Fastchat : Chmod+x

Ayush Agarwal, Sankalan Baidya, Soham Joshi

IIT Bombay

August 2022

Table of Contents

Introduction

Hello, this project, named “FastChat” is a course project of the course CS251 offered in the autumn semester at IIT Bombay. We have designed a working chat application complete with a GUI and a backend. It is complete with the ability to join different groups, send images, and is compatible with the three major platforms of modern computers : MacOS, Windows as well as Linux (Debian). So, without further ado let’s dive in !

GUI

For the purposes of GUI, we have used the tkinter module in python 3.8. Tkinter facilitates a user interface complete with buttons for creating and joining groups, sending image files stored on your machine, switching between different groups. Moreover, the tkinter interface is complete with a chatbox for viewing messages and a chatwindow for sending messages.

GUI : Implementation

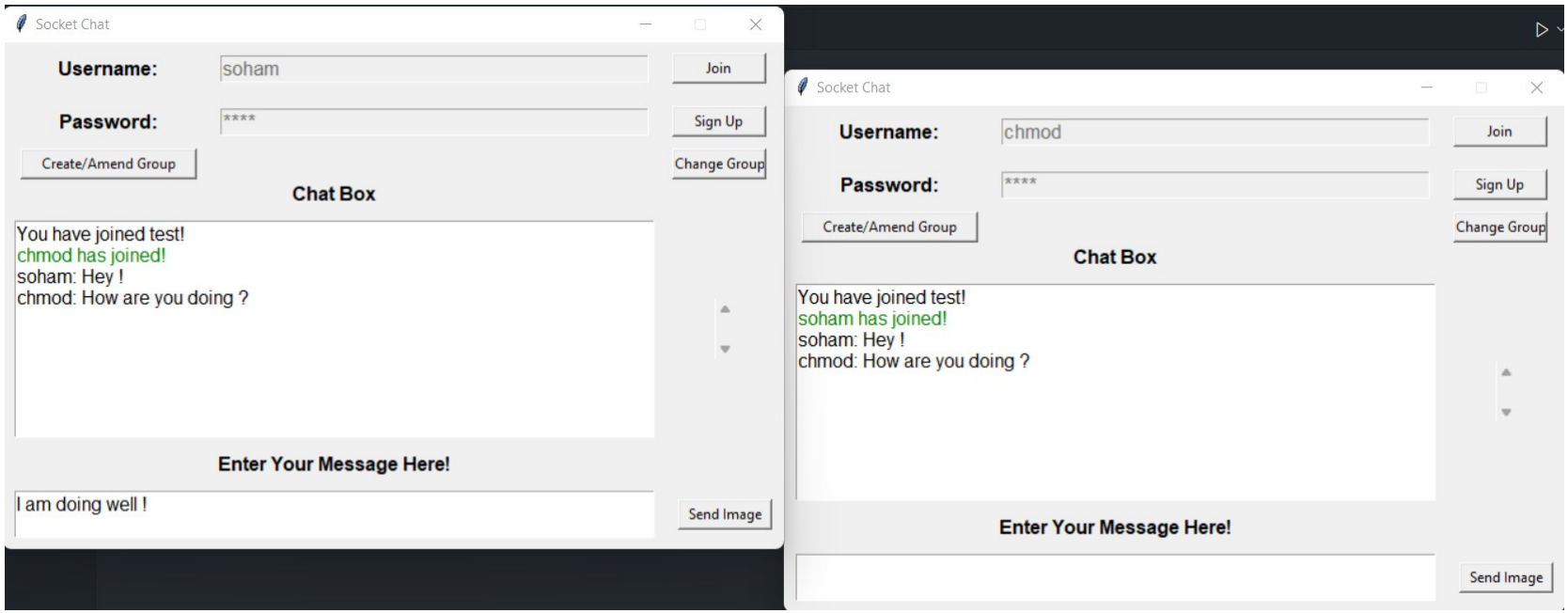

Figure: GUI Overview

Databasing

For the purposes of networking and storing information pertaining to the chat application, we have used PostGreSQL. PostGreSQL is useful especially for dealing with efficient handling of concurrency and having a "delocalised" database in some sense. We have prepared a database schema comprising of four tables.

Schema

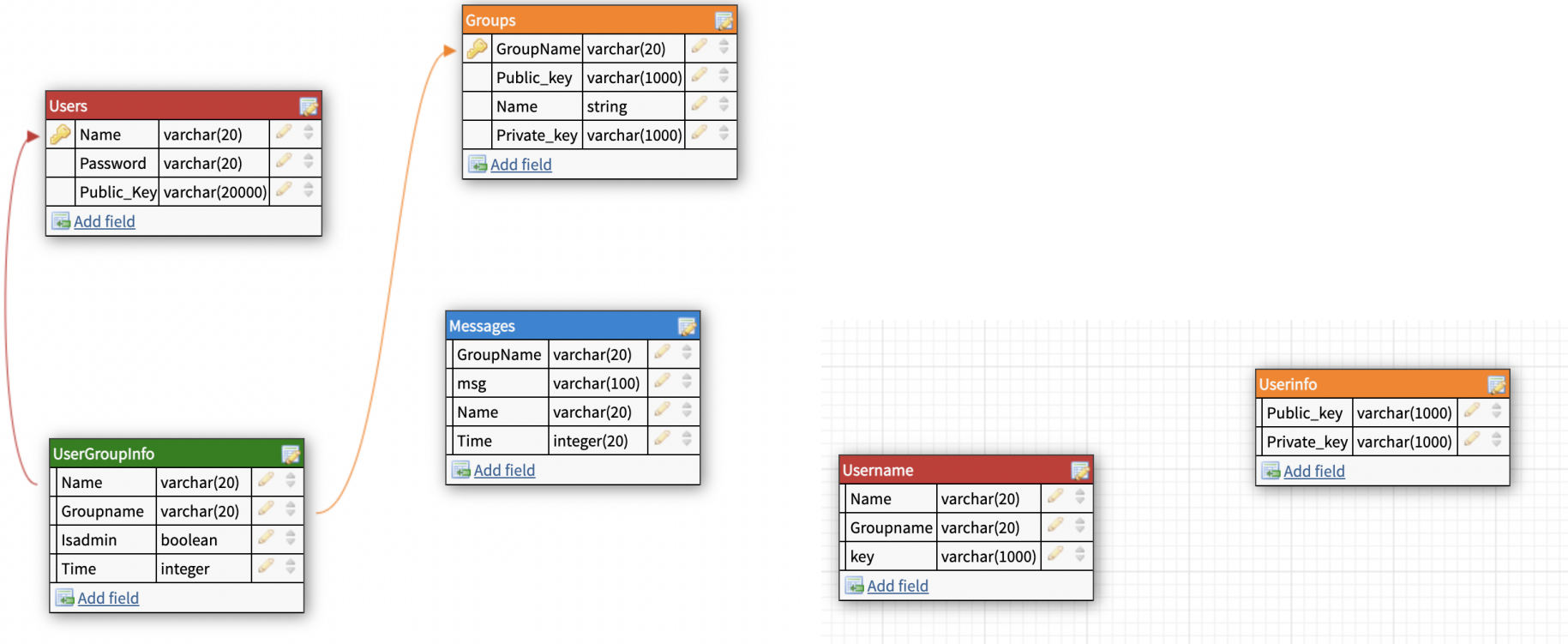

Figure: Database Schema at Server and Client

Login And Signup

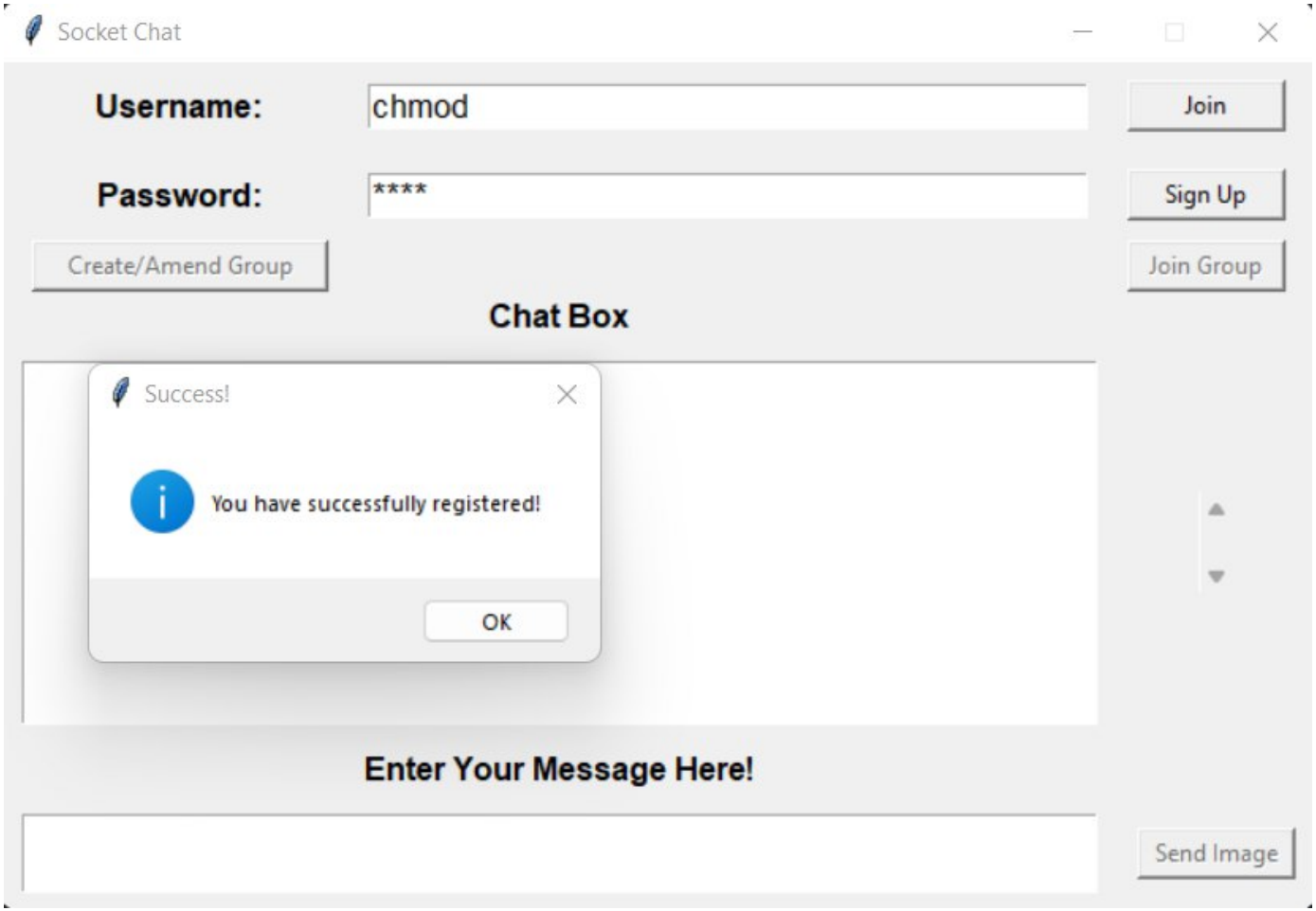

Figure: User registration

Create and Amend group

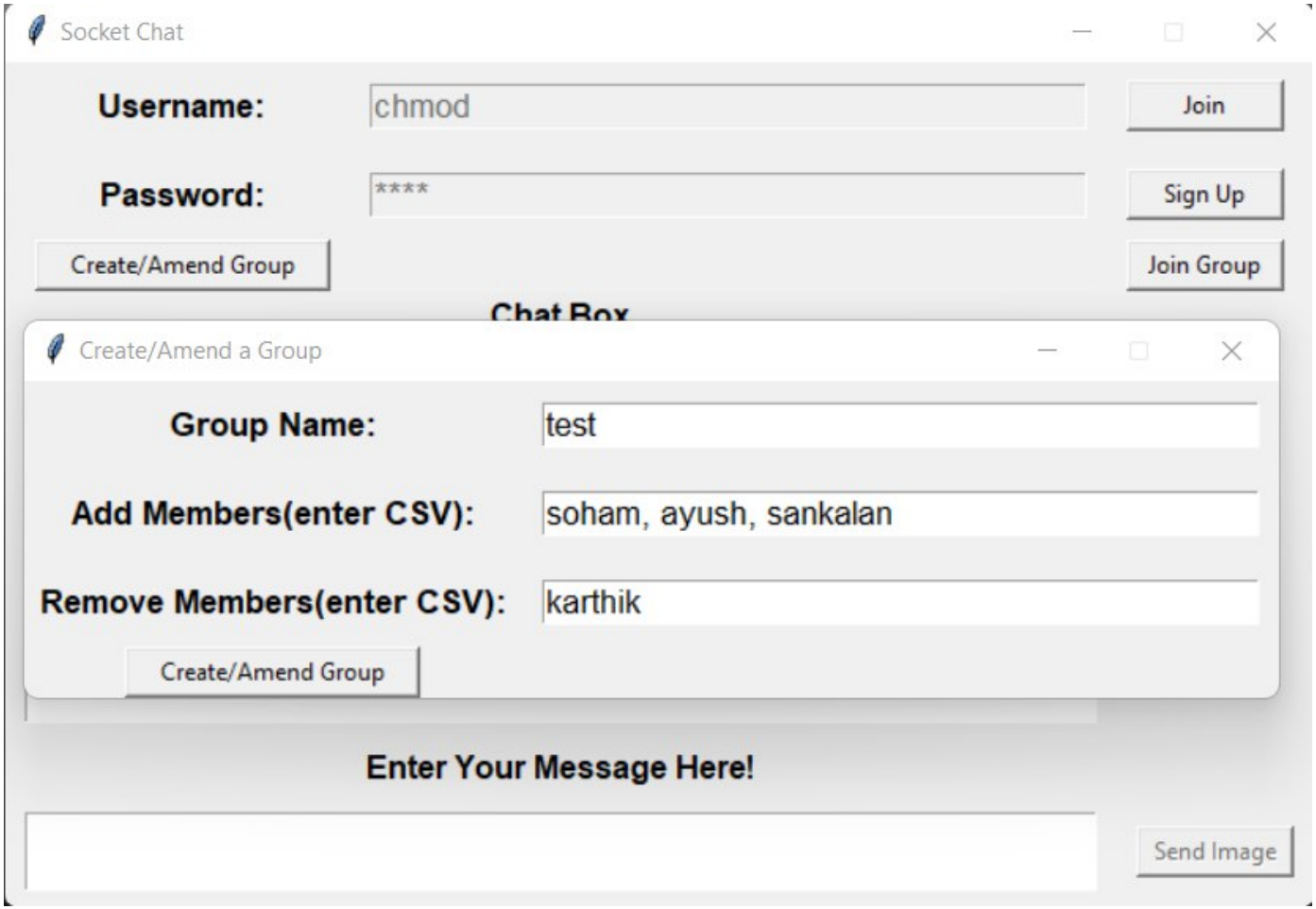

Figure: Group Creation

End-to-end encryption

We are handling end-to-end encryption via a combination of RSA and AES ciphers. Essentially, first every participant generates their own RSA public and private keys. Now, the fernet (AES) key is generated by the group creator and is encrypted using the public keys of the group participants. Now this encrypted message is sent to each of the group participants and they obtain the AES key by decrypting the message using their private key. Now that the keys are established, all messages of a group are symmetrically encrypted and decrypted using the fernet keys. Hence, we achieve semantic security.

Images interface

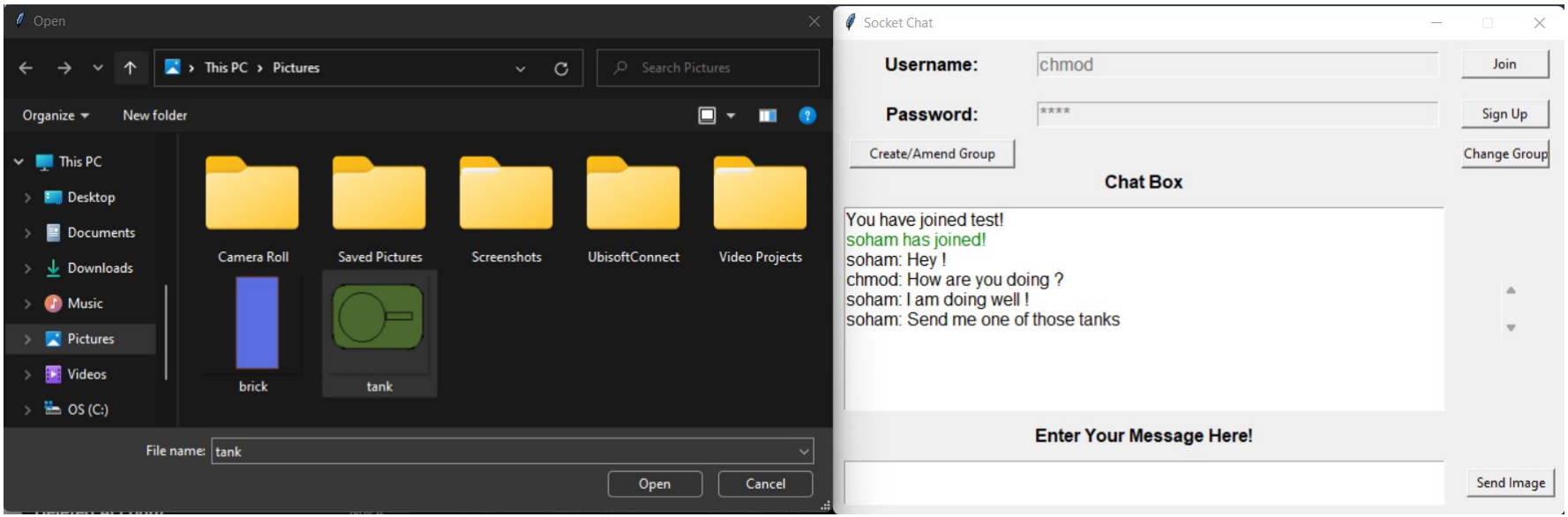

Figure: Images

Mechanism

The mechanism for sending images essentially remains the same as that of messages, with the addition of a few libraries. For the purpose of sending images. we first obtain the file path of the image to be sent using a graphic interface using tkinter. Now that the file path is obtained, we simply convert the images into bytes by reading the file and then reconstruct the image at the destination using the Bytes library.

Load Balance and Strategy

We have implemented a rudimentary version of a multi-server and multi-client set up. For the purposes of optimising the multi-server setup and obtaining the highest throughput possible, we used a load balancing scheme based upon the round-robin scheme.
We have implemented a modified round-robin scheme, where we have switched the servers serially for each socket, hence, obtaining the same effect as a round robin scheme.