Address Book Final Report

The database that this application uses is composed of 4 separate tables. The first is titled “names” and stores the personID, which links all the other tables to name, the first name, and the last name of the entry. The table titled “phoneNumbers” stores the personID, phoneID, and phone number. “emailAddressess” holds the personID, emailID, and email address. Lastly, “addresses” holds the personID, addressID, address1, address2, city, state, and zip code. The entity relationship diagram for the used database is pictured below.

Diagram

Description automatically generated

This program required many different thought processes as to how to go about implementing all the required features of the application. The process I used for the development of this application was similar to the Extreme Programming process. I would spend much time planning and thinking about how to implement a feature, then I would come up with a design and code it into the program (implementation). I would then test to make sure the feature was performing as intended and tested all possible ways and combinations the user could try to use this feature. If there were any bugs or unintended results from the coded segment, I would pinpoint where the issue was and plan a way to solve it. This process was repeated for every aspect of the program, from the GUI to each action the user could press. This repetitive process followed along the steps of the software development life cycle (SDLC). Within the planning phase of the application, I also created a DFD to help visualize the information that was required and transferred from the user to the GUI and then to the database, while also considering what was send back to the user. The DFD can be viewed below.

Diagram

Description automatically generated

When the Addressbook() constructor is first called, the GUI and all its components are generated and organized. The GUI is composed of two separate panes. The first, titled “Address Book Operations”, contains the text fields of where the information is entered and the action buttons (new, insert, search, update, delete and print). The second pane, titled “Search Results”, displays the information of all the Address Book entries or the individuals that are searched. After the GUI is built, the defaultSearch() method is called and performed. This method is used to display the updated database. This method performs an SQL statement to retrieve all information from all entries of all four tables using inner join with their unique personID’s. This results in the information of all the entries to be displayed, allowing the user to view their entire Address Book. After this, the program waits for an action button to be pressed. When the “New” button is pressed, all the text fields will be cleared. When the “Insert” button is pressed, as long as there is at least a first or last name entered, the information will be stored as a new entry in the Address Book database by using an SQL statement. The defaultSearch() method will then be performed after the insertion to show the user the updated database. When the “Search” button is pressed, the program will search the database by either first name, last name, or both (prioritizing both). The “Search Results” pane will then only display the information of the entries that pertain the information that the user searched for. After this, the fillForm() method is called, which will set the text fields on the “Address Book Operations” pane to the information pertaining to the searched entry. This makes the “Update” and “Delete” functions easier to use for the user. Only after a “Search” is performed, can the user use the “Update” and “Delete” actions. The program stores the information of the top search result but will reset this information if another action is performed. If the user changes the information in the text field after a “Search” then presses “Update” this information will be updated within the database for that entry, after a confirmation from the user. If the user presses “Delete” after a “Search” then the entry will be deleted from the database, after a confirmation from the user. The last action is the “Print” operation. This operation will open a print dialog box for the user and will print whatever information is currently being displayed in the “Search Results” pane. All the previously mentioned information is what composes the “Addressbook” class. Lastly, the program contains a “WindowDestroyer” class to close the program when the “X” is clicked. The GUI can be seen below.

Table

Description automatically generated with low confidence

While building this application there were not any major difficulties. Database connection and the use of SQL statements, using the variables that were gathered from the application took some time to figure out, as I did not have a lot of experience using them. Figuring out the syntax for the statements and the use of the Connection and Statement objects also required some reading to get them functioning as intended.

This program offers the user more than just a pane to search and insert entries into their Address Book. The program allows the user to visualize the information the Address Book contains, making the experience easier for the user. This gives the user confidence of knowing what the database holds and not just guessing to see if their Address Book holds the information of different entries they are trying to search. A con for this application is that it must run on the computer that the connection is tied to within the code. This prevents the use of the application on other computers, without going into the code and changing the directory.

In the future, the text fields could be aligned to look neater. When there is any interaction (clicking, scrolling, etc.) with the “Search Results” pane the pane will reset to the information that was being displayed when the program was originally started. This prevented the use of a scrollable pane that would be nice to have if there were more than 25 entries. Although this will reset the pane, it is not an issue with how the operations currently work.

In conclusion, this project was a great test and introduction to the SDLC and required a lot more than just writing the code for an application. This project required much thought provoking plans in order to build a functioning result. From the organization of the GUI to the connection of the database and structure of the SQL statements, the entire process allowed for the implementation and practice of various techniques used in the production of an information system application.