Name:

Anjana Vakil

Do you meet the eligibility requirements outlined at https://wiki.gnome.org/Outreachy#Eligibility (if no, explain why not)?

Yes

Preferred pronoun (e.g. she, he, they):

she

E-mail address:

anjanavakil@gmail.com

IRC nick:

vakila

Internet presence (e.g. web page, blog, portfolio, GitHub, Twitter, LinkedIn links):

Personal site/blog: https://vakila.github.io/
GitHub: https://github.com/vakila
Twitter: https://twitter.com/AnjanaVakil
LinkedIn: https://www.linkedin.com/in/anjanavakil

Location (city, state/province, and country):

Saarbrücken, Saarland, Germany

Education completed or in progress (include university, major/concentration, degree level, and graduation year):

Saarland University - Saarbrücken, Germany - MS Language Science and Technology - 2015
University of California - Berkeley, CA, USA - BA Philosophy - 2007

How did you hear about this program?

Through the Recurse Center (https://www.recurse.com)

Are you applying for Google Summer of Code and, if so, with what organization(s)?

No

Please describe your experience with the organization's product as a user and as a contributor (include the information, as well as a link or an attachment, for the required contribution you made to the project you are interested in here):

I am a longtime user of Mozilla's Firefox, but prior to beginning this application I had never contributed to it. To make my initial contribution to my project of interest, "Test-driven Refactoring of Marionette's Python Test Runner", I spent a fair amount of time getting set up as a Mozilla contributor (e.g. by signing up for IRC and Bugzilla), and gathering a basic understanding of Marionette and how its tests are run, by following the tutorial outlined by the project mentor (https://wiki.mozilla.org/User:Mjzffr/New_Contributors).

Once I was set up and ready to tackle a "good-first-bug", there were unfortunately no entry-level bugs available directly pertaining to the Marionette project. The project mentor suggested I instead work on a bug for another Automation team project, Treeherder/Perfherder, so I then familiarized myself with the basics of that project, and set up its development environment. Addressing the bug (https://bugzilla.mozilla.org/show_bug.cgi?id=1217520), which involved speeding up a script used to import performance data into the web app, ended up becoming a bit more complicated than initially expected due to problems with related scripts, but thanks to the help that the bug's mentor (:wlach) provided over IRC and GitHub comments, I was ultimately able to submit a GitHub pull request making the necessary changes (https://github.com/mozilla/treeherder/pull/1353), which has been merged into the project. This Perfherder bug was a great gateway into contributing to A-Team projects, because it allowed me to use tools I was already comfortable with (git, GitHub, Django), and in the process I learned several new things, both Mozilla-specific (how to use Bugzilla, what Treeherder/Perfherder are) and general (how transactions and bulk object creation work in Django). Given how much I enjoyed working on it, I was pleased when the mentor wlach mentioned another bug related to the same data-importing script that I could help with (https://bugzilla.mozilla.org/show_bug.cgi?id=1257954); I've been communicating with him regarding that bug and will be addressing it in the coming days.

In the meantime, a good-first-bug popped up related to Marionette (https://bugzilla.mozilla.org/show_bug.cgi?id=1258003), which involved adding a check to

the external media tests to throw an error if the user has supplied an empty manifest of video URLs to test. I found fixing this bug somewhat challenging, not because of the required change to the code (which was very simple), but because I was previously unfamiliar with the Mercurial version control system and MozReview. Most of the time I spent working on this bug was reading about these two tools and the Mozilla contributor workflow, and corresponding with the project mentor about them via email and Bugzilla. Once I had gotten a grip on these tools, I was able to successfully submit a patch which has been merged into the mozilla-central repository, as can be seen in Bugzilla.

After finishing those contributions, with help and instruction from the project mentor I have been working on generating ideas for new tests for the `BaseMarionetteTestRunner` class, along the lines of the patches attached to this bug: https://bugzilla.mozilla.org/show_bug.cgi?id=1227367. So far I have proposed two patches containing additional tests, which I have placed in Github gists for the time being at the mentor's request. One patch contains a test to make sure that the `_load_testvars` method is throwing the correct errors when it encounters a missing or incorrectly formatted file (https://gist.github.com/vakila/42061355d95d10a92c3a); the other contains two tests which respectively ensure that the `reset_test_stats` method in that class is being called at the appropriate times, and that it functions as expected by resetting the appropriate test statistic properties (https://gist.github.com/vakila/1c4847a4e3ec44226663). I hope to discuss these patches with the project mentor and make any necessary revisions in the coming days (see the timeline below for more details).

Please describe your experience with any other FOSS projects as a user and as a contributor:

I am a daily user of multiple FOSS projects, including Git, Python, Atom (text editor), Zulip (chat client), and Firefox. In the past I have also used other projects such as OpenOffice/LibreOffice, GIMP, and Ubuntu frequently.

As an applicant for the previous round of Outreachy (December 2015-March 2016), I contributed to the HTML Standard (https://html.spec.whatwg.org/multipage/) by submitting patches to resolve two issues on the project's Github repository (https://github.com/whatwg/html). My first contribution involved removing a potentially offensive choice of terminology (https://github.com/whatwg/html/pull/281), a small but important change through which I learned how to build and edit the Standard and how to use git rebase to squash commits. As a second contribution, I addressed an error in the Standard (https://github.com/whatwg/html/issues/253) which was also affecting certain tests for the specification; in addition to filing a pull request (https://github.com/whatwg/html/pull/288) to resolve the specification issue, I also created an issue in the test repository (https://github.com/w3c/web-platform-tests/issues/2280) and filed a pull request to resolve that as well (https://github.com/w3c/web-platform-tests/pull/2283). Following discussion with the project team members/mentors and some corrections, all three patches I submitted were incorporated in these two repositories (see comments on the Pull Requests for details).

While a participant at the Recurse Center, a full-time self-directed programming retreat, I wrote and contributed to open-source software for a period of three months (fall 2015). I had a particularly memorable experience attempting to contribute to Zulip (https://www.zulip.org/), an open-source chat application written in Python/Django and JavaScript. Working with Zulip was my first attempt at diving into a large FOSS codebase and contributing a new feature. The feature involved sending notifications only to those users who were currently logged into the chat client; this related to many steps of the data flow from frontend to backend and back. Attempting to implement this feature thus taught me a great deal about the internals of Zulip; I kept a log of the work I did at https://github.com/vakila/zulip/wiki/Mention-online-users. Ultimately, however, the feature was abandoned; although it had been initially suggested to me by a co-founder and principal contributor of the project, it was eventually determined that it did not provide a significant enhancement over existing features, and my work did not end up getting merged into the codebase. However, I also contributed to the project by filing a Github issue requesting another feature that myself and other users have often expressed a desire for (https://github.com/zulip/zulip/issues/217); I hope to be able to work on that feature (which should be much simpler to implement) at some point in the future. Additionally, I participated in a team which contributed to RSVPBot (https://github.com/kokeshii/RSVPBot/pull/12), an open-source Zulip chat bot in Python which extracts event contexts from chat messages. For our contribution, we used Test-Driven Development and regular expressions to enable RSVPBot to handle more variable text input when processing messages; this was my first experience spending 95% of development time writing tests, and 5% of time making a tiny change to the code to make those tests pass, which made it all worthwhile!

As a master's student in speech technology, I co-created lex4all (http://lex4all.github.io/lex4all/), an open-source graphical Windows desktop app written in C#. Implementing an algorithm for automatic pronunciation discovery, it uses Microsoft's speech recognition engine for a high-resource language (e.g. English) to generate pronunciations for words in a low-resource language (e.g. Yoruba) from a handful of audio samples. Building lex4all was my first experience in creating an open-source project, and gave me a first opportunity to learn about software licensing. Through this project I also became familiar with a new language and development environment (C# and VisualStudio), and gained my first experience publishing and presenting original work at international conferences (see the bottom of the project web page for links to the related publications).

Please describe any relevant projects that you have worked on previously and what knowledge you gained from working on them (include links):

Below are descriptions of a few projects relevant to the internship technologies (Python/Django and JavaScript) and goals (test-driven development and refactoring). A more comprehensive list of projects I have created/contributed to can be found at https://vakila.github.io/projects/.

For a project-based master's course in Software Engineering, I led a three-person team in creating the prototype for Deeva (http://deeva.mmci.uni-saarland.de/), a platform for experimental research on the personality traits associated with 3D virtual human avatars. The system, a web app in Python/Django, collects crowdsourced judgments about avatars and uses these to drive a genetic algorithm to progressively "evolve" avatars to represent various personality traits. In addition to helping design and develop the Django application, my individual contributions included implementing the genetic algorithm, writing and revising project requirements, and using RedMine for project management. This project taught me a great deal about the process of software development, from first conversations with the "client" (an academic researcher in need of a platform for experiments) to final product delivery. It also gave me my first experience collaborating on software as part of a team, and taught me the importance of communicating effectively with team members and of using detailed issues and milestones to track and guide progress. Through this project I also learned how to build and deploy a web app, got my first taste of the Django framework, became familiar with the Model-View-Controller pattern, and learned about genetic algorithms.

While attending the Recurse Center, I created two open-source projects using test-driven development: Kimi, a Python interpreter for a minimal, Lisp-like programming language (https://github.com/vakila/kimi), and Net-Set, a real-time browser game using functional JavaScript on the backend with Node.js and the Mori library (https://github.com/vakila/net-set). Though these were small, experimental projects, through them I learned a lot about test-driven development and libraries for it (unittest for Python and Mocha for JS). Through my work on Net-Set, I learned about functional programming, and how structuring code in a more functional way makes it much easier to test, by breaking down the code into manageable chunks and avoiding global state.

Finally, in graduate school I contributed to JSnoori (http://jsnoori.loria.fr/), a Java speech processing and analysis application. As part of my master's thesis project at Saarland University, I worked with the JSnoori team at our partner institution LORIA (Nancy, France) to refactor parts of the software to make the program DRYer and more modular. This was my first-ever experience contributing to a large existing project. Through it I learned not only how to navigate a new codebase and integrate myself into a development team to make helpful contributions, but also a great deal about how to refactor code to make it easier to read, test, and utilize from external applications.

What project(s) are you interested in (these can be in the same or different organizations)?

I am interested in working with Mozilla. I'm most interested in the project "Test-driven Refactoring of Marionette's Python Test Runner", and have focused on this project during the application process (as described above).

If for some reason I am selected as an intern but this project is no longer available, I would also be interested one of the following related projects:
- Enhancements to Python testing tool plugin for generation of HTML reports
- Convert Mozmill tests to Marionette

As I have also enjoyed contributing to the Treeherder project in the application process (as described above), I would also be interested in continuing to work on that, though I realize this is likely not possible as it is not listed as an official Outreachy project.

Who is a possible mentor for the project you are most interested in?

Maja Frydrychowicz (:maya_zf)

Please describe the details and the timeline of the work you plan to accomplish on the project you are most interested in (discuss these first with the mentor of the project):

The following timeline is a detailed outline of the work I believe I will need to do to prepare for, carry out, and wrap up the internship. The timeline is of course approximate, and is based on my current understanding of the project, and may reflect misconceptions about Marionette, the test runner, or related tools. It will almost certainly need to be amended as I become more familiar with these tools and the project; as noted below, revising the timeline is a goal for the beginning of the internship, and it may need to be further revised as the internship progresses.

During remainder of application period (March 28-April 1):

- Continue learning about Mercurial (branches, merging, editing history, bookmarks, etc.)
- Discuss the new tests I have proposed and written, as described above (https://gist.github.com/vakila/42061355d95d10a92c3a and https://gist.github.com/vakila/1c4847a4e3ec44226663), with the project mentor. Revise them as necessary.
- As a possible extension to the tests proposed for the `reset_test_stats` method in `BaseMarionetteTestRunner` (https://gist.github.com/vakila/1c4847a4e3ec44226663), propose a patch refactoring the method to be more “future-proof” and easily testable if statistics are ever added/removed/changed (e.g. using a dictionary or list of attributes that count as "statistics").
- Contribute a fix for another small bug (https://bugzilla.mozilla.org/show_bug.cgi?id=1257954) to Treeherder/Perfherder, pending discussion with the bug’s mentor, wlach.

Before the internship begins, if selected (April 1-May 22):

- Build fluency with Mercurial and MozReview/Bugzilla workflow by fixing more small bugs (i.e. more work like https://bugzilla.mozilla.org/show_bug.cgi?id=1258003)
- Finish implementing the simple new tests towards Bug 1227367 as described above, if this has not already been done
- Discuss the planned internship timeline (below) with project mentor, and revise as needed.
- Write initial blog post describing the program, the project, and my contributions so far

During the internship:

WEEKS 1-3: Researching the fundamentals, adding simple tests

- Go through Marionette and Test Runner documentation with a fine-toothed comb to cement my understanding of what it is we're trying to do. Get clarification when necessary from project mentor and/or other team members. Keep an eye out for opportunities to add to/improve the documentation, and make these contributions as necessary.

- Read documentation and source code and get clarification from project mentor to solidify understanding of `BaseMarionetteTestRunner` class and related classes/concepts. I will surely discover various "unknown unknowns", but a few "known unknowns" include:
  - The `Marionette` class (e.g. sessions, scripts/scripting interface, etc.)
  - The `MarionetteTextTestRunner` class
  - Other `*TestRunner` classes (discover which ones exist, how they relate)
  - Emulators
  - Test chunks/chunking
  - Test manifests
  - How "mixins" work and what the content of "mixin_run_tests" might be

Keep an eye out for opportunities to refactor the code to make it more modular (and thus more testable) and maintainable, and discuss these with project mentor.

- Continue implementing simple tests toward Bug 1227367 as described above, if not already complete. While exploring BaseMarionetteTestRunner as described in the previous paragraph, brainstorm ideas for additional simple tests, discuss these with the project mentor, and begin implementing them.

- Begin coming to grips with various relevant components of the Mozilla/Firefox (testing) ecosystem (e.g. Marionette, Gecko, Treeherder, mozbase, mozharness, TaskCluster), how they interrelate, and how they relate to the Test Runner. This will involve reading documentation and possibly examining references to other components/libraries in Test Runner source code.

- Read up on testing tools/libraries used (e.g. Unittest, Pytest, Mock) to better understand their capabilities and how they're being utilized. Keep an eye out for any optimizations that could be made/features that could be exploited better.

- Pending discussion with project mentor, determine best way to organize upcoming/ongoing internship work; i.e. moving away from using Bug 1227367 for everything, and determining the appropriate granularity for opening bugs related to this internship (e.g. using fewer more comprehensive work-in-progress bugs a la Bug 1227367; using more, smaller, more quickly addressed bugs and capturing big-picture work in progress another way; or - most likely? - a combination).

- Speak with project mentor to assess progress so far, and revise the goals/timeline for the remainder of the internship.

- Write 2-3 blog posts. Possible topics include:
  - What I've learned about the Mozilla ecosystem
  - Overview of the Python testing tools we're using as I've understood them
  - My initial experience becoming a member of a distributed team

WEEKS 4-7: Digging deeper for test ideas, examining interfaces

- Continue adding tests that have been brainstormed/discussed in previous weeks, using the new organizational flow that has been established with project mentor. Continue refactoring BaseMarionetteTestRunner based on the discussion with project mentor as mentioned above.

- Determine how the Test Runner has been/might be used by other Mozilla teams. Track down and examine any source code which has been written to customize it, looking for ideas for tests and possible improvements. If possible, identify a few Mozilla developers who have or might have need for the Test Runner, and chat with them about it. Discuss findings and ideas with project mentor, to determine which additional tests would be most useful and/or how the tool could be improved.

- Solidify understanding of different options for running tests (e.g. TaskCluster, mach), how they relate, and what their interfaces are. Look for opportunities to improve/elaborate the documentation of these interfaces, to help others get up to speed with them more quickly in the future. Determine whether improvements to these interfaces are necessary/possible, and if so, begin making those changes. Create additional tests as necessary to make sure that these interfaces are behaving as predicted, and possibly to account for any likely mis-uses on the user's side.

- Speak with project mentor to assess progress so far, and revise the goals/timeline for the remainder of the internship.

- Write 2-3 blog posts (topics TBD based on activity/contributions during this period).

WEEKS 8-11: Following through, improving logging

- Continue adding tests and refactoring code based on findings from the previous weeks.

- Take a closer look at the logging functionality. Pending discussion with project mentor/team members, improve the content/structure of logs to make digesting the most important information easier. This should apply to logs printed to the console and, ideally, also the logs as they are displayed in Treeherder's log viewer. Create tests if necessary to ensure that logging is proceeding as expected.

- Speak with project mentor to assess progress so far, and determine any remaining tasks that must be completed by the internship's end.

- Write 2-4 blog posts (topics TBD based on activity/contributions during this period).

WEEKS 12-13: Wrapping up

- Finish adding tests/making changes that have been identified in the previous weeks.

- Document (e.g. in Bugzilla bugs, or wherever appropriate) any work that is in-progress or ideas that have not yet been implemented, so that myself or other contributors can easily continue with it after the internship ends.

- Speak with project mentor to assess progress in the final weeks and overall.

- Write at least 2 blog posts:
  - A summary of what I have done and the state in which I'm leaving the project
  - Reflections on the Outreachy program and working with Mozilla/the A-Team

Will you have any other time commitments, such as school work, exams, research, another job, planned vacation, etc., between May 23 and August 23, 2016? Please provide exact dates for these commitments and the number of hours a week these commitments take.

I will be giving two talks (on Python bytecode and double-underscore methods/attributes) at the EuroPython conference (https://ep2016.europython.eu/en/), July 17-24. I will therefore not be able to work during that week (or at most contribute a few hours' work), and would propose making up the lost hours in the preceding/following weeks. Other than this obligation, I am available full-time (40 hours/week) for the internship period.

If a student, please list the courses you will be taking between May 23 and August 23, 2016, how many credits you will be taking, and how many credits a full-time student normally takes at your school. Please provide a link or upload your school's academic calendar.

N/A (not a student)