

Name: Ayan Agayeva

Project: Phonebook Management System

Coding Language: Java

Java IDE: Eclipse

Backend Database: DBeaver

YouTube Video: <https://youtu.be/sNjugb1iBLs>

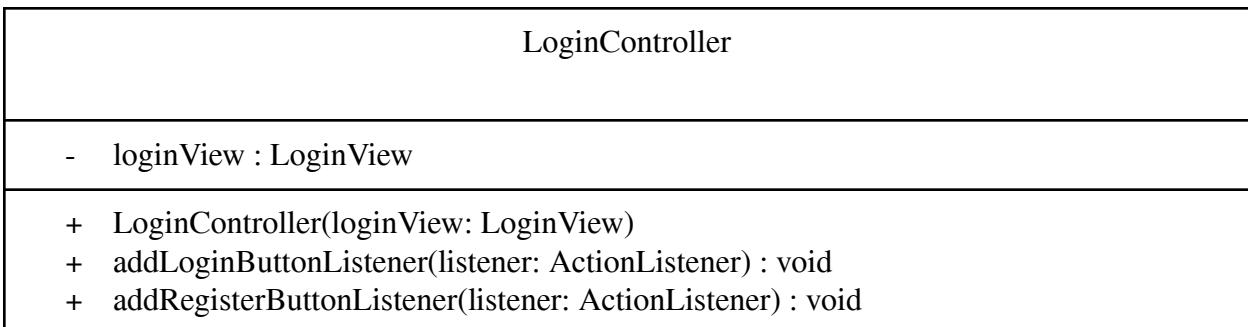
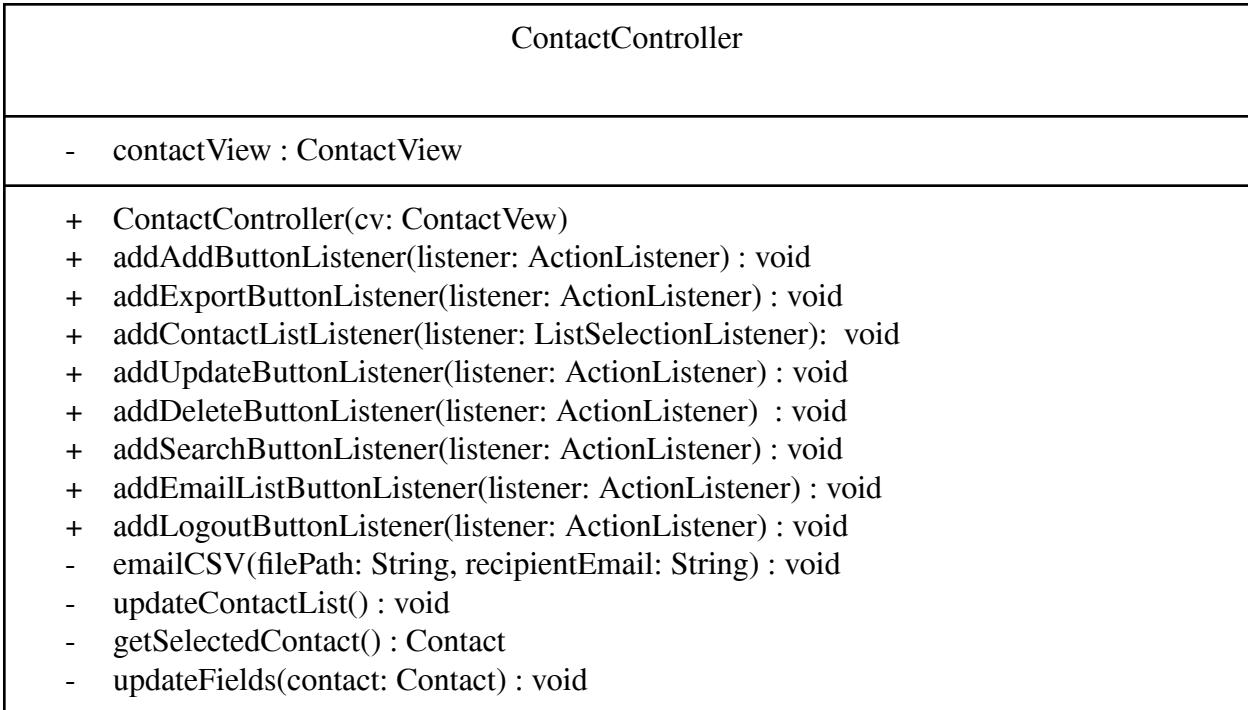
Purpose:

- The purpose of this project is to create and facilitate a phone book. To employ this program, users must register an account. Once their account has been created, they are able to log in to the phone book to access all of its features. Its features include adding contacts (such as their first name, surname, and phone numbers) while being able to update or delete the contacts. They also are given a list of the contacts and can have the list sent to them as a CSV file via email. The list can be searched through as well. Within said list, the contacts are alphabetized and at all times the total number of contacts is displayed.
- Inevitably there are many precautions/code snippets within the program to ensure that the user's inputs are validated and that all of the features/options run smoothly.

Structure:

- I utilized the Model-View-Controller (MVC) architecture and used the code from the class example as a base
- I made many changes particularly to the aesthetics of the program
- The different methods, classes, and overall structure (such as catching errors and creating the components like buttons) followed the same pattern of the code we wrote in class
- I decided to make all my classes more condensed with code rather than creating separate classes for each requirement. I found that if I did this it would be easier to navigate the code instead of having to jump from one file to another. Also, a lot of the classes follow the same pattern/have similar methods

UML Class Diagrams:



- + RegisterController(rv: RegisterView)
- + addRegisterButtonListener(listener: ActionListener) : void
- + addLoginButtonListener(listener: ActionListener) : void

Contact
<ul style="list-style-type: none"> - id : int - firstname : String - lastname : String - phoneNumber : String <ul style="list-style-type: none"> + Contact(id: int, firstname: String, lastname: String, phoneNumber: String) + Contact() + getId() : int + setId(id: int) : void + getFirstname() : String + setFirstname(firstname: String) : void + getLastname(): String + setLastname(lastname: String) : void + getPhoneNumber() : String + setPhoneNumber(phoneNumber: String) : void

ContactDataAccess
<ul style="list-style-type: none"> - connect : Connection <ul style="list-style-type: none"> + ContactDataAccess() + addContact(contact: Contact) : boolean + deleteContact(contact: Contact) : boolean + updateContact(contact: Contact) : boolean + getContacts(): List<Contact> + searchContacts(firstName: String, lastName: String, phoneNumber: String) : List<Contact> + exportContactsToCSV(filePath: String) : void

User

- id : int
 - username : String
 - password : String
-
- + User()
 - + User(id: int, username: String, password: String)
 - + getId() : int
 - + setId(id: int) : void
 - + getUsername() : String
 - + setUsername(username : String) : void
 - + getPassword() : String
 - + setPassword(password : String) : void

UserDataAccess

- connect : Connection
 - currentUserId : int(static)
-
- + UserDataAccess()
 - + registerUser(user: User) : boolean
 - + loginUser(username : String, password : String) : boolean

Util

- URL : String
 - USER : String
 - PASSWORD : String
-
- + getConnection() : Connection
 - + hashPassword(password : String) : String

ContactView

- txtFirstname : JTextField
- txtLastname : JTextField
- txtPhooneNumber : JTextField
- btnAdd : JButton
- btnUpdate : JButton
- btnDelete : JButton
- btnSearch : JButton
- btnLogout : JButton
- btnExport : JButton
- btnEmailList : JButton
- contactList : JList<String>
- listModel : DefaultListModel<String>

- + ContactView()
- + addAddButtonListener(ActionListener listener) : void
- + addUpdateButtonListener(ActionListener listener) : void
- + addDeleteButtonListener(ActionListener listener) : void
- + addSearchButtonListener(ActionListener listener) : void
- + addLogoutButtonListener(ActionListener listener) : void
- + addExportButtonListener(ActionListener listener) : void
- + addContactListListener(ListSelectionListener listener) : void
- + addEmailListButtonListener(ActionListener listener) : void
- + getFirstNameField() : JTextField
- + getFirstName() : String
- + getLastNameField() : JTextField
- + getLastName() : String
- + getPhoneNumberField() : JTextField
- + getPhoneNumber() : String
- + setContactsToModel(contacts: List<Contact>) : void
- + setContactList(contactList: JList<String>) : void
- + getContactList() : JList<String>

LoginView

- usernameField : JTextField
- passwordField : JPasswordField
- loginButton : JButton
- registerButton : JButton
+ LoginView()
+ addLoginButtonListener(listener : ActionListener) : void
+ addRegisterButtonListener(listener : ActionListener) : void
+ getUsername() : String
+ getPassword() : String

RegisterView

- usernameField : JTextField
- passwordField : JPasswordField
- confirmPasswordField : JPasswordField
- loginButton : JButton
- registerButton : JButton
+ RegisterView()
+ addLoginButtonListener(listener : ActionListener) : void
+ addRegisterButtonListener(listener : ActionListener) : void
+ getUsername() : String
+ getPassword() : String
+ getConfirmPassword() : String

Features Implemented

Feature	Class	Purpose
Login functionality	LoginController	Handles user login + manages login view
Registration functionality	RegisterController	Handles user registration + managers register view
Contact management	ContactController	Handles adding, updating, deleting, and exporting contacts
Add a new contact	ContactController	Add a new contact + update list
Remove a contact	ContactController	Remove a contact + update list
Update a contact	ContactController	Update contact info + update list
Alphabetical Contact List	ContactView	Sort the contact list in alphabetical order based on last name
Display total # of contacts	ContactView	Display the total # of contacts in the list at the top
Scroll bar	ContactView	Allows the user to scroll up and down the contact list
Email contact list	ContactController	Method to email contacts list
User management	UserDataAccess	Manage data access for contacts
Database connection	Util	Allow for database connection + password hashing
Login view	LoginView	User interface for user login
Register view	RegisterView	User interface for user

		registration
Contact view	ContactView	User interface for managing contacts
Search contacts	ContactController	Method to search for contacts using any field

Explanations

1. Package: models

Purpose — manage data access and provide data related functionality

- UserDataAccess: Manages the user data access such as login and registration. It utilizes “Util” to establish a database connection and to hash password
- ContactDataAccess: Manages the contact data access such as adding, updating, searching, and deleting the contacts in the database. It also has a method for exporting contacts to CSV

2. Package: views

Purpose — provide user interfaces for all the different parts of the phone book

- LoginView: the user interface for user login including a username field, password field, and buttons for login and registration
- RegisterView: the user interface for user registration including a username, password, and confirm password fields with buttons for registration and returning to the login page
- ContactView: the user interface for managing contacts such as having the contact details, list of contacts, and buttons for adding, searching, updating, deleting, exporting, and emailing
- Each view contains methods for adding listeners to handle user interactions which work simultaneously with their respective controllers

3. Package: controllers

Purpose — handle the relationship between views and models w/ the logic of everything

- LoginController: manages user login functionality by listening for login button events, validating user input with UserDataAccess, and guiding the user to the correct page
- RegisterController: manages user registration by listening for register button events, validating the details the user entered for registration, and establishes a connection with UserDataAccess to register the new users
- ContactController: manages the contacts with adding, deleting, updating, exporting, and emailing while dealing with the user interaction in ContactView
- All the handle logic is handled here

4. Package: MyUtils

Purpose — methods that are used throughout the app

- Util: implements database connection/manages the connection and password hashing

5. Package: testerExample

Purpose — the main method is stored here

- Main: initializes the LoginView and LoginController which starts the user interaction flow

References:

Oracle. (n.d.) *JavaMail*. [Documentation].

<https://docs.oracle.com/en-us/iaas/Content>Email/Reference/javamail.htm>

Stack Overflow. (2013, February 7). Best practices: Sending JavaMail mime multipart emails and Gmail.

<https://stackoverflow.com/questions/14744197/best-practices-sending-javamail-mime-multipart-emails-and-gmail>

GeeksforGeeks. (n.d.). *IO BufferedWriter class methods in Java*.

<https://www.geeksforgeeks.org/io-bufferedwriter-class-methods-java/>

GeeksforGeeks. (n.d.). *Performing database operations in Java SQL: Create, insert, update, delete, select*.

<https://www.geeksforgeeks.org/performing-database-operations-javascript-create-insert-update-delete-select/>

Stack Overflow user. (2014, January 29). *How to perform a search in an SQL database with a Java application*.

<https://stackoverflow.com/questions/21518917/how-to-perform-a-search-in-an-sql-database-with-a-java-application>

Oracle. (n.d.). *How to use GridBagLayout*

<https://docs.oracle.com/javase%2Ftutorial%2Fuiswing%2F%2F/layout/gridbag.html>

Michael71314. (2019, October 14). *Java lesson 22: Inserting images onto the JFrame*.

<https://medium.com/@michael71314/java-lesson-22-inserting-images-onto-the-jframe-a0a0b6540cca>

Baffour, A. (n.d.). *CS 349: Java Programming with Application - Lecture 4: Loops and Fields* [PowerPoint slides]. Canvas

- For the colors of the buttons I just googled java.awt.color colors and went to Google Images to find what I liked

CHATGPT

ChatGPT 3.5 ▾

ChatGPT

In Java, you cannot directly apply HTML or CSS to Java code or text, as Java is a programming language and not a markup language like HTML. However, you can use certain components within Java's GUI libraries such as Swing or JavaFX to display text that is styled using HTML/CSS.

For example, in a Swing application, you can use components like JLabel, JTextPane, or JEditorPane to display HTML-formatted text, including styling such as color and bold. These components support HTML and some CSS styling:

- **JLabel**: You can set the text of a JLabel using an HTML string. For instance, `"**Bold text**"` will display "Bold text" in bold.
- **JTextPane** and **JEditorPane**: These components can render HTML content if you set their content type to `""`. This way, you can use HTML tags within the text to style it, including color and bold.

In JavaFX, you can use components like Label, Text, or TextFlow to style text. These components allow for CSS styling, so you can use methods like `setStyle` to apply CSS styles directly to the text.

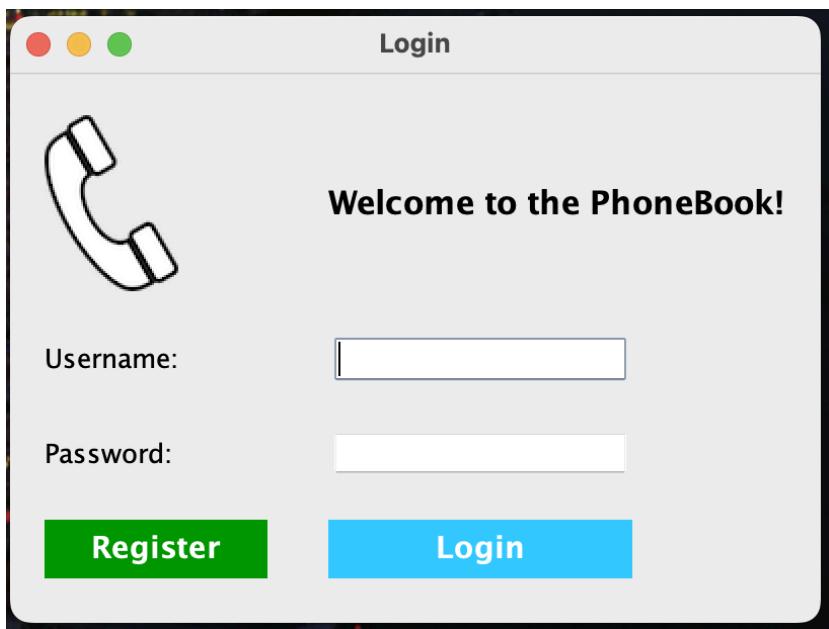
While you cannot directly use HTML/CSS to format text in Java code, you can use these UI components to display styled text as part of a graphical user interface. This allows you to achieve effects like colored and bold text within your Java application.

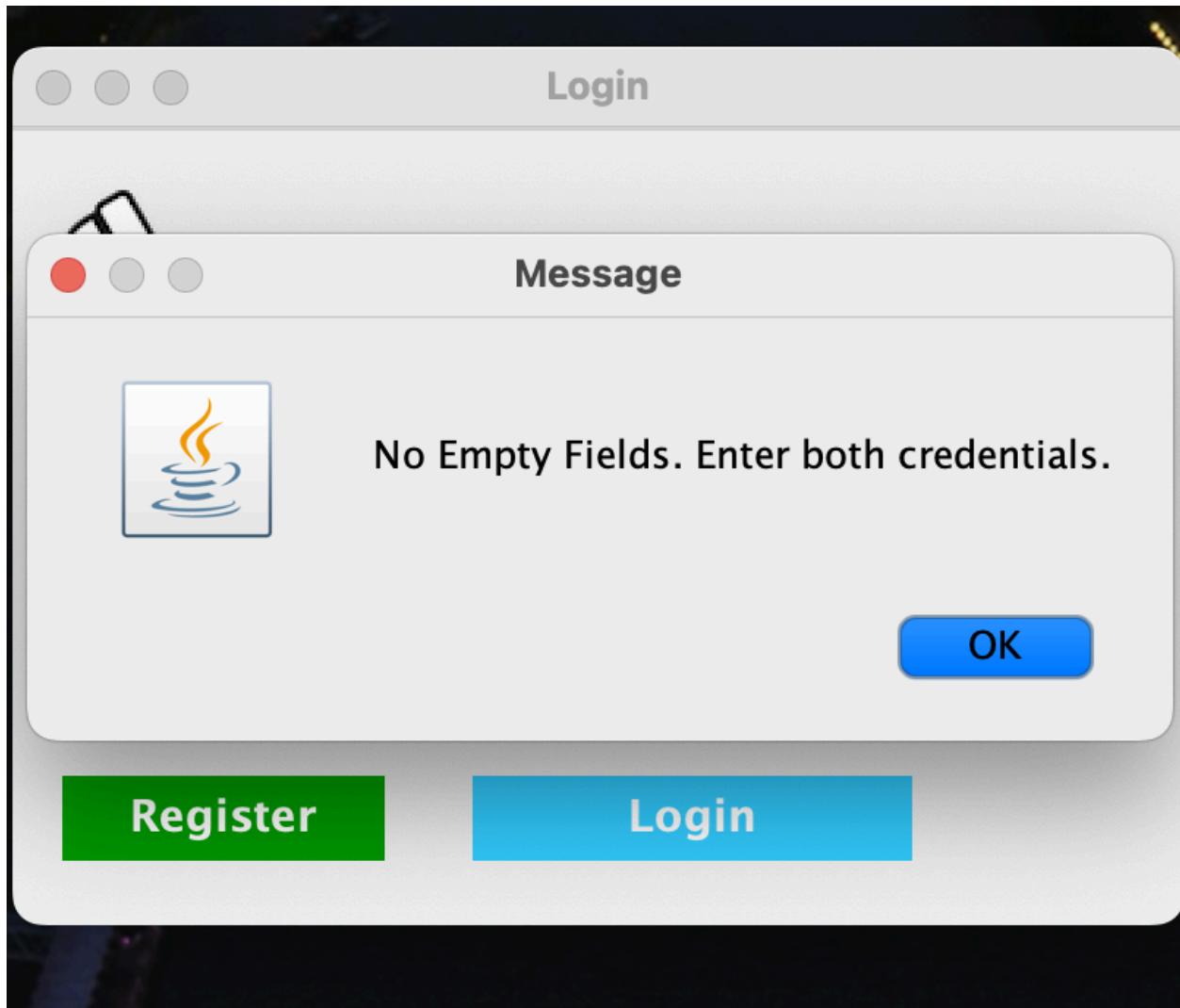
Message ChatGPT

ChatGPT can make mistakes. Consider checking important information.



Screenshots:





Register

Register!

Username:

Password:

Confirm Password:

Register

Go back to login

Register



Register!

Username

Password

Confirm

Register

Go back to login

Message



No Empty Fields. Fill out all 3.

OK

Phonebook

First Name:

Last Name:

Phone Number:

Contact List

Total # Contacts: 1
Ayla Agayeva - 3124567823

Phonebook

First Name:	Max
Last Name:	Verstappen
Phone Number:	7654328765

Add Update

Contact List

Total # Contacts: 1
Ayla Agayeva - 3124567823

Message

Contact added successfully

OK

Search Delete Export Contact List Email Contact List Logout

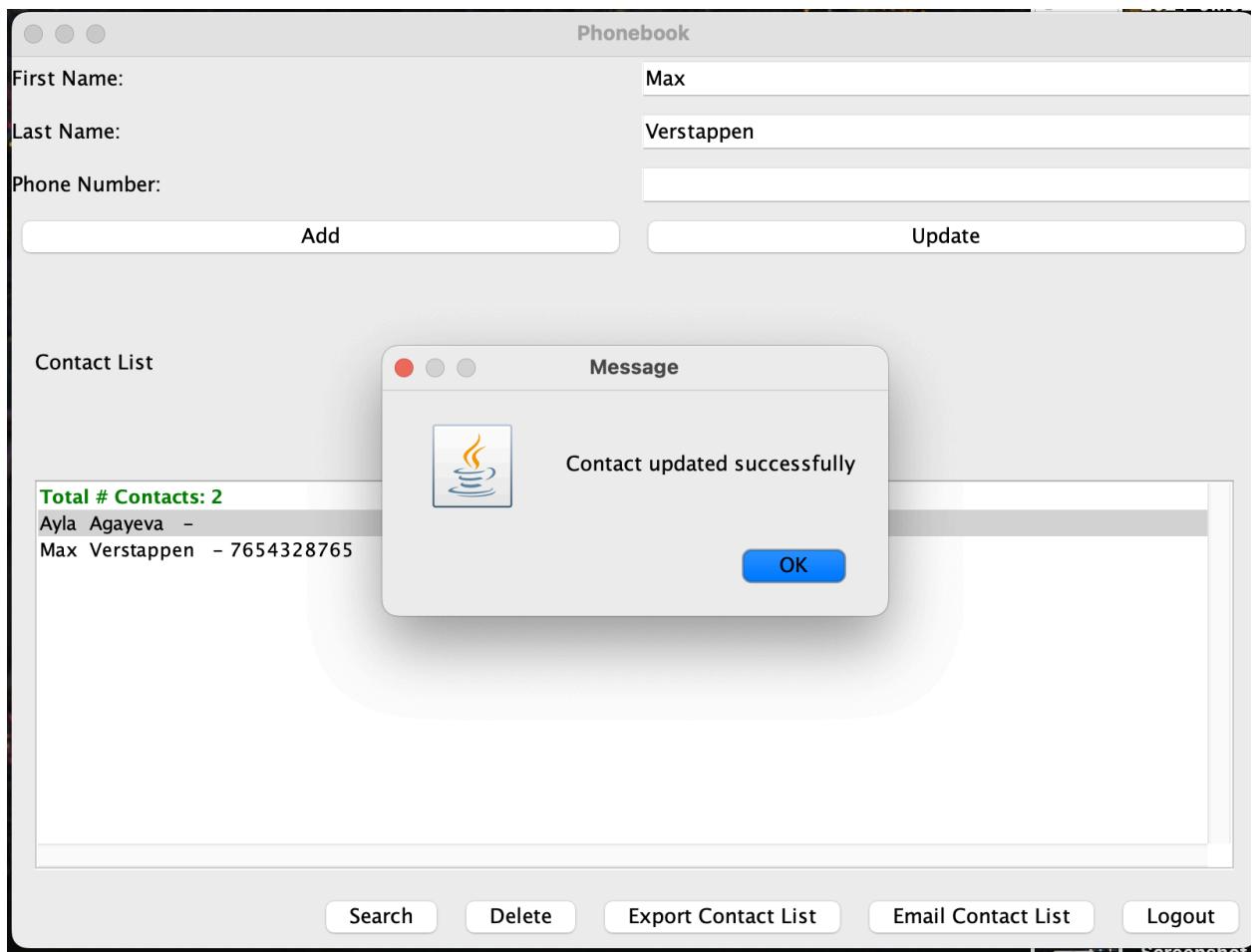
Phonebook

First Name:	Max
Last Name:	Verstappen
Phone Number:	7654328765

Contact List

Total # Contacts: 2

Ayla Agayeva	-	3124567823
Max Verstappen	-	7654328765



Phonebook

First Name: Max

Last Name: Verstappen

Phone Number:

Add Update

Contact List

Total # Contacts: 2

Ayla Agayeva -

Max Verstappen -

Search Delete Export Contact List Email Contact List Logout Screenshot

Phonebook

First Name: Max

Last Name: Verstappen

Phone Number: 7654321234

Add Update

Contact List

Total # Contacts: 2

Ayla Agayeva -

Max Verstappen - 7654321234

Search Contacts

First Name: Max

Last Name:

Phone Number:

Cancel OK

● ● ●

Phonebook

First Name:	Max
Last Name:	Verstappen
Phone Number:	7654321234

Add **Update**

Contact List

Total # Contacts: 1
Max Verstappen - 7654321234

Search **Delete** **Export Contact List** **Email Contact List** **Logout**

Screenshot

Phonebook

First Name: Max

Last Name: Verstappen

Phone Number: 7654321234

Add Update

Contact List

Total # Contacts: 2

Ayla Agayeva -
Max Verstappen - 7654321234

Message

Please select a contact to delete

OK

Search Delete Export Contact List Email Contact List Logout

Phonebook

First Name:	Max
Last Name:	Verstappen
Phone Number:	7654321234

Add Update

Contact List

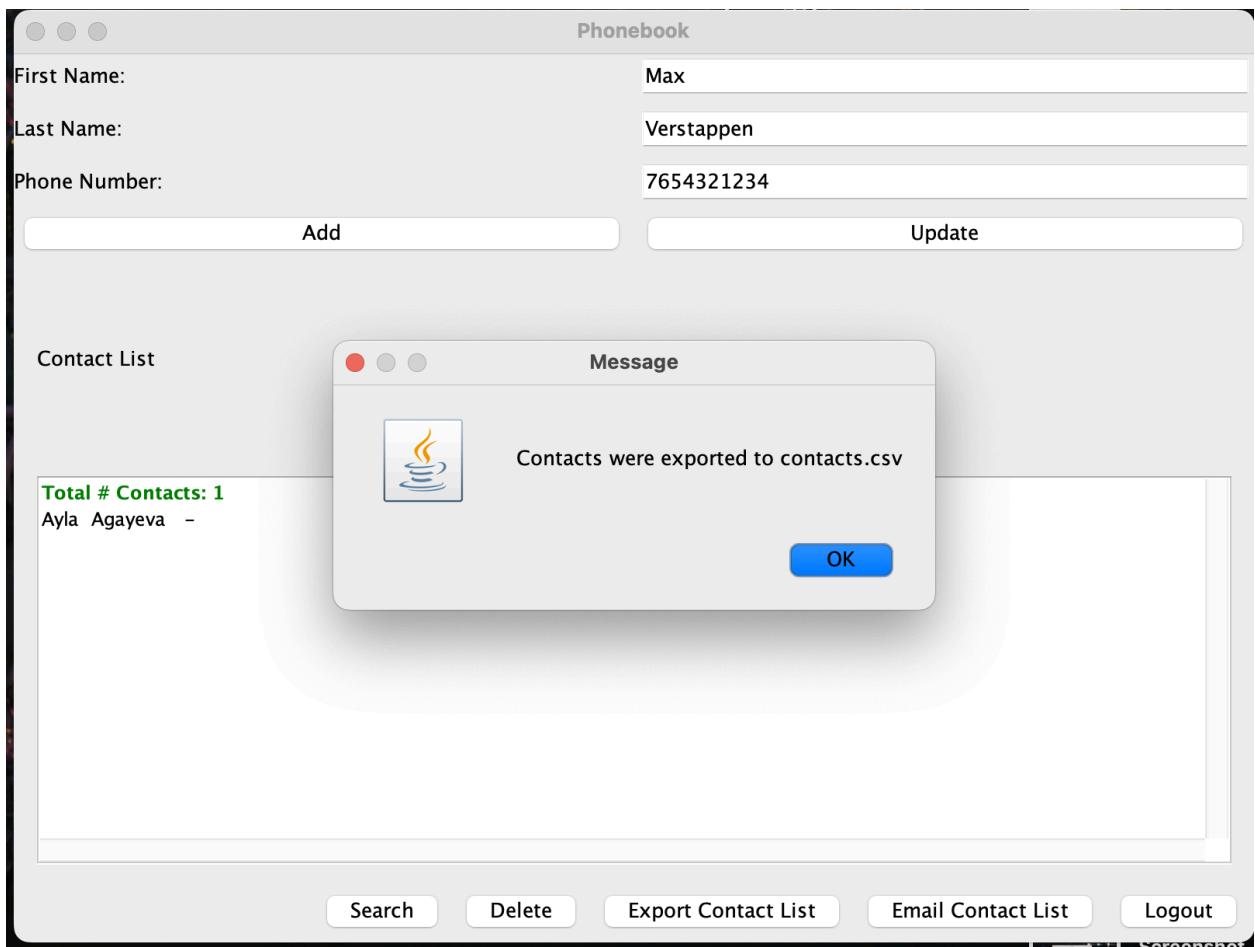
Total # Contacts: 2
Ayla Agayeva -
Max Verstappen - 7654321234

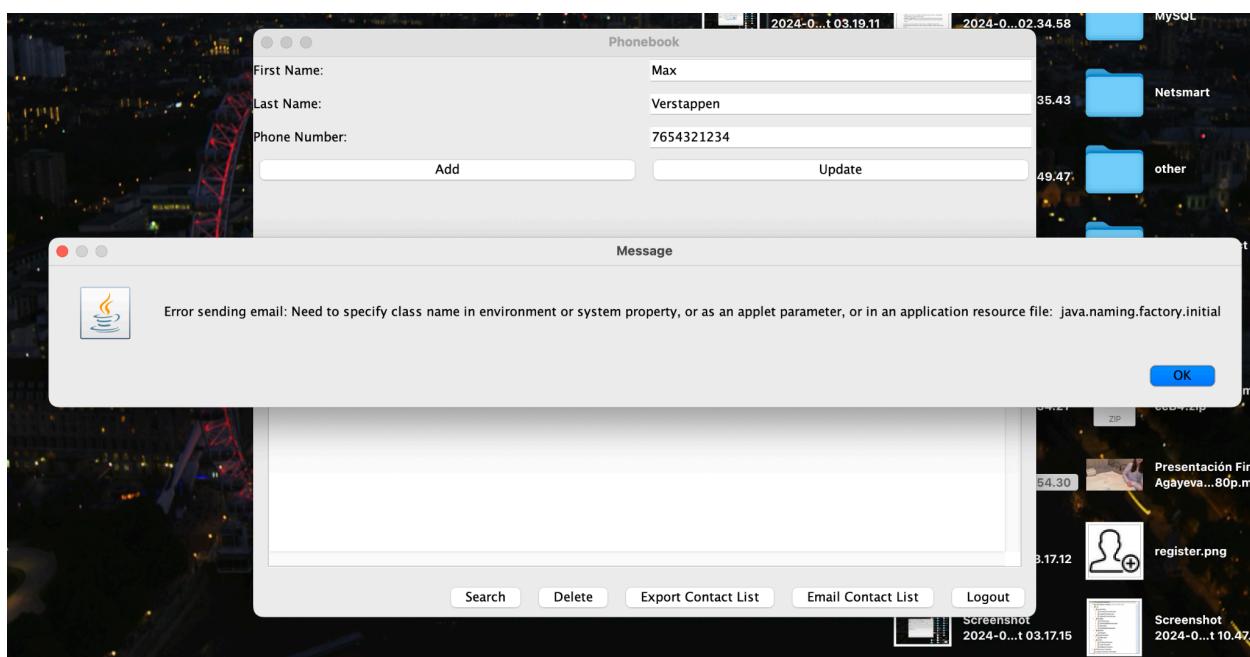
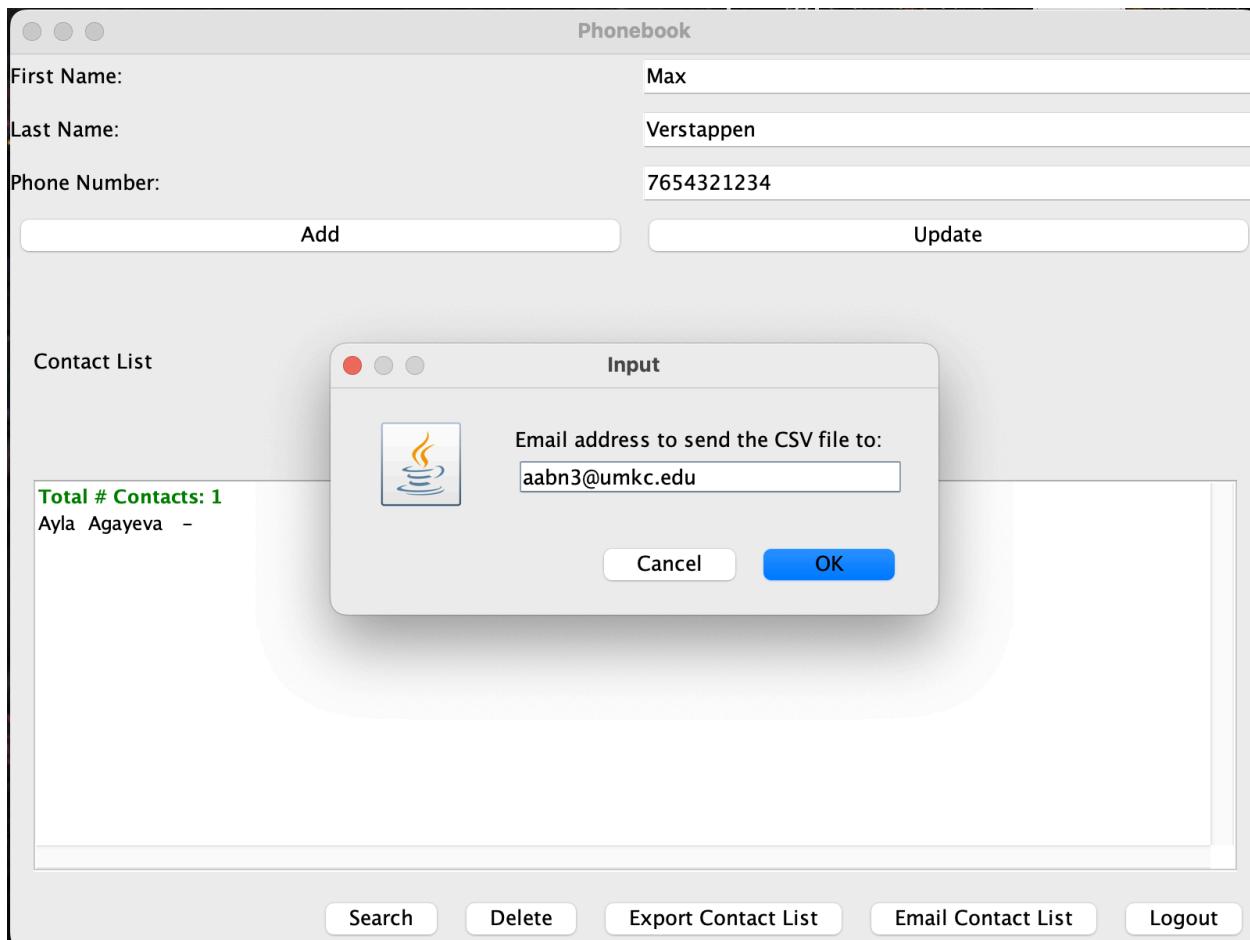
Message

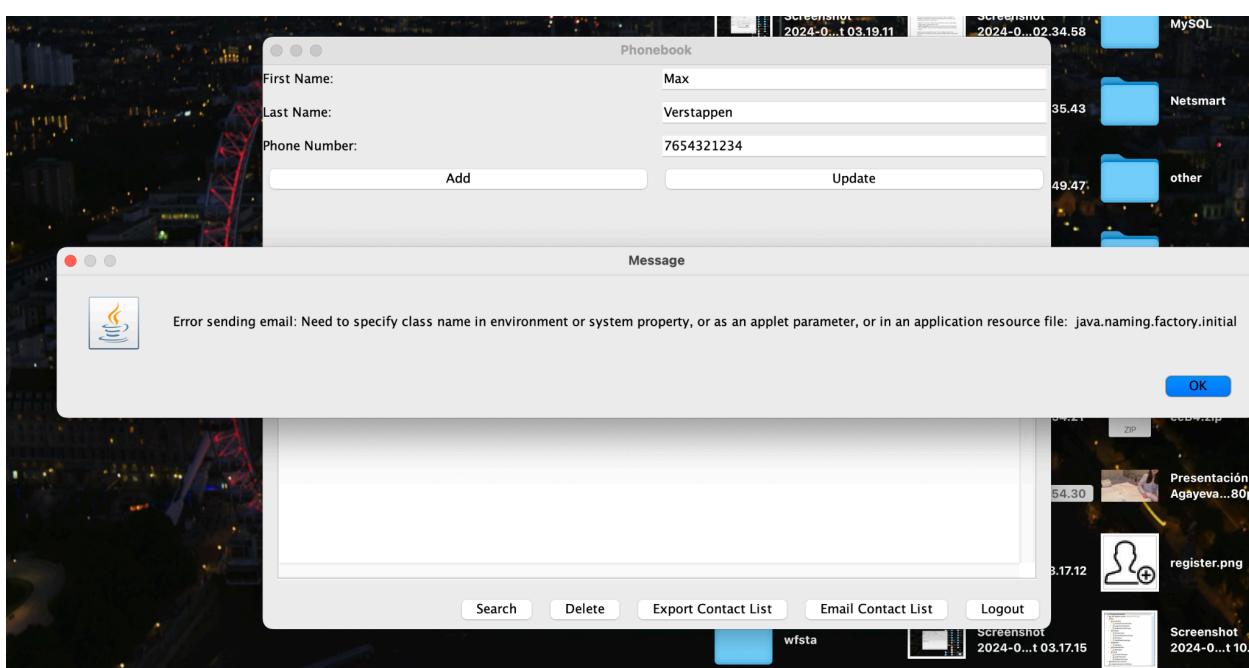
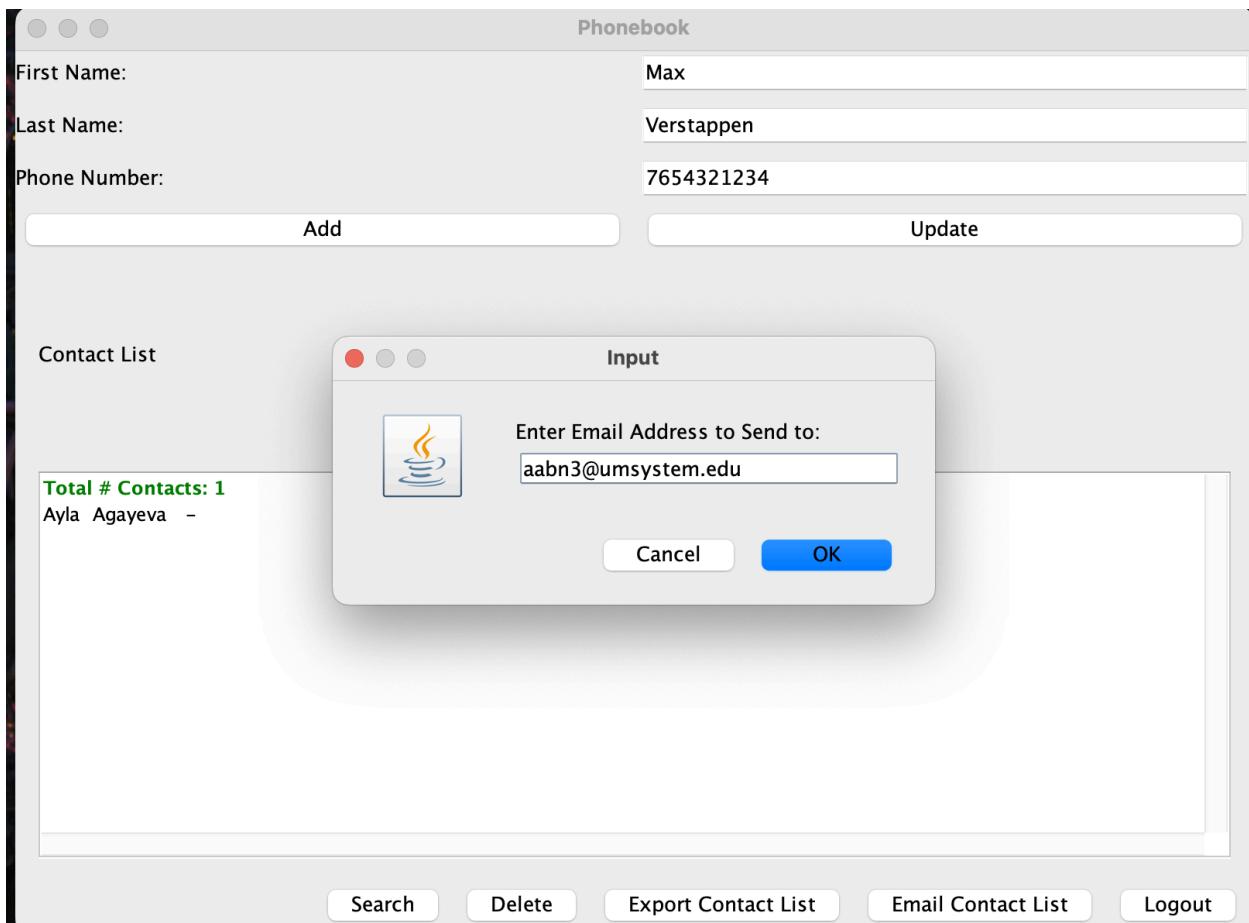
Contact deleted successfully

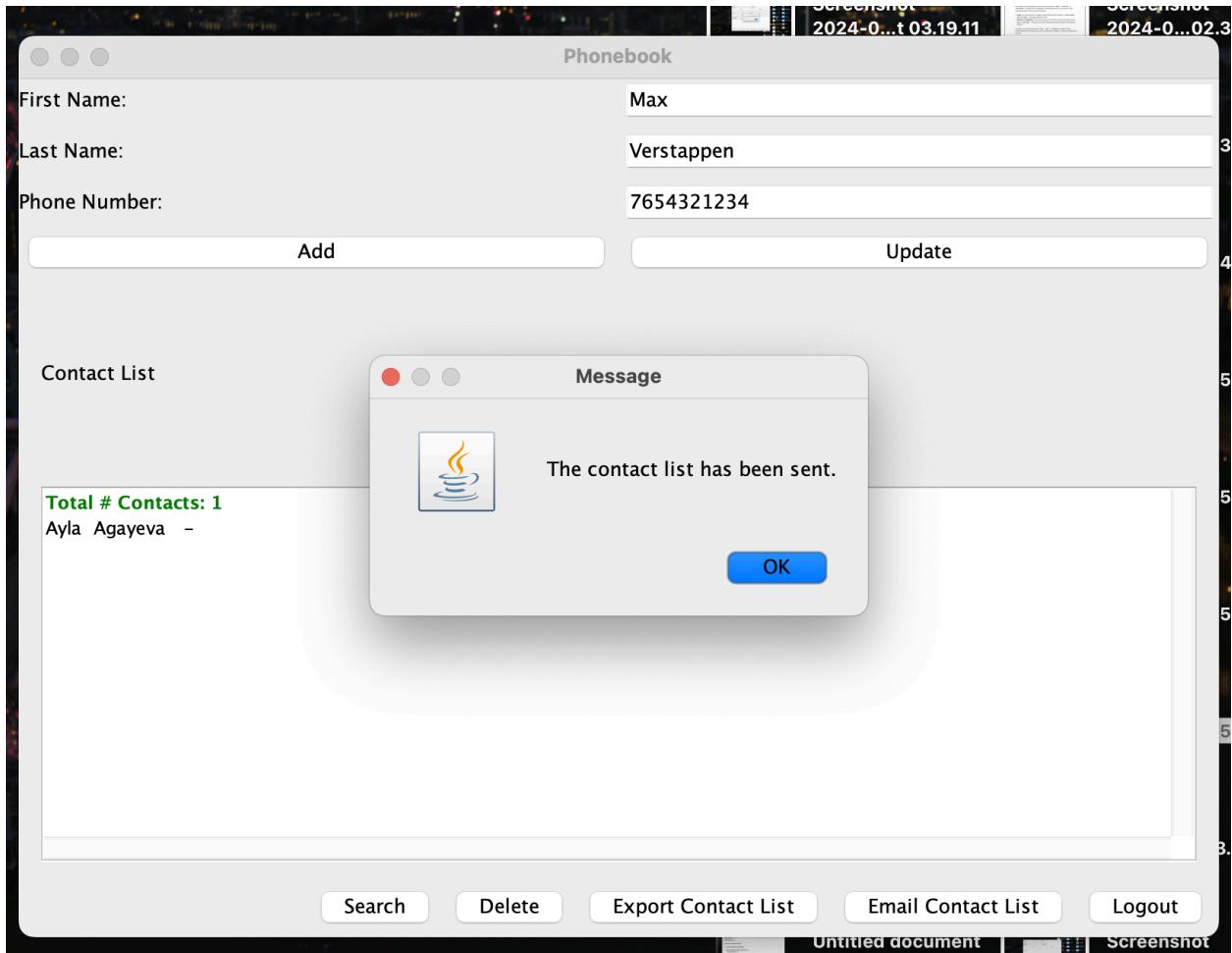
OK

Search Delete Export Contact List Email Contact List Logout





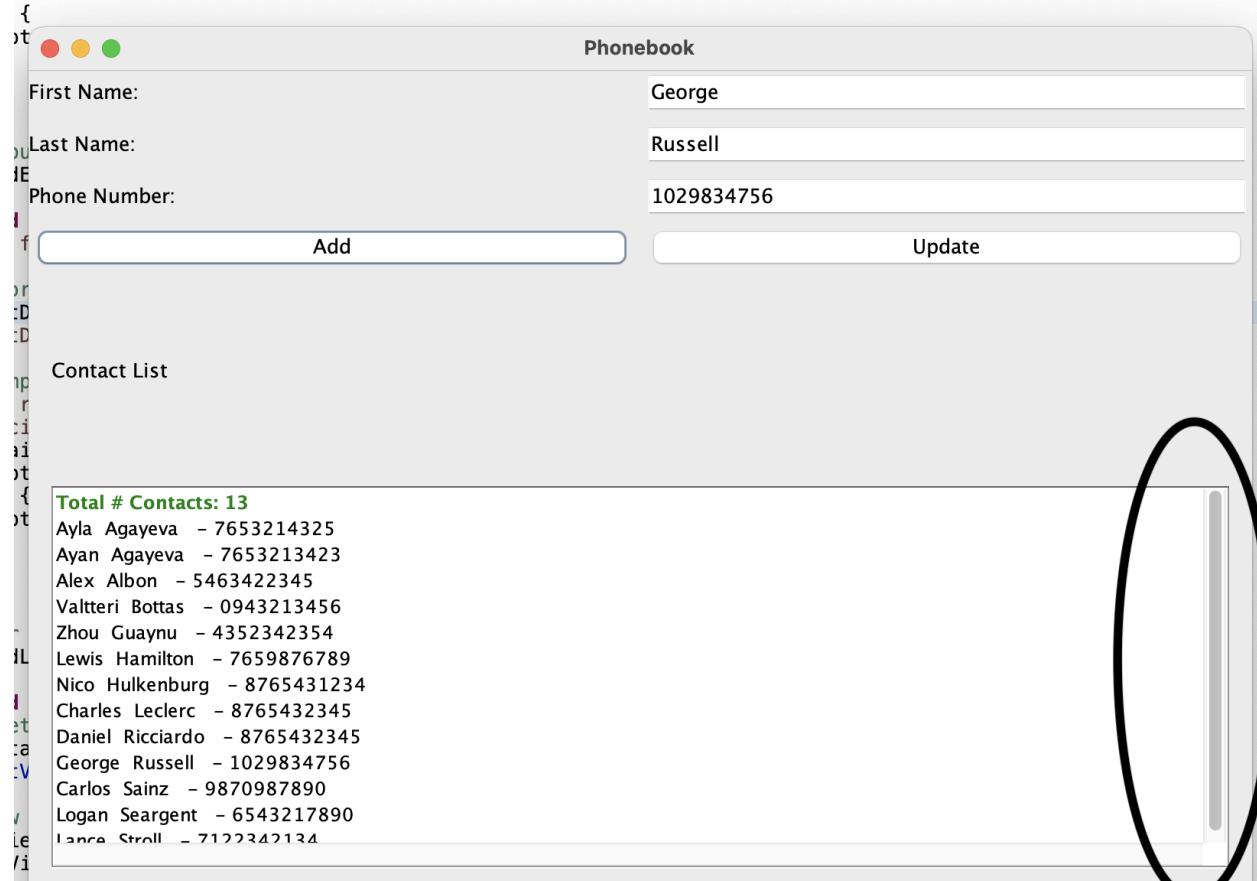




A screenshot of a Microsoft Excel spreadsheet titled "contacts". The spreadsheet contains a single row of data in row 1:

ID	First Name	Last Name	Phone Number
6	Ayla	Agayeva	7653214325
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			

```
    contactsResults = contactData.searchContacts(firstname, lastname, phonenumbers),  
    interactView.setContactsToModel(results);
```



DBeaver 24.0.2 - users

localhost - localhost:3306

Auto

Properties Data ER Diagram

users Enter a SQL expression to filter results (use Ctrl+Space)

	id	username	password
1	1	aaa	d7d2d46252dd0ea7bc5132c490597d85719f0657acdbc3282cd672bd0ba6
2	2	user	d7d2d46252dd0ea7bc5132c490597d85719f0657acdbc3282cd672bd0ba6
3	5	user21	8fab3a60577befd765cd83f3f273cd1a9f25a72356c94052c2194e816829b331
4	6	ayla08	e9c2dfcd841d457961e626132c414a1288949a014ble79b544b5749268362ea
5	10	aylaaa08	82d346ba475d73cc601646191df1ceb39b90b9d826866fb2704579b5a67c70d
6	11	aagayeva	e9c2dfcd841d457961e626132c4f4a1288949a014ble79b544b5749268362ea
7	12	gers1	f5bb6d2a58ff28425bc80d19d792b51808e63b65aa5aa52f4c89a99b9d4690
8	13	Amalie1	9783b22427e5630657132f67216e9e98d4d4ecf3bfd05ca1d0acc24ad98847
9	14	lola1	f87c6c097ece6285b729cd7e9efb2aa18891ebaeb5a1905894f02232f264797
10	15	jk1	c301d1e341d0a3a882d1b44af5a44f6bf09b673e642cd6b5def191e6c2a1b9

Project - General

Name DataSource

Bookmarks Dashboards Diagrams Scripts

Record

Refresh Save Cancel Export data 200 10 10 row(s) fetched - 0.008s (0.001s fetch), on 2024-05-05 CST en_US

DBeaver 24.0.2 - contacts

localhost - localhost:3306 Auto < N/A > 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38

localhost users contacts localhost Script Coffee Users user contacts phonebook_db users contacts localhost Databases

Properties Data ER Diagram

Enter a part of object name here

localhost - localhost:3306

Databases

CoffeeDB

phonebook_dt

Tables

contacts 32K

users 16K

Views

Indexes

Procedures

Triggers

Events

sys

Users

Administrator

System Info

Project - General X

Name DataSource

Bookmarks

Dashboards

Diagrams

Scripts

Grid

Text

1 asd das asd 1

2 6 Ayla Agayeva 7653214325 6

3 7 Ayan A 3214563748 10

4 17 Ayla A 1234567891 11

5 18 sade see 8764352345 12

6 19 Talia Ross 9945153125 13

7 21 Lisa Connor 5463217654 14

8 23 bob marley 5463217654 15

9 27 Ayan Agayeva 7653213423 6

10 28 Charles Leclerc 8765432345 6

11 29 Carlos Sainz 9870987890 6

12 30 Alex Albon 5463422345 6

13 31 Daniel Ricciardo 8765432345 6

14 32 Zhou Guanyu 4352342354 6

15 33 Lance Stroll 7122342134 6

16 34 Valtteri Bottas 0943213456 6

17 35 Lewis Hamilton 7659876789 6

18 36 Nico Hulkenburg 8765431234 6

19 37 Logan Sargent 6543217890 6

20 38 George Russell 1029834756 6

Value X 1

Record

Refresh Save Cancel Export data 200 20 20 row(s) fetched - 0.002s, on 2024-05-05 at 03:39:04

CST en_US