Cheating Bot Detector

Concepts & Early Prototypes

Grein, Cameron Gareth
Department of Electrical Engineering
and Computer Science,
Oregon State University
Corvallis, Oregon, United States
greinc@oregonstate.edu

Yidong Lin
Department of Electrical Engineering
and Computer Science,
Oregon State University
Corvallis, Oregon, United States
linyid@oregonstate.edu

I. Background

We live in interesting times that require us to learn things in a different way than previously. Online school has taken prominence in both college and primary school settings. This environment is very different from how we learned before. One aspect that needed some adjusting is the concept of taking tests and doing homework remotely. The issue arises is the fact that students tend to cheat or attempt to cheat more often. There are various sources all over the internet where students can find the answers to the homework or even exams. This causes a problem for the Universities for both their own copyright as well as the academic honesty of the students.

Our project partner is Carley Ries. Our project is designed to scour the internet to detect websites that are posting documents and images that are related to Oregon state university courses. It would help the university notify faculty members when their teaching materials are compromised. This would also allow us to notify the web hosts that they need to take the documents down. As such, our website cheating detector would help to improve the academic honesty of students, as well as to allow the university to have a long term solution to exam, homework, and other course material from being posted onto the internet.

II. Vision

Growth hypothesis:

Due to the continuous improvement of the Internet, learning aid websites such as Chegg and Coursehero have appeared. They claim to be a social learning network. Users can access old test questions, assignments, answers, notes, etc. through these sites. Of course, these are not free, users need to pay a certain amount to continue to access the content. However, you can still browse the content by publishing the materials, and the 40 documents will provide users with 1 month of free browsing. This has led to the leakage of teaching content in many schools. The reason why our product is beneficial that not only the content of Oregon State University's courses has been compromised. On these websites you can find almost all university courses, tests, and answers. Our current work is only for Oregon State University, if the product can achieve the desired effect. The beneficiaries of the product can be more schools.

Value hypothesis:

The reason why users are encouraged to adopt our products is that the products can find websites that post content about Oregon State University courses, thereby reducing the number of manual searches by staff, reducing students' academic dishonesty, and protecting the school's course content.

Functional requirements:

- Url. Need to provide seed url as the starting point of crawler.
- Set limits on crawling pages. It is necessary to limit the upper limit of crawlers to make crawlers stop operations in time.
- Keyword settings. Set crawl keywords.
- Save current search path.
- Server connection. Need to connect to the specified server, the server should not open all day

Non-functional requirements:

- Performance - efficient execution without crash
- Reliability - suspended programs when abnormal operations are found.
- Scalability - provide interfaces for other software to quote
- Environmental - only support win10 maybe.

III. Prioritized Project Constraints

Time:

Regarding time, it is important to allocate our time such that we can finish this project on time. We only have a specific amount of time each week to work on it as we have other classes and responsibilities that we also need to consider. Considering, about halfway through our timeline we should have a basic version of our project working, then for the 2nd half of the timeline we can perfect it and add more features. Of course, we will be limited by the amount of time we have each week to implement this.

Resources:

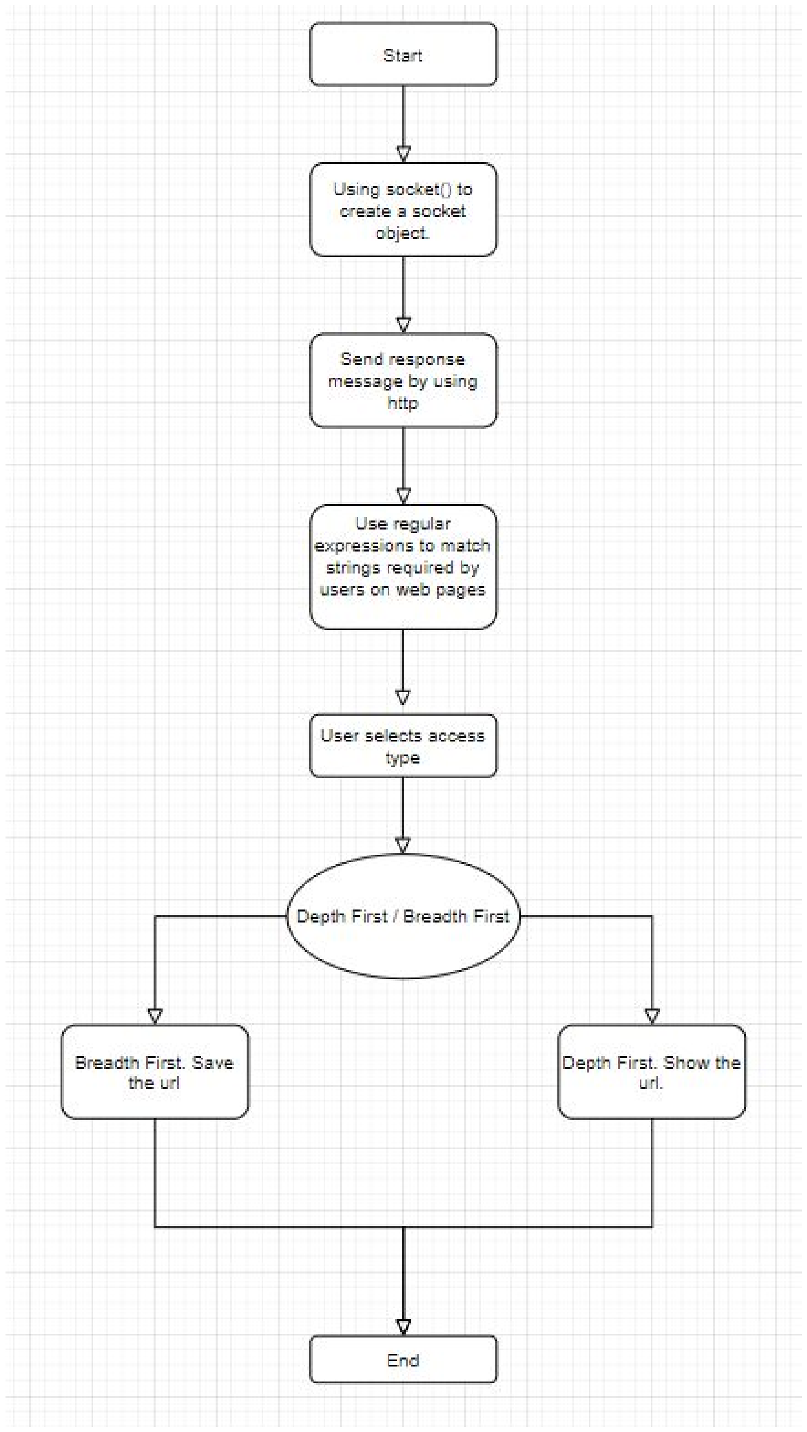
Budget wise we shouldn't have any issues with the development when it comes to money. We should be able to use mostly open source materials and software to develop the bot. I cannot think of any things that we will need funding for. This kind of development is generally lower cost.

Scope:

In relation to the scope of the assignment, we should be able to make a basic scraper that can identify websites that contain the course information. Scraping things such as scrape should be quite achievable in the time span that we are provided. Doing things such as scraping images might be a little too ambitious for the time that we have. There also is the question of if we want this to be used as a detector to manually inform that the content needs to be taken down, or are we planning on implementing an automatic system. An automatic system may be out of the scope of the time that we have to complete this assignment.

IV. Scope

Process Flows:


Start
Using socket() to create a socket object.
Send response message by using http
Use regular expressions to match strings required by users on web pages
User selects access type
Depth First / Breadth First
Breadth First. Save the url
Depth First. Show the url.
End

User Stories:

- As a user/partner, I need the product to automatically search for keywords so that I can reduce my manual search time.
- As the development team, we need a good management system, so that it is easy to manage and expand the product if it is needed.

V. Iteration Plan and Estimate

The first sprint will be making our bot discover which systems and languages we want to program it in. This is an important step as it will determine all of which we will later implement. We should spend a decent amount of time thinking about the decisions we will make here. This will basically be the first sprint.

The 2nd sprint will begin the raw implementation of our code. We need to build an API that we can use in order to implement the bot. In other words, we need to implement the backbone of the bot. We need to ensure that every function and implementation in our code works and does what it is supposed to do. Basically, the bot should work in a basic form when this part is done.

The 3rd sprint is where we will build the frontend of the bot which will allow the user to use the bot. A user doesn't know how to do things such as run scripts, we need to make it usable for the general audience, which will most likely be faculty and students. Most of these people don't have technological skills like we do, the goal is to make it as easy to use as possible.

The last sprint will be finalizing everything and making sure it works. We may want to upload it to download on Oregon state's website hosting. After verifying that everything is working correctly, we can allow users to download and start utilizing our bot.