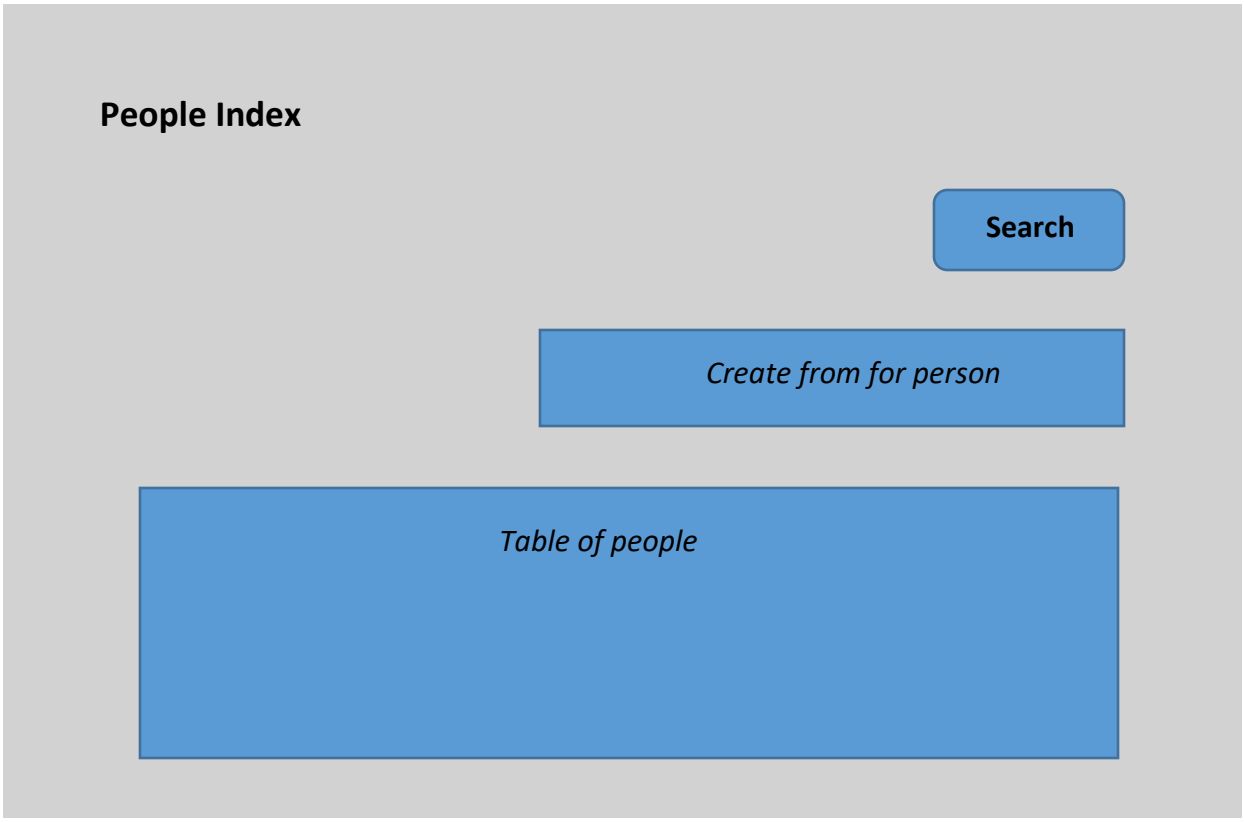
Create a MVC project with a controller with one view to displays a list of people.
These people should have a name, phone number and city.
The controller will let a class that implements IPeopleService take care of the business logic and use a ViewModel to send out the needed data to the view.
The PeopleService class will let a class that implements IPeopleRepo store the people's data.

Required Features:

- A single view that has the following:
  - Html table of people.
    - Each row should show a person, and a link that when clicked, removes that person.
  - Two forms:
    - A form that filters the table content – if you submit the form, the page should be refreshed and only show the people whose names or cities with name containing the string you entered in the form.
    - The other form should let you add a person to the list of people.



Code Requirements:

- Models
  - Person – Person data.
  - CreatePersonViewModel – Use to prevent overposting and to use data annotations to validate inputs when creating new person.
  - PeopleViewModel – container for the information you need in your people view.
  - IPeopleRepo – Interface with following methods.
    - Person Create("parameters needed to create Person excluding id")
    - List<Person> Read()
    - Person Read(int id)
    - Person Update(Person person)
    - bool Delete(Person person)
  - InMemoryPeopleRepo – Implements IPeopleRepo interface and these two fields.
    - Private static List of Person.
    - Private static int idCounter.
  - IPeopleService – Interface with following methods.
    - Person Add(CreatePersonViewModel person)
    - PeopleViewModel All()
    - PeopleViewModel FindBy(PeopleViewModel search)
    - Person FindBy(int id)
    - Person Edit(int id, Person person)
    - bool Remove(int id)
  - PeopleService – Implements IPeopleService interface.
- Your PeopleController shall use the PeopleService class.
- The table data should come from a view model, which should have a list of people, and the search phrase if one exists.

Optional:

- Add buttons to sort the list on the page.
  - Sort in alphabetical order and reverse alphabetical order, by name or by city.
- Add a checkbox which determines whether the filtering should be case sensitive or not.

Resources:

- https://dotnettutorials.net/lesson/view-model-asp-net-core-mvc/

Subjects Covered:

- Models
  - As data source (Repository)
  - As business logic (Service)
  - View Models
    - @Model vs @model
    - @using
    - Data annotations
- Forms
  - GET vs POST