CENTRE OF MATHEMATICAL SCIENCE, UMPSA

BCI1093 DATA STRUCTURE & ALGORITHMS

GROUP PROJECT

SEMESTER I, SESSION 2023/24

Project Title: Elderly Care and Assistance System

Lecturer: Dr. Siti Salwani Binti Yacacob

STUDENT ID	NAME
SD22002	ALMIRA DAMA BINTI SYAHNIZAM
SD22030	TUAN NURSHAFIEKA WAHIDA BINTI TUAN NADIN
SD22038	NURIN NADHIRAH IZZAH BINTI ZAINUDDIN
SD22039	NURUL SYAFIQAH NATASHA BINTI MOHD RAZI
SD22066	SHAHIRA BINTI MOHAIDEEN MEERA

TABLE OF CONTENTS

Case Study (1)

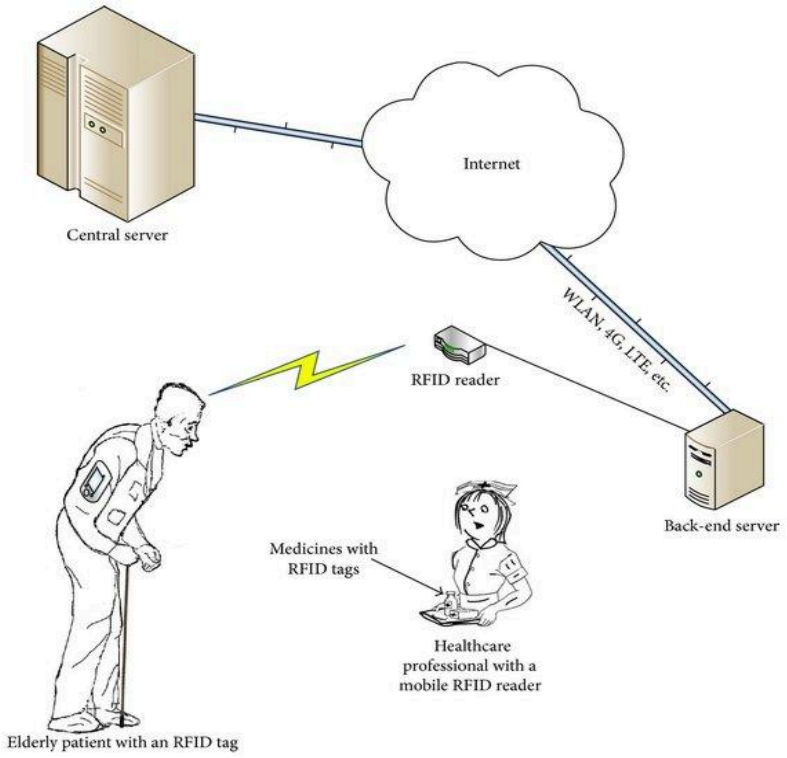
Using information and mobile technology improved elderly home care services.

The UNFPA reported a significant increase in the global elderly population, reaching 810 million in 2013 and projected to hit two billion by 2050. This poses challenges to healthcare systems, with home care services recognized as a cost-effective alternative, offering savings of 25% to 60%. In Taiwan, home healthcare services are reimbursed by the National Health Insurance program, but nurses face high work stress. This study aims to address these issues by developing a cost-effective mobile roving home care management system for small to medium-sized centers, focusing on user requirements, prototype testing, and usability assessments. The system comprises three modules: work scheduling, care service, and patient management, aiming to optimize home care schedules effectively and efficiently, utilizing advanced algorithms.

The interview and questionnaire results identified improper work scheduling as a key issue in the home care system. The manual generation of monthly schedules by the head nurse led to inefficient travel times and inadequate patient care. The existing method lacked dynamic adjustment for increased care time needs, contributing to time pressure and fragmented care. To address these challenges, the study implemented a mobile roving home care management system using AppServ, Microsoft Dynamics CRM, and Google Map API. Developed for smartphones, the system offers functionalities like care route planning and patient health evaluation

Case Study (2)

Smart Solutions in Elderly Care Facilities with RFID System and Its Integration with Wireless Sensor Networks



Taking care of the elderly is important, and technology is making it easier and safer. One smart solution is using RFID systems and wireless sensor networks in elderly care facilities. These technologies help keep track of residents, manage medications, and ensure their safety. In this context, RFID systems use special tags to monitor where residents are, making it easier to assist them quickly in emergencies. These systems also help with medication by automating tracking, reducing the chance of mistakes. Additionally, wireless sensors are used for fall detection and monitoring vital signs, contributing to overall health and safety. This integration of RFID and wireless sensors creates a comprehensive network, providing real-time information and improving overall care.

Smart solutions in elderly care facilities, incorporating RFID systems and their integration with wireless sensor networks, bring forth a range of key features aimed at enhancing the well-being and safety of elderly residents. Through the utilization of RFID tags, real-time tracking is enabled, allowing for precise monitoring of residents within the facility and ensuring swift responses during emergencies. Medication management is streamlined with RFID tags on medication containers, automating tracking and significantly reducing the likelihood of errors in administration.

Access control is strengthened through RFID-based systems, restricting unauthorized access and contributing to a secure environment. Complementing RFID, wireless sensor networks offer advanced functionalities such as fall detection, immediate alert triggering, and continuous monitoring of vital signs. The integration of RFID and wireless sensors ensures seamless data exchange, creating a cohesive network that facilitates centralized monitoring, providing caregivers with comprehensive insights into residents' health and activities. These features collectively contribute to enhanced safety, efficient medication management, and proactive health monitoring, making smart solutions integral to the overall well-being of elderly individuals in care facilities.

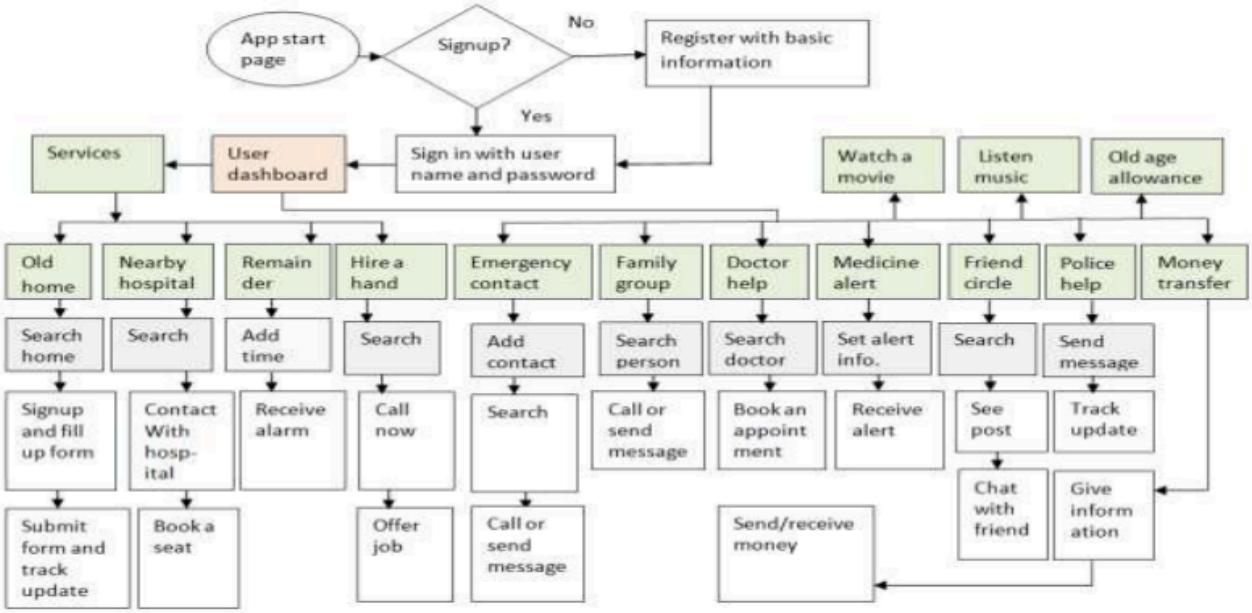
Case Study (3)

A Smart Mobile Application Featuring Elderly People Care and Assistance System for the Bangladeshi Senior Citizens.

The abstract highlights challenges faced by elderly individuals in Bangladesh due to a lack of proper care facilities and infrastructure. Currently, elderly care relies mainly on offline human support, lacking online systems for essential services. To address this, the paper proposes a smart senior citizen assistance mobile application that includes features like old home search, helping hand recruitment, old age allowance application, emergency contacts, hospital search, doctor appointments, recreation, medicine reminders, money transfer, and police help. User feedback indicates high satisfaction, with over 80% praising the application features and 75% appreciating its problem-solving capabilities, user requirement satisfaction, and economic promises.

This section provides an overview of existing works on the care and assistance of elderly individuals. Notable examples include a smartwatch-based mobile app for medication monitoring and task scheduling. Another work focuses on a handwashing assistance system for elderly individuals with dementia, utilizing IMU sensors and smartwatches. Additionally, a human-computer interaction-based mobile app offers health status reminders and medical information assistance. The section also mentions a wireless sensor network-based smart home monitoring system for fall detection and health monitoring, along with an Android app for fall detection using IoT and cloud computing.

Other works include a chatbot-based system for disease and medical assistance, a context-aware recommendation system for activity tracking and health monitoring using wearable devices and sensors, a medication intake reminder system and history checking via IoT, sensors, and mobile applications, and an EEG signal processing and microcontroller-based assistance system. Furthermore, describes a mobile banking application designed for senior citizens, considering user experience and QR code payments. Lastly, discusses an IoT device and RFID-based application, though the specific details are not provided: medicine reminder and remote monitoring system for patients.


App start page
Signup?
No
Register with basic information
Yes
Sign in with user name and password
User dashboard
Services
Watch a movie
Listen music
Old age allowance
Old home
Nearby hospital
Remain der
Hire a hand
Emergency contact
Family group
Doctor help
Medicine alert
Friend circle
Police help
Money transfer
Search home
Search
Add time
Search
Add contact
Search person
Search doctor
Set alert info.
Search
Send message
Signup and fill up form
Contact With hosp- ital
Receive alarm
Call now
Search
Call or send message
Book an appoint ment
Receive alert
See post
Track update
Chat with friend
Give inform ation
Submit form and track update
Book a seat
Offer job
Call or send message
Send/receive money

Main Menu of the elderly care and assistance system

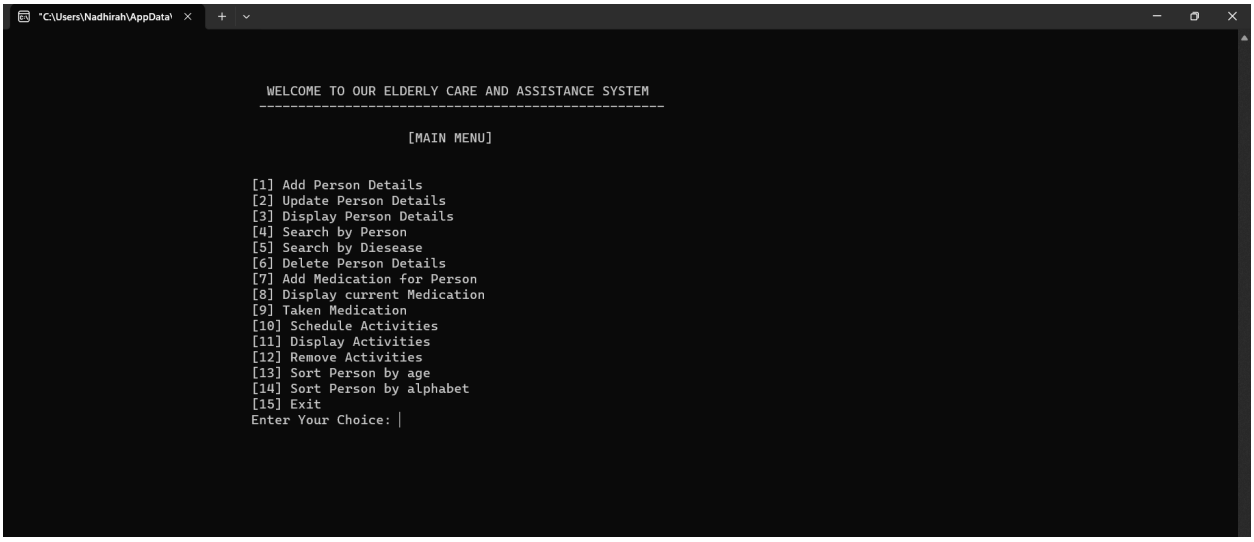


Figure 1: Main Menu of our system

Screenshots input and output of the system

1. **Add Person Details (linked list)**

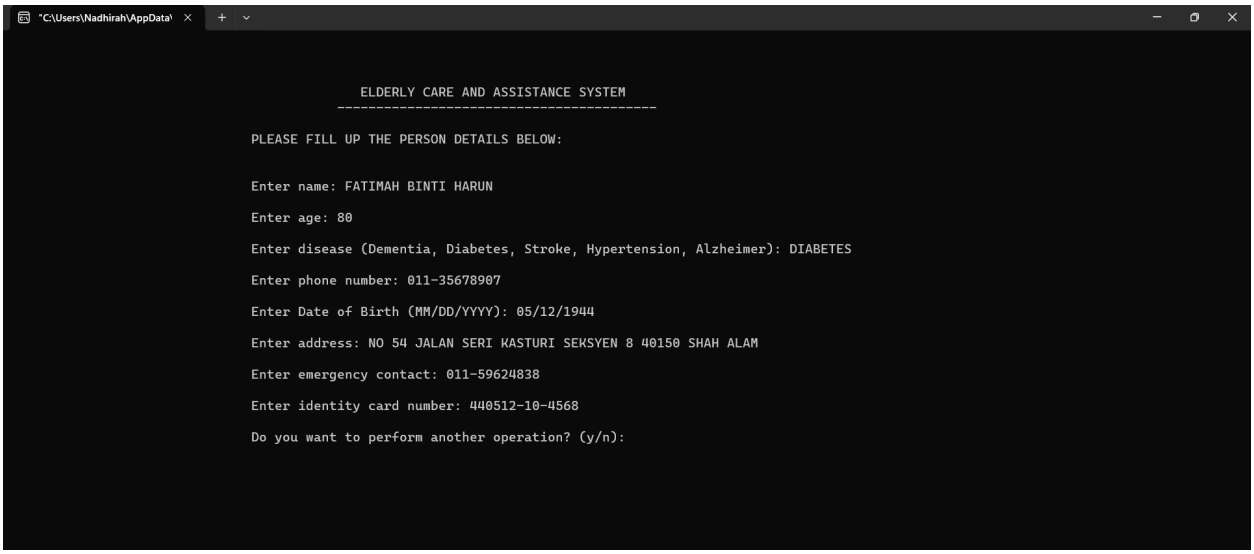


Figure 2: Input of Person Details 1

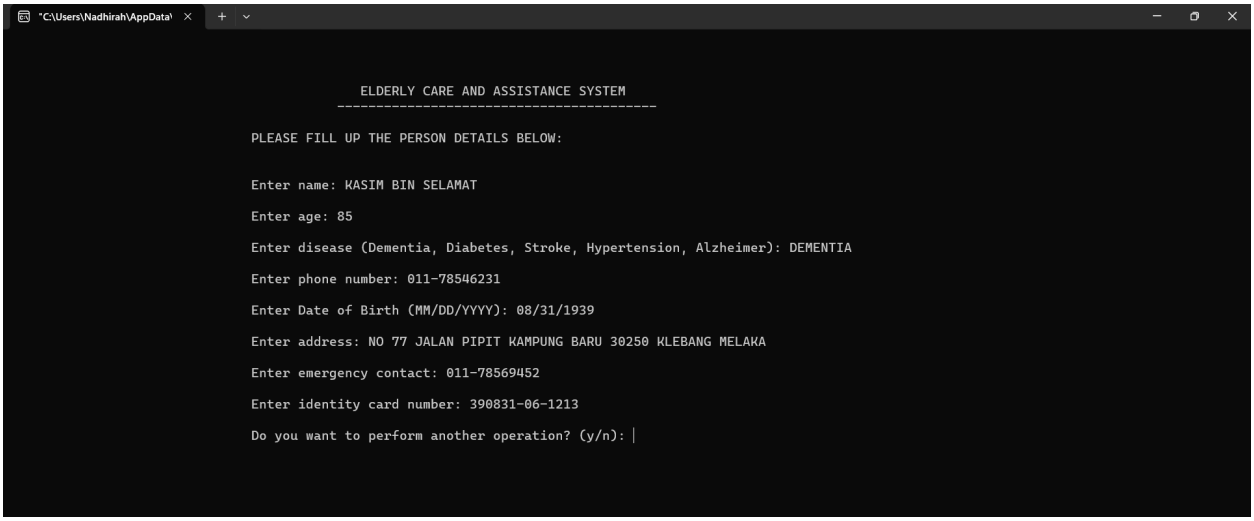


Figure 3: Input of Person Details 2

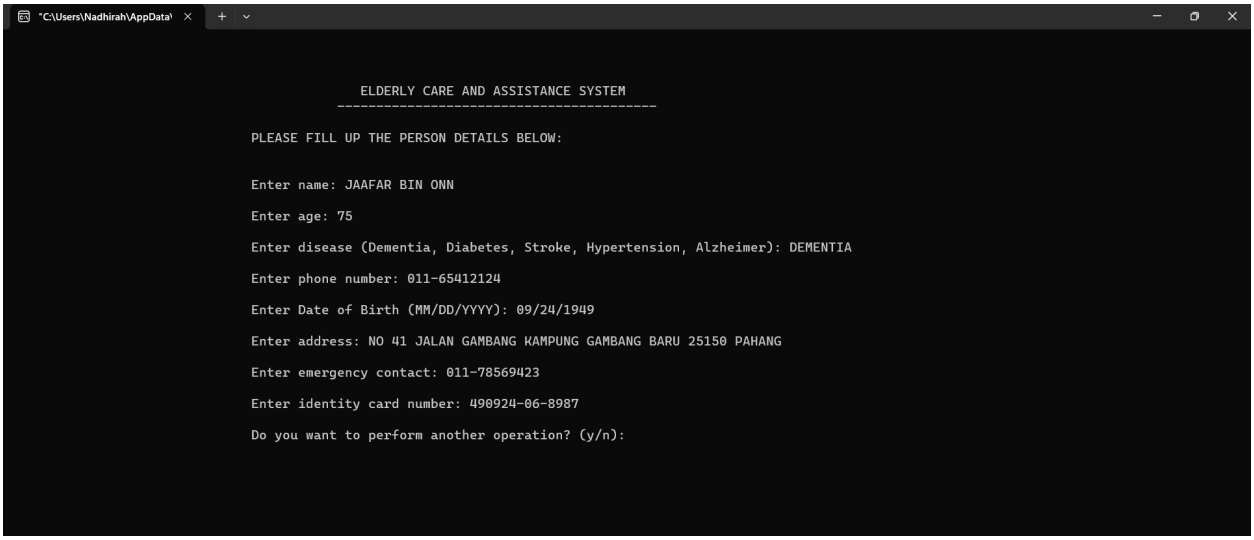


Figure 4: Input of Person Details 1

2. Display Person Details (linked list)

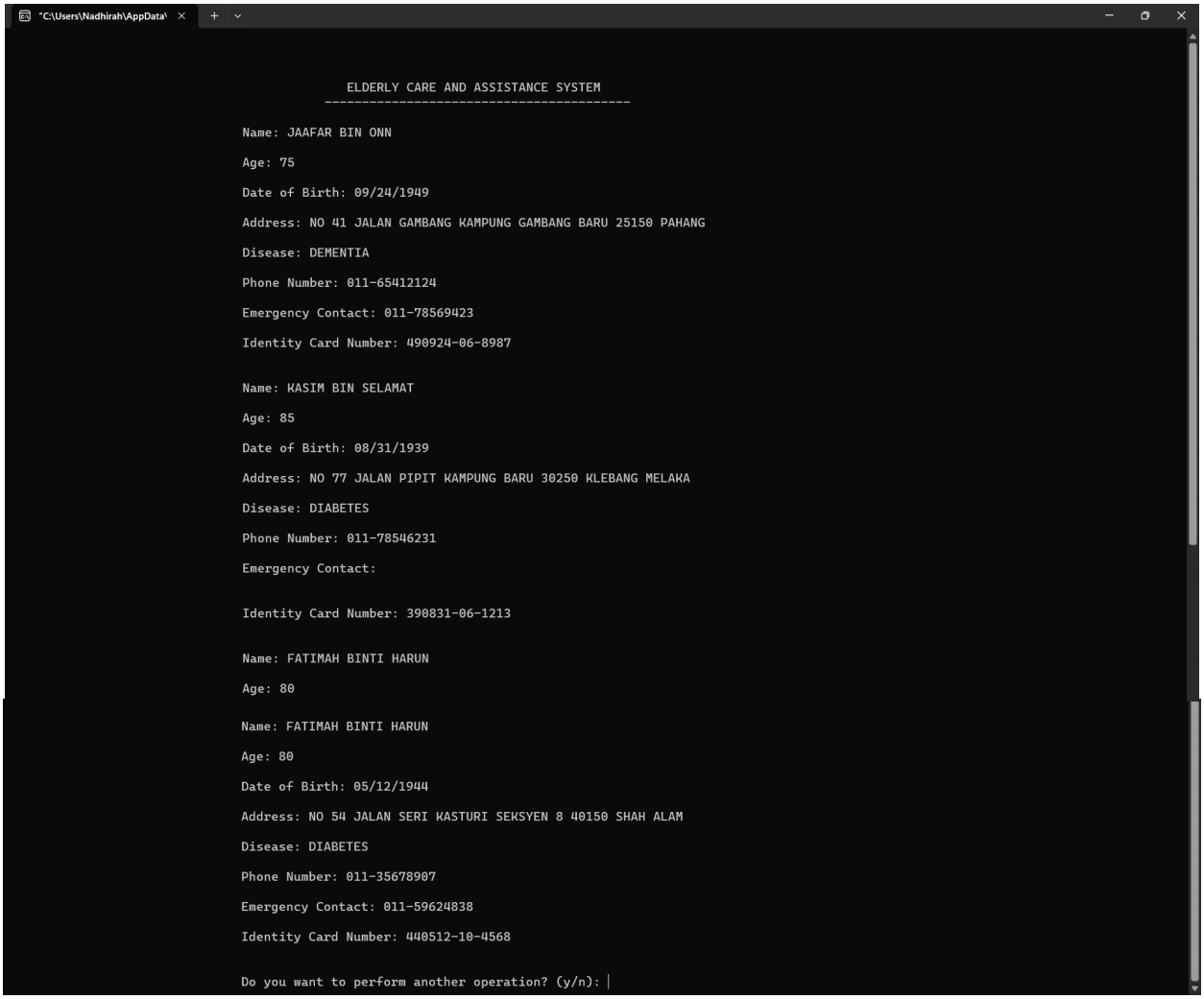


Figure 5 : Output of person details

3. Update Person Details (linked list)

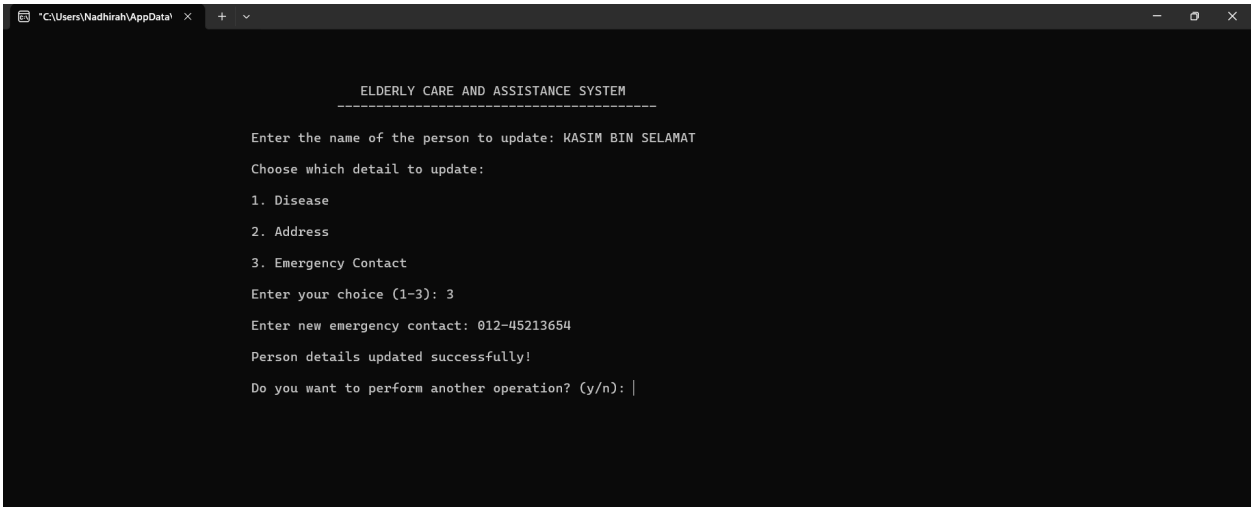


Figure 6: Updating person details for emergency contact

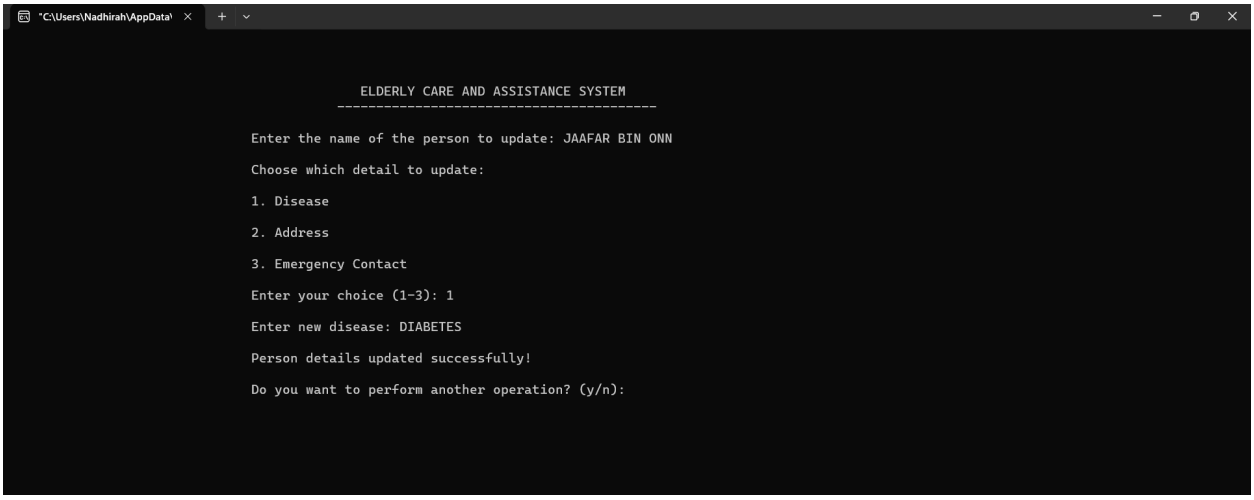


Figure 7: Updating person details for disease

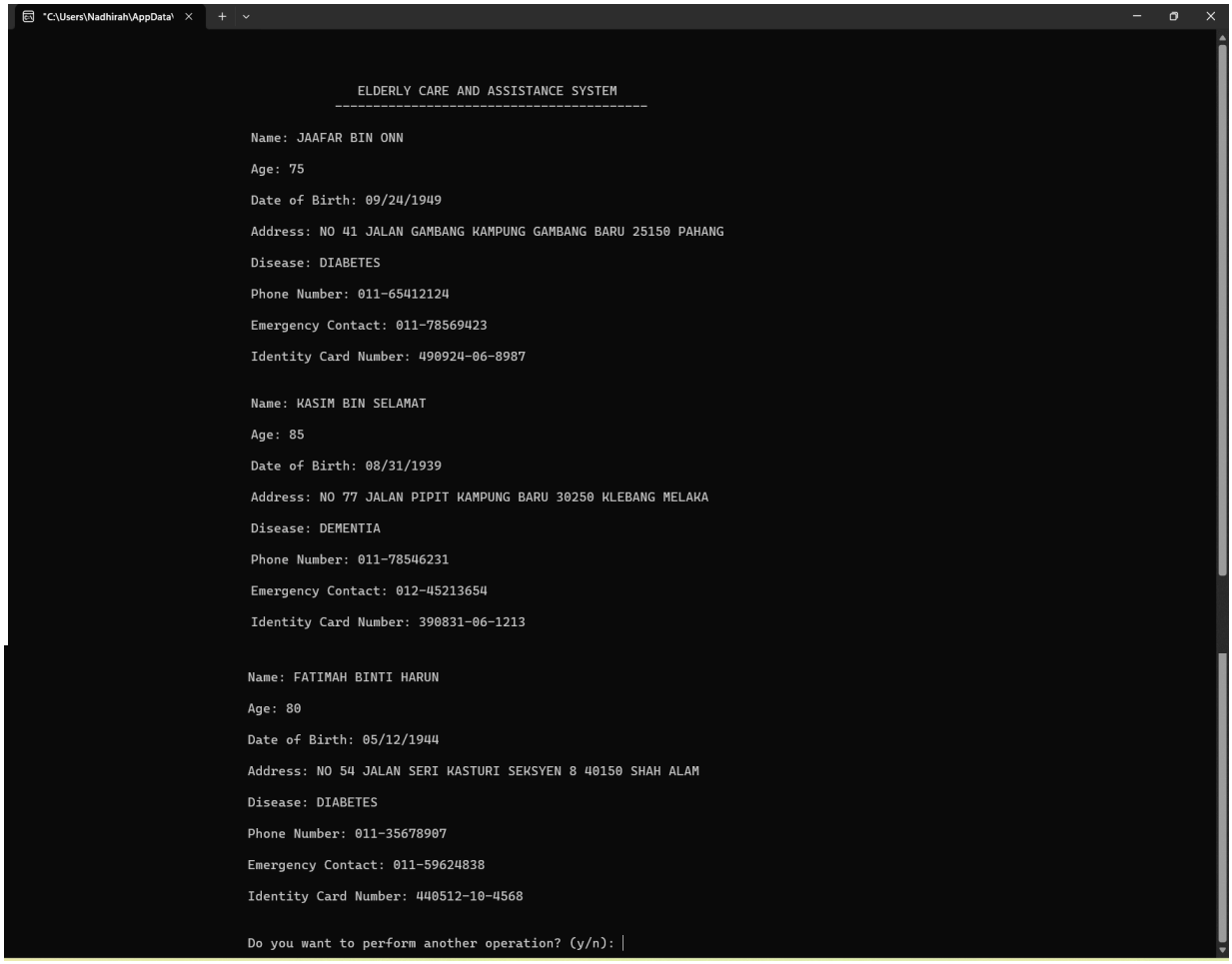


Figure 8: Displaying output of updated person details

4. Search Person Details by Name (Binary Search)

Figure 9: Input name to search person details

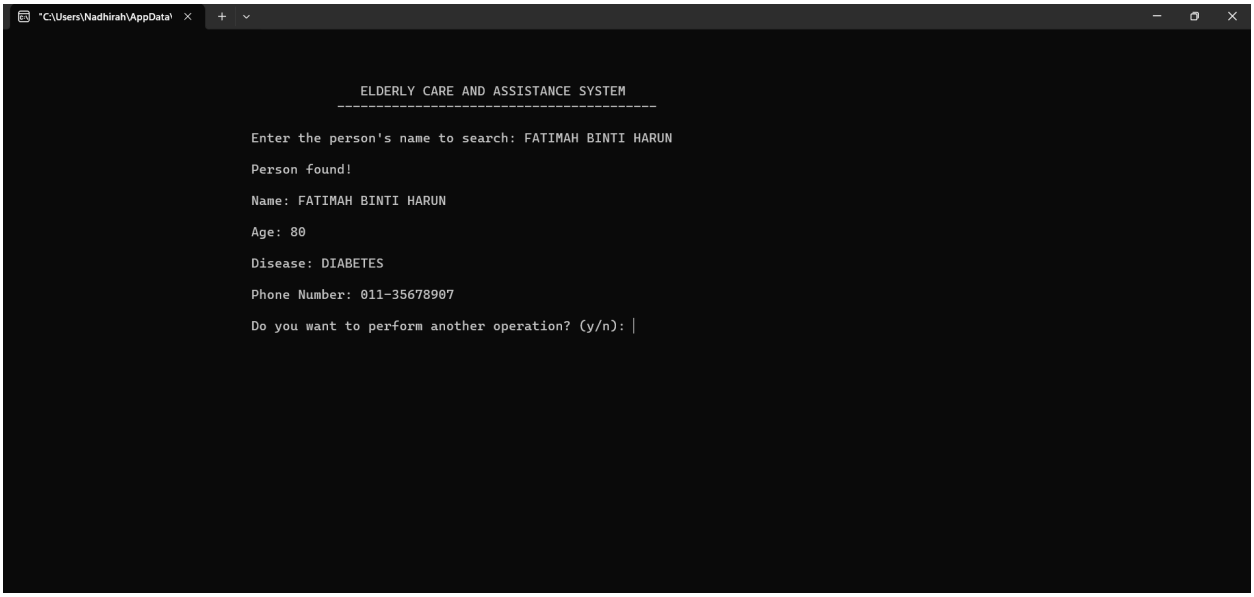


Figure 10: Output of searched person details by name

5. Search Person Details by Disease (Binary Search)

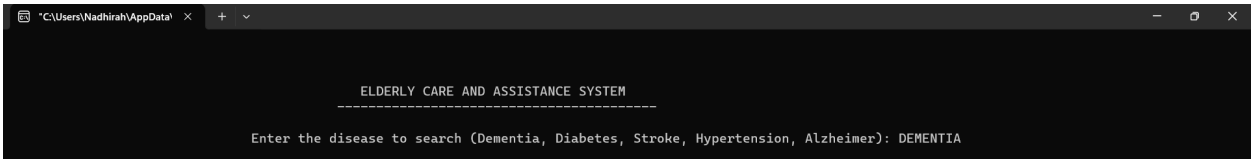


Figure 11: Input of searched person details by disease

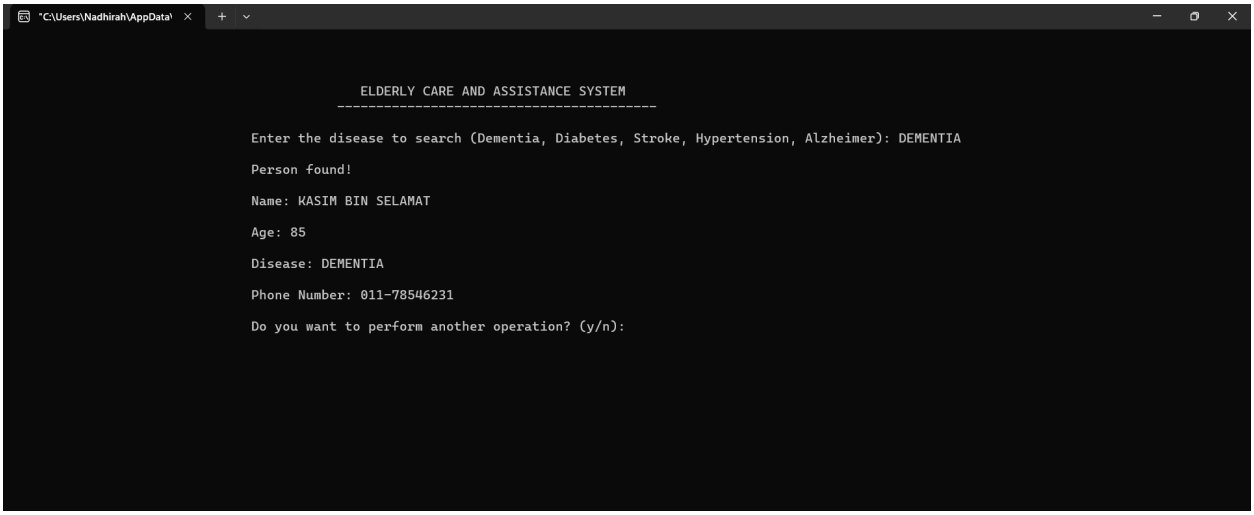


Figure 12: Output of searched person details by disease

6. Delete Person Details (linked list)

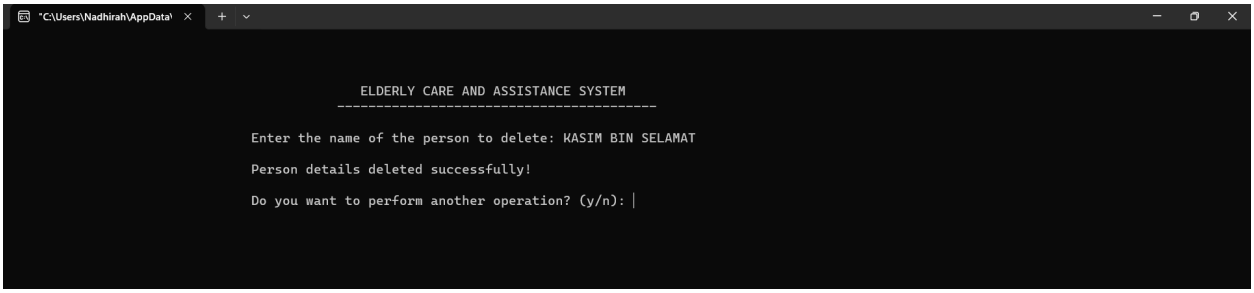


Figure 13: Input of person name to delete

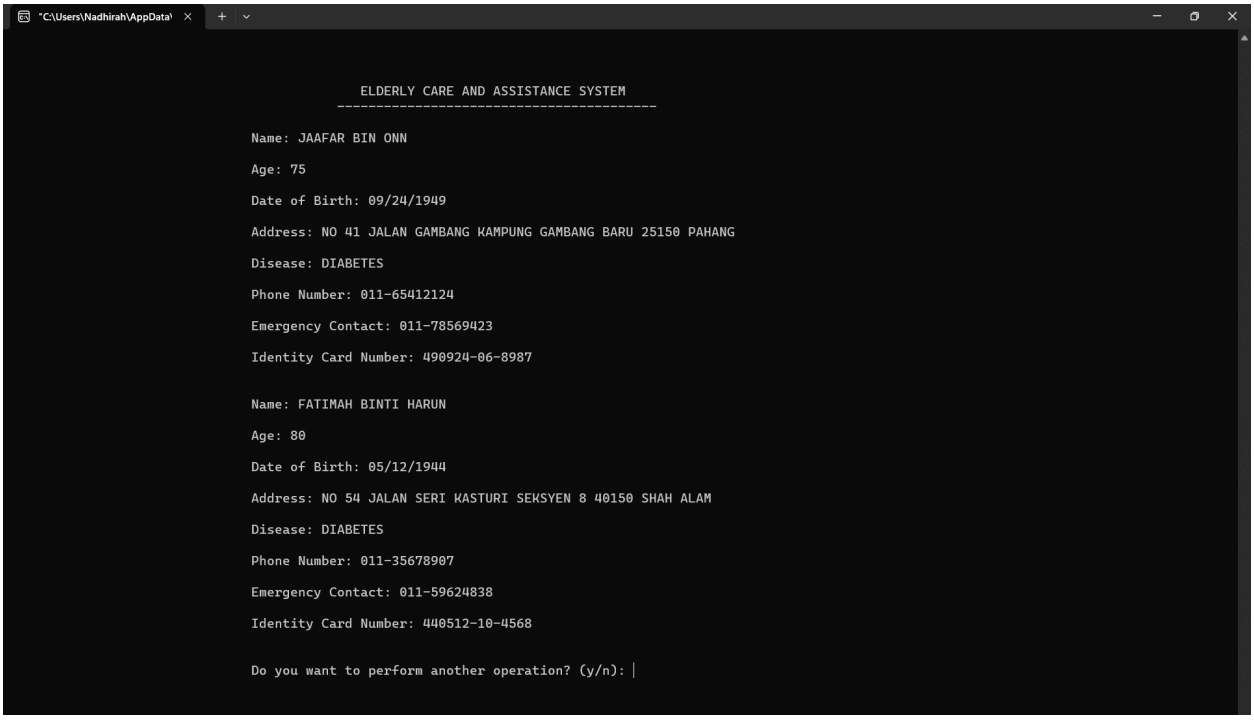


Figure 14: Displaying output after deleting person details

7. Add Medication for Person (stack)

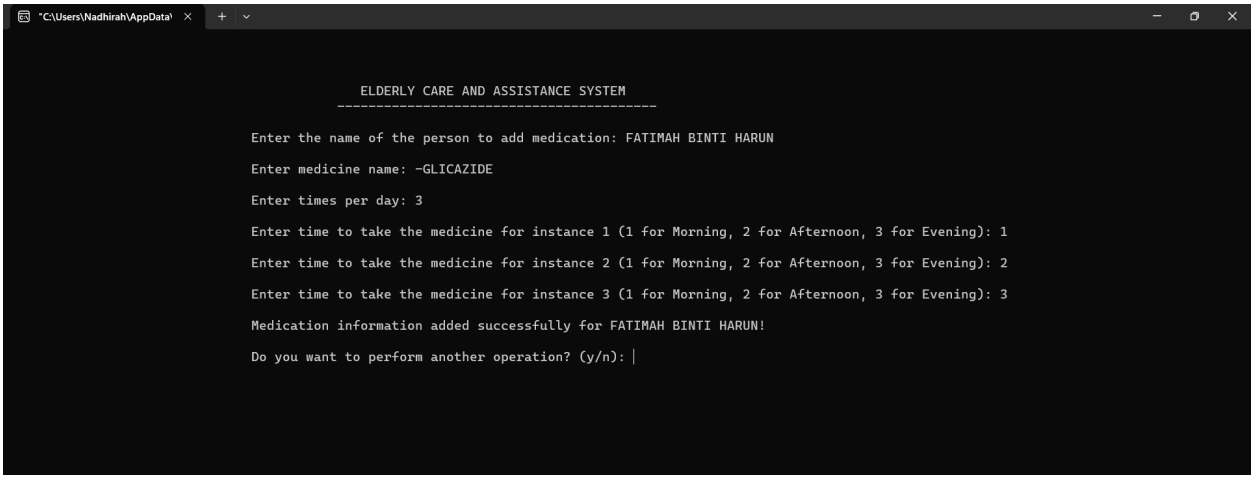


Figure 15: Input for adding medication to a person

8. Display Current Medication

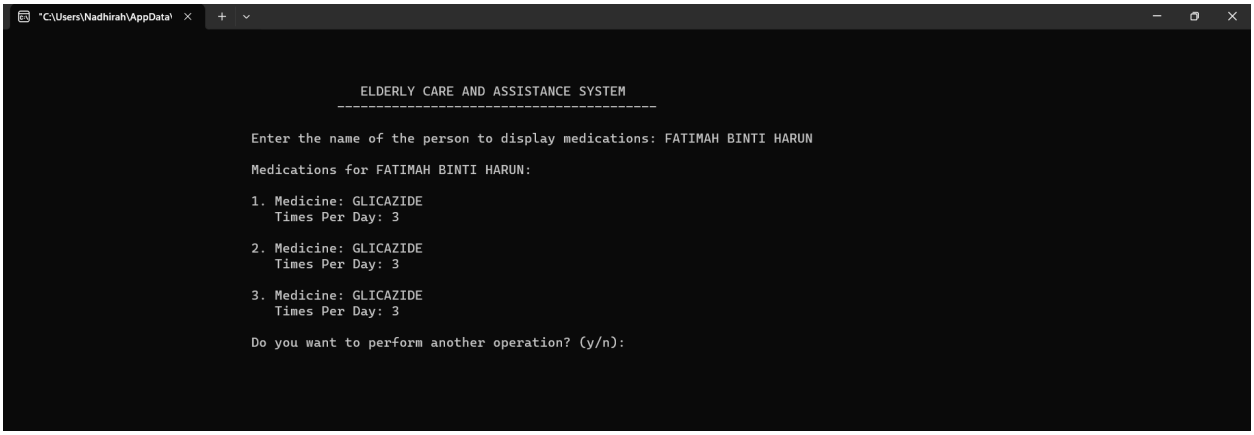


Figure 16: Displaying output for the added medication to a person

9. Taken Medication (stack)

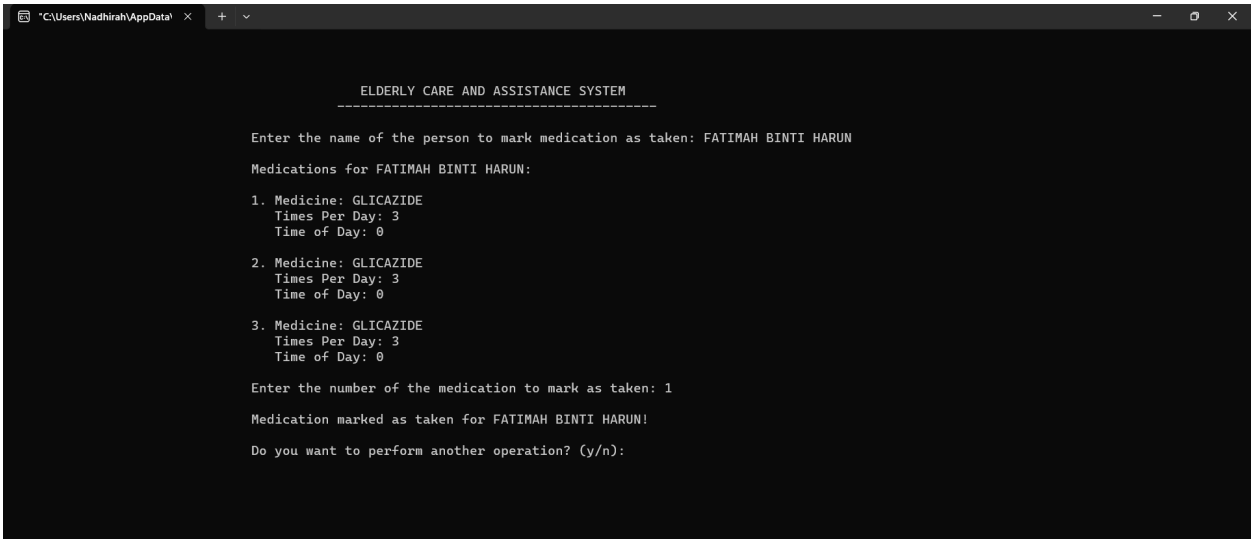


Figure 17: Input to mark medication as taken

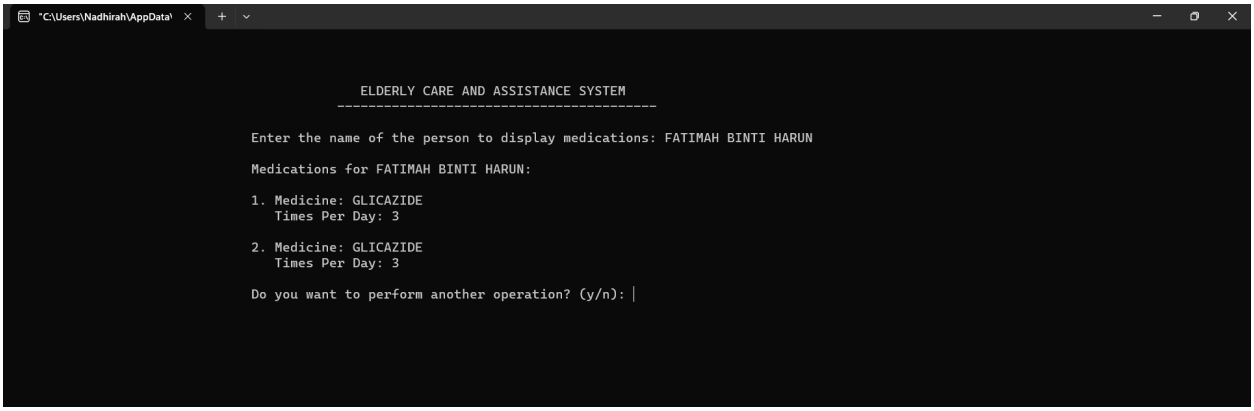


Figure 18: Displaying output after mark medication that is taken

10. Schedule Activities (Queue)

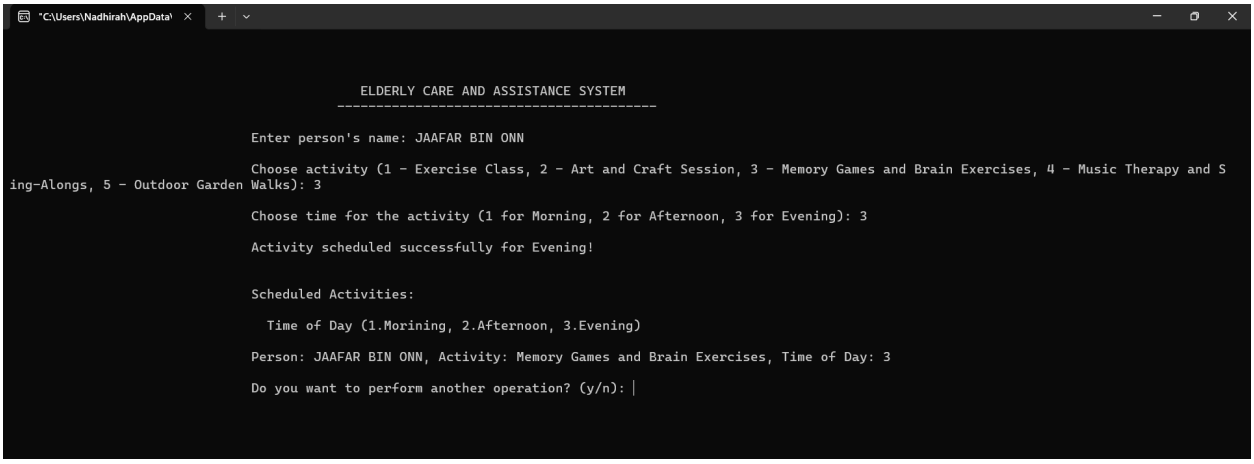


Figure 19: Input of scheduling activities to a person 1

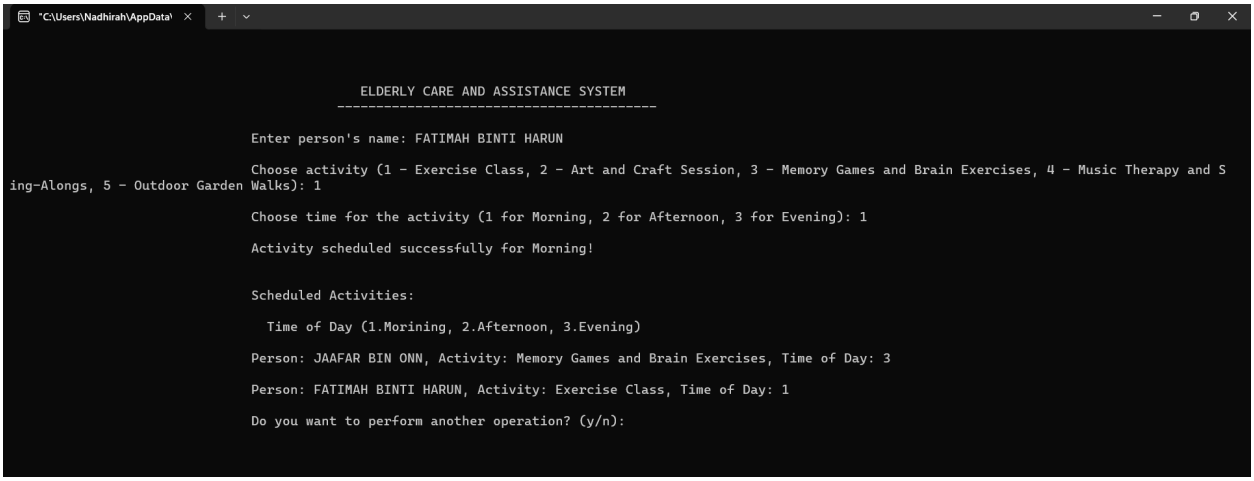


Figure 20: Input of scheduling activities to a person 2

11. Display Current Activities

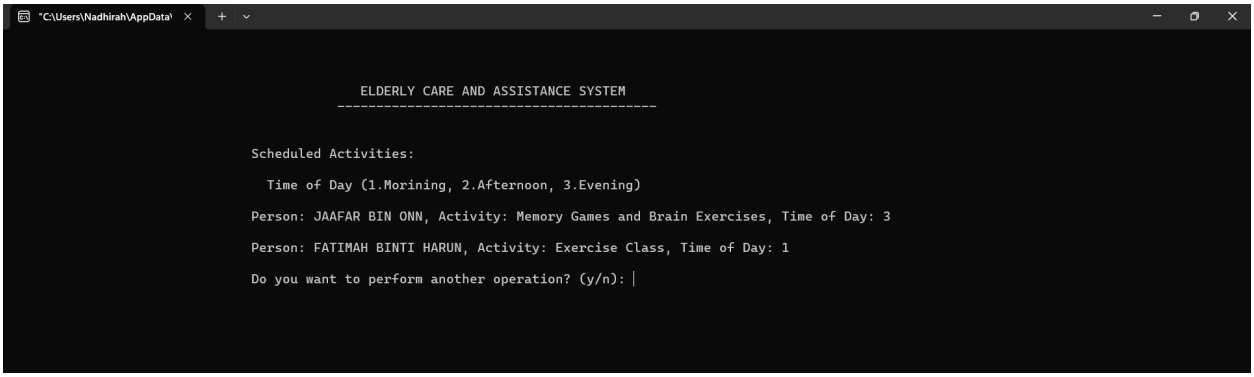


Figure 21: Displaying output of scheduled activities

12. Remove Activities (Queue)

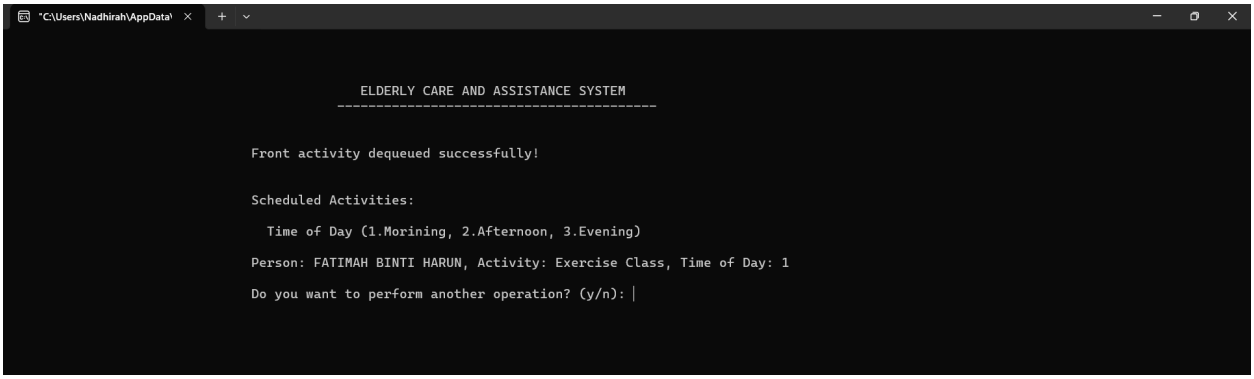


Figure 22: Removing front scheduled activity

13. Sort Person Details by Age (sort by bubble)

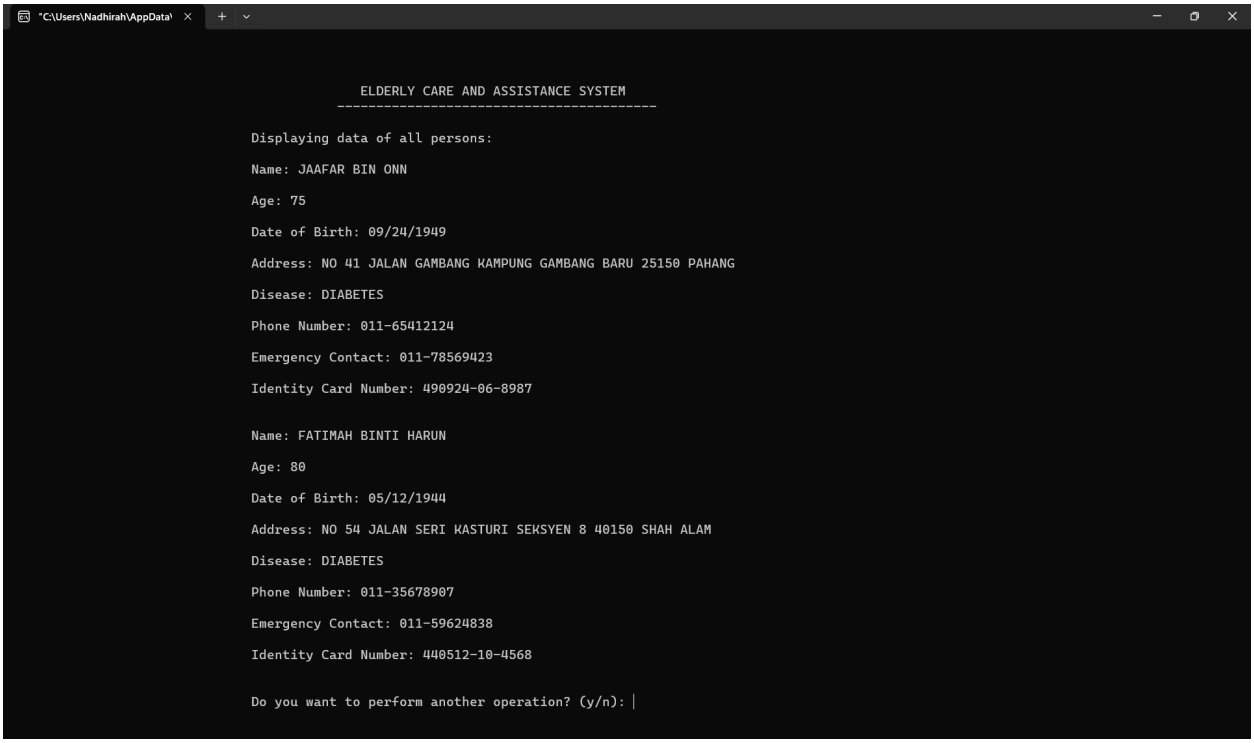


Figure 23: Displaying output of sort person details by age

14. Sort Person Details by Alphabet (sort by merge)

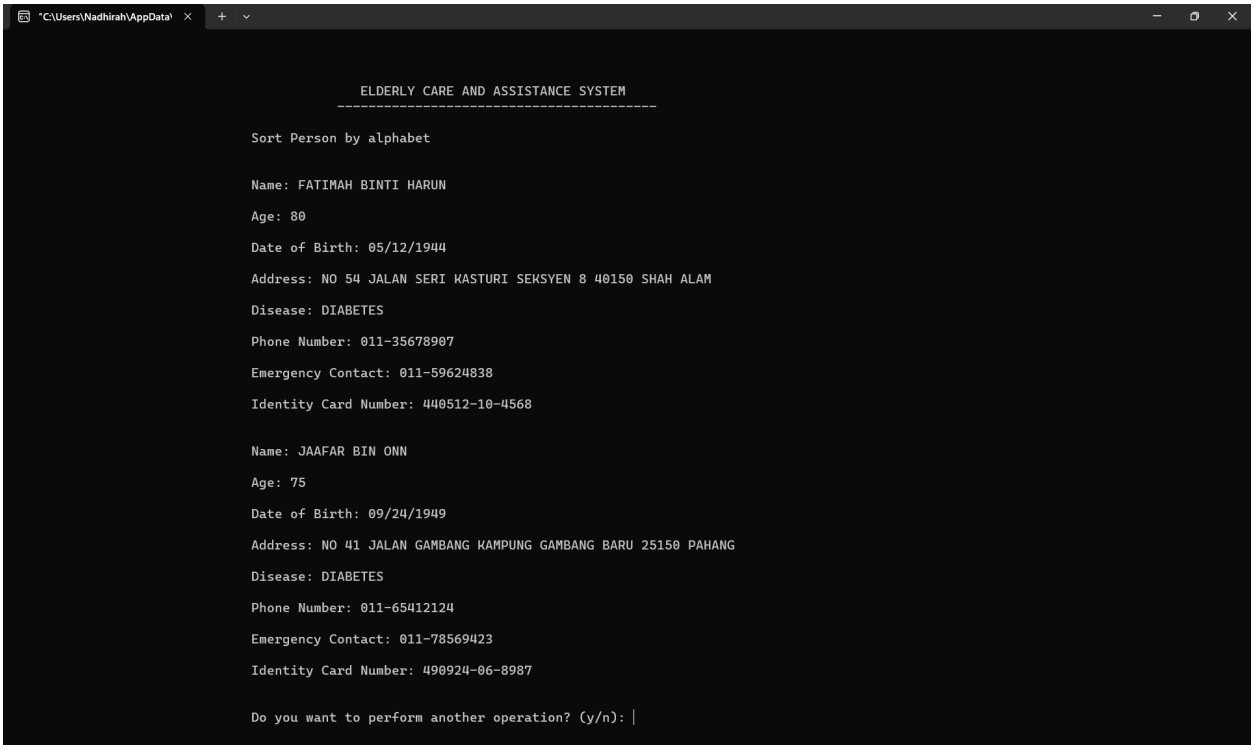


Figure 24: Displaying output of sort person details by alphabet

15. Exit

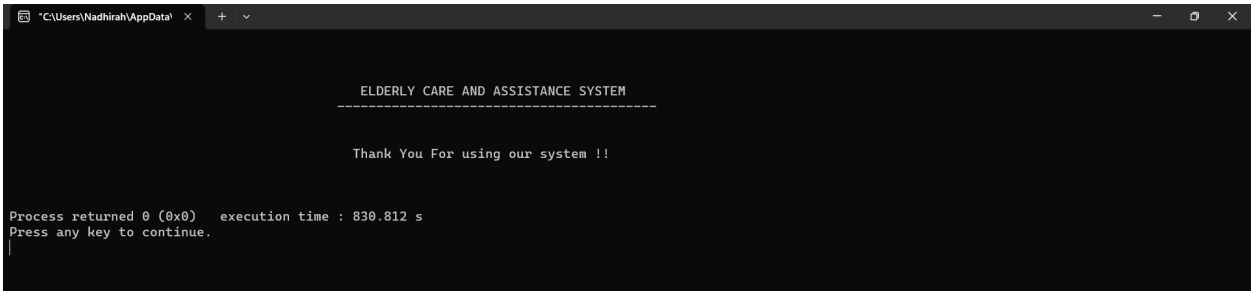


Figure 25: Displaying output when exiting the system

CODING

```c
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#include <ctype.h>

struct details // define struct for person details
{
    char name[150];
    int age;
    char disease[50];
    char phoneNum[20];
    char dob[20];  // Date of Birth
    char address[200];  // Address
    char emergencyContact[30];
    char identityCardNum[30];

};
struct Activity // define struct for activity
{
    char personName[150];
    char activityName[150];
    int timeOfDay;
    struct Activity* next;
};

struct medication //define struct for medication
{
    char personName[150];
    char medicine[50];
    float timeOfDay;
    int timesPerDay, timesTaken;

};

struct MedicationStack
{
    struct medication data;
    struct MedicationStack* next;


};

struct person
{
    struct details data;
    struct person * ptrnext;
    struct MedicationStack* medicationStack;
};

struct person *headptr,*newptr,*curptr,*prevptr;

struct Activity* front = NULL;
struct Activity* rear = NULL;

struct person *merge(struct person *left, struct person *right);
struct person *mergeSort(struct person *head);

void insertDataFromUser();
void displayData(struct person *start);
void updatePersonDetail(char name[]);
void searchByName(const char *targetName);
void searchByDisease(const char *targetDisease);
void deletePerson(char name[]);
void addMedication(struct person* personPtr);
void displayMedicationsByName(struct person* personPtr);
void markMedicationTaken(struct person* personPtr);
void scheduleActivity(char name[]);
void dequeueActivity();
void displayActivities();
void bubbleSort() ;


int main()
{
     // Initialize the head pointer to NULL
    headptr = NULL;

```

```
77:     int choice;
78:     char addMore;
79:
80:
81:     do
82:     {
83:         system("cls");
84:         printf("\n\n\n");
85:         printf("\t\t\t\t WELCOME TO OUR ELDERLY CARE AND ASSISTANCE SYSTEM \n");
86:         printf("\t\t\t\t-------------------------------------------------------\n");
87:         printf("\n");
88:         printf("\t\t\t                          [MAIN MENU]");
89:         printf("\n\n");
90:         printf("\n\t\t               [1] Add Person Details ");
91:         printf("\n\t\t               [2] Update Person Details ");
92:         printf("\n\t\t               [3] Display Person Details ");
93:         printf("\n\t\t               [4] Search by Person ");
94:         printf("\n\t\t               [5] Search by Diesease");
95:         printf("\n\t\t               [6] Delete Person Details ");
96:         printf("\n\t\t               [7] Add Medication for Person ");
97:         printf("\n\t\t               [8] Display current Medication  ");
98:         printf("\n\t\t               [9] Taken Medication ");
99:         printf("\n\t\t               [10] Schedule Activities ");
100:        printf("\n\t\t               [11] Display Activities ");
101:        printf("\n\t\t               [12] Remove Activities ");
102:        printf("\n\t\t               [13] Sort Person by age ");
103:        printf("\n\t\t               [14] Sort Person by alphabet ");
104:        printf("\n\t\t               [15] Exit ");
105:
106:        printf("\n\t\t              Enter Your Choice: ");
107:        scanf("%d",&choice);
108:
109:        switch (choice) {
110:            case 1:
111:                // Insert data
112:                system("cls");
113:                printf("\n\n\n");
114:                printf("\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");

115:                 printf("\t\t\t\t          ----------------------------------------\n");
116:                 printf("\n\t\t                PLEASE FILL UP THE PERSON DETAILS BELOW: ");
117:                 printf("\n\n");
118:                 insertDataFromUser();
119:                 break;
120:            case 2:
121:               // Update person details
122:               system("cls");
123:                printf("\n\n\n");
124:                printf("\t\t\t\t             ELDERLY CARE AND ASSISTANCE SYSTEM \n");
125:                printf("\t\t\t\t          ----------------------------------------\n");
126:               if (headptr != NULL)
127:               {
128:                char nameToUpdate[150];
129:                printf("\n\t\t               Enter the name of the person to update: ");
130:                getchar(); // consume the newline character left in the buffer
131:                fgets(nameToUpdate, sizeof(nameToUpdate), stdin);
132:                strtok(nameToUpdate, "\n");  // remove newline character
133:                updatePersonDetail(nameToUpdate);
134:                }
135:                else
136:                {
137:                printf("\n\t\t               No data to update. Please insert data first.\n");
138:                }
139:                break;
140:            case 3:
141:                // Display all person details
142:                system("cls");
143:                printf("\n\n\n");
144:                printf("\t\t\t\t             ELDERLY CARE AND ASSISTANCE SYSTEM \n");
145:                printf("\t\t\t\t          ----------------------------------------\n");
146:                if (headptr != NULL) {
147:                    displayData(headptr);
148:                } else {
149:                    printf("\n\t\t               No data to display. Please insert data first.\n");
150:                }
151:                break;
152:            case 4:
```

```
153:                system("cls");
154:                printf("\n\n\n");
155:                printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
156:                printf("\t\t\t\t\t          ----------------------------------------\n");
157:                printf("\n\t\t\t              Enter the person's name to search: ");
158:                char searchName[150];
159:                getchar();
160:                fgets(searchName, sizeof(searchName), stdin);
161:                strtok(searchName, "\n");
162:                searchByName(searchName);
163:                break;
164:            case 5:
165:                system("cls");
166:                printf("\n\n\n");
167:                printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
168:                printf("\t\t\t\t\t          ----------------------------------------\n");
169:                printf("\n\t\t\t              Enter the disease to search (Dementia, Diabetes, Stroke, Hypertension, Alzheimer): ");
170:                char searchDisease[50];
171:                getchar();
172:                fgets(searchDisease, sizeof(searchDisease), stdin);
173:                strtok(searchDisease, "\n");
174:                searchByDisease(searchDisease);
175:                break;
176:            case 6:
177:                // Delete person details
178:                system("cls");
179:                printf("\n\n\n");
180:                printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
181:                printf("\t\t\t\t\t          ----------------------------------------\n");
182:                if (headptr != NULL) {
183:                    char nameToDelete[150];
184:                    printf("\n\t\t              Enter the name of the person to delete: ");
185:                    getchar(); // consume the newline character left in the buffer
186:                    fgets(nameToDelete, sizeof(nameToDelete), stdin);
187:                    strtok(nameToDelete, "\n");  // remove newline character
188:                    deletePerson(nameToDelete);
189:                } else {
190:                    printf("\n\t\t              No data to delete. Please insert data first.\n");
191:                }
192:                break;
193:            case 7:
194:                system("cls");
195:                printf("\n\n\n");
196:                printf("\t\t\t\t\t              ELDERLY CARE AND ASSISTANCE SYSTEM \n");
197:                printf("\t\t\t\t\t            ----------------------------------------\n");
198:                if (headptr != NULL) {
199:                    char nameToAddMedication[150];
200:                    printf("\n\t\t                Enter the name of the person to add medication: ");
201:                    getchar(); // consume the newline character left in the buffer
202:                    fgets(nameToAddMedication, sizeof(nameToAddMedication), stdin);
203:                    strtok(nameToAddMedication, "\n");  // remove newline character
204:
205:                    curptr = headptr;
206:                    while (curptr != NULL && strcmp(curptr->data.name, nameToAddMedication) != 0) {
207:                        curptr = curptr->ptrnext;
208:                    }
209:
210:                    if (curptr != NULL) {
211:                        addMedication(curptr);
212:                    } else {
213:                        printf("\n\t\t              Person with name %s not found.\n", nameToAddMedication);
214:                    }
215:                } else {
216:                    printf("\n\t\t              No data to add medication. Please insert data first.\n");
217:                }
218:                break;
219:            case 8:
220:                system("cls");
221:                printf("\n\n\n");
222:                printf("\t\t\t\t\t              ELDERLY CARE AND ASSISTANCE SYSTEM \n");
223:                printf("\t\t\t\t\t            ----------------------------------------\n");
224:                if (headptr != NULL)
225:                {
226:                    char nameToDisplayMedications[150];
227:                    printf("\n\t\t              Enter the name of the person to display medications: ");
228:                    getchar(); // consume the newline character left in the buffer
```

```
229:                 fgets(nameToDisplayMedications, sizeof(nameToDisplayMedications), stdin);
230:                 strtok(nameToDisplayMedications, "\n");  // remove newline character
231:
232:                 curptr = headptr;
233:                 while (curptr != NULL && strcmp(curptr->data.name, nameToDisplayMedications) != 0) {
234:                     curptr = curptr->ptrnext;
235:                 }
236:
237:                 if (curptr != NULL) {
238:                     displayMedicationsByName(curptr);
239:                 } else {
240:                     printf("\n\t\t               Person with name %s not found.\n", nameToDisplayMedications);
241:                 }
242:             } else {
243:                 printf("\n\t\t              No data to display medications. Please insert data first.\n");
244:             }
245:             break;
246:         case 9:
247:             system("cls");
248:             printf("\n\n\n");
249:             printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
250:             printf("\t\t\t\t\t          ----------------------------------------\n");
251:             if (headptr != NULL) {
252:                 char nameToMarkMedication[150];
253:                 printf("\n\t\t              Enter the name of the person to mark medication as taken: ");
254:                 getchar(); // consume the newline character left in the buffer
255:                 fgets(nameToMarkMedication, sizeof(nameToMarkMedication), stdin);
256:                 strtok(nameToMarkMedication, "\n");  // remove newline character
257:
258:                 curptr = headptr;
259:                 while (curptr != NULL && strcmp(curptr->data.name, nameToMarkMedication) != 0) {
260:                     curptr = curptr->ptrnext;
261:                 }
262:
263:     return;     if (curptr != NULL)
264:                     markMedicationTaken(curptr);
265:                 } else {
266:                     printf("\n\t\t               Person with name %s not found.\n", nameToMarkMedication);


267:                 }
268:             } else {
269:                 printf("\n\t\t              No data to mark medication as taken. Please insert data first.\n");
270:             }
271:             break;
272:         case 10:
273:             // Scheduling activities
274:             system("cls");
275:             printf("\n\n\n");
276:             printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
277:             printf("\t\t\t\t\t          ----------------------------------------\n");
278:             if (headptr != NULL)
279:             {
280:                 char nametoSchedule[150];
281:                 printf("\n\t\t              Enter person's name: ");
282:                 getchar();
283:                 fgets(nametoSchedule, sizeof(nametoSchedule), stdin);
284:                 strtok(nametoSchedule, "\n");
285:                 scheduleActivity(nametoSchedule);
286:             }
287:             else
288:             {
289:                 printf("\n\t\t              No data to schedule activity. Please insert data first.\n");
290:             }
291:             break;
292:         case 11:
293:             system("cls");
294:             printf("\n\n\n");
295:             printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
296:             printf("\t\t\t\t\t          ----------------------------------------\n");
297:             displayActivities();
298:             break;
299:         case 12:
300:             system("cls");
301:             printf("\n\n\n");
302:             printf("\t\t\t\t\t            ELDERLY CARE AND ASSISTANCE SYSTEM \n");
303:             printf("\t\t\t\t\t          ----------------------------------------\n");
304:             dequeueActivity();
```

```
305:                 break;
306:             case 13:
307:                 system("cls");
308:                 printf("\n\n\n");
309:                 printf("\t\t\t\t             ELDERLY CARE AND ASSISTANCE SYSTEM \n");
310:                 printf("\t\t\t\t          ----------------------------------------\n");
311:                 bubbleSort();
312:                 break;
313:             case 14:
314:                 system("cls");
315:                 printf("\n\n\n");
316:                 printf("\t\t\t\t             ELDERLY CARE AND ASSISTANCE SYSTEM \n");
317:                 printf("\t\t\t\t          ----------------------------------------\n");
318:                 printf("\n\t\t                Sort Person by alphabet ");
319:                 printf("\n\n");
320:
321:                  if (headptr != NULL)
322:                 {
323:                     headptr = mergeSort(headptr);
324:         return;     displayData
325:                 } else {
326:                     printf("\n\t\t               No data to sort. Please insert data first.\n");
327:                 }
328:                 break;
329:             case 15:
330:                 system("cls");
331:                 printf("\n\n\n");
332:                 printf("\t\t\t\t             ELDERLY CARE AND ASSISTANCE SYSTEM \n");
333:                 printf("\t\t\t\t          ----------------------------------------\n");
334:                 printf("\n\n\t\t\t\t            Thank You For using our system !! ");
335:                 printf("\n\n\n");
336:                 exit(0);
337:                 break;
338:             default:
339:                 printf("\n\t\t                Invalid choice\n");
340:                 break;
341:         }
342:
```

```
343:         // Ask the user if they want to continue
344:         if (choice != 15) {
345:             printf("\n\t\t               Do you want to perform another operation? (y/n): ");
346:             scanf(" %c", &addMore);  // note the space before %c to consume the newline character
347:             getchar();  // consume the newline character left in the buffer
348:         }
349:
350:     } while (choice != 15 && (addMore == 'y' || addMore == 'Y'));
351:
352:
353:         if(addMore=='N'||addMore=='n')
354:         {
355:             system("cls");
356:             printf("\n\n\n");
357:             printf("\t\t\t\t             ELDERLY CARE AND ASSISTANCE SYSTEM \n");
358:             printf("\t\t\t\t          ----------------------------------------\n");
359:             printf("\n\n\t\t\t\t            Thank You For using our system !! ");
360:             printf("\n\n\n");
361:             exit(0);
362:         }
363:
364:     // Clean up: free allocated memory (you may want to do this in a separate function)
365:     curptr = headptr;
366:     while (curptr != NULL) {
367:         prevptr = curptr;
368:         curptr = curptr->ptrnext;
369:         free(prevptr);
370:     }
371:
372: return 0;
373: }
374:
375: // Function to insert a new node with user-input data
376:
377: void insertDataFromUser()
378: {
379:     newptr = (struct person *)malloc(sizeof(struct person));
380:     if (newptr == NULL) {
```

```
381:         printf("Memory allocation failed.\n");
382:         exit(1);
383:     }
384:
385:     printf("\n\t\t               Enter name: ");
386:     getchar();
387:     fgets(newptr->data.name, sizeof(newptr->data.name), stdin);
388:     strtok(newptr->data.name, "\n");
389:
390:     printf("\n\t\t               Enter age: ");
391:     scanf("%d", &newptr->data.age);
392:
393:     printf("\n\t\t               Enter disease (Dementia, Diabetes, Stroke, Hypertension, Alzheimer): ");
394:     getchar();
395:     fgets(newptr->data.disease, sizeof(newptr->data.disease), stdin);
396:     strtok(newptr->data.disease, "\n");
397:
398:     printf("\n\t\t               Enter phone number: ");
399:     fgets(newptr->data.phoneNum, sizeof(newptr->data.phoneNum), stdin);
400:     strtok(newptr->data.phoneNum, "\n");
401:
402:     printf("\n\t\t               Enter Date of Birth (MM/DD/YYYY): ");
403:     fgets(newptr->data.dob, sizeof(newptr->data.dob), stdin);
404:     strtok(newptr->data.dob, "\n");
405:
406:     printf("\n\t\t               Enter address: ");
407:     fgets(newptr->data.address, sizeof(newptr->data.address), stdin);
408:     strtok(newptr->data.address, "\n");
409:
410:     printf("\n\t\t               Enter emergency contact: ");
411:     fgets(newptr->data.emergencyContact, sizeof(newptr->data.emergencyContact), stdin);
412:     strtok(newptr->data.emergencyContact, "\n");
413:
414:     printf("\n\t\t               Enter identity card number: ");
415:     fgets(newptr->data.identityCardNum, sizeof(newptr->data.identityCardNum), stdin);
416:     strtok(newptr->data.identityCardNum, "\n");
417:
418:     newptr->medicationStack = NULL;


419:
420:     newptr->ptrnext = headptr;
421:     headptr = newptr;
422: }
423:
424:
425: // Function to display data of each person in the linked list
426: void displayData(struct person *start)
427: {
428:     curptr = start;
429:
430:     // Traverse the linked list and print each person's data
431:     while (curptr != NULL) {
432:         printf("\n\t\t               Name: %s\n", curptr->data.name);
433:         printf("\n\t\t               Age: %d\n", curptr->data.age);
434:         printf("\n\t\t               Date of Birth: %s\n", curptr->data.dob);
435:         printf("\n\t\t               Address: %s\n", curptr->data.address);
436:         printf("\n\t\t               Disease: %s\n", curptr->data.disease);
437:         printf("\n\t\t               Phone Number: %s\n", curptr->data.phoneNum);
438:         printf("\n\t\t               Emergency Contact: %s\n", curptr->data.emergencyContact);
439:         printf("\n\t\t               Identity Card Number: %s\n", curptr->data.identityCardNum);
440:         printf("\n");
441:
442:         curptr = curptr->ptrnext;
443:     }
444: }
445:
446:
447: // Function to update a specific detail of a person based on their name
448: void updatePersonDetail(char name[])
449: {
450:     curptr = headptr;
451:
452:     // Traverse the linked list to find the person with the specified name
453:     while (curptr != NULL && strcmp(curptr->data.name, name) != 0) {
454:         prevptr = curptr;
455:         curptr = curptr->ptrnext;
456:     }
```

```
457:
458:     if (curptr != NULL) {
459:         // Person found, ask the user which detail to update
460:         printf("\n\t\t               Choose which detail to update:\n");
461:         printf("\n\t\t               1. Disease\n");
462:         printf("\n\t\t               2. Address\n");
463:         printf("\n\t\t               3. Emergency Contact\n");
464:         printf("\n\t\t               Enter your choice (1-3): ");
465:
466:         int detailChoice;
467:         scanf("%d", &detailChoice);
468:
469:         // Consume the newline character in the buffer
470:         getchar();
471:
472:         switch (detailChoice) {
473:             case 1:
474:                 // Update disease
475:                 printf("\n\t\t               Enter new disease: ");
476:                 fgets(curptr->data.disease, sizeof(curptr->data.disease), stdin);
477:                 strtok(curptr->data.disease, "\n");
478:                 break;
479:             case 2:
480:                 // Update address
481:                 printf("\n\t\t               Enter new Address: ");
482:                 fgets(curptr->data.address, sizeof(curptr->data.address), stdin);
483:                 strtok(curptr->data.address, "\n");
484:                 break;
485:             case 3:
486:                 // Update emergency contact
487:                 printf("\n\t\t               Enter new emergency contact: ");
488:                 fgets(curptr->data.emergencyContact, sizeof(curptr->data.emergencyContact), stdin);
489:                 strtok(curptr->data.emergencyContact, "\n");
490:                 break;
491:             default:
492:                 printf("\n\t\t               Invalid choice. No details updated.\n");
493:                 return;
494:         }

495:
496:         printf("\n\t\t               Person details updated successfully!\n");
497:     } else {
498:         printf("\n\t\t               Person not found.\n");
499:     }
500: }
501:
502:
503: void searchByName(const char *targetName)
504: {
505:     curptr = headptr;
506:     int found = 0;
507:
508:     while (curptr != NULL) {
509:         if (strcmp(curptr->data.name, targetName) == 0) {
510:             found = 1;
511:             break;
512:         }
513:         curptr = curptr->ptrnext;
514:     }
515:
516:     if (found) {
517:         printf("\n\t\t               Person found!\n");
518:         printf("\n\t\t               Name: %s\n", curptr->data.name);
519:         printf("\n\t\t               Age: %d\n", curptr->data.age);
520:         printf("\n\t\t               Disease: %s\n", curptr->data.disease);
521:         printf("\n\t\t               Phone Number: %s\n", curptr->data.phoneNum);
522:     } else {
523:         printf("\n\t\t               Person not found.\n");
524:     }
525: }
526:
527: void searchByDisease(const char *targetDisease)
528: {
529:     curptr = headptr;
530:     int found = 0;
531:
532:     while (curptr != NULL) {
```

```
533:         if (strcmp(curptr->data.disease, targetDisease) == 0) {
534:             found = 1;
535:             break;
536:         }
537:         curptr = curptr->ptrnext;
538:     }
539:
540:     if (found) {
541:         printf("\n\t\t               Person found!\n");
542:         printf("\n\t\t               Name: %s\n", curptr->data.name);
543:         printf("\n\t\t               Age: %d\n", curptr->data.age);
544:         printf("\n\t\t               Disease: %s\n", curptr->data.disease);
545:         return;"\n\t\t               Phone Number: %s\n",
546:     } else {
547:         printf("\n\t\t               Person not found.\n");
548:     }
549: }
550:
551: void deletePerson(char name[])
552: {
553:     curptr = headptr;
554:     struct person *temp;
555:
556:     // Special case: deleting the head node
557:     if (curptr != NULL && strcmp(curptr->data.name, name) == 0) {
558:         headptr = curptr->ptrnext;
559:         free(curptr);
560:         printf("\n\t\t               Person details deleted successfully!\n");
561:         return;
562:     }
563:
564:     // Search for the person to delete
565:     while (curptr != NULL && strcmp(curptr->data.name, name) != 0) {
566:         prevptr = curptr;
567:         curptr = curptr->ptrnext;
568:     }
569:
570:     // If the person is found, delete them


571:     if (curptr != NULL) {
572:         prevptr->ptrnext = curptr->ptrnext;
573:         free(curptr);
574:         printf("\n\t\t               Person details deleted successfully!\n");
575:     } else {
576:         printf("\n\t\t               Person not found.\n");
577:     }
578: }
579:
580: // Function to display data of all persons in the linked list
581: void displaydetails()
582: {
583:     printf("\n\t\t               Displaying data of all persons:\n");
584:     displayData(headptr);
585: }
586:
587: // Modify the function definition for the addMedication function
588: void addMedication(struct person* personPtr)
589: {
590:     struct medication newMedication;
591:     printf("\n\t\t               Enter medicine name: ");
592:     getchar(); // consume the newline character left in the buffer
593:     fgets(newMedication.medicine, sizeof(newMedication.medicine), stdin);
594:     strtok(newMedication.medicine, "\n");
595:
596:     printf("\n\t\t               Enter times per day: ");
597:     scanf("%d", &newMedication.timesPerDay);
598:
599:     if (newMedication.timesPerDay > 1)
600:      {
601:         for (int i = 0; i < newMedication.timesPerDay; ++i) {
602:             printf("\n\t\t               Enter time to take the medicine for instance %d (1 for Morning, 2 for Afternoon, 3 for Evening
603:             scanf("%d", &newMedication.timeOfDay);
604:
605:             // Create a new node for the medication
606:             struct MedicationStack* newNode = (struct MedicationStack*)malloc(sizeof(struct MedicationStack));
607:
608:             if (newNode == NULL) {
```

```
609:                 printf("\n\n\t\t              Memory allocation failed!\n");
610:                 exit(1);
611:             }
612:
613:             newNode->data = newMedication;
614:             newNode->next = personPtr->medicationStack;
615:             personPtr->medicationStack = newNode;
616:         }
617:     } else {
618:         printf("\n\t\t              Enter time to take the medicine (1 for Morning, 2 for Afternoon, 3 for Evening): ");
619:         scanf("%d", &newMedication.timeOfDay);
620:
621:         // Create a new node for the medication
622:         struct MedicationStack* newNode = (struct MedicationStack*)malloc(sizeof(struct MedicationStack));
623:
624:         if (newNode == NULL) {
625:             printf("\n\n\t\t              Memory allocation failed!\n");
626:             exit(1);
627:         }
628:
629:         newNode->data = newMedication;
630:         newNode->next = personPtr->medicationStack;
631:         personPtr->medicationStack = newNode;
632:     }
633:
634:     printf("\n\t\t              Medication information added successfully for %s!\n", personPtr->data.name);
635: }
636:
637: void displayMedicationsByName(struct person* personPtr)
638: {
639:     if (personPtr->medicationStack == NULL)
640:     {
641:         printf("\n\t\t              No medications for %s.\n", personPtr->data.name);
642:         return;
643:     }
644:
645:     printf("\n\t\t              Medications for %s:\n", personPtr->data.name);
646:


647:     struct MedicationStack* currentMedication = personPtr->medicationStack;
648:     int count = 1;
649:
650:     while (currentMedication != NULL) {
651:         printf("\n\t\t               %d. Medicine: %s\n", count, currentMedication->data.medicine);
652:         printf("\t\t                Times Per Day: %d\n", currentMedication->data.timesPerDay);
653:
654:         currentMedication = currentMedication->next;
655:         count++;  // Increment count for the next medication
656:     }
657: }
658:
659:
660: void markMedicationTaken(struct person* personPtr)
661: {
662:     if (personPtr->medicationStack == NULL) {
663:         printf("\n\t\t               No medications to mark as taken for %s.\n", personPtr->data.name);
664:         return;
665:     }
666:
667:     printf("\n\t\t              Medications for %s:\n", personPtr->data.name);
668:
669:     struct MedicationStack* currentMedication = personPtr->medicationStack;
670:     int count = 1;
671:
672:     // Display medications and let the user choose which one to mark as taken
673:     while (currentMedication != NULL) {
674:         printf("\n\t\t               %d. Medicine: %s\n", count, currentMedication->data.medicine);
675:         printf("\t\t                Times Per Day: %d\n", currentMedication->data.timesPerDay);
676:         printf("\t\t                Time of Day: %d\n", currentMedication->data.timeOfDay);
677:
678:         currentMedication = currentMedication->next;
679:         count++;
680:     }
681:
682:     int choice;
683:     printf("\n\t\t              Enter the number of the medication to mark as taken: ");
684:     scanf("%d", &choice);
```

```
685:
686:     // Remove the chosen medication node from the stack
687:     if (choice == 1) {
688:         // Special case if the first node is chosen
689:         struct MedicationStack* temp = personPtr->medicationStack;
690:         personPtr->medicationStack = temp->next;
691:         free(temp);
692:     } else {
693:         // General case for other nodes
694:         currentMedication = personPtr->medicationStack;
695:         for (int i = 1; i < choice - 1 && currentMedication->next != NULL; i++) {
696:             currentMedication = currentMedication->next;
697:         }
698:
699:         if (currentMedication->next != NULL) {
700:             struct MedicationStack* temp = currentMedication->next;
701:             currentMedication->next = temp->next;
702:             free(temp);
703:         } else {
704:             printf("\n\t\t               Invalid choice. No medication removed.\n");
705:             return;
706:         }
707:     }
708:
709:     printf("\n\t\t               Medication marked as taken for %s!\n", personPtr->data.name);
710: }
711:
712:
713: void scheduleActivity(char name[])
714: {
715:     curptr = headptr;
716:     int personFound = 0;
717:
718:     while (curptr != NULL) {
719:         if (strcmp(curptr->data.name, name) == 0) {
720:             personFound = 1;
721:             break;
722:         }

723:         curptr = curptr->ptrnext;
724:     }
725:
726:     if (!personFound) {
727:         printf("\n\t\t               Person with name %s not found in the system.\n", name);
728:         return;
729:     }
730:
731:     struct Activity newActivity;
732:     strcpy(newActivity.personName, name);
733:
734:     // Choose activity
735:     printf("\n\t\t               Choose activity (1 - Exercise Class, 2 - Art and Craft Session, 3 - Memory Games and Brain Exercises,
736:     int activityOption;
737:     scanf("%d", &activityOption);
738:
739:     switch (activityOption) {
740:         case 1:
741:             strcpy(newActivity.activityName, "Exercise Class");
742:             break;
743:         case 2:
744:             strcpy(newActivity.activityName, "Art and Craft Session");
745:             break;
746:         case 3:
747:             strcpy(newActivity.activityName, "Memory Games and Brain Exercises");
748:             break;
749:         case 4:
750:             strcpy(newActivity.activityName, "Music Therapy and Sing-Alongs");
751:             break;
752:         case 5:
753:             strcpy(newActivity.activityName, "Outdoor Garden Walks");
754:             break;
755:         default:
756:             printf("\n\t\t               Invalid activity. Please select 1 to 5.\n");
757:             return;
758:     }
759:
760:     // Choose time for the activity
```

```
761:     printf("\n\t\t               Choose time for the activity (1 for Morning, 2 for Afternoon, 3 for Evening): ");
762:     scanf("%d", &newActivity.timeOfDay);
763:
764:     if (newActivity.timeOfDay < 1 || newActivity.timeOfDay > 3) {
765:         printf("\n\t\t               Invalid time. Please select 1, 2, or 3.\n");
766:         return;
767:     }
768:
769:     printf("\n\t\t               Activity scheduled successfully for ");
770:
771:     switch (newActivity.timeOfDay) {
772:         case 1:
773:             printf("Morning");
774:             break;
775:         case 2:
776:             printf("Afternoon");
777:             break;
778:         case 3:
779:             printf("Evening");
780:             break;
781:     }
782:
783:     printf("!\n");
784:
785:     // Create a new node for the activity
786:     printf("\n\t\t               Medications for %s:\n",spersonPtr->data.name);
787:
788:     if (newNode == NULL) {
789:         printf("\n\n\t\t               Memory allocation failed!\n");
790:         return;
791:     }
792:
793:     strcpy(newNode->personName, newActivity.personName);
794:     strcpy(newNode->activityName, newActivity.activityName);
795:     newNode->timeOfDay = newActivity.timeOfDay;
796:     newNode->next = NULL;
797:
798:     // Add the new node to the linked list

799:     if (front == NULL) {
800:         front = rear = newNode;
801:     } else {
802:         rear->next = newNode;
803:         rear = newNode;
804:     }
805:
806:     // Display the updated list of activities
807:     displayActivities();
808: }
809:
810: void dequeueActivity()
811: {
812:     if (front == NULL) {
813:         printf("\n\n\t\t               !!! Activity Queue is EMPTY - Cannot dequeue !!!\n");
814:     } else {
815:         struct Activity* temp = front;
816:         front = front->next;
817:         free(temp);
818:         printf("\n\n\t\t               Front activity dequeued successfully!\n");
819:     }
820:
821:     displayActivities();
822: }
823:
824: void displayActivities()
825: {
826:     if (front == NULL) {
827:         printf("\n\n\t\t               The activity queue is empty\n");
828:     } else {
829:         struct Activity* current = front;
830:
831:         printf("\n\n\t\t               Scheduled Activities:\n");
832:         printf("\n\t\t               Time of Day (1.Morining, 2.Afternoon, 3.Evening)\n");
833:         while (current != NULL) {
834:             printf("\n\t\t                 Person: %s, Activity: %s, Time of Day: %d\n", current->personName, current->activityName, current->timeOfDay)
835:             current = current->next;
836:         }
```

```
    }
}

void bubbleSort()
{
    if (headptr == NULL || headptr->ptrnext == NULL) {
        // No need to sort if the list is empty or has only one element
        return;
    }

    int swapped;
    struct person *last = NULL;

    do {
        swapped = 0;
        curptr = headptr;

        while (curptr->ptrnext != last) {
            if (curptr->data.age > curptr->ptrnext->data.age) {
                // Swap nodes
                struct person *temp = curptr->ptrnext;
                curptr->ptrnext = temp->ptrnext;
                temp->ptrnext = curptr;
                if (curptr == headptr) {
                    headptr = temp;
                } else {
                    prevptr->ptrnext = temp;
                }
                swapped = 1;
            } else {
                prevptr = curptr;
                curptr = curptr->ptrnext;
            }
        }

        last = curptr;

    } while (swapped);


    displaydetails();
}

// Merge Sort function for sorting by name alphabetically
struct person *merge(struct person *left, struct person *right)
{
    struct person *result = NULL;

    if (left == NULL) {
        return right;
    }
    if (right == NULL) {
        return left;
    }

    if (strcmp(left->data.name, right->data.name) <= 0) {
        result = left;
        result->ptrnext = merge(left->ptrnext, right);
    } else {
        result = right;
        result->ptrnext = merge(left, right->ptrnext);
    }

    return result;
}

struct person *mergeSort(struct person *head)
{
    if (head == NULL || head->ptrnext == NULL) {
        return head;
    }

    struct person *slow = head;
    struct person *fast = head->ptrnext;

    while (fast != NULL && fast->ptrnext != NULL) {
        slow = slow->ptrnext;
```

```
913:         fast = fast->ptrnext->ptrnext;
914:     }
915:
916:     struct person *left = head;
917:     struct person *right = slow->ptrnext;
918:     slow->ptrnext = NULL;
919:
920:     left = mergeSort(left);
921:     right = mergeSort(right);
922:
923:     return merge(left, right);
924:
925:     displaydetails();
926: }
```

REFERENCES

Kuo, M., Wang, S., & Chen, W. (2016). Using information and mobile technology improved elderly home care services. *Health Policy and Technology*, *5*(2), 131–142. https://doi.org/10.1016/j.hlpt.2016.02.004

Joshi, G. P., Acharya, S., Kim, C., Kim, B., & Kim, S. W. (2014). Smart Solutions in Elderly Care Facilities with RFID System and Its Integration with Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, *10*(8), 713946. https://doi.org/10.1155/2014/713946

Renaissance: a smart mobile application featuring elderly people care and assistance systems for the Bangladeshi senior citizens. (2023, April 19). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10170060

TASK DISTRIBUTION

STUDENT ID	NAME	TASK
SD22002	ALMIRA DAMA BINTI SYAHNIZAM	● Screenshot input and output ● coding - Main menu - Insert person details - Update person details - Delete person details - Exit system
SD22030	TUAN NURSHAFIEKA WAHIDA BINTI TUAN NADIN	● Case study 1 ● coding - Schedule activities - Remove activities - Display current activities
SD22038	NURIN NADHIRAH IZZAH BINTI ZAINUDDIN	● Screenshot input and output ● coding - Add medication to person - Display current medication - Remove medication
SD22039	NURUL SYAFIQAH NATASHA BINTI MOHD RAZI	● Case study 3 ● coding - Search person by name - Search person by disease
SD22066	SHAHIRA BINTI MOHAIDEEN MEERA	● Case study 2 ● coding - Sort person by age - Sort person by alphabet