# Features upon Features (Not Current)

## Overview: This document details the features of the Angular.Net App used in the course titled "Angular and ASP.Net Core Integration". By using the techniques described in the course, you are already further ahead of where you would be by starting from scratch. In fact, in terms of a projects lifecycle, the techniques used in the Angular.Net Starter App will put you months ahead of schedule.

## Unlike many developer, I spend much time analyzing the workflow and looking for shortcuts to developing an application. The key is automation. Some upfront time, researching and analyzing the workflow, will save you months of development time later, with all your projects. And that is what I can share with you. I've spent months, not only looking for shortcuts to make the development quicker, but also to enhance the performance, and make the application easier to maintain.

## As a Microsoft developer, you have become used to not having the need to drop down to the command line to perform an operation. By automating the development process, you will become less fatigued, and be able to spend your attention doing development.

## The target audience for the course, is the ASP.Net Core Developer that would like to use Angular, as the front-end technology.

## New Features:

The Angular.Net app has many new features! This version embraces the Angular CLI and the Dotnet CLI, which is the main difference from previous versions.

You can also configure your Angular projects build types, build for production, and PWA support directly from within the Angular.Net Studio Application or by using the Angular.Net Studio Browser Extension.

## Prerequisites and References:

In order to use the Angular.Net App you will need to download and install NodeJS. By installing NodeJS you will automatically install NPM, the Node Package Manage. Here are some links:

## Compatibility:

The most compatible web applications will perform with all modern browsers. To ensure that, the feature-set supports all browsers that support ECMAScript 5 (ES5).

## Project Template:

When creating an ASP.Net Core project, you are given 2 chooses. The project template I have chosen is the ASP.Net Core Web Application with the .Net Core. This project template has the benefit of running on Linux, macOS as well as Windows using .Net Core. Then you have 2 more choses for what type of middleware, WebAPI or MVC. Since we have no need for MVC we will chose the WebAPI.

By the way, since many Microsoft developer are coming from a background of MVC, you should know that Angular and MVC are mutually exclusive. Soon you will understand the full benefits of allowing the Angular framework to handle the UI. Not only is it faster and easier to maintain, but it also embraces the web technologies. Sorry to say it, but you are going to have to understand web technologies to be a good, modern web developer.

## 3rd Party Libraries:

There are many 3rd party libraries already integrated into the Angular.Net App, but easy to remove if there is no need for them. Also, there is no need for SignalR, or jQuery or any other dependency libraries. Each library is self-contained. Following is a list of integrated libraries:

Angular Material

rxjs

moment

## Abstracting Debug from Release:

As an ASP.Net Core developer, you may not understand the significant difference between running the application in the Debug mode, verses the Release mode, in terms of JavaScript performance. Well, guess what? There is a big difference! Once you realize that the sooner you will be productive.

Visual Studio and ASP.Net Core make it easy to determine the development configuration mode that you are in and allow you to configure the outputting results according to the configuration mode. This is all handled in the Startup.cs file.

## Using NodeJS to do the Heavy Lifting:

I would have never guessed that I was going to need NodeJS and Gulp to be an efficient application developer. Now I truly can't live without those 2 technologies. In case you didn't know, NPM is a repository for storing Node packages and comes with NodeJS. And Gulp is a JavaScript task runner that lets you automate tasks.

Also, as another feature of the Angular.Net App, I have implemented the ProjectBuild project, which is an easier way to organize your build tasks. By placing your tasks in a different project, you can more easily develop your tasks using TypeScript. More on that later...

## Automation to Streamline the Workflow:

By using Gulp and the Task Runner Explorer, we can bind to these events:

1) Before Build

2) After Build

3) Clean

4) Project Opens

## Creating a Release Build:

The Angular.Net App has many features that can make your application perform faster and make it easier to maintain. Creating a Release (Production) build does many things in one process. Some are performed by the Angular-CLI, and others are performed by a build hook. Following is a list of all the benefits of creating a release bundle using the routines provided by the AngularNetCore Starter App Release Build.

1) Separates the framework from the application

2) Concatenate and bundle the framework with the 3rd party libraries

3) Minify the framework bundle

4) Embedding the images, fonts, and icons into the html

5) Squash the html and css into the components

6) Concatenate and bundle the application

7) Minify the application bundle

8) Enabling the production mode

9) Creating an application manifest

10) Adding Cache-Busting logic

11) Bumping up the application version number

12) Adding the Service Worker logic for offline performance

14) Updating the index.html with new versions

## Creating the Application Manifest:

I've given a lot of attention to the web technologies that enable applications to work off-line. This is like having a web app that performs like a desktop app. The Angular.Net App can perform most function while being complete disconnected from the Internet. This is currently implemented in the release build process and can be completely automated.

## The ProjectBuild Project:

As a proficient software developer, you are not going to want to start from scratch creating a new Release build process for your projects, every time you start a new project. In this course I will show you how to create a reusable Release Build Process that you can use on all your projects.

## Debugging Gulp Files:

Developing and debugging Gulp files can be an interesting challenge. Most developers will struggle with this for months.

When I create gulp tasks I want to be able to:

1) Set breakpoints

2) Step into my gulp task

3) Do searches for references

4) Go to definitions

5) Have full Intellisense

6) Build with TypeScript classes

The ProjectBuild project allow you to do all of these things while debugging your Gulp tasks. All you need to do is start the project (ProjectBuild), then set a break point in the tasklist.js files execute method, then double-click on one of the task in the Task Runner Explorer. Make sure that you select the ProjectBuild in the Task Runner Explorer before double-clicking the task.

## Automatically Versioning the Application:

By using NodeJS, it possible bump up the version number. This can be automated by binding it a build event. During the course I will demonstrate the binding, and the technique for incrementing the version number using NodeJS.

## Exception Handling:

## ASP.Net Core applications need a different method for returning exception data back to the Angular service. The Angular.Net Studio implements a technique to return useful exception information back to the client. ASP .Net Core has methods for returning exception Information, that can used for showing the exception information.

## Passing Parameters to the Server:

ASP.Net Core applications use a different method for abstracting the application parameter. These type of parameter, such as connection string, used to be stored in the web.config. Now they are stored in the appsettings.json. This is also a good place to store the application version number.

## Software Switches to Automate the Development Workflow:

Software switches can make development quicker by setting up your code in such a way to quickly return to an area in the application that you are currently implementing. This eliminate the need for repetitive keystroke and mouse clicks, just to get you to where you need to be in the application to continue working. I'll show you how in the course to make this possible and not running the risk that this test switches affect the production build.

## Benefits using client-side OO Techniques:

Most ASP.Net developer understand the benefit of using object oriented techniques such as inheritance, and encapsulation, but don't understand how that is possible with JavaScript. With the advantage of TypeScript, and transpilers, this actually is pretty easy, and give huge benefits to the integrity and maintainability of the application. The Angular.Net App uses most of these object oriented techniques. Following is a list.

1) Strong data types

2) Classes

3) Modules

4) Inheritance

5) Accessibility (private/public)

## Saving and Displaying Performance Analytics:

The Angular.Net Studio displays performance analytics obtained from Performance Markers that are inserted into the code. During the course we are going to use Chrome Dev Tools to display and analyze timing information.

## Saving the Navigation to the Last Feature:

This may not seem like a big deal, but sometimes it is best to pick-up where you left off using an application. So by saving the last feature that was used before exiting, it's possible to return to that feature when the application is launched again. I don't intend on explaining this feature in the course, but you can see by examining the source that the technology used is Local Storage.

## Publishing to Server Solution:

The Angular.Net Application comes with a separate Solution for publishing to the server. The idea is that it is not necessary to publish unbundled script files to the server. This alternate Solution only includes the folders that are necessary for the server to deliver the application. With this Solution, the time it takes to publish to the server is reduced to 5 percent.

## Full-Features and Mobile Versions:

By having different Application Build Configurations, it is possible to create a full-feature version, and a mobile version of the application. By using the same codebase, you will have the ability to reuse code for 2 different Angular projects. But it doesn't stop there! You can have as many Angular builds as necessary in one Visual Studio project. This is a great way to break up a large-scale Angular project into small projects.

## Real-Time Messaging:

Many developers struggle with the concept of real-time data, and how to accomplish this feat with a web application. Here is a scenario.

Let's say that you need to be notified when an event happens within the system. Maybe another user, within the system, makes a change to the database.

The Angular.Net Application comes with a solution to that problem, and is demonstrated in the Notification feature. This feature allows you to send messages to other users. The users that want to receive your messages will simply subscribe to your channel. Then, they will receive the notifications in real-time. What this means is that it is not necessary to refresh the browser to receive the latest notification.

The important point to make is that the notification is not limited to text. The notification can be simple text, an object, or a complex collection of objects.

## Voice Activated Commands:

Now, here is the feature that puts it all together. With the aid of some of the most modern web technologies, it is possible to activate features of the application, by simply using your voice. This opens up so many, many, new possibilities.

The Notification component, has implemented voice activated commands. What this means is just by using your voice, it is possible to emulate or replace mouse, and keyboard commands. Look Mom! No hands!

Hopefully you will understand the implications, and possibilities. In one of my "follow-up" courses, I plan to demonstrate a fully functional voice activates help system.

Imagine this: Computer... tell me about...

## Already Ready:

The Angular.Net Application comes with the scaffolding for you to immediately begin implementing your own custom feature. The feature is titled "Already Ready".

ASP.Net Core CLI

Integrated Help System

---------------------

Welcome to the Angular.Net Studio! The Angular.Net Studio is a Visual Studio project that contains an Angular 9 Workspace. The Angular Workspace contains 3 Angular projects, each sharing the same code.

This Visual Studio project is designed specifically to showcase techniques for solving enterprise challenges.

Note: The accompanying video tutorials walks through this application, explaining the theory of operation for techniques to solve enterprise challenges.

Here, on this view, you will find a listing of the features for the Angular.Net Studio.

Http Demos

The most typical Http Request are demonstrated here. You will find demons on how to download and upload files with a progress bar, and much more!