Cory DioGuardi

Parker Johnson

Reese Jones

Louden Yandow

Principles of Data Management

Phase 2 Document

Software Application Description

The user interface for our application is, at least for now, on the command line. It will display a way for the user to log in to our system. Depending on who the user is, a menu is displayed that shows them all of the commands they can use. Our system will be solely command based. We discussed how we wanted to format the user commands, and ultimately decided that commands should be preceded with a flag (such as '-s'). Following the flag, the user will input various terms until their command is complete. At this point, the command is sent to our "Command Manager", where the command is converted to a SQL Query. This is also where we check the security of the user and the command they want to use.

For example, since customers, dealerships, and system administrators can access the application, it would not make sense for a customer to be able to query information on other customers, or for a dealership to query information on other dealerships. Therefore, if a user attempts to access data they should not access, we do not let that happen and let them try a different command.

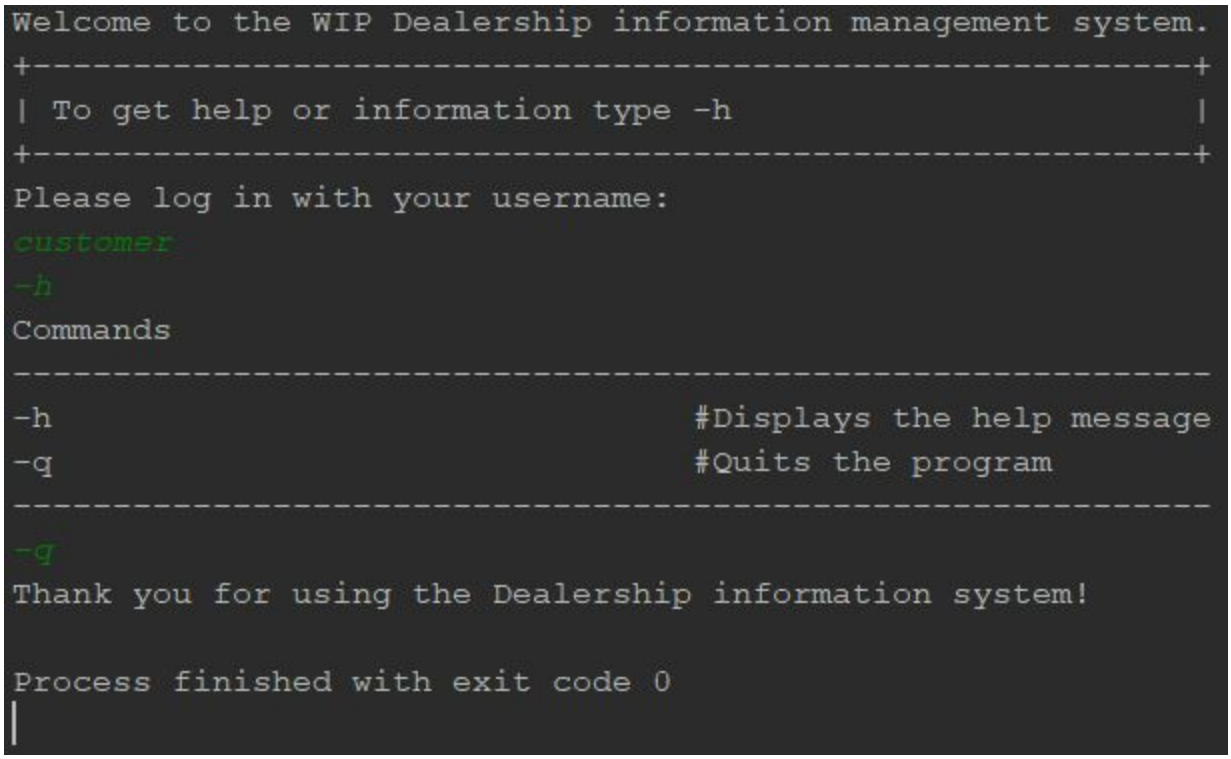
Once we convert a command to a SQL query in our “Query Builder”, we directly query the database for the information. The result from the query is then sent back to the user’s command line, where we will display it in a user-friendly and informative format.

As mentioned, different users have access to different portions of data. A customer (someone who has bought or will buy a car) will be able to view the inventory of different dealerships. This includes seeing the cars in their inventory, the price, and any other important attributes that the cars have. There will also be a command for a user to select and purchase a car to simulate the additions and deletions from a dealership’s database as customers buy cars.

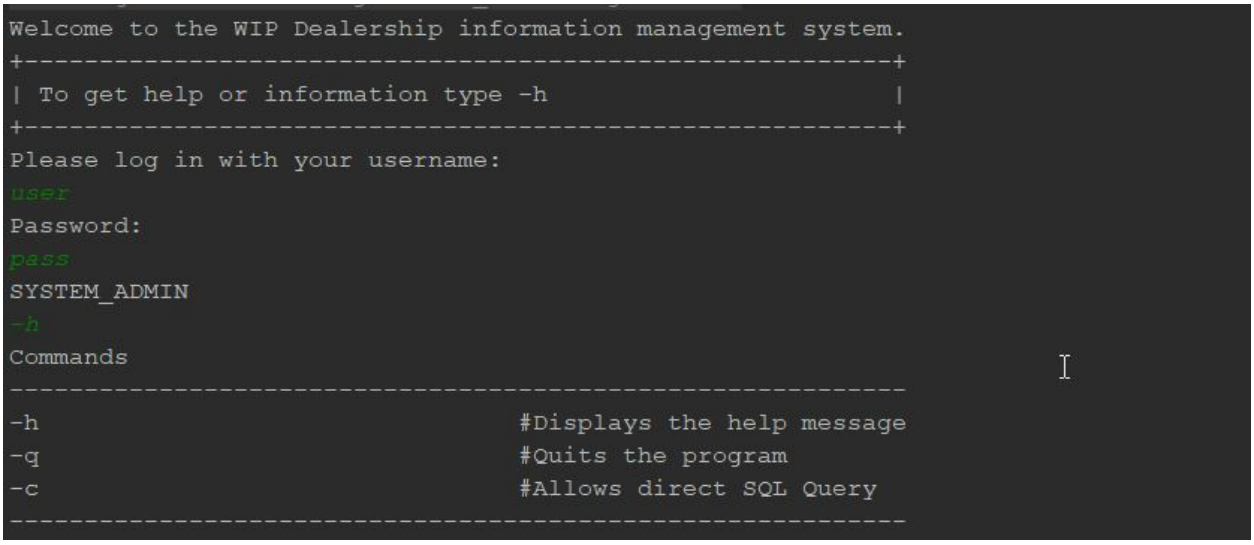
Dealerships will be able to see their own inventory, and can also access sales information. This means that a dealership can determine the date of a sale, the customer (and customer information), the sale price, the vehicle sold, and all other information associated with the sale. It is important to note that we will only allow a dealership to see their own sales. A dealership will not be able to access sale information from other dealerships. This functionality allows complete access to all information associated with the dealership that is using the application, but not other dealerships.

The highest tier user we refer to as “System Administrator”. This user has access to the entire database. Because of the nature of this type of user, we also determined that we want the user to be able to input direct SQL queries into our application to retrieve data. This user can be thought of as the overarching automobile company that distributes its vehicles to different dealerships. Thus, our “company” can keep track of all of its vehicles.

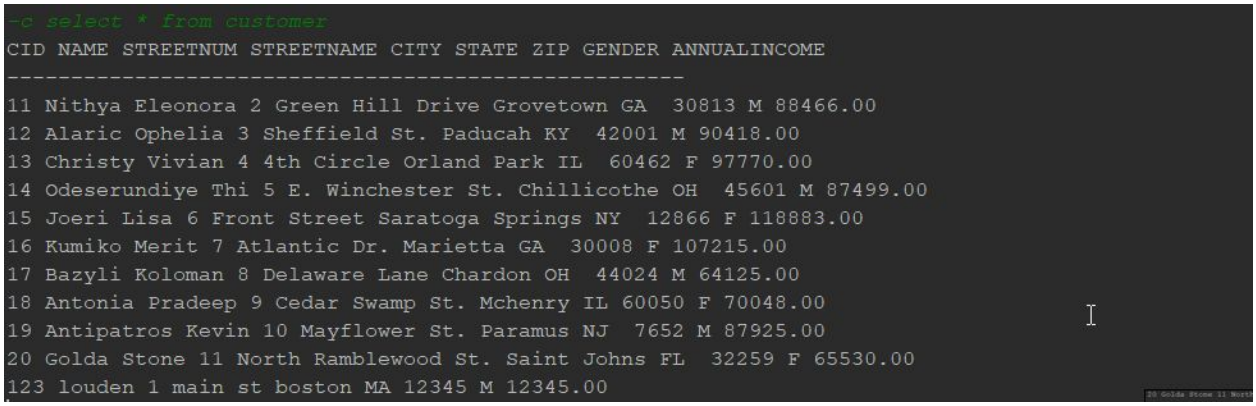
UI Screenshots and Descriptions



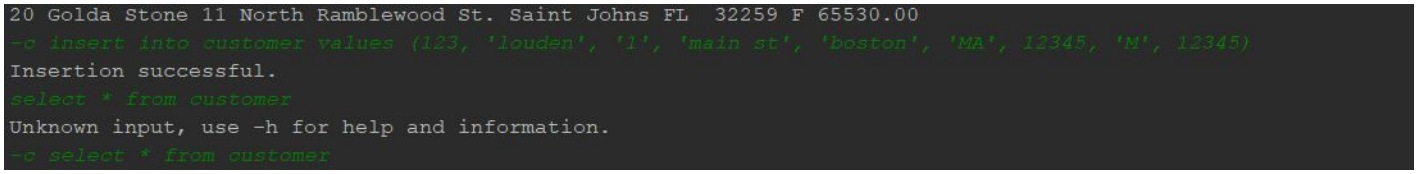
Here we see a simple example of the login system. The user is prompted to log in using their username (NOTE: this screenshot was taken earlier than we asked for passwords). The user is also informed of a way to retrieve usage information about the program, and how we display this information is also shown.



This screenshot shows the same process, but also exhibits the differentiation for types of users. Here, a user with system administrator privileges has logged in. Their ‘help’ command displays a new command unavailable to customers and dealership managers: a “-c” followed by a direct SQL query. This gives the system administrator level users greater access to the database.



This screenshot shows how a system administrator can utilize SQL queries to access data. The user in this case has selected every entry in the ‘Customer’ table. We use this SQL command to directly retrieve the query’s result, and display it. It is important to note that this current display is very basic, and not user-friendly. Going forward, we will endeavor to improve the display of query results.



This is another example of how we allow system administrator level users to directly use SQL queries. Here, the user is inserting a new customer into the customer table.

```
-c select * from customer
CID NAME STREETNUM STREETNAME CITY STATE ZIP GENDER ANNUALINCOME
-----------------------------------------------------
11 Nithya Eleonora 2 Green Hill Drive Grovetown GA  30813 M 88466.00
12 Alaric Ophelia 3 Sheffield St. Paducah KY  42001 M 90418.00
13 Christy Vivian 4 4th Circle Orland Park IL  60462 F 97770.00
14 Odeserundiye Thi 5 E. Winchester St. Chillicothe OH  45601 M 87499.00
15 Joeri Lisa 6 Front Street Saratoga Springs NY  12866 F 118883.00
16 Kumiko Merit 7 Atlantic Dr. Marietta GA  30008 F 107215.00
17 Bazyli Koloman 8 Delaware Lane Chardon OH  44024 M 64125.00
18 Antonia Pradeep 9 Cedar Swamp St. Mchenry IL 60050 F 70048.00
19 Antipatros Kevin 10 Mayflower St. Paramus NJ  7652 M 87925.00
20 Golda Stone 11 North Ramblewood St. Saint Johns FL  32259 F 65530.00
123 louden 1 main st boston MA 12345 M 12345.00
```

This last screenshot is just for proof that the SQL commands do truly function as intended. The user has inserted the new customer, and then selects all customers from the Customer table. The new customer entry is visible at the bottom.

SQL Sample

```
private static void initialize(){
    // The SQL query to create the customer table
    String createCustomer =
            "create table if not exists customer(" +
            "CID numeric(5) not null," +
            "name varchar(20) not null," +
            "streetNum varchar(5)," +
            "streetName varchar(20)," +
            "city varchar(20)," +
            "state varchar(20)," +
            "zip numeric(5)," +
            "gender char(1)," +
            "annualIncome numeric(10,2)," +
            "primary key (CID))";
    // The SQL query to create the vehicle table
    String createVehicle =
            "create table if not exists vehicle(" +
            "VIN varchar(17) not null," +
            "color varchar(20)," +
            "transmission varchar(20)," +
            "engine varchar(20)," +
            "primary key (VIN))";
```

The above screenshot shows how we utilize SQL commands to create tables. This only shows the creation of two tables, but it is enough to show how the rest will be made. Essentially, we use the SQL create table command. To accomplish this, we have a String and just append all of the attributes the table will hold.

We store the command for each table in a String. We then store all of these Strings in an array. Afterwards, we iterate through this array, and execute each command one by one in the database. This method is how we create all of our tables.

This is handled automatically, and each table is hard-coded to be what we want it to be. However, as shown above, the system administrator level users can issue true SQL commands to the application. Through this, it is possible to create more tables without hard-coding them.

The system administrator "-c" commands are assumed to be correctly formatted SQL queries/commands. Because of this, when such a command is entered, we do not interpret it and pass it straight to the database for execution. If they happen to input an incorrect SQL command, then the program errors.

For users that are not system administrators, their commands will be converted to SQL commands. We have yet to fully implement all the user commands, but they will all follow the same pattern: user inputs their command, we take the entire string after the flag and convert it to the correct SQL command, execute that command, and display the result.