Group 18: Thy Do, Brian Moore

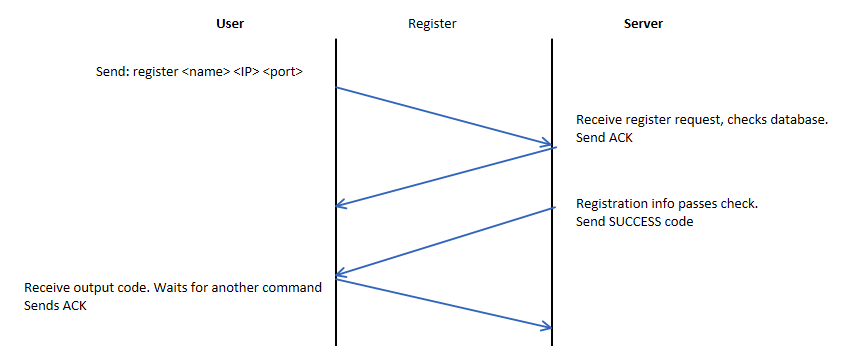
CSE 434

Dr. Syrotiuk

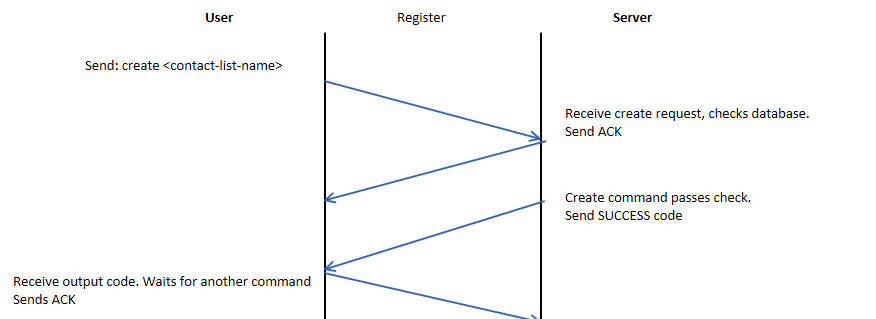
Socket Project Milestone Design Document

**Design**

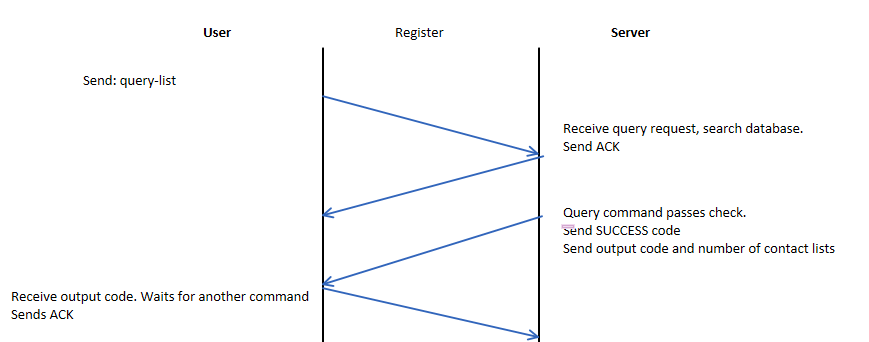
* This socket project is written in Python, currently compatible with python (not python3).
* First, the server is started up, printing its own IP address and waits for a client to send requests. As the client connects to the server, a welcome message is shown along the server IP address, and the user is prompted to enter a command. The server parses client’s message for the command keyword first, then passes it to the proper function.
* There are 2 global databases: a dictionary for each registered user, and a dictionary for all created lists with joined users. Dictionary of registered user is a nested dictionary; each name entry has its own dictionary of IP address and port number. This allows easy implementation, as usernames are keywords to be looked up in the dictionary, we can also query either for the user’s IP address or port number separately. The contact list is a triply nested dictionary. Each entry is the contact list name, they contain entries of joined user, defined similarly to the registered database. We chose this method so data can be looked up using keywords, allowing easy implementation. The databases are accessed using built-in algorithms and methods.
* The server currently supports 20 users and 10 contact lists.
* Clients are sent feedback messages reflecting the outcome of their request. Server end also has internal print messages, showing client’s IP addresses to differentiate sources of requests. Other internal messages are printed to reflect results of different functions to maintain developer’s sanity as we build this program.
* For the milestone, the following commands are implemented:
  + Register: The command format is register <contact-name> <IP-address> <port>. The command will result in FAILURE under these following circumstances: insufficient parameter (it will also send back usage instruction), registered name already exists, invalid IP address, invalid port number, or maximum number of registered users already met. A short explanation follows. SUCCESS is sent back if user’s information is added to the database successfully.



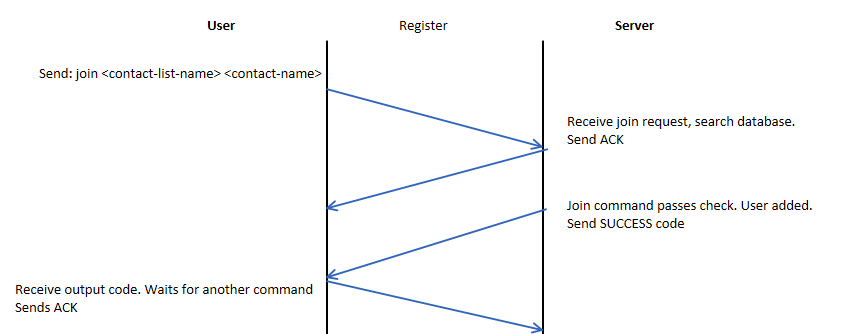
* + Create: Command format is create <contact-list-name>. The command sends back FAILURE if: insufficient parameter entered (sends usage instruction), list name already exists, or exceeded number of existing lists. Sends SUCCESS if list is successfully added to database.



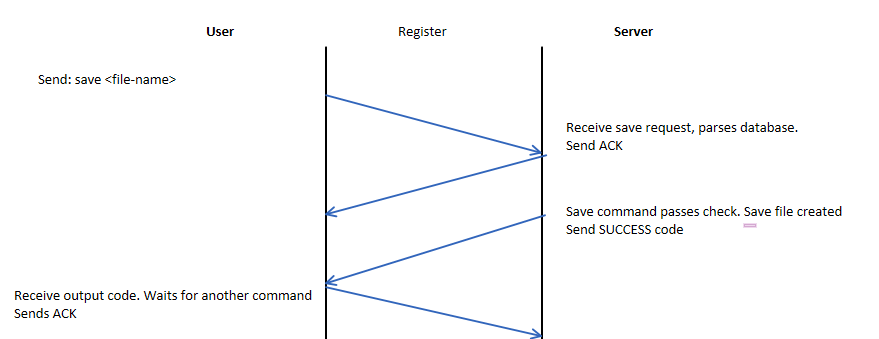
* + Query-lists: Command is query-lists. Sends back number of contact lists along with their names. If no list exits, 0 is sent.



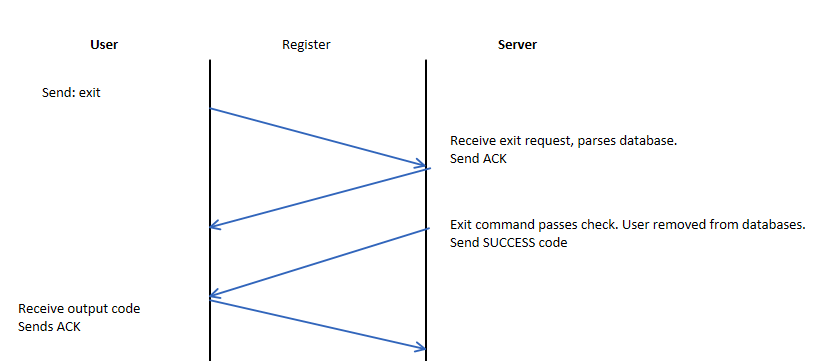
* + Join: Command format is join <contact-list-name> <contact-name>. FAILURE is sent if: contact list does not exist, contact name does not exists in registered database, or contact name already exists in the requested list, or insufficient parameter sent for this command. Otherwise SUCCESS is sent.



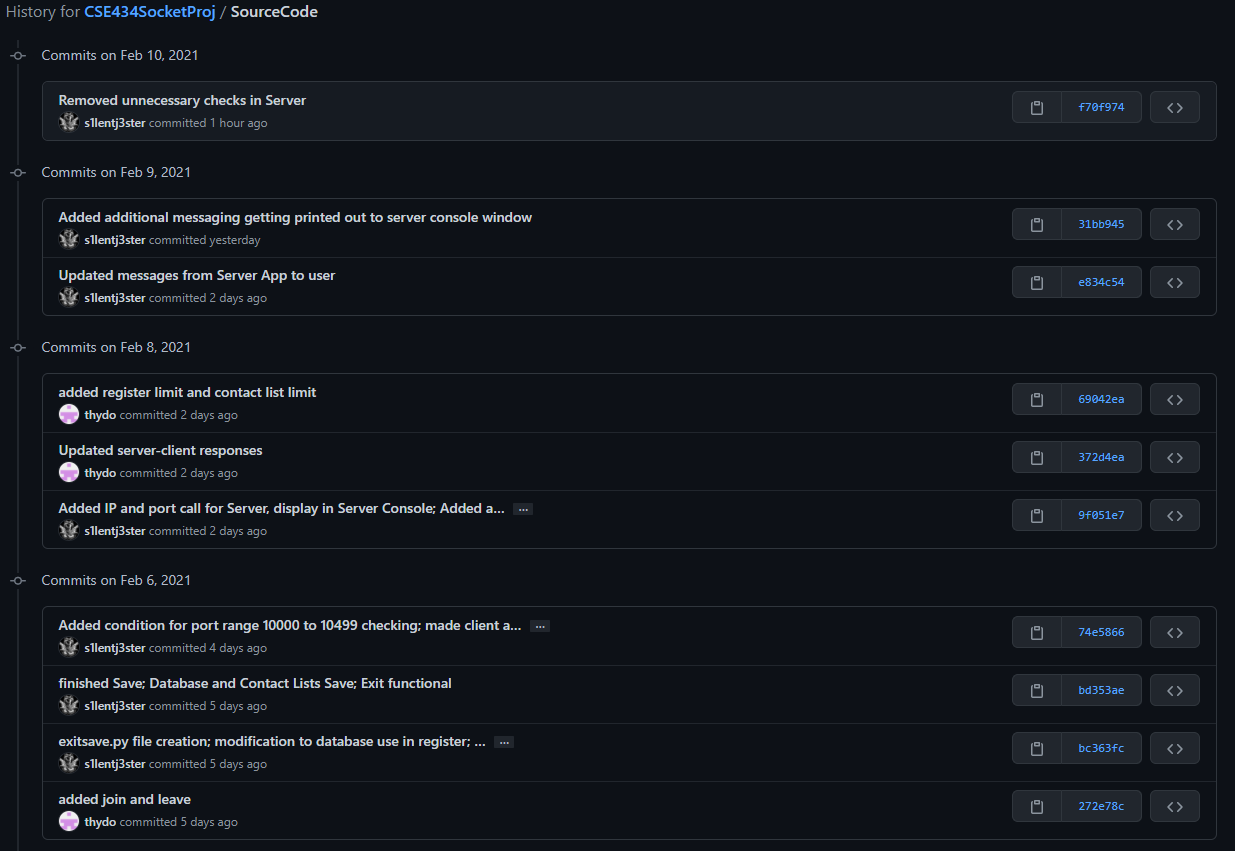
* + Save: Command format is save <file-name> and a text file is save in the format of file-name.txt. FAILURE is sent if command’s parameters are not provided. Otherwise SUCCESS is sent.

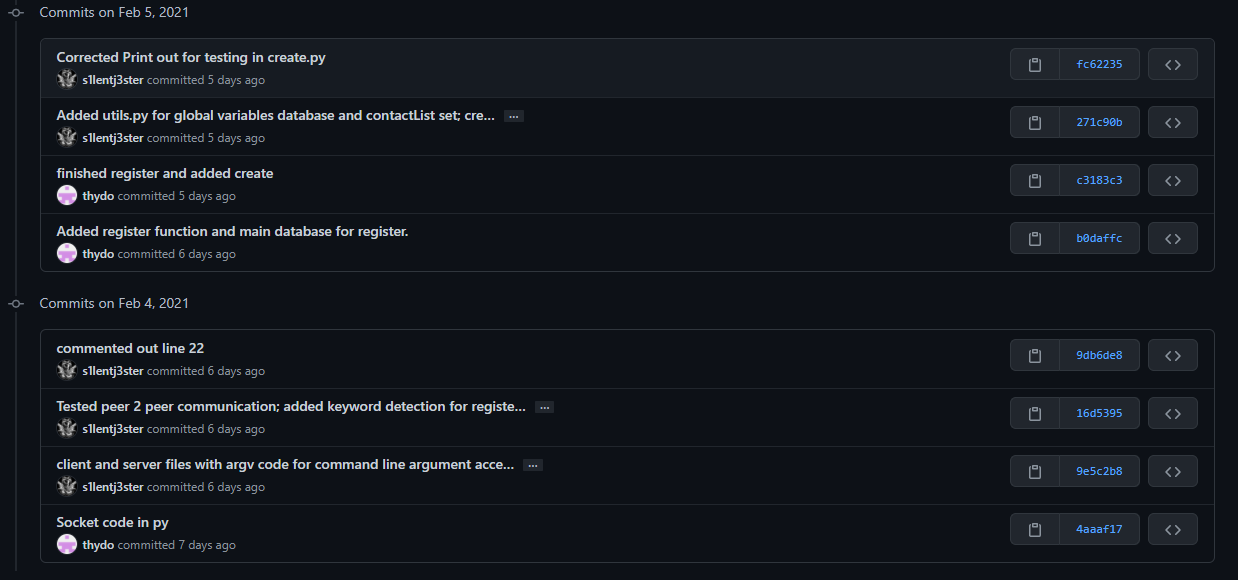


* + Exit: Format of command is exit <contact-name>. Currently, the command will remove the contact-name from the registered database and all joined lists. FAILURE code is sent if the user doesn’t provide sufficient parameters, or name does not exist in registered database or contact lists. SUCCESS is sent otherwise.



**Git history**





**Link to demo video:** <insert here>