## horizontal line



Student Data Management Portal

25.11.2024

**─**

Prepared By

Dev Patel ( 24ITC003 ) &

Khush Pithva ( 24ITC087 )

# Overview

A comprehensive C-based application designed to efficiently manage student records, enabling operations like adding, editing, searching, viewing, and statistical analysis with user-friendly interfaces for both administrators and students.

# Goals

1. Add New Students Data
2. Edit Existing Student Data
3. Delete Data of Students
4. View All Students Data
5. Search For Any Particular Students Data
6. Sort Students Data By Birth Date & Roll No.
7. Exit From The Program

# Concepts Used

1. Functions
2. Structures
3. File Handling

## Functions

Functions like add, view, search, and edit organize the program into reusable modules, improving readability, maintainability, and ease of debugging

## Structures

The student and DoB structures group related fields (e.g., name, roll number, date of birth), providing an organized and efficient way to handle complex data.

## File Handling

The program uses file operations (fopen, fwrite, fread) to store and retrieve student records persistently, allowing data to remain available even after the program ends.

# Methodology Followed

1. #include<stdio.h>
2. #include<conio.h>
3. #include<string.h>
4. #include<stdlib.h>
5. #include<stdbool.h>
6. #include<ctype.h>
7. #define ERROR\_COLOR "\033[1;31m" // Bold Red
8. #define HEADER\_COLOR "\033[96m" // Bright Cyan
9. #define OPTIONS\_COLOR "\033[92m" // Light Green
10. #define INPUT\_COLOR "\033[93m" // Bright Yellow
11. #define BRIGHT\_MAGENTA "\033[95m" // Magenta
12. #define RESET "\033[0m"
13. #define PASS "77777789"
14. int user\_type = 0;
15. typedef struct DoB{
16. int date;
17. int month;
18. int year;
19. }DoB;
20. typedef struct student{
21. char RollNo[10];
22. char name[20];
23. char gender[10];
24. DoB dob;
25. char phonenum[11];
26. char address[50];
27. }student;
28. void Menu();
29. void Menu\_student();
30. void Menu\_admin();
31. void add();
32. void view();
33. void search();
34. void edit();
35. void statistics();
36. void add(){
37. student s;
38. system("cls");
39. printf(HEADER\_COLOR "\n\t<---Enter New Student Data--->\n\n" RESET);
40. printf(INPUT\_COLOR "Enter the Roll No. of Student: " RESET);
41. scanf("%[^\n]",s.RollNo);
42. getchar();
43. printf(INPUT\_COLOR "Enter the Name of Student: " RESET);
44. scanf("%[^\n]",s.name);
45. getchar();
46. printf(INPUT\_COLOR "Enter Gender of Student: " RESET);
47. scanf("%[^\n]",s.gender);
48. getchar();
49. printf(INPUT\_COLOR "Enter the Date of Birth of Student (dd mm yyyy): " RESET);
50. scanf("%d %d %d", &s.dob.date, &s.dob.month, &s.dob.year);
51. getchar();
52. printf(INPUT\_COLOR "Enter the Phone No. of Student: " RESET);
53. scanf("%[^\n]",s.phonenum);
54. getchar();
55. printf(INPUT\_COLOR "Enter Address of Student: " RESET);
56. scanf("%[^\n]",s.address);
57. getchar();
58. FILE\* pf;
59. pf = fopen("data.txt","a");
60. if(pf == NULL) {
61. printf(ERROR\_COLOR "Error: Unable to open the file.\n" RESET);
62. return;
63. }
64. fwrite(&s,sizeof(student),1,pf);
65. fclose(pf);
66. printf("Enter any key to continue: ");
67. char ch;
68. scanf("%c",&ch);
69. if(ch != '\0'){
70. system("cls");
71. Menu\_admin();
72. }
73. }
74. void view(){
75. system("cls");
76. printf(HEADER\_COLOR "\n\t<---View All Students Data--->\n\n" RESET);
77. student s;
78. FILE \*pf;
79. pf = fopen("data.txt","r");
80. if(pf == NULL) {
81. printf(ERROR\_COLOR "Error: Unable to open the file.\n" RESET);
82. return;
83. }
84. while(fread(&s, sizeof(student), 1, pf)){
85. printf(OPTIONS\_COLOR "Name: " BRIGHT\_MAGENTA "%s\n",s.name);
86. printf(OPTIONS\_COLOR "Roll No: " BRIGHT\_MAGENTA "%s\n",s.RollNo);
87. printf(OPTIONS\_COLOR "Gender: " BRIGHT\_MAGENTA "%s\n",s.gender);
88. printf(OPTIONS\_COLOR "Date of Birth: " BRIGHT\_MAGENTA "%02d/%02d/%04d\n",s.dob.date,s.dob.month,s.dob.year);
89. printf(OPTIONS\_COLOR "Phone Number: " BRIGHT\_MAGENTA "%s\n",s.phonenum);
90. printf(OPTIONS\_COLOR "Address: " BRIGHT\_MAGENTA "%s\n\n" RESET,s.address);
91. }
92. fclose(pf);
93. printf("Enter any key to continue: ");
94. char ch;
95. scanf("%c",&ch);
96. if(ch != '\0'){
97. system("cls");
98. if(user\_type == 1){
99. Menu\_admin();
100. }
101. else if(user\_type == 2){
102. Menu\_student();
103. }
104. }
105. }
106. void search(){
107. system("cls");
108. printf(HEADER\_COLOR "\n\t<---Search a Student Data--->\n\n" RESET);
109. char roll[10];
110. printf(INPUT\_COLOR "Enter the Roll No. of Student: " RESET);
111. scanf("%[^\n]",roll);
112. getchar();
113. student s;
114. FILE \*pf;
115. pf = fopen("data.txt","r");
116. if(pf == NULL) {
117. printf(ERROR\_COLOR "Error: Unable to open the file.\n" RESET);
118. return;
119. }
120. int found = 0;
121. while(fread(&s, sizeof(student), 1, pf)){
122. if(strcmp(roll,s.RollNo)==0){
123. printf(OPTIONS\_COLOR "Name: " BRIGHT\_MAGENTA "%s\n",s.name);
124. printf(OPTIONS\_COLOR "Roll No: " BRIGHT\_MAGENTA "%s\n",s.RollNo);
125. printf(OPTIONS\_COLOR "Gender: " BRIGHT\_MAGENTA "%s\n",s.gender);
126. printf(OPTIONS\_COLOR "Date of Birth: " BRIGHT\_MAGENTA "%02d/%02d/%04d\n",s.dob.date,s.dob.month,s.dob.year);
127. printf(OPTIONS\_COLOR "Phone Number: " BRIGHT\_MAGENTA "%s\n",s.phonenum);
128. printf(OPTIONS\_COLOR "Address: " BRIGHT\_MAGENTA "%s\n\n" RESET,s.address);
129. found = 1;
130. }
131. }
132. if(!found){
133. printf(ERROR\_COLOR "\n\nNo Student Data found with Roll No. %s\n\n" RESET,roll);
134. }
135. printf("Enter any key to continue: ");
136. char ch;
137. scanf("%c",&ch);
138. if(ch != '\0'){
139. system("cls");
140. if(user\_type == 1){
141. Menu\_admin();
142. }
143. else if(user\_type == 2){
144. Menu\_student();
145. }
146. }
147. }
148. void edit(){
149. system("cls");
150. printf(HEADER\_COLOR "\n\t<---Edit Existing Student Data--->\n\n" RESET);
151. char roll[10];
152. printf(INPUT\_COLOR "Enter the Roll No. of student you want to change: " RESET);
153. scanf("%[^\n]",roll);
154. getchar();
155. FILE \*pf1,\*pf2;
156. pf1 = fopen("data.txt","r");
157. pf2 = fopen("data2.txt","w");
159. if(pf1 == NULL) {
160. printf(ERROR\_COLOR "Error: Unable to open the Data file.\n" RESET);
161. return;
162. }
163. if(pf2 == NULL) {
164. printf(ERROR\_COLOR "Error: Unable to open the Temperory file.\n" RESET);
165. return;
166. }
168. student s;
169. int found = 0;
170. while(fread(&s, sizeof(student), 1, pf1)){
171. if(strcmp(roll,s.RollNo) != 0){
172. continue;
173. }
174. else{
175. found = 1;
176. break;
177. }
178. }
179. if(!found){
180. printf(ERROR\_COLOR "\n\nNo student found with roll number %s.\nTry Again.\n" RESET, roll);
181. remove("data2.txt");
182. edit();
183. }
184. char yn;
185. printf(INPUT\_COLOR "Do you want to Change Roll No. (Y/N): " RESET);
186. yn = getchar();
187. getchar();
188. if(yn == 'y' || yn == 'Y'){
189. printf(INPUT\_COLOR "Enter the New Roll No. of student: " RESET);
190. scanf("%[^\n]",s.RollNo);
191. getchar();
192. }
193. printf(INPUT\_COLOR "Do you want to Change Name of Student. (Y/N): " RESET);
194. yn = getchar();
195. getchar();
196. if(yn == 'y' || yn == 'Y'){
197. printf(INPUT\_COLOR "Enter the Name of Student: " RESET);
198. scanf("%[^\n]",s.name);
199. getchar();
200. }
201. printf(INPUT\_COLOR "Do you want to Change Gender. (Y/N): " RESET);
202. yn = getchar();
203. getchar();
204. if(yn == 'y' || yn == 'Y'){
205. printf(INPUT\_COLOR "Enter Gender of Student: " RESET);
206. scanf("%[^\n]",s.gender);
207. getchar();
208. }
209. printf(INPUT\_COLOR "Do you want to Change Date of Birth. (Y/N): " RESET);
210. yn = getchar();
211. getchar();
212. if(yn == 'y' || yn == 'Y'){
213. printf(INPUT\_COLOR "Enter the Date of Birth of Student (dd mm yyyy): " RESET);
214. scanf("%d %d %d", &s.dob.date, &s.dob.month, &s.dob.year);
215. getchar();
216. }
217. printf(INPUT\_COLOR "Do you want to Change Phone No. (Y/N): " RESET);
218. yn = getchar();
219. getchar();
220. if(yn == 'y' || yn == 'Y'){
221. printf(INPUT\_COLOR "Enter the Phone No. of Student: " RESET);
222. scanf("%[^\n]",s.phonenum);
223. getchar();
224. }
226. printf(INPUT\_COLOR "Do you want to Address. (Y/N): " RESET);
227. yn = getchar();
228. getchar();
229. if(yn == 'y' || yn == 'Y'){
230. printf(INPUT\_COLOR "Enter Address of Student: " RESET);
231. scanf("%[^\n]",s.address);
232. getchar();
233. }
234. rewind(pf1); // Reset the file pointer
235. student s1;
236. found = 0;
237. while(fread(&s1, sizeof(student), 1, pf1)){
238. if(strcmp(roll,s1.RollNo) != 0){
239. fwrite(&s1, sizeof(student), 1, pf2);
240. }
241. else{
242. strcpy(s1.RollNo,roll);
243. fwrite(&s, sizeof(student), 1, pf2);
244. found = 1;
245. }
246. }
247. fclose(pf1);
248. fclose(pf2);
250. if(found) {
251. if(remove("data.txt") != 0){
252. perror(ERROR\_COLOR "Error deleting data.txt" RESET);
253. }
254. if(rename("data2.txt", "data.txt") != 0){
255. perror(ERROR\_COLOR "Error renaming data2.txt" RESET);
256. }
257. else{
258. printf(OPTIONS\_COLOR "\n\nRoll No. %s has been edited successfully\n\n" RESET, roll);
259. }
260. }
261. else {
262. printf(ERROR\_COLOR "\n\nNo student found with roll number %s.\n" RESET, roll);
263. remove("data2.txt");
264. }
265. printf("press any key continue ");
266. char ch;
267. scanf("%c",&ch);
268. system("cls");
269. if(ch !='\0'){
270. Menu\_admin();
271. }
272. }
273. void delete(){
275. system("cls");
276. printf(HEADER\_COLOR "\n\t<---Delete Existing Student Data--->\n\n" RESET);
277. char roll[10];
278. printf(INPUT\_COLOR "Enter the Roll No. of student you want to delete: " RESET);
279. scanf("%[^\n]",roll);
280. char ch = getchar();
281. FILE \*pf1,\*pf2;
282. pf1 = fopen("data.txt","r");
283. pf2 = fopen("data2.txt","a");
284. if(pf1 == NULL) {
285. printf(ERROR\_COLOR "Error: Unable to open the Data file.\n" RESET);
286. return;
287. }
288. if(pf2 == NULL) {
289. printf(ERROR\_COLOR "Error: Unable to open the Temperory file.\n" RESET);
290. return;
291. }
292. student s;
293. int found = 0;
294. while(fread(&s, sizeof(student), 1, pf1)){
296. if(strcmp(roll,s.RollNo) != 0){
297. fwrite(&s, sizeof(student), 1, pf2);
298. }
299. else{
300. found = 1;
301. continue;
302. }
303. }
304. fclose(pf1);
305. fclose(pf2);
306. if(found){
307. printf(OPTIONS\_COLOR "Student with roll number %s has been deleted.\n" RESET,roll);
308. remove("data.txt");
309. rename("data2.txt", "data.txt");
310. }
311. else{
312. printf(ERROR\_COLOR "No student data found with roll number %s.\n" RESET, roll);
313. }
314. printf("press any key continue ");
315. char c;
316. scanf("%c",&c);
317. system("cls");
318. if(c !='\0'){
319. Menu\_admin();
320. }
321. }
322. void statistics(){
323. system("cls");
324. printf(HEADER\_COLOR "\n\t<---View Statistics of Students--->\n\n" RESET);
325. FILE \*pf = fopen("data.txt", "r");
326. if(pf == NULL){
327. printf(ERROR\_COLOR "Error: Unable to open the file.\n" RESET);
328. return;
329. }
330. student s[100];
331. int count = 0;
332. while(fread(&s[count], sizeof(student), 1, pf)){
333. count++;
334. }
335. fclose(pf);
336. if(count == 0){
337. printf(ERROR\_COLOR "No student data available to sort.\n" RESET);
338. return;
339. }
340. int choicesort;
341. startchoice:
342. printf(HEADER\_COLOR "Sort data by:\n" RESET);
343. printf(OPTIONS\_COLOR "1. Roll Number\n");
344. printf("2. Date of Birth\n" RESET);
345. printf(INPUT\_COLOR "Enter your choice: " RESET);
346. scanf("%d", &choicesort);
347. getchar();
348. if(choicesort == 1){
349. for (int i = 0; i < count - 1; i++) {
350. for (int j = 0; j < count - 1 - i; j++) {
351. if (strcmp(s[j].RollNo, s[j + 1].RollNo) > 0) {
352. student temp = s[j];
353. s[j] = s[j + 1];
354. s[j + 1] = temp;
355. }
356. }
357. }
358. }
359. else if(choicesort == 2){
360. for(int i = 0; i < count - 1; i++){
361. for(int j = 0; j < count - 1 - i; j++){
362. if(s[j].dob.year > s[j + 1].dob.year ||
363. (s[j].dob.year == s[j + 1].dob.year && s[j].dob.month > s[j + 1].dob.month) ||
364. (s[j].dob.year == s[j + 1].dob.year && s[j].dob.month == s[j + 1].dob.month && s[j].dob.date > s[j + 1].dob.date)){
365. student temp = s[j];
366. s[j] = s[j + 1];
367. s[j + 1] = temp;
368. }
369. }
370. }
371. }
372. else{
373. printf(ERROR\_COLOR "Invalid choice.\n" RESET);
374. goto startchoice;
375. return;
376. }
377. printf(HEADER\_COLOR "\nSorted Student Data:\n" RESET);
378. for (int i = 0; i < count; i++) {
379. printf(OPTIONS\_COLOR "Name: " BRIGHT\_MAGENTA" %s\n" RESET, s[i].name);
380. printf(OPTIONS\_COLOR "Roll No: " BRIGHT\_MAGENTA" %s\n" RESET, s[i].RollNo);
381. printf(OPTIONS\_COLOR "Gender: " BRIGHT\_MAGENTA" %s\n" RESET, s[i].gender);
382. printf(OPTIONS\_COLOR "Date of Birth: " BRIGHT\_MAGENTA" %02d/%02d/%04d\n" RESET, s[i].dob.date, s[i].dob.month, s[i].dob.year);
383. printf(OPTIONS\_COLOR "Phone Number: " BRIGHT\_MAGENTA" %s\n" RESET, s[i].phonenum);
384. printf(OPTIONS\_COLOR "Address: " BRIGHT\_MAGENTA" %s\n\n" RESET, s[i].address);
385. }
386. printf("Enter any key to continue: ");
387. char ch;
388. scanf("%c", &ch);
389. system("cls");
390. if(user\_type == 1){
391. Menu\_admin();
392. }
393. else if(user\_type == 2){
394. Menu\_student();
395. }
396. }
397. void Menu(){
398. printf(HEADER\_COLOR "\t<--:Enter Your Mode of Access:-->\n\n" RESET);
399. printf(OPTIONS\_COLOR"1. Admin\n");
400. printf("2. Student\n" RESET);
401. int temp;
402. while(1){
403. printf(INPUT\_COLOR"Enter your mode: " RESET);
404. mode:
405. scanf("%d",&temp);
406. if(temp != 1 && temp!=2){
407. printf(ERROR\_COLOR "Enter Valid Mode: " RESET);
408. goto mode;
409. }
410. else if(temp == 2){
411. user\_type = 2;
412. break;
413. }
414. else{
415. printf(INPUT\_COLOR "Enter Password For Admin Mode: " RESET);
416. char password[10];
417. int i = 0;
418. while(1){
419. char ch = getch();
420. if(ch == '\n' || ch == '\r'){
421. password[i] = '\0';
422. break;
423. }
424. else if(ch == '\b' && i > 0){
425. printf("\b \b");
426. i--;
427. }
428. else if(isprint(ch) && i<8){
429. password[i++] = ch;
430. printf("\*");
431. }
432. }
433. if(strcmp(PASS,password) == 0){
434. user\_type = 1;
435. printf(OPTIONS\_COLOR "\nAccess Granted!\n" RESET);
436. break;
437. }
438. else{
439. printf(ERROR\_COLOR "\nINCORRECT PASSWORD" RESET);
440. printf(INPUT\_COLOR "\nEnter Mode Again: " RESET);
441. goto mode;
442. }
443. }
444. }
445. getchar();
446. printf("Enter any key to continue: ");
447. char c;
448. scanf("%c", &c);
450. if(c != '\0'){
451. system("cls");
452. if(user\_type == 1){
453. Menu\_admin();
454. }
455. else if(user\_type == 2){
456. Menu\_student();
457. }
458. }
459. }
460. void Menu\_student(){
461. printf(HEADER\_COLOR "\n\t<----MENU---->\n\n" RESET);
462. printf(OPTIONS\_COLOR "1. search Student\n");
463. printf("2. View all Student data\n");
464. printf("3. View Statistics\n");
465. printf("4. exit\n\n" RESET);
466. int choice = 0;
467. printf(INPUT\_COLOR "ENTER YOUR CHOICE: " RESET);
468. scanf("%d",&choice);
469. getchar();
470. switch(choice){
471. case 1:
472. search();
473. break;
474. case 2:
475. view();
476. break;
477. case 3:
478. statistics();
479. break;
480. case 4:
481. exit(0);
482. break;
483. default:
484. printf(ERROR\_COLOR "Invalid Choice.\n" RESET);
485. printf(INPUT\_COLOR "Enter any key to Try Again: " RESET);
486. char ch;
487. scanf("%c",&ch);
488. if(ch != '\0'){
489. system("cls");
490. Menu\_student();
491. }
492. }
493. }
494. void Menu\_admin(){
495. printf(HEADER\_COLOR "\n\t<----MENU---->\n\n" RESET);
496. printf(OPTIONS\_COLOR"1. Add Student\n");
497. printf("2. search Student\n");
498. printf("3. View all Student\n");
499. printf("4. Edit student data\n");
500. printf("5. Delete Student\n");
501. printf("6. View Statistics\n");
502. printf("7. exit\n\n" RESET);
503. int choice = 0;
504. printf(INPUT\_COLOR "ENTER YOUR CHOICE: " RESET);
505. scanf("%d",&choice);
506. getchar();
507. switch(choice){
508. case 1:
509. add();
510. break;
511. case 2:
512. search();
513. break;
514. case 3:
515. view();
516. break;
517. case 4:
518. edit();
519. break;
520. case 5:
521. delete();
522. break;
523. case 6:
524. statistics();
525. break;
526. case 7:
527. exit(0);
528. break;
529. default:
530. printf(ERROR\_COLOR "Invalid Choice.\n" RESET);
531. printf(INPUT\_COLOR "Enter any key to Try Again: " RESET);
532. char ch;
533. scanf("%c",&ch);
534. if(ch != '\0'){
535. system("cls");
536. Menu\_admin();
537. }
538. }
539. }
540. int main(){
541. system("cls");
542. printf(HEADER\_COLOR "\tWELCOME TO STUDENT DATA MANAGEMENT PORTAL\n\n" RESET);
543. printf("Enter any key to continue: ");
544. char ch;
545. scanf("%c",&ch);
546. if(ch != '\0'){
547. system("cls");
548. Menu();
549. }
550. }

# Implementation Details

#### **Flow of the Program**

1. **Program Initialization**:
   * The program begins by displaying a main menu with options like Add Student, View Records, Edit, Delete, Search, and Exit.
   * The user selects an option by entering the corresponding choice number.
2. **Menu Navigation**:
   * Based on the user’s input, the program invokes a specific function.
   * Input validation ensures the user enters valid choices and handles invalid entries gracefully.
3. **Core Functionalities**:
   * **Add Student**: Prompts the user to enter student details and appends them to a file using file handling.
   * **View Records**: Reads student details from the file and displays them in a formatted manner.
   * **Search**: Searches for a student based on specific criteria and displays matching records.
   * **Edit Details**: Finds a record, allows the user to update information, and rewrites the updated data back to the file.
   * **Delete**: Identifies the record to be removed and updates the file to exclude the deleted entry.
4. **Exit**:
   * Ends the program gracefully, saving any remaining changes.

#### **Critical Functions/Modules**

1. **File Handling Functions**:
   * **Write Function**: Appends new records to the file while preserving existing data.
   * **Read Function**: Reads all records from the file for viewing or searching.
2. **Search Function**:
   * Iterates through the file to locate specific records based on user-input criteria.
   * Returns matches and handles cases with no results.
3. **Edit Function**:
   * Uses file handling to copy existing records to a temporary file.
   * Allows modifications to the specified record before saving changes back to the main file.
4. **Delete Function**:
   * Similar to the edit function, creates a temporary file excluding the record marked for deletion.
   * Renames the temporary file to replace the old one.
5. **Validation and Formatting**:
   * Ensures data integrity by validating user inputs (e.g., valid roll numbers, proper formats).
   * Displays records in a clean and readable format for better user experience.

# Algorithm

1. **Start Program**
   * Display a welcome message and main menu options.
2. **Display Menu**
   * Show options: Add Student, View Records, Search, Edit, Delete, and Exit.
   * Prompt user for a choice.
3. **Process User Input**
   * Based on the choice, perform the corresponding action:

#### **Option 1: Add Student**

1. Prompt the user for student details (name, roll number, etc.).
2. Validate the inputs (e.g., non-empty, correct format).
3. Open the file in append mode.
4. Write the details into the file.
5. Close the file.
6. Display success message.

#### **Option 2: View Records**

1. Open the file in read mode.
2. If the file is empty, display "No records found."
3. Else, read and display each record in a formatted table.
4. Close the file.

#### **Option 3: Search for a Student**

1. Prompt the user for the search criteria (e.g., roll number, name).
2. Open the file in read mode.
3. Iterate through all records and compare with the search criteria.
4. If a match is found, display the record.
5. If no match, display "Record not found."
6. Close the file.

#### **Option 4: Edit a Record**

1. Prompt the user for the unique identifier (e.g., roll number).
2. Open the file in read mode and a temporary file in write mode.
3. Iterate through records in the main file:
   * If a record matches the identifier, prompt the user for updated details.
   * Write the updated record to the temporary file.
   * Else, copy the record unchanged to the temporary file.
4. Replace the original file with the temporary file.
5. Display success or error message.

#### **Option 5: Delete a Record**

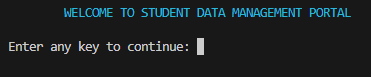
1. Prompt the user for the unique identifier (e.g., roll number).
2. Open the file in read mode and a temporary file in write mode.
3. Iterate through records:
   * If a record matches the identifier, skip writing it to the temporary file.
   * Else, copy it to the temporary file.
4. Replace the original file with the temporary file.
5. Display success or error message.

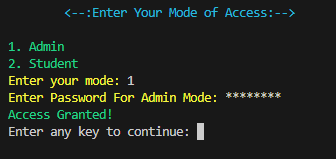
#### **Option 6: Exit**

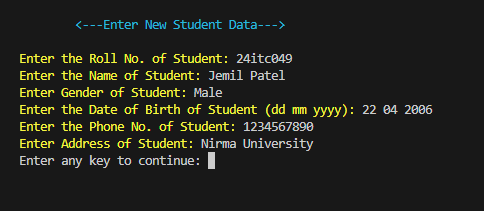
* Display a goodbye message.
* Terminate the program.

1. **Repeat Process**
   * Return to the main menu after completing an operation.
2. **End Program**

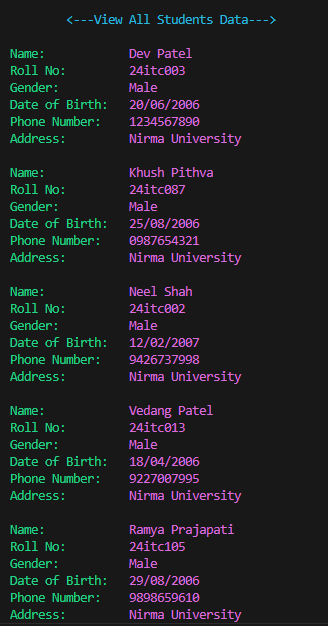
# Output ScreenShots

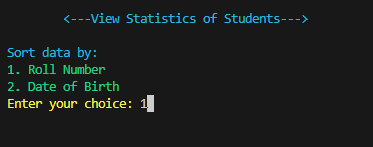


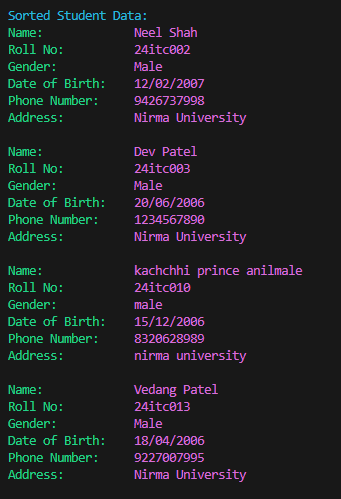


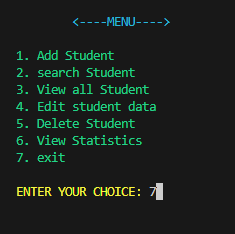


View All the Data









End of the Program….