**A black background with a black square

Description automatically generated with medium confidence**

**DSA521S DATA STRUCTURES AND ALGORITHMS 1, SEMESTER 2, 2024**

**GROUP ASSIGNMENT PRESENTATION: SCORESHEET AND PROJECT DESCRIPTION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Name** | **Student Number** | **Role played in the project** | **Mark over 100% (for lecturers use only)** |
| 1 | Anselmo Martins | 224065955 | Code And GUI |  |
| 2 | Paulo Jorge Manuel | 223007935 |  |
| 3 | Natalia Aristofel | 224075438 | General Documentation + flowchart |  |
| 4 | Matheus Isak Ndeyapune | 223019348 | General Documentation + Readme file |  |
| 5 | Jason de Sousa | 223134783 | Pseudocode |  |
| 6 | Elsa Kumwimba | 223037222 | Flowchart |  |

**MODE OF STUDY:** **FULL-TIME**

**Name of Team Leader (TL): Anselmo Martins**

**TITLE OF PROJECT:**

Intercell

**DATE:**

09/10/2024

1. **DESCRIPTION OF PROJECT**

Our project mas designed to help a recently introduced telecommunications company known as ‘Intercell’. Our software has the facility to accept full name and phone number for every customer and stores the details of every customer. Employees can search for a customer contact details using their full name. The information in the software can be retrieved easily. The interface is very user-friendly.

1. **FUNCTIONALITIES, SERVICES AND REQUIRMENTS (FSR) OF THE SOFTWARE**

**Module**

The app can be broken down into the following main components or modules:

1. **Contact**: Handles all the contact-related operations (add, delete, update, search, store and display).
2. **User Interface**: Handles the graphical interface of the application such as buttons, windows, etc

**Functions for the Application:**

1. **Main Window Functions**:
   * **insertContact()**: Opens a form to add a new contact to the phonebook.
   * **searchContact()**: Searches the contact list based on input from the search bar.
   * **showAllContacts()**: Fetches and displays all contacts in a new file.
   * **deleteContact()**: Deletes a selected contact from the phonebook.
   * **updateContact()**: Opens a form to update details of a selected contact.
2. **ContactModule**:
   * **add\_contact(contact)**: Adds a new contact to the phonebook database.
   * **remove\_contact(fullName)**: Removes a contact from the phonebook.
   * **update\_contact(fullName, newPhoneNumber)**: Updates the information of an existing contact.
   * **get\_all\_contacts()**: Returns a list of all contacts.
   * **search\_contact(query)**: Returns contacts that match the search query.

|  |  |
| --- | --- |
| **MODULE** | **DESCRIPTION/ROLE** |
| Contact | Stores and retrieves customer details and handles all contact-related operations |
| Registration | Handles the graphical interface of the application such as buttons, windows, labels, frame and ensures that a user can directly communicate with the system to store and retrieve customer details |

1. **Pseudocode**

Start

Display launchPage

Display Main Menu

Prompt for user choice

Get user choice

IF choice == "Edit" THEN

Prompt for fullName and newPhoneNumber

Get fullName and newPhoneNumber

IF fullName isEmpty THEN

Display "Please enter name"

ELSE IF contact exists THEN

phoneNumber = newPhoneNumber

ELSE Display "Contact not found"

ENDIF

ENDIF

ELSE IF choice == "Search" THEN

Prompt for fullName

Get fullName

IF fullName isEmpty THEN

Display "Please enter name"

ELSE IF contact exists THEN

Display contact details

ELSE Display "Contact not found"

ENDIF

ENDIF

ELSE IF choice == "Add" THEN

Prompt for fullName and phoneNumber

Get fullName and phoneNumber

IF fullName OR phoneNumber isEmpty THEN

Display "Please enter name and phone number"

ELSE Add contact with fullName and phoneNumber

ENDIF

ELSE IF choice == "Delete" THEN

Prompt for fullName

Get fullName

IF fullName isEmpty THEN

Display "Please enter name"

ELSE IF contact exists THEN

Remove contact with fullName

ELSE Display "Contact not found"

ENDIF

ENDIF

ELSE IF choice == "Show All" THEN

IF contacts list is empty THEN

Display "No contacts available"

ELSE

Display all contacts

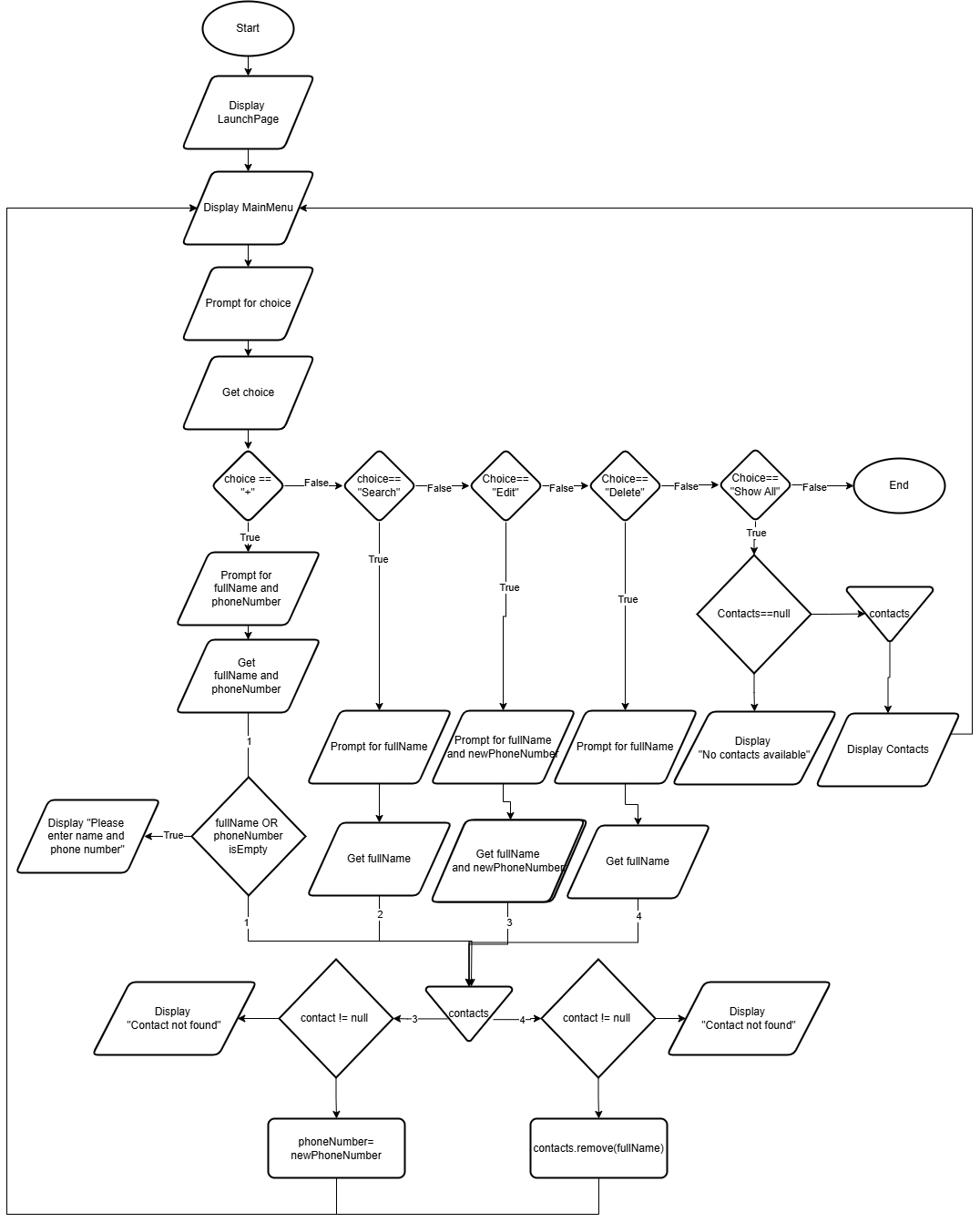
ENDIF

END IF

DISPLAY Main Menu

END

1. **Flowchart**



1. **Readme doc**

**Overview:**

This Java application serves as a simple phone book, allowing users to add, search, edit, and delete contacts. It utilizes Swing for the graphical user interface (GUI) and implements basic functionalities for managing contact information.

**Features:**

Add Contacts: Users can add new contacts with a full name and phone number. Search Contacts: Users can search for contacts by name. Edit Contacts: Users can update existing contact information. Delete Contacts: Users can remove contacts from the phone book. Display All Contacts: Users can view all saved contacts in a separate window.

**Requirements:**

Java Development Kit (JDK) 8 or higher IDE (e.g., IntelliJ IDEA, Eclipse) or command line for running Java applications

**File Structure:**

The project consists of several Java classes:

Main: Entry point of the application. LaunchPage: Initial GUI that launches the main application. MainApp: Main interface for managing contacts. ContactManager: Handles storage and operations on contacts. Contact: Represents a single contact with a name and phone number. AddPerson: GUI for adding new contacts.

**Installation:**

Clone or download the repository containing the Java files. Open the project in your preferred IDE or set up a Java environment in your terminal.

Compile the Java files: bash javac Main.java LaunchPage.java MainApp.java ContactManager.java Contact.java AddPerson.java

Run the application: bash java Main

**Usage Instructions**

Launch the Application: Upon starting, you will see a welcome screen with an "Open" button. Main Menu: After clicking "Open," the main menu will appear with options to add, search, edit, delete, or show all contacts. Adding a Contact: Click the "+" button to open a form where you can enter the contact's name and phone number. Searching for a Contact: Click "Search" and enter the full name of the contact you wish to find. Editing a Contact: Click "Edit" to modify an existing contact's phone number. Deleting a Contact: Click "Delete" to remove a contact by entering their full name. Displaying All Contacts: Click "Show All" to view all saved contacts in a new window.

**Code Explanation:**

Main Class Functionality: The Main class serves as the entry point of the application. It initializes the LaunchPage, which is the first screen users see when they start the application.

LaunchPage Class Purpose: This class creates the initial GUI for the application.

Components:

JFrame: The main window where the welcome message and button are displayed.

JButton: A button labeled "Open" that transitions to the main application.

JLabel: Displays the title "InterCell PhoneBook".

Constructor: Sets up button properties, label styles, and frame settings. The button click event is handled to open the main application and close the welcome screen.

MainApp Class Purpose: This class serves as the main interface for managing contacts. Components:

JFrame: The main menu window where all contact management options are presented. JButtons: Buttons for adding, searching, deleting, editing contacts, and showing all contacts. JLabel: Displays "InterCell" as a title.

Constructor: Initializes ContactManager and sets up GUI elements.

Methods:

initializeComponents(): Configures buttons and labels for interaction.

actionPerformed(): Handles button clicks to invoke respective functionalities like adding or searching contacts.

Contact Class Purpose: Represents an individual contact with attributes for full name and phone number.

Attributes:

fullName: The name of the contact.

phoneNumber: The contact's phone number.

Methods: Getters and setters allow access to these attributes, while toString() provides a string representation for easy display.

ContactManager Class Purpose: Manages a list of contacts with functionalities such as adding, searching, deleting, updating, and sorting contacts.

Attributes: contacts: An ArrayList storing all contact objects.

Methods:

insertContact(): Adds a new contact to the list.

searchContact(): Searches for a contact using binary search; sorts contacts before searching to ensure efficiency.

deleteContact(): Removes a contact by name after locating it with binary search. updateContact(): Updates an existing contact's phone number based on their name. sortContacts(): Sorts contacts using selection sort based on their names.

AddPerson Class Purpose: Provides a GUI for adding a new contact to the contact manager. Components:

JTextField: For entering name and phone number inputs.

JButton: For submitting new contact information.

Constructor: Sets up GUI elements and their layout. It also centers this window relative to the main menu.

ActionListener: Handles submission of contact details, ensuring both fields are filled before adding to the manager.

**Conclusion**

This Java PhoneBook application is designed for ease of use and provides essential functionalities for managing personal contacts effectively. By following this README, users should be able to set up and operate the application without difficulty.

[DSA.github link](https://github.com/ivg-world/DSA-Intercell)