SIFT: Submitty Instant Feedback and Testing

Abedalah Safi, Abrar Zaki, Ayaan Ahmad, Joshua Javillo

Introduction

SIFT or Submitty Instant Feedback and Testing is a standalone app that enables students to autograde their Submitty coding homework assignments without having to rely on Submitty. Oftentimes, working with the Submitty autograder to test and find problems in code can be cumbersome, requiring the user to upload all their files, wait for the servers to process and run them, and then getting your output, all while using up a precious submission. SIFT aims to fix this by giving students the tools to get their homeworks graded on their own computers. Users begin by selecting the language they are using, upload their own source code files, and the expected output files. SIFT then runs the source code and shows the user the comparison between their output and the expected output that was provided in a similar way to Submitty. The goal of SIFT is to streamline the process of creating and testing code so that students spend less time waiting for Submitty, wasting submissions, and can instead get quick results and keep working

Objectives

- Provide students with a resource to quickly and easily test code anywhere, anytime
- Create a user friendly and functional UI
- Learn how to run code written in different languages within Python
- Encourage students to develop better testing habits
- Relieve stress off Submitty servers
- Apply the skills we've learned to create a useful product

Materials and Methods

One of our main goals with S.I.F.T was to make an easy-to-run, smooth application. We decided to run with python because of its extensive library support. Moreover, the whole team has experience in python, so it would make the development process smoother. Our front-end team, composed of Abrar Zaki and Joshua Javillo, researched for ways to create a simple, easy-to-use UI. First we researched **Tkinter** as that is a popular way to make GUIs in python. However, the team found a library built upon **Tkinter**, called **Custom Tkinter** that makes it easier to create UI components. Since we want to be able to run our application across all platforms(Windows, Mac, Linux), we decided on using **Docker** to distribute S.I.F.T.

Frontend/UI

Before anything working on any code, we made a mock up of how we wanted to structure our project. We made our mock ups in figma because it was easy to use and it allowed us to easily collaborate with each other. Once we decided and agreed on our front end design, we had to choose a language and a library to make our frontend with. As stated before in the materials section we decided to make our front end user interface with python and the custom TKinter library. We decided to go with the custom TKinter because we thought that it wasn't too difficult to understand with our group's previous experience in TKinter, also we thought it had some very nice looking widgets that we wanted to use such as buttons, labels, and option menus.

Our design philosophy when making our frontend was that we wanted to emphasize the discoverability of our application. Which means that we want everything to be very straightforward and self explanatory for the user. The buttons that lead to other pages are very clearly labeled with what to expect.

Something that we wanted to implement into our application was the ability to change themes. In the "Settings" frame we allow the user to change the color of the theme that they want the user interface to have. When a new color is selected, the app restarts and takes you back to the first frame with the theme now the color that you chose. Currently there are 5 color themes to choose from, dark, light, red, green and pink.

Conclusion