Introduction

This document describes the implementation and functionality of the **Phonebook Application** developed using **JavaFX**. The program allows users to manage their contact list by adding, updating, deleting, searching, and sorting contacts, as well as displaying all contacts in a table view.

The application is designed with a simple **Graphical User Interface (GUI)** that lets users interact with the phonebook features. The contact data consists of a name and a phone number, and all information is displayed in a table for easy access.

Features and Functionalities

The **Phonebook Application** provides the following key functionalities:

1. Add Contact

- Allows users to add new contacts to the phonebook.
- Input fields: Name and Phone Number.
- The new contact is displayed in the TableView.

2. Update Contact

- Enables users to update the name and phone number of an existing contact.
- The user selects a contact from the table, modifies the details in the input fields, and clicks the **Update Contact** button.

3. Delete Contact

- Users can delete a contact from the phonebook by selecting the contact in the TableView and clicking the **Delete Contact** button.

4. Search Contact

- Users can search for a specific contact by name. The contact details will be populated in the input fields if found.

5. Sort Contacts

- Contacts can be sorted alphabetically by name. The sorting is case-insensitive.

6. Display All Contacts

- Users can view all contacts in the phonebook in a tabular format.

Functions

The main functionalities are implemented in separate functions:

- **addButton.setOnAction**: Adds a new contact by taking input from text fields.
- **updateButton.setOnAction**: Updates the selected contact's details.
- **deleteButton.setOnAction**: Deletes the selected contact.
- **searchButton.setOnAction**: Searches for a contact based on the name provided.
- **sortButton.setOnAction**: Sorts the contact list alphabetically.
- **displayButton.setOnAction**: Displays all contacts in the table view.

The Java code of the Project

```
import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.*;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;


public class PhonebookApp extends Application {
   private ObservableList<Contact> contacts;


   public static void main(String[] args) {
      launch(args);
   }
```

```
    @Override
    public void start(Stage primaryStage) {
        contacts = FXCollections.observableArrayList(); // Initialize contact list

        primaryStage.setTitle("Phonebook Application");

        // Create UI components
        Label nameLabel = new Label("Name:");
        TextField nameInput = new TextField();
        Label phoneLabel = new Label("Phone Number:");
        TextField phoneInput = new TextField();

        Button addButton = new Button("Add Contact");
        Button updateButton = new Button("Update Contact");
        Button deleteButton = new Button("Delete Contact");
        Button searchButton = new Button("Search Contact");
        Button sortButton = new Button("Sort Contacts");
        Button displayButton = new Button("Display All Contacts");

        // TableView to display contacts
        TableView<Contact> tableView = new TableView<>();
        TableColumn<Contact, String> nameColumn = new TableColumn<>("Name");
        nameColumn.setCellValueFactory(cellData -> cellData.getValue().nameProperty());
        TableColumn<Contact, String> phoneColumn = new TableColumn<>("Phone Number");
        phoneColumn.setCellValueFactory(cellData ->
cellData.getValue().phoneNumberProperty());
```

```
        tableView.getColumns().addAll(nameColumn, phoneColumn);
        tableView.setItems(contacts);

        // Layout
        GridPane inputGrid = new GridPane();
        inputGrid.setPadding(new Insets(10));
        inputGrid.setVgap(10);
        inputGrid.setHgap(10);
        inputGrid.add(nameLabel, 0, 0);
        inputGrid.add(nameInput, 1, 0);
        inputGrid.add(phoneLabel, 0, 1);
        inputGrid.add(phoneInput, 1, 1);

        HBox buttonBox = new HBox(10);
        buttonBox.getChildren().addAll(addButton, updateButton, deleteButton, searchButton,
sortButton, displayButton);

        VBox layout = new VBox(10);
        layout.setPadding(new Insets(10));
        layout.getChildren().addAll(inputGrid, buttonBox, tableView);

        // Add event handling
        addButton.setOnAction(e -> {
            String name = nameInput.getText();
            String phone = phoneInput.getText();
            if (!name.isEmpty() && !phone.isEmpty()) {
                Contact contact = new Contact(name, phone);
```

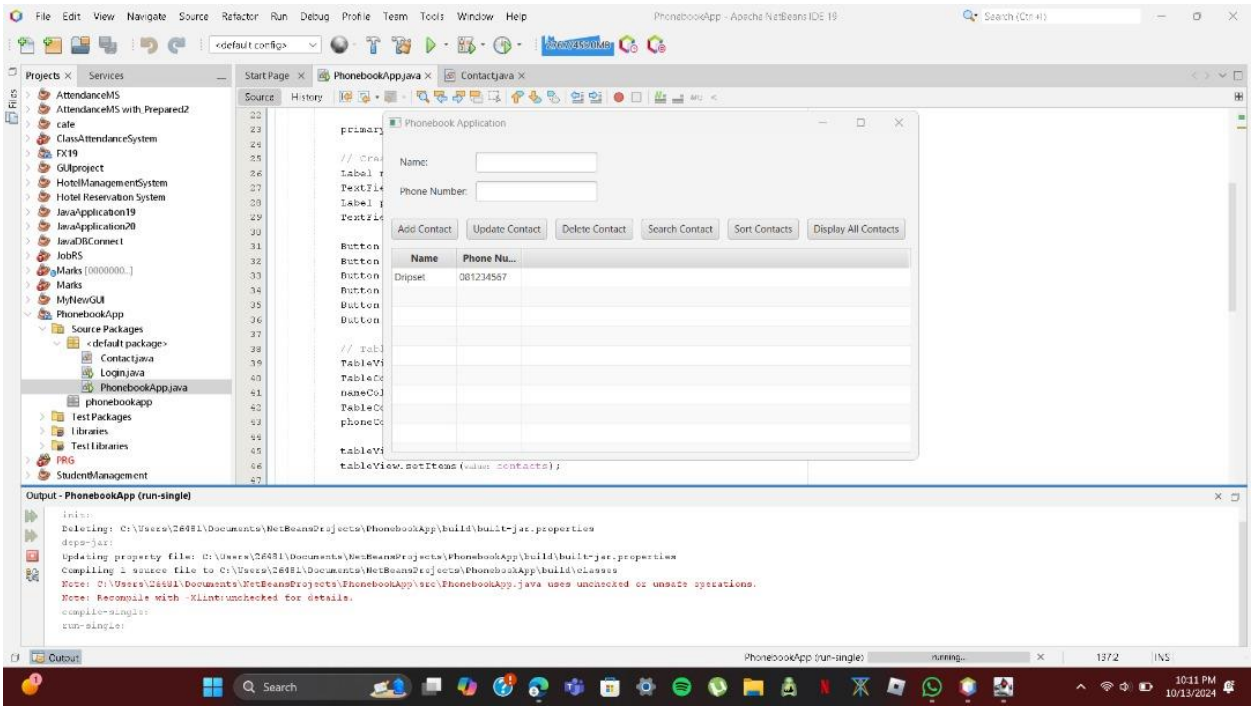
```
        contacts.add(contact);
        clearInputs(nameInput, phoneInput);
    } else {
        showAlert("Please enter both name and phone number.");
    }
});

updateButton.setOnAction(e -> {
    Contact selectedContact = tableView.getSelectionModel().getSelectedItem();
    if (selectedContact != null) {
        selectedContact.setName(nameInput.getText());
        selectedContact.setPhoneNumber(phoneInput.getText());
        tableView.refresh(); // Refresh to show updated info
        clearInputs(nameInput, phoneInput);
    } else {
        showAlert("Please select a contact to update.");
    }
});

deleteButton.setOnAction(e -> {
    Contact selectedContact = tableView.getSelectionModel().getSelectedItem();
    if (selectedContact != null) {
        contacts.remove(selectedContact);
    } else {
        showAlert("Please select a contact to delete.");
    }
});
```

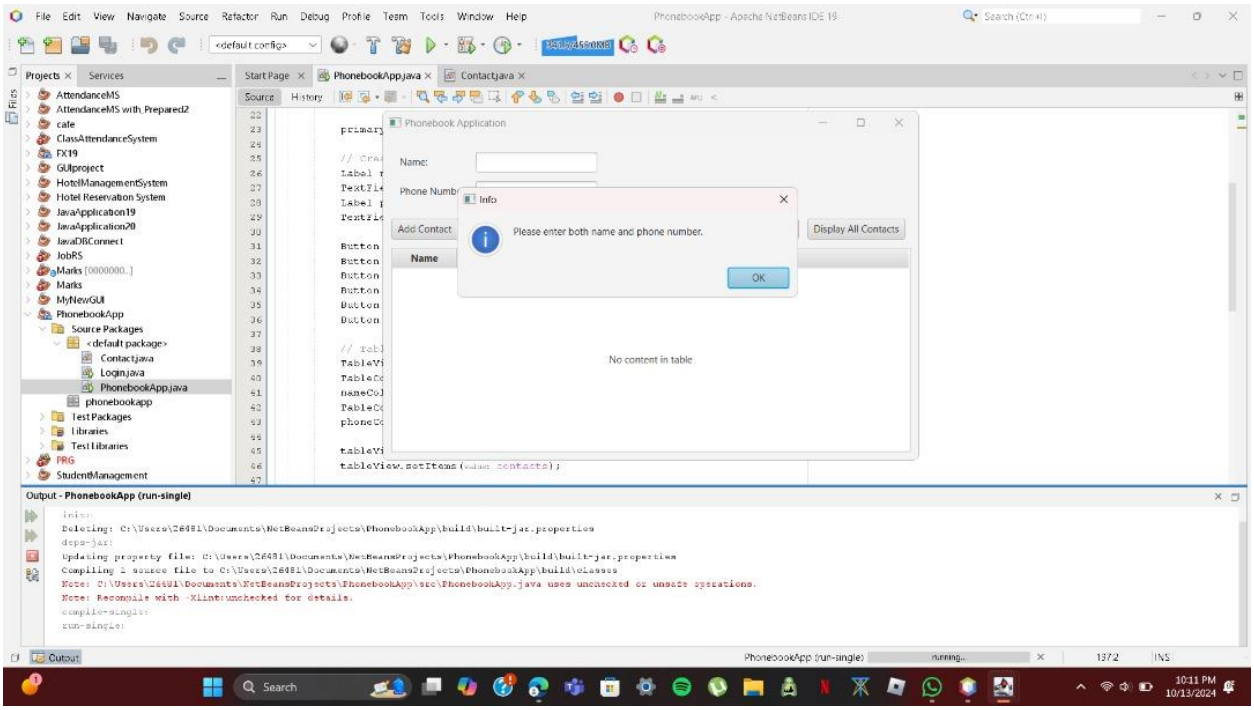
```
    searchButton.setOnAction(e -> {
        String name = nameInput.getText();
        for (Contact contact : contacts) {
            if (contact.getName().equalsIgnoreCase(name)) {
                nameInput.setText(contact.getName());
                phoneInput.setText(contact.getPhoneNumber());
                return;
            }
        }
        showAlert("Contact not found.");
    });

    sortButton.setOnAction(e -> {
        contacts.sort((c1, c2) -> c1.getName().compareToIgnoreCase(c2.getName()));
    });

    displayButton.setOnAction(e -> tableView.setItems(contacts));

    // Set scene and show the stage
    Scene scene = new Scene(layout, 500, 400);
    primaryStage.setScene(scene);
    primaryStage.show();
}

// Clear the input fields
private void clearInputs(TextField nameInput, TextField phoneInput) {
```

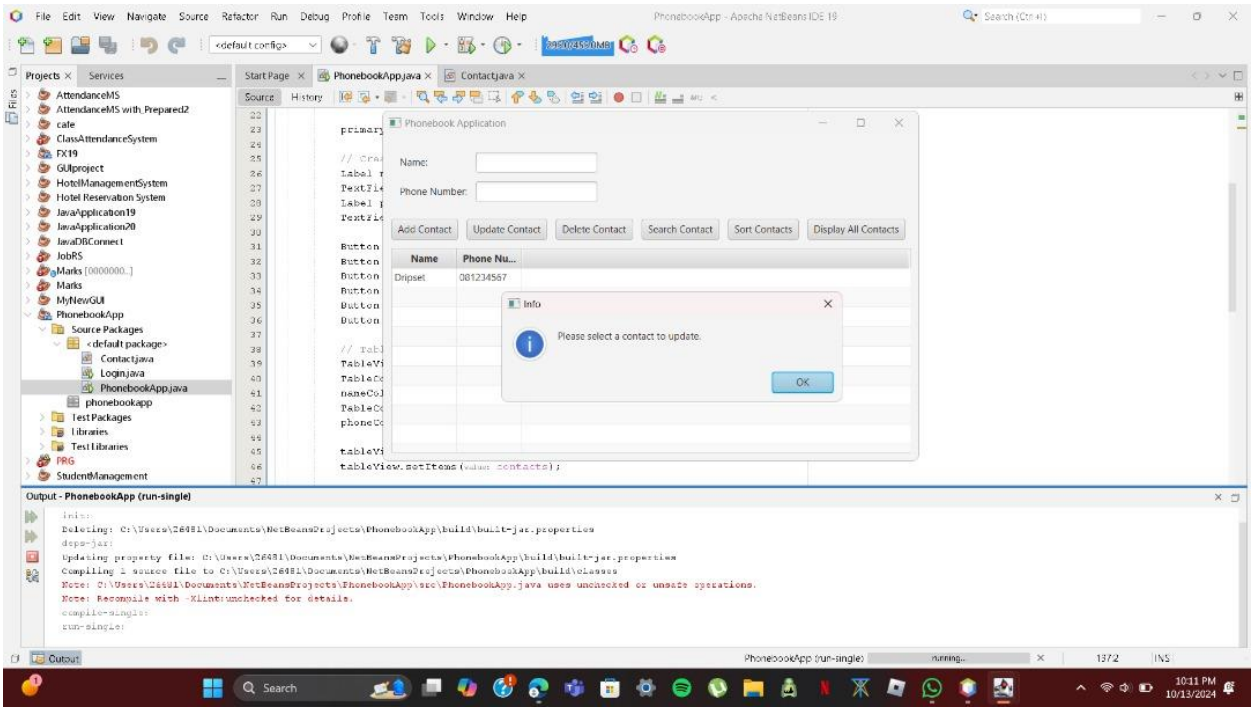
```
        nameInput.clear();
        phoneInput.clear();
    }

    // Display an alert box
    private void showAlert(String message) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Info");
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;

public class Contact {
    private final StringProperty name;
    private final StringProperty phoneNumber;

    public Contact(String name, String phoneNumber) {
        this.name = new SimpleStringProperty(name);
        this.phoneNumber = new SimpleStringProperty(phoneNumber);
    }

    public String getName() {
```

```
    return name.get();
}

public void setName(String name) {
    this.name.set(name);
}

public StringProperty nameProperty() {
    return name;
}

public String getPhoneNumber() {
    return phoneNumber.get();
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber.set(phoneNumber);
}

public StringProperty phoneNumberProperty() {
    return phoneNumber;
}

@Override
public String toString() {
    return "Name: " + name.get() + ", Phone: " + phoneNumber.get();
}
```

}


Phonebook Application
Name:
Phone Number:
Add Contact
Update Contact
Delete Contact
Search Contact
Sort Contacts
Display All Contacts
Name
Phone Nu...
Dripset
081234567
Output - PhonebookApp (run-single)


Phonebook Application
Info
Please enter both name and phone number.
OK
Add Contact
Display All Contacts
Name
No content in table
Output - PhonebookApp (run-single)



Conclusion

The **Phonebook Application** is a simple and efficient contact management tool developed using JavaFX. It provides a well-structured and intuitive GUI for users to perform basic operations such as adding, updating, deleting, searching, sorting, and displaying contacts. The application code is modular, with clear and well-defined functions, making it easy to maintain and extend.