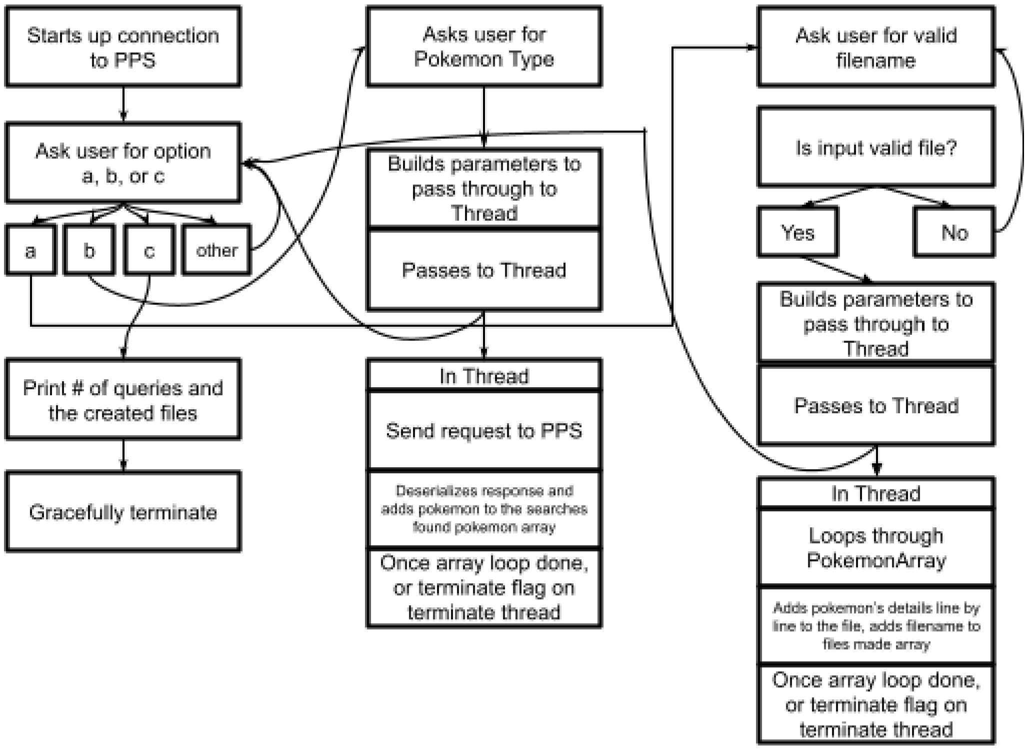
# COMP 2401: Intro to Systems Programming - Summer 2023
# Assignment 4 Design

101256173

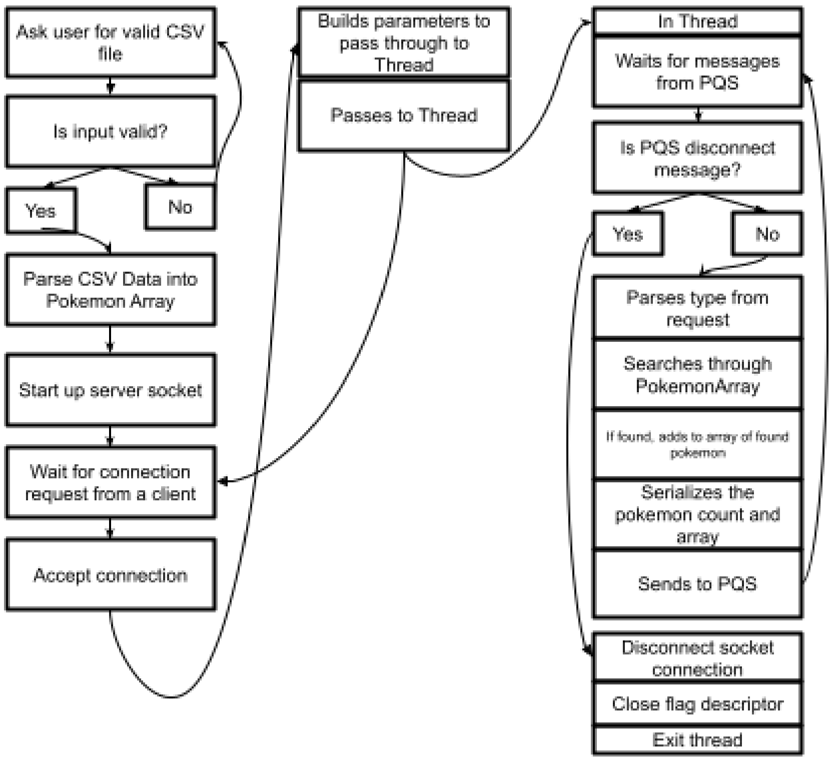
Robert Tseitin

## Program Flow - PQC



Starts up connection to PPS
Ask user for option a, b, or c
a
b
c
other
Print # of queries and the created files
Gracefully terminate
Asks user for Pokemon Type
Builds parameters to pass through to Thread
Passes to Thread
In Thread
Send request to PPS
Deserializes response and adds pokemon to the searches found pokemon array
Once array loop done, or terminate flag on terminate thread
Ask user for valid filename
Is input valid file?
Yes
No
Builds parameters to pass through to Thread
Passes to Thread
In Thread
Loops through PokemonArray
Adds pokemon's details line by line to the file, adds filename to files made array
Once array loop done, or terminate flag on terminate thread

## Program Flow - PSS



Ask user for valid CSV file
Is input valid?
Yes
No
Parse CSV Data into Pokemon Array
Start up server socket
Wait for connection request from a client
Accept connection
Builds parameters to pass through to Thread
Passes to Thread
In Thread
Waits for messages from PQS
Is PQS disconnect message?
Yes
No
Parses type from request
Searches through PokemonArray
If found, adds to array of found pokemon
Serializes the pokemon count and array
Sends to PQS
Disconnect socket connection
Close flag descriptor
Exit thread

**COMP 2401: Intro to Systems Programming - Summer 2023**
**Assignment 4 Design**
101256173
Robert Tseitin

---

## File Structure:

**server.c** - Has the main function of the PPS program, and handles the way the program flows.
**client.c** - Has the main function of the PQC program, and handles the way the program flows.

**utils.c** - Has the implementation for utility functions that are necessary for the program, like getting a valid file from the user.

**utils.h** - Contains forward declarations for utils.c and has the Pokemon structure in it as that's a structure whose utility is necessary for all other files in the program

**thread.c** - Has the functions for the thread routines that get spun off in the main function

**thread.h** - Contains forward declarations for thread.c and contains the structure for the parameter pass through necessary for them to function which gets utilized in the main function

**Reason for separation**: This file structure makes things more readable and assigns a file to all functionality for the program in a manner that is easy to follow and makes sense for those reasons it is separate.

The separation also exists because the utils and thread functions used between the client and server are shared a lot of the time, and both the PQC and PSS need to be separate given how they are two different programs with a different compiled executable.

## Libraries and C Features Used:

**pthread** - Used the library to spin off/create threads and manage them, and also for mutex to ensure threads and the main modifying the same data didn't write at the same time and potentially corrupt or override each other.
**string** - Used it mostly for strcmp to easily compare char arrays that are strings without implementing my own method.

**stdio** - Used for file i/o to open/make files or directories and write to or read them.

**stdlib** - Used for functions that handle memory allocation including free and realloc to dynamically handle memory such as those for arrays of dynamically changing sizes including strings.
**unistd** - Used for closing flag descriptors
**sys/socket** - Used for implementing sockets on both the server and client end to allow for a tcp/ip connections and any related server functionality.
**arpa/inet.h** - Used for converting an ipv4 address to a binary form in network byte order

**COMP 2401: Intro to Systems Programming - Summer 2023**
**Assignment 4 Design**
101256173
Robert Tseitin

---

## Functional and Non-Functional Requirements:

### Functional

**Case 1:** The PPS prompts the admin to enter a file with pokemon descriptions and does so until a valid file is provided. It then starts listening for connections on localhost port 7000 for requests from any clients

**Case 2:** The PQC establishes a connection with the PPS with the appropriate messages, and shows a menu with the options described (1) type search (2) save results (3) exit

**Case 3:** When the user selects option 1 on the PQC, it lets the user input a type, a thread is spun off, a query request is made to the PPS and it responds with serialized data of pokemon with that type 1, within the thread the data gets deserialized and stored in memory outside the scope of the thread.

**Case 4:** If option 2 is selected, the user is asked for a file name, they provide a directory/path for it to be made or accessed. Once a file can be made, it spins off a thread with info about the file, query results, and runs in parallel where it saves the data of the pokemon into the above mentioned file.

**Case 5:** If option 3 is selected, all active queries are terminated, saves, and the connection as well, dynamically allocated data gets gracefully freed, and a list of filenames of saves, and the number of queries is displayed as well. Then, it finishes gracefully.

### Non-Functional

**Performance:** The program is responsive through the use of threads, the PQC has queries to the PSS and saves all run in parallel with the use of mutex to prevent operation overlap and memory corruption as a result. Similarly, the PSS uses threads for performance to allow multiple simultaneous client connections, thus multiple PQC servers can be connected to the PSS and make queries at the same time.

**Maintainability:** The program is split up in to a variety of C and header files, most notably separate C files for the PSS and PQC so both can be edited individually while also having shared C and header files in the form of utility functions and information such as getFile and the pokemon struct. Alongside a separate file for the thread functions to make the code more readable. There is also a makefile which allows for the two programs to be compiled easily while also allowing only the object files dependent on changes you made to make compiling faster as well.