The Appointed

Will Frank | Aaron Templeton | Clyde Gustafson

Abstract

Our final project is a calendar/scheduler web application. Users are able to register and log in to their accounts, and add recurring and one-time appointments, and view them in a calendar. Users can right click anywhere on the calendar or click a button to add an appointment at any time. Either method of adding an appointment opens a modal form where the user can fill in the details for their scheduled event. There are also Locations and Categories for each event, which are defined by the user. These are all rather standard, and useful, features of a calendar program. Most things in life occur just once or in a regular weekly pattern, which our program provides for. Additionally the categories and locations provide color for appointments on the calendar, making it readable at a glance.

URLs

Will Frank: ec2-54-227-183-25.compute-1.amazonaws.com
Clyde Gustafson: ec2-54-175-209-137.compute-1.amazonaws.com
Aaron Templeton: ec2-54-161-115-210.compute-1.amazonaws.com

Introduction

Our final project is a calendar and scheduling web application tool designed for personal and work related use. The application was built using Microsoft's Entity Framework Core in ASP.Net. The application consists of a web client for user interaction and a MSSQL database to store all persistent application and user data.

User's that desire to use the calendar and scheduling application will be able to register with their email and a password of the choice. Each user will have associated calendar appointments, categories and locations.

A User will be able to create categories for appointments with a category name and a category color. A category will assist the User to organize and view their appointments in a reasonable and organized manner. When a category is applied to an appointment, the border color of the appointment is changed to the category color which will help a User understand which appointments are in which categories.

A User will also be able to create locations for appointments in the same manner as a category. A location will be supplied with a name and a color. When a location is applied to an appointment, the appointment's background color changes to the location color. The color will help users understand which appointments are at similar locations.

A user will be able to add an appointment by clicking on the "add appointment" button on the calendar and filling in the modal form that pops up. It will ask for a title, location, category, description, start and end time, and a recurrence type for the appointment. Once the appointment is added, it will be populated into the calendar. A user will be able to edit an appointment and view

details by left or right clicking on an appointment. A left click will bring up a modal for viewing the appointment details, while a right click will show a context menu for editing and deleting appointments. Appointments can also be edited by drag-and-drop to a new date or by resizing the appointment in the week or day view in the calendar.

A recurrence for an appointment can be set for daily recurrence, custom weekly recurrence or none. When a recurrence is set, the appointment is added a single time to the database and the calendar handles the recurrence of the appointment to a date specified. Recurring events can be edited just like regular appointments. They also have drag-and-drop functionality if a user wants to move a recurring appointment to a different set of dates.

The calendar is built using FullCalendar javascript library. It is a free and open-source library. The calendar has views for month, week and day and can also change views by clicking the appropriate arrow buttons. The calendar handles all of the rendering of the appointments, the drag-and-drop functionality and the resize functionality.

Last but not least, a User has some basic settings for changing their email and password. It is accessible by clicking on the user email on the navigation bar at the top of the screen.

Feature Table

Feature Name	**Scope**	**Primary Programmer**	**Time Spent**	**File/Function**	**LoC**
Calendar	Front End	Will	1 hour	calendar.js	40
Appointments	Entire App	Clyde	3 hours	Calendar.js, Appointments Controller	150

Async CRUD for Appointments	Front end	Clyde	4 hours	calendar.js	250
Modals for Appointments	Front End	Aaron	3 hours	Home/index, calendar.js	250
User Accounts	Back End	Aaron	2 hours	Identity files, controllers	100
Recurring Appointments	Front End/Entire App	Will and Aaron	10 hours	calendar.js, Appointments Controller	100
Categories	Whole App	Will and Aaron	4 hours	Categories Controller	250
Locations	Whole App	Will and Aaron	4 hours	Locations Controller	250
Drag-n-drop events	Whole App	Aaron	3 hours	calendar.js, appointment controller	80
Context Menu	Front End	Will	5 hours	calendar.js	50
Resize events	Whole App	Aaron	2 hours	calendar.js, appointment controller	75
UI Design	Front End	Aaron	2 Hours	Layouts and view files, css files	200
Tutorial/Overview	Front End	Clyde	1 hour	Index.cshtml	90

Individual Contributions

Team Member	Total Time	Total Loc
Aaron Templeton	30 hours	~1000
Clyde Gustafson	30 hours	~500
Will Frank	30 hours	~700

Summary

Our team's project was to create an online calendar, similar in features to most any other calendar. To do so we leveraged an existing javascript library called FullCalendar. In conjunction with a couple other small javascript extensions and our server's back end we were able to provide all our desired features. Overall our project is probably at a "good" level. However we did face a few obstacles, most notably with Entity Framework. For whatever reason foreign keys were a massive headache for Identity users, which caused issues in seeding the database and using it normally. Another persistent obstacle was having to transform the data between the format needed by the FullCalendar library to display events, and the format we were using to store our data. Even though we tried to mirror our database design closely to FullCalendar's data types, we didn't get it exactly right and at that point there was too much built up to make it worth changing. Many hours were also spent on debugging our asynchronous requests and researching the documentation of FullCalendar.

Our final most prominent features are our calendar, one time and recurring events, the ability to easily edit events, and user-defined locations and categories for the events. Together these make a fully functional online user-based calendar.