Proposal

By: Justin, Micah, Tang, Unnati

November 2nd, 2023

Introduction

The system will involve a single server that is connected to a database, which can then be accessed by multiple clients, each with their own assigned threads. When accessed, the clients will be encountered with a GUI that provides the user with the options for logging into an existing account, or creating an account by inputting a username, password, and email address. A database accessed by the server will store all account information.

Perspectives:

In the eyes of the customer, this will be a multi-client, single-server system that allows their users to sign into their email through a 2-step verification process, involving the user's username and password, and a verification code.

To the eyes of the user, this will be an application that allows them to sign into their email.

In the eyes of the development team, this system will be a multi-client, single-server system, where each client has their own thread. The server will access a database that stores all client login information (username, email, and password). When the user accesses the server, they are accessing a GUI that allows them to either login to an existing account or create an account. The account will be based on the 3 pieces of information provided by the user. We will also implement a second set of verification that requires the user to input a code sent to the valid email that they set up at creation and will lock them out if they fail to log in correctly after 3 attempts. The development team will also create a method to keep track of the online, registered, offline, and locked accounts.

Development Environment:

The development team will be programming this system using Java in the Eclipse development environment. They will be using either Swift or JavaFX as the GUI library. Gmail will be the email server they use, and they will use My SQL for the database management system.

Concerns:

The main concerns of the development team lie in the ability to integrate multithreading connections with the swift GUI. They are also considering the method in getting the client to properly ask for information, based on what buttons are clicked in the GUI, and how the server will return the correct information from the database.

Questions:

Can we use any emails or just Gmail, outlook, or institutional (in this case, Cal Lutheran) provided email?

Will code authentication be counted in the 3 login attempts?

Can 2 people have the same email address?

What are the minimum requirements for the password, and are there any other password requirements?

Are we writing our own database system?

Can we do a “send new code” feature? Is that allowed within the UI after they need a code for recovering their password? (i.e., “I did not receive an email. Send a new code.”)

Biography:

Justin Bouse

Justin Bouse has experience in Java, C++, assembly, and python. The most complex program he has written is a tic tac toe game application. It received two inputs (one from player one and another from player two) and checked those inputs to determine a winner after each turn. It

would then display an output on the screen, based on the determined result, specifying a win, draw, or loss. At the same time, X's and O's would appear in the specified location on a tic tac toe GUI display board. The application involved a GUI interface using JavaFx and some backend code that was used to implement the game and check for a winner

Micah Wisniewski

Micah Wisniewski has experience in java, python, and C++. The most complex program he has written is a simplified version of the DES encryption algorithm. This program receives a String input message and encodes it using a combination of bit operations, elements in a Feistel algorithm (including key expanders and substitution), a key extraction function, and preprocessing and postprocessing functions. This program would take the String, encrypt it, output the encrypted String, decrypt it, and then output the decrypted String (the same as the original input).

Tang Favish

Tang Favish has programming experience in Python, Java, Swift, and R. The most complex program she has written is a mobile application developed in XCode. This endeavor entailed the creation of user interface components and the implementation of mathematical backend logic, all executed within the Swift programming language. The overarching objective of this application was to streamline the process of equitably distributing restaurant bills, accounting for tax and gratuity, making it easier for people to split the bill of large groups after a restaurant. At its core, the application permits users to manually input detailed dining expenses, along the person who incurred them and their respective prices. Real-time computation, facilitated by Swift's computational capabilities, creates the automatic generation of equitable individual shares, which nixes the need for manual calculations or external tools. Additionally, the application seamlessly integrates with the Swift UI camera module, enabling users to capture and store receipts within the app's framework.

Unnati Diya Maharjan

Unnati Diya Maharjan has experience with Java and Python. The most complex program written by Unnati was a program that sorts databases using a variety of input parameters and effectively and efficiently extracts information based on the provided data.