Contact Management App Project

This presentation outlines an OOP project: Contact management system. It demonstrates key OOP concepts and the challenges encountered during development.

Burak Alp

Project Overview: Contact Management System

Project Details

This project is a contact management system designed to organize personal contact details securely. Its purpose is to provide an efficient and user-friendly way to manage contact information, preventing loss and streamlining communication.

Target Audience

The intended audience is anyone who needs to manage contact details. This could include individuals, professionals, or anyone who frequently communicates with others.

System Architecture: Class Diagram

Contact Class

The Contact class encapsulates the attributes of a contact, including name, phone number, email, and address. It utilizes private fields with getters and setters to ensure data integrity.

ContactManager Class

The ContactManager class manages the collection of Contact objects. It handles operations such as adding, searching, updating, deleting, and displaying contacts.

Key Features: Contact Management Functionality

1. **Add Contact**

   Allows users to input and store new contact information.

2. **Search Contact**

   Enables searching for contacts by name, making it easy to locate specific entries.

3. **Update Contact**

   Provides the ability to modify existing contact details, ensuring information remains current.

4. **Delete Contact**

   Offers a way to remove unwanted contacts, keeping the list clean and organized.

5. **Display All Contacts**

   Displays all contacts in a sorted order, facilitating easy access and management.

Implemented OOP Concepts: Building a Robust System

Encapsulation

The Contact class encapsulates data using private fields and provides access through getters and setters, promoting data security and modularity.

Inheritance

Currently, inheritance is not implemented, but it could be used in the future to create specialized contact types (e.g., business contacts).

Polymorphism

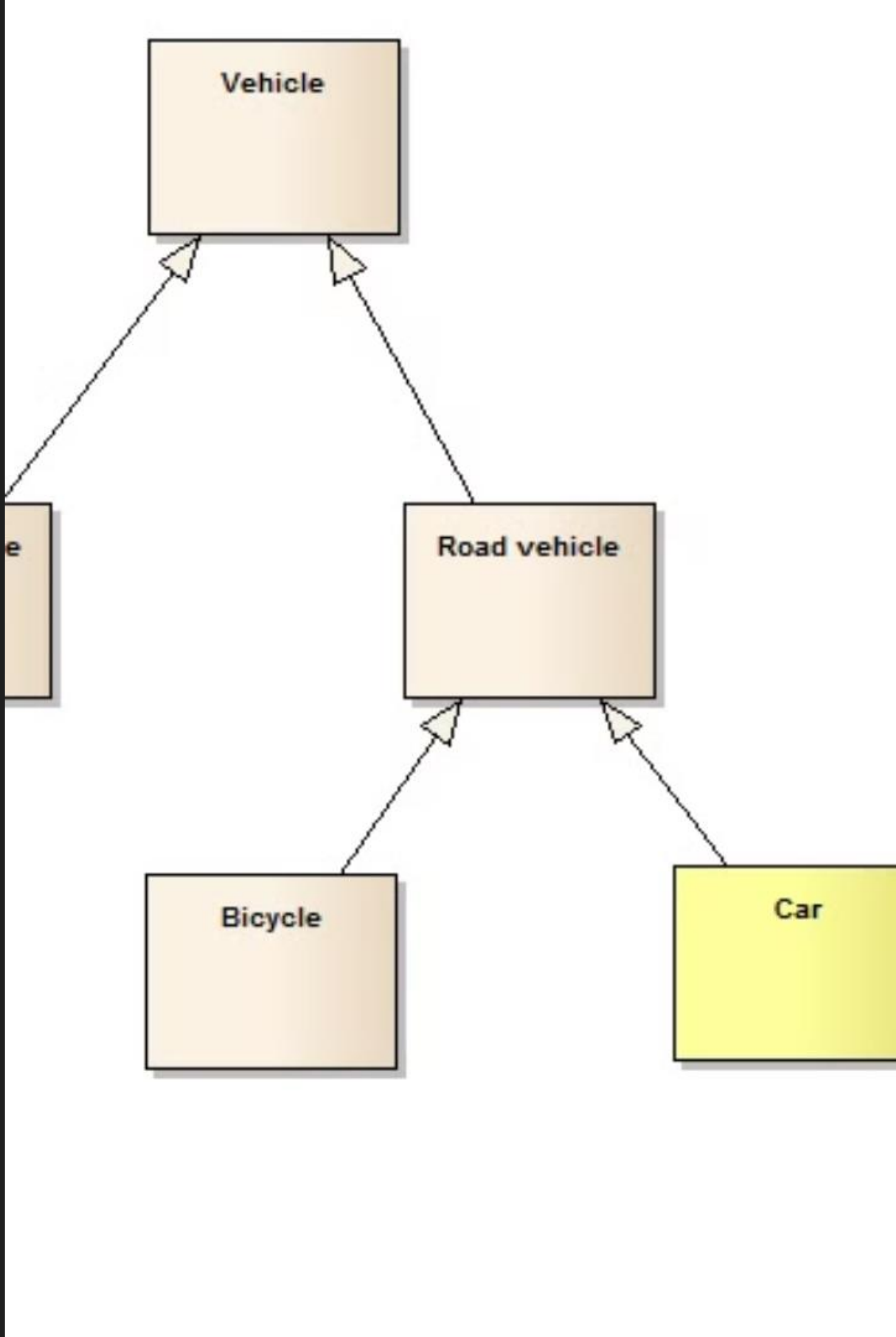
Not used yet, but potential for future implementation through method overloading or overriding to provide flexible contact management.

Abstraction

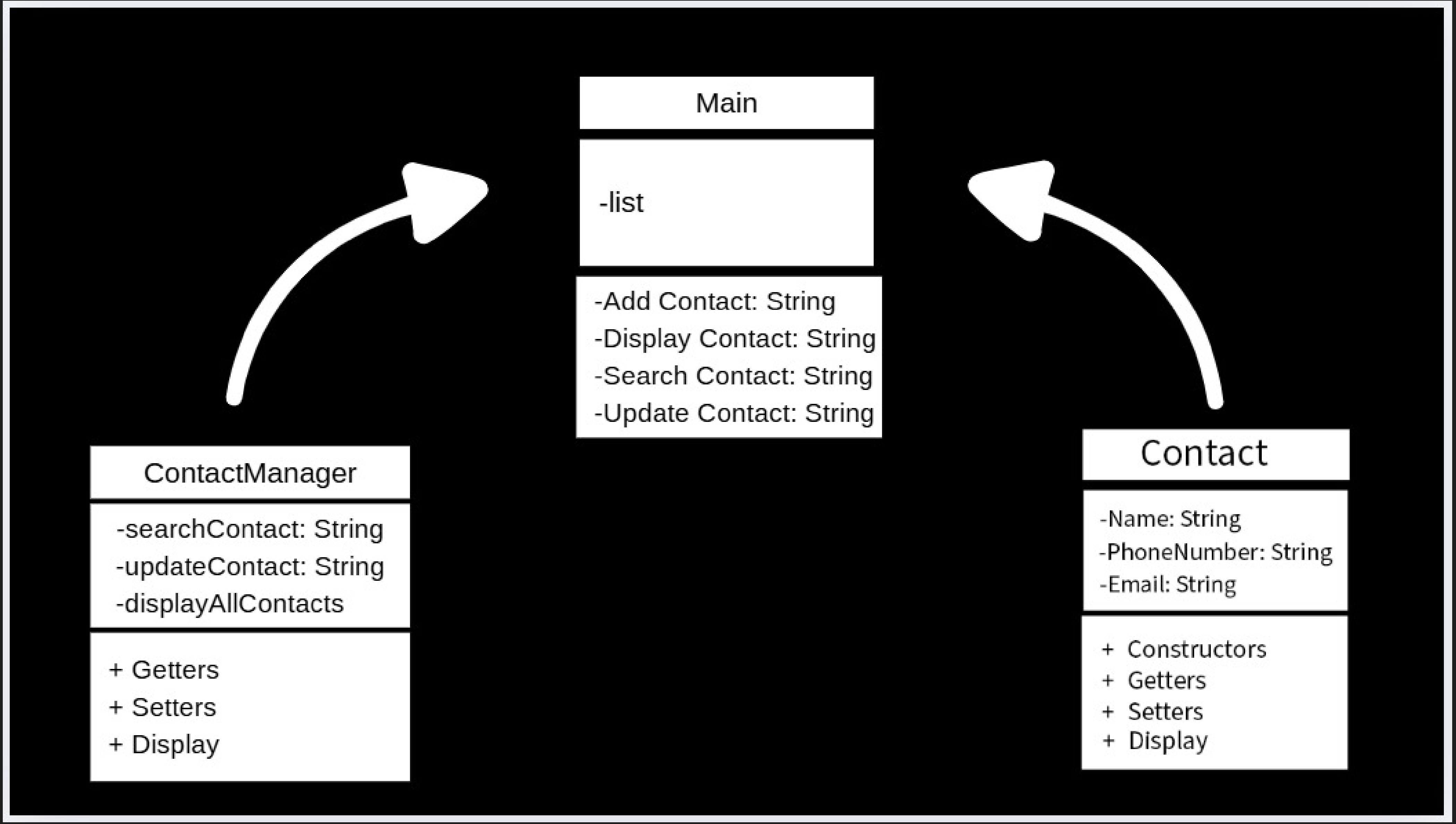
The ContactManager class abstracts complex data management operations, simplifying interaction with the system for users.

Evaluation of OOP Concepts: Examining Implementation

Concept	Implementation Details
Classes & Objects	The Contact class represents individual contacts, and the ContactManager class manages the collection of Contact objects.
Encapsulation	The Contact class enforces encapsulation by using private fields with getters and setters, ensuring controlled access to data.
Inheritance	Inheritance is not yet implemented but can be added in the future to extend the functionality of the Contact class.
Polymorphism	Polymorphism is not implemented yet, but it could be used to create flexible methods for handling various contact types.



UML Diagram of the Project



Challenges Faced: Overcoming Obstacles

File I/O Management

Managing file operations (reading and writing CSV files) without corrupting data proved challenging. The goal was to ensure reliable data storage and retrieval.

Interface Design

Designing an intuitive interface to handle user inputs seamlessly was a key challenge. The aim was to create a user-friendly experience with clear instructions and prompts.

Overcoming Challenges: Solutions and Strategies

1

Error Handling and Data Validation

Robust error handling was implemented for file I/O operations and data validation. This ensured data integrity and minimized the risk of corruption.

2

User Interface Simplification

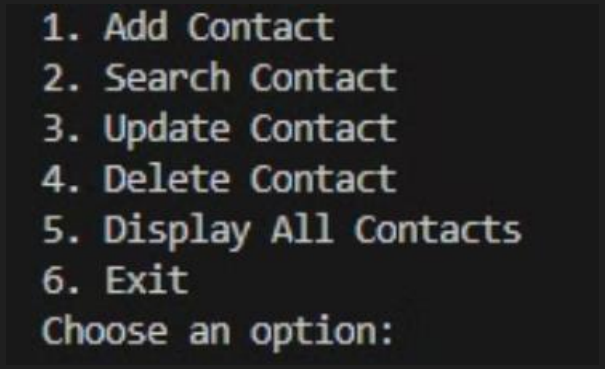
Focus was placed on simplifying the user interface for ease of use. Clear instructions, intuitive prompts, and a logical flow were incorporated.

2

Usage of Code Generator Tools

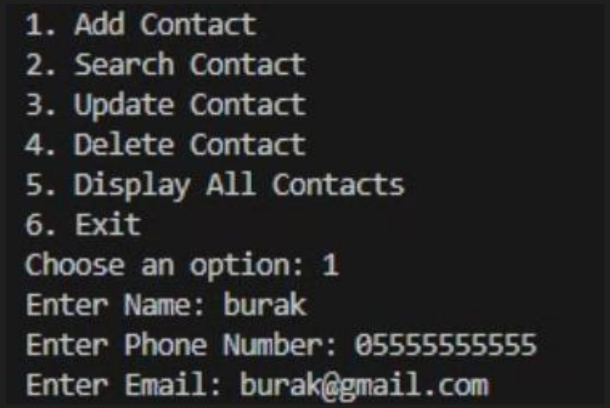
ChatGPT acted as a code suggestion and documentation tool, providing ideas, snippets, and guidance to streamline development without generating the entire code.

Screenshots from the Program



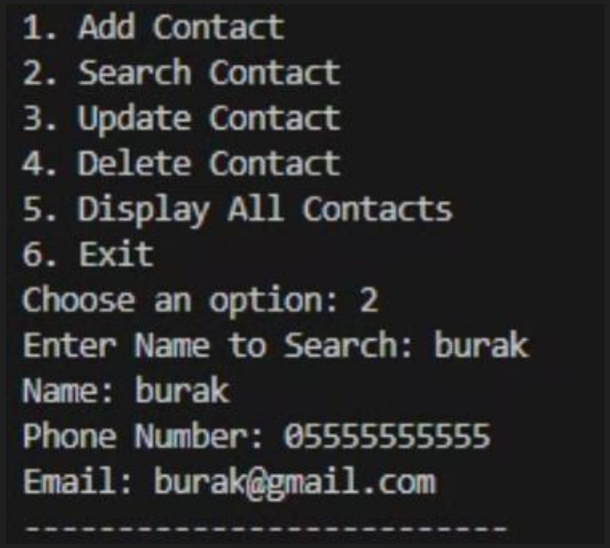
```
1. Add Contact
2. Search Contact
3. Update Contact
4. Delete Contact
5. Display All Contacts
6. Exit
Choose an option:
```

Start of the App



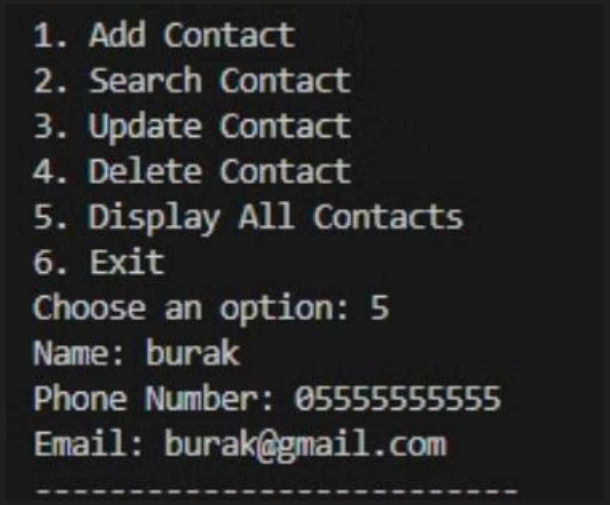
```
1. Add Contact
2. Search Contact
3. Update Contact
4. Delete Contact
5. Display All Contacts
6. Exit
Choose an option: 1
Enter Name: burak
Enter Phone Number: 05555555555
Enter Email: burak@gmail.com
```

Adding Contact



```
1. Add Contact
2. Search Contact
3. Update Contact
4. Delete Contact
5. Display All Contacts
6. Exit
Choose an option: 2
Enter Name to Search: burak
Name: burak
Phone Number: 05555555555
Email: burak@gmail.com
-------------------------
```

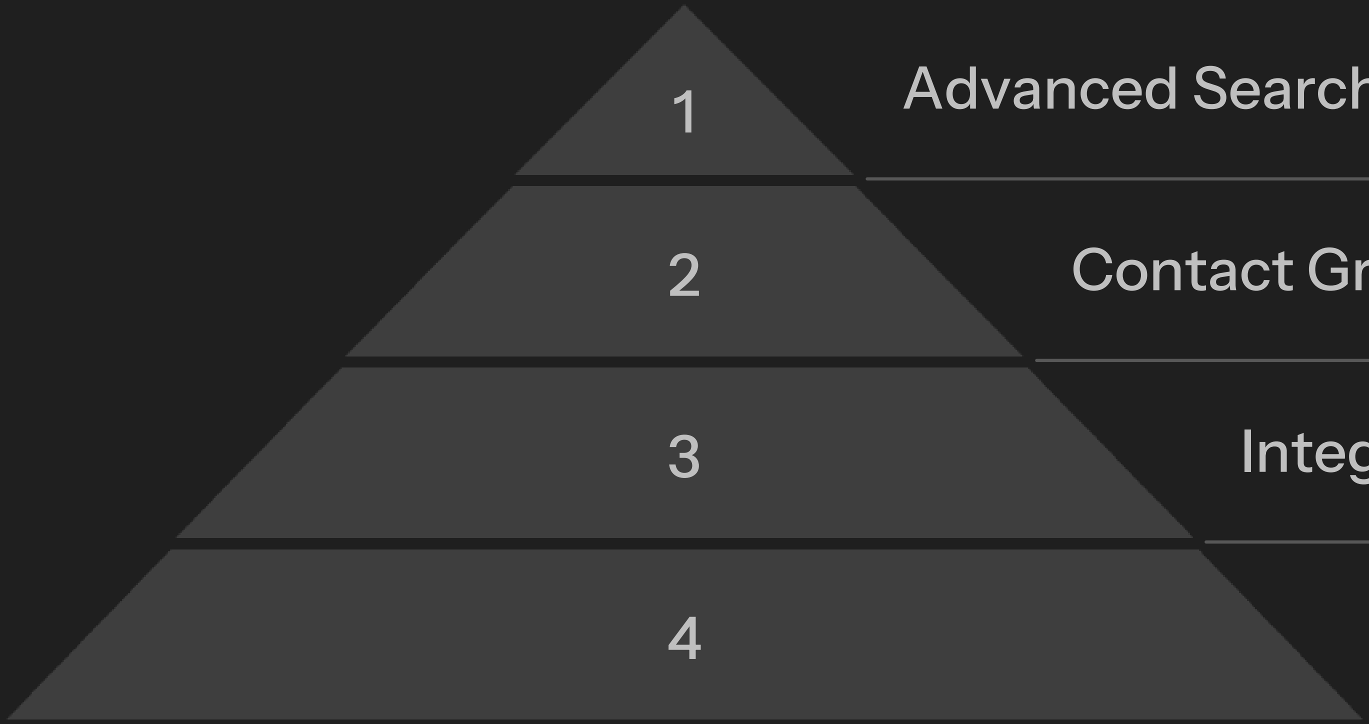
Searching Contact



```
1. Add Contact
2. Search Contact
3. Update Contact
4. Delete Contact
5. Display All Contacts
6. Exit
Choose an option: 5
Name: burak
Phone Number: 05555555555
Email: burak@gmail.com
-------------------------
```

Displaying all Contacts

Future Improvements: Expanding Functionality



Advanced Search Options

Contact Grouping

Integration with Other Applications

Enhanced User Interface

Future improvements could include adding advanced search options, enabling contact grouping, integrating with other applications, and enhancing the user interface.