Overview

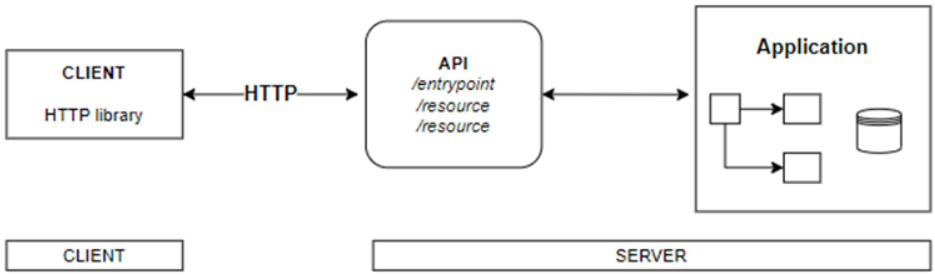
Smart devices such as voice controllers, security lights, smart locks and Wi-Fi-enabled devices can communicate and exchange data over the Internet. Devices form distributed ecosystems that can perform environmental monitoring of homes and buildings. The exposure of IoT device capabilities and data through the Web is fundamental. An IoT Device Management System helps you register, organise, monitor, and remotely manage IoT devices at scale as IoT deployments range in size, complexity, objectives and requirements.

An IoT Device Management System keeps track of the whereabouts of all IoT devices deployed by the organisation. Depending on the type of organisation, different categories of devices are used. Each IoT device is initially categorised and registered. Then, IoT devices are deployed throughout the organisation's buildings in predefined zones. Administrators can view all IoT devices, update their properties, add new devices and move them to other zones.

Representational State Transfer (REST) is a model and architectural style for web services over HTTP. When this model is used for API design, IoT devices can be managed using the Cloud. Therefore, the IoT Device Management System should be implemented as a set of RESTful APIs.

RESTful APIs and services are used as a good practice in the industry to transport data between systems, environments, interfaces and applications. One of the most common use cases is connecting a RESTful API to a data source to manage interaction with the data source. This approach lends many advantages to a solution mostly addressing longevity. RESTful APIs commonly consist of different methods that allow for retrieval of data and manipulation of data (creation, amendment and deletion).

As part of this project, you will create a CRUD RESTful API that will connect to a database storing IoT device data. The API should contain at least one get, post, patch and delete method per resource – aligning to the project's requirements. The RESTful API architecture has several endpoints called over HTTP, invoking application code to update a database.



Prerequisites

Before executing on this project, you will need to take the following into account and action the items appropriately:

- Ensure you can access the NWU Azure tenant by logging into the Azure Portal using your MS Fed account: 12345678@student365.msfed.nwu.ac.za
- Ensure that you have created a resource group to logically group your work. Use the appropriate naming convention
- Ensure that Visual Studios 2022 Community edition and .NET Core 3.1 are installed

Requirements

Functional requirements refer to the functionality that a system must have and how the functions should be performed. Non-functional requirements refer to the aspects of a solution that have an impact on the quality attributes of a system (or platform). These non-functional requirements are deemed as supportive requirements to ensure that the functional requirements are implemented appropriately and according to good software practices.

Please note: *it will be important for you to keep the Overview Repository ReadME file updated throughout the semester as you will be evaluated on the content of the ReadME file as part of your Portfolio of Evidence (POE).*

Feature	**Stories**	**Tasks (to be broken down further)**	**Priority**
GitHub Administration	Create and Configure GitHub Repository	Create a repository named 'CMPG 323 Project 2 - <add your student number>'	1
		Create a ReadME.md file that will be used to describe your project and how stakeholders are to use the report that you have developed	1
	Project Progress	Ensure that the report has been committed and pushed to source control throughout the project	1
		Ensure that the GitHub project has been updated iteratively throughout the project to demonstrate how progress was made	1
Prepare the Data Source	Configure the Database	Create a SQL Server with a secure service account username and password on Azure	1
		Create a F1 tier (free) database on the Azure SQL Server	1
		Run the provided SQL script that will create the relevant tables	1
Project Setup	Create the API project	Clone your GitHub repository	2
		Create a new .NET Core Web API project	1

	Connect the API to the data source	Scaffold the database into the project (connect the project to the database)	1
		Apply dependency injection - add the scaffolded DBContext to the Startup.cs	1
Functionality	Build Device Management Functionality	Create a GET method that retrieves all Device entries from the database	2
		Create a GET method that will retrieve one Device from the database based on the ID parsed through	2
		Create a POST method that will create a new Device entry on the database	2
		Create a PATCH method that will update an existing Device entry on the database	3
		Create a DELETE method that will delete an existing Device entry on the database	3
		Add a private method in the API that checks if a Device exists (based on the ID parsed through) before editing or deleting an item	4
	Build Zone Management Functionality	Create a GET method that retrieves all Zone entries from the database	2
		Create a GET method that will retrieve one Zone from the database based on the ID parsed through	2
		Create a POST method that will create a new Zone entry on the database	2
		Create a PATCH method that will update an existing Zone entry on the database	3
		Create a DELETE method that will delete an existing Zone entry on the database	3
		Add a private method in the API that checks if a Zone exists (based on the ID parsed through) before editing or deleting an item	4
		Create a GET method that retrieves all devices within a specific zone (based on the zone ID that is parsed through)	5

	Build Category Management Functionality	Create a GET method that retrieves all Category entries from the database	2
		Create a GET method that will retrieve one Category from the database based on the ID parsed through	2
		Create a POST method that will create a new Category entry on the database	2
		Create a PATCH method that will update an existing Category entry on the database	3
		Create a DELETE method that will delete an existing Category entry on the database	3
		Add a private method in the API that checks if a Category exists (based on the ID parsed through) before editing or deleting an item	4
		Create a GET method that retrieves all devices within a specific category (based on the category ID that is parsed through)	5
		Create a GET method that will return the number of zones that are associated to a specific category (use the device entity to join the data)	6
Project Close-out	Security	Ensure that authentication has been set up to restrict access to the API	1
		Ensure that no credentials are stored on GitHub	1
	Web API Cloud Hosting	Create an API Service (connected to an F1 tier (free) service plan)	4
		Publish your API to the service hosted on Azure and ensure the API is secure and accessible	4
	Project Documentation	Ensure that the ReadMe.md file in the GitHub repository explains how the user would use the API	1
		Create a reference list document that contains all sites visited and used to complete the project	1

Reading Materials

There are multiple aspects of the abovementioned scope that may be covered by

- Tutorial: Create a web API with ASP.NET Core | Microsoft Docs
- Create a web API with ASP.NET Core controllers - Learn | Microsoft Docs
- ASP.NET Core web API documentation with Swagger / OpenAPI | Microsoft Docs
- Create microservices with .NET and ASP.NET Core - Learn | Microsoft Docs
- Entity Framework Core 3.1 - Getting Started
- Join two entities in .NET Core, using lambda and Entity Framework Core
- Publish an ASP.NET Core web API to Azure API Management with Visual Studio | Microsoft Docs
- Interesting read: Automating ASP.NET Core Web API Creation That Communicates with Your Database in 60 Seconds or Less

Community Engagement

There are many different communities available for you to engage with if you are experiencing any challenges or if you would like to learn more about the technology and possibilities of API Development and Integration:

- LinkedIn Groups
- Stack Overflow
- Microsoft Developer Community User Groups
- YouTube Microsoft Development Influencers

Submission Details

The scope of this project has been issued as an **individual** assignment. Please note that you will need to use GitHub for this project.

Please Note: Ensure that your repository has been shared with the users **autoruby, JacquiM** and **marijkec** so that your project can be marked**.**

Submission: Submit your CMPG 323 Project 2 by providing the relevant information through the Project 2 Submission Form.

Deadline: 17h00 on 8 September 2022 (please note there are no alternative or late submission dates – if you miss this deadline you will forfeit the opportunity)

What to submit:

1. Provide the URL to your GitHub Repository
2. Credentials to connect to your API
3. Provide the URL to your API
4. Provide your reference list file

Marking Considerations

Please take note of the following considerations that will form part of the marking and moderation process:

- A rubric will be provided separately
- Failure to upload any of the requirements for submission will result in 0
- Failure to complete this as an individual assignment will result in 0

- Failure to use .NET Core and Swagger will result in 0.