DisciPlan

Adam Brostowicz (adamii@stanford.edu)
Jeff Garnier (jeff1731@stanford.edu)
James Ross (jross3@stanford.edu)
Danny Thomson (danthom@stanford.edu)

Description of Project:

Our project will be a free browser extension for both Firefox and Google Chrome. The goal of the extension is to collect information about the user's internet usage, to display that information to the user using charts and graphs, and allow the user to create custom plans to change their internet usage however they see fit. Ultimately, DisciPlan will help user's reclaim their lost time from unproductive time sinks that the internet easily provides.

The extension's main job will be to collect internet usage statistics from its users, mainly what websites they visit and how much time they spend on each. It will also categorize the websites that the user is visiting (social media, entertainment, academic, etc). If we cannot determine the category of any website that the user frequents, the user will be able to "tag" a website as a certain category. The extension will also have an interface that displays graphical information about the user's internet usage. The graphs will split the information based on website categories so that the user can see exactly how much time the user is spending on both specific websites and in each category. The graphs will also display category trends over time, so that the user can see any progress the user might make.

In addition to data collection and graphical displays, the extension will also allow user-defined category goals. For example: if a user wants to spend only one hour on social media per day, the user would specify that within the extension's goals page, and the extension would take certain actions after the user reaches the limit. The extension will have a different actions it can take, ranging from blocking those websites after the time limit runs out or simply sending a notification when the time is up. Within the goals page, the user will also be able to add or remove specific websites from the block list, or change the categories.

Each user will have a DisciPlan account. When a user logs into the extension from a different device or browser, all of the user's settings and information will be preserved. This will ideally allow for a seamless experience, and won't let the user cheat the system by switching to another browser, as long as the extension is installed on both of them.

Need For the Product:

Our product is needed because of the immense number of distractions available on the internet. It's really easy to get distracted, lose track of time, and end up spending time getting absolutely nothing done. Oftentimes people only need a small reminder to keep them on task, and any sort of barrier introduced between them and a website like Facebook is probably enough to stop them from wasting hours reading through all of their friends' posts. In addition, it's one thing to know that you have wasted some time on social media sites, but it's entirely another to have an app show you exactly how much time was spent on an unproductive website. Being shown not only how much time a user has spent unproductively, but also improvements will help keep people motivated and eventually learn to spend their time on the internet more productively.

Potential Audience:

Our audience will be people who want to stop wasting time on social media in order to improve their productivity. There is a fairly large audience who recognizes that they spend too much time being unproductive online, and our tool would help alleviate that problem for them. The level of technical sophistication required is quite minimal. All the user needs to know is how to search for extension on Firefox or Chrome. From there, it does not require a high level of sophistication to find and change settings within that extension. Furthermore, the graphs will be visually pleasing and easy to understand for someone with any level of technical sophistication.

Discussion of Competing Products:

DisciPlan is a combination of two different Internet browser extensions. In Google Chrome, these extensions are called StayFocusd and Time Tracker. StayFocusd is an extension that allows users to visit unproductive websites for a set amount of time everyday. These unproductive websites are defined by the user. After exhausting the allowed time for that day, the extension will prevent the user from visiting these websites. By thwarting quick access to these websites, the extension stops users from easily getting off track from their work. Time Tracker is an extension that tracks the user's web usage in the background. The user can create different categories in order to track the links in that category as a whole. For example, a social media category could have Facebook, Twitter, etc. After collecting the data, the extension will provide different graphs that show how the user is using the internet. There are different setting the user can use to show their data.

DisciPlan will take many of the features of StayFocusd and build upon them. StayFocusd does not provide many controls to adjust the behavior of the extension. A user can only define what websites should be restricted and how much time allowed visiting these websites in a day. To improve the extension, DisciPlan will allow different amounts of time for different websites or categories, provide different actions for visiting a unproductive website after exhausting the allowed time, and offer a more clean interface to interact with these settings.

Since DisciPlan will be tracking the user's web usage, DisciPlan will be able to use this data to make recommendations about the user's web usage. Our tracking features will provide more in-depth information about the user's usage than the other web usage statistics extensions. Unlike Time Tracker, we will integrate this information into the new tab page of the browser to ensure that this information will seen and used by the user.

Major Technologies Used:

We plan to to make our extension first for Chrome, and then for Firefox. We chose Chrome first because Chrome is the default browser with our team. As we develop the extension for both browsers, we will require knowing the differences between extensions development for these browsers. Chrome, for example, has easier developer environment setup and debugging thanks to its developer dashboard.

Since we'll be storing statistics on the user's general internet usage, these statistics will be sent back to a server which we will write using Node.js. We chose Node.js because it is build to handle multiple connections at a time. Since our extension will be sending data to the server frequently, the server will have multiple connections at any time. This server will be responsible for writing stats to (and later reading stats from) a database. We will use MongoDB as the database to host our data because MongoDB makes it easy and fast to interface with Node.js.

When users want to view their usage stats, we'll produce pages using HTML and CSS in tandem with JavaScript. Since these pages will include graphics like charts comparing time spent on different categories of websites, we'll use an open source JavaScript charts and graphs library (this may be D3.js, Google Charts, or another charting library). We will the front-end library BootStrap in order to make development of the page easier and better. Bootstrap will allow us to create a well designed page.

Since we are using Node.js and building browser extensions, our project will be largely coded in JavaScript. There will be HTML, CSS, and more JavaScript for the web pages will provide for the user to view their data.

Resource Requirements:

To host our server, we will use the Amazon Web Services (AWS). One of the major advantages of using AWS is that it abstracts many details of hosting a server away from the developer. We will not have to worry about building the infrastructure to host the server, and we will not have to work with any hardware. As a result, this will allow us to focus on developing the server instead of managing it. Unfortunately, AWS is not free, and we will need some AWS credits in order to host our server.

Potential Approaches:

DisciPlan could be released as an independent desktop application or directly integrated into the browser’s source code. However, we decided that a extension is the best approach for this project.

Creating a separate desktop application provides more control of the computer to the developers. However, since DisciPlan's focus is rooted in the browser, we do need that extra control for the features of our project. Instead, that extra control will complicate our code because the application will have to micromanage the browser from outside the browser environment. On the other hand, an extension will be directly integrated into the browsers, and the extension is only browser dependent. We would have to separate desktop applications for OSX and Windows.

The other option of modifying the browser’s source code will also provide more control over the browser. However, there are two problems associated with that approach. First, there is a chance we could introduce security issues by changing the source code of the actual browser compared to installing an extension. An extension has access to limit resources. Thus, if the extension were hacked, the hacker would not have access to all the browser’s resources. Meanwhile, if the code were directly integrated into the browser, a hacker would have access to all the browser’s resources. With this kind of access, a hacker could potentially learn a user’s personal information. The second problem is that users are more likely to download an extension to their browser over a separate browser. A separate browser is not likely to be compatible with other browser extension and will be more difficult to update.

Assessment of Risks:

One potential risk comes from the fact that none of us have ever built a browser extension. There seems to be good documentation on the Internet about how to build extensions for both Chrome and Firefox, but learning how to do this may take longer than we expect, or we might not be able to get it working at all. If we have trouble setting up the extensions, then we are losing valuable time that could be spent building a better extension with more complete and aesthetic features. Since we are building an extension for two different web browsers, we may run into problems with different browser settings. For example, we might have no trouble getting the extension set up on Chrome, but we cannot get it working in Firefox. In this case, we lose a lot of the core functionality because our server would not be needed to share data across different browsers. In order to reduce this risk, we will be diligent about quickly learning how to set up an extension for both Firefox and Chrome. This will allow us to actually start building our extensions as early as possible.

Another possible risk is that we will not be able to get the server talking to the Chrome and Firefox. We have little background setting up a real server outside of class assignments. If we cannot get the server working properly, we will not be able to have the different browsers share the user's browsing usage information. One of the key features of DisciPlan is that the user can use the extensions for Chrome and Firefox to track of all of their Internet usage statistics regardless if they switch browsers or computers. Without the server communicating to the browsers, this will not be possible to achieve. In order to reduce this risk, we need to have a clear plan for developing the server. We will accomplish that goal by setting milestones throughout the quarter. If we have trouble setting up the server, we may need to get guidance from someone who has more experience than us. The major risks in this project are learning how to set up extensions in Chrome and Firefox and a server that we can access from these browsers. We need to start these tasks early so that if we can handle any problems. In doing so, we have more time to actually work on the extensions themselves.

Next Steps:

First, we will begin by building Chrome and Firefox extensions. Once we construct a basic framework for our extensions, we will implement the code to collect and handle user data. After reaching that point on the extensions, we will build our Node.js server. Since we will host the server on AWS, we will have to take time to setup the Node.js on AWS. Once the extensions and the server are setup, we will make the extensions interface with the server to query and store data. After we have all of this working, we need to design the UI for the extensions in order to make them easy to use. At that point of the project, the remaining steps will be fine tuning the system.