



READ ME Last updated February 13 2023

Pauly is a hub for all things school-related.

Build by Andrew Mainella

Saint Paul's High School Student Council 2023-2024

[!IMPORTANT] Functions repo can be found at <https://github.com/Archimedes4/Pauly-Functions>

Technologies

Power by expo using expo router at its core. Application also uses redux for storage.

Backend

Pauly uses Microsoft graph and SharePoint lists as a database

Structure

Calendar

The calendar is organized into documents of years containing collections of months. These months are formatted as a number based on January being 1. Inside each collection documents of days are inside. Each day contains the day, month, and year. As well as the school day and schedule id. If the schedule does not have a value it is schedule one (default schedule).

Commissions

Commissions are organized into documents that are named with their commission id. In each document, there is a

1. Start date
2. End date
3. Hidden
4. Points (what the commission is worth)
5. Selected Page (which page is shown)
6. Value (the type of commission it is)
 1. Approved by issuer
 2. Location
 3. Image
 4. Image and Location
 5. QR Code

Notifications

The notifications page has a board, a message, insights (used and trending) and tasks. The tasks are from microsoft graph using the todo api. The insights are from microsoft graph the.

Refrences

TODO: <https://learn.microsoft.com/en-us/graph/api/resources/todo-overview?view=graph-rest-1.0>
Insights (used, trending): <https://learn.microsoft.com/en-us/graph/api/resources/officegraphinsights?view=graph-rest-1.0>
Board: <https://learn.microsoft.com/en-us/graph/api/driveitem-get-content-format?view=graph-rest-1.0&tabs=http>

Resources

The resource page takes teams posts and displays them. It has access to files and sections. Resource has a news section powered by the wordpress api. Resources has a scholarship database powered by the raindrop api.

Sports

This is the sports page that shows sport highlights and has team rosters. The sports page has embeded youtube videos from the Saint Paul's High School youtube page. It also as videos and images uploaded through government.

Format

Colors

name	value
white	"white"
light gray	"#ededed"
dark gray	"#444444"
maroon	"#793033"
warning orange	"#FF6700"

Extensions

NOTE Extensions are automatically setup in the initialization process

```
{
  "id": "paulyEvents",
  "description": "Pauly Event Data",
  "targetTypes": [
    "Event"
  ],
  "owner": {application id},
  "properties": [
    {
      "name": "eventType",
```

```
        "type": "String"
      },
      {
        "name": "eventData",
        "type": "String"
      }
    ]
  }
}
```

Setup

How to setup Pauly

Step #1

Commands to setup Azure services

```
az login
```

<https://learn.microsoft.com/en-us/cli/azure/ad/app?view=azure-cli-latest#az-ad-app-create>

Save appId for use later

```
az ad app create --display-name Pauly
```

<https://learn.microsoft.com/en-us/cli/azure/ad/signed-in-user?view=azure-cli-latest>

Save userId for use later

```
az ad signed-in-user show
```

<https://learn.microsoft.com/en-us/cli/azure/ad/app/owner?view=azure-cli-latest#az-ad-app-owner-add>

```
az ad app owner add --id {appId} --owner-object-id {userId}
```

<https://learn.microsoft.com/en-us/cli/azure/ad/app?view=azure-cli-latest#az-ad-app-update>

update azure ad app

```
az rest --method PATCH --uri
'https://graph.microsoft.com/v1.0/applications/{id}' --headers 'Content-
Type=application/json' --body "{spa:{redirectUris:
['http://localhost:19006/auth', 'https://paulysphs.ca',
'https://www.paulysphs.ca']},publicClientApplication: {redirectUris:
['com.Archimedes4.Pauly://auth']}, signInAudience: \"AzureADMyOrg\"}"
```

Create Static Web App

<https://learn.microsoft.com/en-us/azure/static-web-apps/get-started-cli?tabs=vanilla-javascript>

Note: The resource group that you use will be tied to the web app. Some other resource group can be used but this method is preferable.

Create Resource Group

```
az group create \  
--name Pauly-SWA \  
--location "eastus2"
```

Create static web app

```
az staticwebapp create \  
--name Pauly-Static-Web-App \  
--resource-group Pauly-SWA \  
--source https://github.com/AMCanada16/Pauly.git \  
--location "eastus2" \  
--branch master \  
--app-location "/web-build" \  
--output-location "build" \  
--login-with-github
```

Create Azure Functions App and deploy functions

Create Function app and storage account

<https://learn.microsoft.com/en-us/cli/azure/storage/account?view=azure-cli-latest#az-storage-account-create>

```
az storage account create --name paulystorage --resource-group Pauly-SWA
```

Create function app

<https://learn.microsoft.com/en-us/cli/azure/functionapp?view=azure-cli-latest#az-functionapp-create>

```
az functionapp create --name Pauly-Functions --resource-group Pauly-SWA  
--storage-account paulystorage --consumption-plan-location eastus2 --  
runtime node --functions-version 4
```

Allow cors for function app

<https://learn.microsoft.com/en-us/cli/azure/functionapp/cors?view=azure-cli-latest>

Note: <http://localhost:19006> is for development purposes and can be removed.

```
az functionapp cors add -g Pauly-SWA -n Pauly-Functions --allowed-origins https://www.paulysphs.ca https://paulysphs.ca http://localhost:19006
```

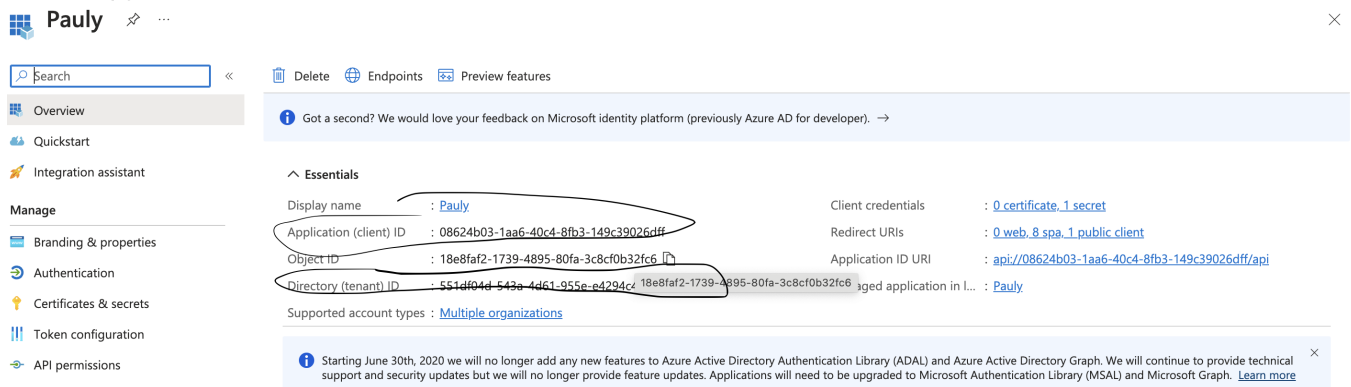
Set Environment Variables

<https://learn.microsoft.com/en-us/cli/azure/functionapp/config/appsettings?view=azure-cli-latest#az-functionapp-config-appsettings-set>

Replace the values in the command. To get the tenant id, client id and client secret go to portal.azure.com.

-Microsoft Entra Id -> App Registrations -> Pauly

Then copy the client Id and Tenant Id



Pauly

Search

Delete Endpoints Preview features

Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer). →

Essentials

Display name : Pauly

Application (client) ID : 08624b03-1aa6-40c4-8fb3-149c39026dff

Object ID : 18e8faf2-1739-4895-80fa-3c8cf0b32fc6

Directory (tenant) ID : 551d404d-543a-4d61-955e-e4294c18e8faf2-1739-4895-80fa-3c8cf0b32fc6

Client credentials : 0 certificate, 1 secret

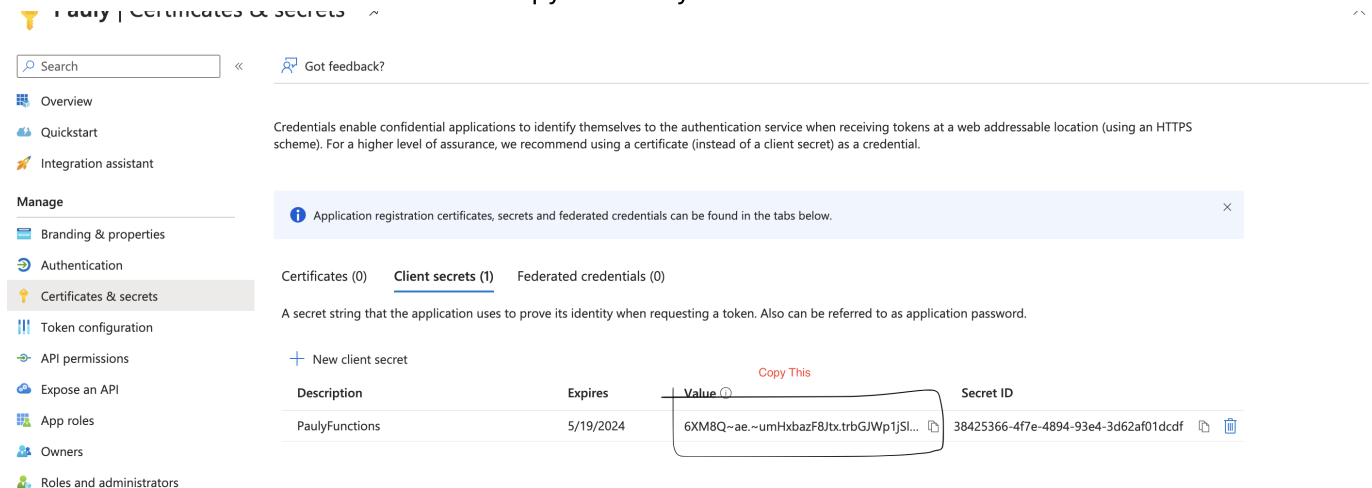
Redirect URIs : 0 web, 8 spa, 1 public client

Application ID URI : api://08624b03-1aa6-40c4-8fb3-149c39026dff/api

Supported account types : Multiple organizations

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)

Then navigate to Certificates & Secrets. Click "New Client Secret" the set the description to "Pauly Functions" or whatever. Press add and copy the newly created client secret and add it to this function.



Pauly | Certificates & secrets

Search

Got feedback?

Overview

Quickstart

Integration assistant

Manage

Branding & properties

Authentication

Certificates & secrets

Token configuration

API permissions

Expose an API

App roles

Owners

Roles and administrators

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) Client secrets (1) Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Secret ID
PaulyFunctions	5/19/2024	6XM8Q~ae~umHxbazF8Jtx.trbGJWp1jSl...	38425366-4f7e-4894-93e4-3d62af01dcdcf

```
az functionapp config appsettings set --name Pauly-Functions --resource-group Pauly-SWA --settings "CLIENTID={ClientId goes here (remove brakets)} TENANTID={TenantId goes here} CLIENTSECRET={client secret goes here}"
```

Attach to github workflow

<https://learn.microsoft.com/en-us/cli/azure/functionapp/deployment/github-actions?view=azure-cli-latest>

```
az functionapp deployment github-actions add --resource-group Pauly-SWA
--repo "https://github.com/AMCanada16/Pauly" --name Pauly-Functions --
login-with-github --build-path "/api"
```

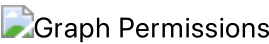
Step #2

Pauly has a config file named PaulyConfig which contains three values these values need to be apart of the main config file inorder for Pauly to work.

- 1. tenant id
- 2. Client ID (the ID of Paulys application)
- 3. org id (the id of Pauly's group)

Setp #3 Initilize Pauly

Go to the admin panel and initilize pauly. Code will break if Pauly is already initilized. If hard reseting Pauly delete Pauly group and Pauly extensions. This can all be done in the graph section of Government.



Maintenance

Rotating client secrets after 160 days. Follow the steps in Set Environment Variables no need to change teant id and client id tho.

Graph Permissions

Reference <https://learn.microsoft.com/en-us/graph/permissions-reference>

Why Consent Is Needed <https://learn.microsoft.com/en-us/graph/api/resources/consentrequests-overview?view=graph-rest-1.0>

All permissions are Delegated permissions

Users

Permission	Admin Consent Required	Description
User.Read	NO	Allows users to sign-in to the app, and allows the app to read the profile of signed-in users. It also allows the app to read basic company information of signed-in users.

Permission	Admin Consent Required	Description
User.ReadBasic.All	NO	Allows the app to read a basic set of profile properties of other users in your organization on behalf of the signed-in user. This includes display name, first and last name, email address, open extensions and photo. Also allows the app to read the full profile of the signed-in user.
People.Read.All	YES	Allows the app to read a scored list of people relevant to the signed-in user or other users in the signed-in user's organization. The list can include local contacts, contacts from social networking or your organization's directory, and people from recent communications (such as email and Skype). Also allows the app to search the entire directory of the signed-in user's organization.
ChannelMessage.Read.All	YES	Allows an app to read a channel's messages in Microsoft Teams, on behalf of the signed-in user.
Channel.ReadBasic.All	NO	Read channel names and channel descriptions, on behalf of the signed-in user.
Calendars.ReadWrite	YES	Allows the app to create, read, update, and delete events in user calendars.
Team.ReadBasic.All	NO	Read the names and descriptions of teams, on behalf of the signed-in user.
Tasks.ReadWrite	NO	Allows the app to create, read, update, and delete the signed-in user's tasks and task lists, including any shared with the user.
Sites.Read.All	NO	Allows the app to read documents and list items in all site collections on behalf of the signed-in user.
Group.ReadWrite.All	YES	Allows the app to create groups and read all group properties and memberships on behalf of the signed-in user. Also allows the app to read and write calendar, conversations, files, and other group content for all groups the signed-in user can access. Additionally allows group owners to manage their groups and allows group members to update group content.

Government

Permission	Admin Consent Required	Description
------------	------------------------------	-------------

Permission	Admin Consent Required	Description
Application.ReadWrite.All	YES	Allows the app to create, read, update and delete applications and service principals on behalf of the signed-in user.
Sites.Manage.All	YES	Allows the app to manage and create lists, documents, and list items in all site collections on behalf of the signed-in user.
TeamMember.Read.All	YES	Read the members of teams, on behalf of the signed-in user.

General Reference

File Structure

[!IMPORTANT] The public folder needs pdfjs to run and the code in components -> pdf -> index.web needs to be updated.

- `_tests_`: Holds tests
- `assets`: Holds images and fonts
- `public`: Holds manifest, well-known and pdfjs to remove a need for a cdn.
- `src`: holds apps code
 - `app`: file structure for expo router
 - `components`: rendered functions that aren;t a page.
 - `hooks`: hooks
 - `redux`: storage reducers and actions
 - `utils`: functions that aren't rendered
 - `constants`: holds types, colors, and some constants
- `rest`: config files env files readme

Oddities

Some things were needed in order to get things working without errors. Here is a list of them please check that these patches are actually needed.

1. React Native Reanimated (fixed in react native reanimated 3.5) Expo is compatable with React Native Reanimated 3.3.

Error: requestAnimationFrame is not defined

Issue: <https://github.com/expo/router/issues/718> Patch: <https://gist.github.com/javascripter/4e4e20e9024a33592437648b718c763d>

2. Expo Fonts Expo fonts and React Native Paper aren't playing nice. Taking over the 3 second threshold to load.

Issue: <https://github.com/expo/expo/issues/12382> Patch: Inside Issue very simple chaning the timeout

Functions

SubmitCommission - Used to generate a valid commission submission. Because users cannot write to Pauly the function firsts validates the commissions submission and adds data like the date. Doing all this with the users credentials. Then users an application adds the submission.

SyncCalendarOrchestator - A durable function composed of activities that sync a iCalendar with Paulys Calendar. The function is mostly idempotent and uses apis that have the same output no matter when called.

Known limitations: -For loops, find and some are used throught the code. Therefore, if many thousands of events were to be tracked the function would fail. -As for being idempotent, if the function was called many times at once and had to create events, the function would create multiple events. This is because the function looks for events that exist and then creates them, but all the orchestrates would have calculated that an event needs to be created and therefore would create events. As long as the orcestrator is not spammed and called many times while other orcestrations are running it will work. It also does not matter when azure decideds to run the function, if the function is run well after it is called it still works. The only limitation in that regard is the token expiering and therefore, the function would fail and could be run again without issue.