School Website

Contents

[Abstract 4](#_Toc103789778)

[Introduction 5](#_Toc103789779)

[Objective 5](#_Toc103789780)

[Project Description 5](#_Toc103789781)

[System Study 6](#_Toc103789782)

[Existing System 6](#_Toc103789783)

[Proposed System 6](#_Toc103789784)

[User Interface 6](#_Toc103789785)

[Technical Specification 7](#_Toc103789786)

[Development Environment 8](#_Toc103789787)

[Software requirements 8](#_Toc103789788)

[Hardware requirements 8](#_Toc103789789)

[Production Environment 9](#_Toc103789790)

[Software requirements 9](#_Toc103789791)

[Hardware requirements 9](#_Toc103789792)

[Application Design 10](#_Toc103789793)

[Application Architecture 11](#_Toc103789794)

[Technical Architecture 13](#_Toc103789795)

[Data Flow Diagram 15](#_Toc103789796)

[School App – ASP.Net Core Web App 15](#_Toc103789797)

[Identity Server 15](#_Toc103789798)

[Application Development 16](#_Toc103789799)

[.Net Core App 16](#_Toc103789800)

[.NET Core Characteristics 16](#_Toc103789801)

[Implementation of a webpage 17](#_Toc103789802)

[Web resource management 18](#_Toc103789803)

[Bundling 18](#_Toc103789804)

[Minification 18](#_Toc103789805)

[Source Control 19](#_Toc103789806)

[Source Control Basics 19](#_Toc103789807)

[What is Git? 19](#_Toc103789808)

[Application Testing 20](#_Toc103789809)

[UI Testing 20](#_Toc103789810)

[Unit Testing 21](#_Toc103789811)

[Conclusion 22](#_Toc103789812)

[Annexure 23](#_Toc103789813)

[Source Code Highlights 23](#_Toc103789814)

[Screenshots 23](#_Toc103789815)

[Publishing & Deployment 23](#_Toc103789816)

[References 24](#_Toc103789817)

# Abstract

Any website we see in the web has its own intention and motivation to reflect the author’s authenticity, credibility, need for relaying information to a broader audience. This project is one such effort in featuring a school’s work and innovations in all their departments.

The school website is made with ASP.Net Core and using Razor pages. This includes the below,

* Responsive design
* Better UI to feature school’s achievements and announcements
* Account module for both Staff and Parents to manage secured information

# Introduction

## Objective

The main objective of this project is to design a website for a school that includes a useful homepage, staff information, news & announcement section, about us page with FAQs, account module for both staffs and parents, Gallery and many more.

Through this website, we are trying to accomplish the below,

* Make the school credible.
* Reach a broader audience.
* Relay valuable information.
* School is up with the times.
* Make the user understand that the school is interested in growth and innovation.

## Project Description

The school website is done with ASP.Net Core and using Razor pages.

The fundamentals – HTML is covered along with the C# using Razor pages, CSS is done using SCSS pre-processor and JS is done using native javascript and jQuery framework and other open-source libraries. Bootstrap is also used to build the site faster.

Gulp – task runner is used to compile, minify, and bundle the needed static files before the project builds.

GitHub – source control is where we host our project source code.

# System Study

## Existing System

## Proposed System

## User Interface

# Technical Specification

This section covers the technical specification of the school website. We use the .Net platform to build the website by leveraging the capabilities in a ASP.Net Core application. We have chosen C# language for our .Net code as it is the most widely used language in the .Net world.

We picked the Windows OS for both development and production environments.

The table below shows the detailed technical specification,

|  |  |
| --- | --- |
| **IDE** | Visual Studio Code or Visual Studio 2019+ |
| **Platform** | .Net |
| **Web Framework** | ASP.Net |
| **.Net Version** | .Net Core 3.1 |
| **Language** | C# |
| **Web pages** | Razor page using CSHTML  <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-3.1> |
| **Web UI** | 1. Bootstrap CSS framework 2. SCSS pre-processor |
| **Client side** | 1. jQuery JavaScript framework 2. Plain JavaScript |
| **Task Runner** | Gulp |
| **Database** | SQL Database |
| **Authentication & Authorization** | Duende IdentityServer |

## Development Environment

The development environment is aligned to the development phase of the SDLC. This is where processes related to software development are carried out. The development environment contains a set of different processes and tools for programming. These are used to develop the final software.

The development environment can also be a physical space where development takes place and where software engineers interact. Another example of the development environment is the integrated development environment (IDE). The IDE provides a platform where tools and development processes are coordinated in order to provide software engineers a convenient way of accessing the resources, they require during the development process.

The common build environment is closely aligned to the development phase of the SDLC. In this environment, software engineers merge the work done in the development environment. Within this environment, software engineers build systems. These are used to automate the process of software compilation.

### Software requirements

We use the below software and tools to develop the school website,

|  |  |
| --- | --- |
| **Software/Tools** | **Reference** |
| .Net Core 3.1  (SDK) | <https://dotnet.microsoft.com/en-us/download/dotnet/3.1> |
| Visual Studio Code | <https://code.visualstudio.com/docs/setup/windows> |
| MS SQL Server 2019 Developer Edition | <https://www.microsoft.com/en-in/sql-server/sql-server-downloads> |
| Node.js v16.14.0 | <https://nodejs.org/en/download/> |

### Hardware requirements

The requirements for the hardware are finalized such that it is in line with the software we use for the development environment,

* 2.4 GHz or faster processor
* 8 GB of RAM
* 500 MB of disk space

## Production Environment

The production environment aligns itself with the implementation phase of the SDLC. At the implementation phase, certain activities such as installation and configuration of the system are carried out. In many cases, the production environment is where the software is put into actual operation. The software becomes ready for use by the end users. The production environment can be a real-time space where software is run and where hardware has been installed. Within the production environment, users are no longer using the software to test it but to accomplish tasks in real-life scenarios.

We have chosen the Windows Server OS for the production deployment.

### Software requirements

|  |  |
| --- | --- |
| **Software/Tools** | **Reference** |
| .Net Core 3.1  (Hosting Bundle) | <https://dotnet.microsoft.com/en-us/download/dotnet/3.1> |
| IIS | <https://docs.microsoft.com/en-us/IIS-Administration> |
| MS SQL Server 2019 Developer Edition | <https://www.microsoft.com/en-in/sql-server/sql-server-downloads> |

### Hardware requirements

The requirements for the hardware are finalized such that it is in line with the software we use for the production environment,

* 2.4 GHz or faster processor
* 8 GB of RAM
* 500 MB of disk space

# Application Design

Application design combines the user interface (UI) and user experience (UX). While UI lends itself to the overall style of the app (including the colors, fonts, and general look and feel), UX focuses on the actual functionality and usability.

Many users abandon an app after using it for the first time. Since users are picky about which apps they use and so quick to abandon those they don’t enjoy, it’s essential to invest time and effort in creating a great user experience. The better the design, the better the chance that a user will engage with it and thus keep using it.

Our school website application design consists of the below,

1. Web pages
   1. To describe the school features, news, and announcements,
   2. That list the staff information,
   3. That shows the whereabouts of the school,
   4. To show the gallery pictures of the school
2. Identity Server
   1. To manage the accounts of the users – staffs and parents
3. Database
   1. To store the school’s data

## Application Architecture

An application architecture describes the patterns and techniques used to design and build an application. The architecture gives you a roadmap and best practices to follow when building an application, so that you end up with a well-structured app.

Software design patterns can help you to build an application. A pattern describes a repeatable solution to a problem.

Patterns can be linked together to create more generic application architectures. Instead of completely creating the architecture yourself, you can use existing design patterns, which also ensure that things will work the way they’re supposed to.

As part of an application architecture, there will be both front-end and back-end services. Front-end development is concerned with the user experience of the app, while back-end development focuses on providing access to the data, services, and other existing systems that make the app work.

The architecture is a starting point or roadmap for building an application, but you’ll need to make implementation choices not captured in an architecture. For example, a first step is to choose a programming language in which to write the application.

There are many programming languages used for software development. JavaScript used with HTML and CSS is currently one of the more popular programming languages for web application development.

Our school website application architecture consists of the below,

* .Net Core web application – dynamic website that has the razor pages with contents of the pages.
* IAM – IdentityServer application that handles the account modules and does authentication using OAuth. This can be extended to use Google Accounts.
* Database – data storage to store user details, school information and other necessary data.

Diagram

Description automatically generated

## Technical Architecture

Technical architecture is creating a structured software solution that will meet the business needs and expectations while providing a strong technical plan for the growth of the software application through its lifetime. It is equally important to the business team and the information technology team. Technical architecture includes the major components of the system, their relationships, and the contracts that define the interactions between the components. The goal of the technical architect is to achieve all the business needs with an application that is optimized for both performance and security while planning for things they know are coming as well as for things they don’t yet envision or dream. Taking the time to design the architecture at the start will save major design changes, code refactoring, and expensive rework later in the project.

Technically, our school website consists of the below,

* Website application – School app
* IAM Implementation – IdentityServer
* Infrastructure – single point of contact between all connections that originate from web applications and ID server.
* Utilities – common utility code that is needed for all apps. Core service and app builder helpers are included here.
* Domain – All our transactional and value objects are included here.
* ID Account modules depicted in the above diagram is a part of ID Server that handles, all account requests.

Diagram

Description automatically generated

## Data Flow Diagram

### School App – ASP.Net Core Web App

A picture containing diagram

Description automatically generated

### Identity Server

Diagram

Description automatically generated

# Application Development

Application development is the process of designing, building, and implementing software applications. It can be done by massive organizations with large teams working on projects, or by a single freelance developer. Application development defines the process of how the application is made, and generally follows a standard methodology.

Web applications are software developed for a specific purpose, hosted on a web server. They can be accessed using any internet-enabled device, like phones, laptops, and desktops. Some examples of web applications are online banking sites, ecommerce websites, and social networks.

## .Net Core App

.NET Core is a new version of .NET Framework, which is a free, open-source, general-purpose development platform maintained by Microsoft. It is a cross-platform framework that runs on Windows, macOS, and Linux operating systems.

.NET Core Framework can be used to build different types of applications such as mobile, desktop, web, cloud, IoT, machine learning, microservices, game, etc.

.NET Core is written from scratch to make it modular, lightweight, fast, and cross-platform Framework. It includes the core features that are required to run a basic .NET Core app. Other features are provided as NuGet packages, which you can add it in your application as needed. In this way, the .NET Core application speed up the performance, reduce the memory footprint and becomes easy to maintain.

### .NET Core Characteristics

Open-source Framework: .NET Core is an open-source framework maintained by Microsoft and available on GitHub under MIT and Apache 2 licenses. It is a .NET Foundation project.

You can view, download, or contribute to the source code using the following GitHub repositories:

* Language compiler platform Roslyn: <https://github.com/dotnet/roslyn>
* .NET Core runtime: <https://github.com/dotnet/runtime>
* .NET Core SDK repository. <https://github.com/dotnet/sdk>
* ASP.NET Core repository. <https://github.com/dotnet/aspnetcore>

Cross-platform: .NET Core runs on Windows, macOS, and Linux operating systems. There are different runtime for each operating system that executes the code and generates the same output.

Consistent across Architectures: Execute the code with the same behavior in different instruction set architectures, including x64, x86, and ARM.

Wide range of Applications: Various types of applications can be developed and run on .NET Core platform such as mobile, desktop, web, cloud, IoT, machine learning, microservices, game, etc.

Supports Multiple Languages: You can use C#, F#, and Visual Basic programming languages to develop .NET Core applications. You can use your favorite IDE, including Visual Studio 2017/2019, Visual Studio Code, Sublime Text, Vim, etc.

Modular Architecture: .NET Core supports modular architecture approach using NuGet packages. There are different NuGet packages for various features that can be added to the .NET Core project as needed. Even the .NET Core library is provided as a NuGet package. The NuGet package for the default .NET Core application model is Microsoft.NETCore.App.

This way, it reduces the memory footprint, speeds up the performance, and easy to maintain.

CLI Tools: .NET Core includes CLI tools (Command-line interface) for development and continuous-integration.

Flexible Deployment: .NET Core application can be deployed user-wide or system-wide or with Docker Containers.

Compatibility: Compatible with .NET Framework and Mono APIs by using .NET Standard specification.

### Implementation of a webpage

In this section, we will discuss about the implementation of a webpage from scratch. Razor pages use the “@page” directive to inform about the URL of the page to ASP.Net

The below image shows the different directives and syntaxes used in a Razor page,

Graphical user interface, text, application

Description automatically generated

Here is a quick intro of what a Razor Page is made of:

1. Each Razor Page starts with an @page directive to indicate that it’s a Razor Page. This is different from Razor Views in MVC, which should not start with @page.
2. The @page directive may be followed by an @model directive. This identifies the corresponding C# model class, typically located in the same folder as the .cshtml page itself.
3. (Optional) You can include server-side code within an @{} block.
4. The rest of the page should include any HTML content you would like to display. This includes any server-side Tag Helpers and Model attributes.

## Web resource management

Web resource management for a .net core app includes the below,

1. Compiling, bundling and minification of static files
2. Serving static files for the application pages to the browser

Bundling and minification are two distinct performance optimizations you can apply in a web app. Used together, bundling and minification improve performance by reducing the number of server requests and reducing the size of the requested static assets.

Bundling and minification primarily improve the first page request load time. Once a web page has been requested, the browser caches the static assets (JavaScript, CSS, and images). So, bundling and minification don't improve performance when requesting the same page, or pages, on the same site requesting the same assets. If the expires header isn't set correctly on the assets and if bundling and minification isn't used, the browser's freshness heuristics mark the assets stale after a few days. Additionally, the browser requires a validation request for each asset. In this case, bundling and minification provide a performance improvement even after the first page request.

### Bundling

Bundling combines multiple files into a single file. Bundling reduces the number of server requests that are necessary to render a web asset, such as a web page. You can create any number of individual bundles specifically for CSS, JavaScript, etc. Fewer files mean fewer HTTP requests from the browser to the server or from the service providing your application. This results in improved first page load performance.

### Minification

Minification removes unnecessary characters from code without altering functionality. The result is a significant size reduction in requested assets (such as CSS, images, and JavaScript files). Common side effects of minification include shortening variable names to one character and removing comments and unnecessary whitespace.

We use Gulp to do the bundling and minification of static files. We also have used Gulp to compile SCSS and obtain plain CSS files.

## Source Control

Source control (or version control) is the practice of tracking and managing changes to code. Source control management (SCM) systems provide a running history of code development and help to resolve conflicts when merging contributions from multiple sources.

### Source Control Basics

Whether you are writing a simple application on your own or collaborating on a large software development project as part of a team, source control is a vital component of the development process. Source code management systems allow you to track your code change, see a revision history for your code, and revert to previous versions of a project when needed. With source code management systems, you can collaborate on code with your team, isolate your work until it is ready, and quickly trouble-shoot issues by identifying who made changes and what the changes were. Source code management systems help streamline the development process and provide a centralized source for all your code.

### What is Git?

Git is an open-source distributed source code management system. Git allows you to create a copy of your repository known as a branch. Using this branch, you can then work on your code independently from the stable version of your codebase. Once you are ready with your changes, you can store them as a set of differences, known as a commit. You can pull in commits from other contributors to your repository, push your commits to others, and merge your commits back into the main version of the repository.

# Application Testing

Application testing refers to the process of testing any software application using scripts, tools, or any test automation frameworks in order to identify errors. It helps teams release bug-free and robust software applications into the real world. It also enables teams to identify bugs in the early stages of development and save development time.

Application testing is carried out in two phases – Frontend or User Interface testing and Backend testing that verifies the database behavior.

Naturally, there are two ways to perform application testing:

Manual Testing – The manual testing approach is used more frequently by developers in the initial phases of the development for testing specific features

Automated Testing – Once the software application is entirely developed, QAs automate test scenarios from the end-users’ perspective to evaluate the usability, functionality, and performance of the application

## UI Testing

UI Testing, also known as GUI Testing, is basically a mechanism meant to test the aspects of any software that a user will encounter. This usually means testing the visual elements to verify that they are functioning according to requirements – in terms of functionality and performance. UI testing ensures that UI functions are bug-free.

Websites comprise web elements created with CSS, JavaScript, and numerous other programming languages. UI testing performs tests and assertions of these elements to validate their efficacy. It is focused on examining visual and structural parts of the software i.e., parts the user would be concerned with, rather than the internal logic of the software.

UI Testing covers the gamut of visual indicators and graphic-based icons – toolbars, fonts, menus, text boxes, radio buttons, checkboxes, colors, and more.

Some of the features included in UI test suites include:

* Functionality
* Visual Design (includes responsiveness)
* Performance
* Usability
* Compliance

## Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit testing is an important step in the development process, because if done correctly, it can help detect early flaws in code which may be more difficult to find in later testing stages.

Unit testing is a component of test-driven development (TDD), a pragmatic methodology that takes a meticulous approach to building a product by means of continual testing and revision. This testing method is also the first level of software testing, which is performed before other testing methods such as integration testing. Unit tests are typically isolated to ensure a unit does not rely on any external code or functions. Testing can be done manually but is often automated.

# Conclusion

# Annexure

## Source Code Highlights

## Screenshots

## Publishing & Deployment

# References

(.net core references)

(bootstrap references)

(gulp references)