This brief project epilogue outlines some final thoughts on the project such as elements of the implementation which differed from the design, as well as some challenges which arose during the entire process.

Final Project Notes:

* The original design was to use text files for data storage, which was later changed to be in-object data fields. The original design can be seen here: “As previously stated, the IRRC infrastructure uses two main components – client and server. The client will communicate with the server using the java socket class. When the client attempts a connection to the server, user authentication is required. Once the user logs in or makes an account, the client will be able to read input from the user to send to the server, and display output back to the user from the server. The server will use channels as conduits of information – channels being the general chat and private chats. The server will keep a limited backscroll for each channel in a text file to display. If more than the specified number of lines are written to a channel, the first line of the channel text file will be removed and the next line will be written at the end. Each time a new message is written to a channel, the server updates the necessary text file and redisplays the file’s contents in the channel history. Every time this happens, the server will resend the text data for the channel to all of the connected clients in the channel.”

This was changed to be a non-file data-oriented design using data structures in the object classes. The reason for this was simplicity and efficiency of the code.

* Certain refinement elements of the client UI such as “To leave a channel, the user can issue a “/leave” command, or simply click the X button on the right-hand side of the channel element.” were removed in favor of a clutter-free interface.
* All other core functional and non-functional requirements remained unchanged and were addressed by the project itself.
* It was noted throughout the implementation process that most of the code modules took longer than originally expected to complete.
* Unit testing during the implementation phase revealed many more faults than originally expected.
* Since many of the faults were addressed in earlier phases, a not very rigorous post-delivery maintenance phase would be expected.