The ATM machine is a client side application that takes simple requests such as creating a customer, creating a new Account and perform withdrawals and deposits. Everything on this application is client side including data storage. Therefore, this would be something that would be installed onto an ATM machine, as allowing a computer to download it would be a huge security risk because the host machine would have all of the data in the system. The program has seven different capabilities. These are as follows: creating a customer, creating an account for that customer, depositing money to an account, withdrawing money from an account, transferring money to an account from another account, viewing an account's transaction log, and viewing all of your accounts per customer.

The backend storage system is purely json text based. As soon as the program starts it reads in every single customer along with their accounts that are stored in their customer folders. There is a root storage folder that contains every user and some config files and helper files. The “users.json” file contains the data for every single user in the system. We designed a JSON parser that maps out every user as a json object and prints it out to the file. The “accountRelationships.json” file maps every account by its account to a user by their ID. This is used for obtaining the owner of a specific account. Every account is then stored within the user’s folders according to who owns the accounts. Each use may have one checking account and one savings account. We store the account in a hashmap where the class is the key so a customer may not have several accounts of the same type.

The Create new account GUI has the user input their username, password, the type of account, and initial deposit for the new account. After the user inputs their username and starts typing their password the program verifies that the username is valid. If the username is not valid the TextField border is set to red, the text is cleared, and the prompt text is set to “Invalid Username”. Once a correct username is entered, the user enters their password into a PasswordField box that hides the password as it is entered. The user then picks which account they wish to make from a ChoiceBox which only has two options, “Checking” and “Saving”. Setting the initial amount is the last thing the user must do before hitting the submit button. If the user does not have an initial deposit of at least $50 and hits the submit button, a warning becomes visible to notify the user must have a higher initial deposit. Once the user hits the submit button the program verifies that every box is filled, and notifies the user if there is an error. If everything is filled correctly the program then checks if the user already has the desired account. If the user does have the account a warning becomes visible notifying the user and the form is cleared. If the user does not have the account the program creates the account for the user, deposits the initial deposit amount into that account, and notifies the user the account was created.

The customer main page GUI offers the customers an overview of their accounts. The GUI first checks if the user has a savings account, checking account, or both. It then prints out the corresponding account number and balance for each account the customer has affiliated with their username. The main page also offers customers the option to access a history of all the transactions made for each account.

When first starting the program it will display the LogInPane GUI. This pane will allow you to log in or create a new account. The login pane consists of a TextField for the username, a PasswordField and two buttons. Once you login, the program checks to make sure that the username exists. If the username does not exist, the GUI will display an error that reads "Could not find User". If the customer enters an incorrect password a "Password incorrect" error will be displayed. The GUI also check to see if both fields have something in them, if not it will display the error "One or both fields are empty" or suggest to create a new user in the CreateUserForm GUI.   
 The DepositPane GUI has a ChoiceBox, TextField, and a button. The ChoiceBox will give two options, deposit to Savings or Checking. The TextField is for the amount that the user wants to deposit. It will make sure that the user input can be parsed to a double, if not the error will display "Amount is Invalid". It is set up so that if the user selects to deposit into a Checking account and it doesn't exist it will give the error "Account not Found".

The withdrawPane GUI is almost identical to the depositPane GUI in terms of visibility. It contains a choice box, text field, and a button. The user will enter the value to be withdrawn in the text field and select the account from which to do the withdrawal using the choice box. The pane will make sure that the user selects a valid account and also ensures that the user has the proper funds, elsewise a warning will appear, with a corresponding error message.

The TransferPane GUI is set up so that the user can enter an account that money will be taken from and an account that money will be placed in, along with an amount to be transferred. It consist of three TextFeilds and a Button. Two of them will handle getting the to and the from accounts, and will return errors if the account doesn't have enough money, the account was not able to be found, and if the amount of money to be transferred isn't a number.

The TransactionListPane GUI is designed to allow the user to view their transactions for a particular account (Checking or Savings). The user simply selects one of the two using a choiceBox and clicks the view transactions button. Once this is done it reads from a designated text file containing attributes for each transaction that was created, and displays it on the GUI, clearing all other buttons and labels to allow space for the transactions to be shown (each item printed as a label).

