**PART I**

The tests were written in JavaScript using the Protractor framework.

They are perfectly functional, but serve as proof of concept, and perhaps not production ready.

I have used the Page Object Model, even though there is only one main page to work with.

The credentials are stored in a json file to provide a small layer of security.

The tests require a selenium server to run on the machine they run on.

That can be changed in the conf.js file.

The conf file also contains instructions to the chrome webdriver to ignore password management and deny notifications.

I have used both implicit and explicit waits.

**PART II**

The idea behind creating page objects is to separate the page elements from the logic of the tests.

Selecting which objects should be inside the page object model comes down to chosing what elements

are actually going to be used in the automation. A good idea is to only pick the most commonly used

objects. In this case those would be username and password fields to automate the login functionality,

the category buttons Slots, Bingo, Poker, Casino. I would also select the close button for the popup windows.

Ideally every single webelement coud be used, but it all depends of the test coverage you want to achieve and the high risk areas. In this case non-functional items such as footer notes or copyright

statement would probably be overkill to document and use in the automated tests.

I have provided a practical example of how I used the page object model in the tests i wrote for the first section.