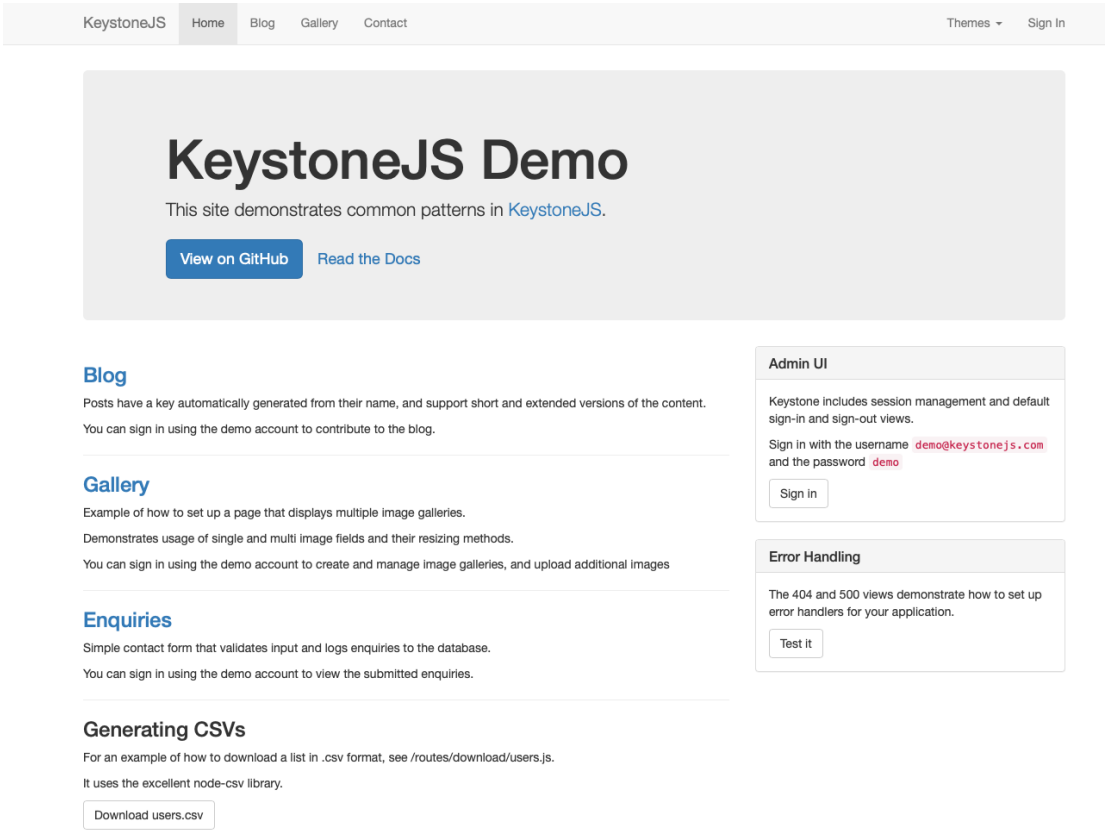
Lab 4 Automated Web Application Testing

Lab Objective:
This Lab aims to help students to learn the concepts of web application testing and to be familiar with an automated capture/replay tool for web application functional testing.

1. SUT Description:
The system to be tested for this lab is KeystoneJS, an open source framework for developing database-driven websites, applications and APIs in Node.js. KeystoneJS makes it easy to create sophisticated web sites and apps, and comes with a beautiful auto-generated Admin UI. To learn more about this platform and its features, please refer to KeystoneJS at GitHub.

2. The Testing Tool
The testing tool will be used in this lab is Selenium. You can use Selenium with Python or Selenium IDE to development your tests. Python is the popular language to development web crawler. You can find a lot of tutorials from Google. Selenium IDE can be installed on Google Chrome and FireFox as a plugin, you can use your preferred browser.

You will need to run KeystoneJS application on Docker. Please install Docker Desktop for Windows or Docker Desktop for Windows on your machine.
After completing the Docker Desktop installation, please run the Docker Desktop application and follow the instructions at Docker Hub to run the KeystoneJS.
If everything goes well, you may see the application is running at http://127.0.0.1:3000/



3. Test Project and Test Suite Generation
In this lab, you will need to develop test requirements for the features of KeystoneJS. You are also required to design test cases to verify the test requirements. These test cases should be organized in terms of the structure of the features to be tested. Each test requirement should have at least one test case to verify (i.e., cover) the requirement. For each test case, you need to describe the test scenario, including the Preconditions (if applicable), Success Guarantee, Main success scenario and Extensions. Since KeystoneJS supports a lot of features, you will be required to test only the following features in order to limit the necessary test effort.

- Post Features
  - Create post on the Admin UI page
  - Edit post on the Admin UI page

    - Delete post on the Admin UI page
    - Search posts by keyword on the Admin UI page
- Comment Features
    - Create comment on Admin UI page
    - Edit comment on Admin UI page
    - Delete comment on Admin UI page
- Category Features
    - Create category on Admin UI page
    - Show posts of the specific category by pressing category name on the "Blog" page
- Enquirie Feature
    - Create enquiry on the "Contact" page
    - Delete enquiry on Admin UI page
- User Feature
    - Create a new user on Admin UI page (Name, Email, Phone, and Password must be set when creating the new user)
- Please note that you must do the assertions for each test scenario to verify the execution results are correct.

4. Test Plan (2~3 pages would be sufficient for the test plan)
To begin with, a plan must be created. Document this test plan, as it will be included with your lab report. This plan should include the following information:

- **Summary/Scope** (briefly describe the application under test and the test requirements or test goal)
- **Features to be tested** (lists each component or feature to be tested)
- **Success criteria of completing the test** (i.e., test pass/fail criteria used for deciding whether the features are PASS or FAIL)
- **Test environment and/or infrastructure** (including hardware, software, test data, etc.)
- **Test approaches** (e.g., describing what approaches will be used to design test cases for each feature to be tested (e.g., user scenarios, ISP, etc.), how to evaluate the adequacy of the test (e.g., coverage), and what tools will be used to perform the test; you may use multiple testing technique to design the test cases to ensure that all the test requirements can be satisfied and the quality of tests can be achieved.)
- **Testing tasks** (or activities required for implementing the approach)

5. Lab Report
The Lab report must include (but not limit to) the following sections:

- Test Plan: briefly describe test requirements, planed test activities, **approach**, and success criteria for the test.
- Test Design: use approaches combining ISP, graph (**use cases**), and logic methods. Below is an example of use case template.

  Please use the template to specify your test scenarios, including the Preconditions, Success Guarantee, Main Success Scenario, and Extensions.

Use Case Section	Comment
Use Case Name	Start with a verb.
Preconditions	What must be true on start, and worth telling the reader?
Success Guarantee	What must be true on successful completion, and worth telling the reader.
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.

- Test Implementation: Selenium test scripts of your test cases and execution result.
- Test Result: execution results and screen snapshots of Selenium test scripts.
- Test Coverage (optional): you may use SimpleCov for Ruby on Rails, Blanket for client-side Javascript, or istanbulfor server-side Javascript to measure code coverage.

Please convert your Lab report to `.pdf` file, `.docx` file will not be accepted.
The Lab report and your Selenium test script must put in the corresponding directory in your GeoProject repository (e.g. `GeoProject -> LabReport -> Lab4`).
Make sure you have pushed your code and Lab report on the GitLabsuccessfully.

Finally, you must open an new **issue** on TA's GeoProject with the following information and format to notify TA that you have done your lab.

TA will run all of your test cases on a clean KeystoneJS environment, which is a whole new Docker container. You have to make sure that all of your test cases can run correctly without any error and pass every assertion from the first one to the last one . If you develop your tests in Python, you may consider to organize your tests into test suites.