Protection of information based on sensitivity and privilege levels

Name: R.D.K.S Rajapakse
Index: 180509T
Module: CS3052
Date: 2020.11.18

1) Source code

main.cpp
```
#include <iostream>
#include <string>
#include "User.h"
#include "Auth.h"
#include "CSVfileHandler.h"
#include "program.h"
using namespace std;

/*
* Main method
* To compile - "g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o
test2.out main.cpp"
* Compilation will create an executable test2.out file
* To run the Executable file - "./test2.out"
* If get a compilation error refer Auth.h file
*/
int main() {

   Auth auth;
   User user;
   string username, password;

   // Read username
   cout<<"Enter username: ";
   cin>>username;

   // Read password
   cout<<"Enter password: ";
   cin>>password;

   try {
       user = auth.login(username, password);
       cout<<"Logged as "<<user.getName()<<endl;
   } catch(invalid_argument &e) {
       cerr<<e.what()<<endl;
       return 1;
   }
   program(user);
   return 0;
}
```

Auth.h
```
#include <iostream>
#include <string>

#define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1

#include <crypto++/md5.h>
#include <crypto++/hex.h>
#include "PasswdReader.h"

/*
* Handle Auth
* MD5 hash generation function
* Required crytopp library (https://cryptopp.com/)
*/
```

```
class Auth {
   private:
       PasswdReader passwdReader;
       map<string, User> users;

       string hashValue(string password) {
           byte digest[ CryptoPP::Weak::MD5::DIGESTSIZE ];
           std::string message = password;

           CryptoPP::Weak::MD5 hash;
               hash.CalculateDigest( digest, (byte*) message.c_str(),
message.length() );

           CryptoPP::HexEncoder encoder;
           std::string output;

           encoder.Attach( new CryptoPP::StringSink( output ) );
           encoder.Put( digest, sizeof(digest) );
           encoder.MessageEnd();

           //std::cout<<output;
           return output;
       }

   public:
       Auth() {
           passwdReader.readPasswdFile();
           users = passwdReader.getUsers();
       }

       User login(string username, string password) {
           string hashPassword = this->hashValue(password);
           if(users.find(username) == users.end()) {
               throw invalid_argument("User not found");
           }
           User user = users.at(username);
           if(user.getPassword() != hashPassword) {
               throw invalid_argument("Password incorrect");
           }
           return user;
       }

};
```

CSVfileHandler.h

```
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
using namespace std;

#ifndef CSVFILEHANDLER_H
#define CSVFILEHANDLER_H
class CSVfileHandler {
   private:
       vector< vector<string> > rows;
   public:
       void read_record() {
           fstream infile("data.csv");
```

```cpp
            string line, word;
            vector<string> row;
            while (getline(infile, line)) {
                istringstream iss(line);
                row.clear();
                while( getline(iss, word, ',') ) {

                    // add all the column data
                    // of a row to a vector
                    row.push_back(word);
                }
                rows.push_back(row);
            }
        }

        void create(){
            // file pointer
            fstream fout;

            // opens an existing csv file or creates a new file.
            fout.open("dataNew.csv", ios::out | ios::app);

            // Read the input
             for (auto i = this->rows.begin(); i != this->rows.end();
++i) {
                vector<string> row = *i;
                // Insert the data to file
                fout<< row.at(0) << ","
                    << row.at(1) << ","
                    << row.at(2) << ","
                    << row.at(3) << ","
                    << row.at(4) << ","
                    << row.at(5)
                    << "\n";
            }
            fout.close();
            // removing the existing file
            remove("data.csv");

            // renaming the updated file with the existing file name
            rename("dataNew.csv", "data.csv");
        }

        vector< vector<string> > getRows() { return this->rows; }

         void setRows(vector< vector<string> > newRows) { this->rows =
newRows; }

        void print() {
              for(auto i = this->rows.begin(); i != this->rows.end();
++i) {
                vector<string> row = *i;
                for(auto j = row.begin(); j != row.end(); ++j) {
                    cout<<*j;
                }
                cout<<endl;
            }
        }
};
#endif
```

DataRow.h

```
#include <string>
#include "User.h"
using namespace std;

#ifndef DATAROW_H
#define DATAROW_H

class DataRow {
   private:
       string name,sickness,drugs,tests;
       int id,age;
   public:
        DataRow(int id, string name, int age, string sickness, string
drugs, string tests){
           this->id = id;
           this->name = name;
           this->age = age;
           this->sickness = sickness;
           this->drugs = drugs;
           this->tests = tests;
       }
       void setName(string name) {
           this->name = name;
       }
       void setAge(int age) {
           this->age = age;
       }
       void setSickness(string sickness) {
           this->sickness = sickness;
       }
       void setDrugs(string drugs) {
           this->drugs = drugs;
       }
       void setTests(string tests) {
           this->tests = tests;
       }

       int getId() {
           return this->id;
       }
       string getName() {
           return this->name;
       }
       int getAge() {
           return this->age;
       }
       string getSickness() {
           return this->sickness;
       }
       string getDrugs() {
           return this->drugs;
       }

       string getTests() {
           return this->tests;
       }
};
#endif
```

DataRowHandler.h

```
#include <vector>
#include <string>
#include "CSVfileHandler.h"
#include "DataRow.h"

using namespace std;

#ifndef DATAROWHANDLER_H
#define DATAROWHANDLER_H

class DataRowHandler {
   private:
       CSVfileHandler csvHandler;
       vector<DataRow> dataRows;
   public:
       void loadDataRows() {
           csvHandler.read_record();
           vector< vector<string> > data = csvHandler.getRows();
           for(auto i = data.begin(); i != data.end(); ++i) {
               vector<string> row = *i;
               try {
                          DataRow dataRow(stoi(row.at(0)), row.at(1),
stoi(row.at(2)), row.at(3), row.at(4), row.at(5));
                   dataRows.push_back(dataRow);
               } catch(out_of_range &e) {
                   continue;
               }
           }
       }

       void updateRows() {
           vector< vector<string> > strRows;
                       for(auto i = this->dataRows.begin(); i !=
this->dataRows.end(); ++i) {
               DataRow row = *i;
               vector<string> strRow = {
                   to_string(row.getId()),
                   row.getName(),
                   to_string(row.getAge()),
                   row.getSickness(),
                   row.getDrugs(),
                   row.getTests()
               };
               strRows.push_back(strRow);
           }
           csvHandler.setRows(strRows);
           csvHandler.create();
       }

       vector<DataRow> getDataRows() { return this->dataRows; }

       void setDataRows(vector<DataRow> dataRowsNew) { this->dataRows
= dataRowsNew; }

       void print() {
                       for(auto i = this->dataRows.begin(); i !=
this->dataRows.end(); ++i) {
               DataRow dataRow = *i;
               cout<<dataRow.getName()<<" "<<dataRow.getAge()<<endl;
           }
       }
};
#endif
```

PasswdReader.h

```
#include <map>
#include <string>
#include "Reader.h"
#include "User.h"

#define PASSWD "passwd"
#define DEL ":"

/*
* Convert raw config file into an object
* and store in a hashMap
*/
class PasswdReader {
   private:
       map<string, User> users;
   public:
       void readPasswdFile() {
           Reader reader;
           reader.readFile(PASSWD,DEL);
           // reader.readFile("passwd",":");
           this->convertToUser(reader.getRows());
       }
       void convertToUser(vector< vector<string> > rows) {
           for(auto i = rows.begin(); i != rows.end(); ++i) {
               vector<string> row = *i;
               try {
                                                                       User
user(row.at(0),row.at(1),row.at(2),row.at(3));
                   users.insert(pair<string, User>(row.at(0), user));
               } catch(out_of_range &e){
                   cout<<"Invalid data record"<<endl;
               }
           }
       }

       map<string, User> getUsers() {
           return this->users;
       }
};
```

program.h

```
#include "User.h"
#include "RefMonitor.h"
#include "DataRowHandler.h"
#include "CSVfileHandler.h"
using namespace std;

/*
* Main program
*/
void program(User user) {
   while (true){
       int inp;
       RefMonitor refMonitor;
       cout<<"#######################################\n";
       cout<<"Press 0 to exit\n";
       cout<<"Press 1 to view patient details\n";
       cout<<"Press 2 to update patient details\n";
       cout<<"#######################################\n";
       cin>>inp;
       if(inp == 0){
           cout<<"Bye!\n";
           break;
       }
       else if(inp == 1) {
           vector<string> dataRows = refMonitor.viewRowData(user);
           for(auto i = dataRows.begin(); i != dataRows.end(); ++i) {
               cout<<*i<<endl;
           }
       } else if(inp == 2) {
           int id, age;
           string sickness, drugs, tests;
               cout<<"****Enter 0 if you do not want to change the
value****\n";
           cout<<"Enter patient id: ";
           cin>>id;
           cout<<"Age: ";
           cin>>age;
           cout<<"Sickness: ";
           cin>>sickness;
           cout<<"Drugs: ";
           cin>>drugs;
           cout<<"Test results: ";
           cin>>tests;
           try {
                string updatedRow = refMonitor.updateRecord(user, id,
age, sickness, drugs, tests);
               cout<<updatedRow<<endl;
           } catch(invalid_argument &e) {
               cerr<<e.what()<<endl;
           }


       }
   }
}
```

Reader.h

```
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
using namespace std;

#ifndef READER_H
#define READER_H

/*
* Read configuration file
* Config File name - passwd
```

```
*/
class Reader {
   private:
       vector<vector<string>> rows;
   public:
       void readFile(string fileName, string del) {
           fstream infile(fileName);
           string line;
           while (getline(infile, line)) {
               istringstream iss(line);
               vector<string> row = this->tokenize(line, del);
               rows.push_back(row);
           }

       }
       // Spliting function
       vector<string> tokenize(string s, string del = ":") {
           vector<string> args;
           int start = 0;
           int end = s.find(del);
           while (end != -1) {
               args.push_back(s.substr(start, end - start));
               start = end + del.size();
               end = s.find(del, start);
           }
           args.push_back(s.substr(start, end - start));
           return args;
       }

       vector<vector<string>> getRows() {
           return this->rows;
       }
};
#endif
```

RefMonitor.h

```
#include "DataRow.h"
#include "User.h"
#include "DataRowHandler.h"
using namespace std;

/*
*    All requests passes through this reference monitor
*     If user do not have access to the DataRow RefMonitor will throw
an Exception
*/
class RefMonitor {
   private:
       DataRowHandler dataRowHandler;

         DataRow initDataRow(User user,int id, string name, int age,
string sickness, string drugs, string tests){
           if (user.getAccess() != "1111") {
               throw invalid_argument("Unauthoze access");
           }
           DataRow row(id, name, age, sickness, drugs, tests);
           return row;
       }
```

```
        void setName(User user, DataRow &row, string name) {
            if(user.getAccess().at(0) != '1'){
                throw invalid_argument("Unauzorized access");
            }
            row.setName(name);
        }
        void setAge(User user, DataRow &row, int age) {
            if(user.getAccess().at(1) != '1'){
                throw invalid_argument("Unauzorized access");
            }
            row.setAge(age);
        }
        void setSickness(User user, DataRow &row, string sickness) {
            if(user.getAccess().at(2) != '1'){
                throw invalid_argument("Unauzorized access");
            }
            row.setSickness(sickness);
        }
        void setDrugs(User user, DataRow &row, string drugs) {
            if(user.getAccess().at(3) != '1'){
                throw invalid_argument("Unauzorized access");
            }
            row.setDrugs(drugs);
        }
        void setTests(User user, DataRow &row, string tests) {
            if(user.getAccess().at(4) != '1'){
                throw invalid_argument("Unauzorized access");
            }
            row.setTests(tests);
        }

        string getStaffView(User user, DataRow row) {
            if(user.getRole() != "S") {
                throw invalid_argument("Unauzorized access");
            }
               return to_string(row.getId()) +", "+ row.getName()+ ",
"+to_string(row.getAge())+",                    "+row.getSickness()+",
"+row.getDrugs()+", "+row.getTests();
        }

        string getPatientView(User user, DataRow row) {
                     if(user.getRole()  !=  "P"  ||  user.getName()  !=
row.getName()) {
                throw invalid_argument("Unauzorized access");
            }
               return to_string(row.getId()) +", "+ row.getName()+ ",
"+row.getDrugs();
        }

    public:
        RefMonitor() {
            this->dataRowHandler.loadDataRows();
        }

        vector<string> viewRowData(User user) {
            vector<string> strRows;
                                       vector<DataRow>   dataRows   =
this->dataRowHandler.getDataRows();

            if(user.getRole() == "S") {
                    for(auto i = dataRows.begin(); i != dataRows.end();
++i) {
                        try {
                            string strRow = this->getStaffView(user, *i);
                            strRows.push_back(strRow);
                        } catch(invalid_argument &e) {}
                    }
                    return strRows;
                } else if(user.getRole() == "P") {
                    for(auto i = dataRows.begin(); i != dataRows.end();
++i) {
                        try {
                                string strRow = this->getPatientView(user,
*i);
                            strRows.push_back(strRow);
                        } catch(invalid_argument &e) {
                            continue;
                        }
                    }
                    return strRows;
                }
                else {
                    throw bad_exception();
                }
        }

            string updateRecord(User user,int id, int age, string
sickness, string drugs, string tests) {
                if(user.getRole() != "S") {
                    throw invalid_argument("Unauthorized");
                }
                int index;

                                        vector<DataRow>   dataRows   =
this->dataRowHandler.getDataRows();
                auto it = find_if(dataRows.begin(), dataRows.end(),
[&id](DataRow& obj) {return obj.getId() == id;});
                if (it != dataRows.end()){
                        // found element. it is an iterator to the first
matching element.
                    // if you really need the index, you can also get it:
                    index = distance(dataRows.begin(), it);
                } else {
                        throw invalid_argument("Can't find the patient
record");
                }

                try{
                    if(age != 0) {
                        this->setAge(user, dataRows.at(index), age);
                    }
                    if(sickness != "0") {
                            this->setSickness(user, dataRows.at(index),
sickness);
                    }
                    if(drugs != "0") {
                        this->setDrugs(user, dataRows.at(index), drugs);
                    }
                    if(tests != "0") {
                        this->setTests(user, dataRows.at(index), tests);
```

```
               }
               dataRowHandler.setDataRows(dataRows);
               dataRowHandler.updateRows();
           } catch(invalid_argument &e) {
               cerr<<e.what()<<endl;
           }

           DataRow row = dataRows.at(index);
              return to_string(row.getId()) +", "+ row.getName()+ ",
"+to_string(row.getAge())+",                    "+row.getSickness()+",
"+row.getDrugs()+", "+row.getTests();

       }
};
```

User.h

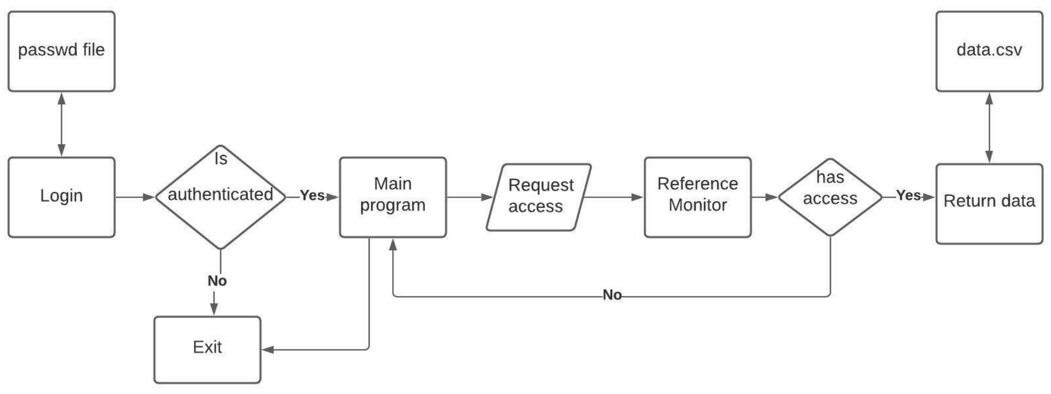
```
#include <iostream>
#include <string>
using namespace std;

#ifndef USER_H
#define USER_H
class User
{
private:
   string name;
   string password;
   string role;
   string access;
public:
   User(string name, string password, string role, string access) {
       this->name = name;
       this->password = password;
       this->role = role;
       this->access = access;
   }
   User() {}
   string getName() { return this->name; }
   string getPassword() { return this->password; }
   string getRole() { return this->role; }
   string getAccess() { return this->access; }

   void print() {
                         cout<<"Name    :"<<this->name<<"/    password
:"<<this->password<<endl;
   }
};
#endif
```

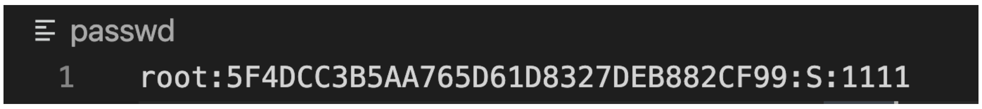
2) **A description of how you decided the access to data records on sensitivity of data.**

- High level overview of the system



- System flow
  - In order to run this system we have to provide the "passwd" configuration file separately. It includes username, md5 password hash, role('S'- staff, 'P'- patient), bit string that indicate writing access to the "data.csv" file
  - Then the user can log into the system using username and password. If a user provides a valid username and password, a user object is created according to the user role and access types and kept in memory.
  - If the user is a patient, he/she can only read their information. It is also limited. They can only view their user id, username and drug perception.
  - Patients are not allowed to write data to the "data.csv" file.
  - Staff members can view all patient details. But their writing access is limited according to their department.
  - Users can exit from the system after they finish their tasks.

- Access control based on data sensitivity
  - Passwd file (Configuration file)

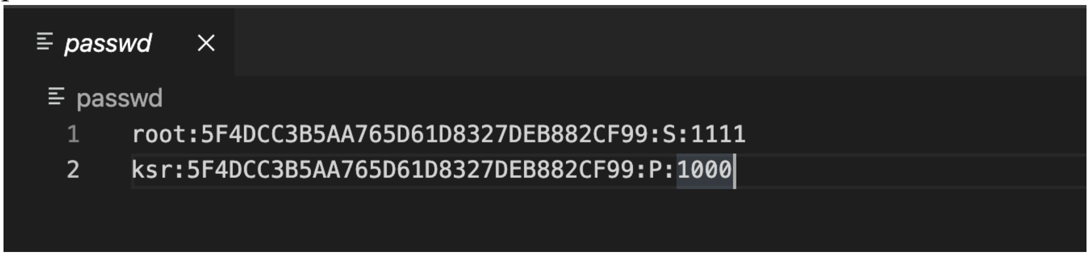


```
passwd
root:5F4DCC3B5AA765D61D8327DEB882CF99:S:1111
```

  - First read all data in the "data.csv" file and create a list of DataRow objects.
  - DataRow objects are only handled through the Reference Monitor. The Reference monitor compares the data access request and the user object saved in the memory to see whether the user has privilege to access the data. If the user has proper access, the reference monitor allows the user to make changes in the DataRow list. Or else reject the request.
  - In the "passwd" file the last string of data represents the writing access. In the "data.csv" file there are five columns. First column represents the record id and the other four represent the actual data. Each bit in this string represents the writing access to each column. If it is one user has the privilege to update that column.

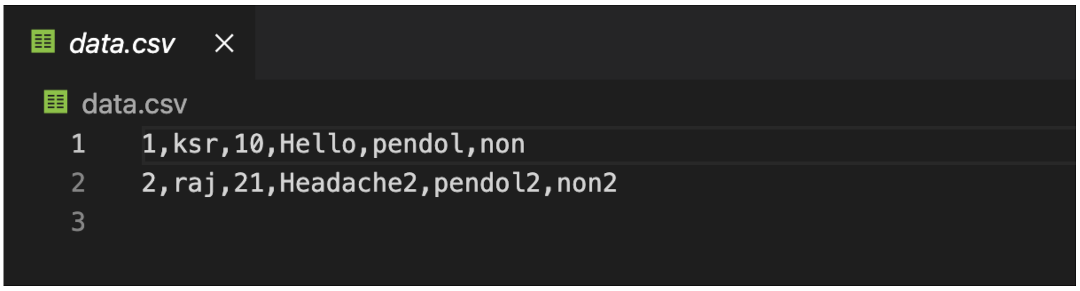
3) Annexes

passwd file



```
passwd
root:5F4DCC3B5AA765D61D8327DEB882CF99:S:1111
ksr:5F4DCC3B5AA765D61D8327DEB882CF99:P:1000
```

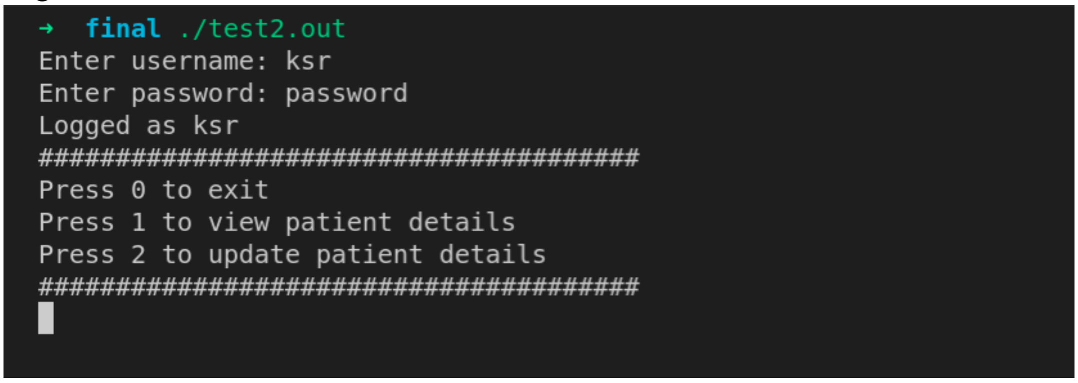
data.csv



```
data.csv
1,ksr,10,Hello,pendol,non
2,raj,21,Headache2,pendol2,non2
```

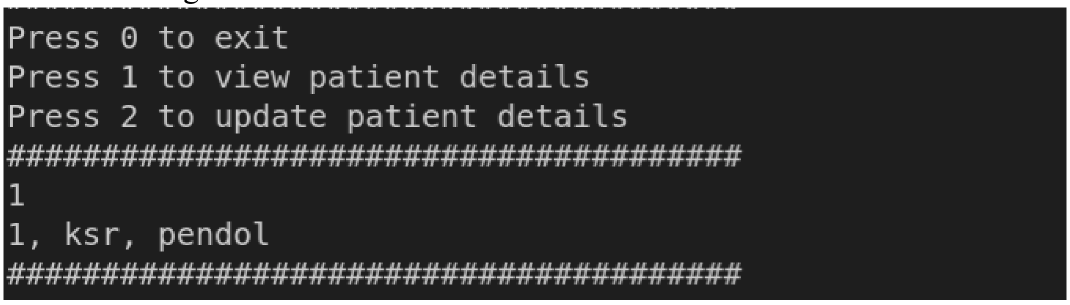
Program execution

Login



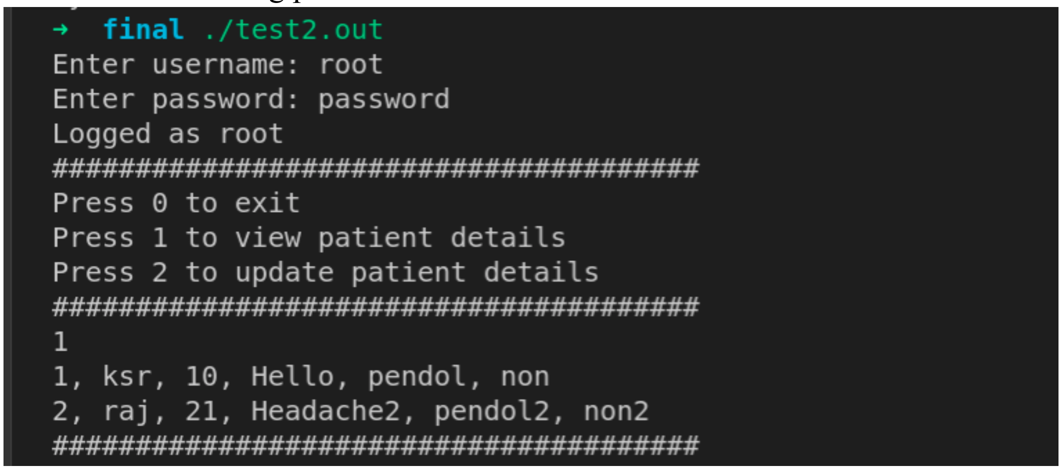
```
➜  final ./test2.out
Enter username: ksr
Enter password: password
Logged as ksr
#######################################
Press 0 to exit
Press 1 to view patient details
Press 2 to update patient details
#######################################
```

Patient viewing details



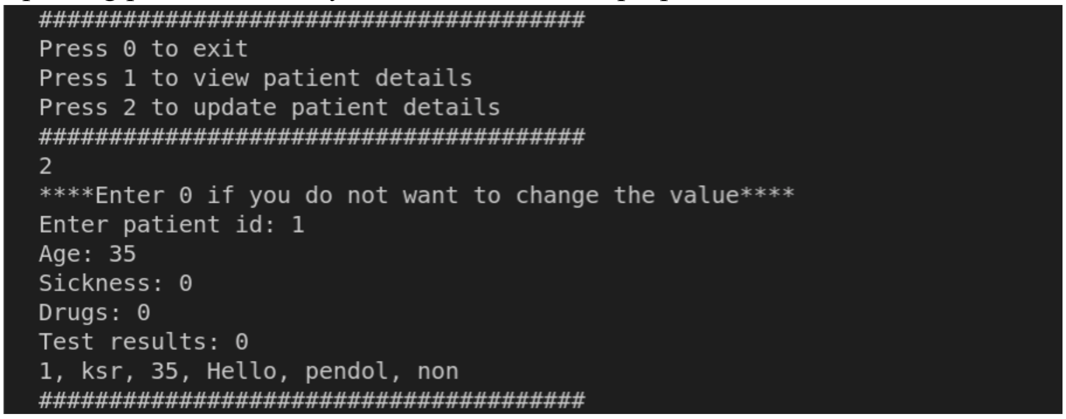
```
Press 0 to exit
Press 1 to view patient details
Press 2 to update patient details
#######################################
1
1, ksr, pendol
#######################################
```

Staff member viewing patients details



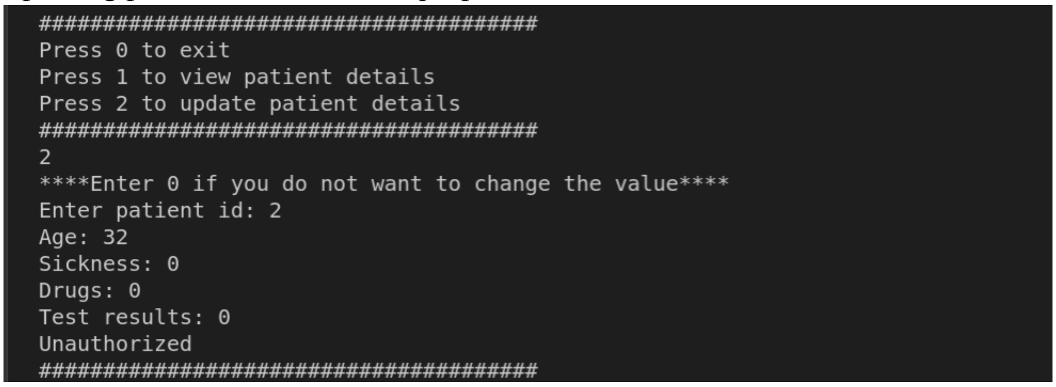
```
➜  final ./test2.out
Enter username: root
Enter password: password
Logged as root
#######################################
Press 0 to exit
Press 1 to view patient details
Press 2 to update patient details
#######################################
1
1, ksr, 10, Hello, pendol, non
2, raj, 21, Headache2, pendol2, non2
#######################################
```

Updating patient details by a staff member with proper access



```
########################################
Press 0 to exit
Press 1 to view patient details
Press 2 to update patient details
########################################
2
****Enter 0 if you do not want to change the value****
Enter patient id: 1
Age: 35
Sickness: 0
Drugs: 0
Test results: 0
1, ksr, 35, Hello, pendol, non
########################################
```

Updating patient details without proper access



```
#######################################
Press 0 to exit
Press 1 to view patient details
Press 2 to update patient details
#######################################
2
****Enter 0 if you do not want to change the value****
Enter patient id: 2
Age: 32
Sickness: 0
Drugs: 0
Test results: 0
Unauthorized
#######################################
```