Ain Shams University

Faculty of Engineering

REAL-TIME TEXT EDITOR

Final Project Report

Team 28 Members

Adel Asaad Sarofeim	18P2949
Marwan Atef Hamed	18P8678
Moataz Alaa Hamed	19P6238
Mohamed Adel Lotfy	18P1724

Supervisor

Dr. Ayman Bahaa

Eng. Mostafa Ashraf

Table of Contents

Table of Figures

Introduction

A real-time collaborative text editor is a great project example for distributed computing as it was very challenging to figure out the best distributed approach in order to handle these types of projects that with careful development can produce amazing applications like visual studio code!

Real-time collaborative text editors are very helpful and becoming more popular nowadays as they help reducing a huge amount of time sending texts and ask for edits from several people.

This application allows several clients to edit a selected file in the same time, where users can see other users as they edit the file word by word.

In this report we are going to be talking about our very own real-time collaborative text-editor, describing almost everything about it with some diagrams to better visualize the working process of the project.

Project Description

In order to properly describe our project, we first have to mention its three main parts:

- Clients
- Super server (SS)
- Bunch of child servers (CS)

When a user opens the application (client.py) this means that a new client connection has been established with our super server.

The client has two threads working independently on each other:

1. connThread which starts as soon as the client starts, this thread connects the client to the super server.
2. Thread which starts upon connection with a child server, this thread receives updates from the server.

Upon client connection to the super server, the SS checks for working (online) children and sends the IP address and PORT number of the first available server found back to the client.

The client disconnects from the SS and connects to the given CS, the CS keeps handling the client by either creating a new file, send a file to the client, and handle files modifications by the client.

If a client doesn't receive a reply from the server for 15 seconds, the connection is closed and clients is rerouted to another server by the SS.

Each CS has two threads (two threads for each client connected):

1. Pong which starts as soon as a client is connected to that server, this thread keeps sending replies to the client.
2. Thread which starts as soon as a client is connected to that server, handles client's modifications to the shared files.

All of the child servers have shared resources (files to be modified) between them. Therefore, to handle the synchronization the following process is implemented:

1. Client makes modification to the version of the resource at a server (CS1 for example).
2. CS1 checks if the SS has the same text stored.
    a. If the SS has the text, ignore publishing.
    b. If the SS has different text, send the new text to the SS.
3. SS sends the new text to all other servers to store.

Beneficiaries of the Project

- Students
- Researchers
- Team members
- Communities
- Groups (Reddit channels, Facebook groups, etc.)
- Any group of people that need an editable shared document

Detailed Analysis

In order to achieve full distribution, clients should not sense the failure and crashes of the server. Therefore, we are going to describe the full working description of our project in these following steps:

1. Client opens the application (client.py).
2. Client presses open button to choose a file to modify, or create button to create a new one.
3. Client is automatically connected to the super server.
4. Super server checks for active servers.
5. Super server returns the IP and Port of first active server found to the client.
6. Client closes connection with the super server and connects to that server.
7. Server handles the client (takes modifications and adds it to the shared resource).
8. Client is always listening to the server.
    a. If server doesn't respond in 15 seconds, connection is closed and we go back to step 2.
    b. If server responds then continue.
9. Client can press the save button to save modification permanently to the server.
    a. This is made as a feature so that if after the client's session if he is not satisfied with what he had done he could just close without saving and nothing will be saved.
    b. Know that if any one of the other clients pressed save on his end these changes will still be saved.
10. Upon pressing open the child server takes the files from the super server and stores a version of it.
11. Upon client's modification to any file at any child server, this server sends the text update to the super server.
12. The super server takes this update and updates the file.
13. The super server then sends the updates to the other servers to update the text version that they have.

Tasks Breakdown Structure

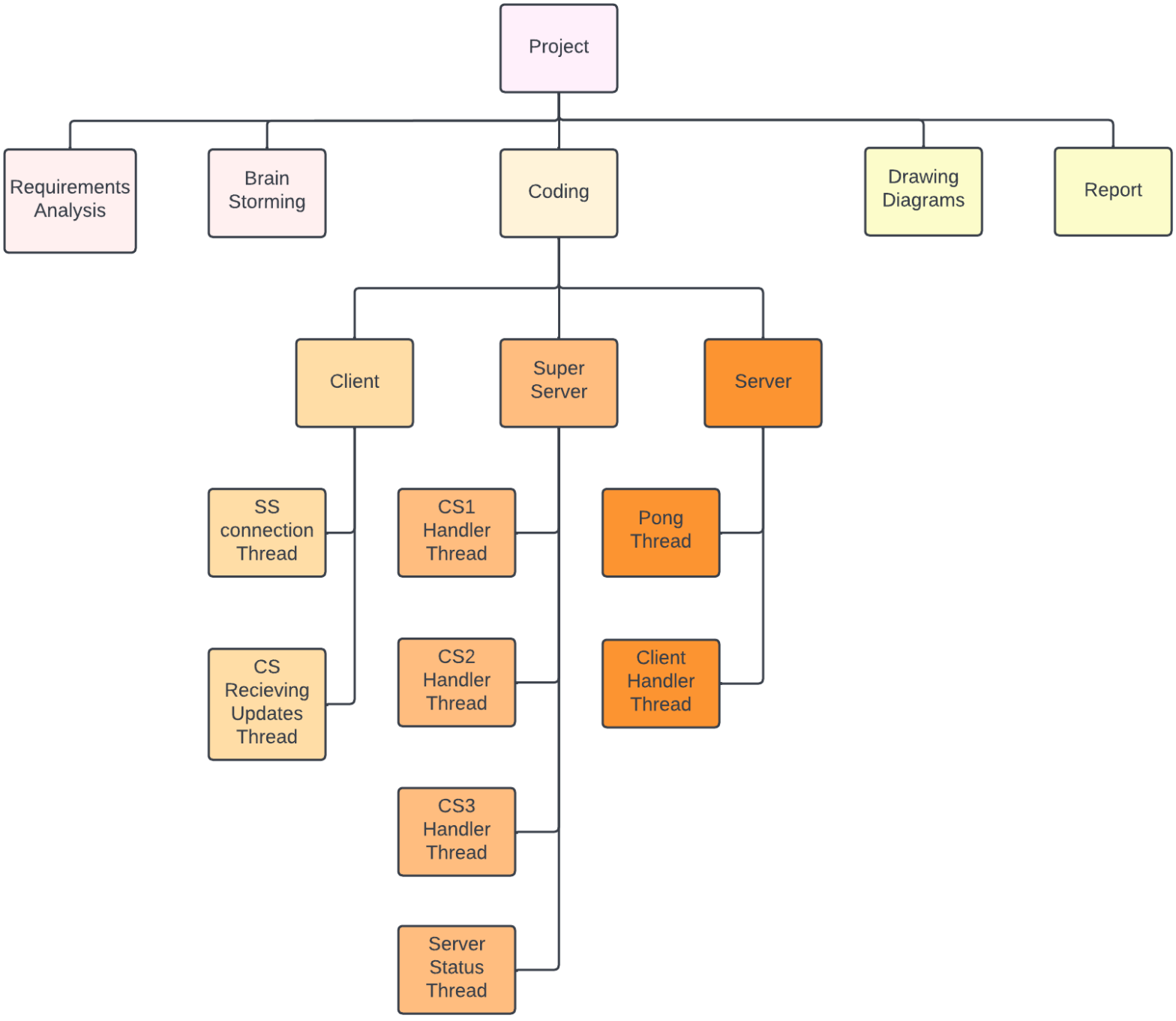

Figure 1: Tasks Breakdown Diagram

Roles of Members

Adel Asaad: Created all of the super server, as well as the handle_client function of the server.

Marwan Atef: Created the initConnSup, recievingUpdates and on_closing functions of the client, as well as handle_client function of the server

Moataz Alaa: Created the onModification, applyDiff and update functions of the client, as well as the handle_client function of the server

Mohamed Adel: GUI modifications, all the diagrams and the report, as well as helping in some functions of the super server.

All students have participated in the brainstorming process and the implementation of most of the functions as we worked in pairs as a driver-navigator style in order to ensure a fault free code.

System Architecture and Design

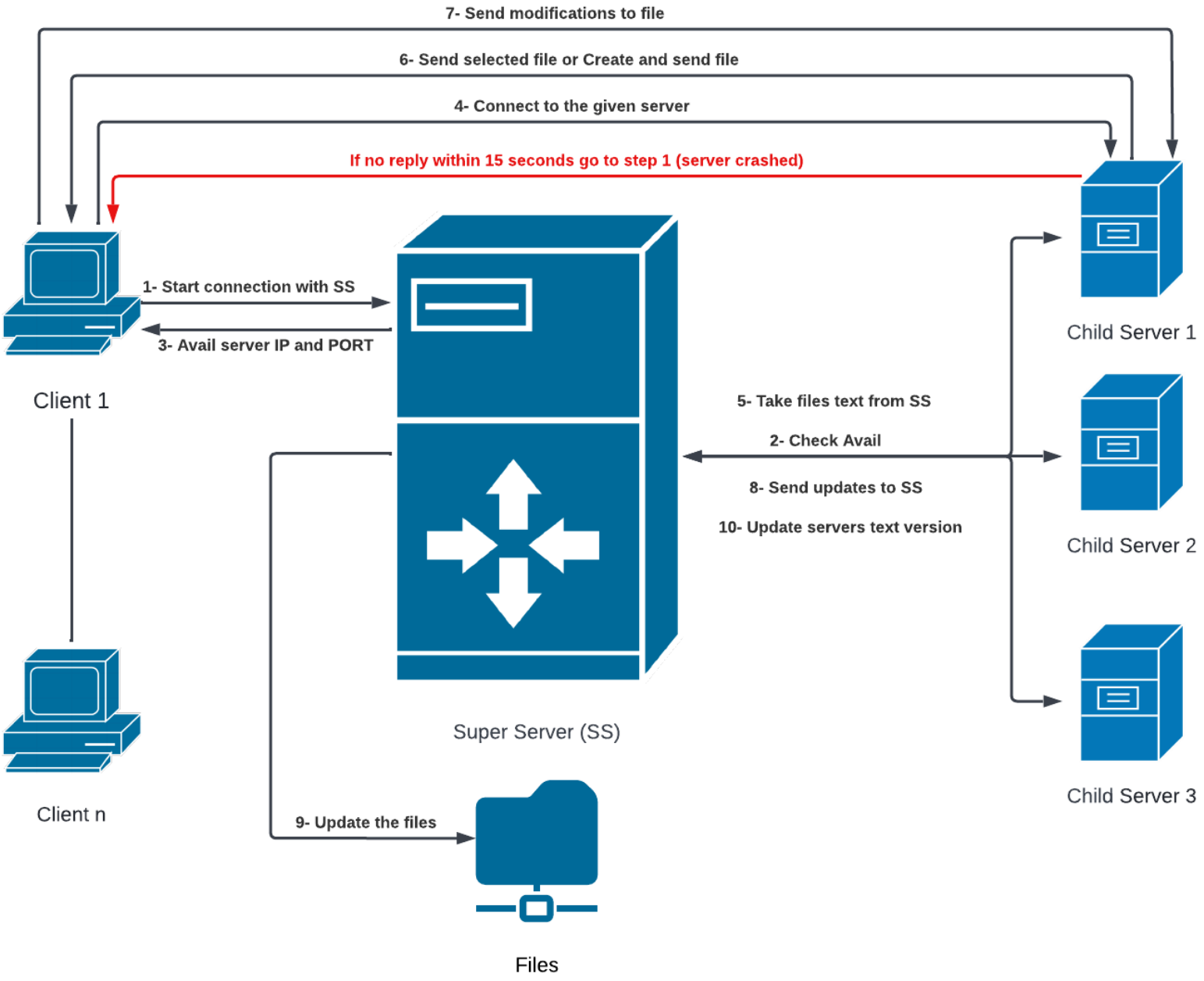

Figure 2: System Architecture & Design

Testing Scenarios and Results

End-User Guide

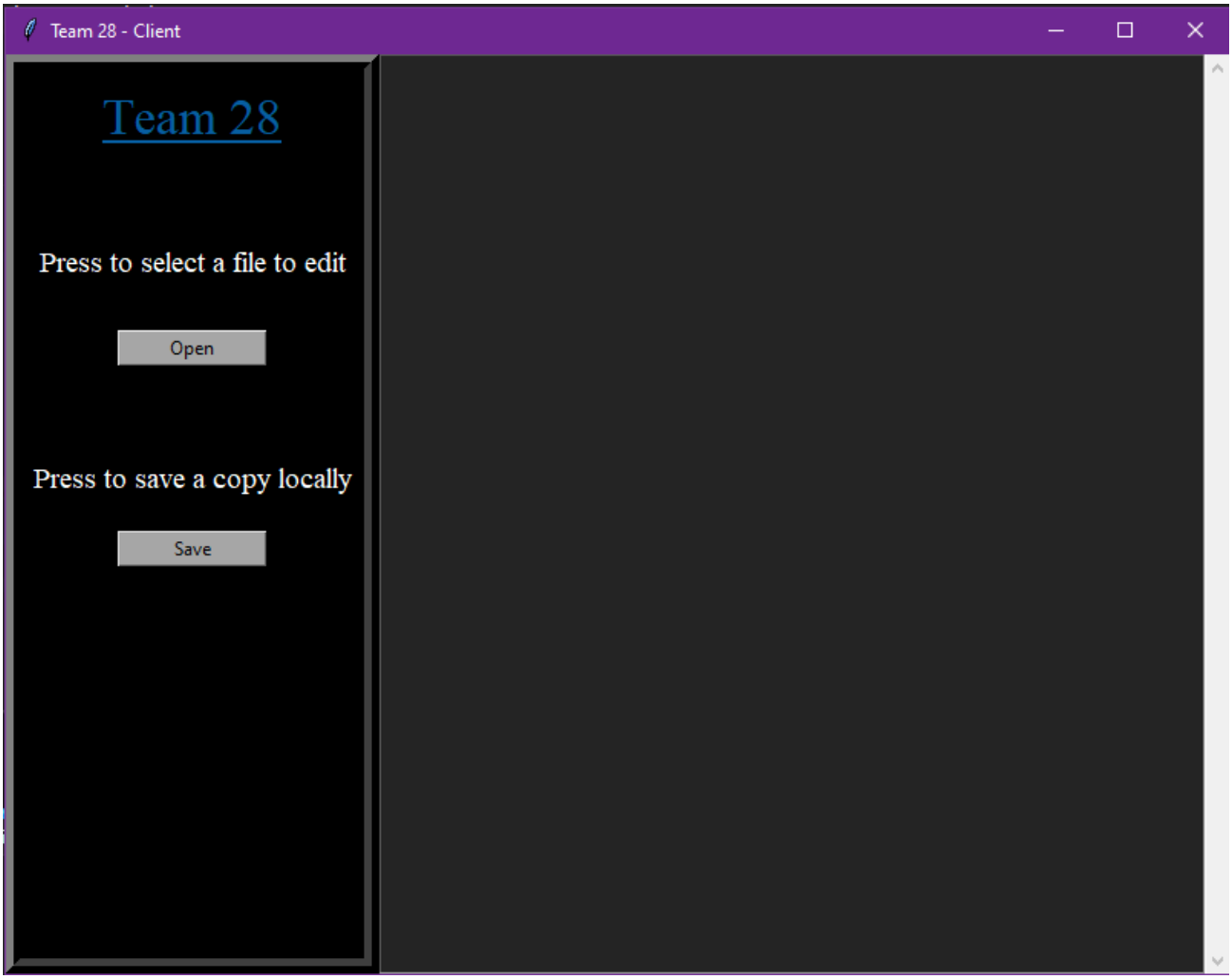

Figure 3: GUI Main Window

Upon pressing open a new window will be opened for him to select a file to be modified.

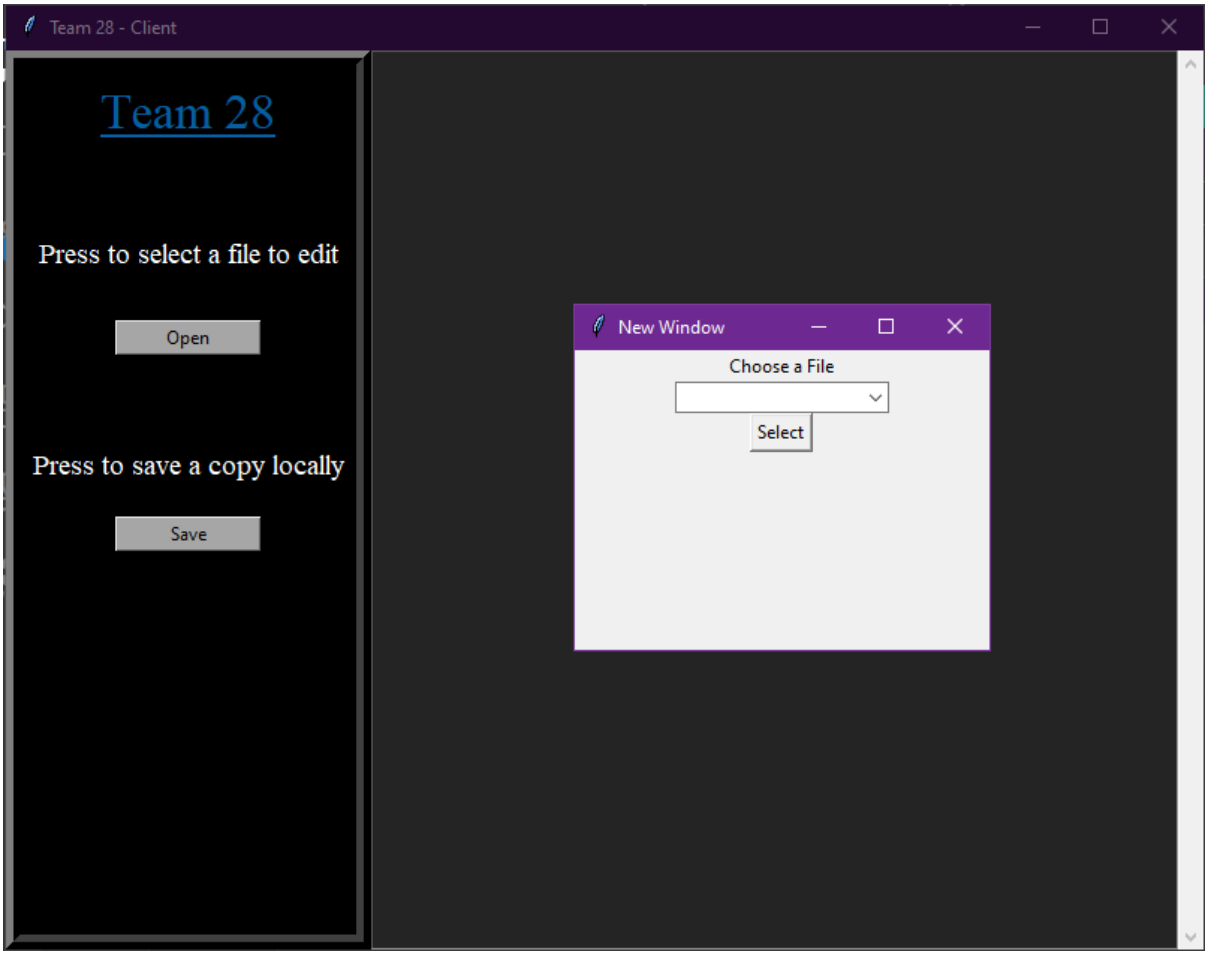

Figure 4: GUI Open Pressed

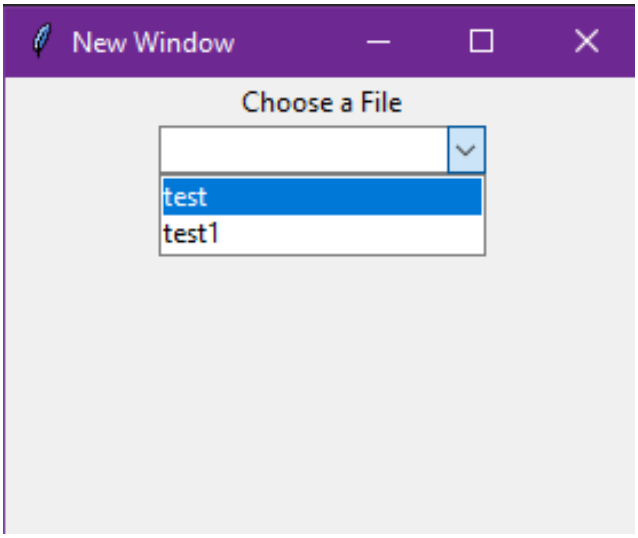

Figure 5: GUI File Selection

Upon choosing a file the user can start typing without waiting.

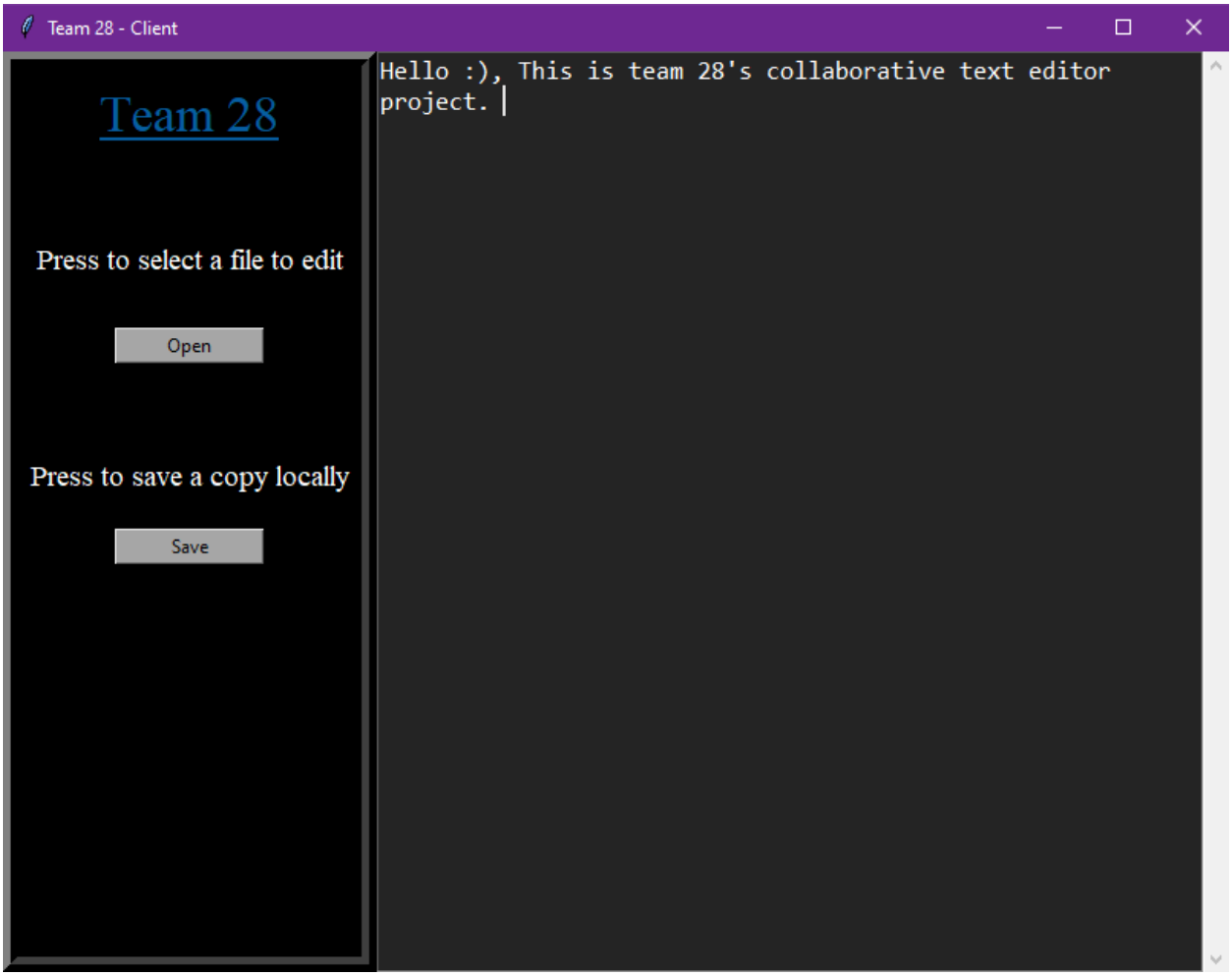

Figure 6: GUI User is Typing Example

The save button closes the file after saving the written text permanently to the file.

Links

Our Github link can be found here

Our Project files and code can be found here

Our Youtube video link can be found here

Conclusion

In conclusion, our project intends to reduce communications delay by allowing users to interact with the document of their choice in real-time which is very helpful!

This project was very stimulating, pleasurable and educational as we had the chance to implement what we have learned during the distributed computing course on a real-life project that is actually very much needed nowadays.

We wished to add lots of features to this exciting project but we did as much as we could given the time constraints, building a collaborative text-editor from the scratch takes a huge amount of time in order to be considered a formidable one (it took google 7 months to build the first version of google docs).

We hope that you find our project to your liking and that you find it as helpful as we did.