# Exercises: Selenium Basics and Page Object Model (POM)

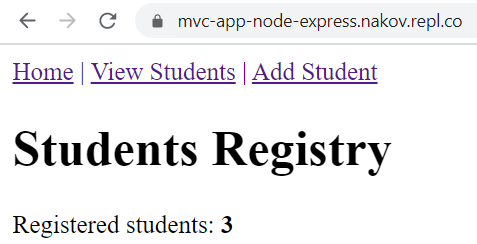
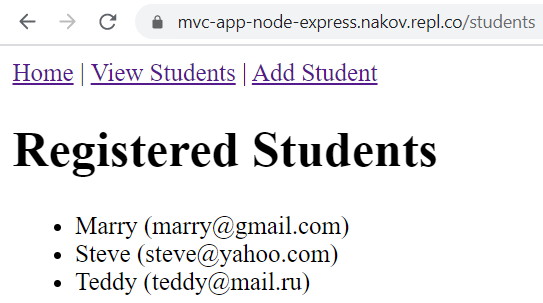
This document defines the homework assignments from the [QA Automation" Course @ Software University](https://softuni.bg/trainings/2550/qa-automation-may-2020).

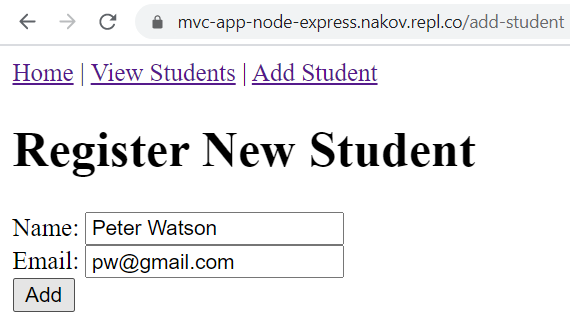
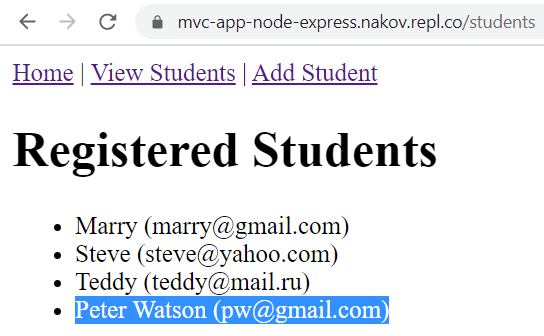
Please submit as homework a single zip / rar / 7z archive holding the source code or whatever approach is needed to finish your tasks.

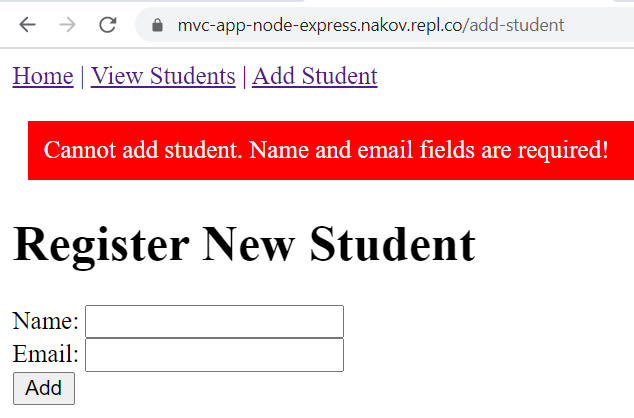
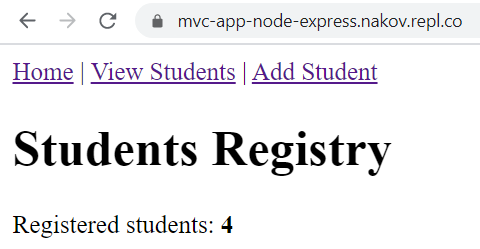
## Automated Testing of “Student Registry” App

Write **automated Selenium UI tests** for the following app, which holds a registry of users:

* Live app URL: <https://mvc-app-node-express.nakov.repl.co>
* Source code: <https://repl.it/@nakov/mvc-app-node-express>

### Hints and Guidelines

Create the following **Page Object classes**:

* BasePage
  + Constructor: BasePage(IWebDriver driver)
  + Virtual property: PageUrl
  + Method: Open() => driver.Url = this.PageUrl;
  + Method: IsCurrentlyOpen() => driver.Url == this.PageUrl;
  + Properties: LinkHomePage, LinkViewStudentsPage, LinkAddStudentsPage, ElementTextHeading
  + Methods: GoToHomePage(), GoToViewStudentsPage(), GoToAddStudentsPage(), GetPageTitle(), GetPageHeading()
* HomePage, inherits from BasePage
  + Constructor: HomePage(IWebDriver driver) + ElementStudentsCount
  + Properties and methods: inherited + PageUrl (assigned correctly)
  + Methods: inherited + GetStudentsCount()
* ViewStudentsPage, inherits from BasePage
  + Properties: inherited + PageUrl (assigned correctly)
  + Methods: inherited + GetStudentsList() 🡪 List<string>
* AddStudentPage, inherits from BasePage
  + Properties: inherited + PageUrl (assigned correctly)
  + Form field properties: FieldStudentName, FieldStudentEmail, ButtonAdd, ElementErrorMsg
  + Methods: AddStudent(name, email), GetErrorMsg()

Write the following **test methods**:

* Test\_HomePage\_HeadingAndTopLinks():
  + Open the home page:
    - var homePage = new HomePage(driver);
    - homePage.Open();
  + Assert the page **title** is correct (window title).
  + Assert the page **heading** is correct (the top heading at the start of the page).
  + Go to “view students” page:
    - homePage.GoToViewStudentsPage();
  + Assert that the current page was redirected to the correct page:
    - var studentsPage = new ViewStudentsPage();
    - Assert.IsTrue(studentsPage.IsCurrentlyOpen());
  + Return back to the home page:
    - driver.Navigate().Back();
  + Test the same way the other links:
    - GoToAddStudentsPage() and GoToHomePage()
* Test\_ViewStudentsPage\_HeadingAndTopLinks():
  + Implement it just like the previous test.
* Test\_ViewStudentsPage():
  + Open the “view students” page.
  + Assert that the GetStudentsList() returns non-empty list of students.
* Test\_AddStudentPage\_HeadingAndTopLinks():
  + Implement it like Test\_HomePage\_HeadingAndTopLinks().
* Test\_AddStudentPage\_ValidData():
  + Open the “add students” page.
  + Generate a unique student name and email:
    - string newName = "New student" + DateTime.Now.Ticks;
    - string newEmail = "email" + DateTime.Now.Ticks + "@email.com";
  + Invoke AddStudent(newName, newEmail).
  + Assert that the current page was redirected to the “view students” page:
    - var studentsPage = new ViewStudentsPage();
    - Assert.IsTrue(studentsPage.IsCurrentlyOpen());
  + Assert the “students list” holds the new student:
    - studentsPage.GetStudentsList() collection should include the new student.
* Test\_AddStudentPage\_InvalidData():
  + Open the “add students” page.
  + Try to add invalid student: invoke AddStudent("", "invalid@email").
  + Assert that the current page was not redirected:
    - Assert.IsTrue(addStudentsPage.IsCurrentlyOpen());
  + Assert that an **error message** was displayed:
    - addStudentsPage.GetErrorMsg() != null

**Run** the tests and ensure they work correctly.

This is how your final set of tests may look like:

