Chelsea Case-Miller

Portfolio 2022-2023

Table of Contents

Workplace Project 2

- Time off Request Form and Calendar

Overview

There has been a need for a simple yet effective solution for staff to book holidays and time off, and for those to be approved and displayed in a calendar so people in each team can know when people are due to be in the office or not.
Previously this has been done using an email template to request time off and an Excel spreadsheet having to be updated each time a request is approved, which has taken up a lot of time as there are hundreds of requests each year. Another issue with keeping on top of this Excel sheet is that each year a new one has to be made and dates added and changed manually, this would take a large portion of time just for a calendar.

The HR department had come to me with an idea for a web based firm calendar with automatically updating events for holidays and time off. We had discussed this at length and a solution was formed.

Tools

The Tools used during the project were; Visual Studio for the development of the ASP.NET Core MVC Project, SQL and SQL Server Management Studio for the database creation and testing my SQL, GitHub for our GIT version management and finally GitHub Actions for our CI/CD needs.

My approach and work

[B2] As I was approached by the HR department to come up with a solution I started with doing some research into how the current process is handled for holidays and time-off. This was handled by using a spreadsheet on our local SharePoint site that showed everyone's holidays that they have taken. This would be updated by HR when they had been given an email from a member of staff requesting time-off and also approval by their head of department. The main issue with this solution, although being easy to understand, was the huge amount of time HR needed to spend creating a new template each year, adding and removing any members of staff, and then finally adding approved time-off to the spreadsheet.

[K3] As the organisation has only recently branched out into the world of software development, primarily for internal use, I take on many of the software development lifecycles roles. The main roles I take on in many projects including this one are; team lead, project manager and developer. These roles would typically be undertaken by multiple people. However, with there only being a small amount of developers and a lack of software development knowledge, I take on these roles myself. My knowledge gives me the ability to take on these roles to fit the requirements and objectives of the business. Furthermore, I understand the implications of my work and how the business would use this long term, ensuring I create a strong piece of software.

If there was a larger need for software development in the business, I could see a shift in these responsibilities. This would be because there would be more staff and further knowledge to fit the appropriate roles. Thus, making the software development lifecycle much more efficient moving forward.

[K1][K4][K6][S15][B7] The HR department and I took part in a face to face discussion, the discussion we held was focusing on the main objectives that the end product would need to fulfil, when talking to HR, since they are from a non-technical background, I adapted my language around terms in software development to a more simple to understand manor, this helped everyone in the meeting understand what was being conveyed and therefore improvements on ideas could be made, in the end we came up with key features that should be included in the end product.

The main requirements for the new system that were identified are:

- A timeline view of individuals time-off
- Grouping of individuals into departments for better overview of department attendance.
- Ability for Heads of department to approve holidays as and when they are requested.
- Ability for HR and partners to approve holidays when a head of department may not be available.
- Ability for individuals to book multiple types of time-off such as; Holidays, TOIL and Study Leave.
- Keep the interface intuitive and familiar to the current system.

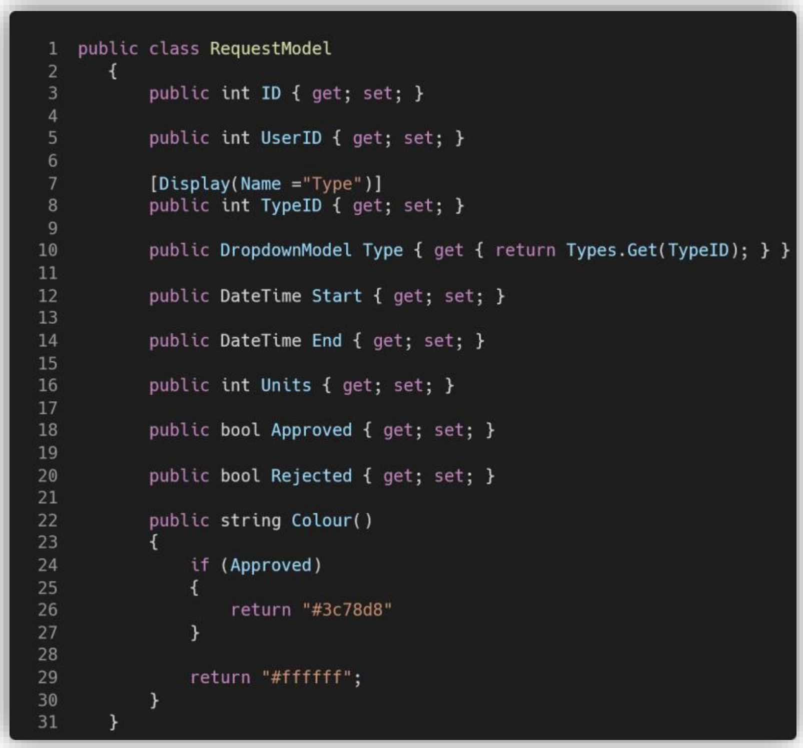
[K1][K2][K5] Now that I had gathered the requirements I moved on to start planning how I would take on the task of creating the software. I planned to use an Agile type of development by first creating a large almost fully fledged program based on the outlined points above, then test this and finally ask for feedback with the HR department to improve and make additions where necessary continuing to Design, Develop, Test and ask for feedback until a solution was found to be suitable for the task long term. As I have been the primary developer within the firm for the past 4 years I accepted full responsibility for many of the roles in the lifecycle such as: Designer in order to create the overarching look and feel of the application along with communicating these with HR, Programmer in order to create the interface and the backend to achieve the goals of the project, Tester to make sure the program works and ongoing maintenance of the deployed application to keep this functioning over the time it is in use and additionally bug fixing and adding new features when required.

[B3][K11][S1][S11] My primary thoughts on the design of the software was what framework to use, I had considered using Python and flask like I had for a previous project but soon realised that I could make use of ASP.Net's MVC (Model, View, Controller) framework along with windows authentication to better secure the program as, unlike my previous project, this will contain some more sensitive information such as names, departments and staff members time-off. C# is also an object orientated language meaning I can make use of classes in order to link my models with the data access layer to make both the writing of code and the use of models as seamless as possible.

[S8] The user interface design considerations I took were to keep the layout and structure to be as similar in design to the already used spreadsheet. This would allow users to quickly adjust and understand the program, making the task of training and support minimal.

[K8] To store the data I decided to use our on-site locally accessed Microsoft SQL server as this could be used for a secure store for the data we wish to store, being that the server is already heavily controlled using our firewall, VLAN's and authentication. Using the SQL server would also allow me to integrate the new time-off system with our Client/Staff database meaning any new users and those who are leaving can be added and removed without extra effort as these would already be entered in the existing staff database.

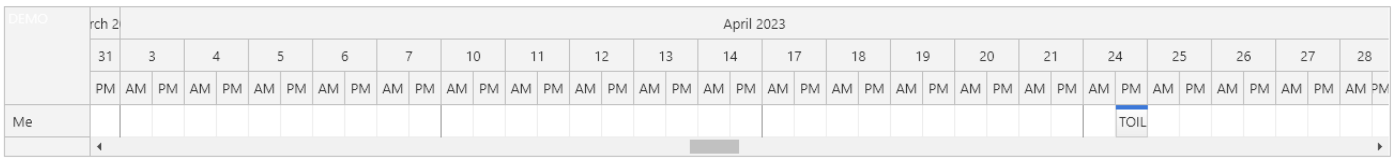
[B3][B5][K8] In addition to the network security around the database server, there are also policies and permissions applied to this meaning only those who have appropriate access can view or edit records for staff, this helps keep us in line with GDPR which states that we should keep information, especially personal information secure and only allow relevant people access it, and only when they are required to. Having this database system with permissions also allows us to keep the data stored on file relevant and up to date as each staff member can alter their own records for name, title, address etc.



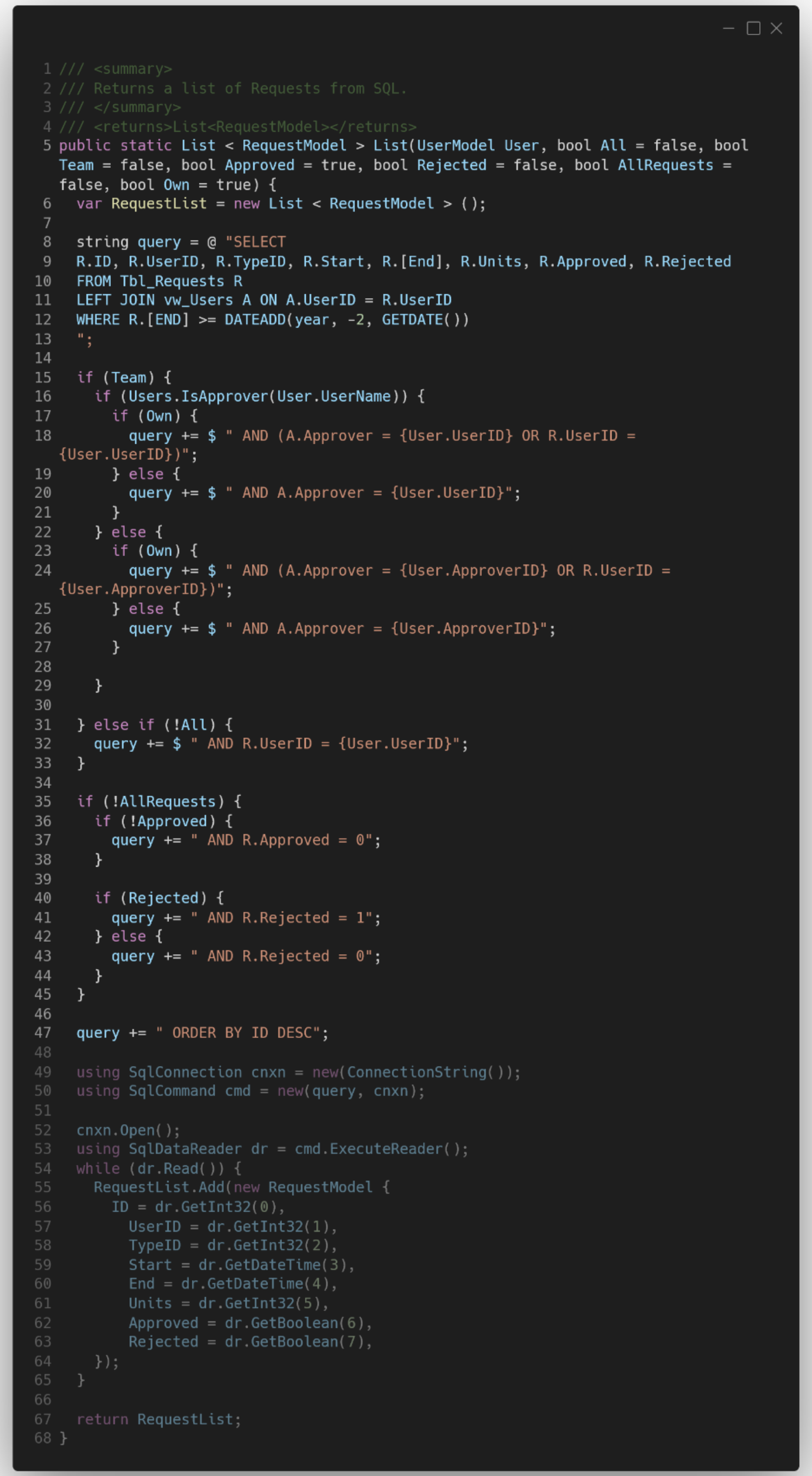
```
public class RequestModel
    {
        public int ID { get; set; }

        public int UserID { get; set; }

        [Display(Name ="Type")]
        public int TypeID { get; set; }

        public DropdownModel Type { get { return Types.Get(TypeID); } }

        public DateTime Start { get; set; }

        public DateTime End { get; set; }

        public int Units { get; set; }

        public bool Approved { get; set; }

        public bool Rejected { get; set; }

        public string Colour()
        {
            if (Approved)
            {
                return "#3c78d8"
            }

            return "#ffffff";
        }
    }
```

I started off creating a model that would represent any requests that would be submitted to the system. I had identified some fields such as the UserID, Start and End dates, and some Boolean values to indicate rejected or approved requests. Using these different data types has allowed me to easily perform operations on them such as filtering using the Boolean values.

[S2][B6] After I had my model created, I then looked at displaying this type of data. This however presented a major issue when creating the front end calendar style view. I knew that I would have to use complex JavaScript to change DOM elements in order to give a simple easy to use interface. I had considered how I would program this myself and came to the conclusion that it may take too much time and resources when something like this may exist already. I had then searched the web for client-side JavaScript frameworks that specialise in displaying calendar-type information. I had looked at multiple solutions and styles however the one that stood out was a library called DayPilot, their code specialises in calendar and timeline data and how to display this. The advantages of DayPilot was that I was able to customise this to fit the project requirements and style that was outlined by the HR department. Such as displaying a breakdown of days into AM and PM fields which allows a more accurate display of when staff are off without being too in depth as to overwhelm users.


DEMO
rch 2
April 2023
31 3 4 5 6 7 10 11 12 13 14 17 18 19 20 21 24 25 26 27 28
PM AM PM
Me
TOIL

I decided to manually test using some sample data as inputs at what this may have looked like, confirmed the style and layout with HR, which they liked, and went ahead with implementing this more thoroughly with a database connection and some additional views in order to add, and list requests.



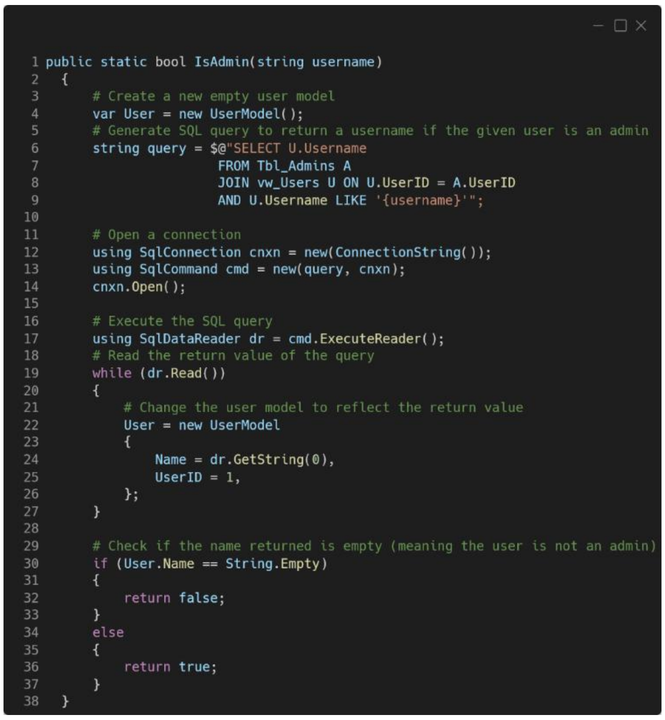
```csharp
/// <summary>
/// Returns a list of Requests from SQL.
/// </summary>
/// <returns>List<RequestModel></returns>
public static List < RequestModel > List(UserModel User, bool All = false, bool Team = false, bool Approved = true, bool Rejected = false, bool AllRequests = false, bool Own = true) {
  var RequestList = new List < RequestModel > ();

  string query = @ "SELECT
  R.ID, R.UserID, R.TypeID, R.Start, R.[End], R.Units, R.Approved, R.Rejected
  FROM Tbl_Requests R
  LEFT JOIN vw_Users A ON A.UserID = R.UserID
  WHERE R.[END] >= DATEADD(year, -2, GETDATE())
  ";

  if (Team) {
    if (Users.IsApprover(User.UserName)) {
      if (Own) {
        query += $ " AND (A.Approver = {User.UserID} OR R.UserID = {User.UserID})";
      } else {
        query += $ " AND A.Approver = {User.UserID}";
      }
    } else {
      if (Own) {
        query += $ " AND (A.Approver = {User.ApproverID} OR R.UserID = {User.ApproverID})";
      } else {
        query += $ " AND A.Approver = {User.ApproverID}";
      }

    }

  } else if (!All) {
    query += $ " AND R.UserID = {User.UserID}";
  }

  if (!AllRequests) {
    if (!Approved) {
      query += " AND R.Approved = 0";
    }

    if (Rejected) {
      query += " AND R.Rejected = 1";
    } else {
      query += " AND R.Rejected = 0";
    }
  }

  query += " ORDER BY ID DESC";

  using SqlConnection cnxn = new(ConnectionString());
  using SqlCommand cmd = new(query, cnxn);

  cnxn.Open();
  using SqlDataReader dr = cmd.ExecuteReader();
  while (dr.Read()) {
    RequestList.Add(new RequestModel {
      ID = dr.GetInt32(0),
        UserID = dr.GetInt32(1),
        TypeID = dr.GetInt32(2),
        Start = dr.GetDateTime(3),
        End = dr.GetDateTime(4),
        Units = dr.GetInt32(5),
        Approved = dr.GetBoolean(6),
        Rejected = dr.GetBoolean(7),
    });
  }

  return RequestList;
}
```

[K9][S16] As pictured above one of the main functions for the database connection was the ability to list all of the current requests based on some parameters, I used C# to create an if-else check for possible choices of parameter values and used them to somewhat dynamically generate the SQL query in order to return the correct results.

This code checks if the list is located in the “My Team”, “My Calendar” or “Firm Calendar” using the All, User and Team parameters, this allows reuse of code and one place to edit the list function, the code can also select a list of all requests in the search results or by default show only accepted or pending requests.

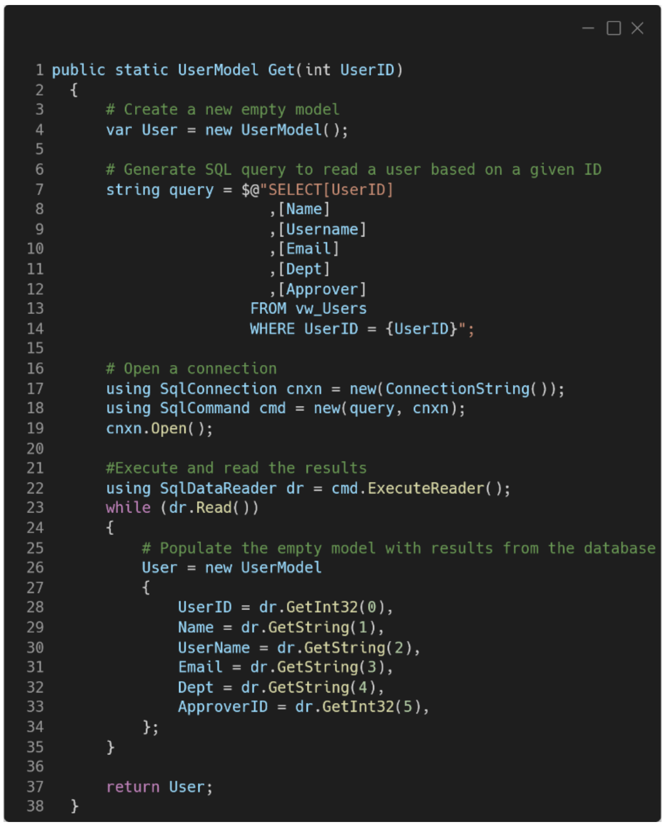
[S3] Expanding on the data access layer, each model in the project that requires data access (Requests, Types and Users) has a data access class. Within these classes contain the CRUD (Create, Read, Update and Delete) type queries to interact with the database where required. For example, users cannot be deleted or updated from the program since this links into a separate system that has permissions and controls in place for our staff database. It could potentially cause issues if both programs updated and deleted data independently.

Some of the data access models have specific data access operations outside of the standard CRUD queries. An example of this is the users model, in which there are specific actions such as “IsAdmin”. This checks to see if the database tag states if they are an administrator, then returns a boolean based on this value. This allows me to use shorter code in my razor webpages to allow access to more restricted areas of the program.



```
public static bool IsAdmin(string username)
  {
      # Create a new empty user model
      var User = new UserModel();
      # Generate SQL query to return a username if the given user is an admin
      string query = $@"SELECT U.Username
                      FROM Tbl_Admins A
                      JOIN vw_Users U ON U.UserID = A.UserID
                      AND U.Username LIKE '{username}'";

      # Open a connection
      using SqlConnection cnxn = new(ConnectionString());
      using SqlCommand cmd = new(query, cnxn);
      cnxn.Open();

      # Execute the SQL query
      using SqlDataReader dr = cmd.ExecuteReader();
      # Read the return value of the query
      while (dr.Read())
      {
          # Change the user model to reflect the return value
          User = new UserModel
          {
              Name = dr.GetString(0),
              UserID = 1,
          };
      }

      # Check if the name returned is empty (meaning the user is not an admin)
      if (User.Name == String.Empty)
      {
          return false;
      }
      else
      {
          return true;
      }
  }
```

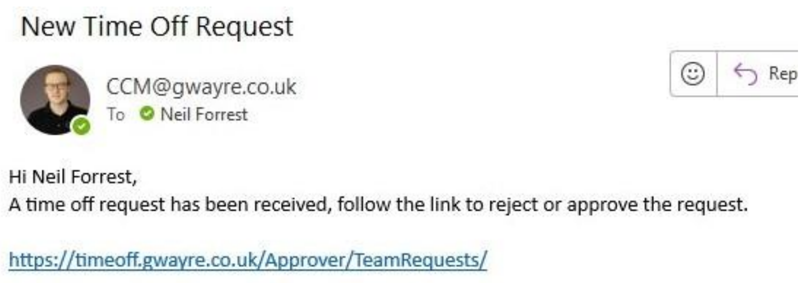
 [K9] Linking back to the more basic CRUD operations; the way these work is by running a SQL query against the database, then parsing the result into a new instance of the MVC’s model. Then, this object is returned for use by the controller and views.



```
public static UserModel Get(int UserID)
  {
      # Create a new empty model
      var User = new UserModel();

      # Generate SQL query to read a user based on a given ID
      string query = $@"SELECT[UserID]
                      ,[Name]
                      ,[Username]
                      ,[Email]
                      ,[Dept]
                      ,[Approver]
                    FROM vw_Users
                    WHERE UserID = {UserID}";

      # Open a connection
      using SqlConnection cnxn = new(ConnectionString());
      using SqlCommand cmd = new(query, cnxn);
      cnxn.Open();

      #Execute and read the results
      using SqlDataReader dr = cmd.ExecuteReader();
      while (dr.Read())
      {
          # Populate the empty model with results from the database
          User = new UserModel
          {
              UserID = dr.GetInt32(0),
              Name = dr.GetString(1),
              UserName = dr.GetString(2),
              Email = dr.GetString(3),
              Dept = dr.GetString(4),
              ApproverID = dr.GetInt32(5),
          };
      }

      return User;
  }
```

[S5][S6] I had made use of manual testing once the main program flow was built, this helped me get an idea of how to software feels to use and if there are any improvements to both the design and code I could make, I carried out testing using a set of test requests pretending like I was an end user requesting a holiday, I collaborated with my line manager as this is the person who would approve my holidays to see if my code for sending emails to and from was working.

To do this I ran a development session on my local machine, submitted some test information that would be typical for an end user to submit, once submitted I used SSMS (SQL server management studio) to look at the table where data should be save when submitted, it had successfully inserted into the database and the next step in testing was to confirm my line manager had gotten the automated email, on behalf of the user, that should have been sent out upon submission of the data, which worked well.


New Time Off Request

CCM@gwayre.co.uk
To Neil Forrest

Hi Neil Forrest,
A time off request has been received, follow the link to reject or approve the request.

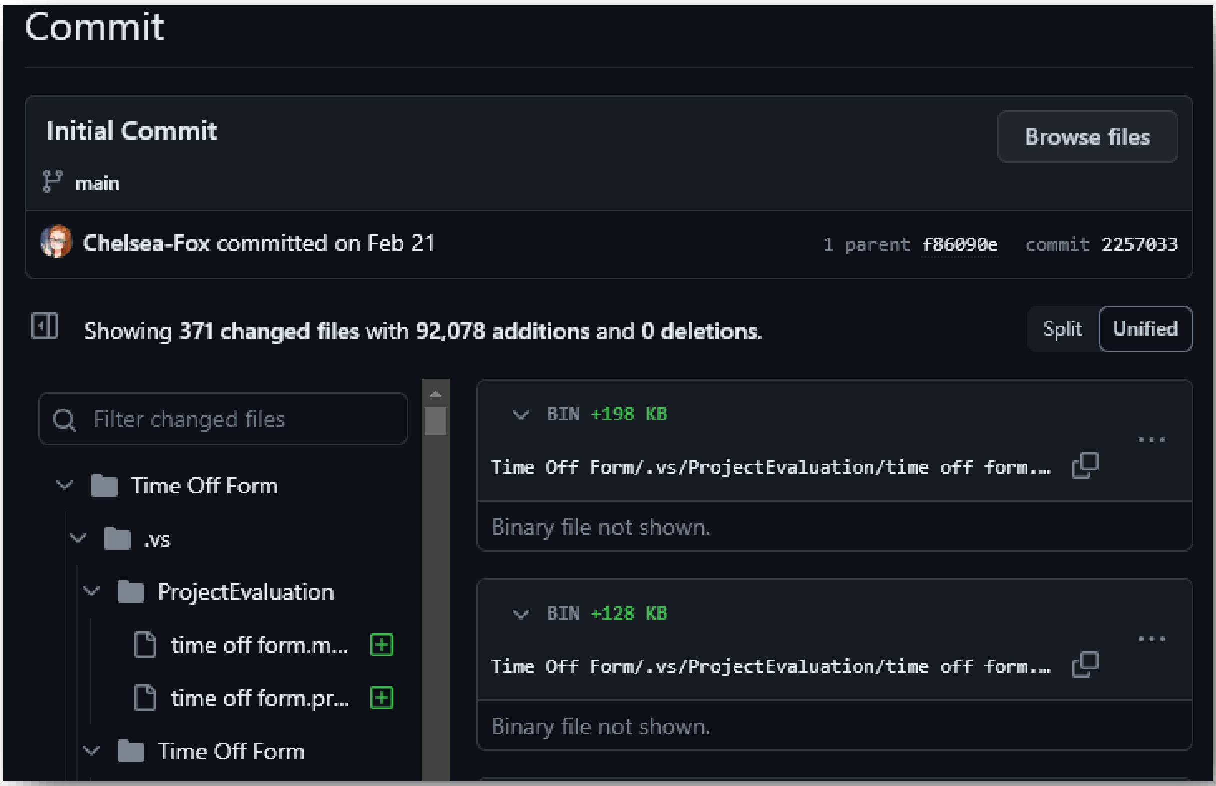
https://timeoff.gwayre.co.uk/Approver/TeamRequests/

[S7] I also used debugging tools such as breakpoints, in order to step through and “watch” variables change, in order to test the flow of the program by allowing myself to submit data and follow this through my program to make sure it was submitted and then selected from the database at the times it was needed.

[K12][S4][S13] Some other testing strategies I used throughout this project was acceptance testing, in which beta releases were given out via conditional firewall access to HR and some selected heads of department, this was to ensure that both the system worked with multiple users at the same time and also that the requirements were met. Although this was primarily classed as acceptance testing, this phase also covered usability and some compatibility testing as users had to navigate their way through the program and also many users in our organisation use different browsers based on their preferences. This allowed me to gather feedback on anything that may need improved, some of the features I had added after the main release, such as the calendar invite email was thought of by these beta users and implemented after the first round of agile development.

[K12] While my role may not be within a dedicated software house, I am well aware of various testing methodologies, including unit testing. In practice, unit testing involves the use of a testing framework such as NUnit, where I could write test cases for each function or method within the project. These test cases would be designed to cover a range of scenarios, including typical inputs such as integers for a math function, edge cases negative ints or floats for example, and error conditions such as a string or object. Automated testing tools can then execute these tests regularly, providing rapid feedback on whether any changes to the code have introduced regressions or errors, if I were to implement unit testing I could use GitHub actions to run these tests and give me feedback on each change that I make, I could also use these results to determine automatic merging and or publishing of the project.

[S14] Once the main portion of code was written I felt it was a good point to upload to a new repository on our private GitHub in order to be able to share and collaborate more easily if needed. To do this I created a new repository in GitHub and used GitHub Desktop, an app that provides an easy to use GUI (graphical user interface) to interact using git to publish changes to a repository on GitHub.


Commit

Initial Commit

main

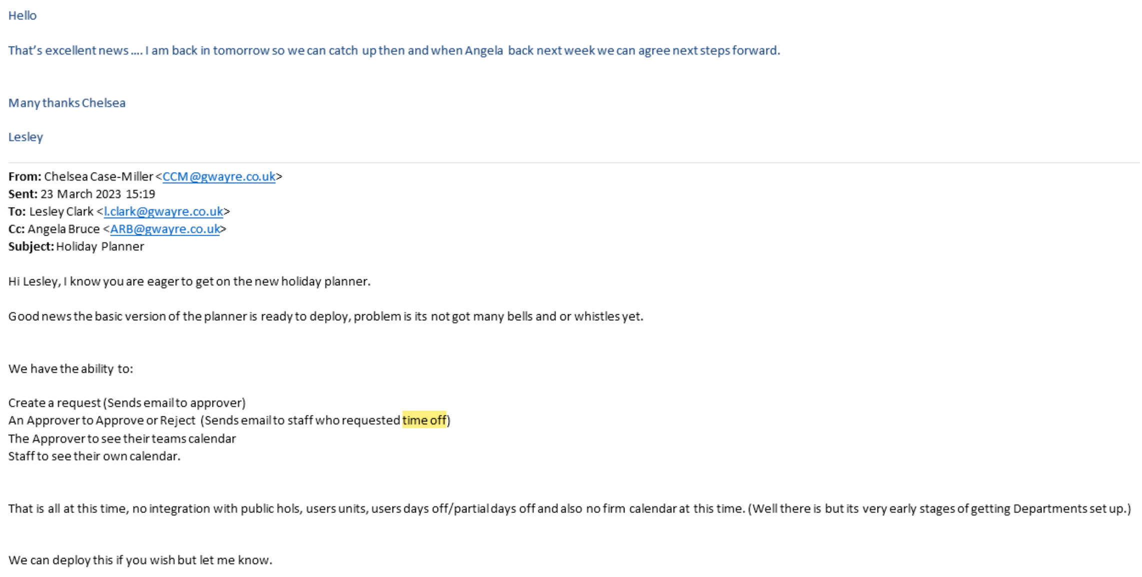
Chelsea-Fox committed on Feb 21 1 parent f86090e commit 2257033

Showing 371 changed files with 92,078 additions and 0 deletions.

BIN +198 KB
Time Off Form/.vs/ProjectEvaluation/time off form.…
Binary file not shown.

BIN +128 KB
Time Off Form/.vs/ProjectEvaluation/time off form.…
Binary file not shown.

[B7] Once I was confident in the basic release of the project I had contacted HR via email about opening up the planner for public use and user testing.


Hello

That's excellent news .... I am back in tomorrow so we can catch up then and when Angela back next week we can agree next steps forward.

Many thanks Chelsea

Lesley

From: Chelsea Case-Miller <CCM@gwayre.co.uk>
Sent: 23 March 2023 15:19
To: Lesley Clark <l.clark@gwayre.co.uk>
Cc: Angela Bruce <ARB@gwayre.co.uk>
Subject: Holiday Planner

Hi Lesley, I know you are eager to get on the new holiday planner.

Good news the basic version of the planner is ready to deploy, problem is its not got many bells and or whistles yet.

We have the ability to:

Create a request (Sends email to approver)
An Approver to Approve or Reject (Sends email to staff who requested time off)
The Approver to see their teams calendar
Staff to see their own calendar.

That is all at this time, no integration with public hols, users units, users days off/partial days off and also no firm calendar at this time. (Well there is but its very early stages of getting Departments set up.)

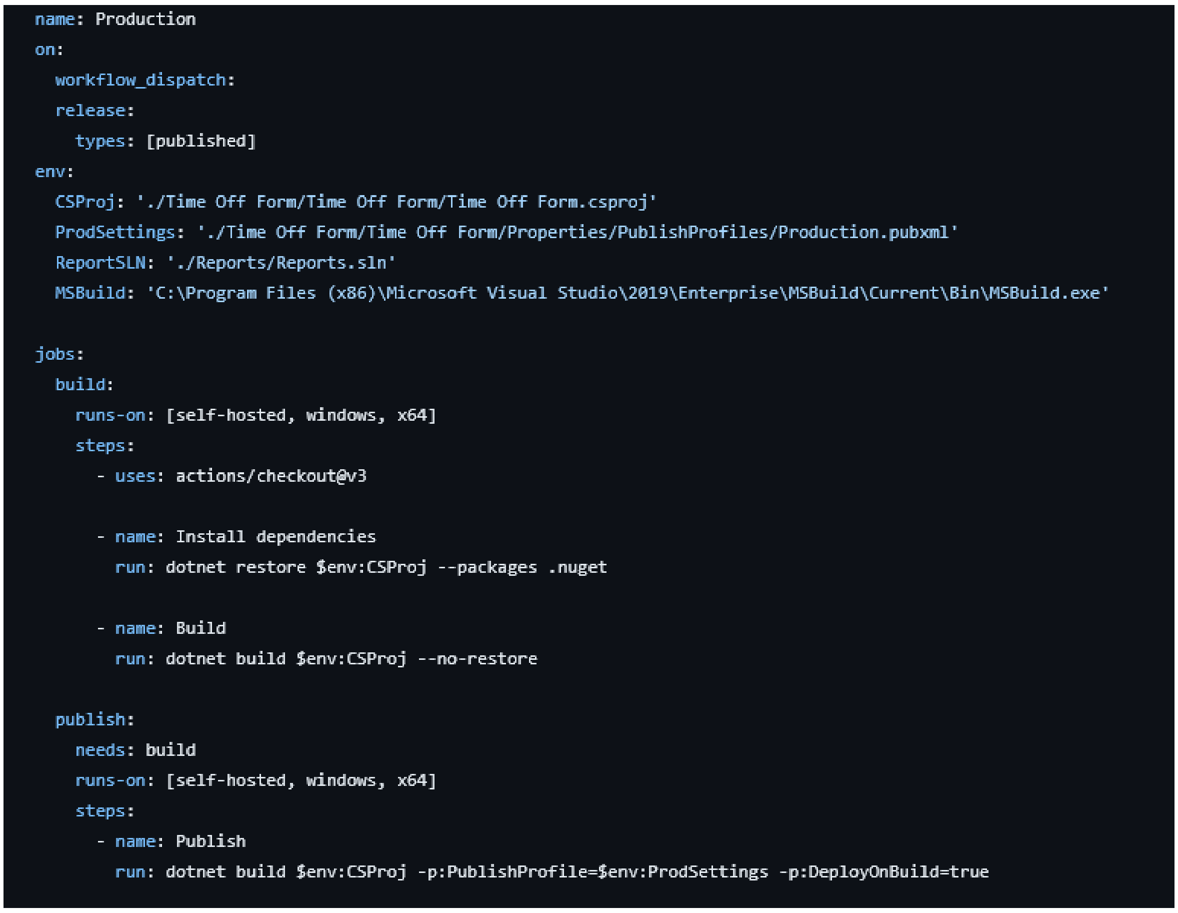
We can deploy this if you wish but let me know.

We then set up an in-person meeting to go over what to do, we had decided to go ahead with a release to HODs (Heads of Departments) with the main goal of a larger scale test in case there were bugs or anything unexpected as before this time I had only tested as a sole developer.

[S10] When publishing the repository to GitHub, this creates a new commit which allows us to keep track of our version control and allows us to track any changes between versions. Once uploaded to GitHub I deployed the code to IIS on our APPs server via CI/CD Pipelines in GitHub, the pipeline uses MSBuild alongside publish settings to specify the production environment, in this case we had pointed this to timeoff.gwayre.co.uk (an internally routed site hosted using IIS)

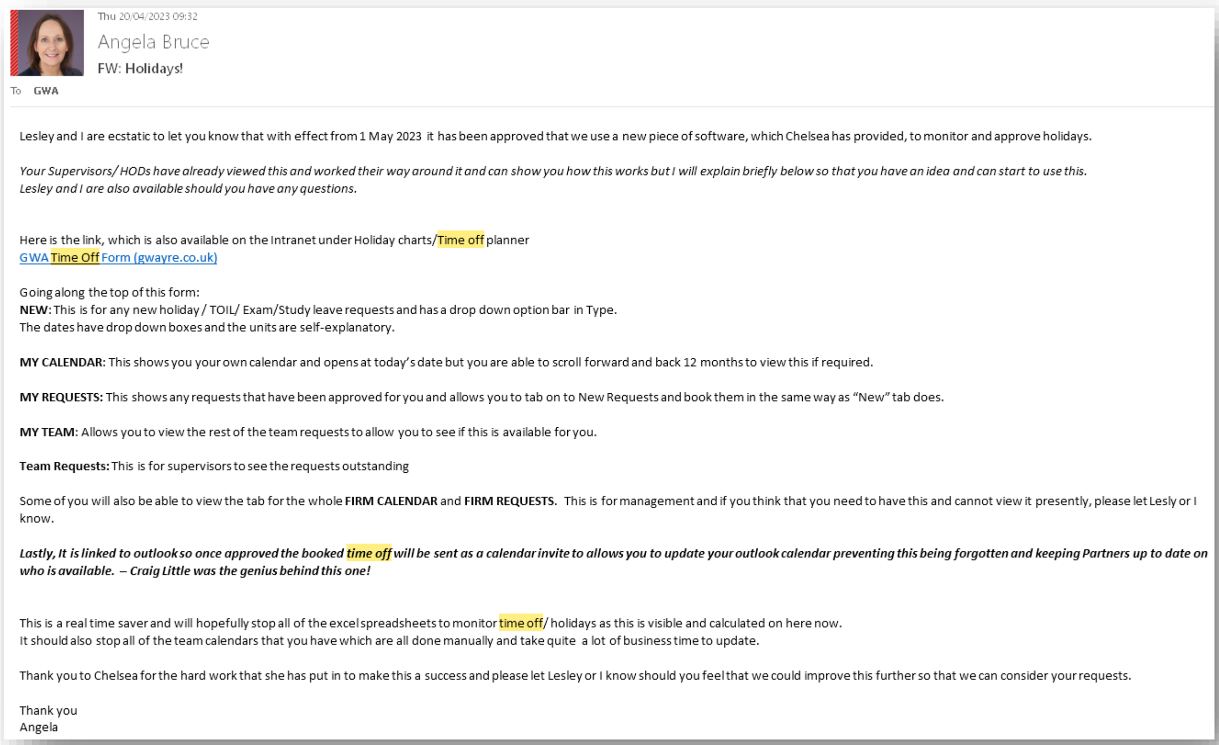
[S10] The following screenshot shows the Github action pipeline in which goes through multiple steps to publish the program to the internal IIS site.

First some environment variables are added to more easily reference paths later in the pipeline. Then it runs through the 2 main jobs. The first being "build" in which the GitHub runner fetched the code from the published release branch, installs any dependencies and then builds the solution to ensure the code has no runtime errors. Once built and the program has no runtime errors, it then uses the publish profile which I set up to send the program to IIS as a deployed app.



```yaml
name: Production
on:
  workflow_dispatch:
  release:
    types: [published]
env:
  CSProj: './Time Off Form/Time Off Form/Time Off Form.csproj'
  ProdSettings: './Time Off Form/Time Off Form/Properties/PublishProfiles/Production.pubxml'
  ReportSLN: './Reports/Reports.sln'
  MSBuild: 'C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\MSBuild.exe'

jobs:
  build:
    runs-on: [self-hosted, windows, x64]
    steps:
      - uses: actions/checkout@v3

      - name: Install dependencies
        run: dotnet restore $env:CSProj --packages .nuget

      - name: Build
        run: dotnet build $env:CSProj --no-restore

  publish:
    needs: build
    runs-on: [self-hosted, windows, x64]
    steps:
      - name: Publish
        run: dotnet build $env:CSProj -p:PublishProfile=$env:ProdSettings -p:DeployOnBuild=true
```

After the heads and HR had agreed to release the project we went ahead with publishing this and releasing it officially via a firm-wide email sent from HR.



Thu 20/04/2023 09:32

Angela Bruce

FW: Holidays!

To GWA

Lesley and I are ecstatic to let you know that with effect from 1 May 2023 it has been approved that we use a new piece of software, which Chelsea has provided, to monitor and approve holidays.

Your Supervisors/ HODs have already viewed this and worked their way around it and can show you how this works but I will explain briefly below so that you have an idea and can start to use this.
Lesley and I are also available should you have any questions.

Here is the link, which is also available on the Intranet under Holiday charts/Time off planner
GWA Time Off Form (gwayre.co.uk)

Going along the top of this form:
NEW: This is for any new holiday / TOIL/ Exam/Study leave requests and has a drop down option bar in Type.
The dates have drop down boxes and the units are self-explanatory.

MY CALENDAR: This shows you your own calendar and opens at today's date but you are able to scroll forward and back 12 months to view this if required.

MY REQUESTS: This shows any requests that have been approved for you and allows you to tab on to New Requests and book them in the same way as "New" tab does.

MY TEAM: Allows you to view the rest of the team requests to allow you to see if this is available for you.

Team Requests: This is for supervisors to see the requests outstanding

Some of you will also be able to view the tab for the whole **FIRM CALENDAR** and **FIRM REQUESTS**. This is for management and if you think that you need to have this and cannot view it presently, please let Lesly or I know.

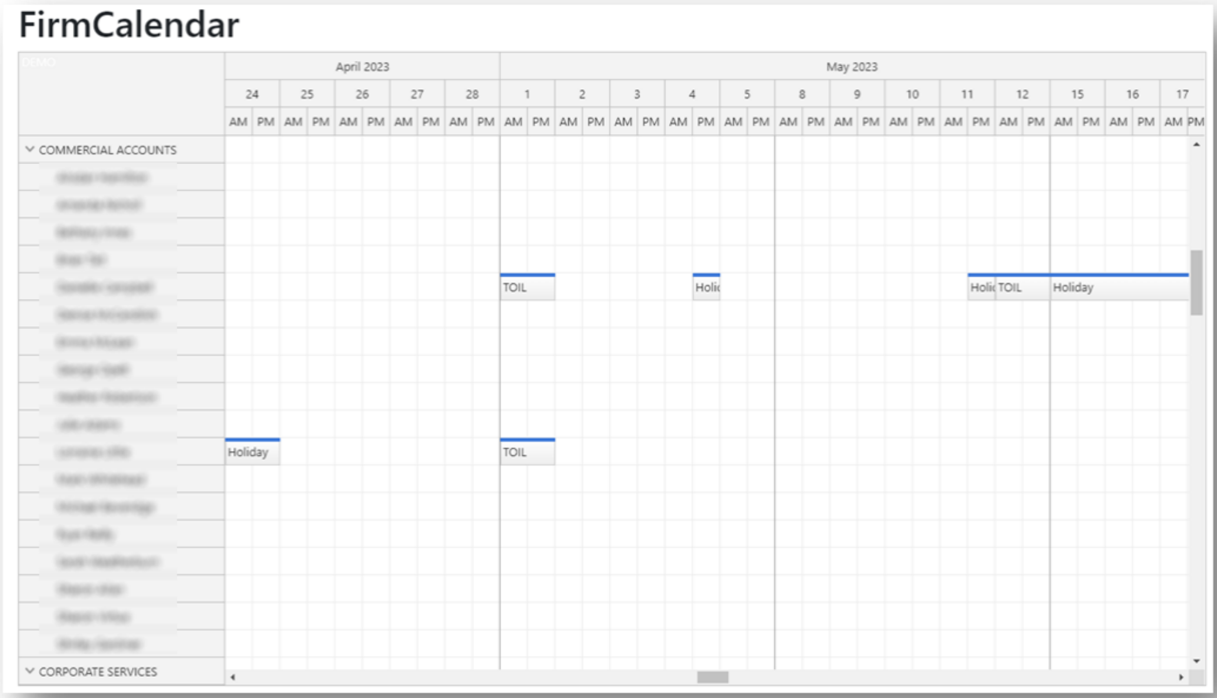
Lastly, It is linked to outlook so once approved the booked time off will be sent as a calendar invite to allows you to update your outlook calendar preventing this being forgotten and keeping Partners up to date on who is available. – Craig Little was the genius behind this one!

This is a real time saver and will hopefully stop all of the excel spreadsheets to monitor time off/ holidays as this is visible and calculated on here now.
It should also stop all of the team calendars that you have which are all done manually and take quite a lot of business time to update.

Thank you to Chelsea for the hard work that she has put in to make this a success and please let Lesley or I know should you feel that we could improve this further so that we can consider your requests.

Thank you
Angela

[B8][B9] Throughout this project, my skills have been tested and developed. The project itself was created through the use of C#, ASP.NET and MVC. I have programmed in these languages before, but never a project that required a lot of time and patience. My personal skills for patience and resilience were tested as I was presented with several errors throughout the project. When these errors appeared, I chose to take a step back and re-evaluate the work. This helped me to understand the errors that were appearing and allowed me to fix them. Another struggle I had was the difficulty in finding appealing visuals for the project, I understand that it has to be easy to use but also not boring to look at. I chose to resolve this by spending time online looking at other projects for inspiration. This helped me to create visuals that were appealing but practical.

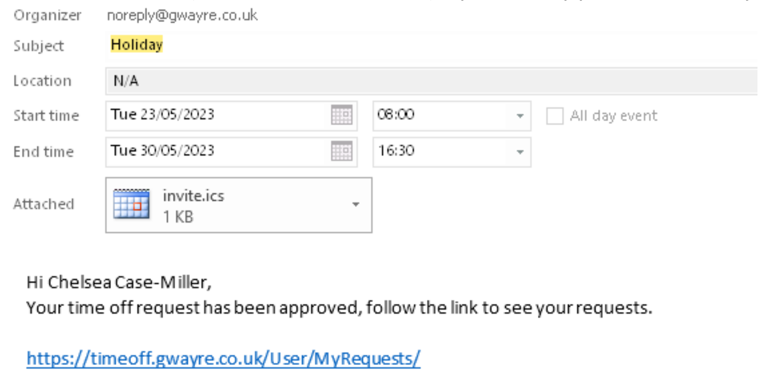


The end product is now in use firm-wide and has improved the way holidays and TOIL are used and booked, both decreasing the workload for HR and making the whole request process smoother and more user friendly.

Additions after release

After the initial release there were many features people had in mind that I have since added. This was the main reason for choosing an agile type workflow as I presumed there would be changes and improvements along the way, and the agile workflow allows me to constantly improve the program with smaller additions each time, the features that I have since added are:

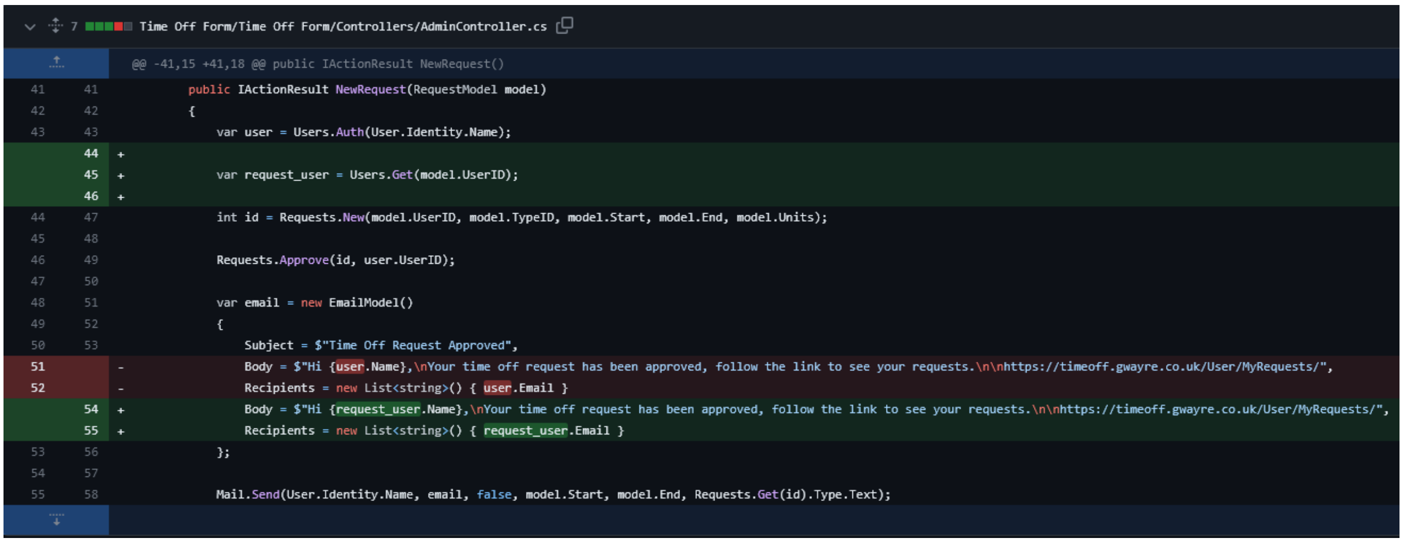
Emails with ICS (calendar invite files) upon an approved holiday.



Organizer	noreply@gwayre.co.uk
Subject	Holiday
Location	N/A
Start time	Tue 23/05/2023 08:00 ☐ All day event
End time	Tue 30/05/2023 16:30
Attached	invite.ics 1 KB

Hi Chelsea Case-Miller,
Your time off request has been approved, follow the link to see your requests.

https://timeoff.gwayre.co.uk/User/MyRequests/



```
7 Time Off Form/Time Off Form/Controllers/AdminController.cs

@@ -41,15 +41,18 @@ public IActionResult NewRequest()
        public IActionResult NewRequest(RequestModel model)
        {
            var user = Users.Auth(User.Identity.Name);
+
+           var request_user = Users.Get(model.UserID);
+
            int id = Requests.New(model.UserID, model.TypeID, model.Start, model.End, model.Units);

            Requests.Approve(id, user.UserID);

            var email = new EmailModel()
            {
                Subject = $"Time Off Request Approved",
-               Body = $"Hi {user.Name},\nYour time off request has been approved, follow the link to see your requests.\n\nhttps://timeoff.gwayre.co.uk/User/MyRequests/",
-               Recipients = new List<string>() { user.Email }
+               Body = $"Hi {request_user.Name},\nYour time off request has been approved, follow the link to see your requests.\n\nhttps://timeoff.gwayre.co.uk/User/MyRequests/",
+               Recipients = new List<string>() { request_user.Email }
            };

            Mail.Send(User.Identity.Name, email, false, model.Start, model.End, Requests.Get(id).Type.Text);
```

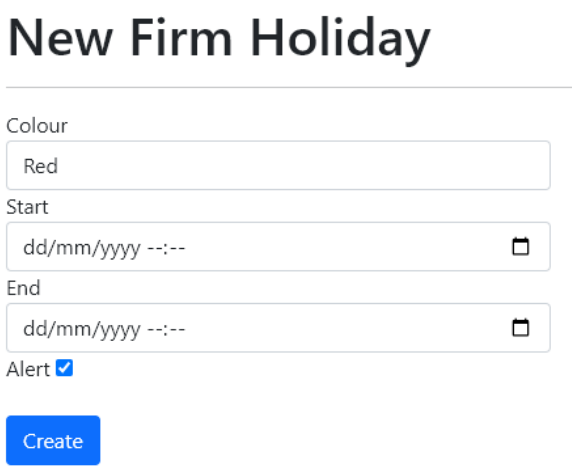
When I originally deployed the feature of sending an automatic email upon the approval of a request, some users had noted that the email contained the approvers name rather than the requesters.

[S7] This generated some confusion about what requests were being approved. Therefore I started a debugging session to find the issue. I had used my knowledge of the programs structure in order to narrow down where the issue was present. I had found where I had implemented the creation of the email object and used my breakpoint before this so that I could step into each line and analyse the variables via the use of the watch list. This showed me what variables contained the incorrect name and I had found that when developing I had used the current user, who in this case would be the approver, and used their name in the email rather than the requester.

To change this I used the data access layer to read the user details from the request model, and used string interpolation to add this to the email, whist also removing the incorrect variable for the approvers' name.

Firm-Wide Holidays with coloured identifiers

[B8] There was a need for a solution to add firm-wide holidays to the calendar, so I leveraged the DayPilot tools to add coloured columns to the calendar view based on data stored in the database, I had the idea of colouring the holidays in different colours as when creating this there was a unique holiday of the kings coronation which needed to be recorded as a different type of holiday since this was not included in staffs PTO, this helps staff understand where their PTO is going and when they are off.



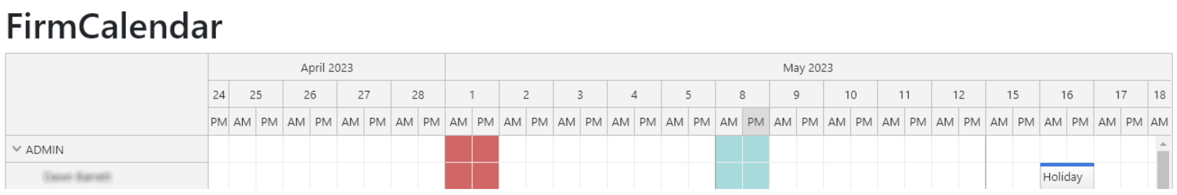
New Firm Holiday

Colour: Red

Start: dd/mm/yyyy --:--

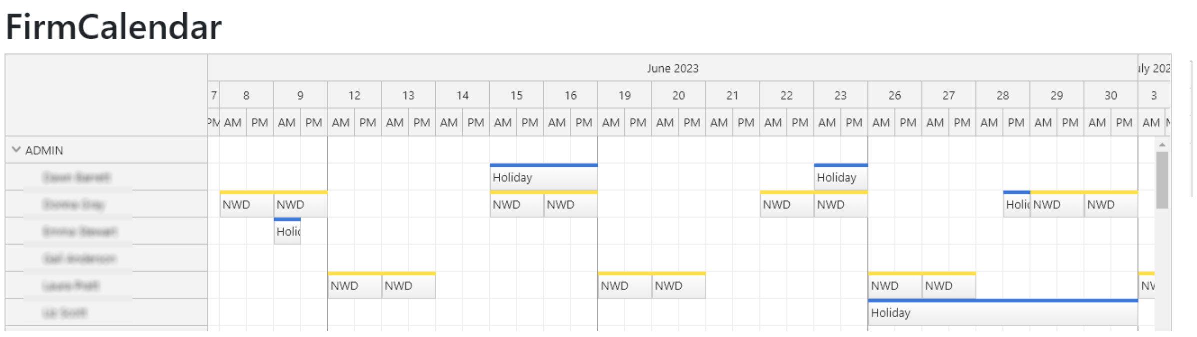
End: dd/mm/yyyy --:--

Alert ✓



FirmCalendar

Link to our staff/client database to provide automatic view of non-working days (NWD) for part time staff.



FirmCalendar

[B9] Although this project is currently considered complete, there are many potential possibilities for additional features and development which, if requested by the HR department as needs or wants, I will be ready to develop using the continued agile type methodology.