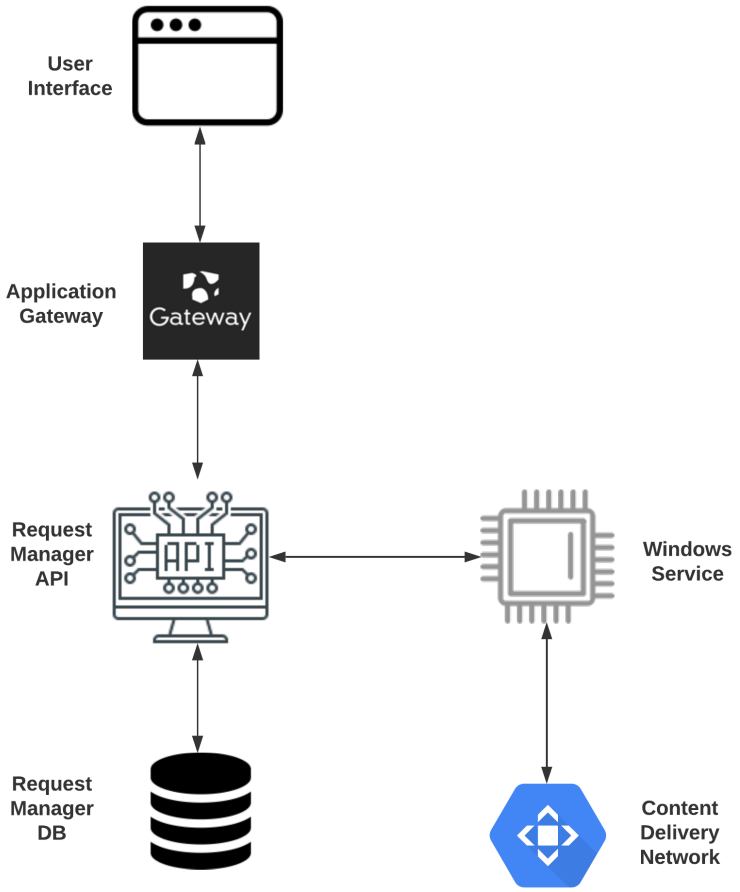
Background Request Processing System

WORK IN PROGRESS

Background information

As we introduce more and more features in the product, the need for different types of reports come along with it. An aggregated report for a single classroom for few students and for a single week can be generated relatively easily and on the fly but generating reports at the school level for all the students and for the whole academic year is challenging. The goal of this document is to outline a high-level system architecture that can help us achieve on-demand background processing of different user's requests and can be used from different parts of the 3P system.

High-Level System Architecture



Core Components of this system

Request Manager API
A generic API to keep track of different types of user requests, its details, status and any artifacts the request processors creates. This API will be used by different application gateways and different request processor windows services.

Windows Service
Purpose build request processor service which can process one or multiple types of requests. It coordinates with different APIs and data sources to process the specific requests. It is responsible for updating the status of the requests, creating and saving the artifacts in CDN if there is any.

Responsibilities of different components

Request Manager API

- Generic API to submit and retrieve user's requests based on user id and request type.
- Each request contains information about the submitter, request type (GenerateSpellingReport, GenerateUPFReport etc), request details (json data containing the details of the request).
- Keeps track of the status (Not started, In-Process, Finished, Failed, etc) of a request.
- Keeps records of the artifacts for requests.
- keeps a record of the error message if the request processor failed to process the request.
- Provides the functionality for a request to run periodically - Future enhancement.

Request Processor Windows Service

- Periodically pull information from the request manager API to retrieve new unprocessed requests.
- Retrieve data from different APIs and data sources to process the request.
- If an artifact is expected for that specific type of request, the service will create the artifacts, saves it in CDN, and update the request details through the request manager API.
- If the service failed to process the request, it updates the request details with the failure reason through the request manager API.

API Endpoints

Endpoint	Verb	Note	Request Body	Response
/requests	POST	Submit new request	`{ "type": "GenerateSpellingReport", "title": "3A Weekly Report", "profileId"" 2345, "requestDetail": "{json}" }`	`{ "requestId": "2a4611b1-a6d8-4c45-908e-570e8b5596ea" }`

/requests/{requestType}/{userProfileId}	GET	Retrieve a single type of existing requests for a specific user		[
				{
				"requestId": "2a4611b1-a6d8-4c45-908e-570e8b5596ea",
				"title": "3A Report",
				"artifacts": [
				"http://west.cdn.mathletics.com/GenerateSpellingReport/{requestId}/report.xlsx",
				"http://west.cdn.mathletics.com/GenerateSpellingReport/{requestId}/report_combined.xlsx"
				],
				"status": 2,
				"errorMessage": ""
				}
				]

/requests/{requestId}	PUT	Update the status, artifacts, error message of a requests	{
			"artifacts": [
			"/GenerateSpellingReport/{requestId}/report.xlsx",
			"/GenerateSpellingReport/{requestId}/report_combined.xlsx"
			],
			"status": 2,
			"errorMessage": ""
			}	

Important note

- The API should allow a certain number of unfinished requests for a single user. In short, it should work as a queue. Once the queue is full, until one of the requests from the queue is processed, the user should not be able to submit another request. The queue should be based on the request type and user.
- This polling approach can be replaced by a pub-sub strategy once we have a proper message bus infrastructure in place.