UK Income Tax Calculator Technical Design

Table of Contents

Proposed Design

Due to the simplicity of the requirements, the solution should consist of a monolith architecture that both serves the web site, performs the tax calculations, and retrieves data directly from the database.

API

The tax calculation API should be exposed as a single endpoint *CalculateTax* that accepts an integer value as a query parameter to represent the salary. The response should be a serialized instance of *CalculateTaxResponse*.
Template url: *https://<host>:<port>/TaxCalculator/calculatetax?salary=<integer>*>

CalculateTaxResponse object

Property Name	Type	Description
GrossAnnualSalary	Int	Total salary including tax
GrossMonthlySalary	Decimal	Total monthly salary including tax. *GrossAnnualSalary / 12*
NetAnnualSalary	Decimal	Annual salary excluding tax. *GrossAnnualSalary - AnnualTax*
NetMonthlySalary	Decimal	Total monthly salary excluding tax. *NetAnnualSalary / 12*
AnnualTax	Decimal	Total annual amount of tax paid. *Sum of tax paid per band.*
MonthlyTax	Decimal	Total monthly amount of tax paid. *AnnualTax / 12*

Database

The database will consist of a single table. Each row contains data for a single tax band.

TaxBand table

Column Name	Type	Description
ID	Int	Auto generated row ID
Name	varchar(15)	The name of the tax band
LowerLimit	int	The lowest value that the tax percentage relates to.
UpperLimit	Int	The highest value that the tax percentage relates to. A value of -1 indicates no upper limit.
Rate	int	The percentage tax rate to apply to this salary range

Controller

There should be one controller named *TaxCalculator*, which exposes a single endpoint *CalculateTax* that accepts the GET http method.
The endpoint will accept an integer from the request query string and an instance of *ISender* which provides the functionality of the MediatR library. The body of this method will translate the salary integer into a *CalculateTaxQuery* object and pass that to the MediatR library.
When a request hits the API endpoint, the salary value is passed to the *CalculateTaxHandler* which implements the MediatR *IRequestHandler* interface.
The handler forwards the salary through to the *TaxCalculationService.*

Tax Calculation Service

Accepts an instance of the *ITaxBandRepository* as a constructor parameter.
This service iterates through the tax band details held in the repository and calculates the amount of tax paid for each, adding the results to the appropriate property of the result object.
After the tax has been calculated for all bands, the return object is returned from the handler, which in turn, returns the response to the client.

Third-party components

Applications

Docker
MySQL

Libraries used for implementation

Entity Framework Core
MediatR
Angular

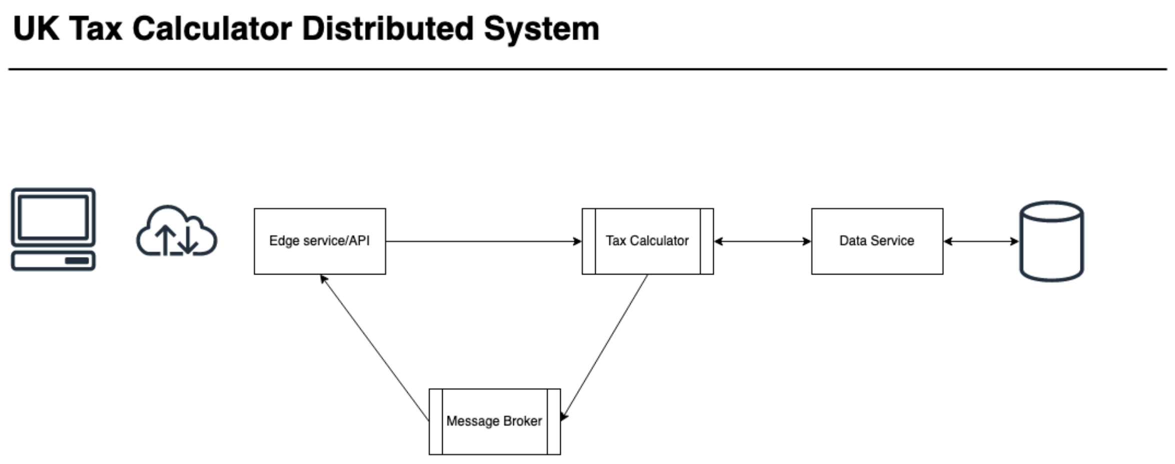
Libraries used for testing

FluentAssertions
Moq

Alternative solutions

To create a more horizontally scalable solution then I would propose the following:

- An edge service that hosts the web site and exposes the API endpoints
- Tax calculator microservice for business logic.
- Data service that translates database tables to domain objects.
    - Exposes RESTful API for communication.
- Message broker used for asynchronous messaging between components (RabbitMQ, AWS etc...)



Edge service

The edge service is responsible for both hosting the SPA web site and exposing the API to client machines.
On receipt of a request, a unique request Id will be generated and, along with the incoming request, will be translated into a domain message and write it to the *TaxCalculationRequest* queue.

The service will also respond to messages written to the *TaxCalculationResponse* queue. Messages written to this queue will contain the results of the tax calculations for distinct requests. Using the unique request Id generated on receipt of the request, the appropriate client will be looked up and the response sent to that client via SignalR.

Tax Calculation Service

This service listens for messages written to the *TaxCalculationRequest* queue.
On notification that a new message is available in the queue, the service will call out to the DataService to retrieve the up to date Tax Band information. Using this data, the service will process the tax calculations and write the results to the *TaxCalculationResponse* queue in a *CalculationResults* message.

Data Service

The data service will expose a REST API that allows callers to access the Tax Band data.
It will be responsible for retrieving the tax band data from the database and translating it to the domain models.