Technical Report

Kansas City Electric

Henry Fundenberger

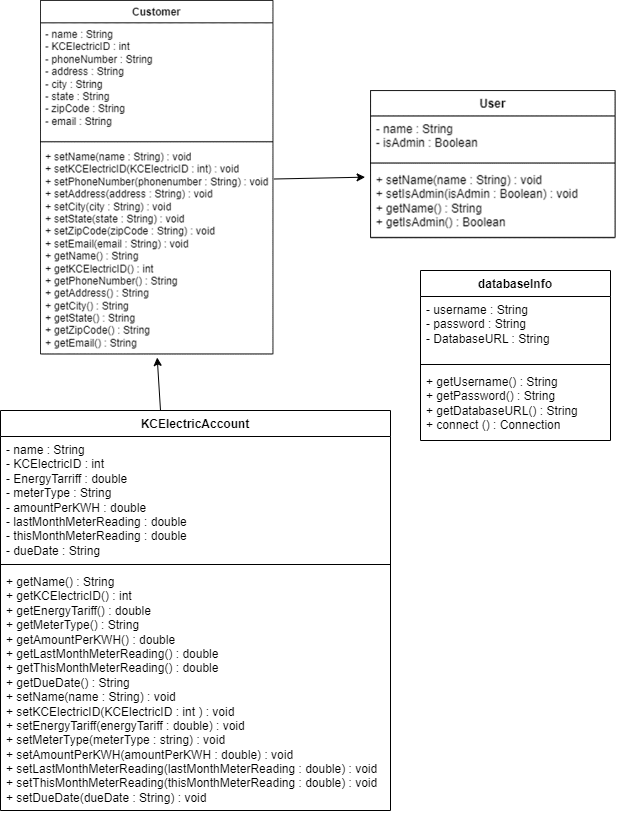
Dr. Adu Baffour

**Link to Demonstration Video:**

**Username And Password for Relevant Accounts**

* Login Screen
  + Username: admin
  + Password: password

**UML Class Diagrams**

****

**Features Implemented**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Implemented (Full Partial)** | **Any Comments** |
| Login Screen | Full |  |
| Main Dashboard | Full | Main navigation page “Home” in the navbar |
| Customer Dashboard | Full |  |
| Record New Customer | Full | Has entry verification, needs appropriate error messages |
| Update Customer | Full | Warns user if customer does not exist |
| Remove Customer | Full | Warns user if customer does not exists, assumes you are searching by first + last name |
| Find Customer | Full | Assumes you are searching by whole name rather than part of a name, i.e first + last name |
| Display Customer Energy Usage | Full | No access if customer doesn’t exist |
| Manage Payments Dashboard | Full |  |
| View Payment Status | Full | Need to fix dollar formatting |
| Make a Payment | Full | Need to fix dollar formatting |
| Create Invoice | Partial | No email functionality |

**Explanation of Main Code**

* Login window is broken up between two files, loginMenu and loginBody. The login body extends JPanel and holds the labels and text entry boxes, as well as getters. The login menu file extends JFrame. Adds the login body to its center and adds a button to the south side of the screen. When this button is clicked it takes the data currently in the login text fields for username and password. Utilizing the database info class, it connects to the database and constructs a query to see if that username and password combination exists in MySQL. If it does, we login if it’s an admin account otherwise we prompt the user they have entered invalid credentials and they can try again.
* The maindashboard java file is what’s loaded right after the login screen and can be navigated to at any time the navigation menu and selecting home. The whole purpose of this page is to navigate to sub directories that will have separate purposes. This is done by utilizing JButtons as well as action listeners for each button to load a new Gui constructor in a different file and then destroy the current window. The current directory options are “Manage Customer Accounts”, “Display customer energy usage”, “Create customer invoices”, “Manage Payments”. These buttons line up with the basic system requirements and are how I have chosen to access the pages that implement the requirements for this program.
* The customer dashboard is the first option from the main dashboards choices of buttons. This is where we have the choice to create, edit, or remove a customer. These are the only three buttons on this page, as well with a menu bar that allows the user to go back to the homepage. Each button has an action listener that takes it to the appropriate page.
* The record new customer page asks for a customers basic information, something that needs to be collected and can be different between every user. This is done through two files the newCustomer file which extends the JFrame and acts like the host for our gui page, and the newCustomerInfo file which extends the JPanel and holds all of the labels and text entry boxes. After the user fills out all the information, if they don’t a message will pop up telling them to complete the fields, this is used along with the databaseInfo class to connect to our database and createa a new customer entity record with the provided data, as well as a corresponding KCElectricAccount entity record with the same name and id. Since they are new users this auto fills with new data. The database connection is closed, and the user is prompted with a success message where they can return back to the customer dashboard.
* When accessing the update customer button from the customer dashboard you are taken to the generic find customer search file. This gui is used all through out the program any time you need to find a instance of a specific customer from the database, if the customer doesn’t exist this part will tell you. After entering the name and hitting submit, if the customer exists then you will be taken to a screen similar to the record new customer page, however it will auto populate with the searched for customers information, and everything is editable except for the name. Once you have changed all you need to change, hit submit and if everything is formatted correctly you will be given a confirmation message and taken back to the customer dashboard.
* The final option in the customer dashboard is the remove customer, this page is simple. All it does is prompt the user to a name using a text entry box and after hitting submit, using the databaseinfo class it checks to see if there is a user with that name in the database and if there is we just send a delete query. A confirmation is sent to the user and they are redirected back to the customer dashboard.
* Now back on the main dashboard we can select the “Display Customer Energy Usage” button, doing this will bring up the normal FindCustomer gui passing in a special prompt to let it know where it’s supposed to direct to after entering a name. After entering a valid name and hitting enter, we can see all of the KC Electric account stats for the specific user. This is done through two files, the file displayCsutomerEnergySearch extends the JFrame while displayCustomerEnergyFields extends JPanel, and holds all the labels and text entries for the specific data. We also make the fields uneditable here as it’s just to display information. We display their meter readings, their KWH price, an energy tarrif amount, their meter type, their account number, when they payed last, and their current bill due. There is also a home button that will take you back to the main dashboard.
* Upon clicking the create customer invoices button on the homepage we are once again taken to a find customer page, once entering a valid name a customer object is created and passed to the appropriate next constructor. In this case being the viewInvoice file. This kind of acts as a way to display both the customer information as well as creating a KCElectric account object in the code and getting appropriate information from the database. We display the customers information i.e their name, phone number, address, email, and account number. Then we show the calculations on how the total bill is calculated. Then by using the info in the KCElectricAccount object we display the appropriate data that corresponds with the calculations. After showing all calculations the user can either go home through the navbar, or the button on the bottom left, or they can export a PDF that can be emailed to the customer. This PDF is titled the customer's name + invoice and is ready to be dragged into the email.
* The last button on the main dashboard is the manage payments button. Once clicked a new gui is loaded up the managePaymentsDash file is loaded on this dashboard there are three buttons, one for viewing payment status, one for making a payment, and one to return home, as well as the normal navigation menu bar.
* When clicking the view payment status button we are navigated to the FindCustomer gui once again with a search keyword passed in to navigate to the correct page. After typing in a valid customer name we are taken to the viewPaymentStatus gui which shows the calculated bill, from the database values, and displaying if it has been payed or not yet. There are no other functions on this page except going back or the navigation menu bar. The go back button redirects the user back to the manage payments tab.
* When clicking the make a payment button, we are sent to the FindCustomer with a special message passed through to load the correct next page. After putting in a valid name a customer object and KCElectricAccount object are created and are used in the payCurrentBill gui. This gui displays the current balance due, calculated from values stored in the database, and gives the user the option to go back or pay now, upon clicking pay now, the database is updated to make last meter read equal to current meter read so they are equal. This updates the values across the whole application.

**Screenshots of System**

Login Page

Graphical user interface, application

Description automatically generated

Main Dashboard

Graphical user interface, application

Description automatically generated

Manage Customers Dashboard

Graphical user interface, application

Description automatically generated

Creating A New Customer

Graphical user interface, application

Description automatically generated

Updating a Customer

Graphical user interface, application, table

Description automatically generated

Remove Customer with Confirmation

Graphical user interface, application

Description automatically generated

Remove Customer with Invalid Entry

Graphical user interface, text, application

Description automatically generated

General Find Customer Search

Graphical user interface, application

Description automatically generated

Display Customer Energy Usage

Graphical user interface, application, Word

Description automatically generated

Manage Payments Dashboard

**Graphical user interface, application

Description automatically generated**

View Payment Status before Paying

Graphical user interface, application, Word

Description automatically generated

Pay Current Bill

Graphical user interface

Description automatically generated

View Payment Status after Paying

Graphical user interface, application, Word

Description automatically generated

Viewing Invoices

Table

Description automatically generated