**Project Name**

SteelPass

**Project Description**

A password manager application that uses the CMU Graphics library. It provides input interfaces to the user to enter login credentials and give the user the option to generate complex passwords. It then stores the entries in an encrypted database to keep the user’s credentials secure. It also has extra security features that will be elaborated on after the TP2 stage.

**Competitive Analysis**

The popular majority of the password managers available online store the credentials online for easier access. Examples include solutions like NordPass, 1Password, and Dashlane. However, that is not always the best security practice since it leaves the user vulnerable to online breaches to their master password manager’s credentials or breaches the data centers themselves that store the passwords for all users. Instead, my project gets its inspiration from software like KeePassXC, a both free and open-source password manager. As opposed to the previously mentioned solutions, KeePassXC stores the passwords locally on the system running it. My project will be similar to KeePassXC in that regard and will also try to include extra features.

**Structural Plan**

The final result will consist of a GUI implementation that lets the user input as many login credentials as they wish and then take those credentials, encrypt them, and save them to a local database file. Later, when the user accesses the application and enters their master decryption key(s), the application is going to list the saved entries and allow them to add more entries. For the GUI design part, the project will use Object Oriented Programming to define elements like text boxes and buttons that will encapsulate many necessary processing functions too. Regarding files, the project will most probably end up using only two base files, the program itself and the database, plus extra text files that may include dictionary words to generate passwords.

**Algorithmic Plan**

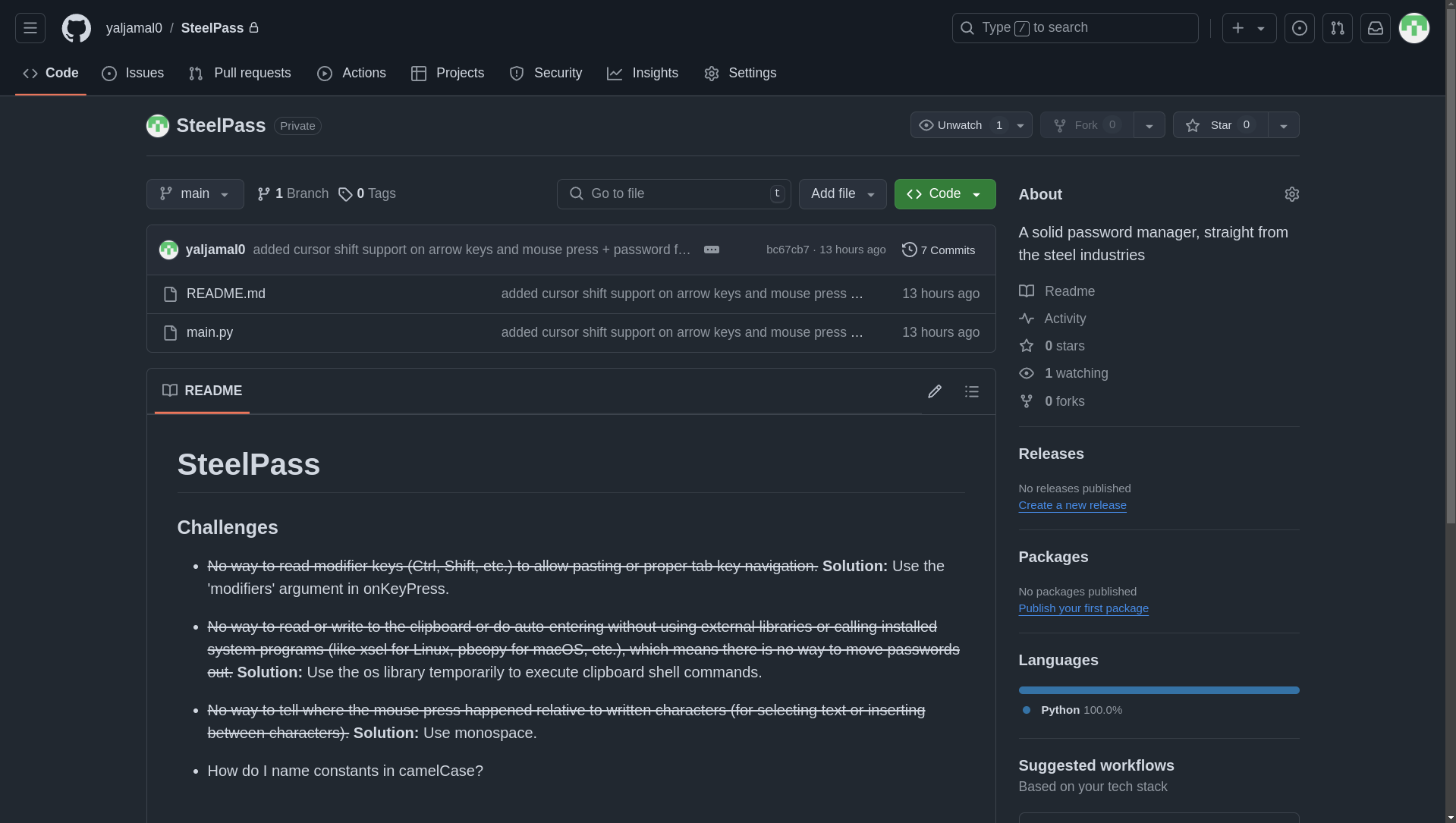
The trickiest parts will be to design properly functioning elements like text boxes and forms and to protect the encryption/decryption process by applying multiple time-based security measures. For the GUI elements, I will use OOP to make reusing and managing each element easier. For the security measures, I am planning to encrypt the unlocked database whenever the user locks the screen or leaves the application unused for a long time to protect against reading the decrypted credentials directly from the RAM when the user is not around. I will also limit the time the copied passwords stay in the clipboard to prevent accidental pasting of passwords in the wrong places and to prevent attackers from reading passwords off the clipboard.

**Timeline Plan**

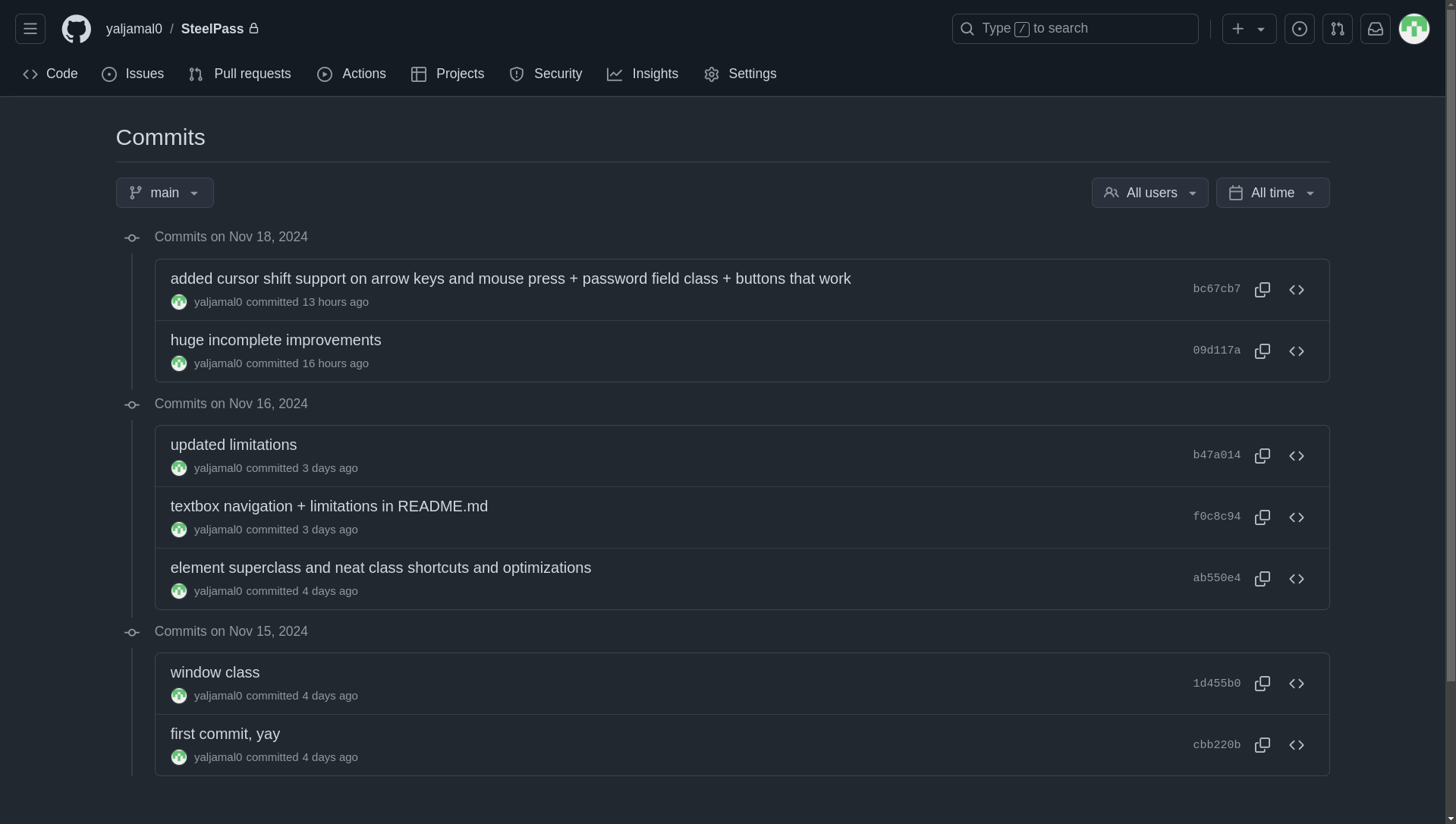
* By TP1: Design basic text boxes and buttons that work and save filled entries to a database file.
* By TP2: Integrate different text boxes and buttons into a form that lets users enter credentials and generate complex passwords. Also, have a separate entries view that shows the user all of their saved entries.
* By TP3: Use strong encryption algorithms and time-based security measures to protect both locally stored and copied credentials.

**Version Control Plan**

I am using the git version control system and periodically pushing code changes to GitHub. That will keep my code saved somewhere else in case I lose local access to it and help me to keep track of changes without keeping them stored in the main project file as comments or dead code.



*My private repository of my term project on GitHub.*



*The commits I have made so far. Some commits include major changes.*

**Module List**

Alongside the CMU Graphics library and some other built-in Python libraries, I am planning to use external encryption libraries like ‘cryptography’ to encrypt/decrypt login credentials. I will also use the ‘clipboard’ library to copy/paste credentials from and to the application. Finally, I might use keyboard libraries to perform auto-writing. All will happen only after TP2.

**TP2 Update**

I have not made any changes to the structure of my program.