Engineering Privacy in Software
Problem set #1: Hello, World!

Reminders:

- You may work with others on your team, but not outside of your team.
- **EXCEPTION** for this assignment: it is fine to have additional help with getting your computing environment set up, getting software installed, and / or debugging. Just be sure to note who helped you and where you needed help.
- Still stuck? See Canvas for office hours.
- References for code or *ideas* that are not your own must be provided. Academic Integrity Violations nearly always result in failing the course, not just the assignment.

The goal of this assignment is to help you learn a set of commonly used software engineering tools. For some of you this will be basic; for others it is a firehose of new things. Please be patient with yourself and others. No one is born knowing any of this; we all learn over time. For the parts that are easy for you, take advantage of this opportunity to help classmates – which will help you learn in greater depth, and be good practice for technical communications skills. If there are topics new to you, practice asking for and receiving help. You will undoubtedly need to do so throughout your entire career. Get comfortable with it.

Part 1, Git

Observation: students often think they "know GitHub" but actually do not understand it well at all. In particular, be sure you understand directory structures, merging, and conflicts.

That said, some student do already know enough to get through this course and beyond. To accommodate different starting points, you have no homework due for Git. But you are highly encouraged to teach yourself (or review) GitHub, and **there will be a quiz** to help motivate you. One good resource: <https://www.w3schools.com/git/default.asp>

To understand Git (and how it differs from GitHub) it often helps to use the command line. For OS X users, Terminal ships pre-installed. For Windows, you might try PowerShell.

Here is an example of checking the version number and the path to the installed version of Git from the command line (note that my prompt is `am40$` and yours will be something different)

```
am40$ git --version
git version 2.24.3 (Apple Git-128)
am40$ which git
/usr/bin/git
```

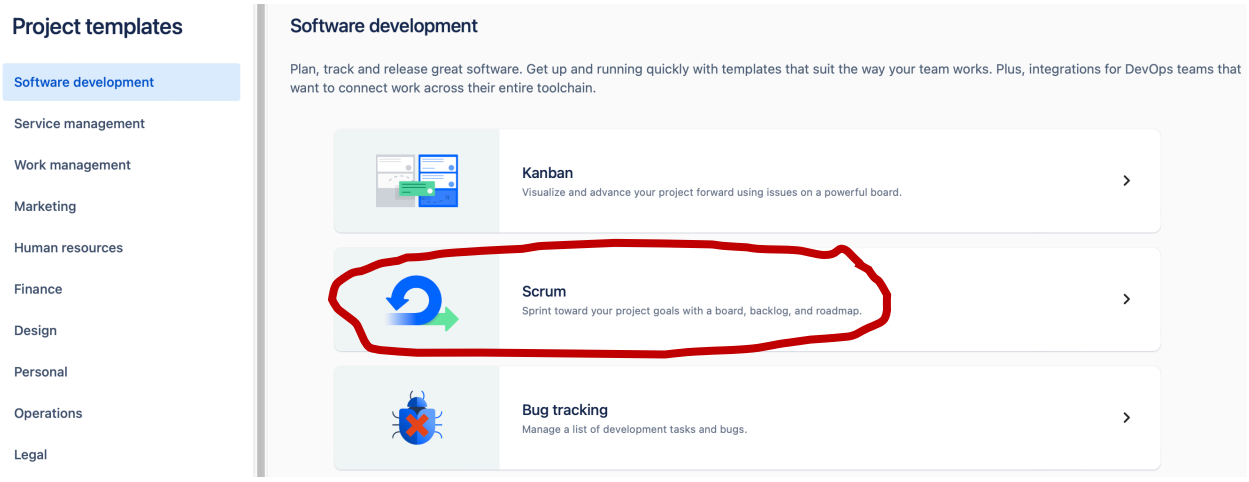
Optional but useful things for your team to try out…

A. For each person on your team: install (or update) Git, and check the version number from the command line.
B. Create a new repo for this course. Be sure all teammates can use it.
C. Create at least two branches, one for "production" (e.g. working code) and one for in-progress code.
D. Edit a simple text file while a teammate also edits it, and then resolve conflicts.
E. Revert to a prior version of a checked in text file.

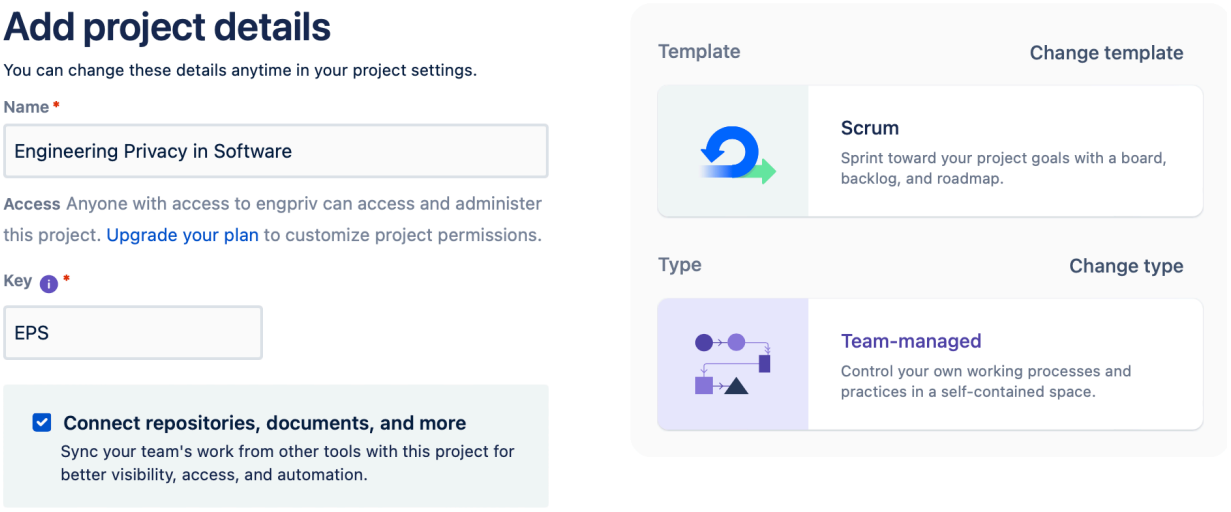
Part 2, Jira

As you organize your work, you might keep a to do list. How do software teams organize work across the whole team? Often with tools like Jira. You can use a free trial for this course, available from <https://www.atlassian.com/software/jira>

There are a zillion questions asked while you get Jira set up, but you can change things later so do not be overly concerned. Worst case, you can start over completely. Feel free to experiment. The Scrum template is likely most helpful for your team:



You might try something like this, and link to your GitHub repo:



There are many tutorials and help files. This assignment is a good chance to begin to explore how Jira works, and to talk through how you might use it as a team in this course.

Your assignment is to turn in (on Canvas) the following files:

- Create a Jira Project. For the team, submit a screenshot that shows you have created four sprints: (1) Tooling, (2) Specifying, (3) Implementing & Demonstrating, (4) Concluding (feel free to use dates from the syllabus, or change them later.)
- Create at least two tasks per team member, one in the backlog and one in the Tooling sprint. Submit a screenshot (or more than one) of the Backlog screen to show this. To make this actually useful, you could look ahead to the rest of this assignment, but placeholders will receive full credit.
- Start the Tooling sprint. Submit a screenshot of any team Board with at least one task in progress and at least one task done.

Part 3, Browser extensions

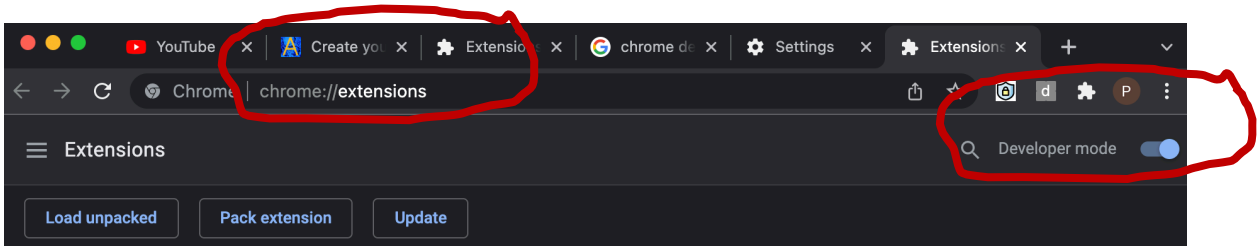
When working with a new programming language or computing environment, it is traditional to create a program that displays "Hello, World!" as a quick test to make sure everything works correctly. Getting things set up can take longer than you expect, and can be more frustrating to debug than just about anything else. Do not get discouraged. Every team member is strongly encouraged to try to get "Hello, World!" working alone, asking teammates for help only if stuck for more than 20 minutes.

While you can complete this assignment without using Git and Jira, you are encouraged to consider this as an opportunity to get more comfortable with both tools as a team. The time you invest now will let you move more quickly together in later assignments.

These instructions are extended from "DIY Adblocker - an introduction to write chrome extensions," Adrian Stoll (July 5, 2017) <https://www.adrianstoll.com/dyi-adblocker/> updated with some advice from "Chrome Extension Tutorial: Migrating to Manifest V3 from V2," Shahed Nasser (February 11, 2021) < https://blog.shahednasser.com/chrome-extension-tutorial-migrating-to-manifest-v3-from-v2/> and "chrome.declarativeNetRequest," Chrome Developers, Google. < https://developer.chrome.com/docs/extensions/reference/declarativeNetRequest/>

Everything below is for Chrome; feel free to use a different browser. Screenshots are on OS X; feel free to use a different OS. **Tip:** if you do not want to mess up your Chrome preferences, create a second user account on your laptop and login to that user account first.

1. Use the developer mode for Chrome. One way to do this is to visit the URL `chrome://extensions/`
and slide the Developer mode toggle (in the upper right corner) to on:



2. To get started, create a new directory with the name of your extension.

3. Add the file `manifest.json` into the directory you created. The file `manifest.json` lists the components of the extension and describes what permissions it needs to run correctly. You can copy this file from Canvas, and it contains:

```
{
  "name": "sample domain blocker",
  "version": "1.0",
  "description": "A simple domain blocker",

  "declarative_net_request": {
    "rule_resources": [{
      "id": "ruleset_1",
      "enabled": true,
      "path": "rules.json"
    }]
  },

  "permissions": [
    "declarativeNetRequest"
  ],

    "manifest_version": 3
}
```

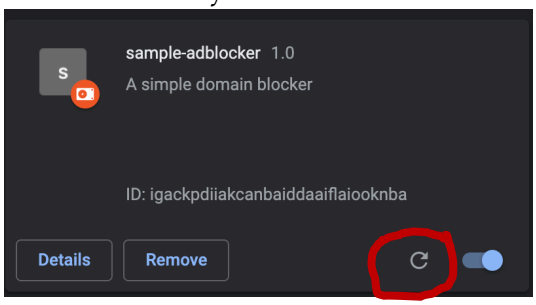
4. In the same directory, create a file named `rules.json` which is also available on Canvas, and contains:

```
[
  {
    "id": 1,
    "priority": 1,
    "action": { "type": "block" },
    "condition": {"urlFilter": "xkcd.com", "resourceTypes":
["main_frame"] }
  }
 ]
```

5. Before you try out your new extension, in your Chrome browser navigate to <https://xkcd.com> to ensure the website loads. You should see a comic strip.

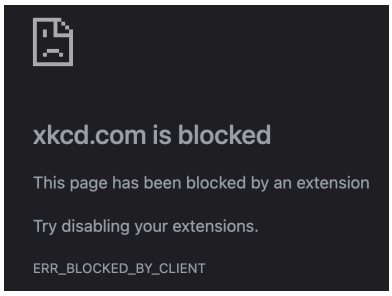
6. Ok, let's load your extension! In `<chrome://extensions>`, in the upper left corner there is a button for "Load unpacked". Click that, then navigate to the directory you created and click "Select". You do not need to pick a file, just the directory.

7. You should see your new extension listed, something like:



   Tip: notice the circled **reload** button – you will want that later when you test new code.

8. In a new browser tab once again navigate to <https://xkcd.com>. This time it should fail to load. Your extension blocks the domain! It should look something like this:



Congratulations! If you knew a website served malware, you could now block that website from ever loading in your browser. AdBlockers are built using this approach too.

Your assignment is to turn in (on Canvas) the following files:

- From each team member, a screenshot of step 7 showing that you successfully added your browser extension to Chrome.
- Once for the team, a written explanation that describes what `declarativeNetRequest` does. Pretend you are explaining this for a lawyer at your company, and do not use technical jargon. (Hint: read the documentation.)
- Once for the team, change your browser extension so that it writes "Hello, World" to the log file. (Hint: look in the documentation for `console.log`) Upload a screenshot of the log and submit your source code file.