Application Projects

Cookbook Application- Working with a group we created a graphical Cook Book application using Java. This project was done for a software science course and the focus was on creating larger programs with multiple classes while maintaining readability and robustness. A heavy focus was placed on documentation including: requirements identification, use-case diagrams, sequence diagrams, and UML diagrams. My contributions included a graphical planner that updated shopping lists and ingredient lists stored as text files based on the user's recipe selections, and the java panels to view recipes. This was accomplished by implementing and extending the Java.Swing classes with multiple self written classes that made use of inheritance and polymorphism

Spreadsheet Application - A spreadsheet implemented in C#. This was a semester long project built using Visual Studio combining multiple self written classes. I first wrote classes dealing with the back end logic. This included classes to: parse user input and check if it was in a valid format using regular expressions, a class representing a dependency graph object used to keep track of the relationships between different variables in each cell of the spreadsheet, and delegate and lookup functions to keep track of the value of variables in the spreadsheet. These classes were then used as utilities for the graphical component of the spreadsheet. The GUI was built using Visual Studio's Microsoft Form Application to adhere to the Model-View-Controller design pattern.

Spreadsheet With Server - This was a group project to add functionality to the spreadsheet application. We sought to create a backend server responsible for hosting multiple spreadsheets that could be edited by multiple users simultaneously in real time. A protocol was developed by the class for sending spreadsheets between the client and the server, and everyone had to adhere to that protocol so that everyone's clients would work on everyone's servers. I did much of the back-end programming for the server using C++. I made extensive use of sockets to listen for incoming users. When a user connected to the server their socket would be opened and spun onto a new thread that handled their session to maintain asynchronous behavior. Users could join the same session and a queue making use of mutex locks was used to ensure user's change requests happened in the right order without tripping on another user's request.

Boggle Game Application and Server - This project implemented a client GUI for playing Boggle, and a back-end server to keep track of game logic and game history. Using a GUI built in C# and asynchronous sockets the client GUI connected to the server and requested to join a game. The server looked for other users requesting a game and spun a new thread to handle their Boggle session. Much of the back-end logic for the boggle game was given by the instructor of the course. The server I implemented also made use of an SQL database to keep track of a user's game history. Words played, wins, losses, and past games were stored in simple tables in this database, and the server made use of SQL calls to update and maintain this database.

Systems Projects

Simple Shell - A very simple shell written in C. This program took in command line arguments from the user and parsed them. A jobs list was maintained of currently running applications. A user could request a job to be started in the background or foreground of the shell. A foreground job request used the execve linux system call to run the executable. A background jobs request was spun onto a new thread and started using the execve linux system call and the job list updated.

Malloc - A simple implementation of malloc in C using an explicit list to keep track of free blocks, headers and footers for allocated blocks. This involved extensive pointer arithmetic and manipulation in C.Optimizations were made for coalescing and remallocing aswell.

Finite State Automata/Pushdown Automata/Turing Machines/Grammars - Built multiple automata using JFLAP, including DFA's and PDA's to check if an input belongs to some regular language. Built multiple turing machines in JFLAP, including a three tape turing machine for parsing memory locations in a finite length string. Also implemented a simple grammar using JavaCC.