

PTO Time Management High Level Design Document

*Deepa Rajasabeson*

*Application Development*

*October 11, 2019*

# Introduction

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding and can be used as a reference manual for how the modules interact at a high level.

# Purpose

Currently, work hours can be lost to PTO because time-off availability is manually handled and approved by managers. This project is to create a tool that allows managers to set parameters to automatically manage/control/approve how many team members in the Nashville and Schaumburg office can request PTO on a daily basis, on holidays, on a first-come first-serve basis. This tool automates time off and shift swap requests allowing the Nashville and Schaumberg management teams to effectively manage these requests. The current process for managing these requests is done through inefficient system utilizing Outlook and Outlook Calendar to manage the process. This application will replace the current system.

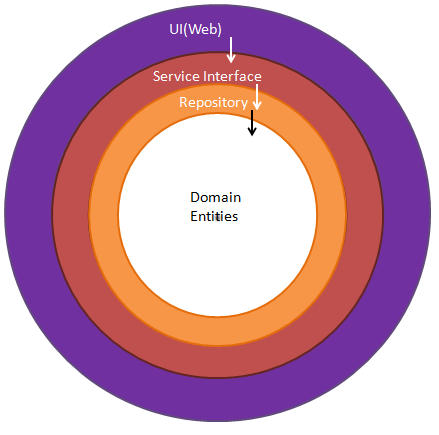
# General Description

# High Level Design

This application will use .NET Core 2.1 with and ReactJS front end to handle the view of the user, and an MVC entry point to manage authentication and authorization and HTTP calls from the React JS front end. The middle tier of the application will utilize the Onion Architecture Pattern to manage and separate concerns throughout the application.

## Onion Architecture

Onion Architecture is an application architecture that relies heavily on the Dependency Inversion Principle. The Onion Architecture term was coined by Jeffrey Palermo in 2008. This architecture provides a better way to build applications for better testability, maintainability, and dependability on the infrastructures like databases and services. Between the layers of the Onion, there is a strong *dependency rule*: **outer layers can depend on lower layers, but no code in the lower layer can depend directly on any code in the outer layer**.

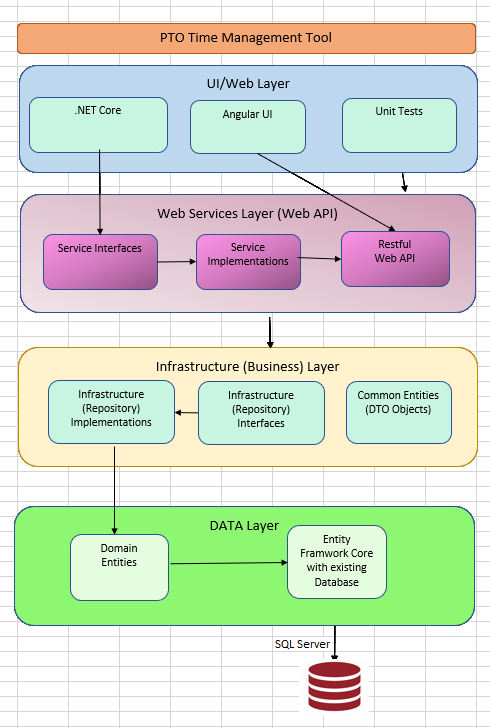


It consists of layers as :

1. Presentation layer
2. Service layer (Web API)
3. Business layer
4. Domain Entities (Data) layer.

Each layer communicates with the layer below it through interfaces. The internal layers never depend on external layer.

**Architecture Diagram for Time Off Request Tool :**



Projects in Application

|  |  |  |  |
| --- | --- | --- | --- |
| **Folder** | **Holds** | **Project Name** | **Description** |
|  |  |  |  |
| Test | MSTest Project | PTOTMT.Test.MSTest | A .Net Core MSTest Unit Tests project that contains unit tests. |
| Web | Web .Net Core UI Main Project with Angular as ClientApp | PTOTMT.Web | An ASP.Net Core 3.0Test Web Application project with API Controller for a RESTful HTTP Service. This project has integrated Angular 8 application set up to create Angular 8 UI Framework along with jQuery, Bootstrap and with additional technologies to better the user experience.  It is the most external part of an application by which the end user can interact with the application (User Interface).  It builds loosely coupled applications with in-built dependency injection in ASP.NET Core.  It represents the UI/Web layer of the Onion Architecture. |
| Common | Common methods in infrastructure (Business Layer) - (Data Transfer Object) | PTOTMT.Common | A .Net Core Class library project to hold Base Data Transfer Classes and methods in Business Layer.  It also holds DbContext class.  It represents the Business Layer of the Onion Architecture. |
| Repository | Repository Interfaces and implementations | PTOTMT. Repository | A .Net Core Class library project to hold interface **definitions and implementations** for data  **repositories** in infrastructure (Business Layer) to retrieve data and manipulate data in SQL server data storage using Entity Framework. Through these objects the Business Layer interacts (passes data to) with the beneath Web Services Layer.  It represents the Business Layer of the Onion Architecture. |
| Service | Web API | PTOTMT. Service | A .Net Core Web API project that hosts Web API End Points for all CRUD operations on all table data.  It represents the Service Layer of Onion Architecture . |
| Service | Service Interfaces and Implementations | PTOTMT. Service. WebCalls | A .Net Core Class Library project that holds **implementations for all interfaces** created in PTOMT.Services project for the purpose of generating business objects to execute all web service call methods which would fetch secured data from already hosted web services by making http call.  It represents the Service Layer of Onion Architecture . |
| Domain | Domain Entities to use in Data Layer | PTOTMT.Domain | A .Net Core Class library project that contains the classes for Entity Framework Code First Migration objects. It holds DbContext class. The Entity Framework Code First data access approach needs to create a data access context class that inherits from the DbContext class. |
| Database | Sql Server Database Project | PTOMT.Database | A Visual Studio’s Sql Server Database Project to manage our database scripts.  We can create a new database project and import database schema from an existing database, a .sql script file. We can then invoke the same visual designer tools (Transact-SQL Editor, Table Designer) available for connected database development to make changes to the offline database project, and publish the changes back to the production database. The changes can also be saved as a script to be published later. Using the Project Properties panel, we can change the target platform to different versions of SQL Server (including SQL Azure). |

# Database Schema

3 Schema

|  |  |  |
| --- | --- | --- |
| **PTO** | **Security** | **Config** |
| Request | Users | RequestType |
| Quota | TeamUser | Location |
|  | Role | Status |
|  | Team  Title |  |

A screenshot of a cell phone

Description automatically generated

# Databases

PTOTimeManagement, PTOTimeManagementTest for development and testing..

Connection String

Data Source=DESKTOP-GR4UU4J;Integrated Security=True;Persist Security Info=False;Pooling=False;MultipleActiveResultSets=False;Connect Timeout=60;Encrypt=False;TrustServerCertificate=False

## Table Definitions

* A table that contains all PTO requests submitted by Sales Operations Team
* Request Status (Accepted, Cancelled, WaitingForApproval)

### PTO. Request

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| UserId | uniqueidentifier | yes | Foreign Key |
| RequestTypeId | uniqueidentifier | yes | Foreign Key |
| Description | nvarchar(50) | no |  |
| StartDateTime | datetimeoffset | yes |  |
| EndDateTime | datetimeoffset | yes |  |
| Hours | decimal (4,1) | yes |  |
| StatusId | uniqueidentifier | yes | Foreign Key |
| QuotaId | uniqueidentifier | yes | Foreign Key |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | userId |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

PTO.Quota

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| Name | nvarchar(30) | no |  |
| Description | nvarchar(50) | no |  |
| StartDateTime | datetimeoffset | yes |  |
| EndDateTime | datetimeoffset | yes |  |
| OriginalHours | decimal (4,1) | yes |  |
| RemainingHours | decimal (4,1) | yes |  |
| TeamId | uniqueidentifier | yes | Foreign Key |
| IsActive | Bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

Security.User

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| FirstName | nvarchar(30) | yes |  |
| LastName | nvarchar(30) | yes |  |
| TitleId | uniqueidentifier | yes | Foreign Key |
| NTLogin | nvarchar(12) | yes |  |
| EmailAddress | nvarchar(50) | yes |  |
| RoleId | uniqueidentifier | yes | Foreign Key |
| ReportToUserId | uniqueidentifier | yes | Foreign Key |
| LocationId | uniqueidentifier | yes | Foreign Key |
| TeamFunctionId | uniqueidentifier | yes | Foreign Key to Security.Team |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

Role :

1. Manager
2. Developer 1
3. Developer 2

Security.Role

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| Name | nvarchar(30) | yes |  |
| Description | nvarchar(50) | no |  |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

Security .TeamUser

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| UserId | uniqueidentifier | yes | Foreign Key |
| TeamId | Uniqueidentifier | yes | Foreign Key |
| EffectiveDate | datetimeoffset | yes |  |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

Teams :

1. Help Desk 8am-9pm
2. OnBoarding / Recovery (OBR) 9am-6pm
3. Leadership / Admin 8am-5pm
4. Sales Order Entry (SOE) 8am-6pm

MaxShiftSlideHours : 3 Hours

Security.Team

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| Name | nvarchar(30) | yes |  |
| Description | nvarchar(50) | no |  |
| MaxShiftSlideHours | decimal(4,1) | yes |  |
| ShiftStartTimeLimit | decimal(4,1) | yes |  |
| ShiftEndTimeLimit | decimal(4,1) | yes |  |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

Location :

1. Nashville
2. Schaumburg

Config.Location

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| Name | nvarchar(30) | yes |  |
| Description | nvarchar(50) | no |  |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

Config.RequestType

Time-off Category :

1. Vacation
2. Personnel Swap
3. Flex Time
4. Float
5. Bereavement
6. Shift Slide
7. Jury Duty

(Need to prepare work flow for each category)

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| Name | nvarchar(30) | yes |  |
| Description | nvarchar(50) | no |  |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

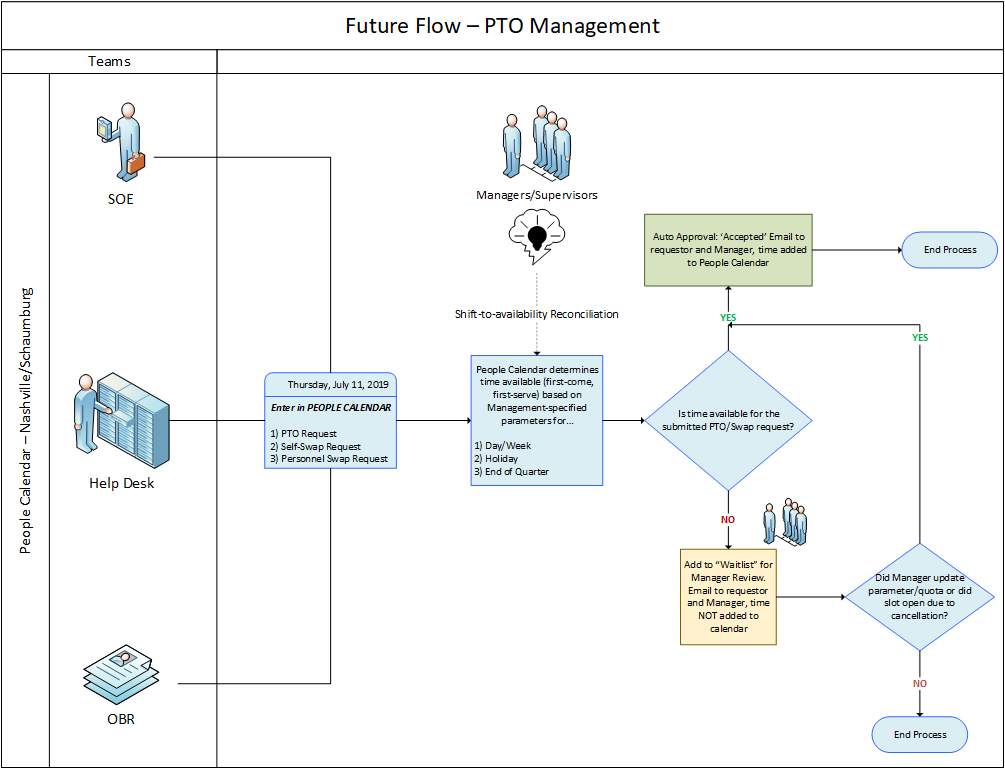
Status :

1. WaitList
2. Approved
3. Cancelled

Config.Status

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Required?** | **Description** |
| Id | uniqueidentifier | yes | Primary Key |
| Name | nvarchar(30) | yes |  |
| Description | nvarchar(50) | no |  |
| IsActive | bit | yes |  |
| CreatedBy | uniqueidentifier | yes | Foreign Key |
| CreatedOn | datetimeoffset | yes | Default value as GETUTCDATE() |
| UpdatedBy | uniqueidentifier | no | Foreign Key |
| UpdatedOn | datetimeoffset | no | Default value as GETUTCDATE() |

# Expected Process Work Flow :

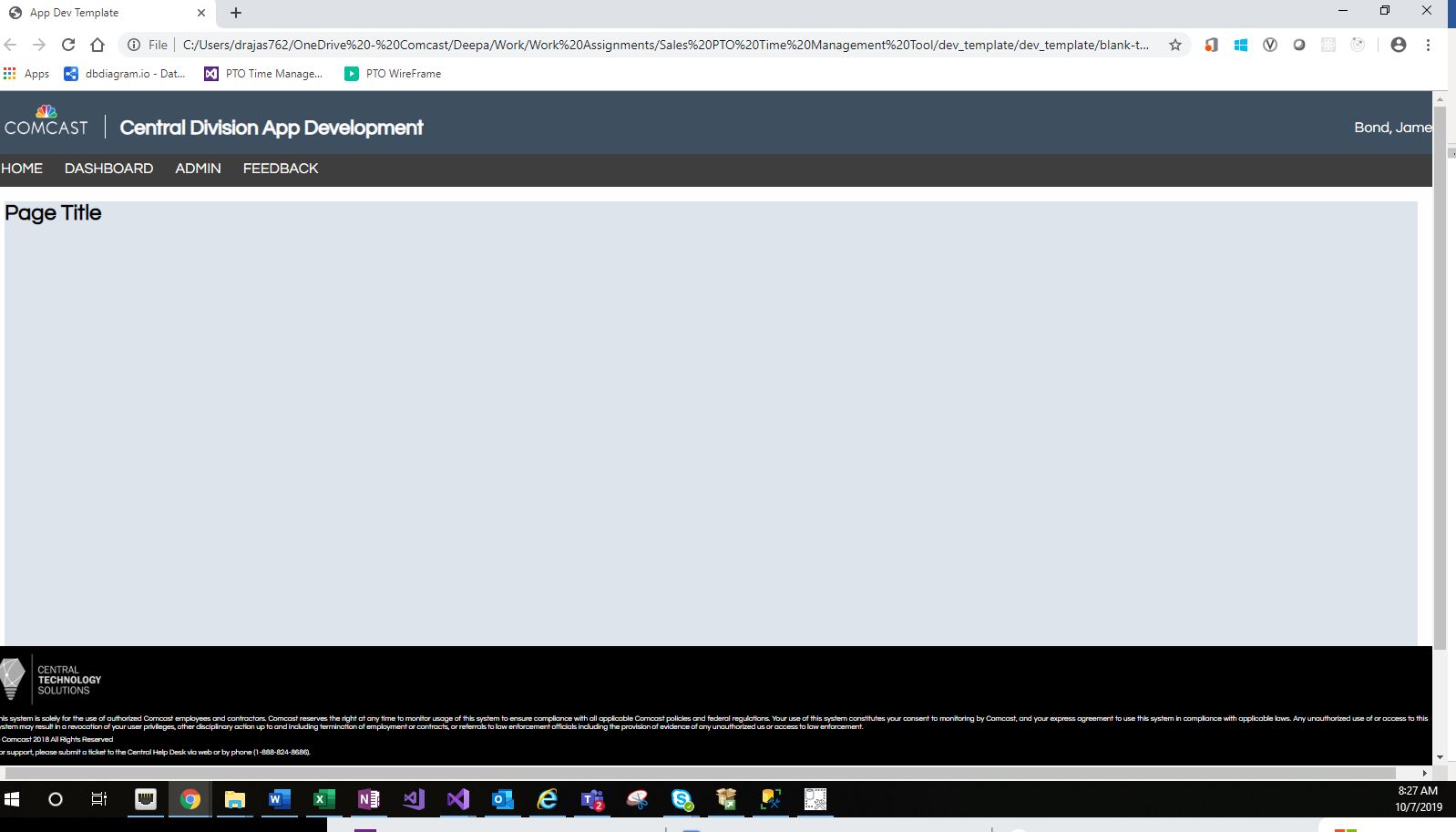


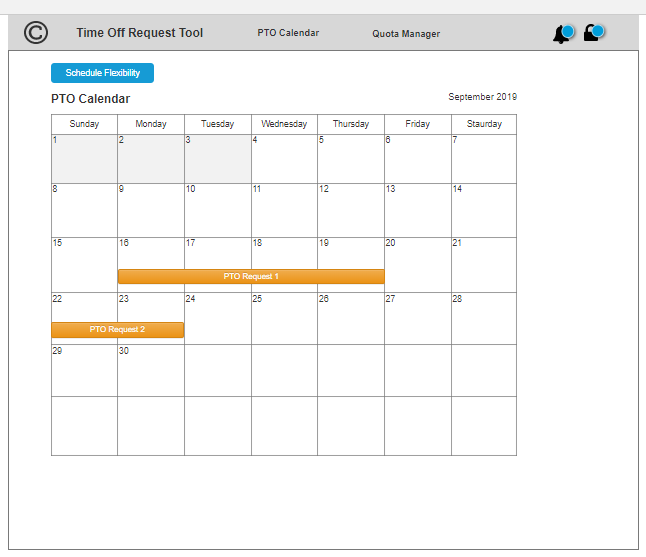
|  |  |  |
| --- | --- | --- |
|  |  |  |

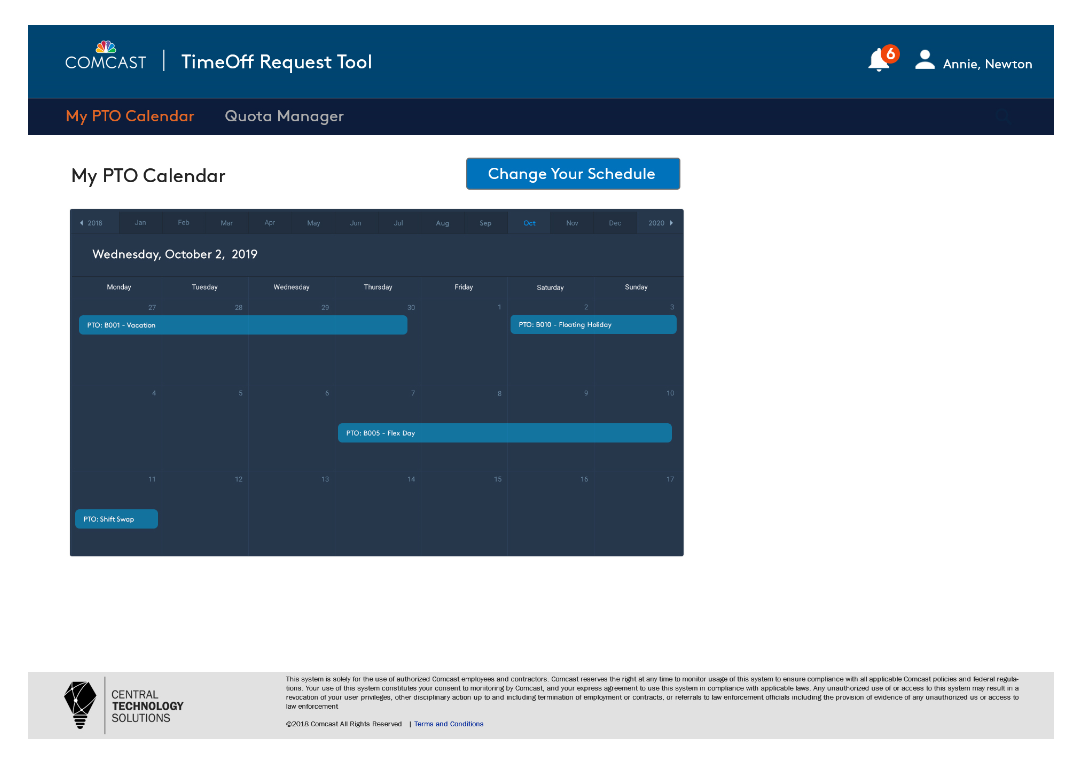
A screenshot of a social media post

Description automatically generated**User Interface Features**

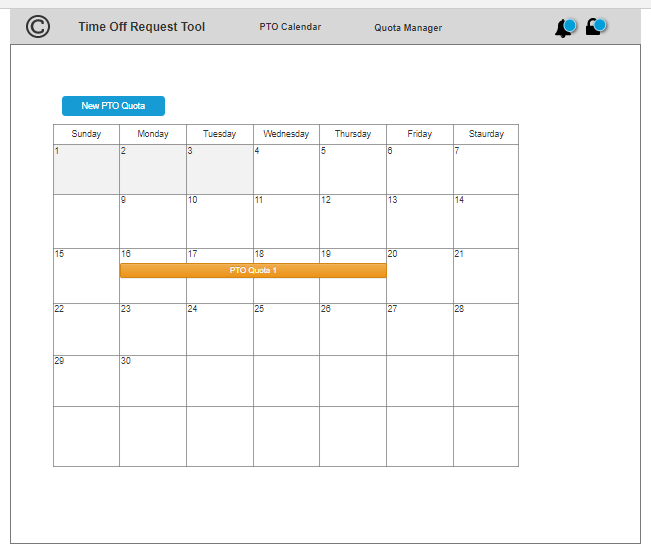
**Templates :**







Quota Manager :



PTO Quota Editor (Manager View) :

A screenshot of a social media post

Description automatically generated

Edit Shift Quota (Manager View) :

A screenshot of a social media post

Description automatically generated

Create PTO Request Form (Employee View) :

A screenshot of a social media post

Description automatically generated

**Shift Slide** - Schedule Flex (Employee View) :

A screenshot of a social media post

Description automatically generated

**Self-Shift Swap** - Schedule Flex (Employee View) :

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generated

**Shift Swap** (with another Employee) - Schedule Flex (Employee View) :

A screenshot of a social media post

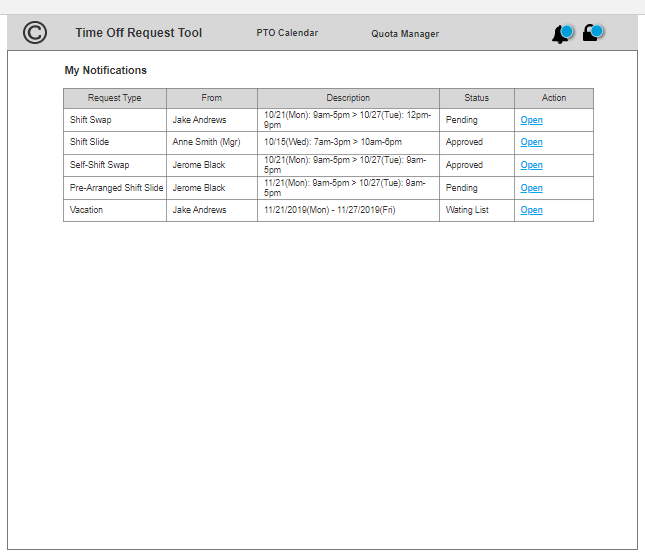
Description automatically generated

My Notifications – Manager View :A screenshot of a social media post

Description automatically generated

My Notifications (Employee View) :

(Same as Manager View except “View Waiting List” button is not in Employee View)



Waiting List (Manager View) :

A screenshot of a social media post

Description automatically generated

My Notifications Details (Shift Swap Request) - Employee View :

A screenshot of a social media post

Description automatically generated

My Notifications Details (Shift Swap Request) – Manager View :

A screenshot of a cell phone

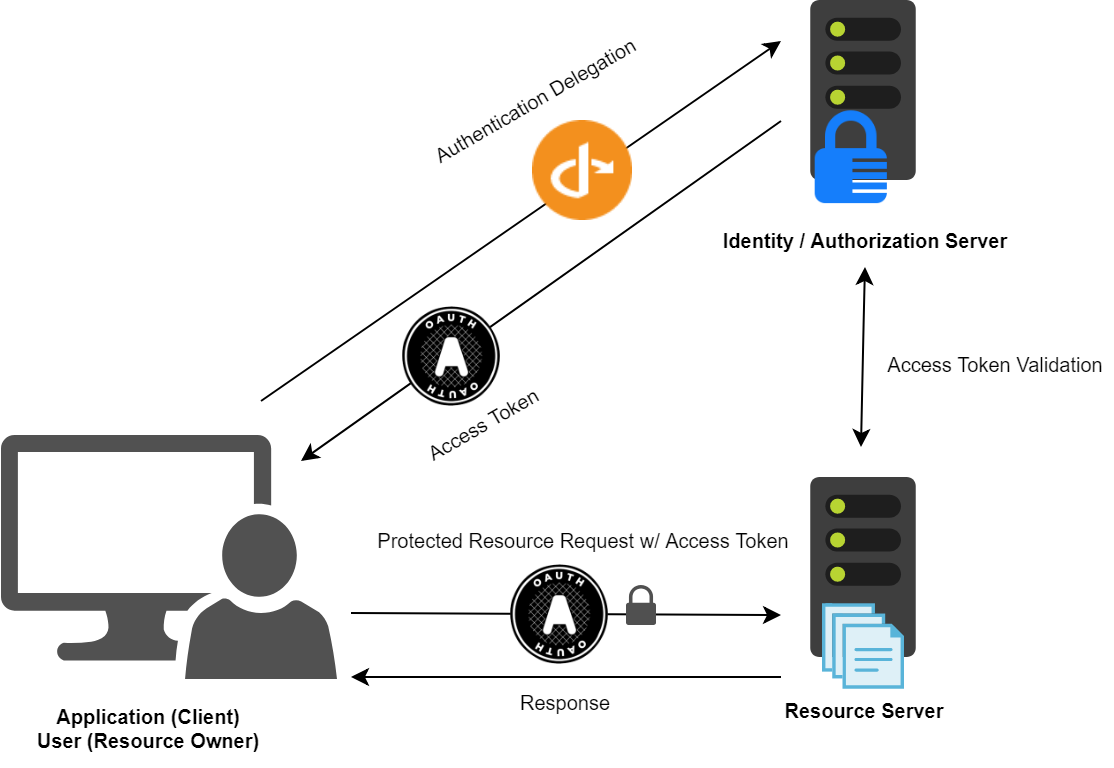
Description automatically generated

**Project Share point Site :**

<https://appdevprojects.cable.comcast.com/sites/DefaultCollection/PTO%20Time%20Management/Shared%20Documents/Forms/AllItems.aspx>

Diagram showing concepts and protocols fit together in a basic authentication and authorization scenario :

OIDC/OAuth flow



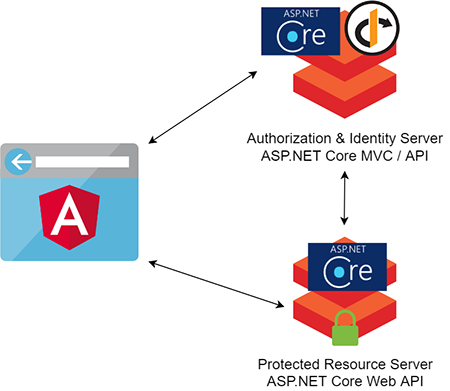
**To summarize this workflow:**

1. The user (resource owner) initiates an authentication request with the authorization server.
2. If the credentials are valid and everything checks out the authorization server obtains end-user consent and grants the client application an access token.
3. The access token is attached to subsequent requests made to the protected resource server.
4. The authorization server validates the access token; if successful the request for protected resources is granted, and a response sent back to the client application.

## **Architecture**

Our solution architecture has three main components:

* [SPA client application](https://github.com/mmacneil/AngularASPNETCoreOAuth/tree/master/src/Spa/oauth-client) - Angular
* [Authorization/Identity server](https://github.com/mmacneil/AngularASPNETCoreOAuth/tree/master/src/AuthServer) - ASP.NET Core MVC and IdentityServer4
* [Resource server](https://github.com/mmacneil/AngularASPNETCoreOAuth/tree/master/src/Resource.Api) - ASP.NET Core Web API



1. **OpenID Connect and OAuth 2.0 Framework for ASP.NET Core :**

**PM>** Install-Package IdentityServer4

1. ASP.NET Core Identity Integration for IdentityServer4 :

**PM**> **Install-Package** **IdentityServer4**.AspNetIdentity