School of Computing

Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

DATA STRUCTURE & ALGORITHMS
(SECJ2013)

SEMESTER 1 2023/2024

Project and Assignment Software Documentation
Mini Project

By

Tan Jing Zhe	A22EC0280
Nipuhawanj Lai Ze Min	A22EC0233
Chong Siew Zhen	A22EC0149
Yap Jun Cheng	

SECTION 08

Lecturer:
DR JOHANNA

10 JANUARY 2024

PART 1: INTRODUCTION

1.1 Synopsis Project:

The Patient Appointment Management System is a streamlined and efficient solution designed to enhance the patient experience at healthcare facilities. The process begins with patients providing their name, matric number, and a brief description of their symptoms to the receptionist upon arrival. The receptionist then assists the patients in registering and issues a unique queue number.

Once registered, patients are required to input their assigned queue number into the system. This number serves as their virtual placeholder in the queue. The system ensures a first-come-first-served approach, allowing the medical staff to attend to patients in an organized manner.

When it's the patient's turn, the doctor calls out the corresponding queue number to initiate the consultation. The doctor reviews the patient's details and addresses their medical concerns. After the consultation is complete, the system automatically updates the queue, replacing the completed number with the next in line.

1. Efficient Registration Process:
Patients provide essential information to the receptionist for quick and accurate registration.

2. Queue Number Assignment:
Receptionists issue a unique queue number to each patient, facilitating a fair and organized order of consultation.

3. Patient Involvement:
Patients actively participate by entering their queue number into the system, reducing waiting times.

4. First-Come-First-Served Approach:
The system ensures a fair and transparent order of consultation, minimizing delays and optimizing the workflow.

5. Automatic Queue Update:
After each consultation, the system dynamically updates the queue, replacing the completed number with the next patient in line.

The Patient Appointment Management System aims to create a seamless and organized healthcare experience, improving efficiency for both medical staff and patients. By leveraging technology to manage appointments and queues, healthcare facilities can enhance patient satisfaction and optimize resource utilization.

1.2 Objective of The Project

1. Efficiency Improvement:
Streamline and expedite the appointment process to minimize waiting times, optimize resource utilization, and enhance overall operational efficiency within the healthcare facility.

2. Enhanced Patient Experience:
Prioritize the improvement of the patient experience by reducing waiting times, providing a transparent and fair queue management system, and fostering a positive and stress-free environment.

3. Accurate Patient Information:
Ensure the accurate collection and recording of patient details during registration, reducing errors in medical records, and supporting the delivery of high-quality healthcare services.

4. Fair Queue Management:
Implement a first-come-first-served approach to managing patient queues, ensuring fairness and transparency in the order of medical consultations, which contributes to patient satisfaction.

5. Optimized Workflow for Medical Staff:
Design the system to provide healthcare professionals with an organized and efficient workflow, allowing them to focus on patient care and minimizing administrative burdens, ultimately improving overall staff productivity.

PART 2: SYSTEM ANALYSIS AND DESIGN (USE CASE, FLOWCHART AND CLASS DIAGRAM)

2.1 System Requirements

Use case diagram

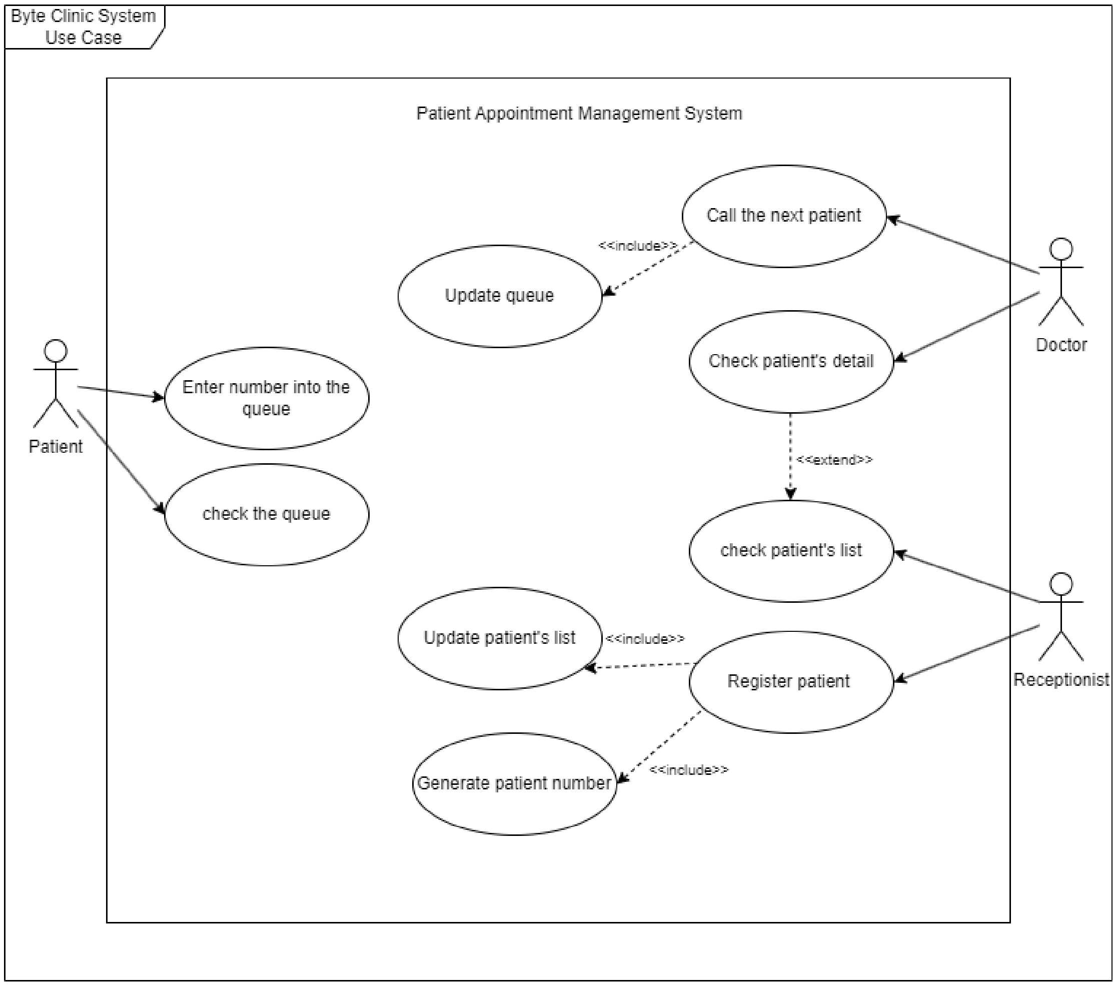


Figure 1 : Use Case Diagram for Patient Appointment Management System

Use Case Description for Patient Appointment Management System

The system users are patients, receptionists and doctors.

Actor	Task
Patient	The patient needs to register their queueing number at the counter. The patients also can check the current queue list.
Receptionist	The receptionist helps the patient register their information, such as name, matric number and also symptoms. The

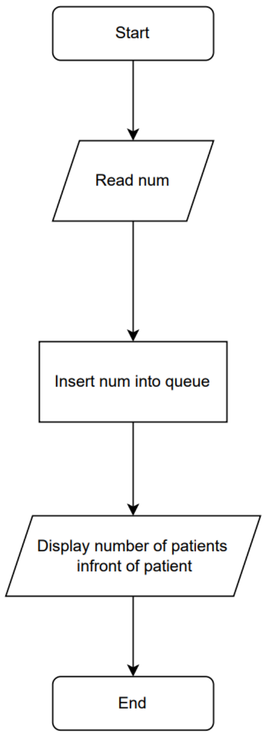
	receptionist also can check the patient’s list, such as name, queueing number and also description for the symptoms.
Doctor	The doctor can see the current first patient’s details and also call for the next patient.

Detail Description for Each Use Cases
The system has 6 main use cases.

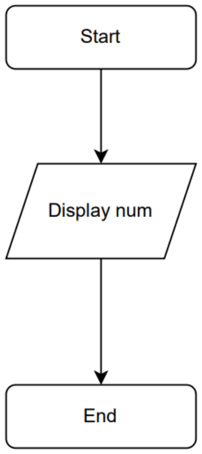
Use Case	**Purpose**
Enter number into the queue	The system prompts the patient to enter their number into the queue for a waiting doctor calling them.
Check the queue	The patients can check the current queue list.
Call the next patient	The doctors can call the next patient. Thus, the system will update the queue’s list.
Check patient’s details	The doctor can check the current first patient’s details in the queue.
Check patient’s list	The receptionist can view the patient’s list in detail, such as their name, queueing number and description.
Register patient	The receptionist helps the patient register his/her name, symptoms and matric number. The system will automatically generate the queueing number for the patient.

2.2 System Design

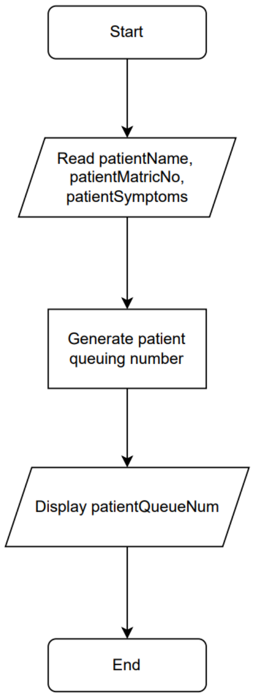
Algorithm: Flowchart for each module.
FlowChart 1: Add Number into queue



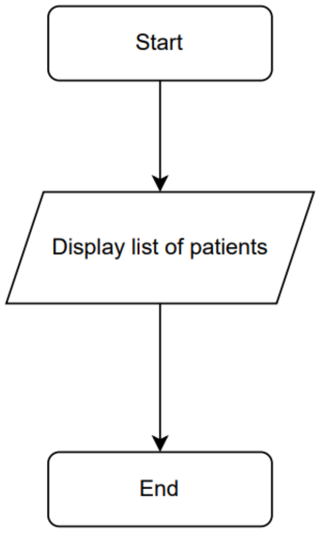
FlowChart 2: Check patient queue



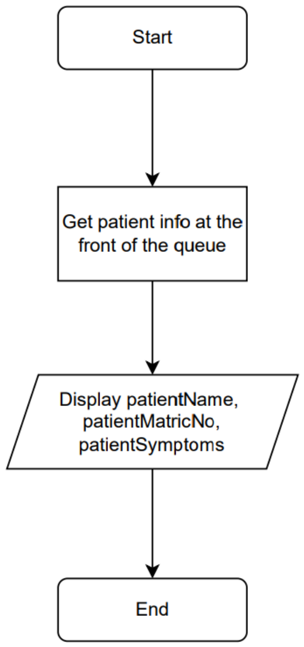
FlowChart 3: Register patient


Start
Read patientName, patientMatricNo, patientSymptoms
Generate patient queuing number
Display patientQueueNum
End

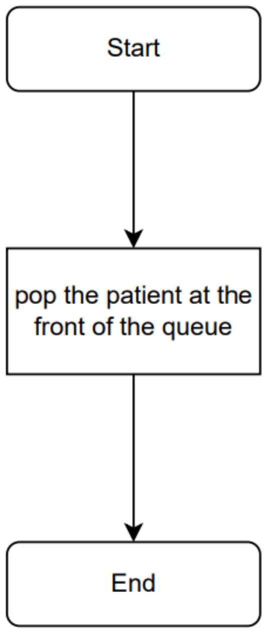
FlowChart 4: Check patient list



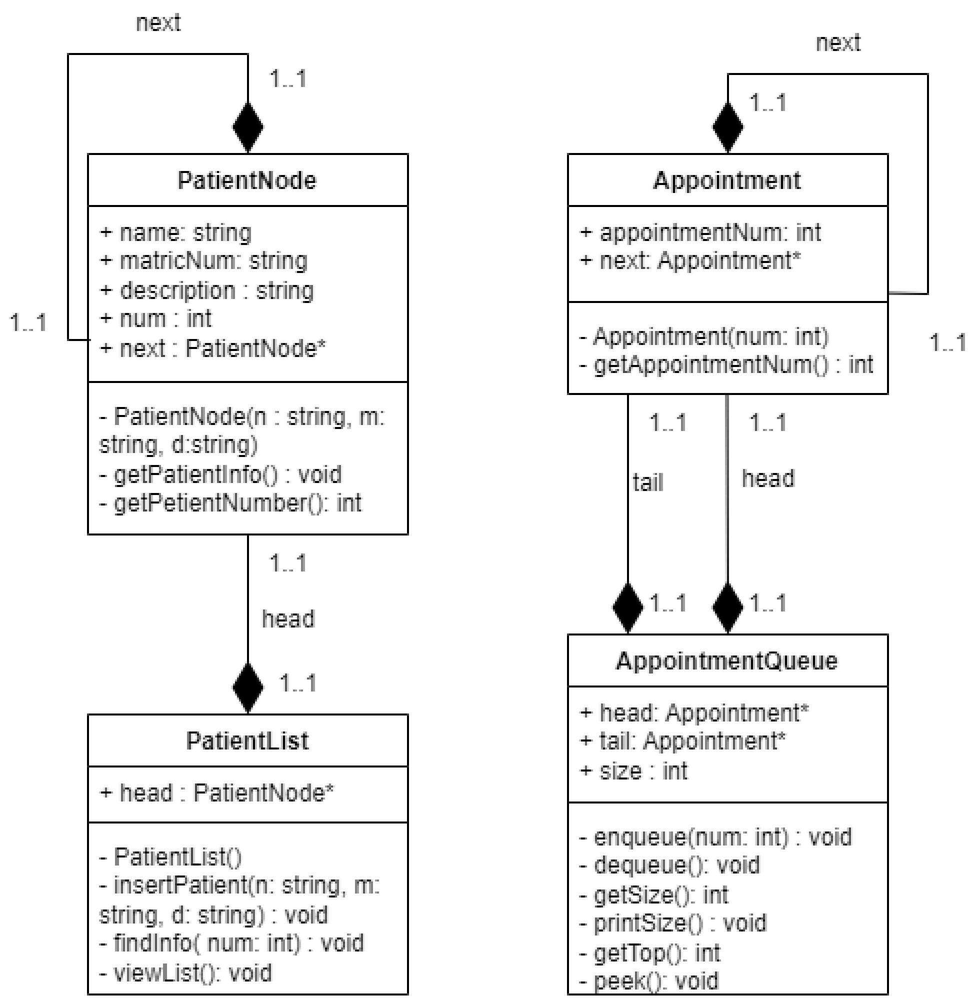
FlowChart 5: Check patient detail


Start
Get patient info at the front of the queue
Display patientName, patientMatricNo, patientSymptoms
End

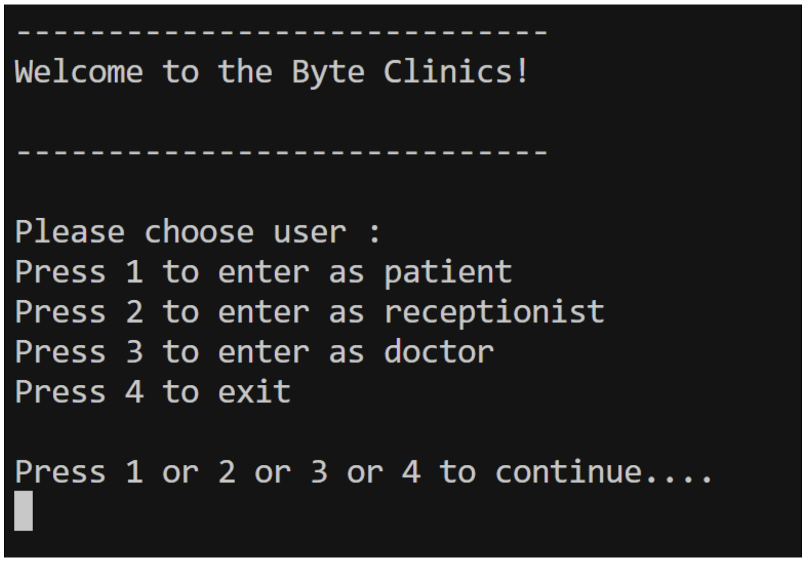
FlowChart 6: Call the next patient



Class diagram



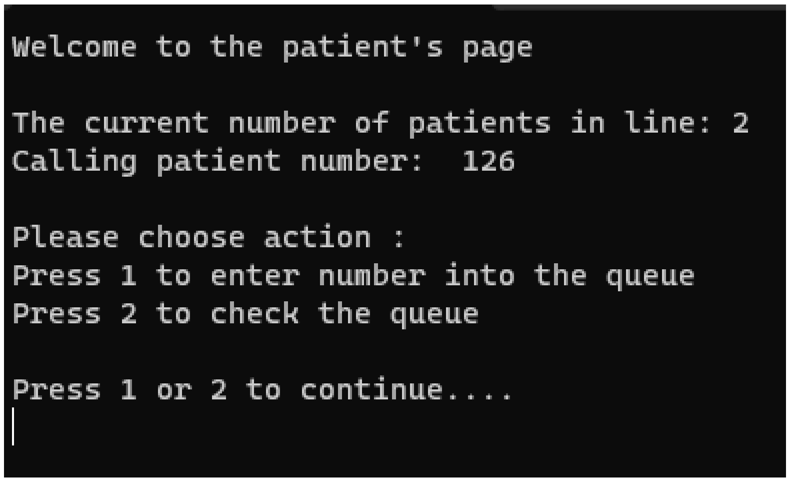
PART 3: SYSTEM PROTOTYPE



```
-----------------------------
Welcome to the Byte Clinics!

-----------------------------

Please choose user :
Press 1 to enter as patient
Press 2 to enter as receptionist
Press 3 to enter as doctor
Press 4 to exit

Press 1 or 2 or 3 or 4 to continue....
```

Skrin 1: *Main menu*

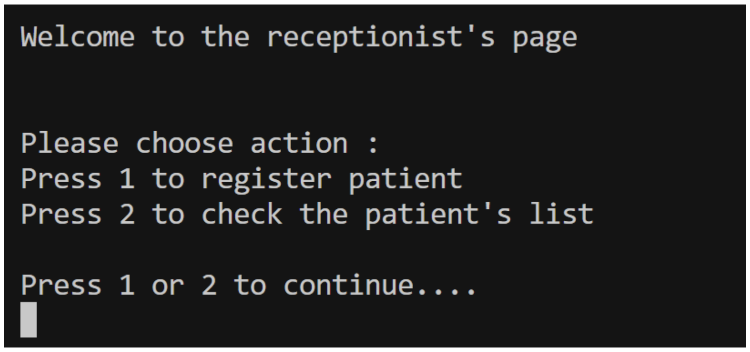
Screen 1: The user must insert an integer value in the range 1-4. If the user enters another number, the system will display the main menu again.



```
Welcome to the patient's page

The current number of patients in line: 2
Calling patient number:  126

Please choose action :
Press 1 to enter number into the queue
Press 2 to check the queue

Press 1 or 2 to continue....
```

Skrin 2: *Patient Page*

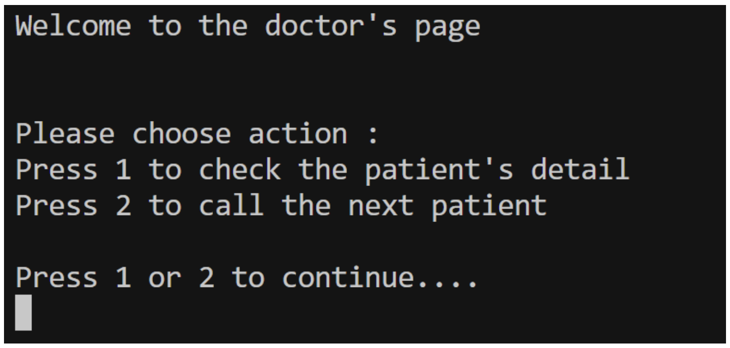
Screen 2: The patient must insert an integer value in the range 1-2. If the user enters another number, the system will prompt the user to press exit for exiting the system or 1 to go to the main page.



```
Welcome to the receptionist's page


Please choose action :
Press 1 to register patient
Press 2 to check the patient's list

Press 1 or 2 to continue....
```

Skrin 3: *Receptionist Page*

Screen 3: The receptionist must insert an integer value in the range 1-2. If the user enters another number, the system will prompt the user to press exit for exiting the system or 1 to go to the main page.

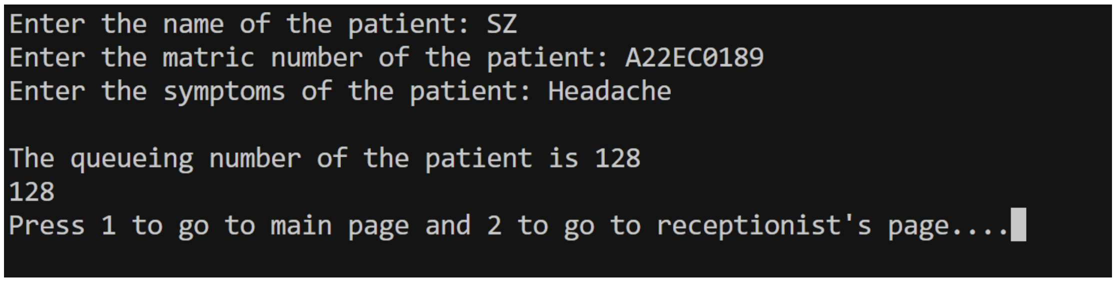


```
Welcome to the doctor's page


Please choose action :
Press 1 to check the patient's detail
Press 2 to call the next patient

Press 1 or 2 to continue....
```

Skrin 4: *Doctor Page*

Screen 4: The doctor must insert an integer value in the range 1-2. If the user enters another number, the system will prompt the user to press exit for exiting the system or 1 to go to the main page.

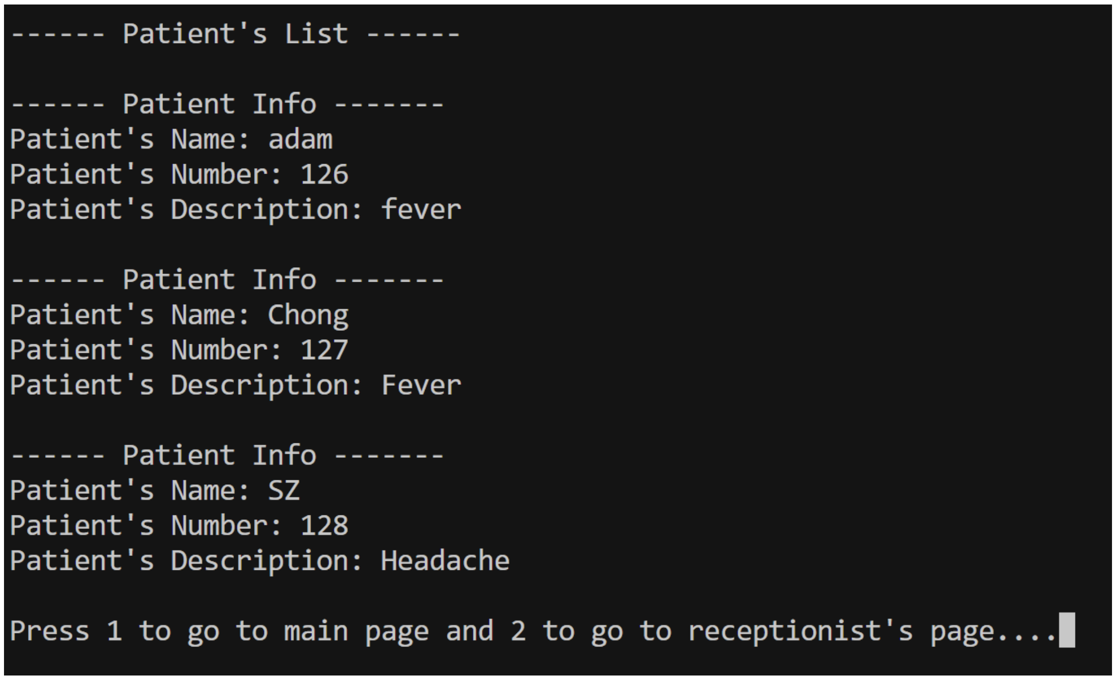
RECEPTIONIST



```
Enter the name of the patient: SZ
Enter the matric number of the patient: A22EC0189
Enter the symptoms of the patient: Headache

The queueing number of the patient is 128
128
Press 1 to go to main page and 2 to go to receptionist's page....
```

Skrin 5: *Register Patient Page*

Screen 5: The system prompts the receptionist to enter the patient's name, matric number and symptoms. Then, the system will show the queueing number for that patient.

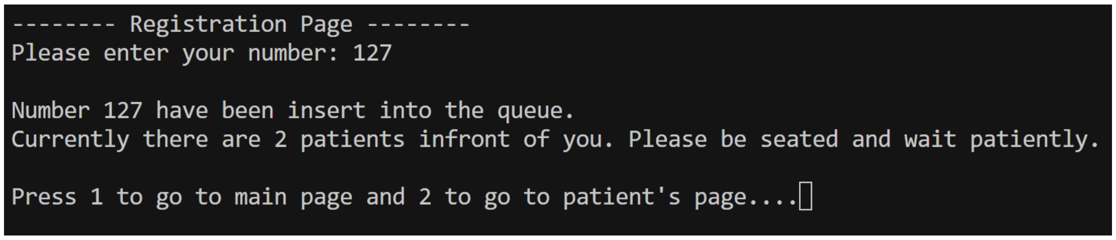


```
------ Patient's List ------

------ Patient Info -------
Patient's Name: adam
Patient's Number: 126
Patient's Description: fever

------ Patient Info -------
Patient's Name: Chong
Patient's Number: 127
Patient's Description: Fever

------ Patient Info -------
Patient's Name: SZ
Patient's Number: 128
Patient's Description: Headache

Press 1 to go to main page and 2 to go to receptionist's page....
```

Skrin 6: *Show Patient List Page*

Screen 6: The system will show the patient details and prompt the user to press 1 for going to the main page and 2 for going to the receptionist's page.

PATIENT

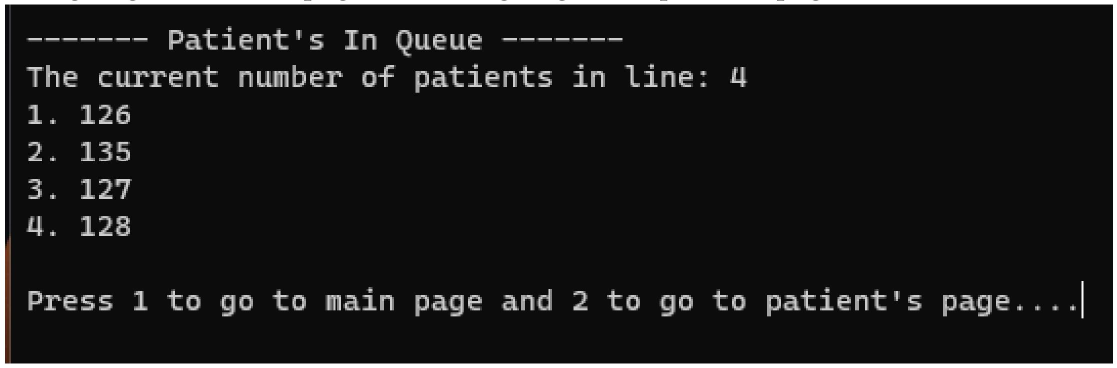


```
-------- Registration Page --------
Please enter your number: 127

Number 127 have been insert into the queue.
Currently there are 2 patients infront of you. Please be seated and wait patiently.

Press 1 to go to main page and 2 to go to patient's page....
```

Skrin 7: *Patient Registration Page*

Screen 7: The system will prompt the patient to enter their queueing number and display a successful message after the patients key in their number. The system prompts the user to press

1 for going to the main page and 2 for going to the patient's page.

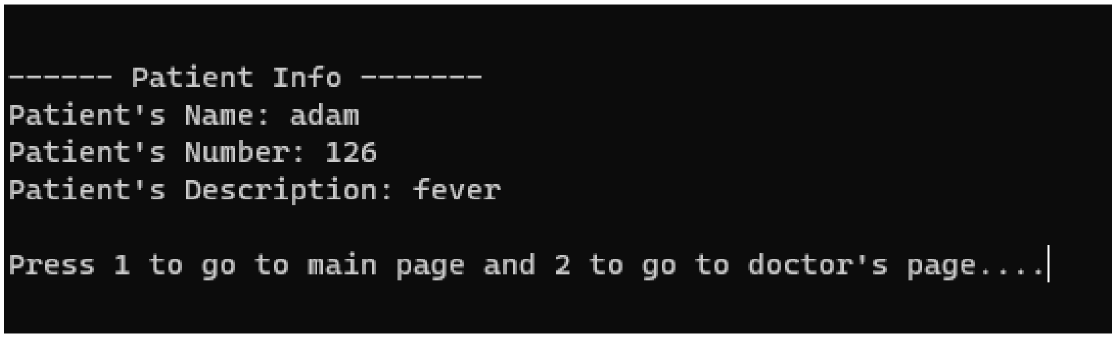


```
------- Patient's In Queue -------
The current number of patients in line: 4
1. 126
2. 135
3. 127
4. 128

Press 1 to go to main page and 2 to go to patient's page....
```

Skrin 8: *Patient Check Queue Page*

Screen 8: The system will show the current number of patients and the list of their number. The system prompts the user to press 1 for going to the main page and 2 for going to the patient's page.

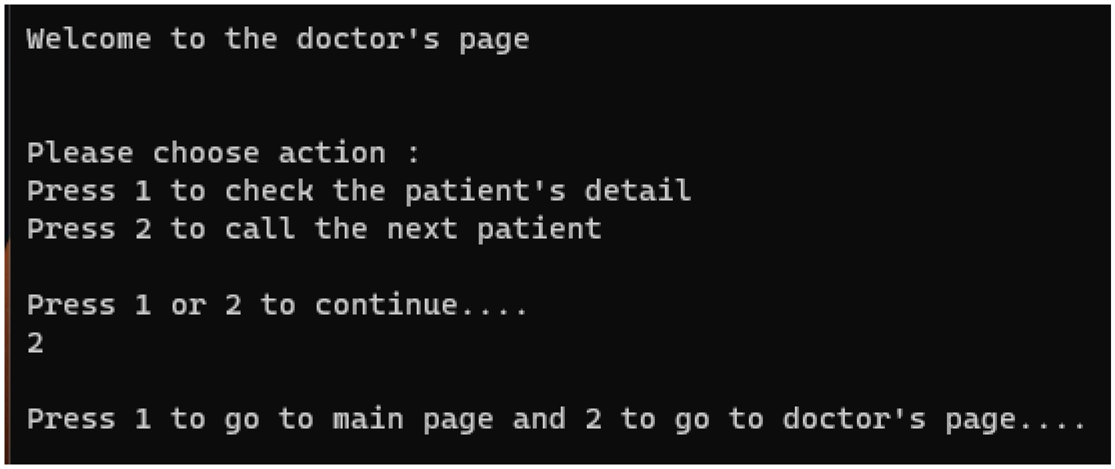
DOCTOR



```
------ Patient Info -------
Patient's Name: adam
Patient's Number: 126
Patient's Description: fever

Press 1 to go to main page and 2 to go to doctor's page....
```

Skrin 9: *First Patient Details Page*

Screen 9: The system will show the details of the first patient in the queue. The system prompts the doctor to press 1 for going to the main page and 2 for going to the doctor's page.



```
Welcome to the doctor's page


Please choose action :
Press 1 to check the patient's detail
Press 2 to call the next patient

Press 1 or 2 to continue....
2

Press 1 to go to main page and 2 to go to doctor's page....
```

Skrin 10: *Calling Next Patient Page*

Screen 10: After calling the next patient, the system prompts the doctor to press 1 for going to the main page and 2 for going to the doctor's page.

PART 4 : APPENDIX

List of Data Files/ File Output/hard copy of the source code

```
#include <iostream>
#include <string>
using namespace std;

int globalNum = 126;

class PatientNode
{
private:
    string name;
    string matricNum;
    string description;
    int num = globalNum;

public:
    PatientNode *next;
    PatientNode(string n, string m, string d)
    {
        name = n;
        matricNum = m;
        description = d;
        globalNum += 1;
        cout << "\nThe queueing number of the patient is " << num << endl;
    }
    void getPatientInfo()
    {
        cout << "------ Patient Info -------" << endl;
        cout << "Patient's Name: " << name << endl;
        cout << "Patient's Number: " << num << endl;
        cout << "Patient's Description: " << description << endl;
    }
    int getPatientNumber()
    {
        return num;
    }
};

class PatientList
{
```

```
private:
    PatientNode *head;

public:
    PatientList()
    {
        head = NULL;
    }

    void insertPatient(string n, string m, string d)
    {
        PatientNode *node = new PatientNode(n, m, d);

        if (head == NULL)
        {
            head = node;
            node->next = head;
        }
        else
        {
            PatientNode *temp = head;
            while (temp->next != head)
            {
                temp = temp->next;
            }
            temp->next = node;
            node->next = head;
        }
        cout << node->getPatientNumber();
    }

    void findInfo(int num)
    {
        PatientNode *temp = head;

        if (temp == NULL)
        {
            cout << "There is no patient record in the system." << endl;
        }
        else
        {
            do
            {
```

```cpp
                if (temp->getPatientNumber() == num)
                {
                    temp->getPatientInfo();
                    break;
                }
                else
                {
                    temp = temp->next;
                }
            } while (temp != head);

            if (temp == head)
            {
                cout << "The patient number is not found in the system.
Please check the number again.";
            }
        }
    }
};

class Appointment
{
private:
    int appointmentNum;

public:
    Appointment *next;
    Appointment(int num)
    {
        appointmentNum = num;
    }
    int getAppointmentNum()
    {
        return appointmentNum;
    }
};

class AppointmentQueue
{
private:
    Appointment *head;
    Appointment *tail;
    int size = 0;
```

```
public:
    void enqueue(int num)
    {
        Appointment *node = new Appointment(num);
        if (head == NULL)
        {
            head = node;
            tail = node;
        }
        else
        {
            node->next = NULL;
            tail->next = node;
            tail = node;
        }
        cout << "Number " << tail->getAppointmentNum() << " have been
insert into the queue. " << endl;
        cout << "Currently there are " << size << " patients infront of
you. Please be seated and wait patiently.";
        size++;
    }
    void dequeue()
    {
        if (head == NULL)
        {
            cout << "The queue is empty!" << endl;
        }
        else
        {
            Appointment *temp = head;
            head = temp->next;
            size--;
        }
    }
    int getSize()
    {
        return size;
    }
    void printSize()
    {
        cout << "The current number of patients in line: " << size << endl;
    }
    void getDetails()
    {
        Appointment *temp = head;
        if (head == NULL)
        {
            cout << "The queue is empty!" << endl;
        }
        else
        {
            int num = 1;
            cout << "------- Patient's In Queue -------" << endl;
            printSize();
            while (temp != NULL)
            {
                cout << num << ". " << temp->getAppointmentNum() << endl;
                temp = temp->next;
                num++;
            }
        }
    }
    int getTop()
    {
        return head->getAppointmentNum();
    }
    void peek()
    {
        Appointment *temp = head;
        cout << "Calling patient number:  " << temp->getAppointmentNum() <<
endl;
    }
};

int mainPage()
{
    int selection;
    cout << "-----------------------------" << endl;
    cout << "Welcome to the Byte Clinics!\n";
    cout << "-----------------------------\n"
         << endl;
    cout << "Please choose user : " << endl;
    cout << "Press 1 to enter as patient" << endl;
    cout << "Press 2 to enter as receptionist" << endl;
    cout << "Press 3 to enter as doctor" << endl;
    cout << "Press 4 to exit" << endl;

    cout << "\nPress 1 or 2 or 3 or 4 to continue...." << endl;
    cin >> selection;
    return selection;
};

int patientPage()
{
    int selection;
    cout << "\nPlease choose action : " << endl;
    cout << "Press 1 to enter number into the queue" << endl;
    cout << "Press 2 to check the queue" << endl;

    cout << "\nPress 1 or 2 to continue...." << endl;
    cin >> selection;
    cin.ignore();
    return selection;
};

int receptionistPage()
{
    int selection;
    cout << "\nPlease choose action : " << endl;
    cout << "Press 1 to register patient " << endl;
    cout << "Press 2 to check the patient's list" << endl;

    cout << "\nPress 1 or 2 to continue...." << endl;
    cin >> selection;
    cin.ignore();
    return selection;
};

int doctorPage()
{
    int selection;
    cout << "\nPlease choose action : " << endl;
    cout << "Press 1 to check the patient's detail " << endl;
    cout << "Press 2 to call the next patient " << endl;

    cout << "\nPress 1 or 2 to continue...." << endl;
    cin >> selection;
    cin.ignore();
```

```cpp
    return selection;
};

int main()
{
    PatientList patientList;
    AppointmentQueue queue;
    patientList.insertPatient("adam", "A22EC0280", "fever");
    queue.enqueue(125);
    queue.enqueue(135);
    queue.getDetails();
    int selection;

main:
    system("cls");
    selection = mainPage();

    if (selection == 1)
    {
    patient:
        system("cls");
        cout << "Welcome to the patient's page\n"
             << endl;
        queue.printSize();
        queue.peek();
        selection = patientPage();
        if (selection == 1)
        {
            int patientNum;
            cout << "Please enter your number: ";
            cin >> patientNum;
            queue.enqueue(patientNum);
            goto patientChoice;
        }
        else if (selection == 2)
        {
            queue.getDetails();
            goto patientChoice;
        }
    }
    else if (selection == 2)
    {
    receptionist:
        system("cls");
        cout << "Welcome to the receptionist's page\n"
             << endl;
        selection = receptionistPage();
        if (selection == 1)
        {
            system("cls");
            string name, matricNum, description;
            cout << "Enter the name of the patient: ";
            getline(cin, name);
            cout << "Enter the matric number of the patient: ";
            getline(cin, matricNum);
            cout << "Enter the symptoms of the patient: ";
            getline(cin, description);
            patientList.insertPatient(name, matricNum, description);
            goto receptionistChoices;
        }
        else if (selection == 2)
        {
        }
    }
    else if (selection == 3)
    {
    doctor:
        system("cls");
        cout << "Welcome to the doctor's page\n"
             << endl;
        selection = doctorPage();
        if (selection == 1)
        {
            patientList.findInfo(queue.getTop());
        }
        else if (selection == 2)
        {
            queue.dequeue();
        }
    }
    else if (selection == 4)
    {
        goto exit;
    }
    else
    {
        cout << "Please enter the available choices only" << endl;
        system("cls");
        goto main;
    }

choices:
    cout << "\nPress any key to exit or press 1 to go to main page: ";
    cin >> selection;
    cin.ignore();
    if (selection == 1)
    {
        goto main;
    }
    else
    {
        cin.ignore();
        goto exit;
    }

patientChoice:
    cout << "\nPress 1 to go to main page and 2 to go to patient's
page....";
    cin >> selection;
    cin.ignore();
    if (selection == 1)
    {
        goto main;
    }
    else if (selection == 2)
    {
        goto patient;
    }
receptionistChoices:
    cout << "\nPress 1 to go to main page and 2 to go to receptionist's
page....";
    cin >> selection;
    cin.ignore();
    if (selection == 1)
    {
        goto main;
    }
    else if (selection == 2)
    {
        goto receptionist;
    }
doctorChoices:
    cout << "\nPress 1 to go to main page and 2 to go to doctor's
page....";
    cin >> selection;
    cin.ignore();
    if (selection == 1)
    {
        goto main;
    }
    else if (selection == 2)
    {
        goto doctor;
    }
exit:
    cout << "Thanks for using the Byte Clinic system!";
}
```

---- END OF DOCUMENTATION -----