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. As such:

- 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.

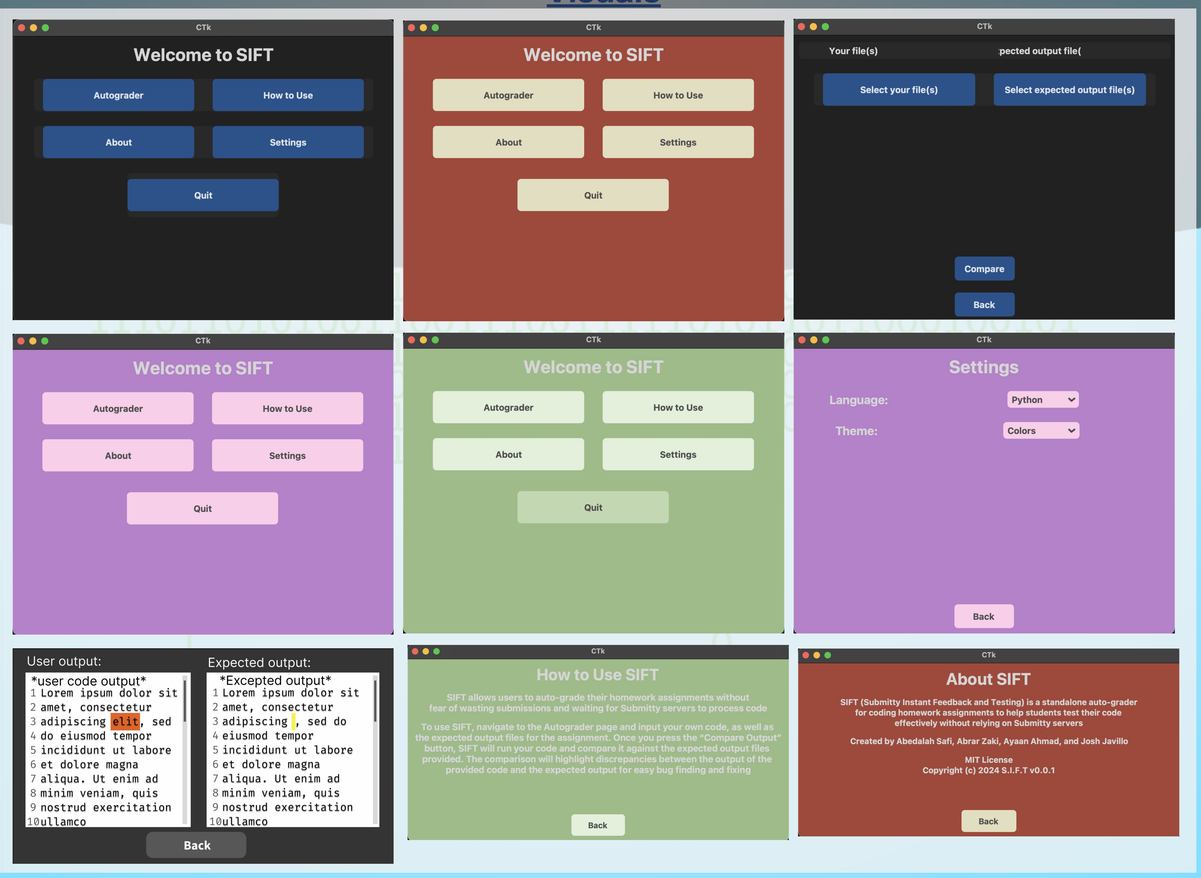
Frontend/UI

- We made our mock ups in figma
  - it was easy to use and it allowed us to easily collaborate with each other.
- 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.
  - The library 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.
- we implemented into 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.
  - 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.

Accomplishments

- We started S.I.F.T from scratch, making our UI over the semester
  - We also implements different color option for our UI
- In our front end, the user can select what files we they want to run, through their native OS' file system
- We created a functionable user interface using CustomTkinter, a UI library build on top Tkinter for python
- We created a backend that runs programs in C++, other python programs, and java
  - This is done through the python subprocess system
- The way we construct paths for our files is OS agnostic, meaning our paths should work on any Operating System
- We joined our backend with our frontend
  - At the moment the programs we run through our print their output to the console

Visuals



Future Goals

Some of the ideas and goals we have for the future include:

- Allowing users to select individual courses and have all the necessary resources already available for them without extra downloads
- Allowing professors to upload materials and manage their courses on the app to allow the features mentioned above
- Incorporate AI for giving users feedback and tips on how to improve their code/coding habits
- Possibly creating a better UI from scratch for more control and more flexible aesthetics
- Make the app distributable (through platforms such as Docker)

Acknowledgements

- CustomTkInter by Tom Schimansky
- TkInter by Steen Lumholt and Guido van Rossum
- Python by Guido van Rossum
- RCOS professors, coordinators, and mentors