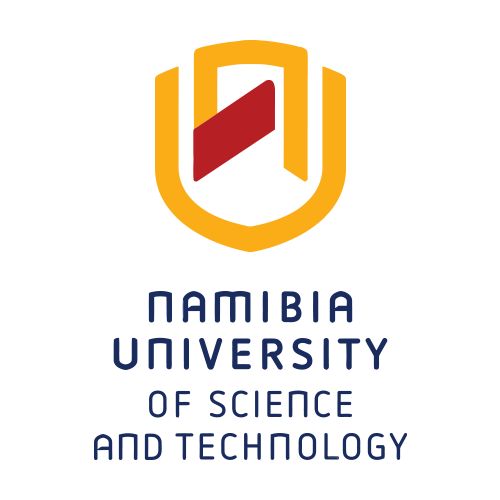
**Data Structures and Algorithms**

**Group Project: Phonebook Application**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SN** | **Name** | **Student Number** | **Specialization** | **Role played** | **Mark/100 (for lectures use only)** |
| 1 | Courtney Van Wyk | 224085182 | Informatics | Documentation and management |  |
| 2 | Amaru-Jay October | 224028782 | Software engineering | Contact insertion and update |  |
| 3 | Lazarus Petrus | 224009176 | Software engineering | Contact deletion |  |
| 4 | Letuuya Amungulu | 224026186 | Cyber security | Contact search |  |
| 5 | Opomanga Kavela | 224064827 | Software engineering | Contact saving and loading |  |
| 6 | Hilifavali Nikodemus | 224082000 | Cyber security | Contact sort and display |  |

**Documentation**

**Phonebook Application**

**Section A**

**1. Overview:**

This is a simple phonebook application built using Java for a Namibian telecommunications company. The app allows users to add, search, delete, update, sort, and manage contacts with basic linear data structures. It also supports saving and loading contacts from a file.

**2. Features:**

* **Add Contact:** Allows users to add new contacts to the phonebook.
* **Search Contact:** Enables users to search for a contact by name, using matching.
* **Delete Contact:** Removes a contact from the phonebook based on the provided name.
* **Update Contact:** Updates the phone number of an existing contact.
* **Display All Contacts:** Lists all contacts stored in the phonebook.
* **Sort Contacts:** Sorts the contact list alphabetically by name.
* **Save to File:** Saves all contacts to a text file.
* **Load from File:** Loads contacts from the text file back into the phonebook.

**3. Data Structures:**

* **ArrayList:** The application uses an Array List to store the contact list. This array allows for efficient contact management, including adding, deleting, and updating contacts.

**4. Classes:**

* **Contact:**
  + Represents a single contact with a name and phone number.
* **Phonebook (JFrame):**
  + The main class extending JFrame that provides the graphical user interface (GUI) for managing contacts.
  + Contains buttons and text fields for user interaction.

**5. Methods and Functions:**

1. **insertContact():**
   * Handles adding new contacts after validating input fields for the correct name and phone number format.
2. **searchContact():**
   * Implements search functionality based on the provided name. Finds and displays matching contacts.
3. **deleteContact():**
   * Deletes a contact from the phonebook after searching for it by name.
4. **updateContact():**
   * Updates the phone number of an existing contact by searching for the name.
5. **displayAllContacts():**
   * Displays all the contacts stored in the phonebook in a text area
6. **sortContacts():**
   * Sorts the contacts alphabetically by name
7. **saveContactsToFile():**
   * Writes the contact list to a file (contacts.txt) to store the data persistently.
8. **loadContactsFromFile():**
   * Reads the contact list from a file (contacts.txt) to restore data.
9. **exitApp():**
   * Exits the application after confirming with the user.

**6. User Interface:**

* The GUI consists of three sections:
  + **Input Panel:** For entering contact names and phone numbers.
  + **Display Panel:** A text area for showing all contacts.
  + **Operation Panel:** Contains buttons for search, delete, update, display, sort, save, load, and exit operations.

**7. Error Handling:**

* The app ensures that invalid data (missing or incorrect phone numbers) is not accepted and displays appropriate error messages.
* It also checks for empty contact lists and provides messages when necessary.

**8. Limitations:**

* The phone number is limited to 10 digits.

**PSEUDOCODE**

**Main Menu:**

1. Start the application

2. Display the main window with the following options:

- Add Contact

- Search Contact

- Delete Contact

- Update Contact

- Display All Contacts

- Sort Contacts

- Save Contacts to File

- Load Contacts from File

- Exit Application

3. user input and perform the corresponding action.

**Insert Contact:**

1. Prompt the user to enter the contact's name and phone number.

2. Validate that the name and phone number are not empty.

3. Validate the phone number to ensure it contains exactly 10 digits.

4. Check if a contact with the same name already exists:

- If it does, show an error message "Contact already exists".

- If not, proceed.

5. Add the new contact to the list.

6. Display a success message.

7. Clear the input fields.

**Search Contact:**

1. Prompt the user to enter the name to search for.

2. Validate that the name is not empty.

3. Search for contacts whose names start with the input string.

- If one or more contacts are found, display the matching contacts.

- If no matching contact is found, show a message "Contact not found".

**Delete Contact:**

1. Prompt the user to enter the name of the contact to delete.

2. Validate that the name is not empty.

3. Search for the contact by name.

- If found, remove the contact from the list and show a success message.

- If not found, display an error message "Contact not found".

**Update Contact:**

1. Prompt the user to enter the name of the contact to update.

2. Validate that the name is not empty.

3. Search for the contact by name.

- If found, prompt the user to enter a new phone number.

- Validate the new phone number (must be 10 digits).

- Update the contact's phone number and show a success message.

- If the new phone number is invalid, show an error message.

- If the contact is not found, display "Contact not found".

**Display All Contacts:**

1. Check if the contact list is empty:

- If empty, display "Phonebook is empty".

- If not empty, display the name and phone number of all contacts.

**Sort Contacts:**

1. Sort the contact list alphabetically by name.

2. Display a success message "Contacts sorted alphabetically"

**Save Contacts to File:**

1. Open a text file named "contacts.txt".

2. Write each contact's name and phone number to the file, separated by a comma.

3. Close the file.

4. Display a success message "Contacts saved to file".

**Load Contacts from File:**

1. Open the "contacts.txt" file.

2. Read each line, splitting the contact name and phone number by commas.

3. Add each contact to the list.

4. Close the file.

5. Display a success message "Contacts loaded from file".

**Exit Application:**

1. Display a confirmation dialog asking, "Are you sure you want to exit?".

2. If the user confirms, close the application.

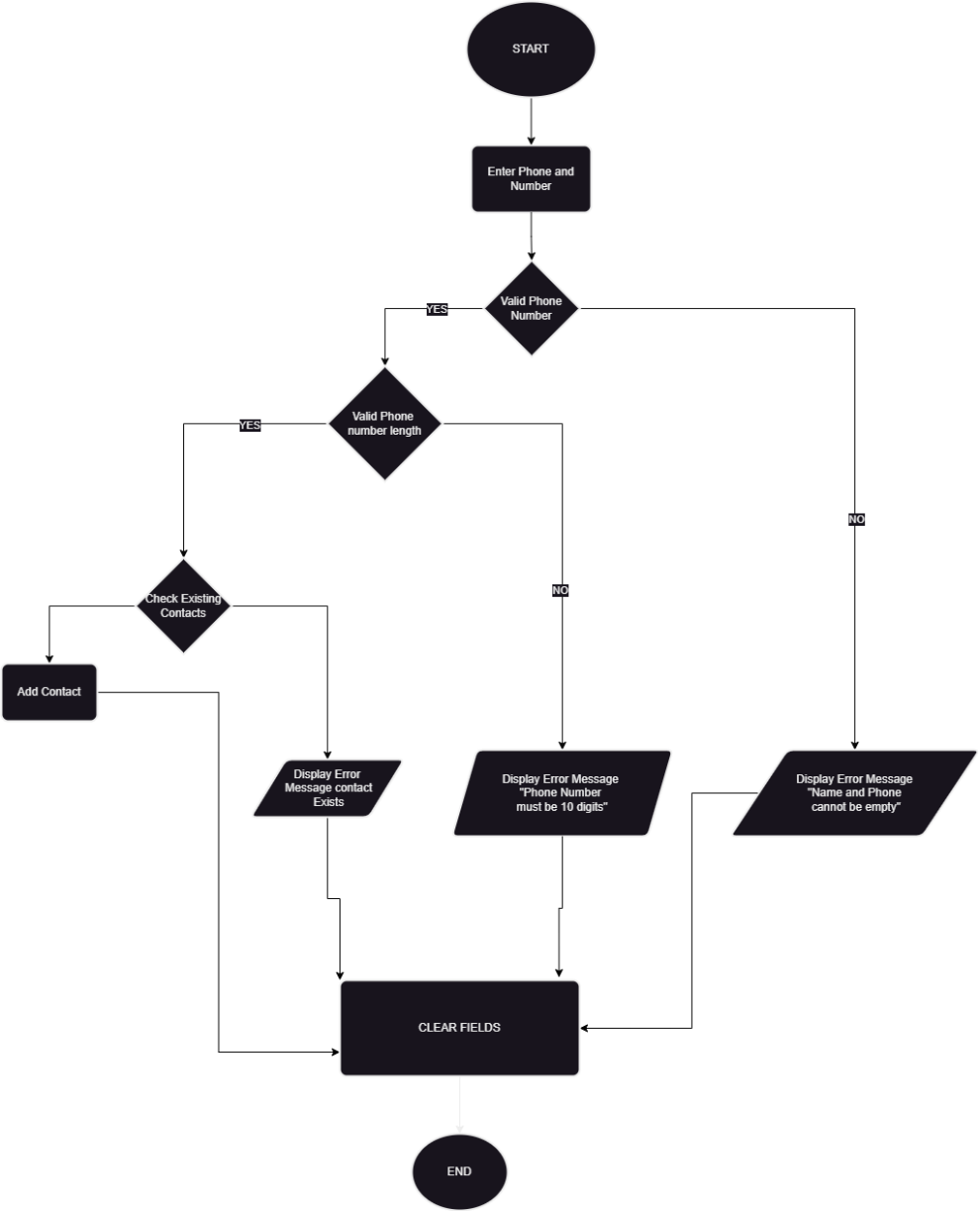
**FLOWCHARTS**

**Main Menu**

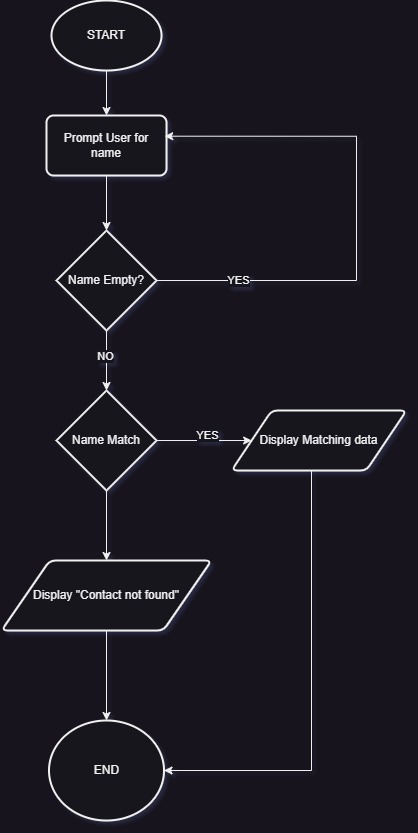
A screen shot of a diagram

Description automatically generated

**Insert Contact**



**Search Contact:**

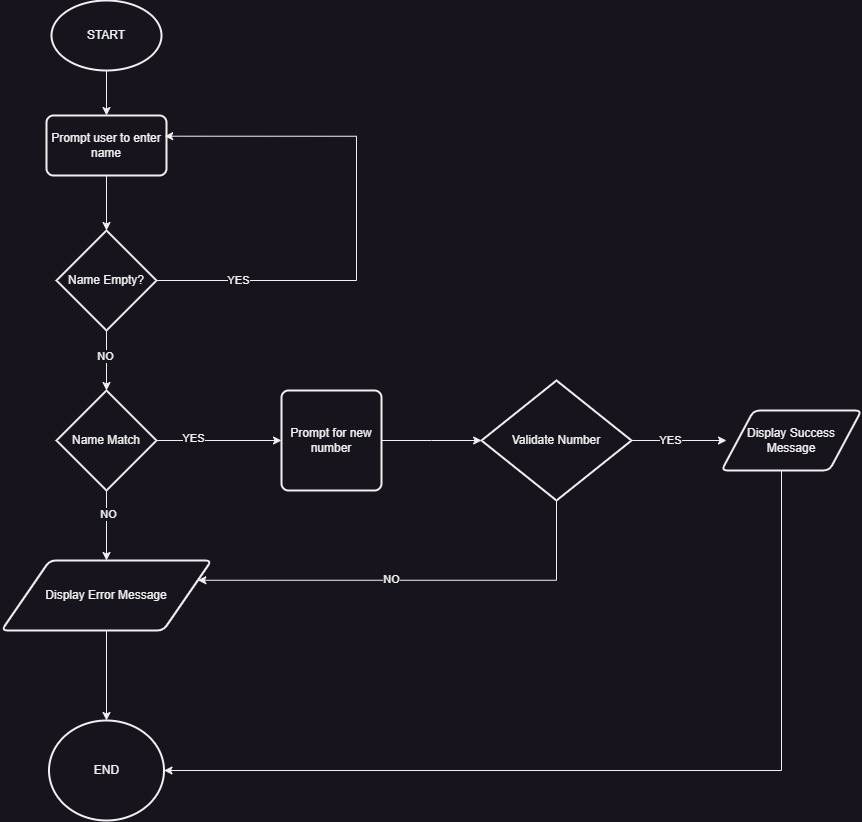


**Delete Contact:**

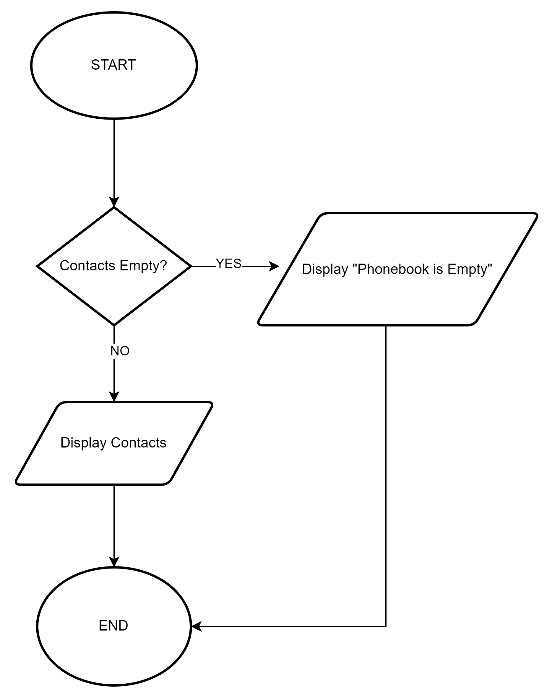
A diagram of a computer flowchart

Description automatically generated

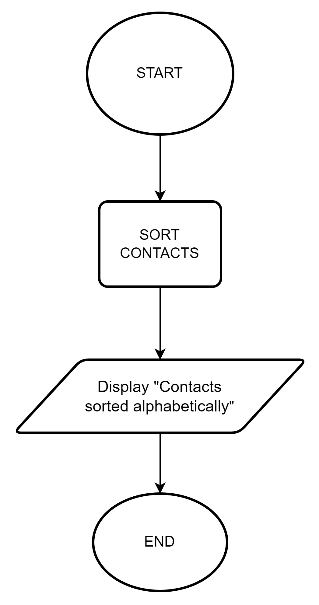
**Update Contact:**



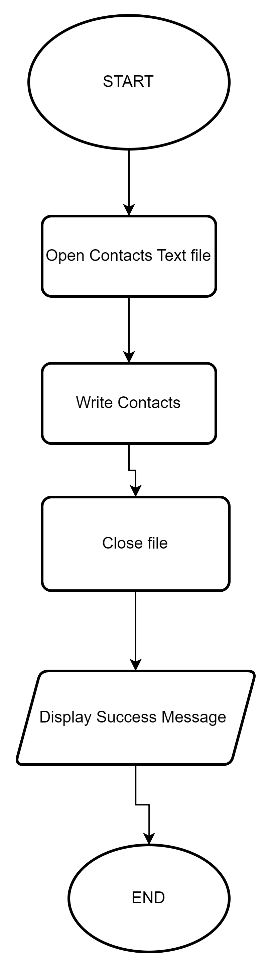
**Display All Contacts:**



**Sort Contacts:**



**Save Contacts to File:**



**Exit Application:**

