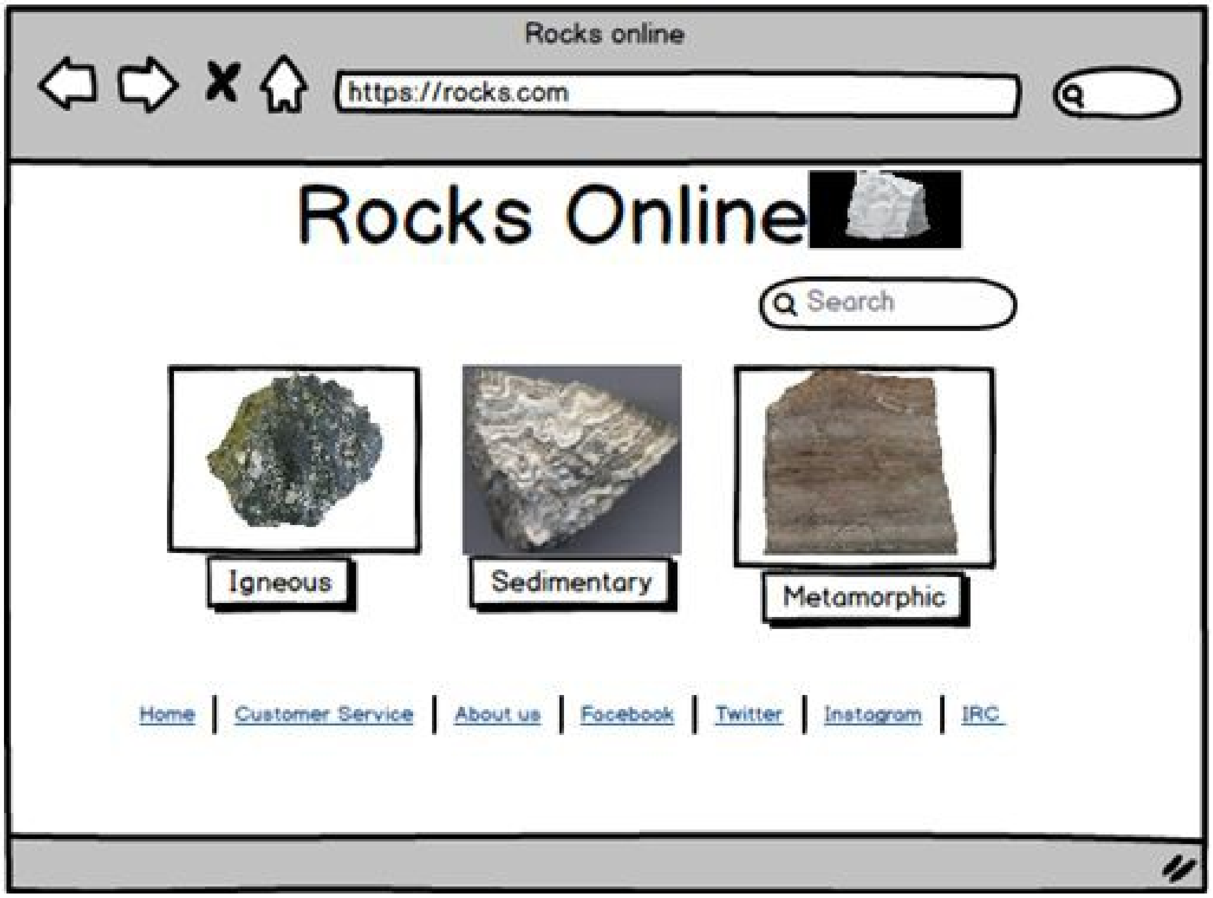
FIT2107
Week 8 Workshop

This week we practice scenarios related to Blackbox and Whitebox testing. The mocking scenario is optional.

Work in pairs.

More Black-box testing

This following question was from the FIT4004 unit from 2016.



Apply blackbox testing techniques to identify the partitions and test cases to test the Rocks Online application.

More white-box testing

Part 1

There is another source code file on Moodle, which has a function binsearch() implementing a binary list (array) search.

Write down a *test suite* that achieves 100% *branch coverage* for binsearch().

Part 2

You have a Python function that takes two strings as inputs, head and tail. It has the following line of code in it:

```
 if head and tail and not any(x.isupper() for x in head):
```

Come up with test inputs (values for head and tail) that achieve MC/DC coverage for this line.

Hint: `any(x.isupper() for x in head)` returns `True` if there is an uppercase letter in the string `head`

Mocking (Optional)

Last week we wrote unit test cases for the PriorityQueue in Python.

We are going to add an additional method to the class - `tojson(self).` This should return a string with a representation of the priority queue as a JSON[1] object.

To implement this method, you should use the built-in `json` module[2].

Your method should contain no more than a couple of lines of code.

Implement `tojson` and write at least two unit tests for it. Have at least one that uses the `mock` module to check that the correct arguments are being passed to json.dumps()

Extend the code from last week and write mocks.

[1] JavaScript Object Notation - https://docs.python.org/3/library/json.html
[2] https://docs.python.org/3/library/json.html