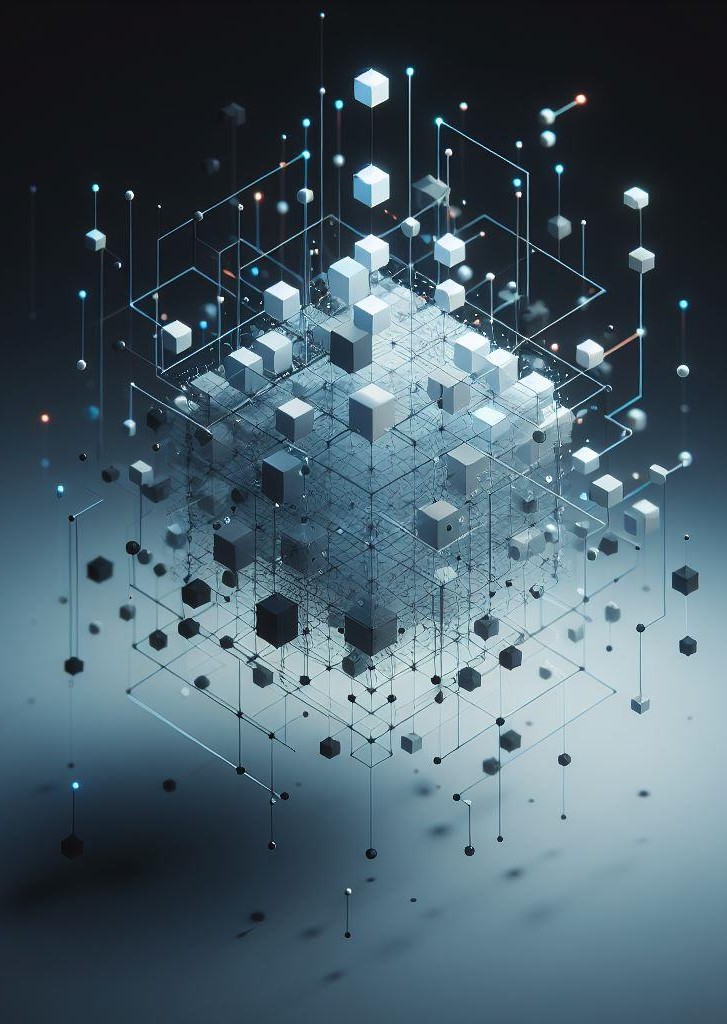
[Date]



DATA STRUCTURE

Group Project: Phone Book

GROUP MEMBERS

MAHI IMRAN BP 4324 AMEN TEGEGNE BU 7929 AMANUEL AYALEW IY9455

Phone Book

This phone book application is a simple program designed to manage your contacts. Here's a quick rundown of its features:

* Store **Contacts:** Add new contacts with their first name, last name, email address, and phone number.
* Organized **Listing:** Contacts are stored alphabetically by first name for easy searching.
* **Search Functionality:** Find specific contacts by their name or phone number.
* **Edit Contacts:** Update existing contact information if something changes.
* **Delete Contacts:** Remove unwanted contacts from your list.

The code is written in C# for a Windows Forms application and implements basic phone book functionality. Here's a breakdown of the code:

# Classes:

* Contacts: This class represents a single contact in the phone book. It has properties for FirstName, LastName, Email, and PhoneNumber, and a Next property to link it to the next contact in the list.
* Form1: This class represents the main form of the application. It contains various controls like text boxes, buttons, and a DataGridView control to display contacts. It also contains methods for various functionalities like inserting, searching, deleting, and updating contacts.

# Data Structure:

* A singly linked list was chosen to store contact information. This allows for efficient insertion of new contacts at the beginning or within the list, which is a frequent operation in a phone book application. **For Dynamic Memory Allocation:** Singly linked lists don't require a pre-defined size like arrays. This allows the application to grow organically as the number of contacts increases. **Simplicity:** Singly linked lists are relatively simple to implement and understand, making them a good choice for this basic phone book application.

# Functionality:

* Inserting Contacts: The insert method takes contact details and inserts them into the linked list in alphabetical order based on the first name.
* Displaying Contacts: The displayContacts method iterates through the linked list and populates the dataGridView2 control with contact information.
* Searching Contacts: The SearchContact and searchcontactphone methods search for contacts based on name or phone number, respectively. If a match is found, contact details are added to the dataGridView1 control. These is using modified Linear Search.
* Deleting Contacts: The DeleteContacts method searches for a contact by name and removes it from the linked list if found.
* Updating Contacts: The Update method searches for a contact by name and updates its information with values entered in the update panel.

# Challenges Faced:

* Error Handling: Ensuring proper validation of user input, especially for phone number format.
* User Interface (UI) Design: Balancing functionality with a user-friendly and intuitive interface.
* Search Efficiency: While searching by name or phone number works.

# Solutions Implemented:

* Input Validation: The code includes basic checks for empty fields and attempts to validate phone number format using long.TryParse.
* UI Design: The application uses separate panels for update and search functionalities to avoid cluttering the main view.
* Search: While not fully optimized, the current search offers basic functionality. Further improvements can be implemented for larger datasets.

UNZIP TO ACCESS THE PROGRAM