Coding-Assessment

Technical Specification

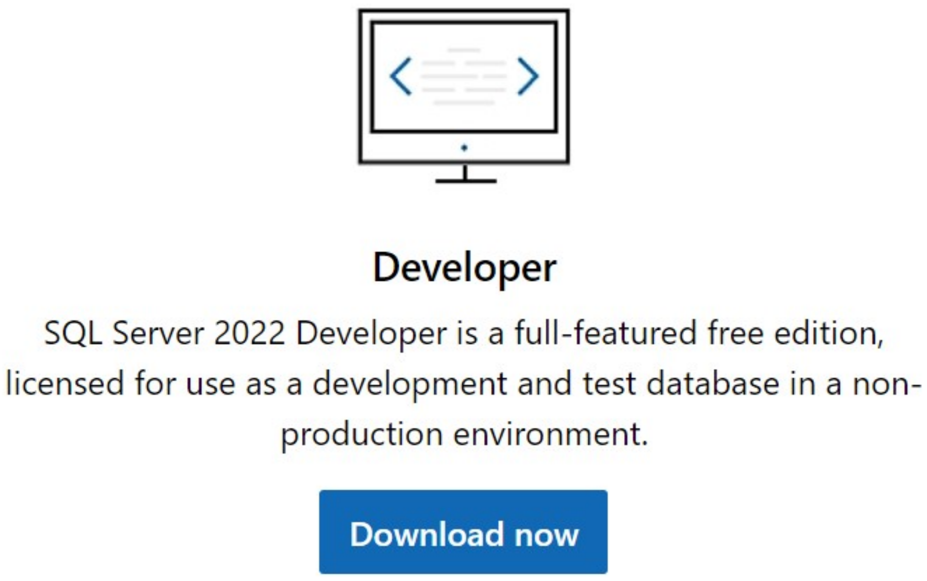
Software Used

- SQL Server 2022 Developer
- Visual Studio 2022 Community
- Node 20+

Instructions on how to Setup the Project

Database

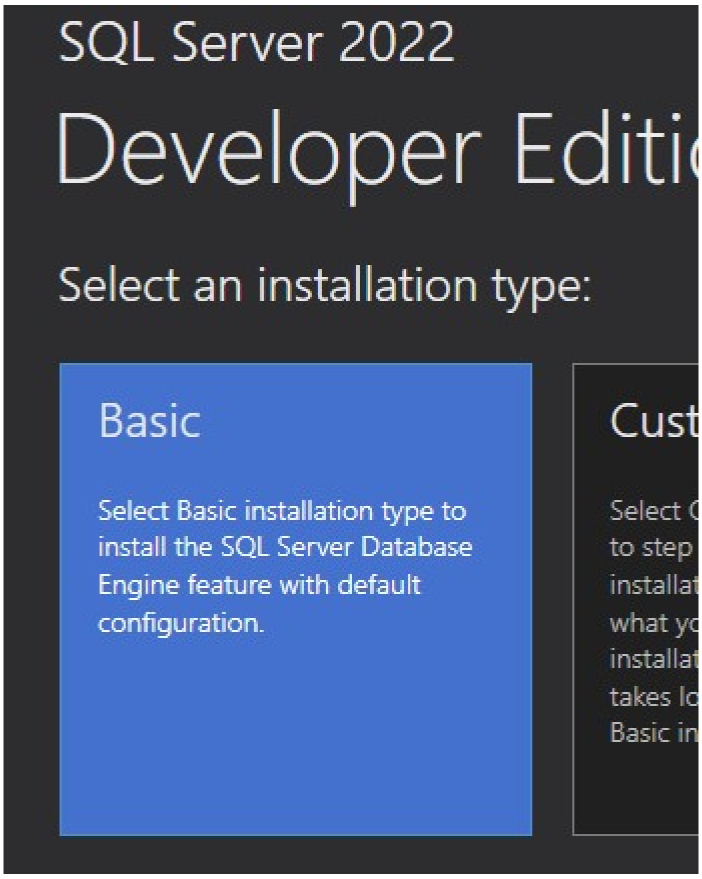
- Navigate to the first link above (SQL Server 2022 Developer) and download the Developer installer



Developer

SQL Server 2022 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

- Perform a basic installation to ensure SQL Server is installed


SQL Server 2022

Developer Editi

Select an installation type:

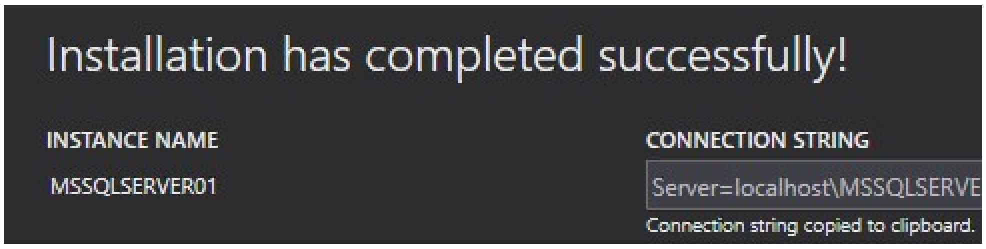
Basic

Select Basic installation type to install the SQL Server Database Engine feature with default configuration.

Cust

Select C
to step
installat
what yo
installat
takes lo
Basic in

- At the end of the installation please ensure that the connection string has localhost as the server and copy the connection string to clipboard


Installation has completed successfully!

INSTANCE NAME

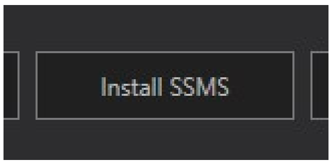
MSSQLSERVER01

CONNECTION STRING

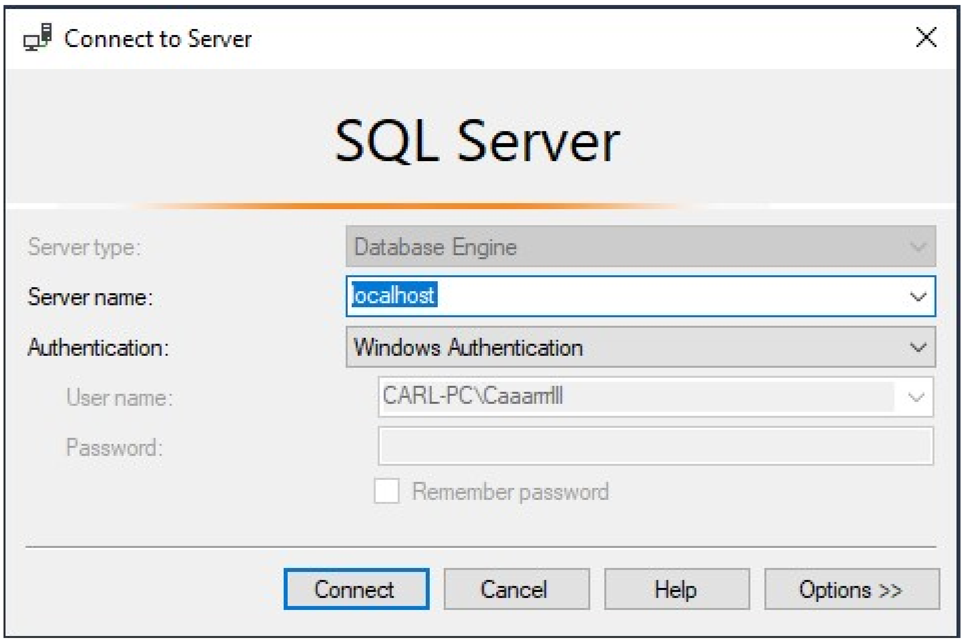
Server=localhost\MSSQLSERVE

Connection string copied to clipboard.

- After SQL Server has been installed, any SQL Management Studio is required, the installer has a prompt to install Sql Server Management Studio (install with all the defaults)

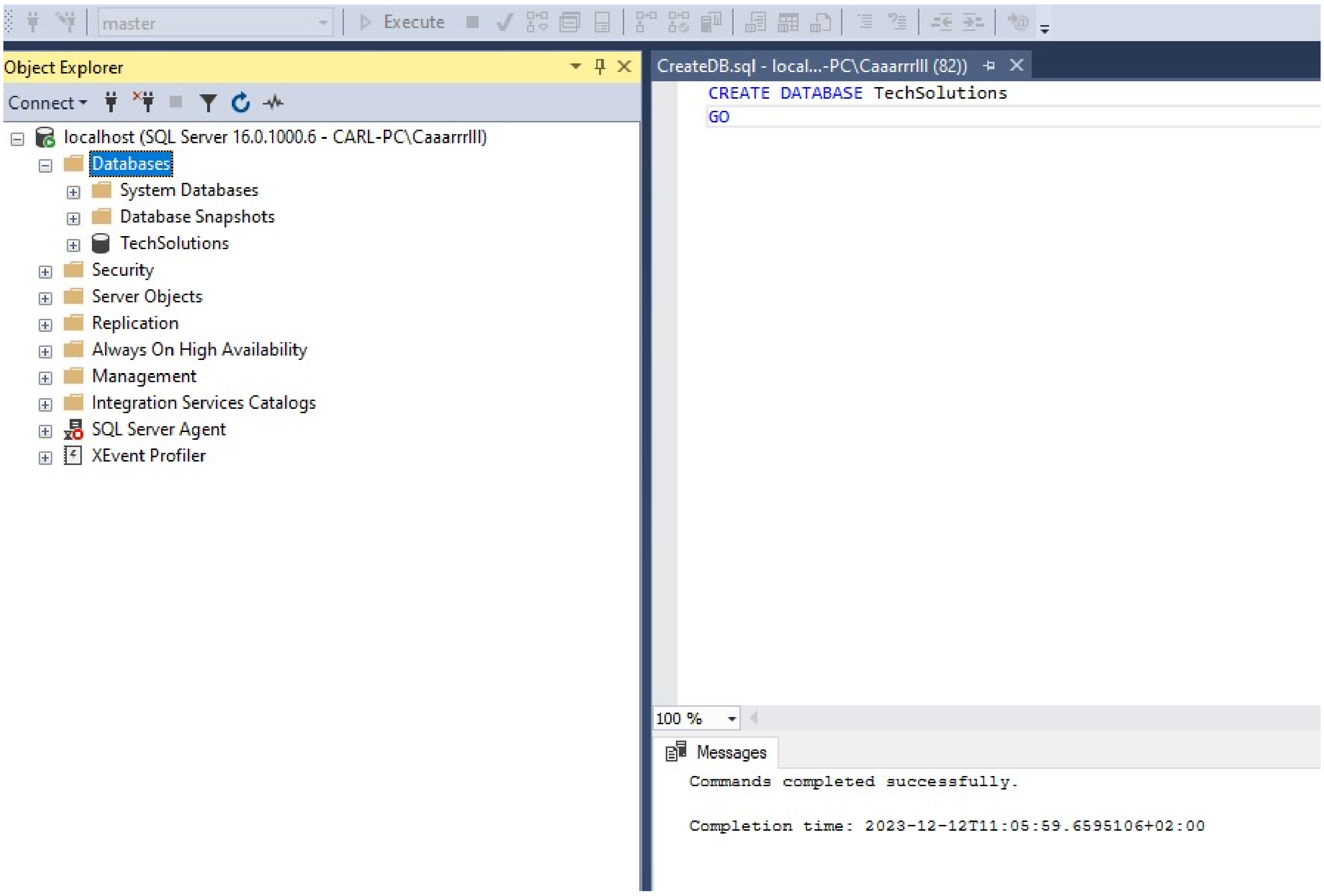


- If SSMS was installed simply open it and change the server name to localhost and connect



Server type: Database Engine
Server name: localhost
Authentication: Windows Authentication
User name: CARL-PC\Caaarrlll
Password:
Remember password

- After connecting to the local database simply drag the CreateDB.sql script (in the scripts folder) into the editor box hit execute and refresh the connection to see the database (TechSolutions) should be under the list now.

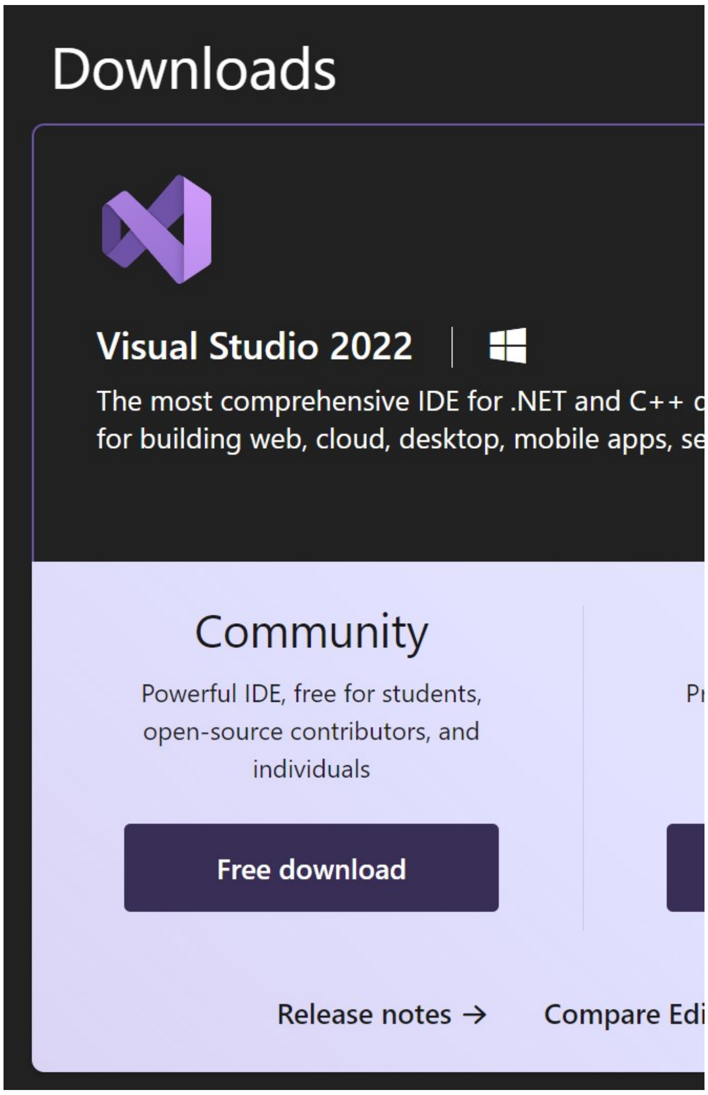


```
CREATE DATABASE TechSolutions
GO
```

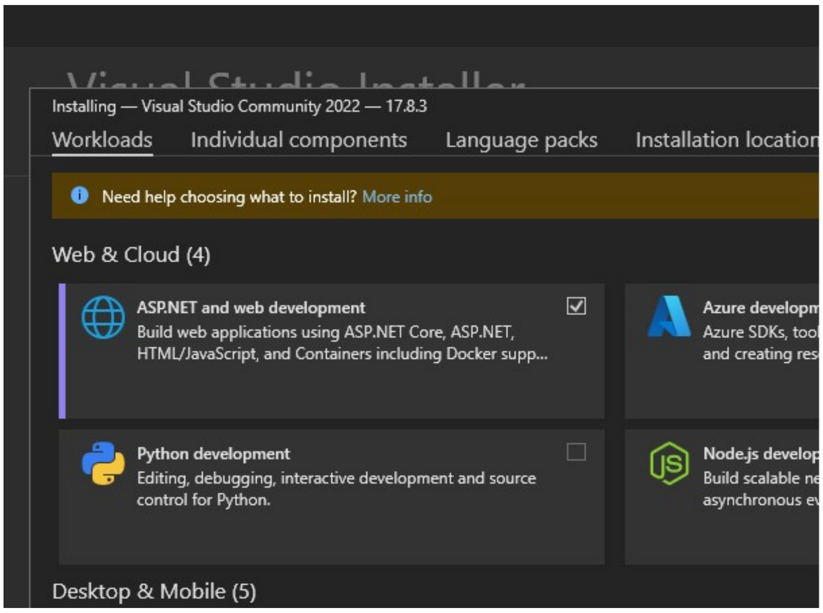
```
Commands completed successfully.

Completion time: 2023-12-12T11:05:59.6595106+02:00
```

IDE

- Download the Visual Studio Community Installer from the second link above and run it


Downloads
Visual Studio 2022
The most comprehensive IDE for .NET and C++ d
for building web, cloud, desktop, mobile apps, se
Community
Powerful IDE, free for students, open-source contributors, and individuals
Free download
Pr
Release notes →
Compare Edi

- Under Workloads select ASP.NET and web development and click install



Installing — Visual Studio Community 2022 — 17.8.3

ASP.NET and web development [✓]
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...

Python development
Editing, debugging, interactive development and source control for Python.

- After installation double click the TechSolutionsCRM solution file to open the project. Navigate to (in the top menu bar) Tools > Command Line > Developer Powershell and install the Entity Framework Core tools by running the following command:

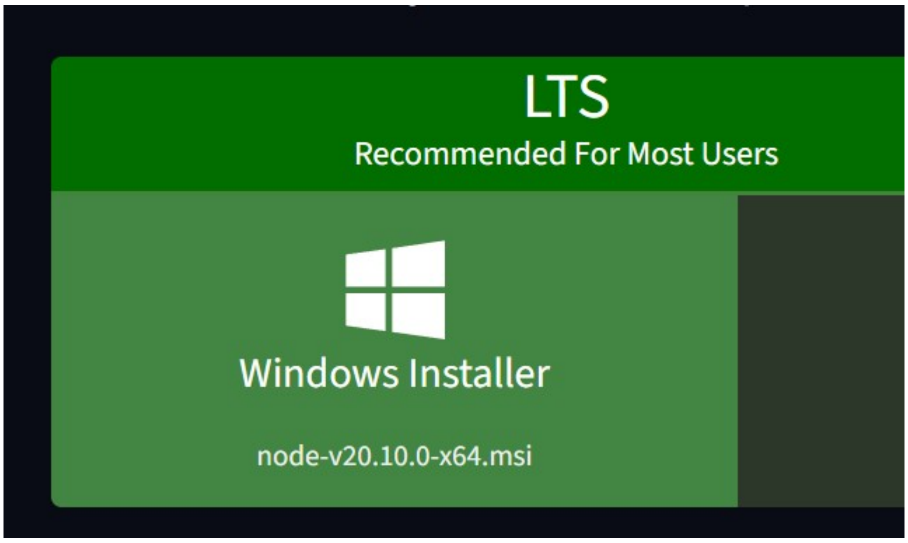
```
dotnet tool install --global dotnet-ef
```

This will give a message if already installed or if this fails please ensure that the .NET 8 SDK is installed on the machine and try again.

- Once Entity Framework tools are installed, in the same console run the following command to create all tables and to seed some data
`dotnet ef database update --context IdentityContext; dotnet ef database update --context TechSolutionsCRMContext`
These 2 concatenated commands fire the entity core updates per context. With these migrations 2 Customers are inserted with 1 of them having an address.
- Keep Visual Studio open till later.

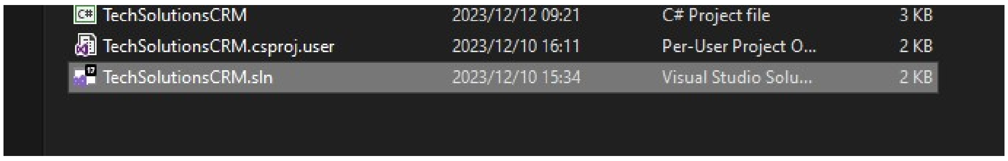
Frontend

- Navigate to the NodeJS download link above and download the installer and install using default settings.



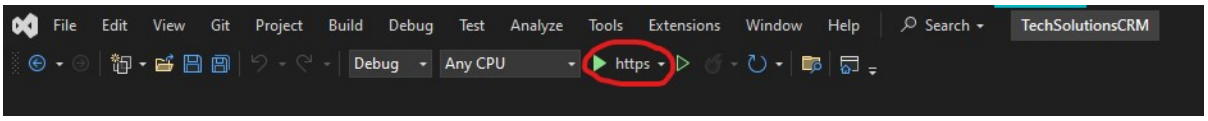
Running the Project

- Open the TechSolutionsCRM solution in Visual Studio



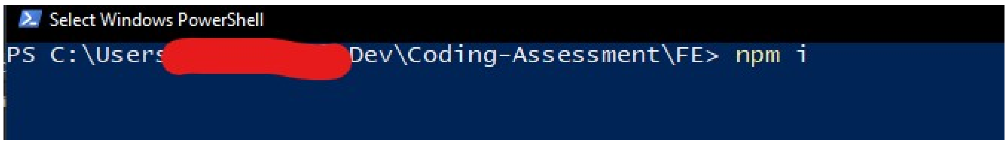
TechSolutionsCRM	2023/12/12 09:21	C# Project file	3 KB
TechSolutionsCRM.csproj.user	2023/12/10 16:11	Per-User Project O...	2 KB
TechSolutionsCRM.sln	2023/12/10 15:34	Visual Studio Solu...	2 KB

and run the project (F5)



, this will start the project and should open a swagger page with the customers controller as well as the Identity Controller (TechSolutionsCRM).

- Open a terminal in the FE (Frontend) folder and and run
`npm i` or `npm install`



```
PS C:\Users[redacted]Dev\Coding-Assessment\FE> npm i
```

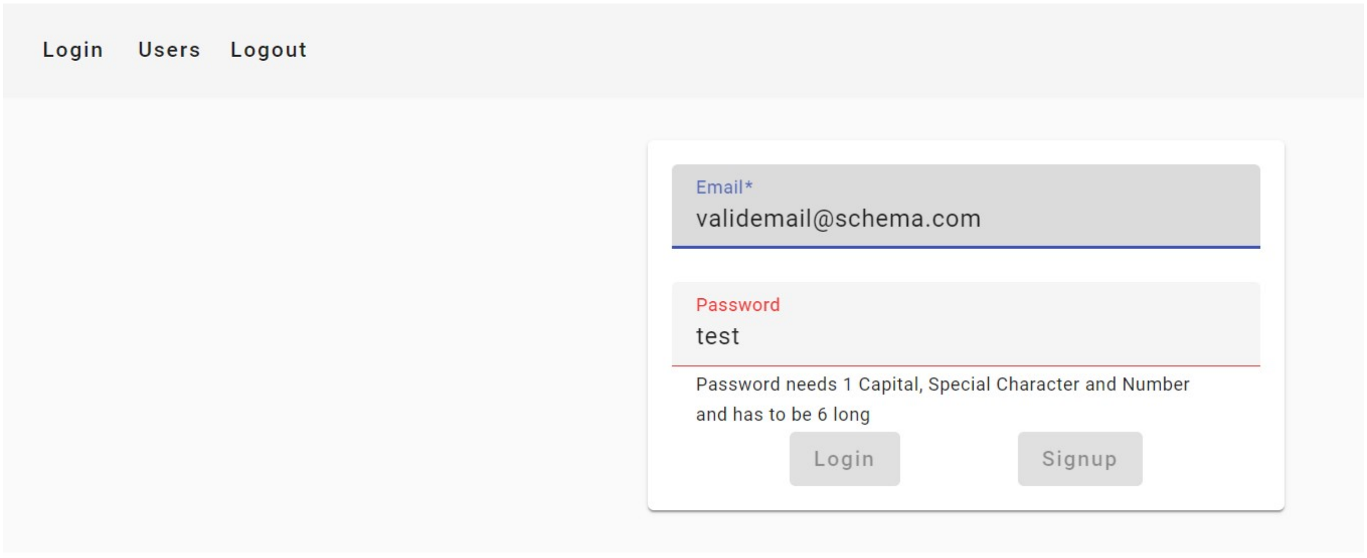
after the installation is finished run the following command to start the frontend `npm run start` and the application should open in a browser window. If it does not you can open the following link to view the application.

this link

•

Navigating the application

The application will open on the login page, please register as an employee (any valid email will do) and the password has some specifications.

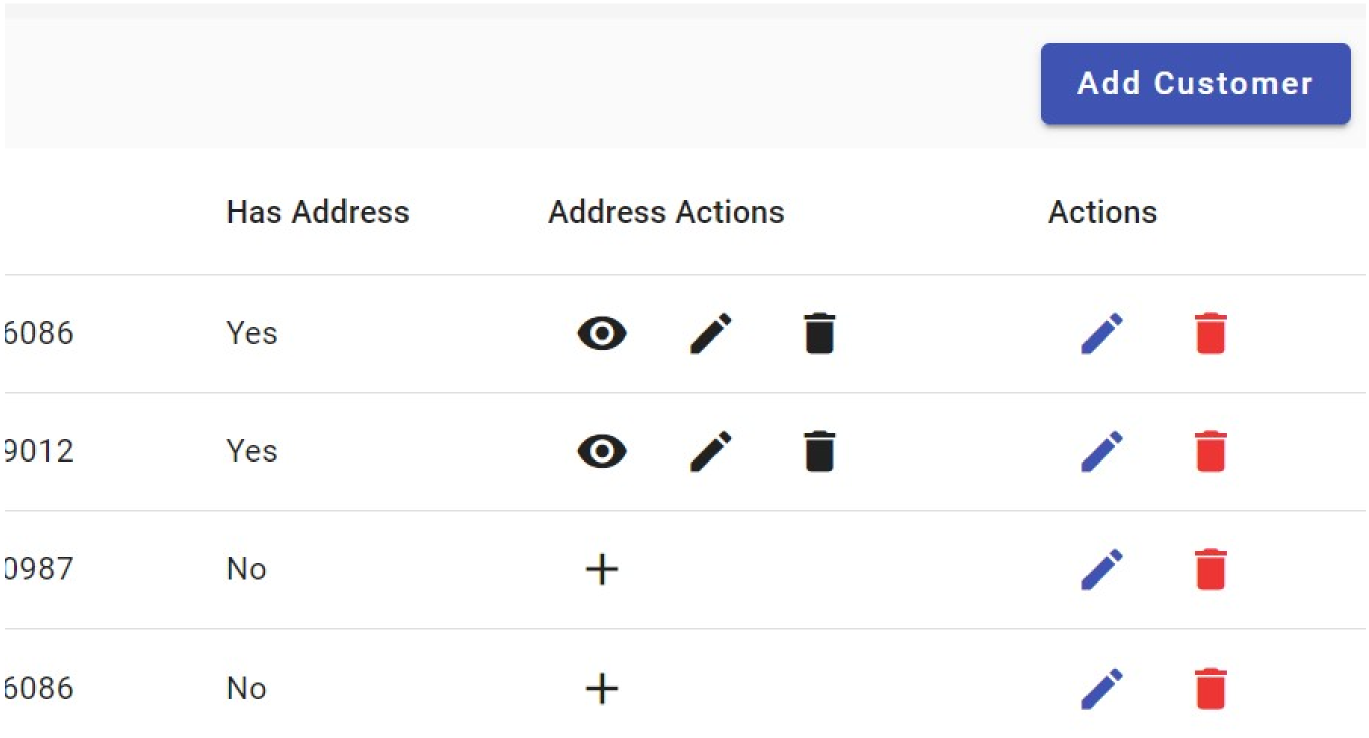


Email*
validemail@schema.com

Password
test

Password needs 1 Capital, Special Character and Number and has to be 6 long

Once logged in you will be redirected to the CRM page where you can view the customers and their addresses, you can also edit the customers and edit/add their addresses.



	Has Address	Address Actions	Actions
6086	Yes	👁 ✎ 🗑	✎ 🗑
9012	Yes	👁 ✎ 🗑	✎ 🗑
0987	No	+	✎ 🗑
6086	No	+	✎ 🗑

There are 4 actions for addresses, namely:

- Add Address (Eye Icon)

- Edit Address (Pencil Icon)
- Delete Address (Trash Can Icon)
- View Address (Plus Icon)

And they perform the tasks as described above.

Right of them there are customer actions, namely:

- Edit Customer (Pencil Icon)
- Delete Customer (Trash Can Icon)

And finally in the top menu bar there is a logout button which will log you out of the application and redirect you to the login page. This also clears the tokens out of session storage so you will have to log back in again.

Frameworks

- Microsoft SQL
- .NET Core, Entity Framework & Identity
- Angular & Material

Database

The database is manually created using a script (CreateDB.sql) located in the Scripts folder in the root of the project. The contents of the database are managed by the API and not manually maintained using scripts.

The database consists of 2 tables for customers, namely:

- [CRM].[Customer]
- [CRM].[Addresses]

and a set of tables required by IdentityServer for authorization.

The CRM tables are are linked in a one (customer) to many (addresses) relationship, where one customer can have many addresses. However the way the code is managed a user will only ever have one address. The Creation and maintenance of these tables is handled by Entity Framework and the migrations are created and stored in the on the API to allow for easy development and rollbacks in case of system failures.

This decision also allows for easy recovery in case of critical system failure.

API and Architecture

Located in the sub folder API is a .NET Core 8.0 WebAPI using Entity Core Framework & Identity Core Framework The API is designed with clean architecture and KISS principle in mind and is split into 3 main segments:

- Controllers :
  The "Experience Layer" which allows the api to be easily used by external applications
  - The Customer Controller which allows for Data viewing and editing on customer data (including addresses) by the company employee
  - The Identity Controller which allows for employee registration and login

- Services :
  The "Business Layer" which contains the business logic and is used to interact with the context (database)
  - The Customer Service which implements the functionality of the Customer Interface
  - The Address Service which implements the functionality of the Address Interface
- Context:
  The "Data Layer" which contains the database context and migrations
  - TechSolutionsContext (Customer Context) which is the main connection responsible for handling the [CRM] Schema in the database
  - IdentityContext which is the connection responsible for handling the Identity tables in the database

This choice was made because it allowed me to build the API in a way that is is easy to work with and maintain, as well as it allows to be easily extended in the future. The possible extensions that could be made to this Architecture would be adding a repository tto allow abstraction on the Context layer to simplify code. From past experience I decided to not add these right away because it could increase complexity and reduce simplicity if not required.

Frontend

The Frontend is built on Angular's Domain Module Architecture, this approach bundles core domain ideas in one module that can be lazy loaded and shared across the application. This allows for easy maintenance and extension of the application. This principle also allows the project to scale incredibly well while also keeping coupling to a minimum. The basic rules for Domain Module Architecture can be stated as keep an idea in a module, in my applications case, the login and registration code falls in its own module and then the CRM code falls in another. Base Services should be created for network calls and should not modify the data, this allows for any module to use the same service and not have to worry about, and a module gets a domain specific service when data has to be modelled or modified for that domain.