# Exercise: Continuous Integration – Part I

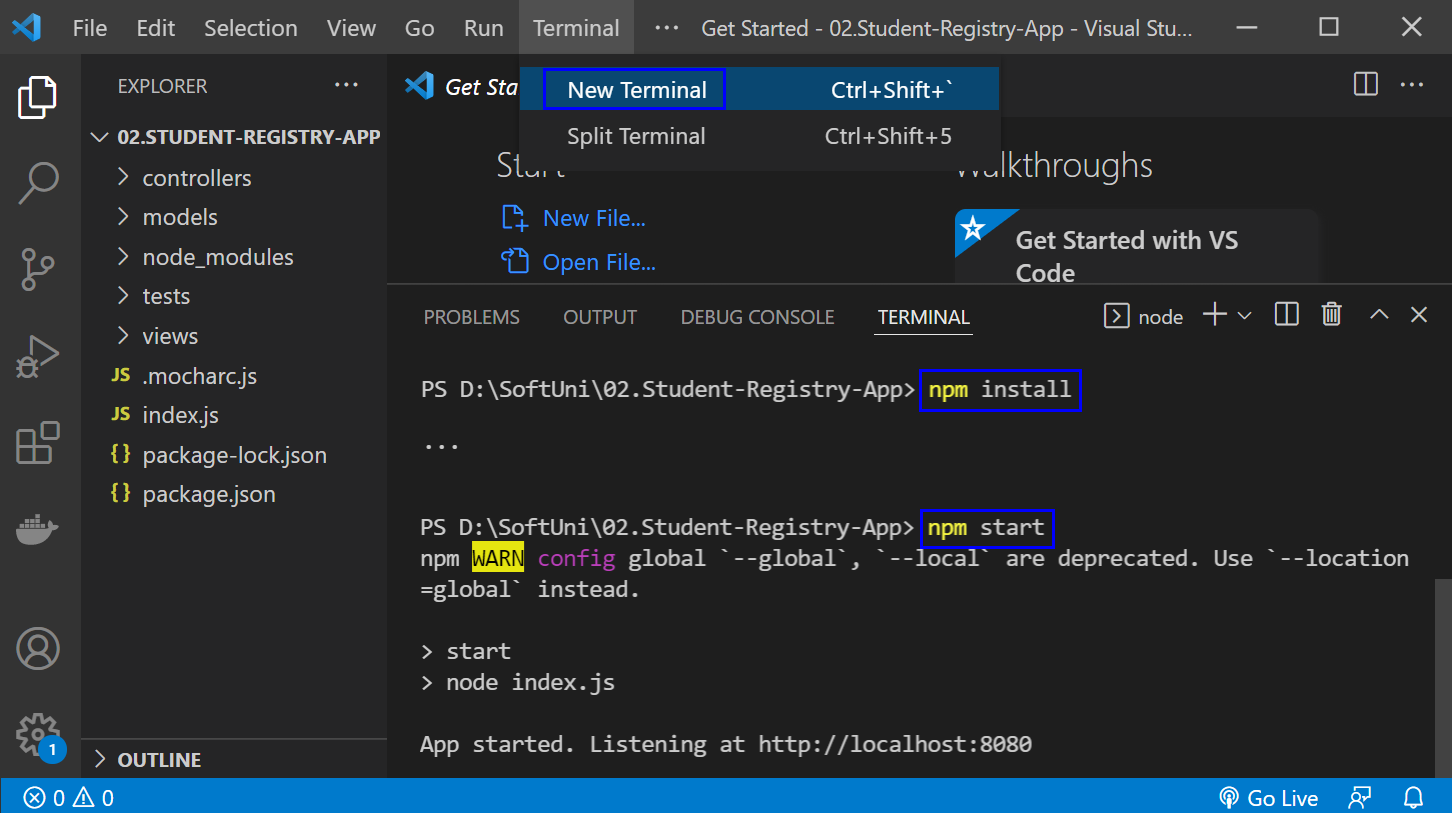
Exercises for the ["Back-End Test Automation"](https://softuni.bg/trainings/4399/back-end-test-automation-february-2024) course @ SoftUni

## "Student Registry" App – Node.js App

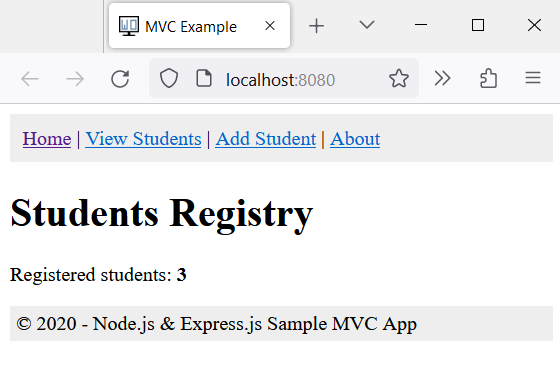
### Step 1: Run the App Locally

We have the "Student Registry" Node.js **app** in the **resources**. Your task is to **create a CI workflow** in GitHub Actions to **start and test the app** on three different Node.js versions.

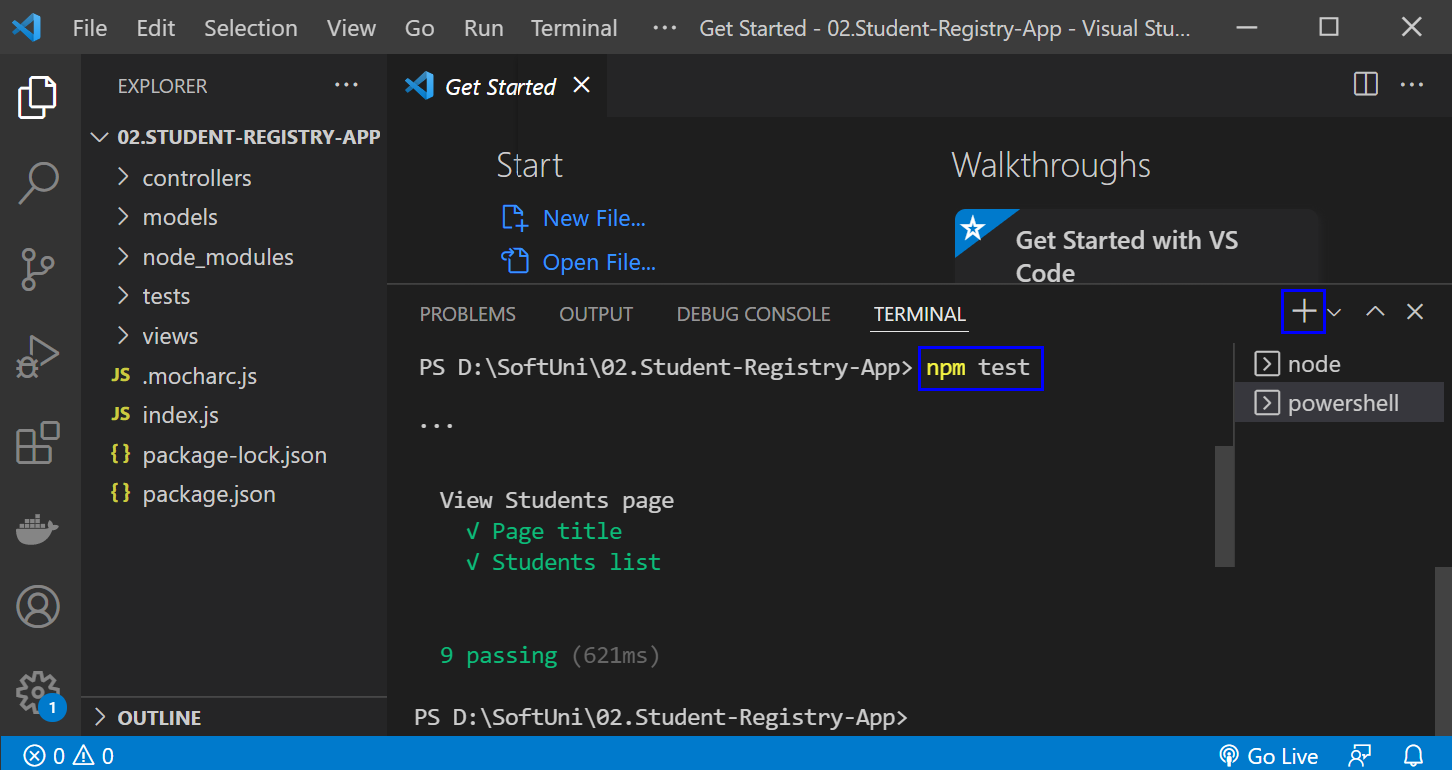
Let's first **start the app locally** in Visual Studio Code. To do this, you should **open the project**, open a **new terminal** from [Terminal] 🡪 [New Terminal] and **execute** the "npm install" and "npm start" **commands**:



The "npm install" **command** **installs app dependencies** from the package.json **file** and "npm start" **starts the app**. You can **look at the app** on <http://localhost:8080>:



Then, you can **return to** VisualStudioCode, open a **new terminal** with [+] and **run** "npm test" to **run the app tests**. They should be **successful**:



**NOTE**: if the **app was not started**, **tests would fail** because these are integration tests and are executed on the running app.

### Step 2: Create a GitHub Repo

Now you should **upload the app code to** GitHub.

It's a good practice to start using the console and not the interface of GitHub, in case you haven't started doing so yet.

If you don't have Git already installed on your machine, follow the provided installation instructions from the resources.

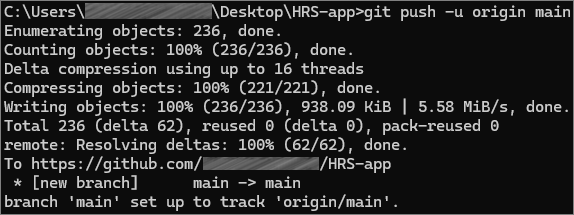
Try using the following commands in order to initialize a repository in your project directory, add the code to the repo, commit and push:











After running the commands, check you GitHub repo – the application code should be visible.

### Step 3: Create and Run Workflow

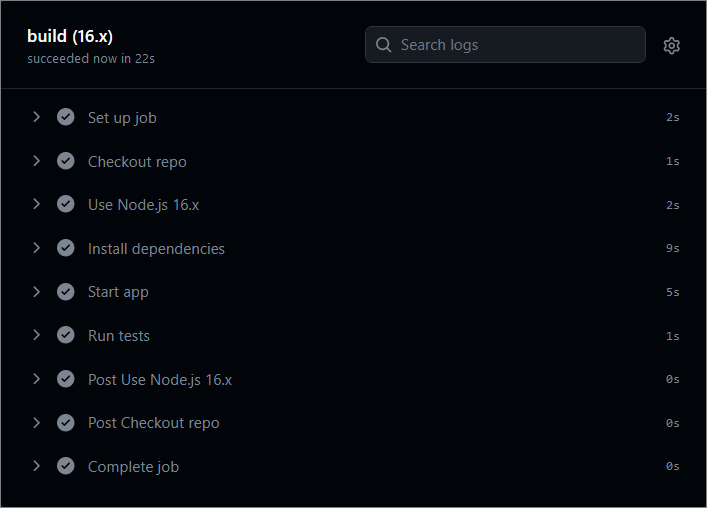
Now you should **create a** GitHubActionsCI **workflow** to **start and test the app**. You can use the **following template**:



**Before you commit** the **generated** YAML **workflow file**, you should:

* **Change the YAML file name** to something more meaningful
* **Examine the workflow**, the **job** you have and its **steps**
* **Run the job** on the **last Node.js versions**: **18.x**
* **Change the workflow name**
* **Modify workflow job steps**: you should **use the three commands** which we used above to **start and test the app**, not the ones you have in the generated YAML file or **your workflow won't be successful**
* **Add names for each step** in your workflow job

Finally, **run the workflow job** and make sure that **it is successful**:



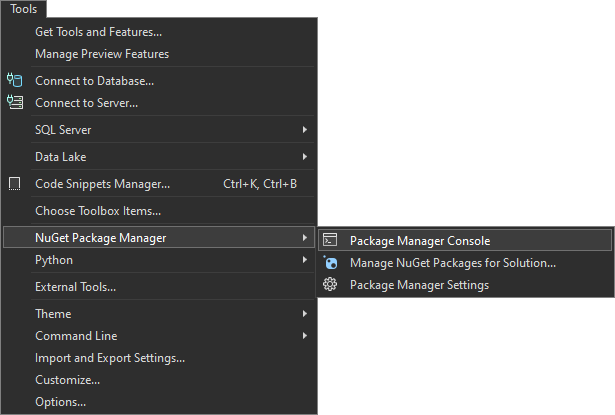
## "HouseRentingSystem" App – ASP.NET Core MVC app

### Step 1: Run the App Locally

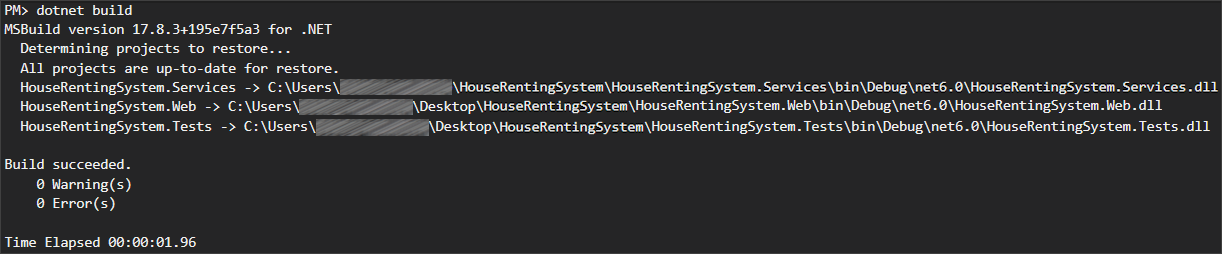
We have the "HouseRentingSystem" ASP.NET Core MVC **app** in the **resources which has some unit and integration tests already**. Your task is to **create a CI workflow** with **GitHub Actions** to **start and test the app.**

It's a good practice to first **start the app locally** in Visual Studio, in order to be sure everything works properly and as expected.

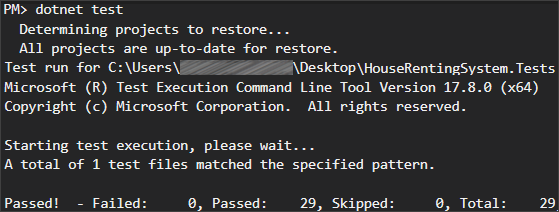
Open **Visual Studio** and from there navigate to the **Tools** menu. Select **NuGet Package Manager** and select **Package** **Manager** **Console**:



Let's first build the application by using the **dotnet build** command:



After you have **ensured** that the **build** was **successful**, you can **run** the **tests**, too, by using the **dotnet test** command:



**NOTE:** Visual Studio has built-in test runners that allow you to run your tests directly from the IDE. This is the simplest way to execute tests if you're already working within Visual Studio. However, it's **better** to get used **using** the **console**.

**After** we have ensured that the **tests** **run** **successfully**, we can proceed with the next step.

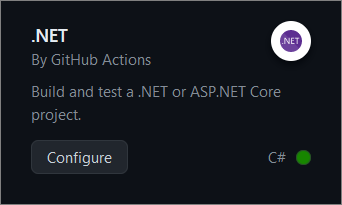
### Step 2: Create a GitHub Repo

Now you should **upload the app code to** GitHub. Try using the CLI and the commands from the previous task to add the code to the repo and commit it.

### Step 3: Create and Run Workflow

Now you should **create a** GitHubActionsCI **workflow** to **start and test the app**.

You can use the **following template**:



**Before you commit** the **generated** YAML **workflow file**, you should:

* **Change the YAML file name** to something more meaningful
* **Examine the workflow**, the **job** you have and its **steps**
* **Run the job** on .NET version **6.0**
* **Change the workflow name**
* **Modify workflow job steps**: you should **have jobs for** 
  + **Setting up .NET Core**
  + **Restoring dependencies**
  + **Building the app**
  + **Running the tests**
* **Add names for each step** in your workflow job

Finally, **run the workflow job** and make sure that **it is successful**:

