Table of Contents

[Project Overview: 2](#_Toc59489203)

[Backend specification: 2](#_Toc59489204)

[Environment Description: 2](#_Toc59489205)

[Installed Packages: 2](#_Toc59489206)

[Script to Execute: 2](#_Toc59489207)

[Database design: 2](#_Toc59489208)

[Application folder structure: 3](#_Toc59489209)

[API Design: 3](#_Toc59489210)

[Available Methods: 5](#_Toc59489211)

[Frontend specification: 7](#_Toc59489212)

[Environment Description: 7](#_Toc59489213)

[Running the code: 7](#_Toc59489214)

[Application Folder Structure: 7](#_Toc59489215)

# Project Overview:

This project was to create a survey form for Compass Education. I have created 2 separate project to implement this functionality. Front end uses React with Context API, and backend uses Entity Framework core 3.1.

# Backend specification:

## Environment Description:

IDE used: Visual studio 2019

Framework: .ASP.net Core

Framework Version: 3.1

## Installed Packages:

Microsoft.EntityFrameworkCore – To use the .net core functionalities

Microsoft.EntityFrameworkCore.SqlServer – To use Microsoft SQL server functionalities

Microsoft.EntityFrameworkCore.Tools - To execute and run migration script such as

Microsoft.AspNetCore.Mvc.NewtonsoftJson - To utilize the newtonsoft features to parse the data  
Swashbuckle.AspNetCore.SwaggerGen - To generate the swagger open API documentation

## Script to Execute:

This part is optional

1. Add-migration “initial migration”
2. Update-database

## Database design:

3 different entities created with 1-M relationship between Survey => Question and Question => Option.

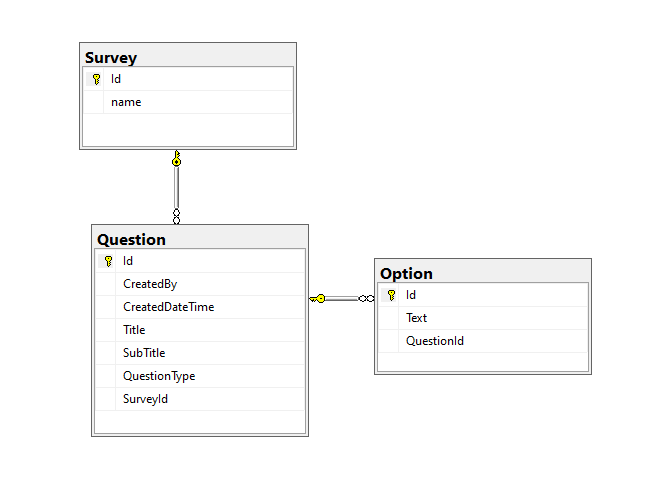


Figure : Database diagram, Compass Survey System

## Application folder structure:

The overall folder structure of the application is shown below. Controller contains the methods to be invoked from external parties (Front end application). Due to the limited work scope, authentication mechanism is not implemented. So, the people having access to the api will be able to GET/POST data. Only survey controller is exposed to external application.

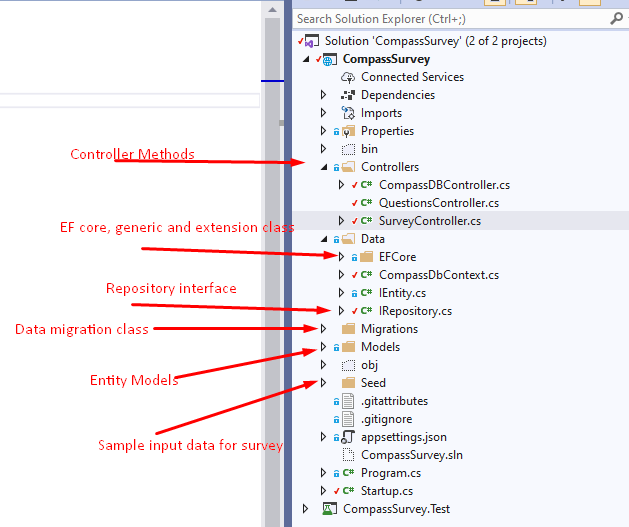


Figure : Application folder structure

Data folder contains the classes such as Entity context, Repository Interface, generic classes to perform CRUD operation and implementation in Entity Framework Core. Migration file is created and system will create database during the runtime. So, the system expects the connection to local database with Trusted\_Connection = True.

Startup.cs class contains the important configuration methods such as registering the repository, allowing the cross platform application access, controllers, and Swagger configuration.

Swagger is an open API framework used to create the api documentation so that the user can walk through the application without using any third party tools such as Postman, However the use of postman is recommended rather than directly testing in swagger UI due to the better visibility of application in Postman.

Startup class also initialize the configuration service to migrate the database during runtime. So if the database doesn’t exist, system will create it.

## API Design:

The structure follows the base URL followed by the corresponding endpoints. For the specific project only one endpoint i.e. GET: /api/v1/Survey is sufficient, but I have implemented the post and delete methods as well so that we can add some dummy data to test it. There won’t be any user interface for adding the records, so it must go through the Swagger UI or Postman.

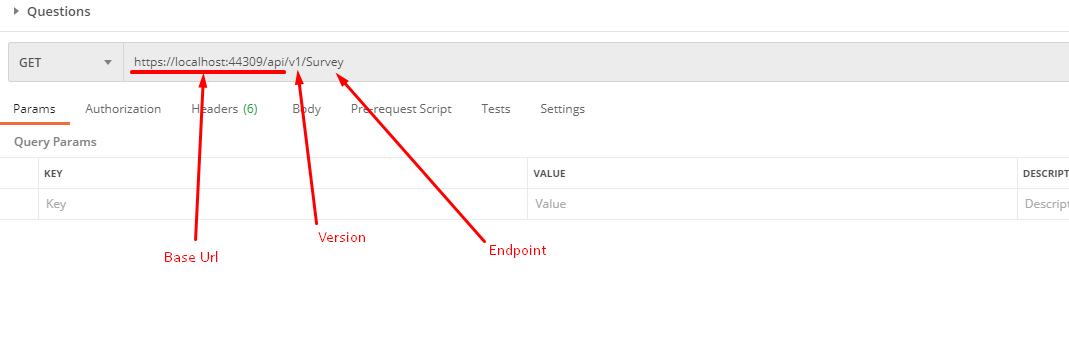
The breakdown of URL information is shown below: 

Figure : API url

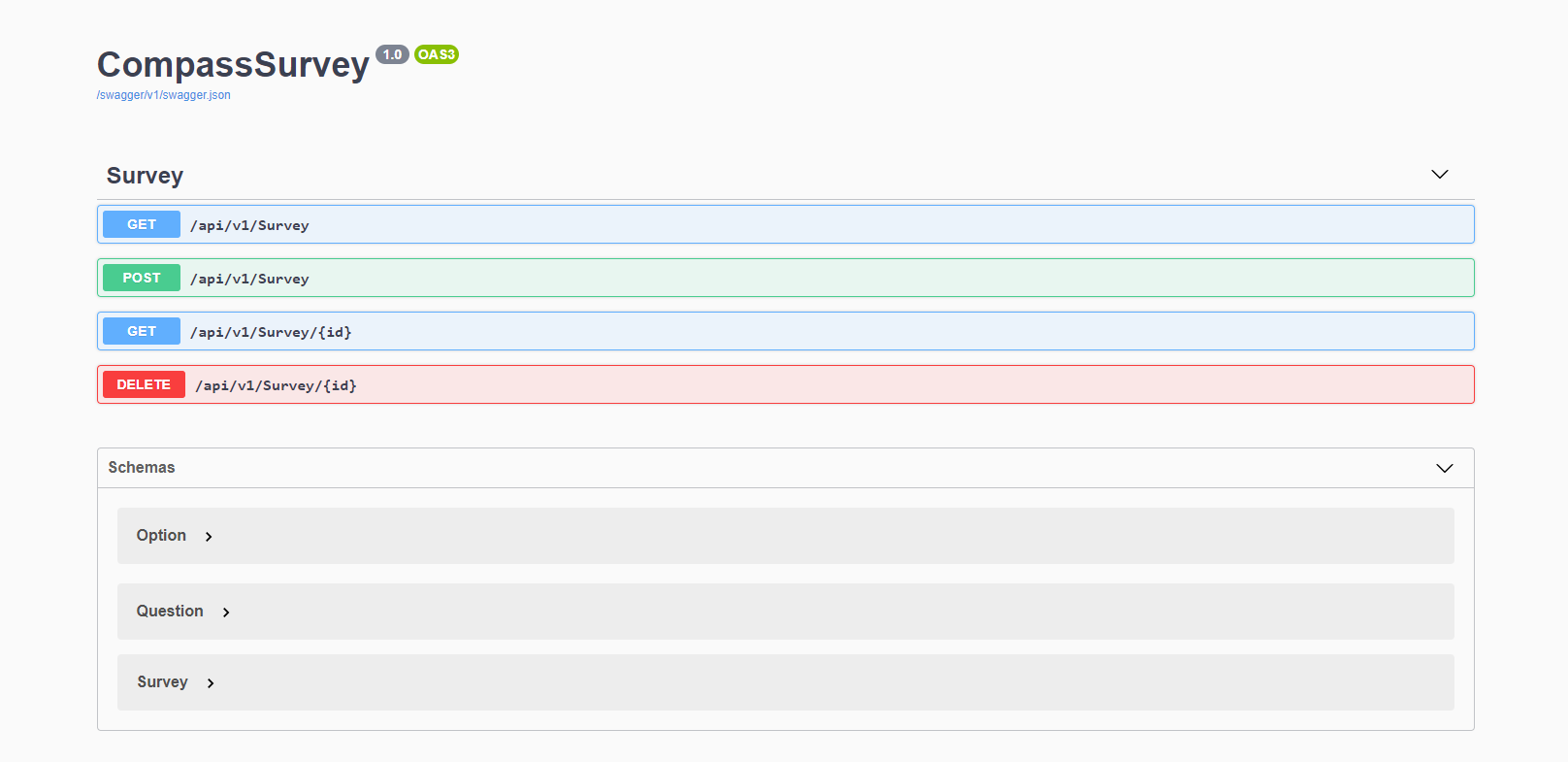
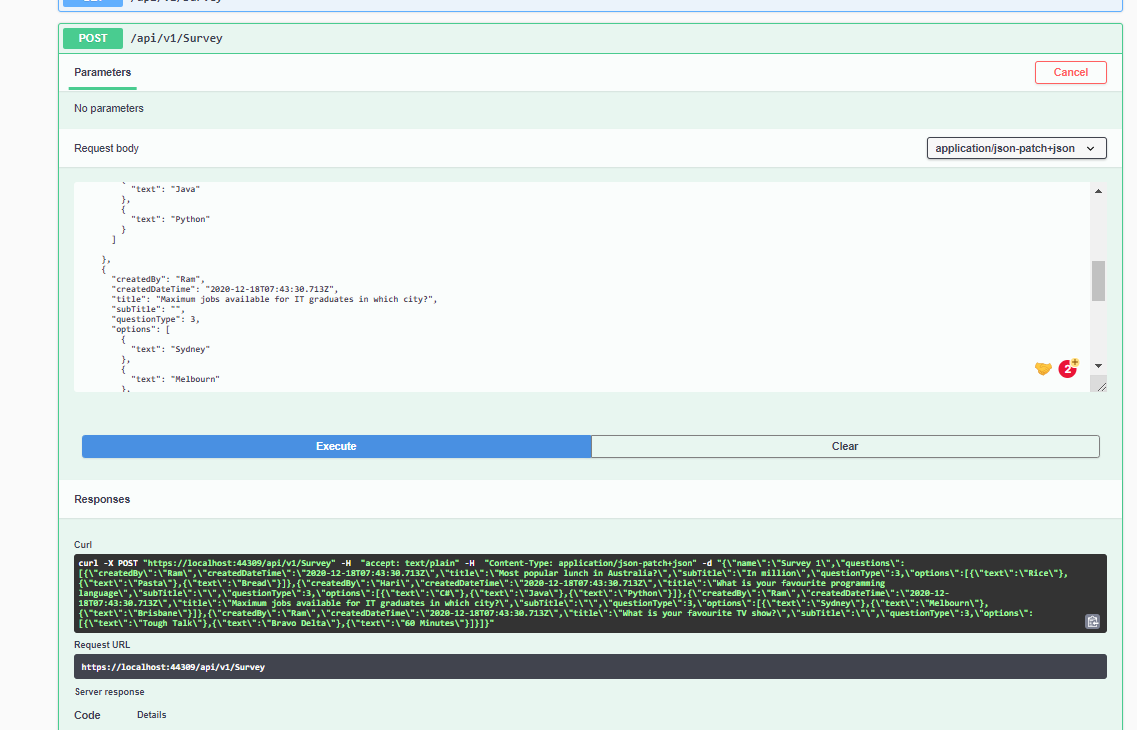
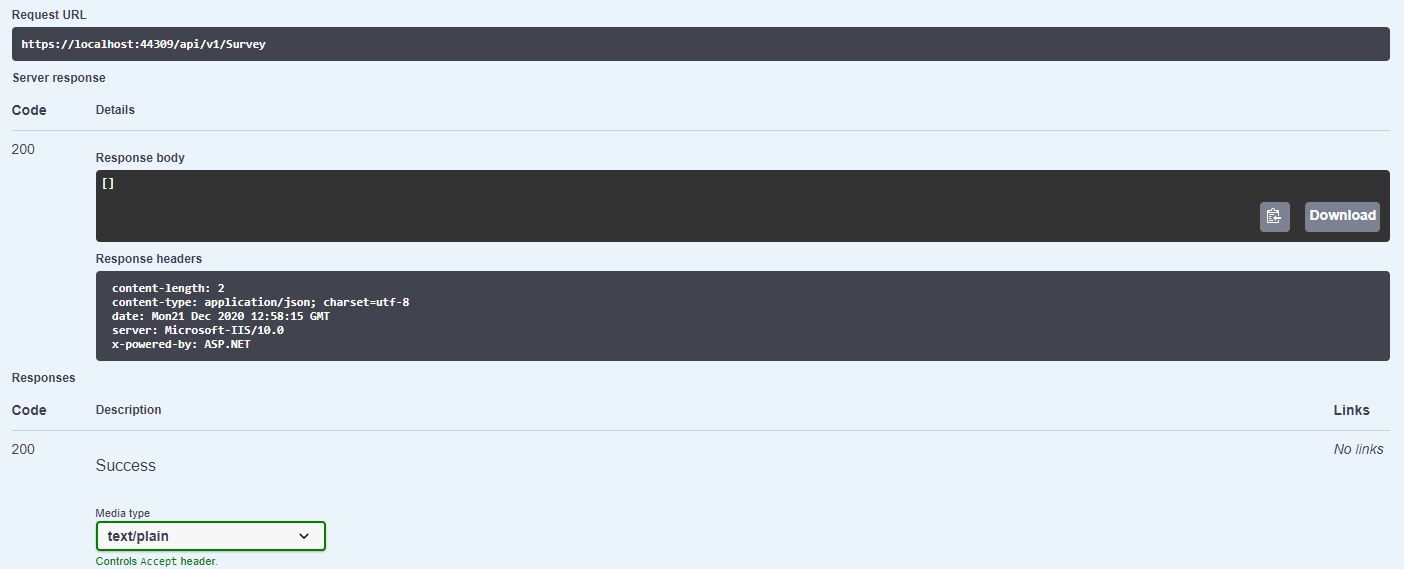
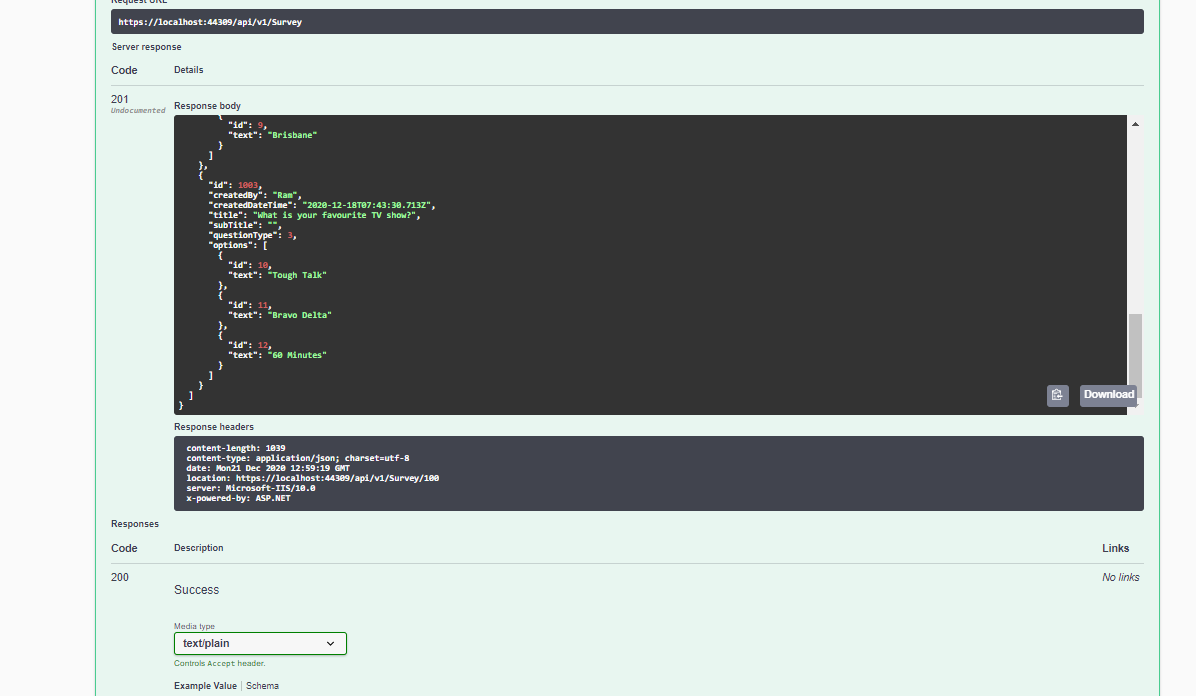


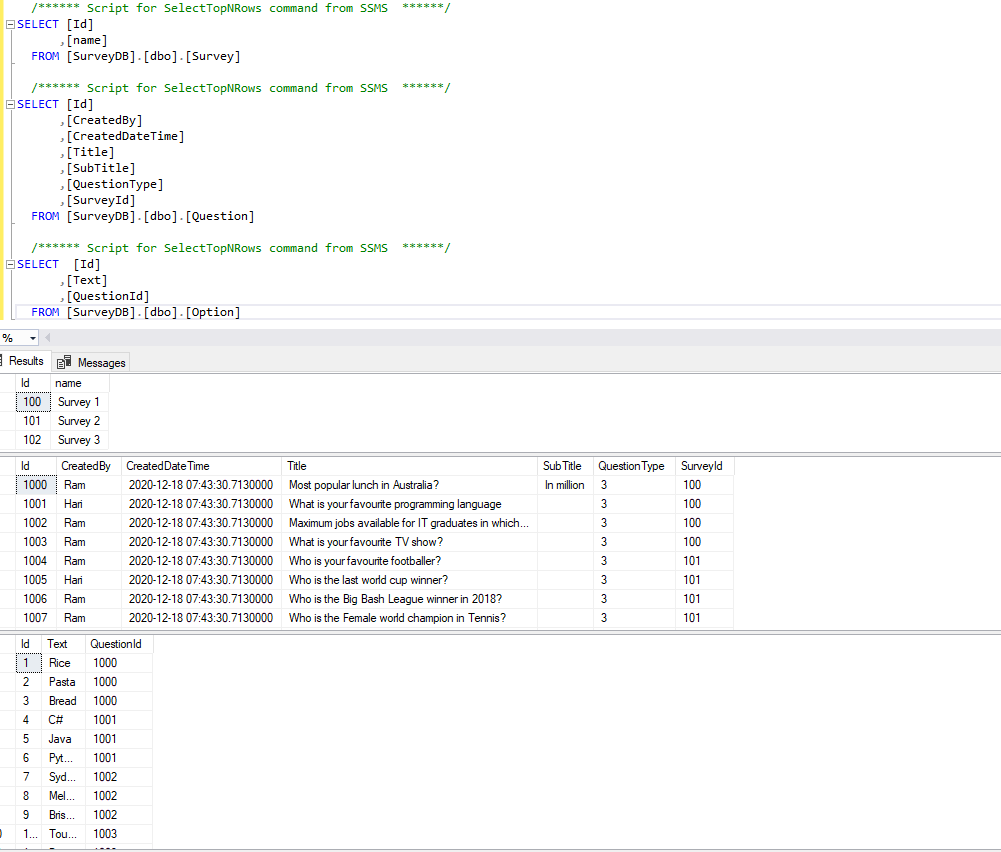
Figure : Swagger UI snapshot

## Available Methods:

1. Get all survey data – This is only used for this application
2. Get Specific survey with Id
3. Post Survey data – including the nested questions (multiple) and options (multiple)
4. Delete the survey data – cascading delete is applied and it will delete the questions and options as well.

Few snapshots showing the method invocation in swagger: 





# Frontend specification:

The front end is developed using React and Context API to maintain the state of the data.

## Environment Description:

IDE Used: Visual Studio Code

Framework: React

Installed Packages:

 "bootstrap": "^4.5.3",

    "react": "^17.0.1",

    "react-bootstrap": "^1.4.0",

    "react-dom": "^17.0.1",

    "react-router": "^5.2.0",

    "react-router-dom": "^5.2.0",

    "react-scripts": "4.0.1",

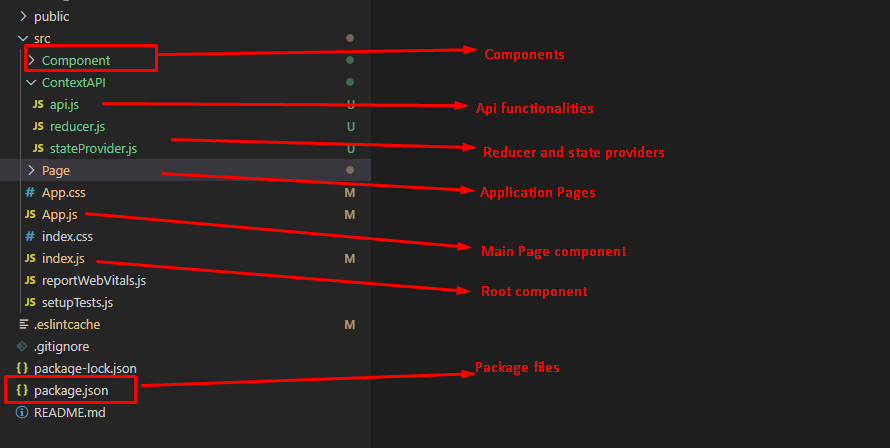
    "react-thunk": "^1.0.0",

## Running the code:

Execute **npm start** into your terminal in the project directory.

## Application Folder Structure:

The application contains the components, pages components, API component, and reducer and context API functionalities. React framework works on lego block structure, so the small component is responsible for bigger component (I have names as page just for common understanding).



Lego?

Following structure shows how the react components assembles to form and very nice single page application.

