**Instructions to Run Application**

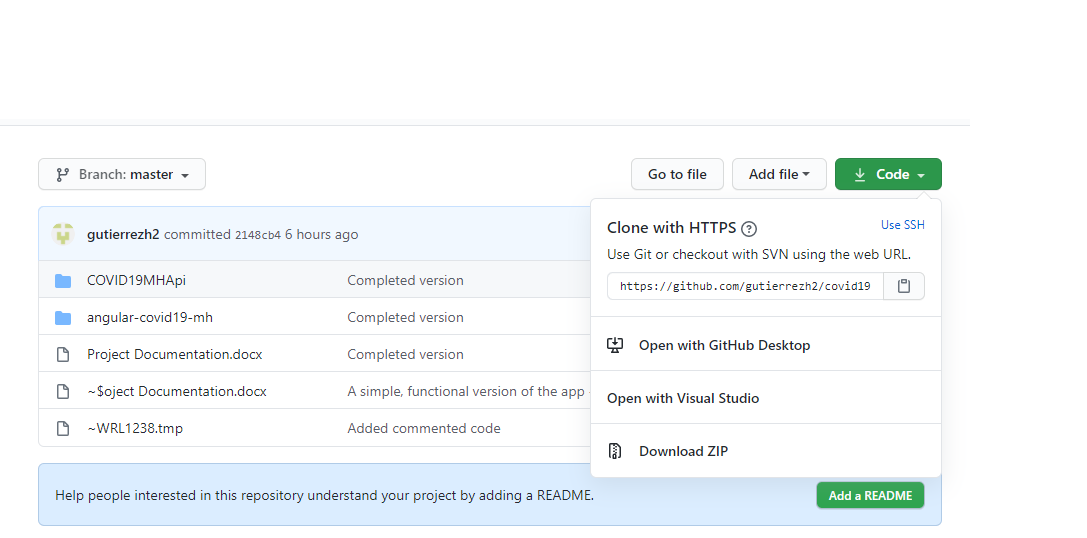
* **Note:** This project was coded using a Windows machine.
* There are two parts to running this application, running the **Web API**, and running the **Angular Application**.

**Recommended Installations**

* **Web Api**:
  + Obtained requirements from the Microsoft Web API Tutorial: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&viewFallbackFrom=aspnetcore-&tabs=visual-studio-code>
  + **Visual Studio Code:** <https://code.visualstudio.com/download>
  + **C# for Visual Studio Code:** <https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp>
  + **.NET Core 3.1 SDK or later** (I installed **SDK 3.1.301**): <https://dotnet.microsoft.com/download/dotnet-core/3.1>
* **Angular Application:**
  + Obtained requirements from the Angular Tutorial: <https://angular.io/guide/setup-local>
  + **Node.js:** <https://nodejs.org/en/>
  + **Angular CLI:** Run the following command in Command Prompt or Terminal: **npm install -g @angular/cli**

**Part 1.** **Obtaining the Source Code**

1. Given the link provided in the email, click on the link, which leads to the GitHub page containing the source code.
2. Click on the green **Code** button and:
   1. **Clone** the repository.
   2. Alternatively, if you don’t have a local repository set up, click on the **“Download ZIP”** button to download the folder. Once downloaded, extract the folder to whichever folder is convenient for you.



**Figure 1.** The link to the GitHub page should lead to a page that looks like this. (2) Click on the Code button to make the dropdown menu above appear. (2.b) The link to the “Download ZIP” option.

**(2)**

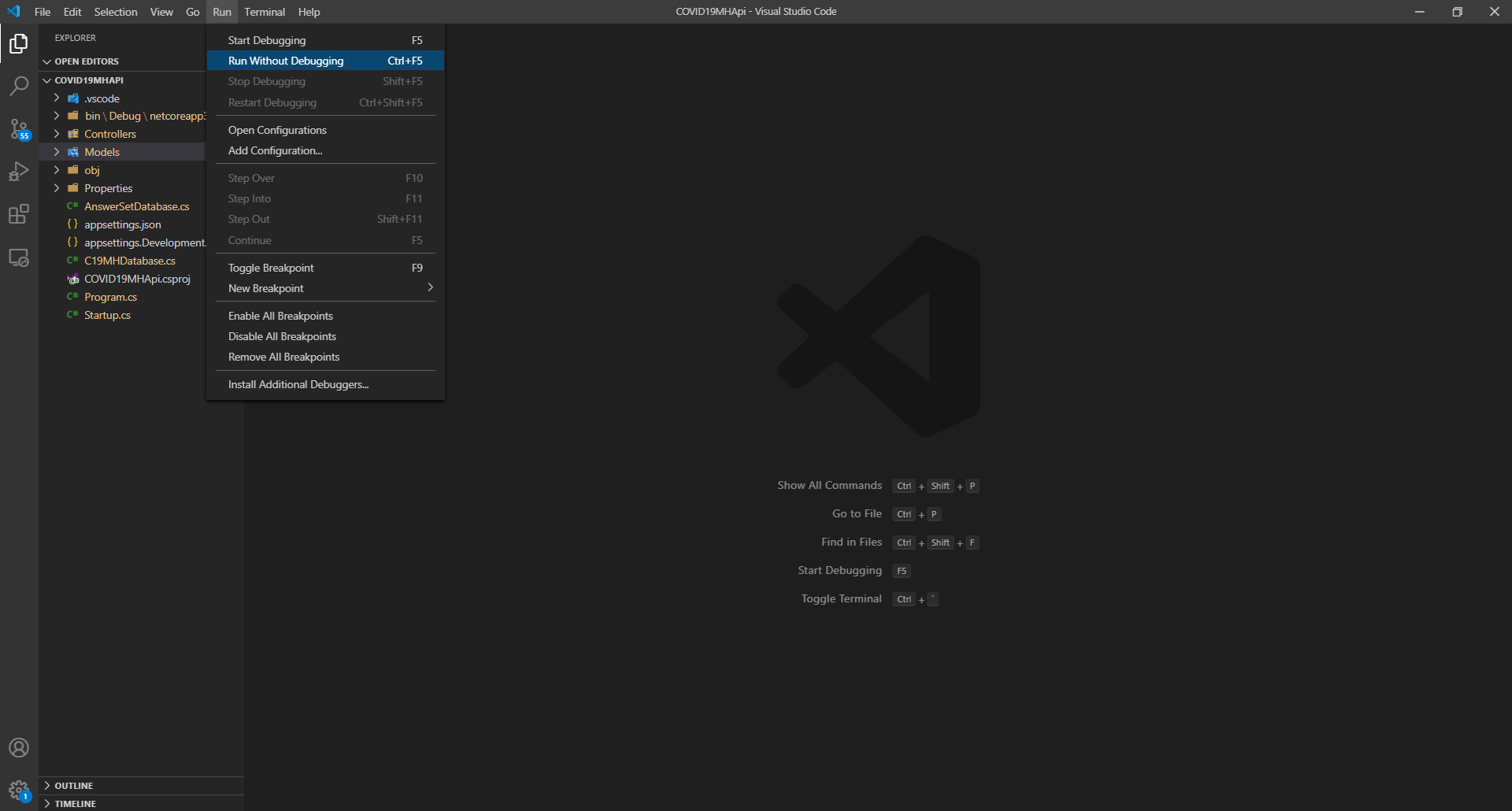
(**2.b)**

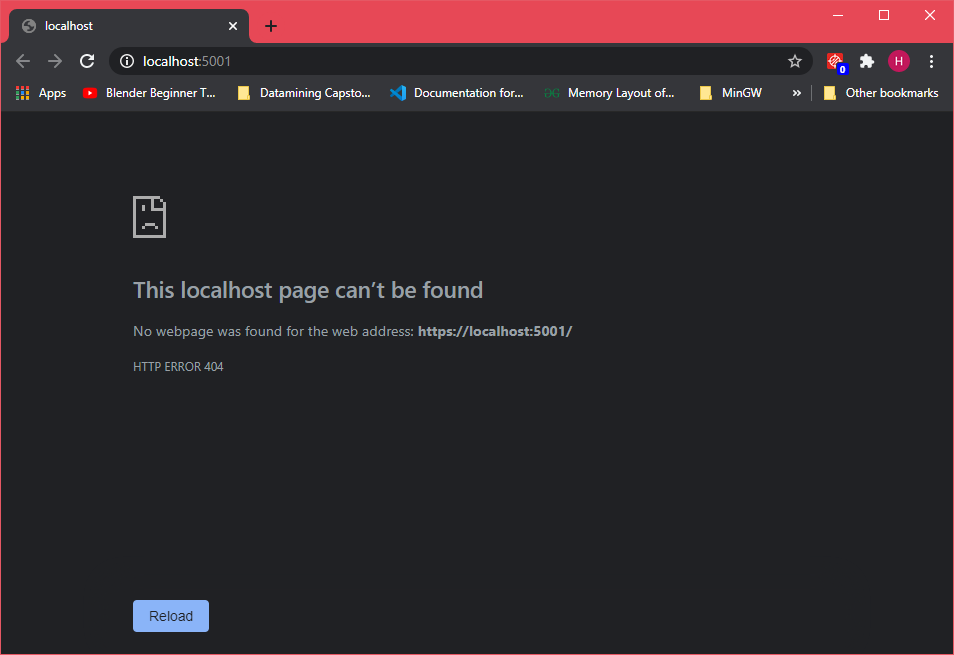
1. Navigate to the **“covid19-mh-app”** folder inside of File Explorer. Inside, it contains the Web API source code, in the **“COVID19MHApi”** folder.
2. Open the **“COVID19MHApi”** folder using your preferred IDE or Code Editor (I used **Visual Studio Code**).

**Part 2. Running the Web API**

1. Once the folder is opened inside your Code Editor, run the application. An indication of whether it’s running successfully or not is that it opened up a new instance of your default Internet browser, opening up to an empty page.

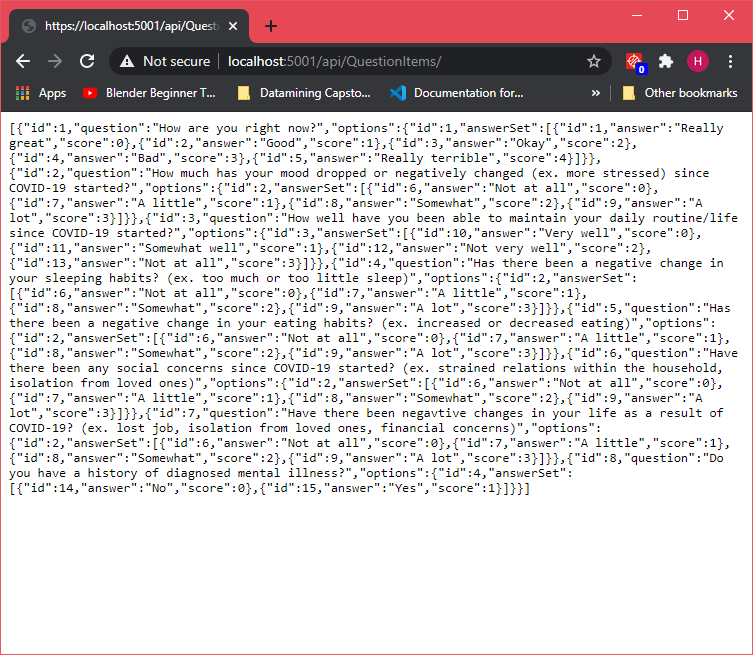
**Figure 3.** On Visual Studio Code, you can click on **“Run Without Debugging”**, as shown below, to run the application.



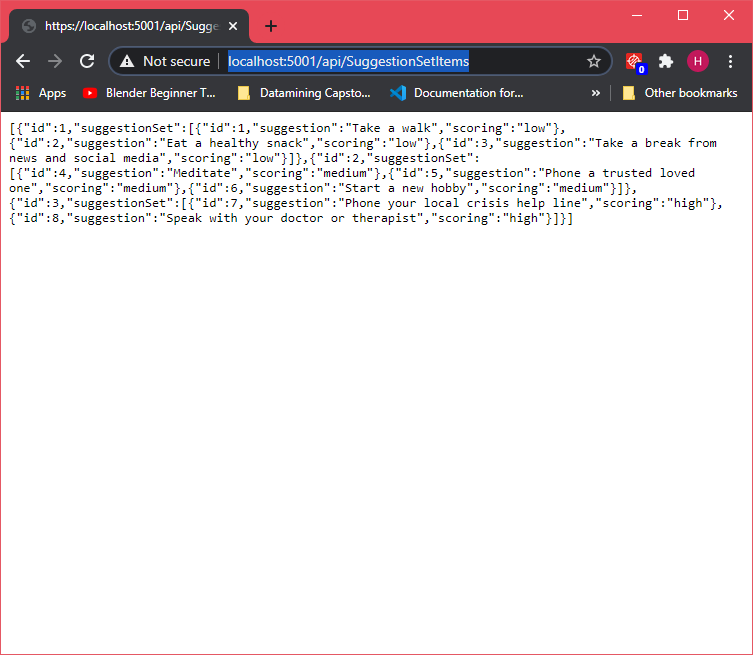


**Figure 4.** This screen appears on your default Internet browser, to indicate that the Web API has started running.

1. The screen below may have appeared instead. To run the web API, click on …
2. To test to see if you can obtain information out of the Web API, type the following links into your Internet browser:
   1. The seeded **QuestionItems** database: <https://localhost:5001/api/QuestionItems/>

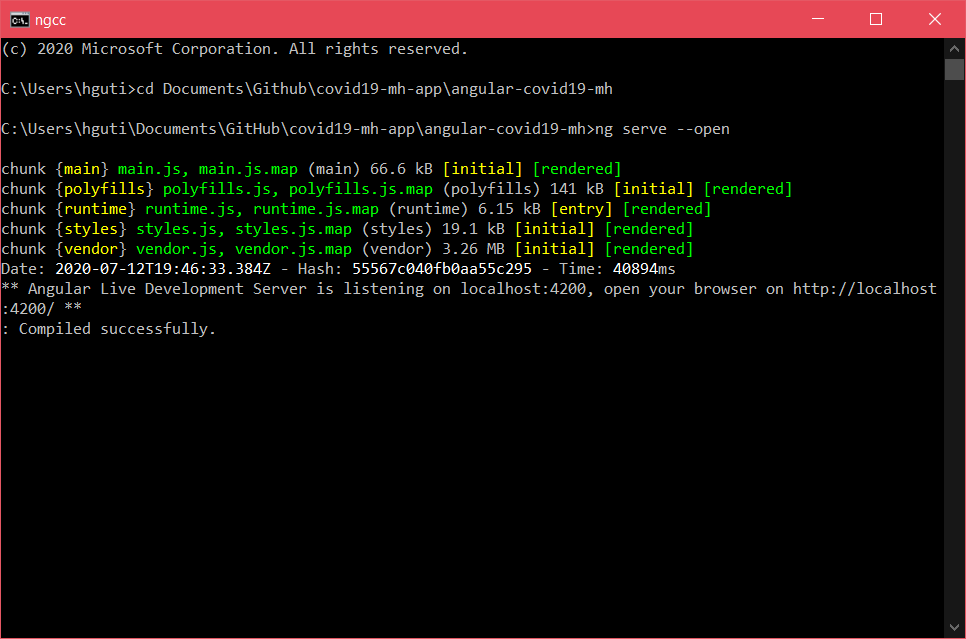


* 1. The seeded **SuggestionSetItems** database: <https://localhost:5001/api/SuggestionSetItems>



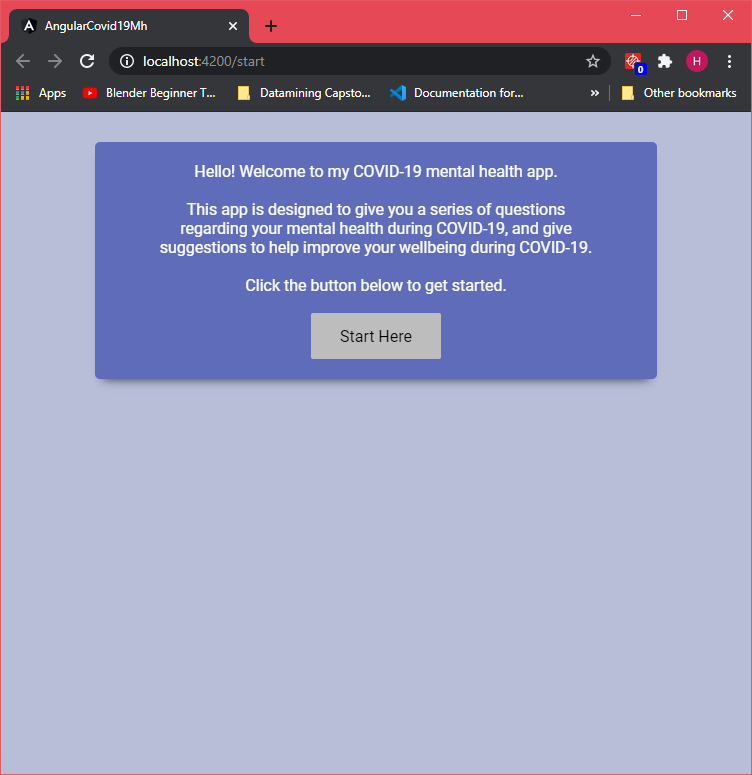
**Part 3. Running the Angular Application**

1. Open a new Command Prompt or Terminal screen.
2. Within Command Prompt orTerminal, navigate to the **“angular-covid19-mh”** folder.
3. Once in this folder, run the command **ng serve --open** to compile and run the Angular Application. This should open up a new instance of your default Internet browser.



**Figure 4.** This is what appears on Command Prompt when you run the **ng serve --open** command.

1. If successful, it should open up to the application, like so:



**Roadblocks**

* **Time Management:** Since I am working a full-time job, many of the ideas that I had were pushed back, in favour of getting a working product out by the due date. This was especially muddled by how I was unfamiliar with the technologies, making it more difficult to judge how long tasks were going to take.
* **Learning New Technologies:**

**Refactoring Opportunities**

* **[Back-end]** Add a **User** entity/object and scaffold a controller for it, to allow proper HTTP client methods (**PUT, POST, GET, DELETE**) for that entity/object.
  + **Issues with Current Code:**
    1. Currently not coded in a way where multiple users can use the application, even within an In-Memory Database
    2. Storing user information (ie. **userAnswers**) in the front-end, when it should be in the back-end
  + **How Solution will Improve Current Code:**
    1. If coded to include a **User** entity/object, then it will be easier to refactor in a future iteration to add functionality for multiple users
    2. Increased separation of roles between the front- and back-end – the front-end will be used for mainly view-related roles, while the back-end will primarily be used for data storage and manipulation
* **[Front-end]** Refactor the form to use Reactive Forms.
  + **Issues with Current Code:**
    1. While the code is able to collect information as the user interacts with the web application, some of it might’ve been done with somewhat “smelly” code, or using unconventional solutions, such as replacing the <p> element for the error message to update what questions weren’t answered yet.
  + **How Solution will Improve Current Code:**
    1. This will make collecting user information cleaner, from a coding perspective.
* **[Testing]** Automate testing using Selenium.
  + **Issues with Current Method:**
    1. Since there isn’t a lot you can do with the web application, manual testing was doable, however, it wouldn’t be scalable had the application been developed on a larger scale.
* **[Front-end]** Style refactoring:
  + Add icons
  + Add sliding animations for the questions
  + Change the start, finish, and other navigation buttons to look more streamlined and modern
* **~~[Back-end & Front-end]~~** ~~Remove the~~ **~~selectedAnswer~~** ~~and~~ **~~isSelected~~** ~~fields in the~~ **~~Question~~** ~~entity/object.~~
  + **~~Issues with Current Code:~~**
    1. ~~Contains extra fields/properties within the~~ **~~Question~~** ~~entity/object that aren’t being used.~~
  + **~~How Solution with Improve Current Code:~~**
    1. ~~Removes extra fields that aren’t being used, keeping the code less cluttered.~~
* **~~[Back-end & Front-end]~~** ~~Refactor so that the database references an~~ **~~AnswerSet~~** ~~DB that has all the answer sets.~~
* **~~[Front-end]~~** ~~Refactor back and next questions into one method; there is no need to have two if similar code is used (and use that one method for the Finish button too?)~~
  + ~~Pass variable to indicate whether Back or Next button called it to know what needs to be done; otherwise, subtract or add 1 from the HTML file and pass that calculation over~~
* **[Front-end]** Style refactoring:
  + ~~Refactor to put common styles in the global styles.css~~
* **~~[Back-end & Front-end]~~** ~~Clear out all unnecessary objects or code, such as base-panel, mock-dbs, etc. Weather ForecastController, etc~~
* **~~[Front-end]~~** ~~Ensure that padding/margin is consistent between the start, question and suggestion panels~~
* **~~[Front-end]~~** ~~Refactor the Questions panel.ts so that the last two methods are in the suggestions.ts~~
* **~~[Front-end]~~** ~~Delete the base panel without errors!~~