# Assumptions

* A VPN is being used to connect the network
* Only employees can connect to the network
* Shared files are stored on a local file server
* User files are stored on user devices
* Chatroom servers are already running on the network and reset every night
* User uses Windows OS

# Controls

* Only authorized individuals can access the important devices.
  1. A keycard is required to enter the building.
  2. A man trap is in place at the entrance way to the building.
  3. When a reader is used, a log is created with the user’s ID to identify who entered.
  4. A passcode and keycard is needed to enter server rooms.
* No unauthorized usage or users
  1. All devices are connected via ethernet to the network.
  2. A firewall is in place to protect the devices inside the network.
  3. To connect from the outside or to other branches, a VPN is required.
  4. An IDS is on the network.
  5. Unused ports are closed.
* Up to date protection
  1. Security updates are installed on devices monthly.
  2. Backups are created monthly on most devices, weekly on important systems.
  3. Backups are stored locally and on various servers around the world.
  4. Up to 10 backups are kept from various different times.

# Database

Custodian, Salesperson and Warehouse have access to the Inventory schema, but certain roles have certain access and certain roles to the database.

**Salesperson:**

Can view the Inventory schema itself, as well as the certain views.

Can update the Inventory schema, by either selling or buying planes.

**Warehouse:**

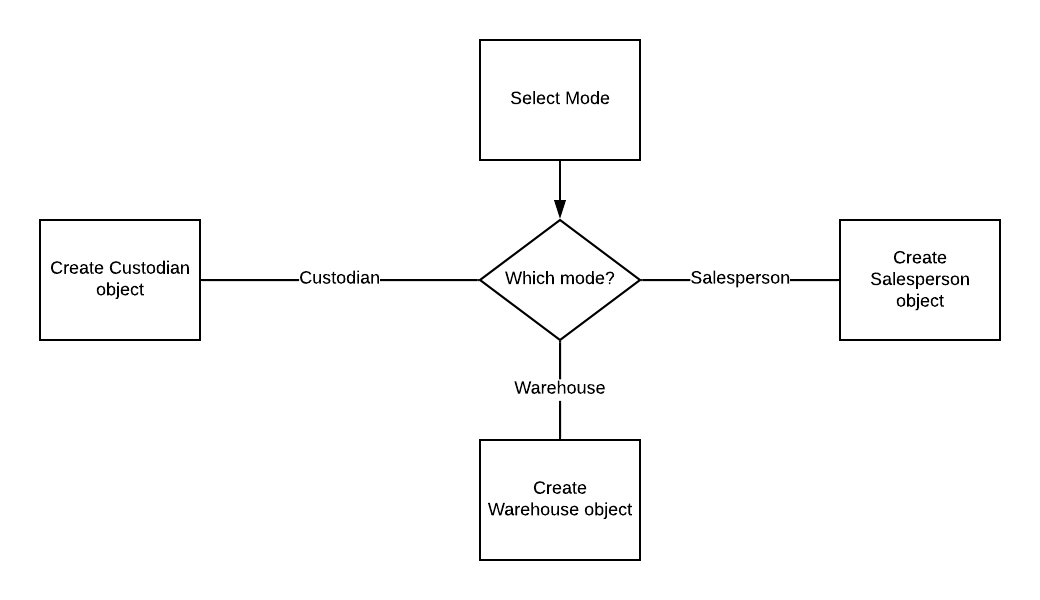
Can only view the Inventory Views.

**Custodian:**

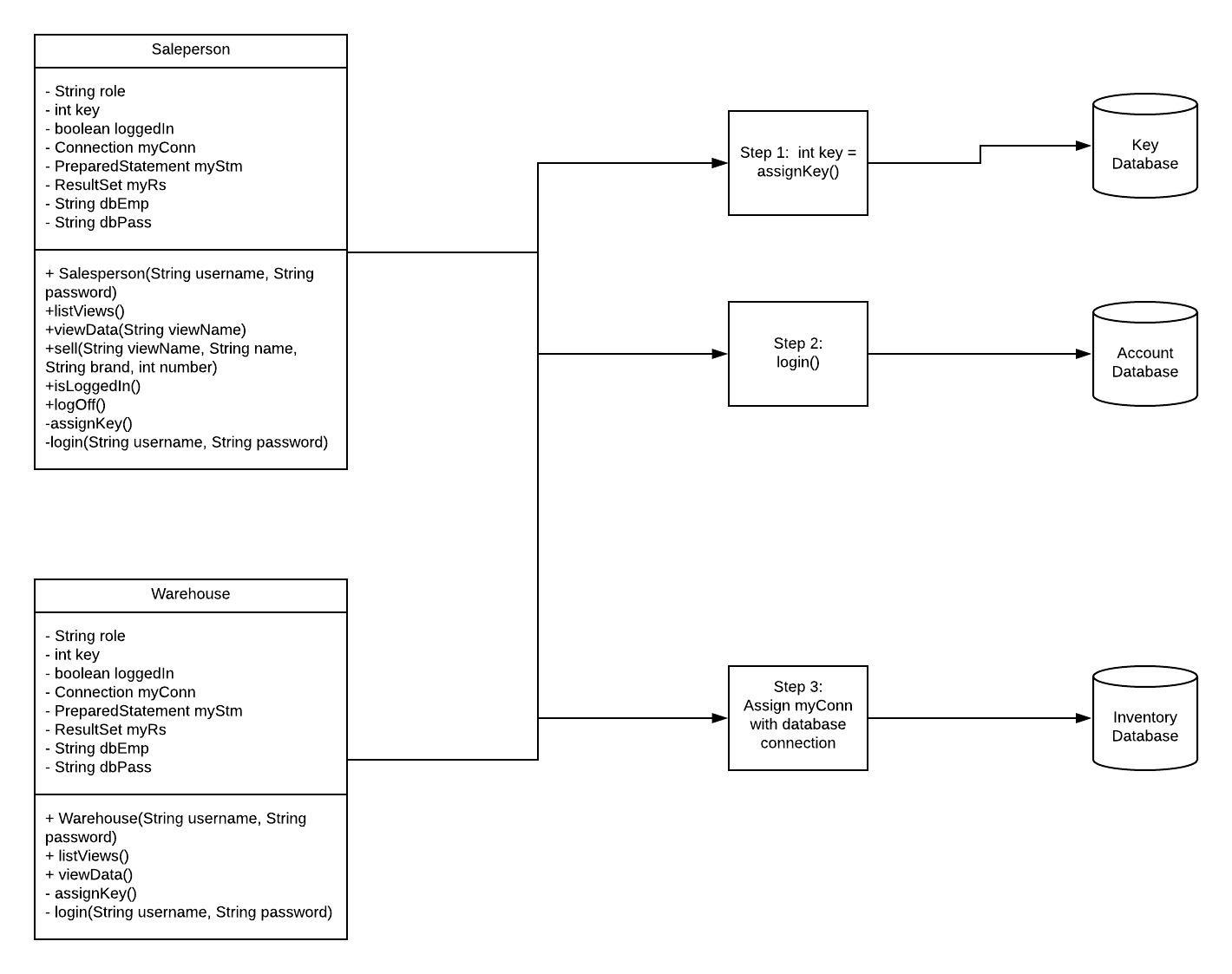
Has access to all schemas on the database. However, they can only edit the Account schema

Can update the Account table, by creating a user or deleting a user.

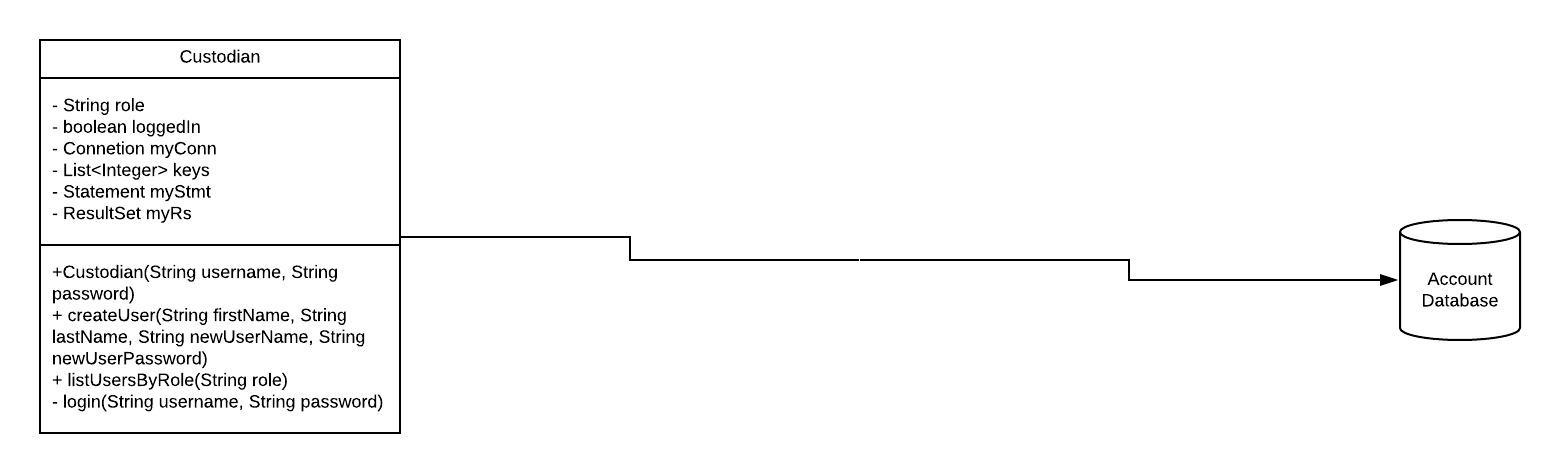
**Database Program Flow:**



For Salesperson and Warehouse, a read-only database account is given. It will query the key database, asking for the key based off the object that was created (either Salesperson Object or Warehouse Object). The key is then assigned to the Object, and then the login function is called. The login function encrypts the username and password parameters with the key retrieved from the database. The encrypted username/password combo is then used in a select query where it compares the encrypted combo with the account database. If it returns 1 record, then the account is verified. The chance of it returning more than 1 is very low, as duplicated usernames are not allowed.



The Custodian does not need to go through the key or account database as the account is directly associated with MySql Server



# Chatroom

To secure communications between branches and employees, users connect to a chatroom to discuss information. AES is used to encrypt information to and from clients where it’s integrity will be checked by a hash. All symmetric keys are exchanged using the Diffie-Hellman algorithm.

**Chatroom program flow:**

A close up of a map

Description automatically generated

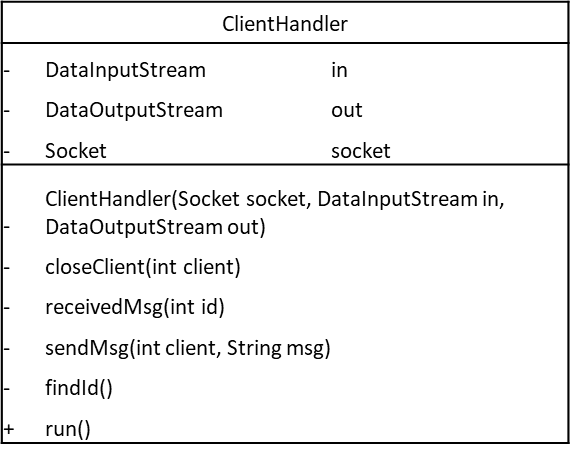
The client class creates a new client to connect to a server. The client sends messages to the server where they will then be rerouted to all clients connected. Before sending a message, the client will send an encrypted ping message. The server will then go though all keys to find the key and check it’s validity.

|  |  |  |
| --- | --- | --- |
| chatroom | | |
| - | JTextArea | msgArea |
| - | JTextField | msgField |
| - | JMenuBar | menuBar |
| - | DataOutputStream | toServer |
| - | DataInputStream | fromServer |
| - | InetAddress | address |
| - | Socket | socket |
| - | int | prime |
| - | int | generator |
| - | int | randNum |
| - | int | key |
| - | String | host |
| - | boolean | runnable |
| + | buildGUI() | |
| + | createMenuBar() | |
| + | Client(int port, String host) | |
| - | receiveMsg() | |
| - | sendMsg(String msg) | |
| + | menuSelected(MenuEvent e) | |
| + | run() | |
| + | diffieHellman() | |

The server class creates a server that accepts clients for a chatroom. Once a client connects the diffieHellman() function exchanges keys for the creation of a symmetric key. The server uses the ClientHandler class to manage sending messages to and from each individual client

|  |  |  |
| --- | --- | --- |
| Server | | |
| - | JTextArea | msgArea |
| - | DataOutputStream | toClient |
| - | DataInputStream | fromClient |
| - | ArrayList<ClientHandler> | clients |
| - | int | prime |
| - | int | generator |
| - | int | randNum |
| - | ArrayList<ClientHandler> | keys |
| + | diffieHellman() | |
| + | buildGUI() | |
| + | Server(int port) | |

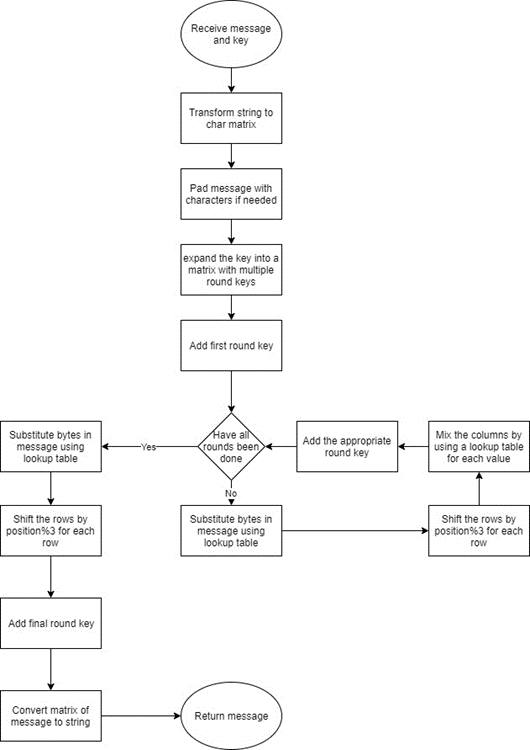
The clienthandler class is used to create a thread for the client on the server’s end. Once a clienthandler receives a message, it will then loop through all Clienthandlers in the arraylist the server has. The message is then send encrypted to each client. If a ClientHandler receives an exit request, it’ll notify other clients that the user is leaving and then use closeClient(int client) to leave and close all necessary components to close the client.



# Encryption

AES encryption is used throughout this project. The key is randomly generated upon connecting to a server with the Diffie-Hellman algorithm. The encryption and decryption algorithms return an encrypted or decrypted string.

**AES program flow:**



NOTE: The program flow for the decryption is simply the previous flowchart in reverse

The AES class is used to encrypt information. Using the bytesub, shiftRows, mixColumns and addKey functions the information is encrypted to maintain confidentiality. Various lookup tables are used, especially for the mixColumns, to protect from timing attacks.



Graphical User Interface

**Login:**

A new user is not allowed to create their own account; it must be done by their custodian directly onto the database.

Once a user put in their username, password, role, and branch, the information is sent to its associated function to authenticate the user.

If the user is allowed to access the system, the user will be sent to Session.

|  |  |  |
| --- | --- | --- |
| **Login** | | |
| **-** | Final String[] |
| **-** | Final String[] | Branches |
| **-** | JTextField | tfUsername |
| **-** | JPasswordField | pfPassword |
| **-** | JComboBox | cbRole |
| **-** | JComboBox | cbBranch |
| **-** | JLabel | lbUsername |
| **-** | JLabel | lbPassword |
| **-** | JButton | btnLogin |
| **+** | Login() |  |
| **+** | boolean | Authenticate (String, String, String) |
| **+** | void | ActionPerfomed(ActionEvent) |

**Session:**

In Session, a user can choose to either join one of ten open chatrooms, or access the file system.

|  |  |  |
| --- | --- | --- |
| **Session** | | |
| **-** | JList | list |
| **-** | DefaultListModel | chatRoomList |
| **-** | JLabel | lbUserInfo |
| **-** | JButton | btnLogin |
| **-** | JButton | btnAccess |
| **-** | JButton | btnLogOut |
| **-** | String | Username, branch, role |
| **-** | Final int | CHATROOM\_PORT |
| **+** |  | Session (String, int, String) |
| **+** | void | ValueChanged(ListSelectionEvent) |
| **+** | void | ActionPerformed(ActionEvent) |

**ABCFileSystem:**

File System session is made up of 2 parts: a file selection part that a user chooses from either their machine, or from the company file system, and an activity part that a user either upload or download in accordance with the selected file.

|  |  |  |
| --- | --- | --- |
| ABCFileSystem | | |
| - | JFileChooser | fc |
| - | JButton | btnFind |
| - | JButton | btnChoose |
| - | JButton | btnSave |
| - | JButton | btnUpload |
| - | JLabel | lbFile |
| - | String | UserName, userRole, userBranch |
| + | void | buildGUI |
| + |  | ABCFileSystem (String, String, String) |
| + | void | ActionPerformed (ActionEvent) |
| - | String | GetAccessiblePath() |

# Filesystem

**Branches and roles:**

There are three branches in the company: Vancouver(Head Quarter), Germany, New York.

Each branch will have three roles: Custodian, Salesperson, Warehouse.

**Accessibility:**

All staffs are limited to access the File System during their working hours: 8AM – 5PM in their own time zone.

Their accessible paths are determined by their role and branch.

Custodians will have access to every files and folders in the File System, and the other two roles will have access to their own folders in their own branch.

**Upload:**

User can upload a file from their system to the company file system by clicking on “Select a file to upload” and “Upload”. The selected file will be copied into their accessible folder.

**Download:**

User can download a file from the company file system to their system by clicking on “Find a file from FS” and “Download”. The selected file will be copied into their “Downloads” folder.

Special Mechanisms

**Time Zone**:

Access to the chatrooms and file system is controlled through a function based off of time. The database’s time zone is set to PST. A query is sent to the server that will retrieve the current time in a 24-hour format. Based off the user’s branch, which is retrieved during the login process, the access times will be different. Only users with the role Salesperson have time restrictions as they have the ability to edit the database.

If the branch is Vancouver, access time is between 8am to 5pm.

If the branch is Germany, access time is between 11pm and 8 am.

If the branch is New York, access time is between 5am to 2 pm.

**Audit Logging:**

All functions are logged. It does not matter if the function failed or not, it will be logged. Only Custodians can view these logs.

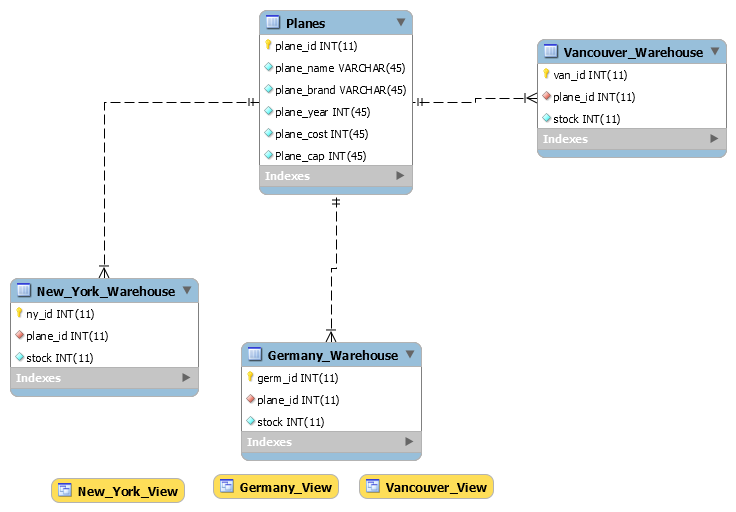
**SQL Injection Prevention:**

JDBC has a function in where it makes it harder for SQL injection to happen. It uses placeholders and the variables are to be inserted “manually”, while at the same time encapsulating them in quotation marks.

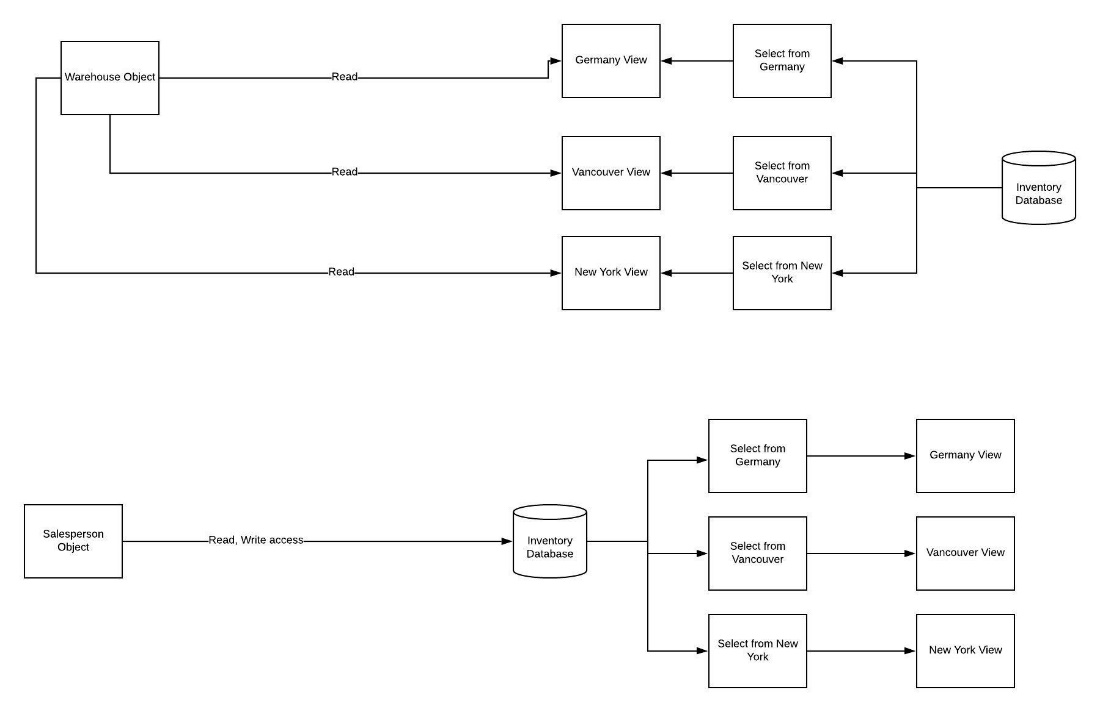
**Multilevel Database:**

The Inventory database is multi-level.

Warehouse is only able to see views based of SELECT statements querying the Planes Table and corresponding branch warehouse. The only fields that are returned for Warehouse are plane\_name (alias Plane Name), plane\_brand (alias Brand), plane\_year, (alias Plane Year), and stock(alias Stock, from Warehouse).



Salesperson is able to access all the tables in the Inventory database and is able to write to it as well. All edits to the corresponding warehouse will be shown in its corresponding view.



**Accounts:**

**Custodian**

Username: root

Password: root

**Salesperson**

Username:testGermSale

Password:testGermSale

Branch: Germany

Username: testVanSale

Password:testVanSale

Branch: Vancouver