Note: I solved all these problems as if I would have in a hackathon, which means that I used as many libraries as possible to minimize the work I had to do. I hope this is in the spirit of the questions. Also, these were written in Python 2.

Problem 1:

First, I imported selenium web driver, which makes the problem trivial. This library is essentially just an api that allows you to make calls to a browser that you have installed. In order to get it to work, I downloaded the chrome web driver and placed it in my local directory. Then, I make a call for chrome to get the webpage at the url, and then import the page source. Chrome does all the heavy lifting of executing the javascript before passing it to me. Then I just write that html to a file. If I wanted the actual javascript, or the pre-rendered page, I would use BeautifulSoup.

Problem 2:

First, I looked at the data in excel and noted that there were only 100 examples, and only 35% of them were positive classifications. I also noticed that someone had deliberately inserted a bunch of special characters and escape characters that would have to be removed. I imported the data from the csv using pandas and then converted it to a numpy array, throwing out the id numbers which I assumed to be random. I split the array so I had a data array and classification array. Then I removed the special characters from the data array using python’s regular expression library. I used a bag of words representation, running the descriptions through a count vectorizer which returned a one-hot vector, which I than converted to it term frequency–inverse document frequency. This is a standard representation which works to minimize the weight of words which appear frequently and the individual weight of words which appear in longer documents.

To produce my final model, I tested a naïve gaussian bayes classifier, a gradient boosted tree classifier, a multilayer perceptron and a support vector machine using 35 fold cross validation. I could have used 10 but I figured I might as well be precise, and my laptop ran it in under a minute. Note: I manually tuned the hyper-parameters for all but the naïve bayes. The svm slightly outperformed the mlp, which isn’t surprising because only 100 data points should be linearly separable. I then trained the svm on the full set of training examples.

Cross-fold validation suggests my accuracy is around 90%, which compared to a constant classifier score of 65% is pretty good. However, at only 100 training examples, I can’t imagine this model is very generalizable.