the framework and tools used in web development process using ASP.NET Core.

Overview

This paper spans across a range of subjects such as software development process and deploying a web application in more than one OS platforms. To help the readers navigate, the paper is divided into four parts and a brief overview is presented here.

Background. Chapter II contains a brief introduction to the fundamentals of web application development process and Microsoft .NET web framework. It also explains the basics of a new .NET Core and ASP.NET Core frameworks. It describes about capabilities of cross platform web application development which has been made possible because of modern software practices.

Design and implementation. Chapter III is more focused on software development process of a timesheet management web application called MyTime. It describes the overall design phase of the application which includes logical architecture of the system and design approaches utilizing UML diagrams such as use case diagrams and sequence diagrams. It presents functionalities of the timesheet application and provides database design diagram of the application.

Deployment. Chapter IV explains different configuration settings required for deploying the ASP.NET Core application mentioned in previous chapter. It presents the work on the actual implementation and provides examples of code to illustrate important steps to obtain cross platform capabilities. It describes main steps to deploy the application to production environment in two different OS platforms i.e. a Windows 10 system using IIS and a Linux based system - Ubuntu using Nginx.

Discussion. Chapter V provides the analysis of the application developed using cross platform approach, summarizes the need and advantages of using ASP.NET core framework to build such web applications. It also includes recommendations for future work.

Chapter II: Cross Platform Web Technology

Fundamentals of Web Application Development

It is very helpful to understand the fundamentals of web application development infrastructure before explaining about the ASP.NET Core and its features. In general, the fundamental components in web application development are hosting platform and application framework. Figure 1 below shows basic components of a software stack required for web application development:

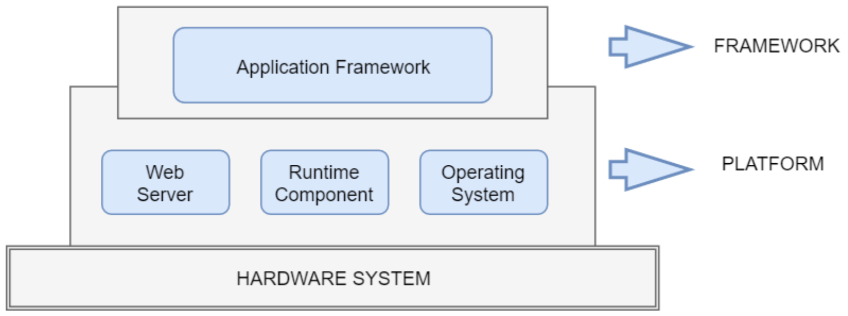


Figure 1: Fundamental Infrastructure for Web Development Stack

Web hosting platform. The platform provides host environment for web application deployment. It is a low-level API which provides an implementation of the HTTP fundamentals. It helps server applications to serve web pages containing static and dynamic contents over HTTP. Hosting platform primarily consists of an operating-system, web-server and runtime components. Operating system provides an environment on which programs runs. Web/Http server serves in handling http request from the browser and send response back to client. Runtime components provides type safety, memory management and code compilation services.

Web application framework. It provides web application development environment. Frameworks are reusable components that provides most required common functionalities of any application. For examples: codes for input-output operations, database connection, security attributes, threading etc. It consists of set of class libraries within which we can implement custom codes to build a web application. Application framework also provides files organization structure. Technically, framework operates on top of the hosting platform.

Introduction to .NET Framework and ASP.NET

.NET Framework is an oldest and well-known application development stack developed by Microsoft which includes both framework and platform. It facilitates in building, deploying and running various software applications such as windows GUI, command line, web application and web services on Windows machine (Richter, 2002). It contains numerous libraries containing wide range of functionalities. The major components of this platform are Common Language Runtime (CLR), Base Class Library (BCL) and Application Framework Library which are shown in Figure 2 below:

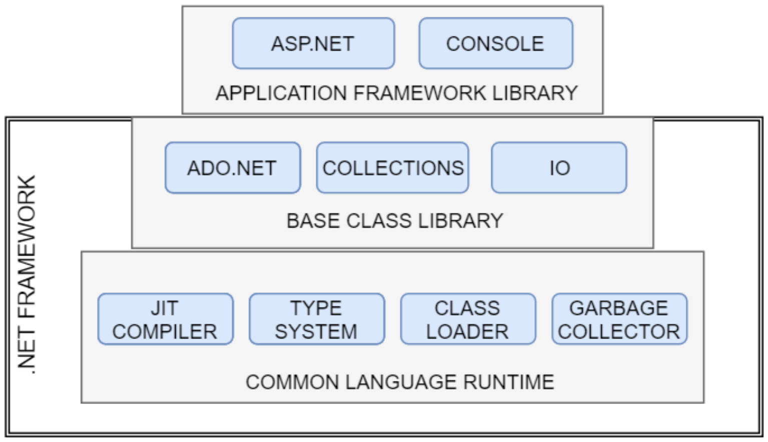


Figure 2. ASP.NET Framework in .NET Platform

Common language runtime (CLR). This is a virtual engine which provides a runtime environment for .Net applications. It provides various services such as code verification, object management, security, garbage collection, memory management, etc. to manage the execution of

.net applications. The primary job of runtime is to load .NET libraries, locate the entry point of an application and execute it whenever a request is received from a web server. The runtime, web server and OS collectively provide the infrastructure to host and execute an application (Watkins, Hammond, & Abrams, 2002).

Base class library (BCL). The base class library includes classes, interfaces and types which are prebuilt functionality which can be used while developing an application. For example, System.Configuration provides classes that gives access to .NET configuration settings, System.IO helps in reading and writing data streams and files, System.text allows string manipulation, System.Collection provides classes representing general purpose collections such as lists, arrays, queues, hash tables etc., System.data provides functionality to access various data sources as SQL.

Application framework library: ASP.NET. ASP.NET is a web platform extension of .NET framework class libraries that provides basic functionalities for creating web-based applications targeting the Windows platform. It not just supports all the features of base class libraries but also includes a set of web specific services, such as http request processing, secure authentication etc. The heart of ASP.NET is System.Web.dll which works for windows and IIS. It provides a way to run web application in IIS that allows interaction with HTTP requests and responses.

Web server: IIS. Web server is a layer that accepts the network requests routed by OS and then pass the request off to an application framework. Internet Information Services (IIS) is a web server for Windows platform built by Microsoft which handles ASP.NET functionality.

Built-in functionalities of web server are user authentication, content caching and serving static files without having to execute it by application server.

Limitation of ASP.NET. ASP.NET is a mature framework, as it has evolved into stable web framework over time since it was first released in 2002. But it has grown huge and complex because it contains all the core classes that a web framework needs during development. The .NET web stack intertwines many features in a giant monolithic infrastructure that has very little separation of concerns. So, when an application is built using this framework, it becomes crowded with framework components that get included by default and hence makes it unnecessarily bulky.

Also, ASP.NET web application essentially targets windows web hosting server and Windows OS, which prevents its use on non-Windows platforms. Therefore, it lacks cross platform capabilities. It is based on the System. Web assembly, which contains functionality coupled with Windows. But web applications today require capabilities to run on multiple platforms and are expected to be highly performant. ASP.NET Core was designed to meet those requirements. It was created by completely redesigning an existing .NET web framework infrastructure from ground up. Figure 3 below shows high level overview of ASP.NET compared to ASP.NET Core.

Figure 3 below represents the current state of .NET ecosystem. Now the ecosystem has new runtime .NET Core which helps to fulfill the requirements of building modern applications.

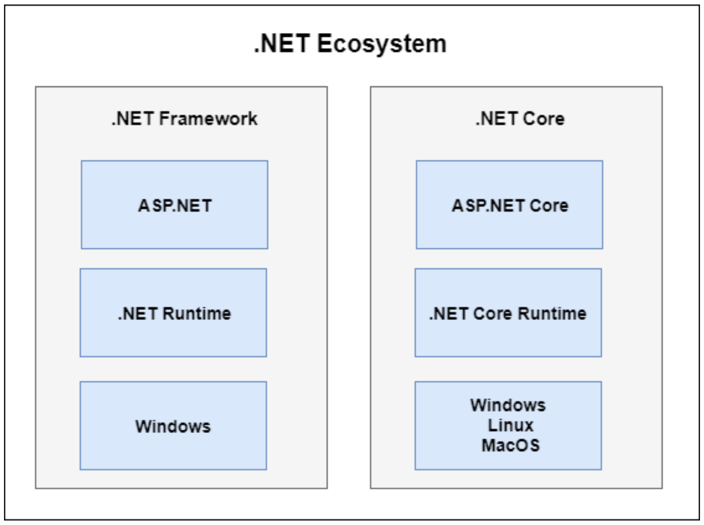


Figure 3. .NET Ecosystem

Introduction to .NET Core

.NET Core is a lean version of .NET framework which was released in 2016. It is designed with modularity and performance in mind. The main idea of .NET Core framework is a pull mechanism which means that the application starts with a light infrastructure and the required modules are loaded via NuGet packages as per need of application itself. It consists of a subset of .NET framework class libraries which is called CoreFX and includes an optimized runtime which is called CoreCLR. Figure 4 shows main components of the framework.

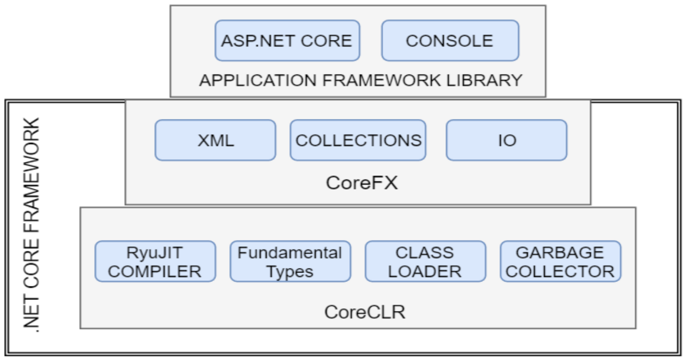


Figure 4. ASP.NET Core in .NET Core Platform

Core CLR. It is a .Net Core runtime which includes most basic components for a runtime environment such as JIT Compiler, type system and class loader. The Core CLR is compiled for each OS platform so that different setups are created for each platform. Hence this provides power to run .net core application on any platform.

CoreFX. .Net CoreFX is a collection of fundamental libraries like IO, collections, file systems, XML which are subset of class libraries in full .NET framework. It is also termed as Unified BCL because it defines a common base layer of primarily needed libraries.

Web server (Kestrel). .Net Core runtime uses cross platform web server called Kestrel. Since the platform itself is highly decoupled in terms of use of operating system and web server, now the web server for .Net Core platform is not restricted to IIS only. Kestrel includes asynchronous IO libraries and only does I/O work. All non-IO work including HTTP parsing and framing is done by using standard fully featured web server.

ASP.NET Core Framework

ASP.NET Core is a latest .NET web development stack. It is a cross platform framework for building modern cloud-based web applications. It encourages modular structure of an application because it involves modular components which provides flexibility while creating software solutions. ASP.NET Core applications are purely cross platform in nature as they can run on any operating system such as Windows, Mac and Linux.

It is constructed with modern software design principles on top of the new .NET core platform which makes it a fully featured web framework. It uses MVC architecture, uses a modular HTTP request pipeline which is an efficient way to process requests using small discrete modules, uses dependency injection which considered as software best practices (Olivera & Bruchet, 2017.

Features:

- It is completely modular because it is based on NuGet packages that provides advantage of including just those components which are necessary for the application. This approach makes it a lightweight framework.
- It provides seamless transition from on premises to cloud-based deployments.
- It provides cross platform capabilities and helps to develop portable applications.
- It uses MVC pattern as its default web architecture which provides separation of concerns and helps to build easily testable and scalable applications.

ASP.NET core request pipeline. The http request pipeline specifies how web application should respond to http requests. When application starts, it runs only the essential components needed to start the server. Additional features are initialized by invoking the respective middleware. There are features like authentication, logging, security, MVC, entity framework, etc. which are needed in lifetime of a request are loaded as middleware services (Oliveira & Bruchet, 2017).

Figure 5 shows the basic flow of http request from client and corresponding response from the server.

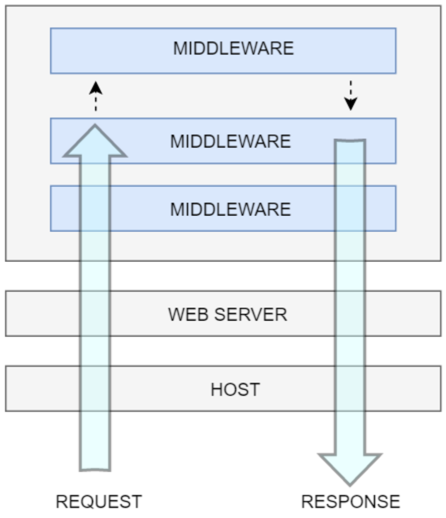


Figure 5. Http Request Pipeline in ASP.NET Core Framework

As shown in Figure 5, working principle in a request pipeline follows clean request response pattern. A browser makes a request to an application and a web server receives the request and pass to a stack of middleware. Middleware is a software that is assembled into an application pipeline to handle requests and responses. Each middleware component can either manipulate a request and pass that to next middleware for further processing or correspond directly and not execute rest of the pipeline.

ASP.NET core MVC. The ASP.NET Core MVC is the implementation of MVC pattern in ASP.NET Core framework. MVC is a software architectural pattern which serves in separating the application in three major components namely; Model, View and Controller. It helps in defining the responsibility for each of the components and structuring their interactions in achieving the overall goal of the application (Oliveira & Bruchet, 2017).

- Model: It represents the domain data. It is only component among three which can talk to the Database.

- View: View as its name implies, is responsible for what to present to the user. Basically, the components will incorporate view templates which helps in including programming logic in the HTML file. It primarily handles the look and feel of user interfaces.
- Controller: The controller is responsible for interacting with Model and View components. As soon as it receives a request from server, it talks to the model and sends the appropriate view to the user.

MVC provides separation of concerns which brings flexibility to the web application development. Hence each component can be managed, tested and scaled separately and individually.

Entity Framework Core

The Entity Framework is a data access technology which provides automated mechanism for accessing and storing data in database. It works as an object relational mapping (ORM) framework which bridges the gap between two paradigms: (a) Object Oriented paradigm, and (b) Relational paradigm.

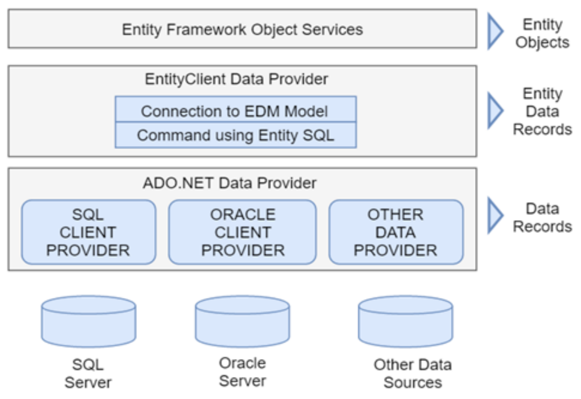


Figure 6. Entity Framework Core

The Figure 6 above shows the major layers of the framework. Brief description of each layer is listed below:

Entity data model (EDM). It is a link between model and a database which decouples the application from the underlying data store. Technically, it is a conceptual model schema that is designed to reflect business domain (Kanjilal, 2008). Conceptual model relies on additional metadata, which describes structure of database (DB structure schema) and a map between a conceptual model entities and database tables. Figure 7 below shows basic structure of EDM.

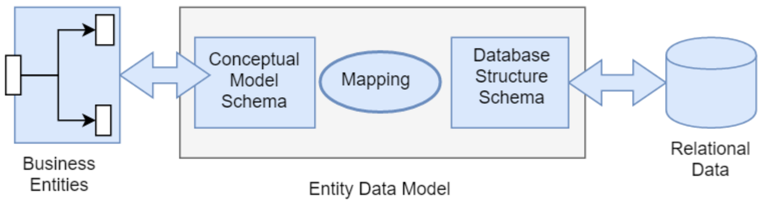


Figure 7. Entity Data Model

We can see that it is not a direct reflection of database so a complex relationship like inheritance and encapsulation can be defined in EDM. The mapping will help find Entity Framework their way back to tables and columns in database

If the system has a legacy database, EDM Designer can be used to reverse engineer database schema into a model and then the obtained model can be customized as per business needs. But, if database is not ready for system, a new EDM can be created using designer tool where entities and their relationship can be defined as per requirements. Then database schema can be generated using the EDM. Now the framework itself pair up the existing with the classes in an application.

ADO.NET. ADO.NET is set of classes in .NET Base Class Libraries which provides interface to access relational databases and XML documents. ADO.NET Data Provider is a major component within the technology.

Data-Provider handle Connection object, Command object and Data Adapter object. The Connection object provides connectivity to the data source. The Command object enables access to database commands to return data, modify data, run stored procedures etc. The Data-Adapter provides the bridge between the Data-Set object and the Data Source.

Entity client provider. It is a data provider which is a layer above ADO.NET data providers. It offers same programming abstractions as ADO.NET data providers but adds mapping capabilities that translate queries expressed in terms of model into equivalent queries in terms of tables in the databases. It uses Entity SQL which adds traditional SQL with constructs necessary for querying in terms of higher level modelling concepts in EDM.

Object services. In Object services layer, entities are represented as object instances of data classes. These objects are mapped to the entity types in the model. It supports querying with LINQ to Entities. It also keeps track of changes made to the objects and when a state change happens, it is reflected to database.

Advantages:

- Application can work in terms of domain centric conceptual model, including types with inheritance, entity relationship.
- Applications are decoupled from hard coded dependencies on a database system.

- Mappings between conceptual model and the storage specific schema can be changed without changing the application code.
- Enhance developer productivity as the framework writing less code.

Chapter III: Developing Application Using ASP.NET Core

MyTime Application: Description

This project is a web-based timesheet management software. The system can be utilized to log and manage the amount of time spent by an employee on various projects. User can request/review time offs and view timesheet reports. Timesheet management is one of the essential process in any workplace as it helps in tracking employee's time and invoicing accurately for work that has been done. It also helps to analyze detail information about project cost and timeline. This software solution can be utilized by any organization to effectively manage the timesheet process.

The system has three kinds of roles, Employee role, Supervisor role and Admin role. An admin user will be responsible for registering users, adding and updating projects and tasks, assigning project and tasks to employees, etc. A supervisor user can review timesheet submitted by under his/her supervision, review time offs, assign projects/tasks as well as submit and view timesheet history. An employee user can create and submit timesheet, request time off and view previous timesheet history. All users can create reports to view their timesheet and time off request history. The application has following core features:

- Timesheet Creation and Submission: All employee users will be able to enter and submit timesheet on a weekly basis. A user can go back seven days and update timesheet, all timesheet entries before that period will be available in read only mode. Submitted timesheet goes to the supervisor and the supervisor user will can either approve or reject the timesheet.

- Project Management and Assignment: An admin user can add or update projects into the system. The projects can also be assigned to an employee user. An employee user will be able to submit timesheet only on the project assigned to him/her.
- User Registration: An admin user can register an employee as a user of the timesheet management system. The admin user will assign a username which will be the same as email address of the employee as well as set a password and assign a role to the new user. The newly created used will be able to log into the system with the correct credentials.
- Timesheet Review and Approval: An admin or supervisor user can review timesheet submitted by the employee user. System by default show the weekly timesheet view however timesheet of previous weeks can also be viewed. After reviewing the timesheet, it can be either approve or unapproved by the user.
- Time off Review and Approval: An admin or supervisor can review time-off requested by the employee user. They can select among the employee who are under their supervision and either approve or unapproved the time off requested by accepting/rejecting the link on list of time off entries.
- View Reports: The system allows users to view timesheet reports grouped by the project. To see reports, users can select the date range for the week they want for reports. The admin user will be able to run reports for all users in the system whereas all other users will be able to run report for themselves only.

System Architecture

The application implements layered architecture which is one of the most widely used architecture for web application development. Figure 8 below shows the logical architecture of the timesheet management system.

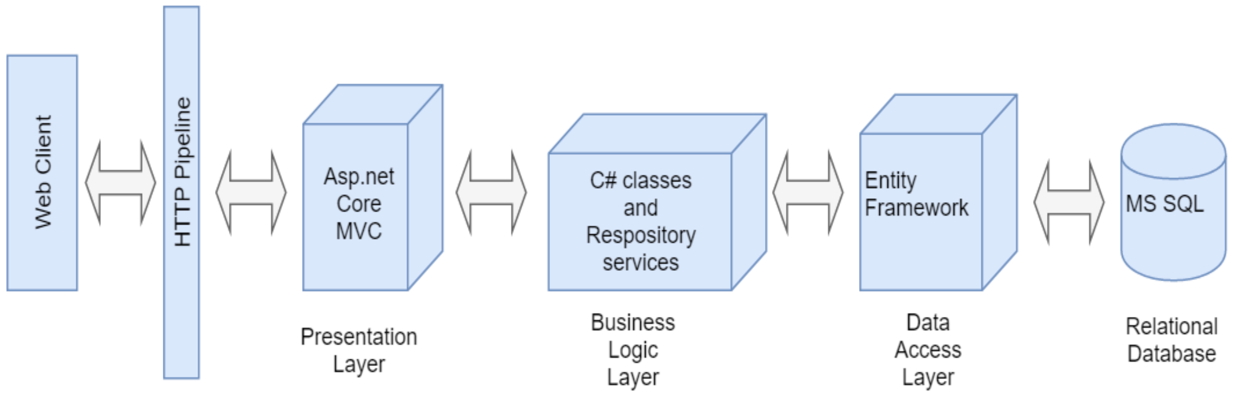


Figure 8. System Architecture

The application implements layered architecture which is one of the most widely used architecture for web application development. Using this architecture helps logically structuring the application according to its responsibilities and concerns. Since it follows separation of concern principle, even if the code base grows, the structure of the application stays organized.

In this approach, the complete logic of the application is included in a single project which is compiled to a single assembly and finally deployed as a single unit. The project contains behavior of the application, presentation, business logic, services and data access logic. In the application, users make requests through the presentation layer which interacts with the business logic layer. It calls the Data access layer for data access requests. In this way, each layer handles its own responsibility.

Figure 9 shows layered pattern used for the application which includes data access layer, business logic layer and user interface layer.

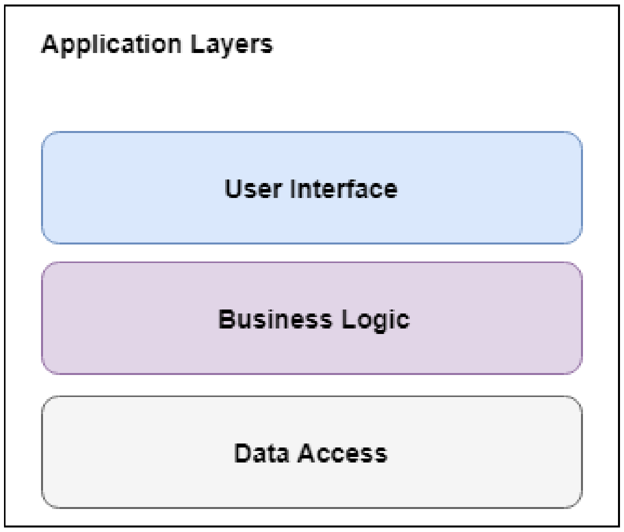


Figure 9. Logical Layering of the System

- Data Access Layer: Data access Layer contains functionality of creating, returning, updating and deleting entities in the database. Entity Framework core is used in this layer which works as object relational mapping tool to perform data access functionalities from SQL server.
- Business Logic Layer: Business logic layer contains the core functionality of the application and encapsulates the relevant business logic. In this layer, class library will be used to hold all the custom logic for application.
- Presentation Layer: Presentation Layer contains the user oriented functionality responsible for managing user interaction with the system. This layer will use MVC views to provide user interface.

System Design

Use case diagram. A use case shows an interaction between the application and the users of the application. There are three types of users in the system namely: employee, supervisor and an admin user.

Admin user is responsible for the following use cases: Login, register user, Update Projects, Update Tasks, Update Employee, Assign Projects, View Report and Logout.

Supervisor user is responsible for the following use cases: Login, Assign Projects, Record Timesheet, Request Time-Off, View Reports, Review timesheet, Review Time offs and Logout.

Employee user is responsible for the following use cases: Login, Record timesheet, Request Time-Off, View Report and Logout.

Figure 10 shows the use case diagram for the MyTime application.

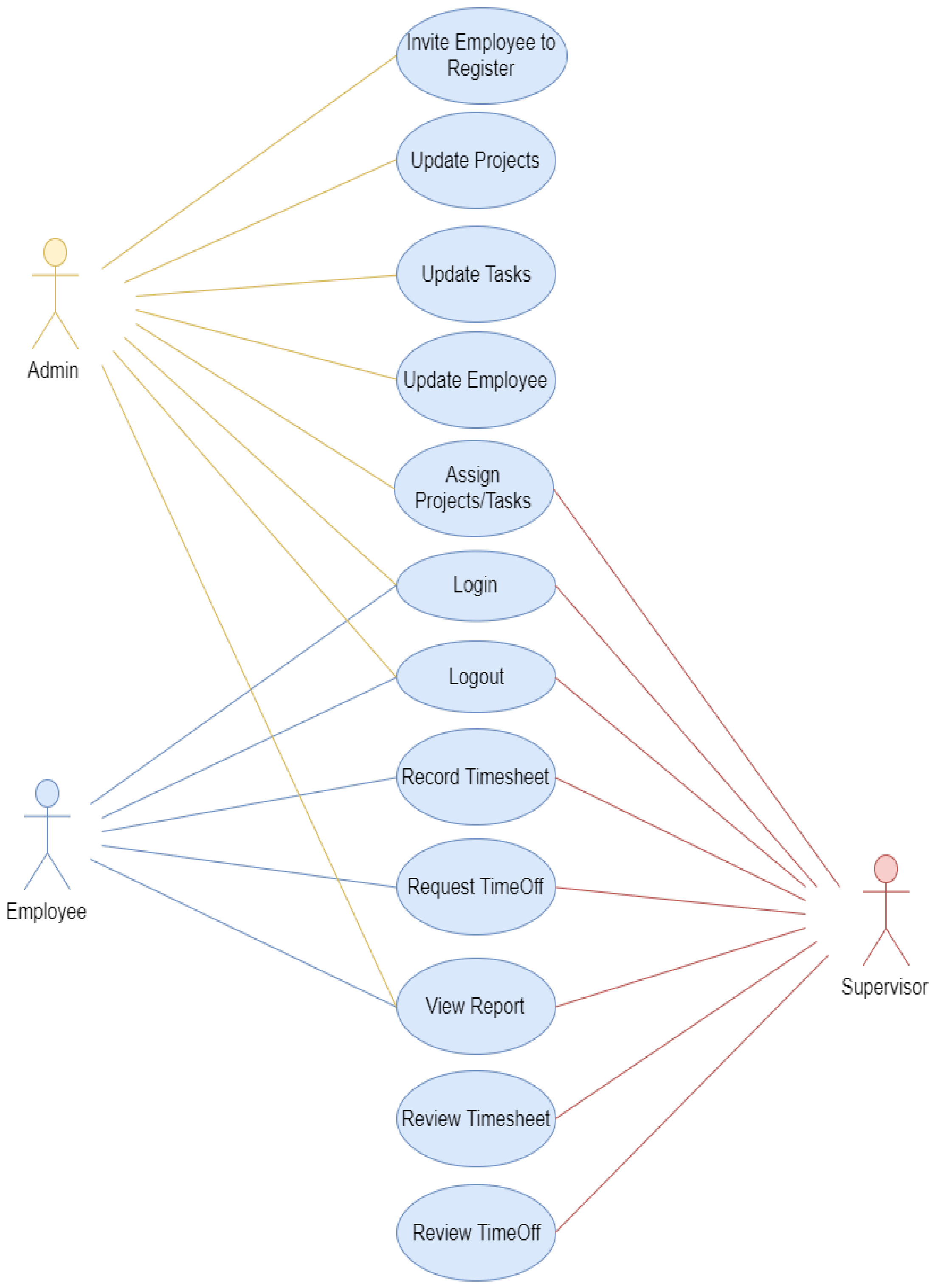


Figure 10. Use-Case Diagram for the Application

The description for the various use cases for the application is presented below.

Login Use Case:

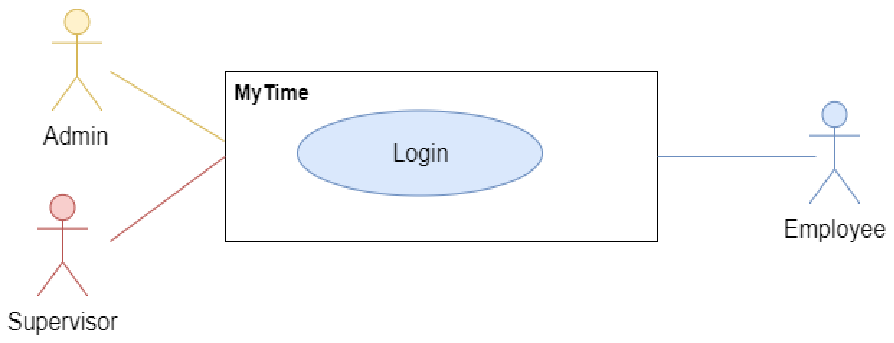


Figure 11. Login Use-Case

Table 1

Login Use Case Description for MyTime Application

Brief Description: The Login use case enable the admin/supervisor/employee users to login into the application.
Step-by-Step Description: • Enter the user credentials at the login screen—Username and Password • System validates username and password entered by the user and allows to view the screen dedicated to admin/supervisor/employee based on user role.

Assign Projects/Tasks Use Case:

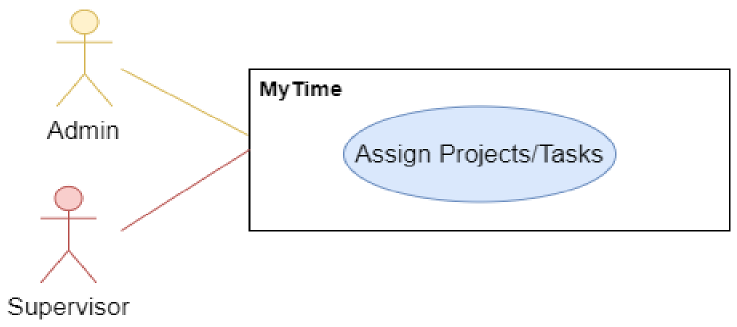


Figure 12. Assign Project/Task Use Case

Table 2

Assign Project/Task Use Case Description for MyTime Application

Brief Description: The Assign Projects/ Tasks use case enables the system's admin/supervisor user to assign project to employees. The employee user can view tasks of each project.
Step-by-Step Description: • In an Assign project screen, admin/supervisor enters the following information: Employee name and Project name • Assign a project then helps to connect a project to an employee.

Update Projects Use Case:

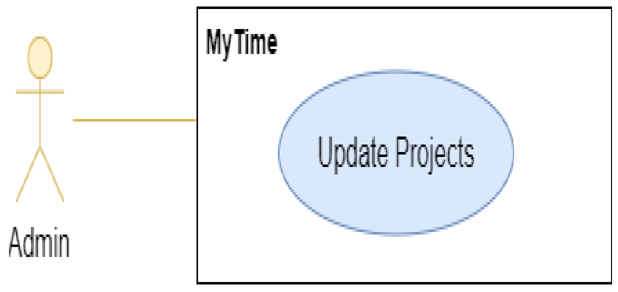


Figure 13. Update Project Use Case

Table 3

Update Project Use Case Description for MyTime Application

Brief Description: The Update Projects use case enables admin/supervisor to add and edit a project in the system.
Step-by-Step Description: • In add project screen, user enters information details of a project: ➢ Project Code ➢ Project Name ➢ Description ➢ Status ➢ Start Date ➢ End Date • Project screen has a create button which allows adding new project to the system. • Similarly, in Edit project screen, user can edit above mentioned information regarding already existing project in the system and save it.

Update Tasks Use Case:

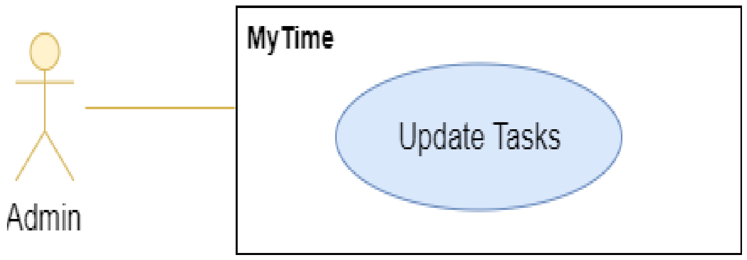


Figure 14. Update Task Use Case

Table 4

Update Task Use Case Description for MyTime Application

Brief Description:

The Update Tasks use case enables admin/supervisor user to add and edit tasks that can be used for project in the system.

Step-by-Step Description:

- In add task screen, user enters information details of a task.
  - ➢ Project ID
  - ➢ Task Code
  - ➢ Task Name
  - ➢ Description
  - ➢ Status
  - ➢ Start Date
  - ➢ End Date
- Task screen allows adding a new task to a project in the system using create option.
- Similarly, in edit task screen, user can edit above mentioned information regarding already existing task in the system and save it.

Update Employee Use Case:

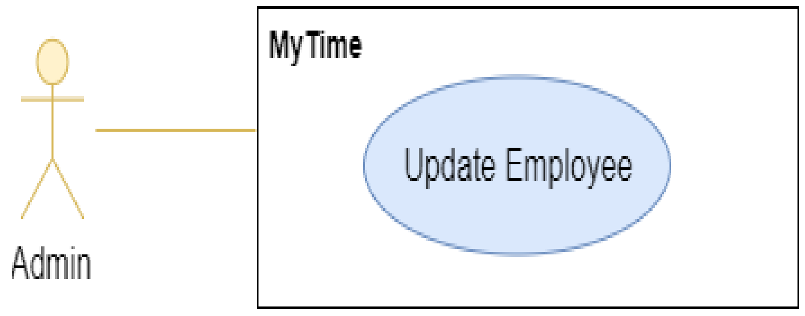


Figure 15. Update Employee Use Case

Table 5

Update Employee Use Case Description for MyTime Application

Brief Description: The Update Employee use case enables admin/supervisor user to add/update employee related information into the system.
Step-by-Step Description: • The Employee page lists all the employees on the system. The user clicks on the Edit icon next to the employee to be edited. The user can update the following information: ➢ Employee First Name ➢ Employee Last Name ➢ Supervisor Name ➢ Employment Type ➢ Department Name • User clicks on the Save button to update the employee data entered in the page.

View Report Use Case:

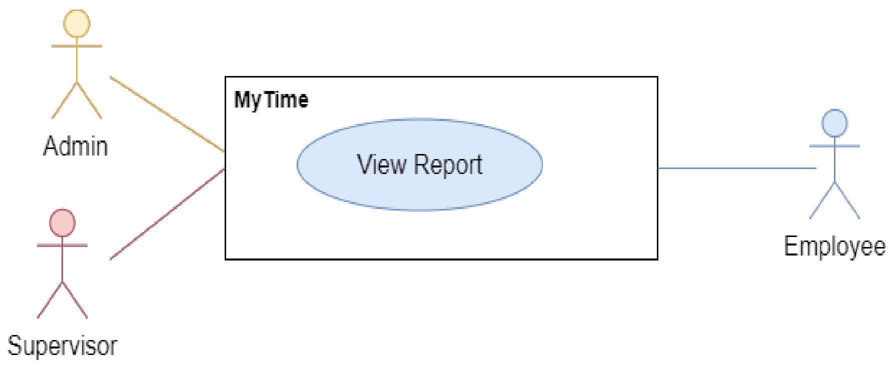


Figure 16. View Report Use Case

Table 6

View Report Use Case Description for MyTime Application

Brief Description: The View Report use case allows users to view timesheet submitted for the selected date range. The timesheet data is grouped by project and task. The Supervisor user will be able to select direct report employee from the selection criteria page and run report for each employee. The regular employee user will only be able to see report for oneself.
Step-by-Step Description: • User selects Reports link on the main menu. • Select the start and end date to run the report. The supervisor user should select one of the direct report users to view the timesheet report form. • Click on the Run button displays report on the screen. The report will be grouped by project/ task and will display all timesheets submitted during the selected date range.

Record Timesheet use case:

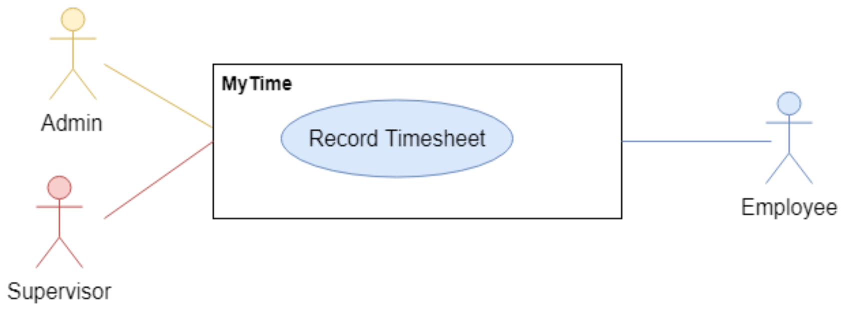


Figure 17. Record Timesheet Use Case

Table 7

Record Timesheet Use Case Description for MyTime Application

Brief Description: The Record Timesheet use case enables employee user of the Timesheet system to record and submit their timesheet. Employees can log time spent against a project and task.
Step-by-Step Description: • Display the Day View which allows user to enter project, task, and hours spent information. • The Project dropdown provides user with all the projects he is currently involved. • The Task dropdown provide user with all the tasks for the selected project. • User enters hours spent on the task for each project. • User selects project, then task, and can enter hours required to finish the task. • After all the information has been entered, user can click submit to log the hours spent on task/project for the day. • User can navigate back and forth and submit or vie timesheet submitted. User can go back seven days from current data to submit or append timesheet entries.

Request Time-off Use Case:

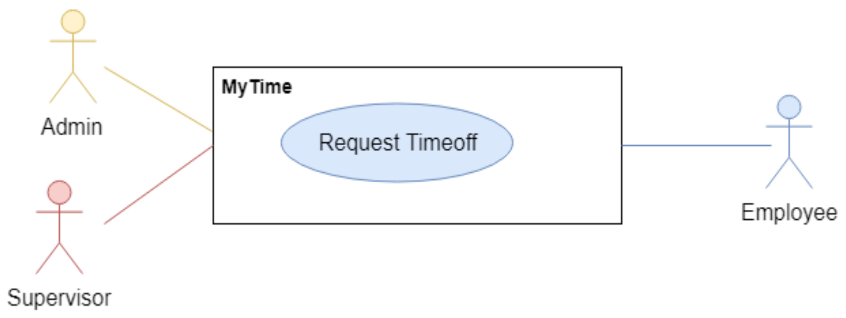


Figure 18. Request Time Off Use Case

Table 8

Request Time Off Use Case Description for MyTime Application

Brief Description: The Request Time-Off use case enables employee user of the Timesheet system to request time off. Time Off page lists all the previous future time off made by the employee.
Step-by-Step Description: • User fill out following information: ➢ Time Off Category ➢ Start Date ➢ End Date ➢ Number of Hours ➢ Notes • The time off category dropdown lists all time off categories. The start date and end date fields provide date time picker control to select proper data. • After all the information has been entered, user can click submit to request time off. User can also view status of the requested timesheet as it gets reviewed by the supervisor.

Review Timesheet Use Case:

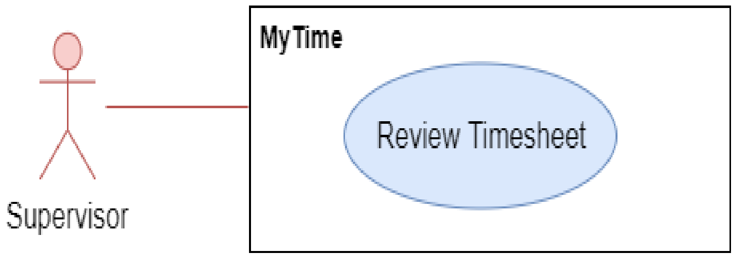


Figure 19. Review Timesheet Use Case

Table 9

Review Timesheet Use Case Description for MyTime Application.

Brief Description: The Review Timesheet use case enables supervisor user of the Timesheet system to review timesheet requested by their direct report employees. Review Timesheet screen provides a list of all employees reporting to a supervisor.
Step-by-Step Description: • The user can select an employee to view the timesheet submitted by the employee. • The data range selector control will show the current week by default. User can go to previous or next week by clicking the forward or backward arrow on the data range control. • The user can approve the timesheet by clicking on the Accept link of timesheet entries. • The user can reject timesheets that need to be rejected by clicking on the Reject link on the list of timesheet entries grid.

Review Time Off Use Case:

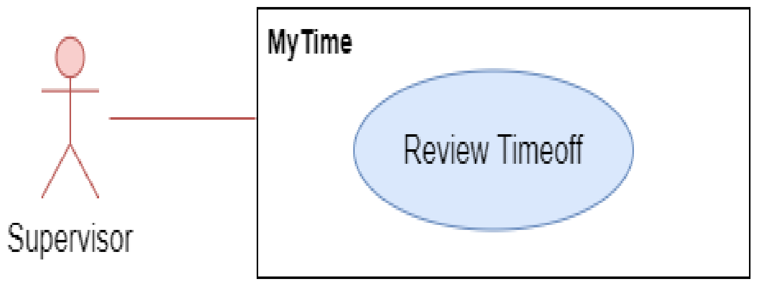


Figure 20. Review Time Off Use Case

Table 10

Review Time Off Use Case Description for MyTime Application

Brief Description: The Review Time Off use case enables supervisor user of the Timesheet system to review time offs requested by their direct report employees. Review Time-Off page provides a list of all employees reporting to the supervisor user.
Step-by-Step Description: • The user can select an employee to view the time off request submitted by the employee. • The data range selector control will use the current week by default. User can go to previous or next week by clicking the forward or backward arrow on the date range control. • The user can approve the Time Off entries by clicking on the Accept link on the list. • The user can reject Time Offs that need to be rejected by clicking on the Reject link on the list of Time Off entries.

Register Employee Use Case:

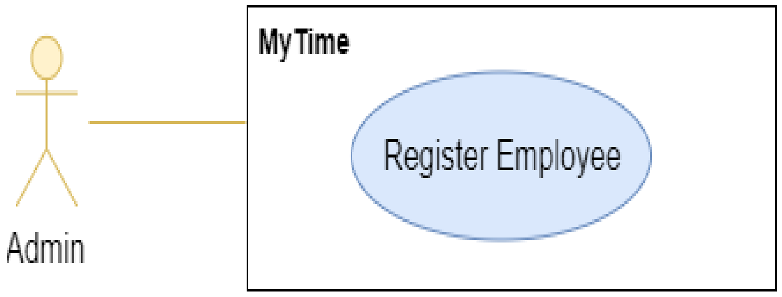


Figure 21. Register Employee Use Case

Table 11

Register Employee Use Case Description for MyTime Application

Brief Description: The Register Employee use case enables Admin user to register new employee user into the Timesheet system.

Step-by-Step Description:
 	
• In a Register screen, admin user enters information details for an employee:
  ➢ First Name
  ➢ Lat Name
  ➢ Email
  ➢ Password
  ➢ User Role
• User clicks Register button and it will create an employee account in the system.

Logout Use Case:

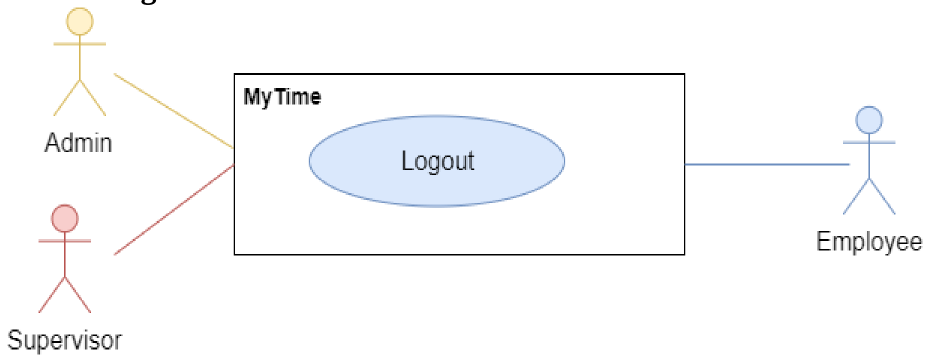


Figure 22. Logout Use Case

Table 12

Logout Use Case Description for MyTime Application

Brief Description: The Logout use case enables the system's admin/supervisor/employee user to log out from the system.
Step-by-Step Description: • User clicks on the Logout button. • System clears the user session data and logs out of the system.

Sequence diagram. A sequence diagram depicts the realization of a specific scenario of the use case.

The sequence diagram for the realization of the login scenario:

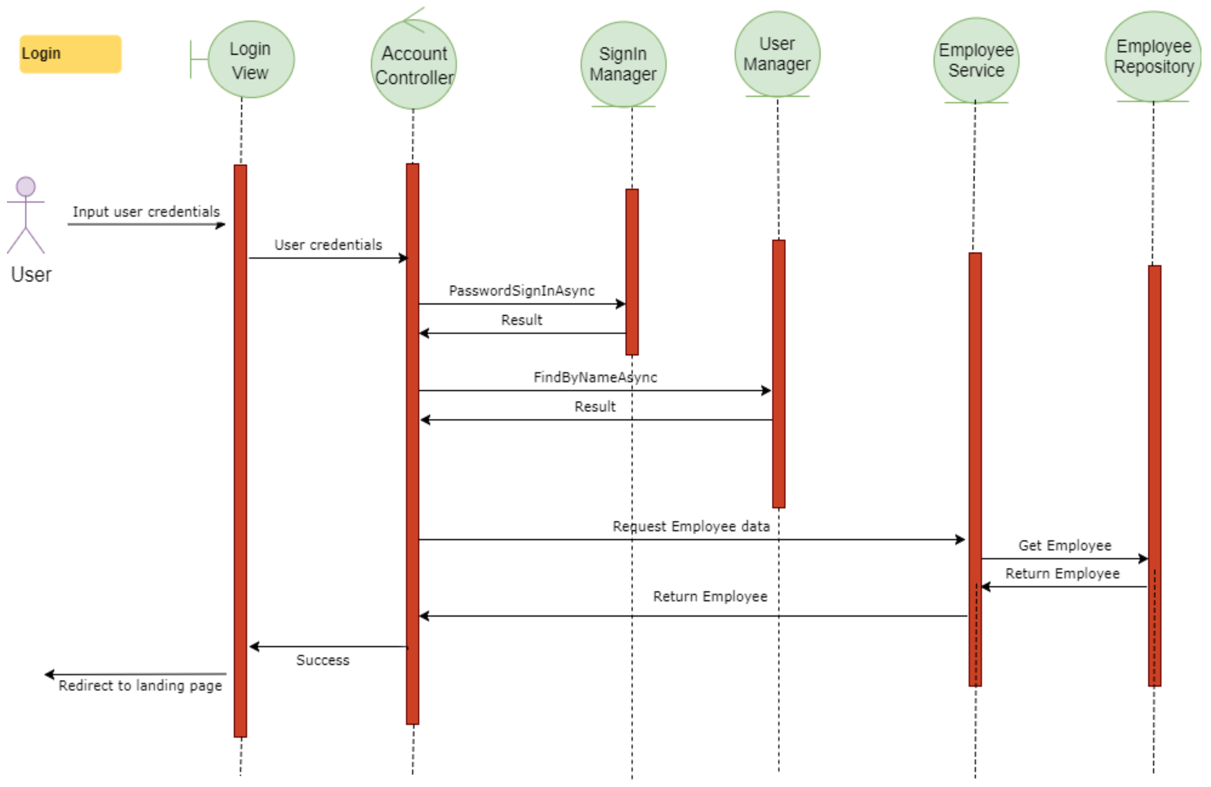


Figure 23. Sequence Diagram for Login Scenario

The sequence diagram for realization of the employee register scenario:

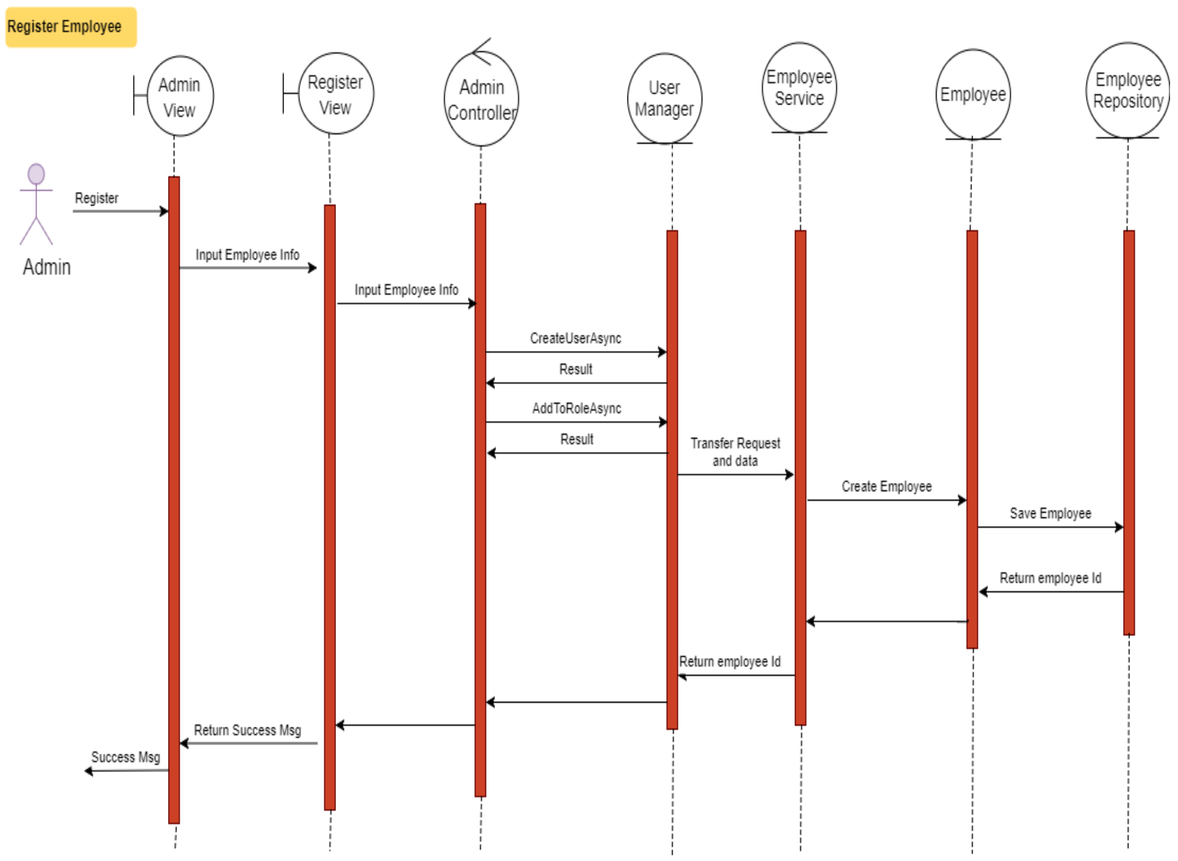


Figure 24. Sequence Diagram for Employee Registration Scenario

The sequence diagram for realization of the record timesheet scenario:

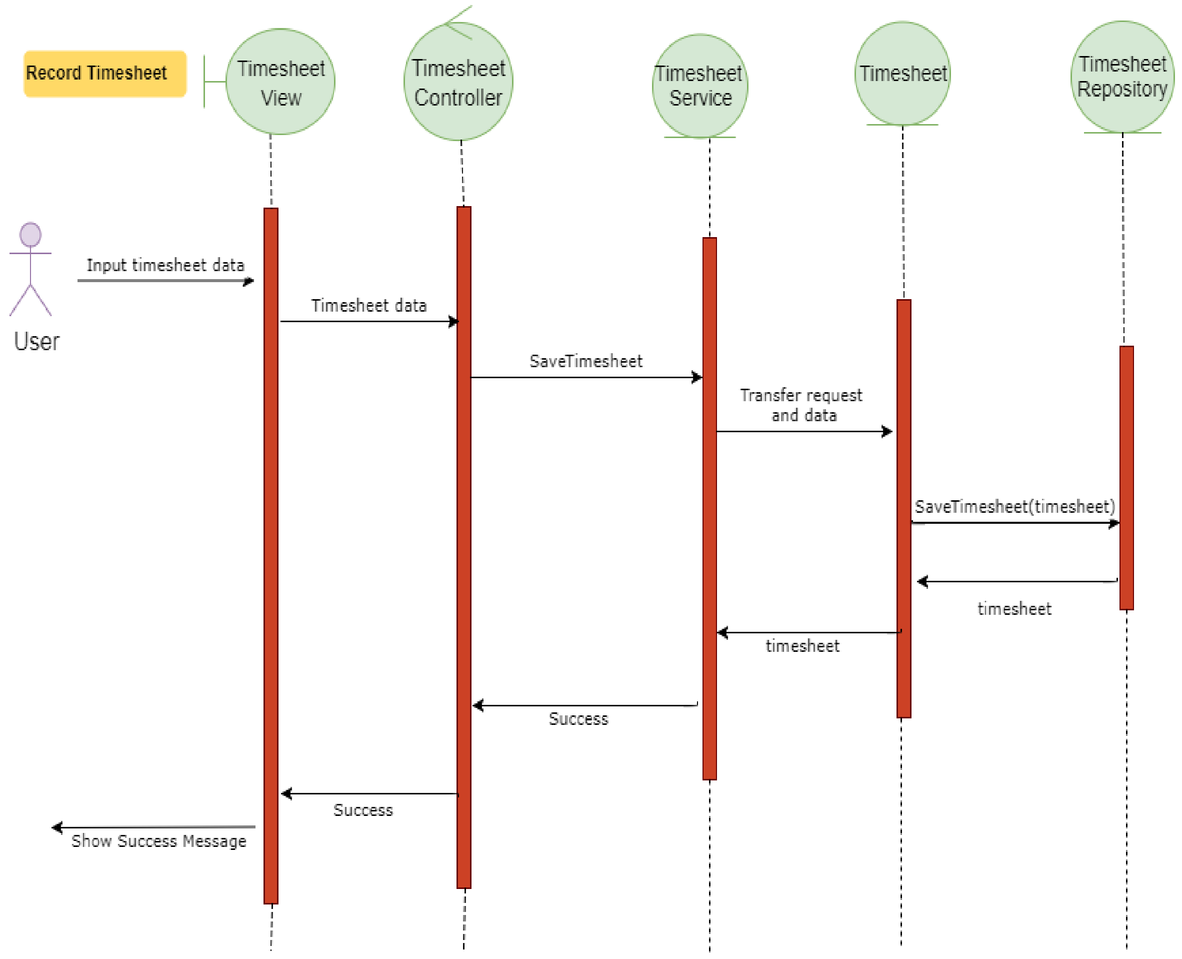


Figure 25. Sequence Diagram for Record Timesheet Scenario.

The sequence diagram for realization of the Review timesheet scenario:

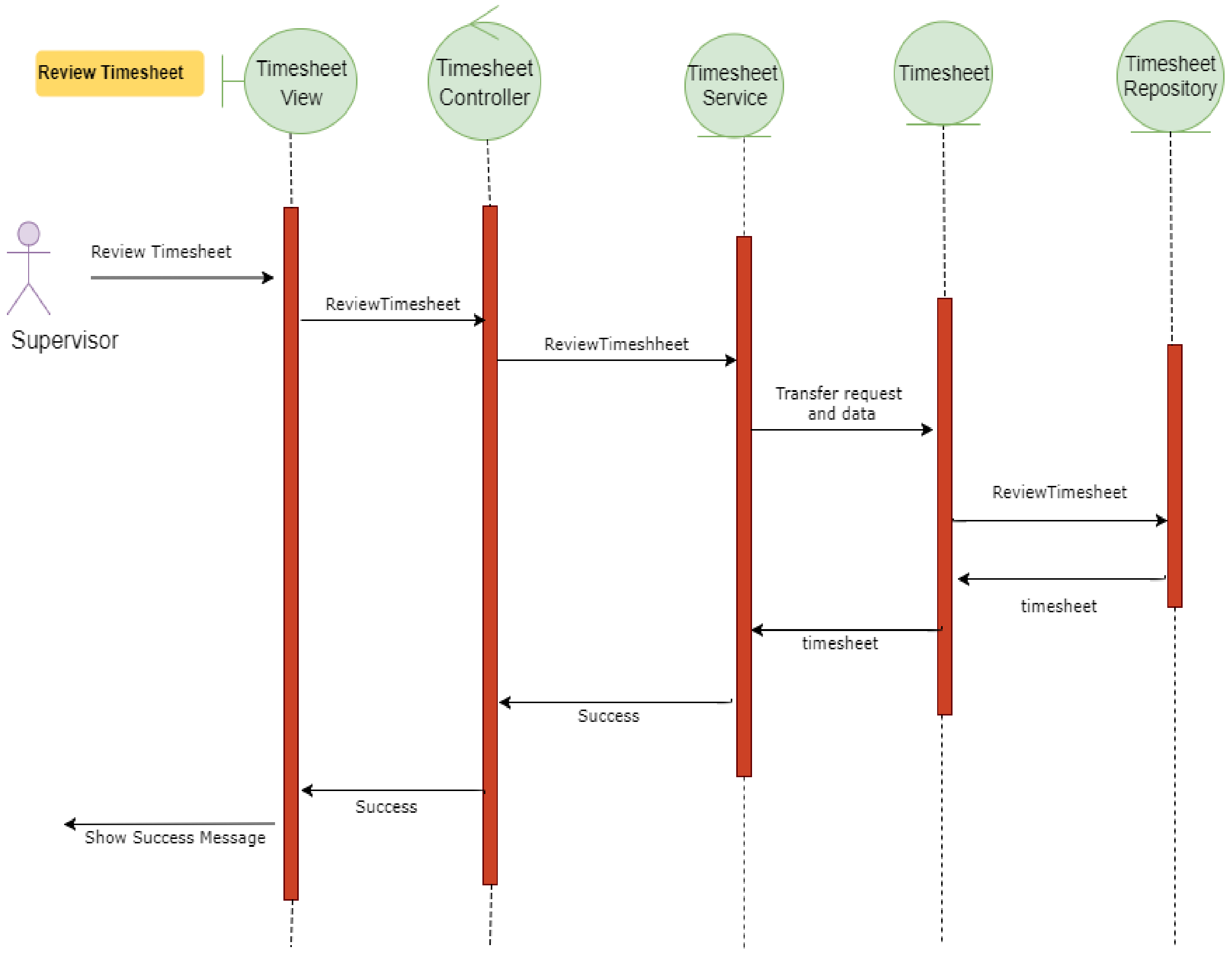


Figure 26. Sequence Diagram for Review Timesheet Scenario

The sequence diagram for realization of the request time off scenario:

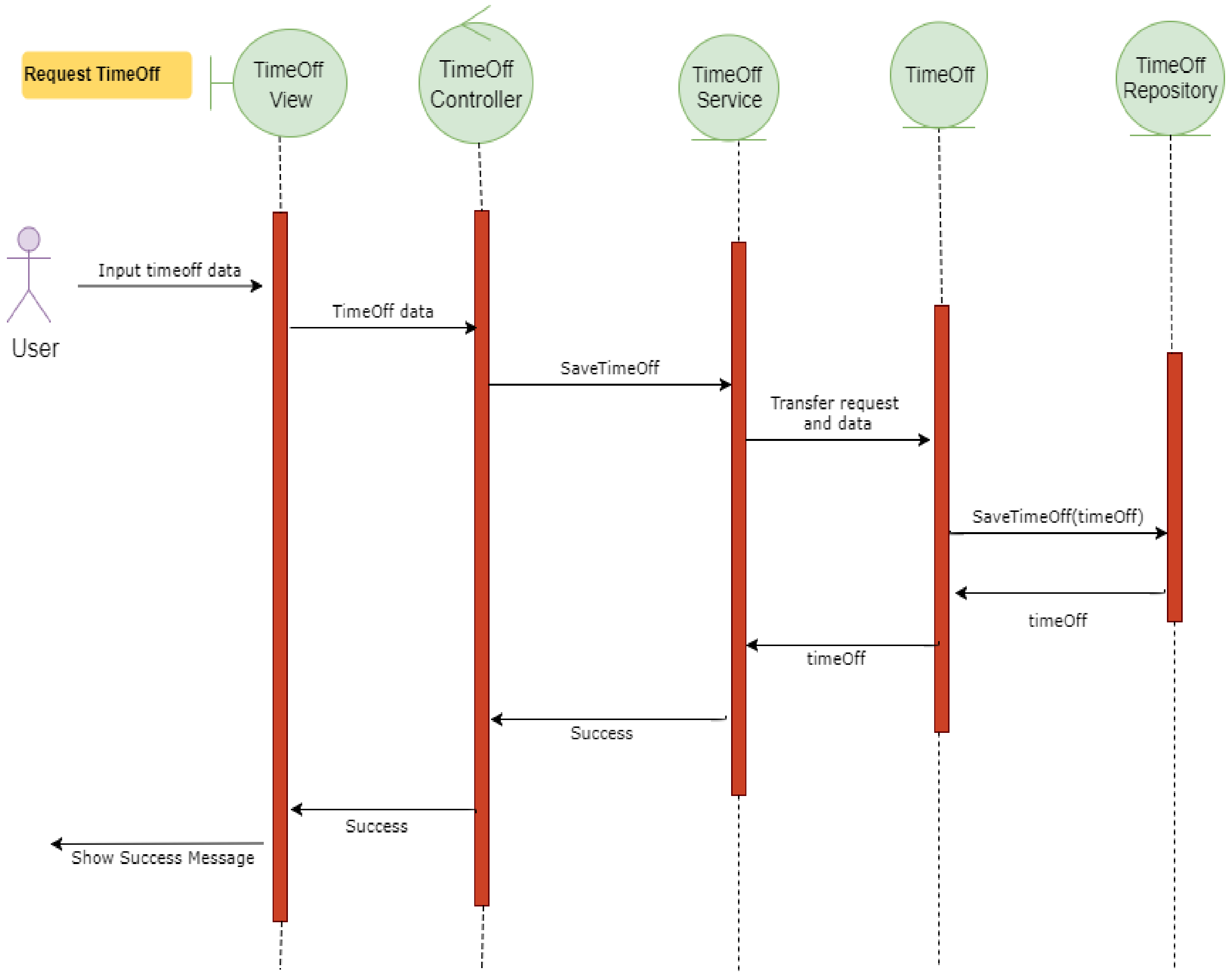


Figure 27. Sequence Diagram for Request Time Off Scenario

The sequence diagram for realization of the review time off scenario:

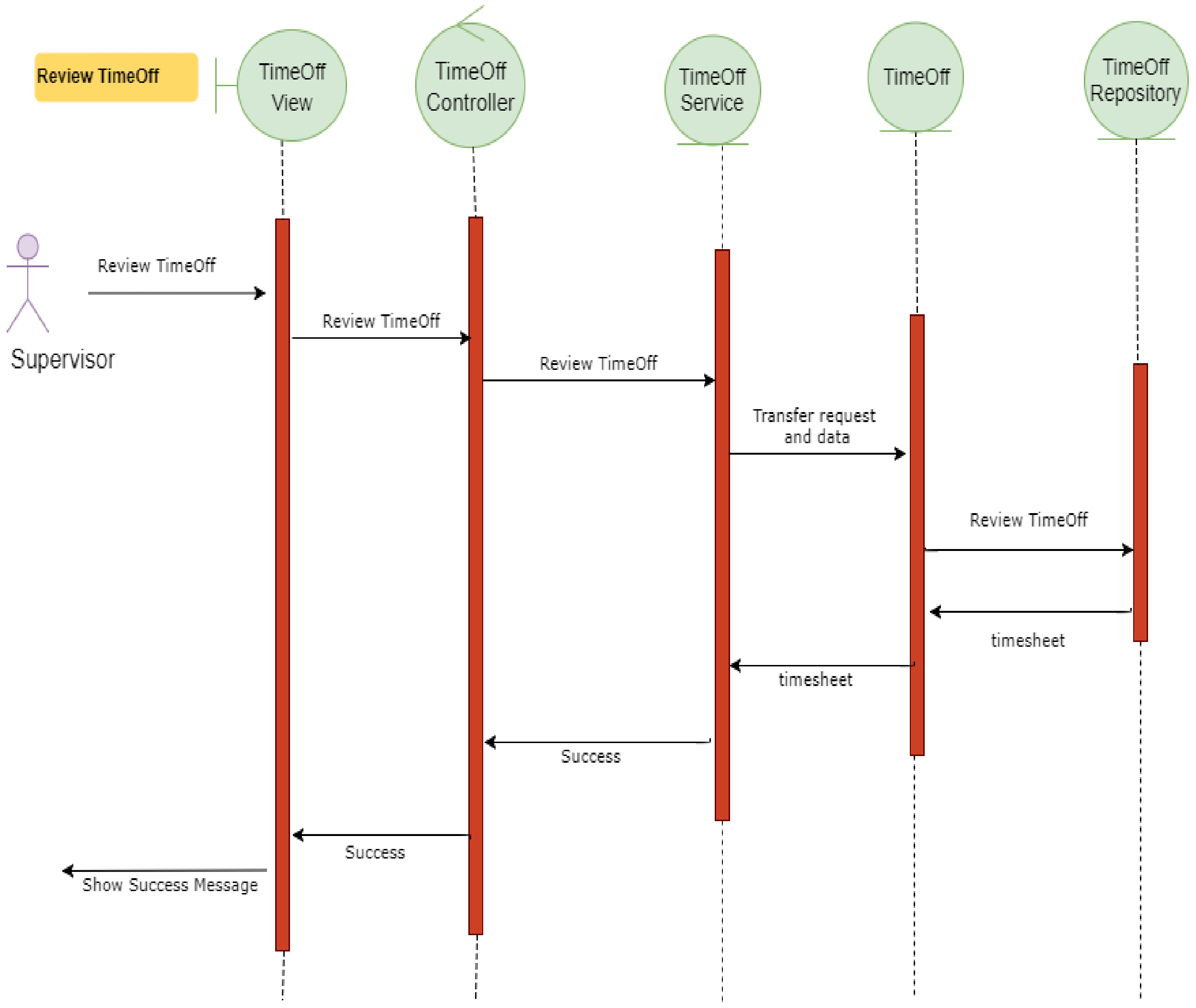


Figure 28. Sequence Diagram for Review Time Off Scenario

Database Design

The entity relationship diagram given below represents the database tables defined for MyTime application.

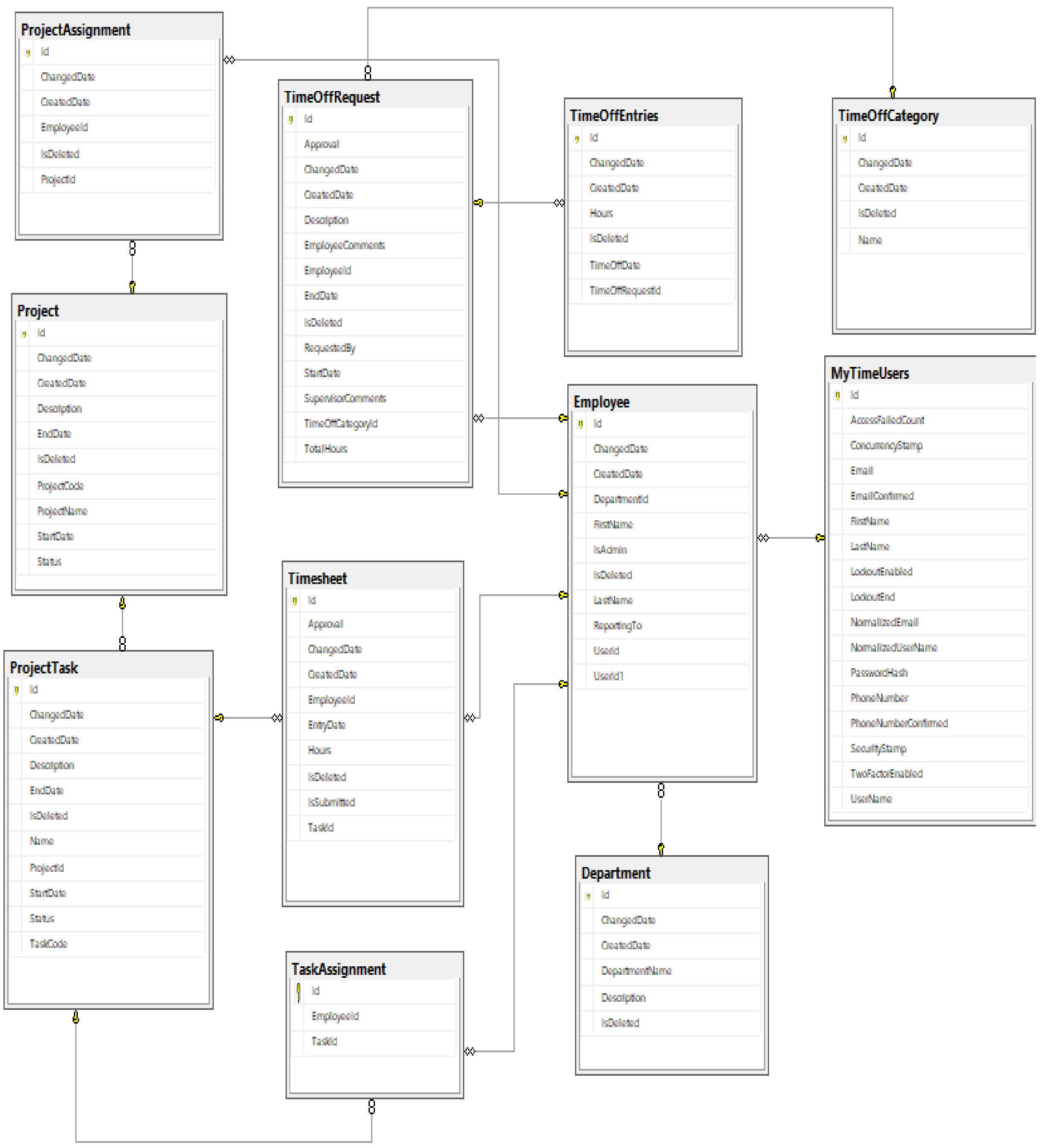


Figure 29. Entity Relationship Diagram for MyTime Application.

Chapter IV: ASP.NET Core Application Deployment

ASP.NET Core Application Deployment

With classical ASP.NET applications, the only option to deploy is on a Windows based system with Internet Information Services (IIS) as web server. The application pool hosts the ASP.NET application and the application is instantiated by built-in ASP.NET hosting features in IIS (Strahl, 2016). An application pool can host one or more applications as shown in Figure 30. In this type of hosting scenario, IIS remains tightly coupled to the application, calling into the specific exposed methods for different stages of the request.

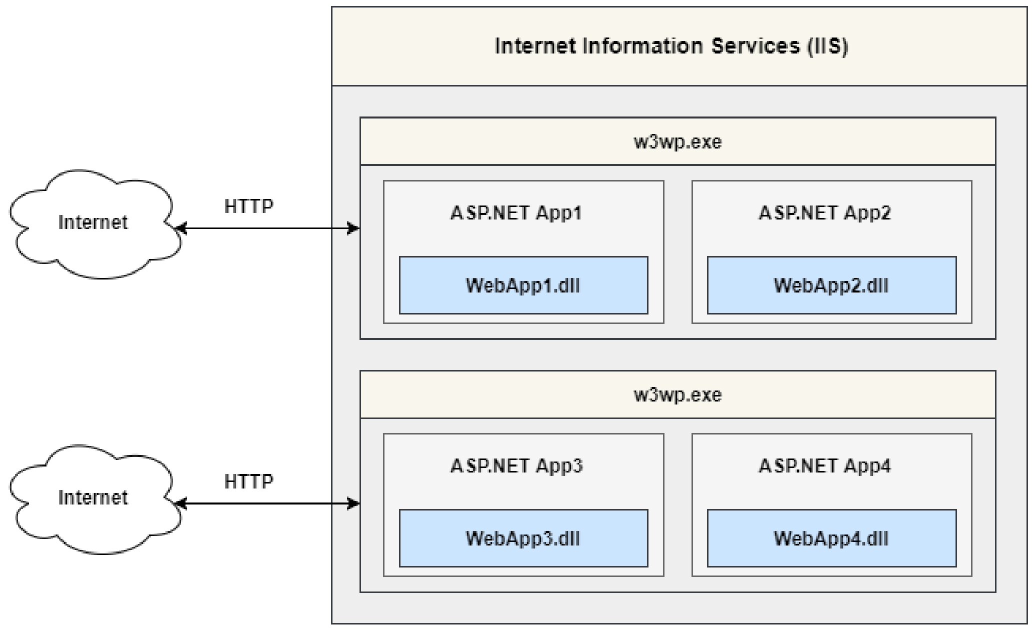


Figure 30. ASP.NET Application Hosting Using IIS Server

ASP.NET Core Application is a standalone console application that is invoked through the “dotnet” runtime command. It does not load into the IIS worker process but loads in a native IIS module called AspNetCoreModule which runs the console application.

With ASP.NET Core there are more options on how to build and deploy the application. There are three choices to be made; which framework to target, which server to host with and how to expose the server to the internet. ASP.NET Core application can run on full .NET framework or .NET Core, using .NET Core provides the ability to run the application cross platform between Windows, Mac and Linux.

ASP.NET Core ships with two server implementations; Kestrel and Web listener. Kestrel is cross-platform web server whereas Web listener runs only on Windows. Microsoft does not recommend exposing Kestrel to the public internet directly. Kestrel works great for serving dynamic content from ASP.Net Core, however, it is not designed to provide features of full server like SSL authentication, compression of static content, static file caching, URL rewriting, etc. which other servers such as IIS, Apache or Nginx provide.

Ways of Deploying .NET Core Application

There are two ways of deploying .NET Core application: Framework-Dependent Deployment and Self-Contained Deployment.

Framework-Dependent Deployment (FDD): This type of deployment only contains the application specific files and assemblies and relies on the presence of shared system-wide version of .NET Core on the target system. The deployment package contains .dll files that can be launched by dotnet utility from the command prompt.

Self-Contained Deployment (SCD): This type if deployment contains the application specific files and assemblies including the.NET Core libraries and runtime components. The deployment package includes an executable which is a renamed version of the platform specific .NET Core host and a .dll file which is the actual application.

The MyTime app will be deployed as a framework-dependent package.

Deploying MyTime Application in Windows Platform Using IIS

To host the ASP.NET Core application into IIS, .NET Core Hosting Bundle has to be installed. The bundle consists of the .NET Core Runtime, .NET Core Library and the ASP.NET Core Module.

The AspNetCore module creates the reverse proxy between IIS and the Kestrel server. The ASPNetCoreModule is a native IIS module that hooks into the IIS pipeline and redirects all traffic to the backend ASP.NET Core application (Strahl, Ross, & Dykstra, 2017). This way all requests bypass the IIS pipeline and are forwarded to the ASP.NET Core process. Figure 31 shows an example of hosting of application using Kestrel web server and IIS reverse proxy (Strahl, 2016).

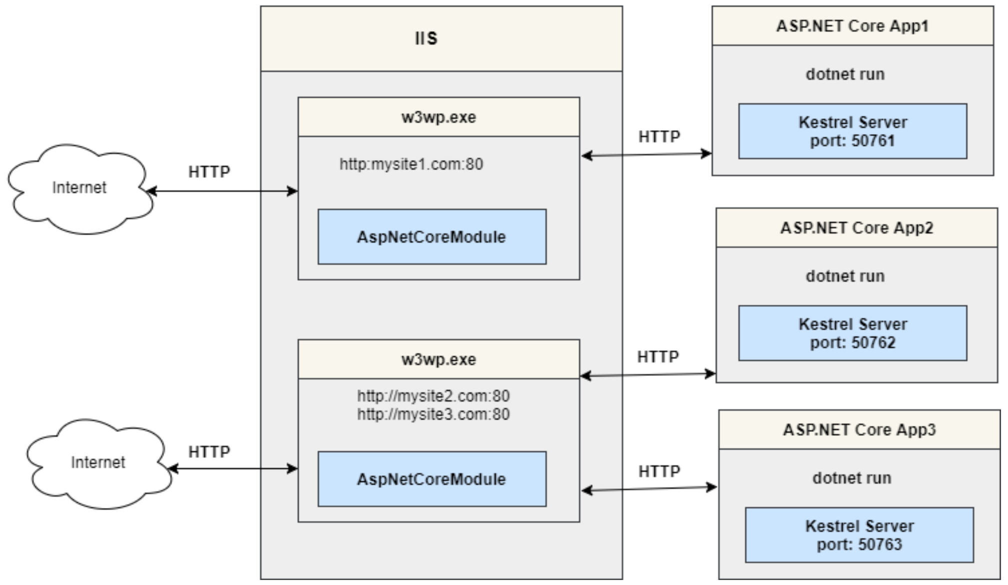
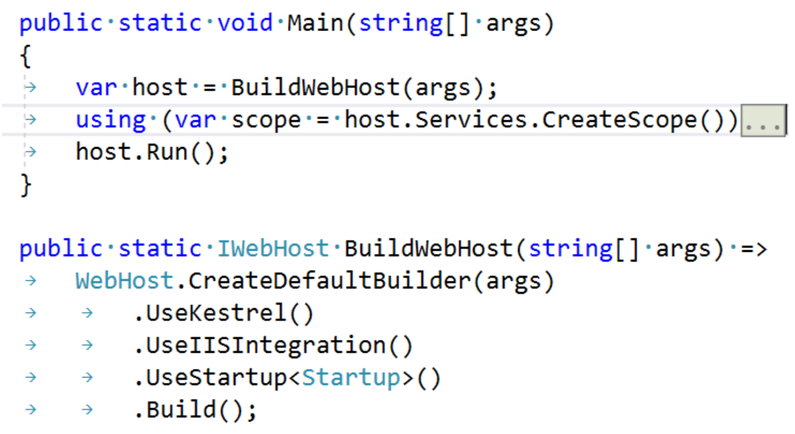


Figure 31. Application Hosting Using Kestrel and IIS as Reverse Proxy

When a HTTP requests come in from a Web client, IIS forward the request to the ASP.NET Core application on the HTTP port configured for the application. In the example shown above, IIS is hosting three ASP.NET Core applications, two of which are running one IIS process (w3wp.exe) and the third application is running on another IIS process by itself. In this scenario, IIS acts a reverse proxy simply forwarding requests to the ASP.NET Core web application running the Kestrel Web server on a different port. Kestrel picks up the request and pushes it into the ASP.NET Core middleware pipeline which then handles the request and passes it on to the application logic.

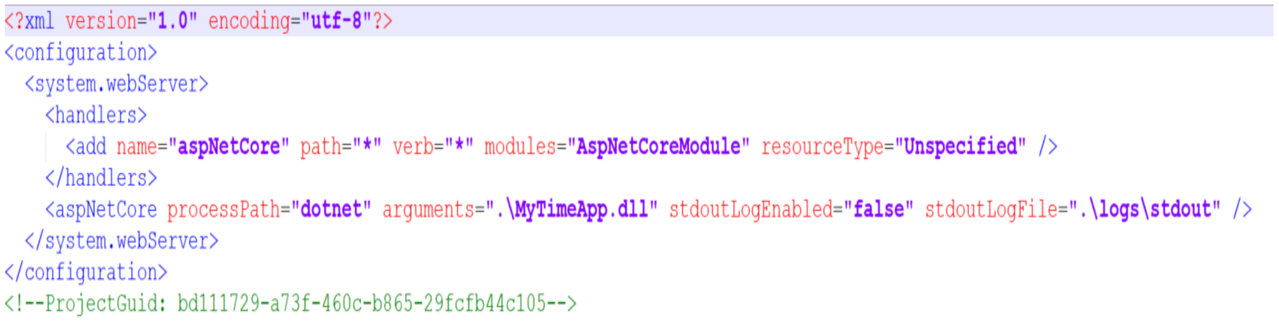
Kestrel then starts the application through the Main () method present in the application. The Main () method present in Program.cs is the starting point in .NET Core application which then builds and runs WebHost builder. Figure 32 provides a code snippet of using Build and Run methods which builds IWebHost object that hosts the app and start listening for incoming HTTP requests. The request is then pushed to the *Configure()* method of the Startup class. The Configure method is where the application's request processing pipeline is configured. The resulting HTTP output is then passed back to IIS which then pushes it back out over the Internet to the HTTP client that initiated the request which could be either a browser, mobile client or application.



```
public static void Main(string[] args)
{
    var host = BuildWebHost(args);
    using (var scope = host.Services.CreateScope())...
    host.Run();
}

public static IWebHost BuildWebHost(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseKestrel()
        .UseIISIntegration()
        .UseStartup<Startup>()
        .Build();
```

Figure 32. Code Snippet for Build and Running Web Host from Program.cs file

The ASP.NET Core module is configured from the Web.config file found in the application's root, which points at the startup command and the argument which are used to launch the .Net core application. Figure 33 shows web configuration settings for the application.

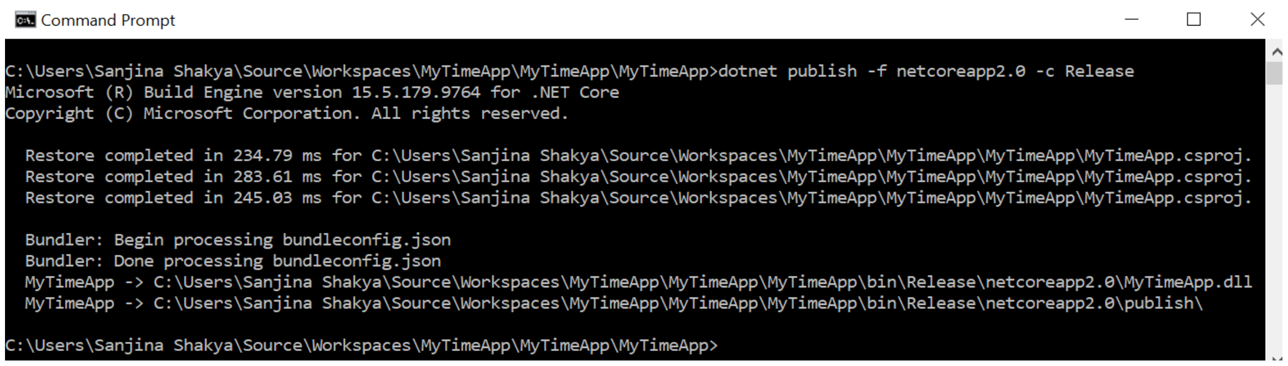


```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModule" resourceType="Unspecified" />
    </handlers>
    <aspNetCore processPath="dotnet" arguments=".\MyTimeApp.dll" stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout" />
  </system.webServer>
</configuration>
<!--ProjectGuid: bd111729-a73f-460c-b865-29fcfb44c105-->
```

Figure 33. Web Configuration Setting for MyTime Application

The *dotnet publish* command compiles the application code and copies the necessary files into a publish folder. The publish folder contains .exe and/or .dll files along with the configuration files, static content and MVC views. Passing -c switch to the *dotnet publish* command allows to pass the Release build configuration, the following command will create framework dependent package under "/bin/Release/netcoreapp2.0/publish" folder. Figure 34 shows use of publish command to publish the application.

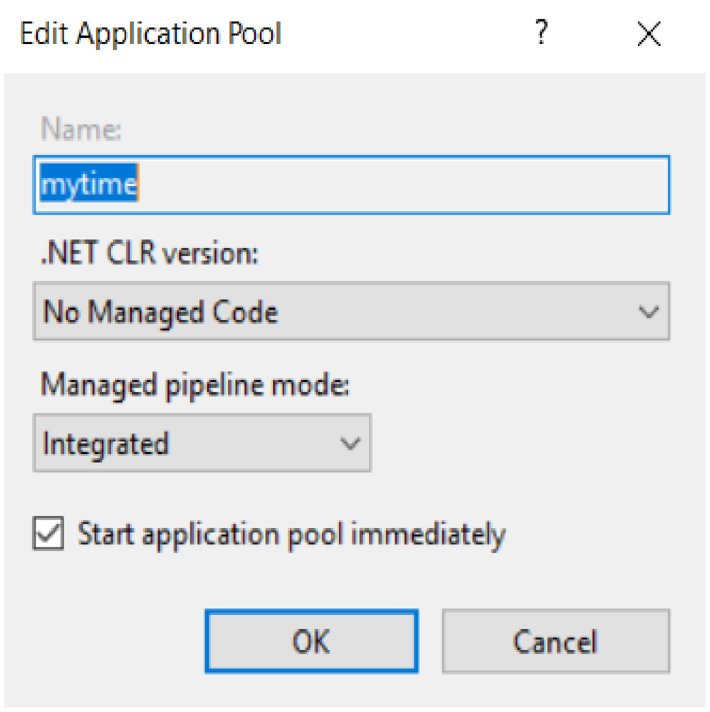
>dotnet publish -f netcoreapp2.0 -c Release



```
C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp>dotnet publish -f netcoreapp2.0 -c Release
Microsoft (R) Build Engine version 15.5.179.9764 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 234.79 ms for C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp\MyTimeApp.csproj.
  Restore completed in 283.61 ms for C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp\MyTimeApp.csproj.
  Restore completed in 245.03 ms for C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp\MyTimeApp.csproj.

  Bundler: Begin processing bundleconfig.json
  Bundler: Done processing bundleconfig.json
  MyTimeApp -> C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp\bin\Release\netcoreapp2.0\MyTimeApp.dll
  MyTimeApp -> C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp\bin\Release\netcoreapp2.0\publish\

C:\Users\Sanjina Shakya\Source\Workspaces\MyTimeApp\MyTimeApp\MyTimeApp>
```

Figure 34. Running Publish Command

The next step is to create a website on IIS, which will also create an app pool with the same name. The contents of the published folder can be copied to any folder and the folder can used as physical path for the site. Since the app pool acts only as a proxy to forward requests, it should be set to use No Managed Code as shown in Figure 35. This completes the application publish process in IIS.


Edit Application Pool

Name: mytime

.NET CLR version: No Managed Code

Managed pipeline mode: Integrated

☑ Start application pool immediately

Figure 35. Application Pool Setting for MyTime Application

Deploying Application in Linux-Based System—Ubuntu Using Nginx

Unlike the previous versions of ASP.NET, the ASP.NET Core applications can be run directly on a Linux system. This is great option for web applications that needs to be deployed on a Linux based server or other Linux based cloud platforms such as AWS.

Install .Net Core in Ubuntu: To be able to run ASP.NET core application, the .NET Core runtime package has to be installed on the Linux system. .NET Core 2.0 is supported on most of the recent versions of popular Linux distributions such as Red Hat Enterprise 7.0, CentOS 7, Fedora 26, Debian 8.7, Ubuntu 17.04/16.04, openSUSE 42.2, etc. The MyTime application is deployed in a VMware workstation running Ubuntu 16.04 LTS.

- The first step is to setup the host package feed with following command.
  $ sudo sh -c 'echo “deb [arch=amd64] https://packages.microsoft.com/repos/microsoft-ubuntu-xenial-prod xenial main”>/etc/apt/sources.list.d/dotnetdev.list’
- The second step is to run apt-get update command which downloads the package lists from the repositories and updates them to get information on the newest versions of packages and their dependencies. $ sudo apt-get update
- The next step is to install dot net core using the following apt-get install command.
  $ sudo apt-get install dotnet-sdk-2.0.0

These set of commands installs dotnet core 2.0.0 into the Ubuntu system.

Kestrel self-hosting: The application is self-hosted in the Kestrel server. Kestrel works great for serving dynamic content from ASP.Net Core, however, it is not designed to provide features of full server like SSL authentication, compression of static content, static file caching, URL rewriting, etc. which other servers such as IIS, Apache or Nginx would provide (Shirhatti, 2017). Nginx will be used which will work as reverse-proxy to the Kestrel server. Figure 36 shows use of Nginx as reverse proxy to the Kestrel server which is used for self-hosting ASP.NET Core application.

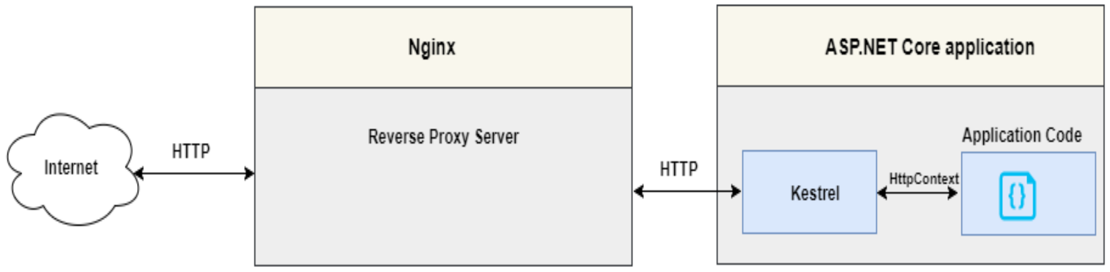
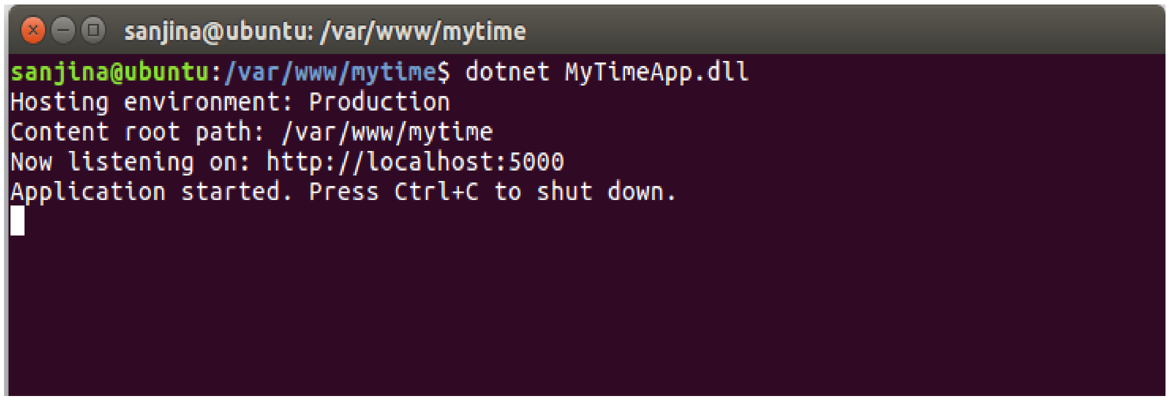


Figure 36. Hosting MyTime Application using Nginx as Reverse Proxy

The application is published from the Windows environment using dotnet publish command and the published binaries files are copied over to /var/www/mytime folder in a Linux machine. The application can be run using dotnet command. This will start the Kestrel server and will start listening on localhost:5000.

$ dotnet MyTimeApp.dll



```
sanjina@ubuntu:/var/www/mytime$ dotnet MyTimeApp.dll
Hosting environment: Production
Content root path: /var/www/mytime
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

Figure 37. Running MyTime Application in Ubuntu System

Installing Nginx and Setting Configuration: Microsoft recommends using reverse proxy in front of Kestrel server in order to make the application available on port 80. Nginx server is used as reverse proxy to host MyTime application. Nginx is available in Ubuntu’s default repositories, so it can be installed by running the apt-get commands below.

$ sudo apt-get update

$ sudo apt-get install nginx

This command installs Nginx and required dependencies. The Nginx server can be started using service start command below.

```
$ sudo service nginx start
```

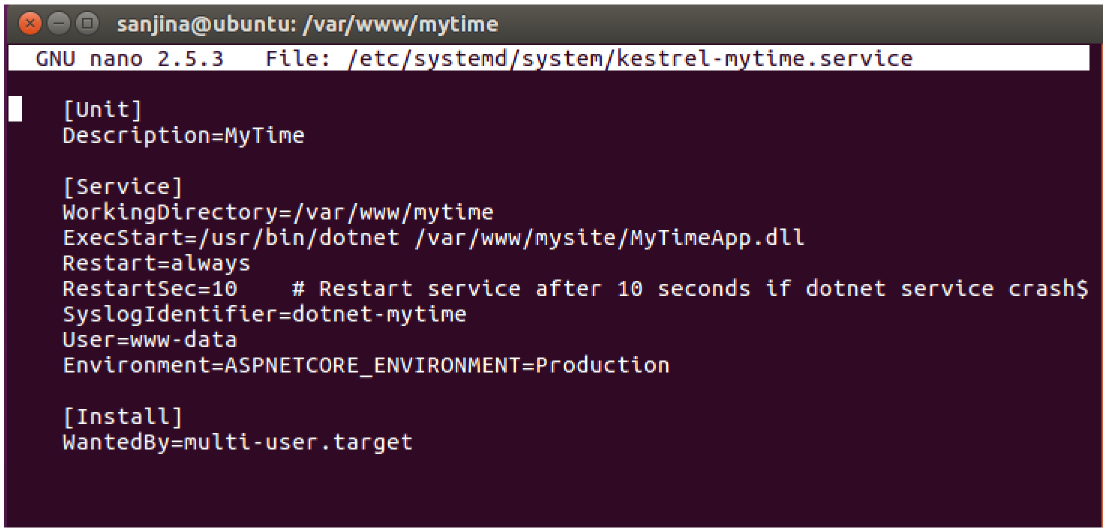
Here are some configuration changes needed in order to configure nginx to serve as reverse-proxy to the application. This configuration will make Nginx to start listening on port 80 and proxy the incoming request to Kestrel running at localhost:5000.

```
$ sudo nano /etc/nginx/conf.d/default.conf
server {
        listen 80;
         location /
         {
         proxy_pass http://localhost:5000;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection keep-alive;
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
        }}
```

Start Kestrel Automatically: In order to control the web application service and have it start automatically when the machine boots, a daemon service named kestrel-mytime.service was created. In this service, the WorkingDirectory property should hold the location of the folder where the application is published and the ExecStart property should provide dotnet command

and the name of the web app dll. The configuration required for the service is shown in Figure 38 below:

$ sudo nano /etc/systemd/system/kestrel-mytime.service



```
sanjina@ubuntu: /var/www/mytime
  GNU nano 2.5.3    File: /etc/systemd/system/kestrel-mytime.service

    [Unit]
    Description=MyTime

    [Service]
    WorkingDirectory=/var/www/mytime
    ExecStart=/usr/bin/dotnet /var/www/mysite/MyTimeApp.dll
    Restart=always
    RestartSec=10    # Restart service after 10 seconds if dotnet service crash$
    SyslogIdentifier=dotnet-mytime
    User=www-data
    Environment=ASPNETCORE_ENVIRONMENT=Production

    [Install]
    WantedBy=multi-user.target
```

Figure 38. Nginx Configuration

The service can be enabled using systemctl enable command as shown below.

$ sudo systemctl enable kestrel-mytime.service

It can be manually started and status can be checked using following commands.

$ sudo systemctl start kestrel-mytime.service

$ sudo systemctl status kestrel-mytime.service

Application User Interface and Screen Shots

This section presents some of the screen shots from the MyTime Application which shows different functionalities the application offers.

The user has to be logged in with proper credentials to use the application. An admin user will register new employee for the first time into the system with proper roles. Here are some of the important pages with screenshots and short description.

Login screen: The Login page shown in Figure 39 has input fields for username and password. Both fields are required fields and after user enter credentials and clicks on Login button, application will validate the user credentials and will take to a Day view page with menu options based on user's role.

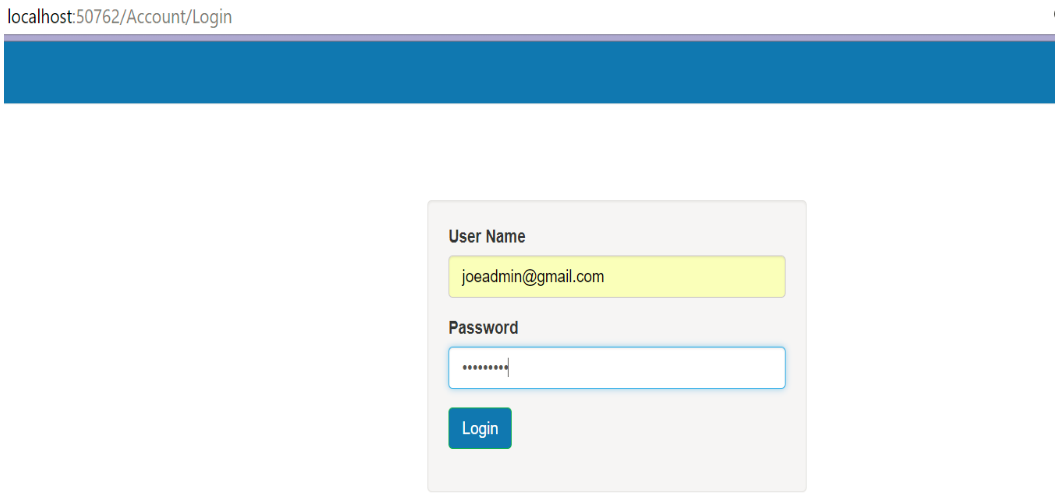
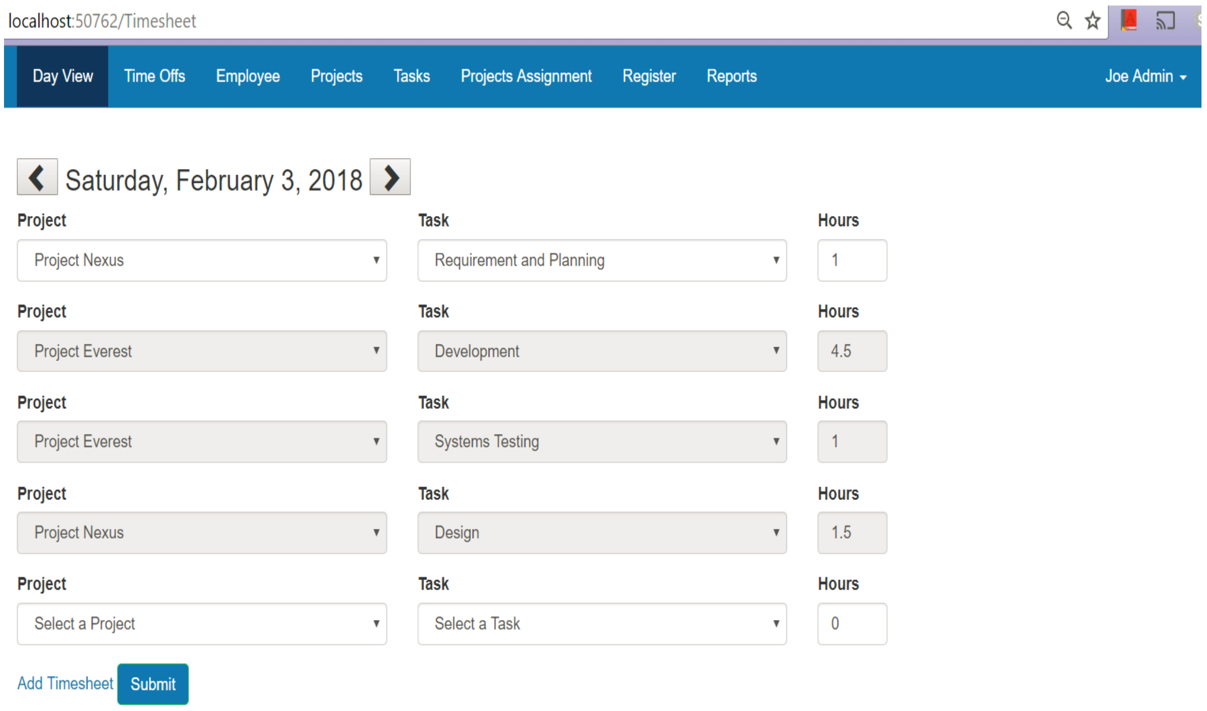


Figure 39. Login Page for MyTime Application

Day view screen: The day view page provides user to enter and submit time entries. Users can navigate to previous or next day by clicking on the left or right arrow buttons. The day view contains dropdowns to select a project and its task and a textbox field to enter number of hours. The time entries that have already been approved cannot be edited. Users can go back to seven days and edit and submit their timesheet. User can add more time entries by clicking on Add Timesheet link.



localhost:50762/Timesheet

Day View | Time Offs | Employee | Projects | Tasks | Projects Assignment | Register | Reports | Joe Admin

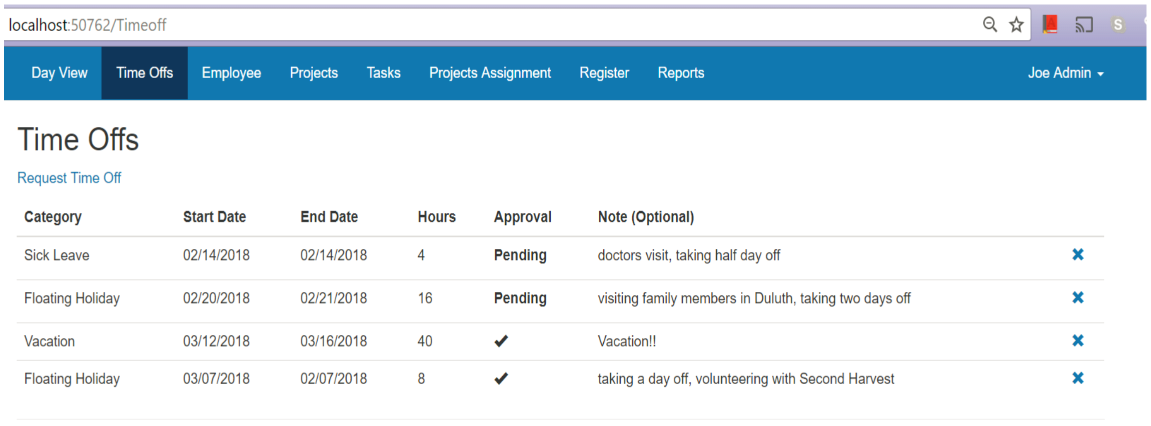
Saturday, February 3, 2018

Project	Task	Hours
Project Nexus	Requirement and Planning	1
Project Everest	Development	4.5
Project Everest	Systems Testing	1
Project Nexus	Design	1.5
Select a Project	Select a Task	0

Add Timesheet Submit

Figure 40. Day View Screen for MyTime Application

Time off request screen: The time off page lists all the recent and upcoming time off requests. The approval column displays the status of the request as (Approved/Rejected/Pending). The user can request time off by clicking on the Request Time Off link.



localhost:50762/Timeoff

Day View | Time Offs | Employee | Projects | Tasks | Projects Assignment | Register | Reports | Joe Admin

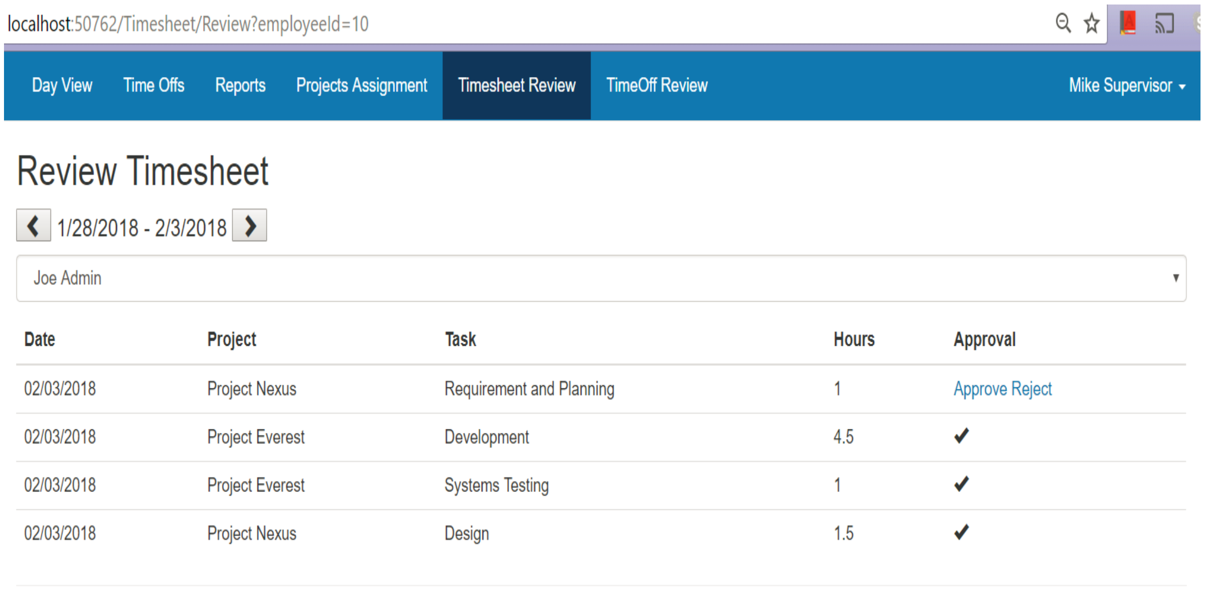
Time Offs

Request Time Off

Category	Start Date	End Date	Hours	Approval	Note (Optional)	
Sick Leave	02/14/2018	02/14/2018	4	**Pending**	doctors visit, taking half day off	✖
Floating Holiday	02/20/2018	02/21/2018	16	**Pending**	visiting family members in Duluth, taking two days off	✖
Vacation	03/12/2018	03/16/2018	40	✓	Vacation!!	✖
Floating Holiday	03/07/2018	02/07/2018	8	✓	taking a day off, volunteering with Second Harvest	✖

Figure 41. Time Off Request Screen for MyTime Application

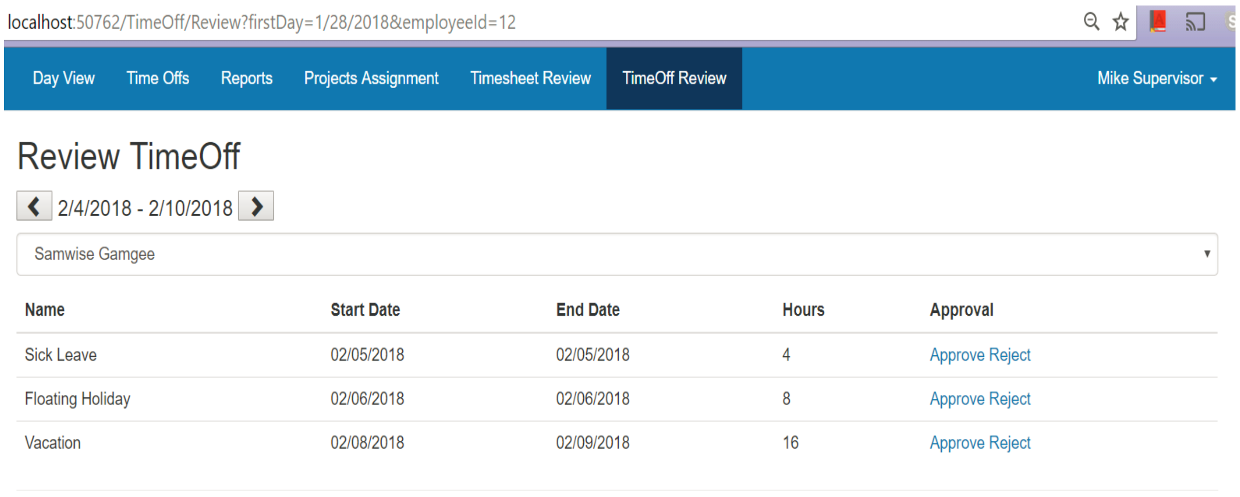
Review timesheet screen: Figure 42 shows timesheet review screen for an employee. The review timesheet is only available for Supervisor user, this page lists down timesheet submitted by the selected employee for the selected week. User can select the employee from drop down and the timesheet will be displayed in a tabular format. User can either Approve or Reject a timesheet by clicking on the respective link under Approval column.



Date	Project	Task	Hours	Approval
02/03/2018	Project Nexus	Requirement and Planning	1	Approve Reject
02/03/2018	Project Everest	Development	4.5	✔
02/03/2018	Project Everest	Systems Testing	1	✔
02/03/2018	Project Nexus	Design	1.5	✔

Figure 42. Review Timesheet Screen for MyTime Application

Review time off screen: Review time-off page is similar to review timesheet page, it provides a tabular interface for all time off requests made by selected employee for selected week. User can either Approve or Reject a time-off request by clicking on the respective link under Approval column.


localhost:50762/TimeOff/Review?firstDay=1/28/2018&employeeId=12

Review TimeOff

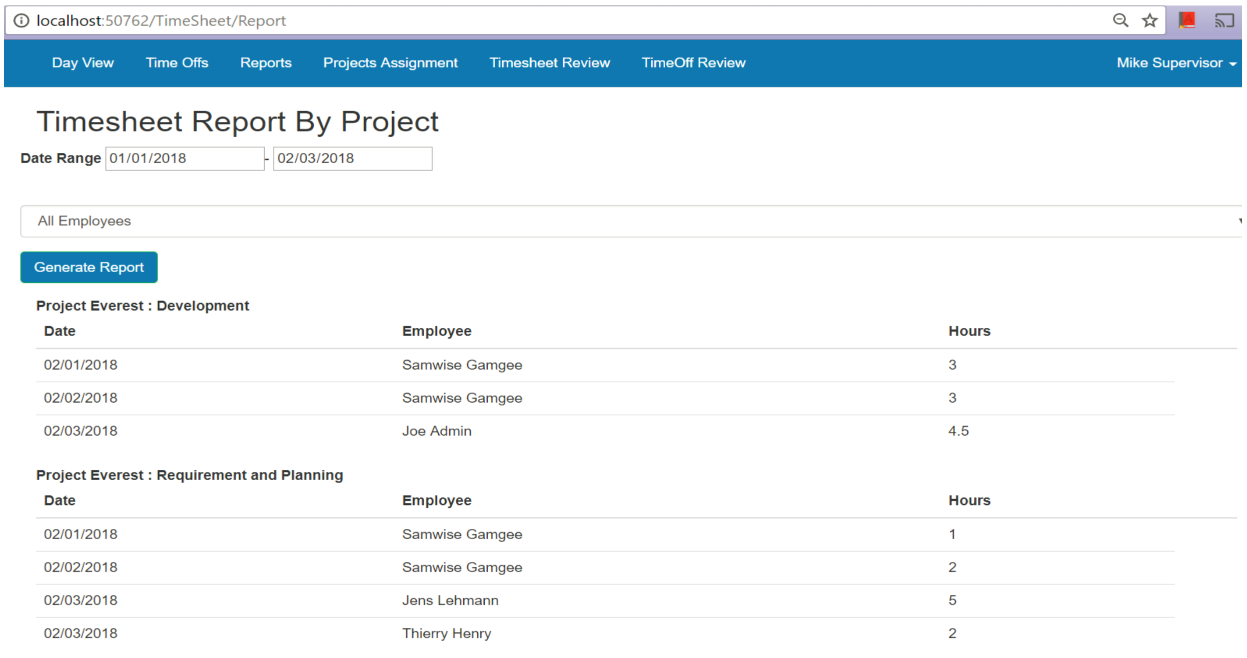
2/4/2018 - 2/10/2018

Samwise Gamgee

Name	Start Date	End Date	Hours	Approval
Sick Leave	02/05/2018	02/05/2018	4	Approve Reject
Floating Holiday	02/06/2018	02/06/2018	8	Approve Reject
Vacation	02/08/2018	02/09/2018	16	Approve Reject

Figure 43. Review Time Off Screen for MyTime Application

Reports screen: The reports page provides timesheet information for a selected date range grouped by project and task. Admin and supervisor users can either run reports for all employees or for each individual employee by selecting from the dropdown. An employee user will be able to select date range and can only generate personal timesheet report.


localhost:50762/TimeSheet/Report

Timesheet Report By Project

Date Range 01/01/2018 - 02/03/2018

All Employees

Generate Report

Project Everest : Development

Date	Employee	Hours
02/01/2018	Samwise Gamgee	3
02/02/2018	Samwise Gamgee	3
02/03/2018	Joe Admin	4.5

Project Everest : Requirement and Planning

Date	Employee	Hours
02/01/2018	Samwise Gamgee	1
02/02/2018	Samwise Gamgee	2
02/03/2018	Jens Lehmann	5
02/03/2018	Thierry Henry	2

Figure 44. Report Screen for MyTime Application

Chapter V: Conclusion

ASP.NET Core is a powerful framework for developing modern web applications because it is designed to be portable across multiple platforms for maximum code reuse and code sharing. In current context of software development, cross-platform development has become a popular alternative because using this approach prevents from having to create multiple projects for each targeted platform to be operable. Therefore, it not only reduces development cost but also ease development effort and enable faster development cycle.

ASP.NET Core framework is designed with modern software design principles. It uses MVC pattern which helps to decouple components in the system. It uses modular request pipeline which is an efficient way to process requests using small modules. This approach serves to make a lightweight application which greatly enhance performance of the application. It can easily be integrated with object relational mapping tools such as Entity Framework which makes application development faster by eliminating the need for programmers to write complex SQL queries. The framework provides seamless integration to modern client-side frameworks such as Bootstrap, AngularJS, ReactJS, etc. which makes it better equipped to develop Single Page Applications.

The objective of this project was to understand the cross-platform capabilities of ASP.NET Core web application and develop a timesheet management system using the framework. In this paper, I have explained cross platform approach utilized by .NET core framework which serves in building web application in modern and modular fashion. This project would be good reference for anyone trying to gain fundamental understanding of ASP.NET Core framework and gain practical knowledge on developing an application based on the framework and deploying it in multiple platforms. The web framework can be used in building various types of dynamic websites such as e-commerce sites, content-based websites or n-tier applications.

References

Aroraa, G. (2017). Understanding the problems with the monolithic architectural style. *Building Microservices with .NET Core 2.0*. Birmingham, UK: Packt Publishing.

Charbeneau, E., Bristowe, J., & Basu, S. (2017). The state of .NET in 2018: How the new .NET Standard is making you a better developer [White paper]. Retrieved from https://www.paceit.co.uk/wp-content/uploads/2016/06/the-state-of-dotnet-in-2018.pdf

Kanjilal, J. (2008). *Entity framework tutorial: learn to build a better data access layer with the ADO.NET entity framework and ADO.NET data services.* Birmingham, UK: Packt Publishing.

Oliveira, J., & Bruchet, M. (2017). *Learning ASP.NET core 2.0: Build modern web apps with ASP.NET Core 2.0, MVC and EF Core 2.* Retrieved from https://www.safaribooksonline.com/library/view/learning-aspnet-core/9781788476638/2c9dda13-211f-4cdb-ae4a-53857e901bd1.xhtml

Richter, J. (2002). The architecture of the .NET framework development platform. *Applied Microsoft .NET Framework Programming*, Washington, DC: Microsoft Press.

Shirhatti, S. (2017). *Host ASP.NET core on linux with nginx*. Retrieved from https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?tabs=aspnetcore2x

Strahl, R. (2016). Publishing and running ASP.NET core applications with IIS. Retrieved from https://weblog.west-wind.com/posts/2016/Jun/06/Publishing-and-Running-ASPNET-Core-Applications-with-IIS

Strahl, R., Ross, C., & Dykstra, T. (2017). *Introduction to ASP.NET core module*. Retrieved from https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/aspnet-core-module?tabs=aspnetcore2x

Watkins, D., Hammond, M., &Abrams, B. (2002). Introducing the .NET framework. *Programming in the .NET Environment*. Boston, MA: Addison-Wesley Professional.