**WELCOME INTERFACE**

As we start our program, it depicts welcome message. It illustrates 3 components. First one is a head message showing "**Contact Management System Project in c**". Second one is the name of the programmers who made this management system. Third portion showing a greeting message “**WELCOME TO CONTACT MANAGEMENT SYSTEM**” to the user and prompt him/her to enter any key.

**A screenshot of a computer screen

Description automatically generated with medium confidenceFigure 1**

Here, we have used 3 functions. First one is **welcomeMessage()**, second one is **headMessage()**,  
third one is **printMessageCenter()**. We oriented this via newline character (**\n**), tab character (**\t**) and space character.

**MAIN MENU**

**Figure** A screenshot of a computer

Description automatically generated with medium confidence**2**

After choosing any character by the user, our program enters a new window that has a menu consist of-

[1] Add a new contact , [2] Show all contacts , [3] Search for contact , [4] Edit a contact , [5] Delete a contact , [6] Delete all contact , [7] Clear window , [i] User guideline , [a] About us , [0] Exit.

Users must choose any of the given option to perform the corresponding task. We stored these options in a function called **menu()**. **menu()** function is called by the **main()** function. A **switch** case is used to take the user input which will execute the given task.

**ADD A NEW CONTACT**

This is the first option of out project menu. By using this option user can easily add contact. If a user enters **1** as input, user can see this type of output.

**Figure** A screen shot of a computer

Description automatically generated with medium confidence**3**

If the user clicks **1**, he will enter in an interface which will prompt him to enter his name, phone number and email. Name, phone number and email must be unique and should not exceeds 20 character (for name and phone number) or 30 characters (for email). If it happens, console will show length error and already exist error accordingly. If all the information is given without an error, the contact will be added successfully. Then the user must choose either back by entering **b** or exit by entering **0**.

**addNewContact()** function is called by the **main()** function as user entered **1**. A global structure **Newcontact** is declared to customize a data type. Then under this data type a variable is declared which then prompt the user to enter name, phone number and email. After scanning users’ information with **scanf()** function, **GoBackOrExit()** function is called.

**SHOW ALL CONTACTS**

This is the second option of out project menu. By using this option user can easily see all contacts.

A screenshot of a computer screen

Description automatically generated with low confidence**Figure 4**

If the user enters **2** to see all the contact he saved, he will see a table containing 3 columns. 1st column for the name, 2nd column for the phone number and 3rd column consists of email. Additionally, it shows how many contacts has been stored. Then the user must choose either back by entering **b** or exit by entering **0**.

The **ShowAllContacts()** function is responsible for displaying all the contacts stored in a text file named **All-Contact**. The **headMessage()** function is called to display a header message, indicating that the following content is for displaying all contacts. A file pointer **AllContactTextFile** is declared to open the "**All-Contact.txt**" file for reading. The program then prints a table header for the contact information, with columns for Name, Phone Number, and Email. A while loop reads each line from the file using **fgets()**. The **PrintLineWithSpace()** function is called to print each line of contact information with proper formatting. The **PrintLineWithSpace()** function takes two parameters: a pointer to the name (a character array) and a character indicating the type of contact information ('**n**' for Name, '**p**' for Phone Number, '**e**' for Email). Inside the **PrintLineWithSpace()** function, the name is formatted by calculating the total space needed to align the text properly within the designated column width. The formatted name is then printed along with the appropriate padding. For the 'e' case, a horizontal line is printed after each email entry. After printing all the contacts, the total number of contacts is displayed. The **GoBackOrExit()** function is called to provide options for the user to go back or exit the program.

**SEARCH CONTACT**

This is the third option of out project menu. By using this option user can easily search for contact.

A screenshot of a computer

Description automatically generated with medium confidence**Figure 5**

If the user wants to search a contact and to know the information of the corresponding contact, he must search it by giving the name. Then the file will search for the name and print the respective information of the contact (For example, phone number, email). Then the user must choose either back by entering b or exit by entering 0.

Searching and displaying contact information based on a user-provided name. Here is an explanation of the code. A variable named **FoundContact** is initialized to 0. This variable will be used to track if a contact matching the provided name is found. A character array named **Name** with a size of **100** is declared to store the name of the contact being searched. User is prompted to enter the name of the contact they are looking for. The entered name is read using **scanf()** and stored in the **Name** array. A newline character is appended to the **Name** array to match the format of the contact information in the file. A file pointer named **AllContactFile** is declared to open the "**All-Contact.txt**" file for reading. Variables for line length (**LineLength**) and a character array to store each line of the file (**Line**) are initialized. A while loop is used to read each line from the file using **fgets()**. The **LineCount** variable is incremented to keep track of the lines read. If the current line count is 1 and the name in the line matches the searched name, **FoundContact** is set to 1, and **NeedToPrintLine** is set to 3 to indicate that the contact information needs to be printed. If **NeedToPrintLine** is greater than 0, it means the program is currently processing the lines of the found contact's information, and the lines are printed accordingly. After printing the contact information, **NeedToPrintLine** is decremented to indicate the next line to be printed. When **LineCount** reaches 3, it means all the lines for a contact have been processed, so **LineCount** is reset to 0. After processing all the lines in the file, the file is closed. If **FoundContact** is still 0, it means no contact with the provided name was found, and an appropriate message is displayed. Finally, the **GoBackOrExit()** function is called to provide options for the user to go back or exit the program.

**EDIT CONTACT**

This is the fourth option of out project menu. By using this option user can easily edit contact.

If the user wants to edit a contact, he must choose option **4** and then enter his name. This program allows the user to see the previous name he /she has given and prompt him/her to enter a new name. If user do not want to change the name rather, he wants to fix the other information he can press **0**. Then a statement "**contact updated successfully**" is shown in the console. After performing the task, the user must choose either back by entering **b** or exit by entering **0**.

**A screenshot of a computer screen

Description automatically generated with low confidenceFigure 6**

Several variables are declared, including **LineCount** to track the lines read, **FoundContact** to determine if the contact to be edited is found, **SkipLines** to indicate the lines that need to be skipped, and arrays to store the given name, new name, new phone number, new email, and the full updated contact information. The user is prompted to enter the name of the contact they want to edit. The entered name is read using scanf and stored in the **GivenName** array. If the length of the entered name exceeds 20 characters, an error message is displayed using the **ErrorAndRestart()** function, and the function returns. A newline character is appended to the **GivenName** array to match the format of the contact information in the file. Two file pointers, **OurExistingFile** and **NewTempFile**, are declared. **OurExistingFile** is opened in "**r**" mode to read the existing contact file, and **NewTempFile** is opened in "**w**" mode to write the updated contact information. A while loop is used to read each line from the existing contact file using **fgets()**. The **LineCount** variable is incremented to keep track of the lines read. If the current line count is 1 and the name in the line matches the given name, **FoundContact** is set to 1, and **SkipLines** is set to 3 to indicate that the subsequent lines need to be skipped. If **SkipLines** is greater than 0, it means the program is currently processing the lines of the found contact's information. Depending on the line count, different fields of the contact information are processed. For each field (name, phone number, email), the program displays the prior value and prompts the user to enter a new value. The entered new values are validated for length and existence using the **AlreadyExists** function. The new values are concatenated to the **NewFullContact** string. When LineCount reaches 3, it means all the lines for a contact have been processed, so LineCount is reset to 0, and the **NewFullContact** string is written to the **NewTempFile**. If **SkipLines** is 0, it means all lines for the current contact have been processed, so the original line is written to **NewTempFile**. After processing all the lines in the file, both the existing contact file and the new temporary file are closed. If **FoundContact** is still 0, it means no contact with the given name was found, and an appropriate message is displayed. The temporary file is then removed. If a contact is found, a success message is displayed, and the original contact file is removed. The temporary file is renamed to replace the original contact file. Finally, the **GoBackOrExit**() function is called to provide options for the user to go back or exit the program.

**DELETE A CONTACT**

This is the fifth option of out project menu. By using this option user can easily delete a contact.

**A screenshot of a computer screen

Description automatically generated with low confidenceFigure 7**

If the user wants to delete a contact, contact name must be given. Then the file will search for the name and delete the respective information of the contact (For example name, phone number, email) the contact file. The contact file is then updated with the changes. Then the user must choose either back by entering **b** or exit by entering **0**.

Various variables are declared, including **lineCount** to track the lines read, **FoundTheContact** to determine if the contact to be deleted is found, **SkipLines** to indicate the lines that need to be skipped, and an array to store the given name. The user is prompted to enter the name of the contact they want to delete. The entered name is read using scanf and stored in the **GivenName** array. If the length of the entered name exceeds 20 characters, an error message is displayed using the **ErrorAndRestart()** function. A newline character is appended to the **GivenName** array to match the format of the contact information in the file. Two file pointers, **OurExistingFile** and **NewTempFile**, are declared. **OurExistingFile** is opened in "**r**" mode to read the existing contact file, and **NewTempFile** is opened in "**w**" mode to write the updated contact information. A while loop is used to read each line from the existing contact file using **fgets()**. The **lineCount** variable is incremented to keep track of the lines read. If the current line count is 1 and the name in the line matches the given name, **FoundTheContact** is set to 1, and **SkipLines** is set to 3 to indicate that the subsequent lines need to be skipped. If **SkipLines** is greater than 0, it means the program is currently processing the lines of the contact's information to be deleted. In this case, the **SkipLines** variable is decremented. If **SkipLines** is 0, it means all lines for the current contact have been skipped, so the current line is written to **NewTempFile**. If **lineCount** reaches 3, it means all the lines for a contact have been processed, so **lineCount** is reset to 0. After processing all the lines in the file, both the existing contact file and the new temporary file are closed. If **FoundTheContact** is still 0, it means no contact with the given name was found, and an appropriate message is displayed. The temporary file is then removed. If a contact is found, a success message is displayed, and the original contact file is removed. The temporary file is renamed to replace the original contact file. Finally, the **GoBackOrExit()** function is called to provide options for the user to go back or exit the program.

**DELETE ALL CONTACTS**

This is the sixth option of out project menu. By using this option user can easily delete all contacts.

This interface allows the user to delete all contacts by confirming their choice and removing the existing contact file. A new empty file is then created to replace the deleted file, effectively deleting all the contacts. Then the user must choose either back by entering **b** or exit by entering **0**.

**A screenshot of a computer

Description automatically generated with medium confidenceFigure 8**

Some variables are declared, including Option to store the user's choice for deleting all contacts. **getchar()** is called to consume the newline character from the previous input. The user is prompted to confirm whether they want to delete all the contacts by pressing '**Y**' or '**N**'. The user's input is read using scanf and stored in the Option variable. If the user enters '**Y**' or '**y**', indicating a positive confirmation, the code inside the if statement is executed. The "**All-Contact.txt**" file is removed using the remove function to delete the existing contact file. A new empty file with the same name is created using **fopen()** with "**a**" mode (append) and immediately closed using **fclose()**. This step ensures that an empty file with the same name is created for storing new contacts. A message indicating the deletion process is displayed. A loop is used to print the characters of the "**Deleting....**" string one by one with a delay of 40 milliseconds between each character using the **sleep()** function (assuming it is a custom implementation of a delay function). Once the loop is completed, a success message is displayed. Finally, the **GoBackOrExit()** function is called to provide options for the user to go back or exit the program.

**CLEAR WINDOW**

This is the seventh option of out project menu. By using this option user can easily clear the window.A screenshot of a computer

Description automatically generated with medium confidence

**Figure 9**

To clear the window, we use **system(“cls”)** function.

**USER GUIDELINE**

In this section we show the guideline for user. It helps user to use this application easily.

**A screenshot of a computer

Description automatically generated with medium confidenceFigure 10**

**ABOUT US**

In this part we talk about use. Here we give our Information.

**A screen shot of a computer

Description automatically generated with medium confidenceFigure 11**

**EXIT**

If the user wants to exit, then he/she must press **0**. Then console will give a message that he wants to exit or not. If user press **Y** in any case, then the program will exit, and a goodbye message will be shown.

**A screenshot of a computer

Description automatically generated with medium confidenceFigure 11**

**A screenshot of a computer message

Description automatically generated with low confidenceFigure 12**

**---------**