# **Project 2** (Due 31 August)

# Overview

*EcoPower Logistics* sells a wide range of parts and components essential for installing, functioning, and optimising solar energy systems. These parts are designed to harness and convert solar energy into usable electricity. The products that the company offers are, for example: solar panels, mounting systems to securely install solar panels on rooftops, inverters that convert DC electricity produced by solar panels into alternating current (AC) electricity, batteries, and monitoring and control systems to allow users to track the performance, energy production, and overall health of their solar energy systems.

*EcoPower Logistics* specialises in managing and optimising the intricate processes involved in moving goods, encompassing warehousing and transportation, from the initial source to the final destination based on individual customer requirements. *EcoPower Logistics* comprehensive logistics solutions tackle the complexities of acquiring, storing, and transporting resources and goods globally, requiring meticulous coordination. Every facet of logistics, from transportation to packaging, inventory management, and warehousing, must operate seamlessly and in perfect harmony. Through our logistics management system, *EcoPower Logistics* receive and process orders, while the warehouse serves as a secure storage facility.

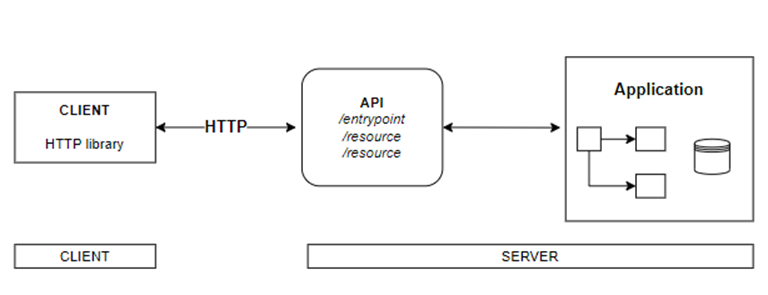
Prior to transportation, all orders placed by customers undergo meticulous packaging to ensure the utmost security and proper labelling of goods. Each order a customer places contains several solar products to install a solar facility at a customer location. *EcoPower Logistics* prioritises the safe delivery of solar products by inspections of vehicles to detect any pre-existing faults or damages that could potentially impact the condition of the goods, particularly on longer routes. EcoPower Logistics assigns tracking IDs to every order to enhance efficiency at every stage, facilitating seamless scanning and processing and providing users with access to real-time order status updates.

Given the ever-growing reliance on logistics to sustain businesses and drive economic activities, *EcoPower Logistics* recognise the imperative of uninterrupted operations and increasingly leverages technology to adapt and enhance services. Few industries hold as much potential for API integration as the logistics sector. From Amazon to local, family-owned enterprises, all players within the logistics ecosystem, including shippers, couriers, and delivery services, stand to reap significant advantages by utilising a logistics API. APIs offer many benefits, from fostering business agility to improving the overall customer experience. By seamlessly integrating API solutions into their operations, logistics companies can unlock new opportunities for growth, efficiency, and service excellence.

Representational State Transfer (REST) is a model and architectural style for web services over HTTP. EcoPower Logistics services can be managed using the cloud when this model is used for API design. Therefore, the *EcoPower Logistics* 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 CRUD methods that allow for retrieval of data and manipulation of data (create, read, update and delete).

As part of the *EcoPower Logistics* project, you will create a CRUD RESTful API that will connect to a database storing basic logistics 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](https://portal.azure.com) using your MS Fed account: [12345678@student365.msfed.nwu.ac.za](mailto: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 Studio 2022 Community edition and .NET Core 6 are installed

# Requirements

Functional requirements refer to a system's functionality 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 essential 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).*

You are expected to implement a CRUD RESTful API following the instruction provided next. You have previously created a GitHub repository for Project 2 where all code will be stored. Be aware that GitHub will record all updates to your code (we will know when you started your project and what you did 😊).

Ensure you have attended all classes and made yourself aware of all requirements and documentation available on eFundi.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Stories** | **Tasks (to be broken down further)** | **Priority** |
| GitHub Administration | Create and Configure GitHub Repository | Create a repository or use the one created in project 1 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 Customer Management Functionality | Create a GET method that retrieves all Customer entries from the database | 2 |
|  |  | Create a GET method that will retrieve one Customer from the database based on the ID parsed through | 2 |
|  |  | Create a POST method that will create a new Customer entry on the database | 2 |
|  |  | Create a PATCH method that will update an existing Customer entry on the database | 3 |
|  |  | Create a DELETE method that will delete an existing Customer entry on the database | 3 |
|  |  | Add a private method in the API that checks if a Customer exists (based on the ID parsed through) before editing or deleting an item | 4 |
|  | Build Order Management Functionality | Create a GET method that retrieves all Order entries from the database | 2 |
|  |  | Create a GET method that will retrieve one Order from the database based on the ID parsed through | 2 |
|  |  | Create a POST method that will create a new Order entry on the database | 2 |
|  |  | Create a PATCH method that will update an existing Order entry on the database | 3 |
|  |  | Create a DELETE method that will delete an existing Order entry on the database | 3 |
|  |  | Add a private method in the API that checks if an Order exists (based on the ID parsed through) before editing or deleting an item | 4 |
|  |  | Create a GET method that retrieves all orders for a specific customer (based on the customer ID that is parsed through) | 5 |
|  | Build Product Management Functionality | Create a GET method that retrieves all Product entries from the database | 2 |
|  |  | Create a GET method that will retrieve one Product from the database based on the ID parsed through | 2 |
|  |  | Create a POST method that will create a new Product entry on the database | 2 |
|  |  | Create a PATCH method that will update an existing Product entry on the database | 3 |
|  |  | Create a DELETE method that will delete an existing Product entry on the database | 3 |
|  |  | Add a private method in the API that checks if a Product exists (based on the ID parsed through) before editing or deleting a product. | 4 |
|  |  | Create a GET method that retrieves all products for a specific order (based on the order ID that is parsed through) | 5 |
| 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. List all endpoints and additional aspects implemented. | 1 |
|  |  | Create a reference list document using the Harvard referencing style that contains all sites or other sources 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](https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio)
* [Create a web API with ASP.NET Core controllers - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/modules/build-web-api-aspnet-core/)
* [ASP.NET Core web API documentation with Swagger / OpenAPI | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-3.1)
* [Create microservices with .NET and ASP.NET Core - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/paths/create-microservices-with-dotnet/)
* [Entity Framework Core 3.1 - Getting Started](https://procodeguide.com/programming/entity-framework-core-in-asp-net-core/)
* [Join two entities in .NET Core, using lambda and Entity Framework Core](https://jd-bots.com/2022/01/24/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](https://docs.microsoft.com/en-us/aspnet/core/tutorials/publish-to-azure-api-management-using-vs?view=aspnetcore-6.0)
* Interesting read: [Automating ASP.NET Core Web API Creation That Communicates with Your Database in 60 Seconds or Less](http://thejpanda.com/2020/08/10/python-automating-asp-net-core-web-api-creation-that-communicates-with-your-database-in-60-seconds-or-less/)

Badges and Certifications

Complete a badge or a certification before the due date on Agile, Scrum, GitHub and related topics. Many learning opportunities are available that will give you credit for your learning. You must upload evidence with your project submission that you have achieved the badge or certification, and you will receive marks for it. You will again receive marks for all badges and certifications achieved from 21 July until 3 November in your final Portfolio of Evidence. No recognition will be given after the project has been submitted.

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:** Your repository must be set as *private* and only shared with the users **autoruby, JacquiM** and **marijkec** to mark your project**.**

**Submission**: Submit your CMPG 323 Project 2 by providing the relevant information through the form to be provided.

**Deadline**: 17h00 on 31 August 2023 (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 as required
3. Provide the URL to your API
4. Provide a list of all the endpoints you have implemented
5. Provide your reference list file in Harvard referensing style

**Warning:**

In no circumstance should you ever make your GitHib repository or Kanban project *public*. Any student caught doing this will get 0%. The student sharing his code and work is as guilty and the one doing the copying.

# 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.
* Failure to host the project on cloud would result in a loss of 30% of the total achievable mark
* Failure to update the GitHub project would result in a loss of 10% of the total achievable mark
* Failure to implement source control would result in a loss of between 30% and 55% of the total achievable mark