Design Specification

> *For the completion of the fundamentals of software engineering course at the University of Texas at Arlington*

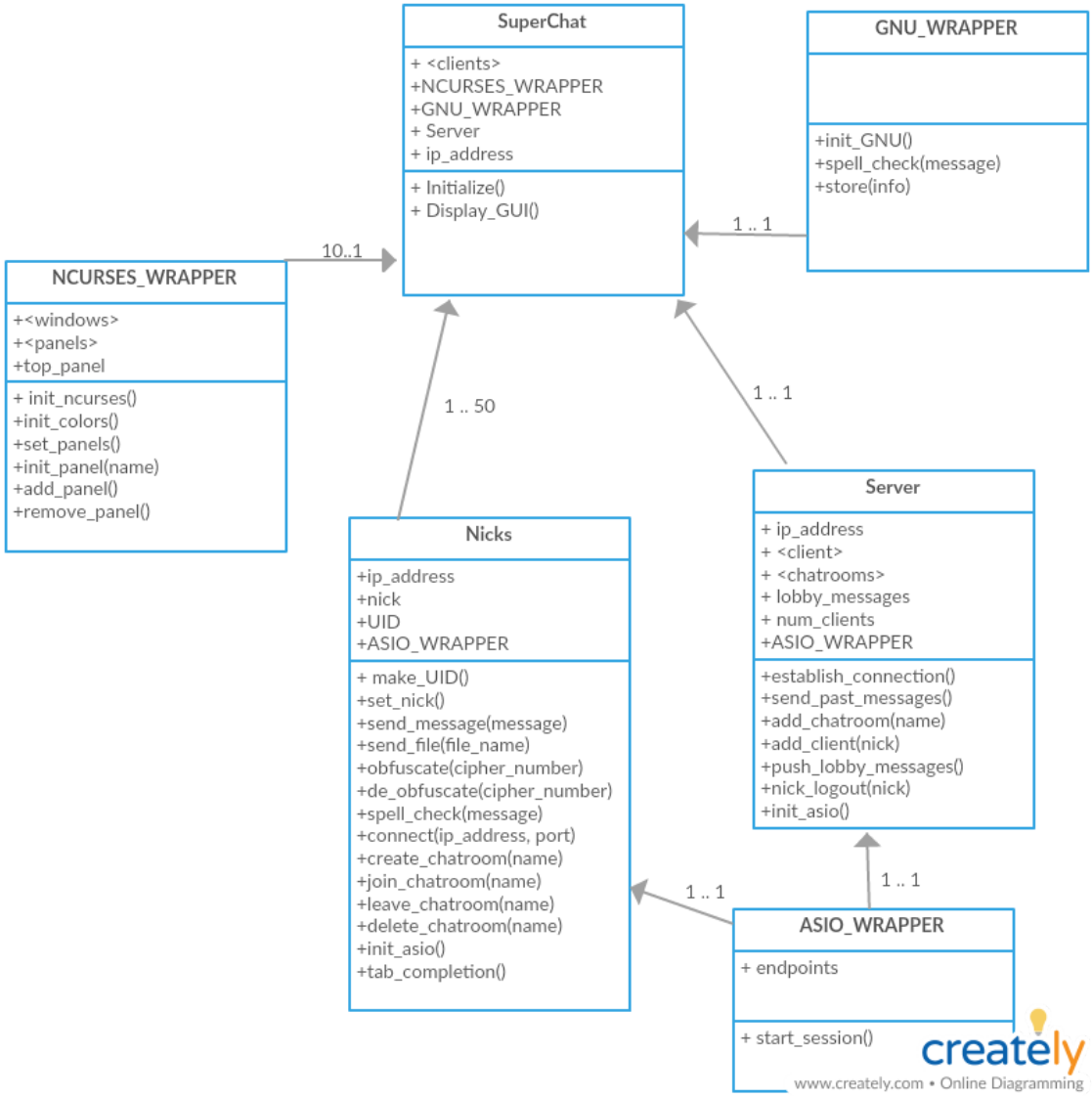
By: Group 2

> *Gerardo Rodriguez and Luis Flores*

SuperChat, a lightweight terminal application, has been considered as an application for development and our team has considered it feasible with the given temporal and financial restraints. The following document specifies a static model which our team will use to develop the application, a written description of each class in the static model, a listing of classes to specify whether they belong on the server or the client, and a revised requirements table.

Static Model Class Diagram

The following class diagram specifies the relationships each object has with every other object while specifying the functions and attributes that each object will have.



Class Descriptions

SuperChat

The SuperChat class is merely an interface class meant to show the structural design of the software. This class will not be implemented but will serve the developers in appropriately identifying the relationships within the system.

Server

The server class serves up to 50 clients and is meant to be run on a single machine, a single IP address, and a single port. The server —in conjunction with the clients— will work in a fat-client architecture to optimize server speed for quick message delivery. Considering the fact that the graphical user interface is text-based and the system's language, a fat-client should not be an issue.

The server will be able to log users out, establish connections, and print the past few messages of any room.

Nicks

The client has an array of functions alongside its networking capabilities (not limited to connecting into a server). The client can send and receive files to and from all users, respectively, send and receive messages to and from all users, respectively, encrypt and decrypt messages according to a Caeser cipher, and create and delete chatrooms.

GNU Wrapper

The GNU Wrapper calls functions to store constant user information and to spell check messages before being sent out to the chat. The GNU Wrapper wraps around the GNU readline library which contains functions that will allow the program to spell check messages and aid in storing constant information.

NCurses_Wrapper

The NCurses_Wrapper wraps around the NCURSES library. The attribute is a vector of windows of size 10 to appropriately limit the number of chatroooms available. The attribute is a vector of panels that can dynamically change if a user requires help or if they wish to obfuscate a message. The respective methods initialize NCURSES, the panels, and modify any visuals.

ASIO Wrapper

The ASIO Wrapper is an interface that initializes a TCP session between a server and a nick. The relevant attribute for Nicks is the endpoints attribute while the start_session() method is relevant to both the server and the nick.

Server or Client?

Server

ASIO Wrapper and the server wrapper will be used in the server class.

Client

The client class will use the Ncurses Wrapper, the ASIO Wrapper, the GNU Wrapper and the Nick wrapper.

Requirements Table

UID	(N/F)	Source	(C/S/B)	Written Requirements	Notes	Class & Method
001	F	<ctime>	C	The messages will be tagged with the date they were sent on	N/A	nick / send_message
002	N	<ctime>	S	Use library <ctime> to query time and date	N/A	nonfunctional
003	F	Client nick	B	The messages will be tagged with the nick	N/A	nick/ send_message
004	N	Client input	S	The nicks will be stored in a	N/A	nonfunctional

				string vector		
005	N	Client input	B	The nick entered by the user shall be verified by the server to determine any collisions with any retired or active nicks.	N/A	nonfunctional
006	F	<ctime>	C	The messages will be tagged with the time they were sent on	N/A	nick/ send_message
007	N	N/A	S	The server should only be established on ports above 1024	N/A	nonfunctional
008	F	Server	S	The server should provide users with past messages of a chatroom	N/A	server/send_past_messages
009	N	Client input	B	The messages must be sent 1 second after user enters send	N/A	nonfunctional
010	N	<ncurses>	C	The user interface for the client will be created with ncurses	N/A	nonfunctional
011	N	N/A	B	The application will be developed in C++11	N/A	nonfunctional
012	N	N/A	B	The application's networking capacities will be carried out	N/A	nonfunctional

				by the ASIO library		
013	F	User input	B	The users will be identified by nicks specified by the user	N/A	nick / setNick and send_message
014	F	User input	B	User input will be spell-checked by an external dictionary	N/A	GNU_WRAPPER / spellCheck
015	F	User input	C	Tab completion will allow the client to complete words used often by the client	N/A	nick / tabCompletion()
016	F	User directory	B	File transfer will be allowed from one user's home directory to another user's home directory.	N/A	nick / send_file
017	N	User directory	B	The file transfer feature will be implemented using sftp (safe file transfer protocol)	N/A	nonfunctional
018	F	Server	B	Up to ten chatrooms will be allowed in a specific server	N/A	server and NCURSES_WRAPPER/ add_chatroom(name)
019	N	Server	S	The chatrooms and their information will be stored in a chatroom object vector	N/A	nonfunctional

				with error checking to prevent an 11th room from being created.		
020	F	User input	B	Any user can create or remove a chat room if the chatroom is empty	N/A	nick / create_chatroom
021	N	User input	B	The user can create a chatroom by typing and entering #c in the lobby	N/A	nonfunctional
022	N	User input	B	The user can remove a chatroom by typing and entering #r in the respective chatroom	N/A	nonfunctional
023	F	User input	C	The user may obfuscate their message	N/A	nick / obfuscate message
024	N	User input	C	The user may encrypt their message by typing and entering #e. The system will query the user with a question "Caesar cipher: " to specify the encryption method. To remove the encryption feature, the user types and enters #e again.	N/A	nonfunctional

025	N	User input	C	The user may decrypt any obfuscated messages by typing #d. The system will query the user with a question "Caesar cipher: " to specify the decryption method. To remove the decryption feature, the user types and enters #d again.	N/A	nonfunctional
026	F	Server	S	The server will give the user an option to logout	N/A	server/ nick_logout
027	N	User Input	C	The user can enter #help for a help menu showing all the command a user is allowed to use	N/A	nonfunctional
028	F	Server	S	The server will automatically start with a default lobby chatroom that cannot be deleted	N/A	server / initASIO
029	F	Server	S	The server will support up to 50 users	N/A	server / add_user
030	N	User input	C	The user can block or "ban" another user's messages by using #b	N/A	nonfunctional
031	F	Server	S	The server will	N/A	NCURSES_WRAPPER /

				show how to logout, get to the lobby, and change chatrooms at the top of the server		init_ncurses
032	B	User Input	S	When the user enters #chatrooms a new window will show the user all the available chatrooms	N/A	NCURSES_WRAPPER / init_ncurses
033	F	User input	S	The chatrooms window will tell the user how many people are in each chatroom	N/A	NCURSES_WRAPPER / init_ncurses
034	F	Server	C	The chatroom will show when anew user has joined the current chatroom	N/A	NCURSES_WRAPPER / init_ncurses
035	F	Server	C	The chatroom will show which user created the chatroom	N/A	NCURSES_WRAPPER / init_ncurses
036	N	Server	C	The user can return to the main lobby chatroom by typing #lobby	N/A	nonfunctional
037	N	Server	C	The chatrooms window will tell the user if they've entered incorrect options when	N/A	nonfunctional

				chosing a chatroom		