Matthew Yaswinski and Josh Del Toro

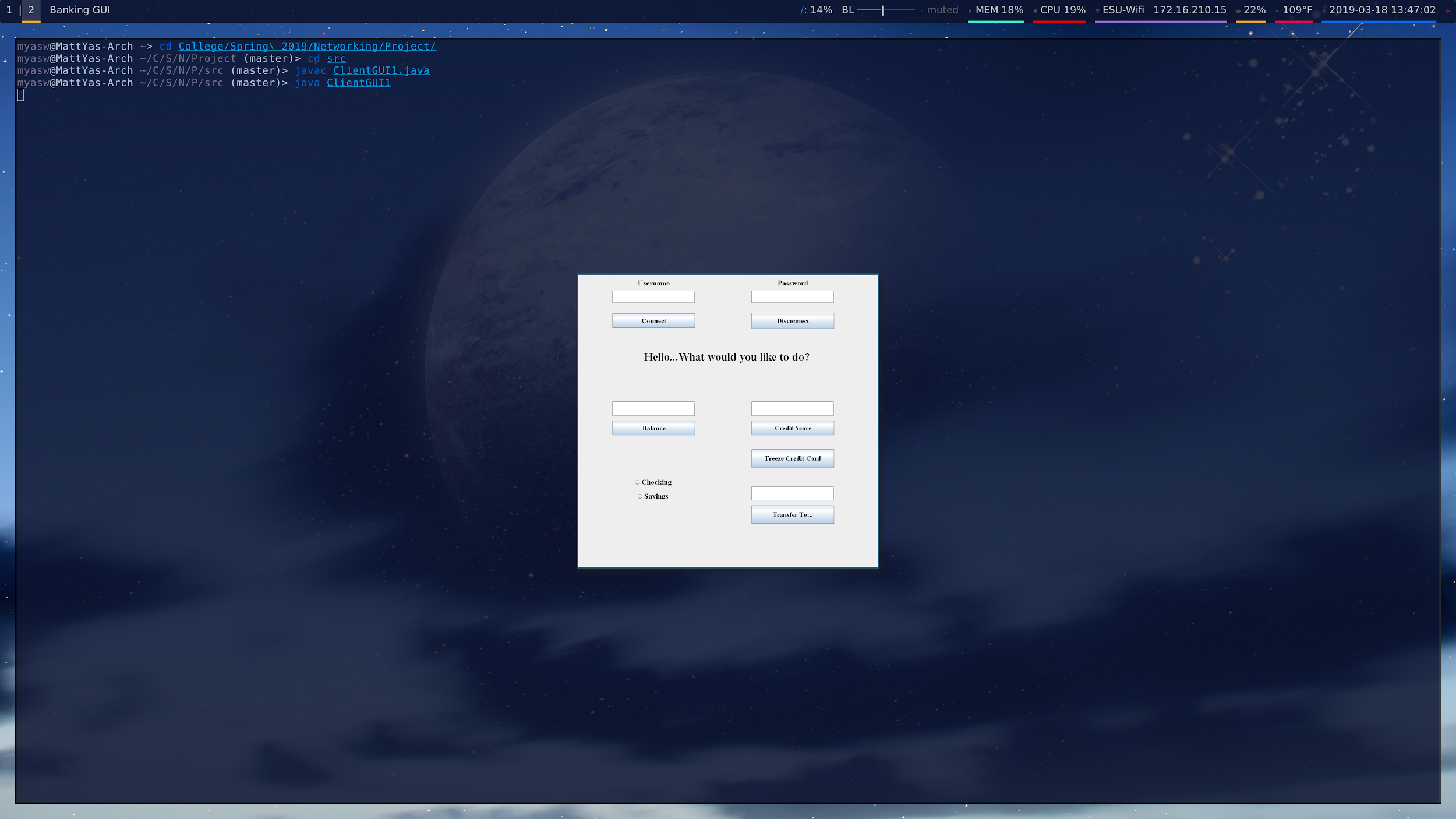
Dr. Kimm

Networking

21 March 2019

The Banking App

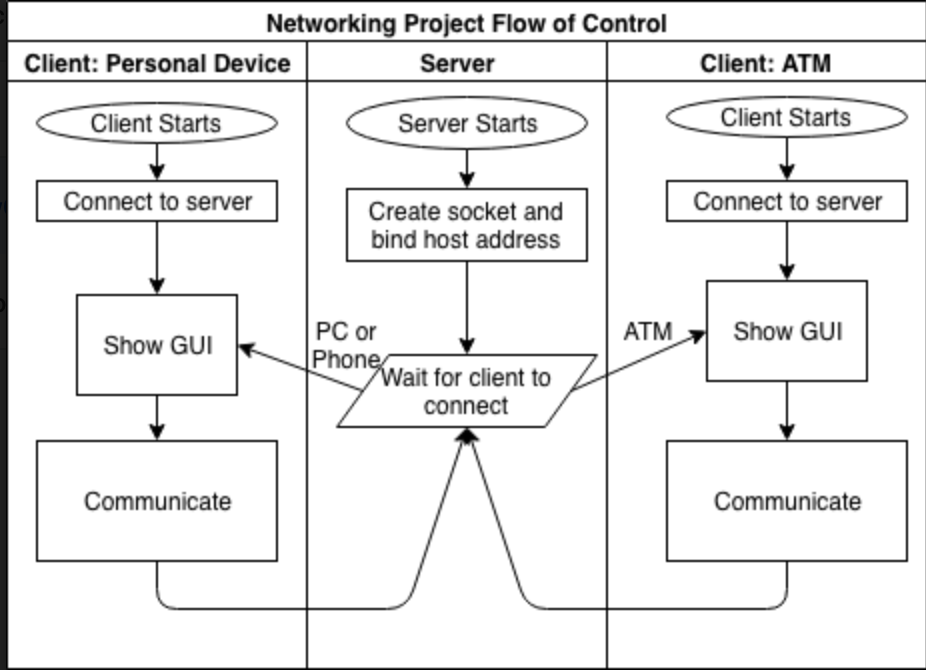
For our final project, we will be creating a banking app that will be accessed by either a mobile device or a personal computer. We hope to also develop an ATM client with the same server that handles mobile use so that users may be able to withdraw cash. From this app, the user will be shown a GUI where he or she will have access to at least two accounts: Checking and Savings. A possible feature that may be added is for the user to create more accounts. The interface will be filled with text fields and buttons for people to display values and enter values. Some of the buttons include Balance, which will display the balance of either the Checking account or the Savings account. The balance will be displayed in a text field just above the button. Another button is the Freeze Credit Card button. This will freeze a user’s credit card when the button is pressed. Another button is the Transfer to… button. This will allow for funds to be transferred between the user’s Checking and Savings account, and the user can specify where they want the funds being transferred to, which means that money will be taken out of the other one. A GUI is shown below.



The user will be able to log in by entering their Username and Password at the top of the GUI, and then pressing connect. Once this is done, a message below these boxes saying “Hello [name]. What would you like to do?” will be shown. The user will also be able to disconnect from the server by pressing the Disconnect button, which will clear all the fields to prevent a breach in confidentiality.

This app will have numerous applications. One such application is on a phone or a PC to manage a bank account. Many banks have a mobile app for people to check their balance or transfer funds between accounts. This project is only on a local router, but if the ability to use it over the Internet were added, it would be possible to accomplish this. However, a more ready use for this program would be on an ATM. ATM’s allow people to do a variety of things, such as withdraw money. A few more buttons would make it possible for this to be done.

Now that we have established the uses of the banking app, it is necessary to discuss the general architecture of the backend and frontend processes. The control flow graph below will help us explain this architecture.



In the flow of control, control starts with the server. The server starts up, creates a socket, binds the host address to the socket, manages errors, and then listens for clients. At this point, the server will be able to fork control to multiple clients, but it must wait for such clients to connect. Now that it is waiting, both Personal device clients and ATM clients may be started up. The client programs will connect to the server immediately upon start up. Then, they will show the GUI to the user. The user may now enter their credentials, so that they, as individuals, are connected to their accounts on the backend. They may proceed to communicate with the server and use any features that are provided to them in the GUI. When they are done, they will yield control of the network resources that they are using back to the server, which will continue to wait for more clients to connect. Due to our use of a router, we will limit the amount of clients that may connect to the server at one point to 30 users.

Another feature that we plan to add to our app is the ability to create an account. At a minimum, the user will have to create a unique username and type in the password of their choice twice. If the username is not taken and the password matches the confirmation password, then an account will be created for them. If we have the resources, we may also implement email confirmation and profile customization. Email confirmation would consist of sending an email to the user’s email account and having them verify that they exist. This would help reduce the amount of bots that create accounts. Profile customization would allow the user to enter more information about themselves, which could help with identification in the case that they forgot their password. They would be able to add the following: first name, last name, three security questions, address, and phone number. Users would also be able to edit this information.

The user’s communication is the most important factor within this project. The personal device client will begin by connecting to the server through a predefined port. Then, the user has two options. If the user has already created their account, they may enter their username and password to the respective fields, and then press connect. Upon the server recognizing these two data points from a text file, a message will be displayed as such: “Hello [Name]. What would you like to do?” In the case that the user has not created an account, they will instead do so through a “Create an account” redirect. They will create a username and a password, and then confirm the password. They will get an error if their username has already be taken. They will also be given an error if their passwords does not match the confirmation password. Otherwise, their account will be created and added to a text file, and then they will be able to view their account information. Now that the user is connected to their account, they have will be able to transfer funds between their checking and savings accounts. Depending on which radio button is selected and how much money is entered in the field above the ‘Transfer To…’ button, the text file will be changed so that the amount of money chosen is moved to whichever account is selected. In the case that the account not selected does not have enough money when the ‘Transfer to…’ button is clicked, the user will receive a notification that they do not have sufficient funds. The Credit Score button will read a value out of a text file. If this app were to be used commercially, we would have a third party, such as Equifax, gather this data. The balance button reads the amount of the selected radio button account, whether it be checking or savings. The ‘Freeze Credit Card’ button will first ask the user to confirm that they would like to do so. Afterwards, it changes a value within the text file. Commercially, this would deactivate a credit card. If the user wanted to reactivate the credit card, they would have to call the bank that owns the app. All values that must be displayed in the GUI will be above their respective buttons, and values can be typed into the field above the ‘Transfer to…’ button.

A feature that we would like to add to our system is transferring funds between users. This feature would be slightly more difficult to implement then other features. First, we would have to verify that the user attempting to send money has the money that they want to send. Then, we would have to verify that the user they are trying to send money to exists. Finally, we would have to remove the money from the sender’s account and add it to the receiver’s account.

Another potential feature for the future would be a transaction history. For each user, we would keep track of all deposits, withdrawals, and other transfers of money. The user would then be able to view all of their transactions from a single page. If they have multiple accounts, we may either label each transaction with an account number, or we will split the transaction history page by account.

Apps, including this one, must be maintained. To do this, we may implement a hierarchy of users. There will be a system administrator, with full access to editing, removing, and adding users. There will also be the bank teller, who should be able to view any user’s bank account information. The last type of user that must be implemented is the consumer, the main audience of this app. The separation of these roles will allow system administrators to maintain the app, bank tellers to make financial decisions, and customers to utilize their bank.

Below is an image of a draft ATM GUI.

