جامعة أبوظبي
ABU DHABI UNIVERSITY

HEALTH MANAGEMENT SYSTEM

CSC 311 – Java for Internet

Spring 2019-2020

Section 22&66

Instructor: Dr. Adel Khelifi

DONE BY: -

SAEED ALHMOUDI (1070332)

FATIMA ISMAIL (1061485)

KEVIN JOHN (1072928)

Table of Contents

Introduction

Background of Project

The current COVID-19 outbreak has caused immense population of the world suffering with economic and health crisis (WHO, 2020). There are a number of countries all around the world dealing with an outburst of consistently increasing COVID-19 cases – for which the governments of all the suffering countries are in search to cope with this situation at their best. As observed this disease spread is based on human-to-human contact, therefore a large proportion of the world's population is encapsulated by declaring global lockdown – this in turn has caused adverse snaps of economic instability (IMF, 2020). It is noted that about 213 countries and territories all around the world are encountering this crisis (WHO, 2020) – that includes UAE, dealing with the pandemic with all efforts and trying to cope with situation for maintaining the economic stability. Keeping in view the wreaking situation of COVID-19 crisis, the IT department is pleaded to input innovative measures for contributing to reduce the extreme propensity of COVID-19 crisis (IT News, 2020). Though there are a number of software projects pre-proposed and implemented to facilitate medical domain with various applications such as efficiency of information handling, testing and assessing the cause of a metabolic disruption, investigating the causes behind the fatality of a human organ, scanning the internal system of a human being etc. Along with these applications, the maintenance of patients' record is a mandatory requirement at health care system (Institute of Medicine, 1997). This project is an optimized version of health care information management system that is designed with an extended functionality of integrating the Ministry of health for getting an easy access to the number of patients admitted in the hospital.

Scope of Project

The system is developed to keep the records of patients and their diseases. This system registers the new users and can also search the It will let the hospitals to make the treatment methods easy by keeping record of the departments available in the hospitals. It will give information of availability of beds and doctors. It can easily keep record of patients with several diseases and disease spread rate can be easily known by the system. Fast and easy appointment to the patient can be given. Ministry of health can be aware of the health conditions in an area. This systems' idea is not developed from scratch rather its idea is taken from Hassantunk fire alarm system. Hassantunk system connects the fire alarm of all the buildings with Ministry of Interiors and department of fire. System is developed for virtual communication between patients and hospitals. System has made the searching process easy and less time consuming.

Project Features

- Intact security integration with username and password protection
- Providing a local backup when the server is down
- Allows easy retrieving of data by searching the patients’ record with the unique ID (NIC of patient)
- Displays the statistical record for the number of patients admitted in a hospital
- Prompts a warning message as the threshold limit for maximum number of patients is reached
- A user-friendly application with a GUI incorporation

Project Limitations

Counting on the current approach of system development, there are certain limitations in the proposed application, such as:

- The system require further implementation before a practical usage of application can be rendered
- The system currently doesn’t incorporate the check for re-admission of patients,
- The number of hospitals and the maximum limit for data handling is not ascertained now.
- Server status can be shown at backend only
- The GUI integration for adding user access was not incorporated
- Maximum system load test has not been conducted

These limitations are present due to the restrictions of lockdown, and also because the practical testing of software by allowing multiple interactions with users is not convenient.

Discussion

Assumptions of the system

- System is available online on a network
- System can be accessed from anywhere
- Appointment is given to the patient when he/she comes over the counter (this is made by selecting if the patient requires admission or a walk in examination)
- Hospital module is only accessed by the admin of the hospital.
- Statistics of only the department is shown to the Ministry

Description of the project

The project describes the design and development of a Java based application presenting an optimal design of information management for keeping the records of patients, and presenting that to UAE Ministry, so that the government can take measures to handle the situation such as COVID-19, assessing the number of patients admitted in a hospital, and analyzing the number of hospitals that can admit more patients if the capacity of admitting the patients is reached at maximum. The program was built by using Java programming language, this provides a number of features such as event-driven programming (buttons for login, search, add a patient record, delete a patient record), catch blocks for exception handling, parallel programming with multi-threading, along with this, following approaches describe the main highlights for designing the system application such as:

Multithreading – multithreading was used for attaining maximum CPU utilization by allowing optimality of simultaneous execution of user-driven commands. Two threads were created one as person and the other as patient, both threads were instructed for execution when MySQL DBS connection is timed out the data of persons and patients is written into a text file other than MySQL server. Threading allows the scanning and uploading of data from the text file by reading it line by line, adding data into the MySQL server, as the connection is established, and then deleting the specific data from the text file for supporting memory efficiency.

Classes & Object Oriented System Design – Since Java is a purely object oriented programming language, therefore, it provides enriched features and framework for creating object-based programming by writing classes and sub-classes with their particular functionalities. Our program is based on describing classes for the system, such as Ministry Class, Patients Class, threading class, MySqlConnector class, MySqlController class, health department class, and hospital management class.

Encapsulation – encapsulation provides security integration for ensuring the data members and methods of certain classes cannot interfere with the functionality of other classes. This helps in keeping the system design incorporated with security and efficiency of data integrity. Particular data members of classes were kept encapsulated such as referring to the login details of the person, hospital and Ministry are kept private.

Requirement Collection and Analysis

Functional Requirements

- Register/Login/Logout – Hospital, Ministry department, and patient can register and Login/Logout from system
- Authenticate – System shall authenticate the user ID and passwords of the registered user
- Confirm Appointment – Patients can register for appointment
- Record maintenance – System shall store the record of patients
- Statistical Analysis – System shall perform the statistical analysis according to the number of patients admitted in the particular department of hospital
- Search/delete/update records – the hospital management can search the record of patient, the Ministry office can search the record of each hospital, the hospital can delete the record of patient if the patient is discharged or died
- Backup – the system generates a text document if the application is not connected with

Nonfunctional Requirements

- Authentication – the system generates error when the password is not against the mentioned criteria (the password field must not be left blank), the username criteria (username field must not be left blank, and there must be the use of characters only), the NIC criteria (the flied must not be blank and there must be the use of numeric values only)
- Optimality of speed – the system must not take more than 4 seconds to login/logout, register, and authenticate the functional requirements of the system
- Security integration – the system provides a secure framework for data integration such as compliance with GDPR (general data protection regulations)
- Statistical Analysis – the system generates a warning as soon as the limit for number of patients is reached in a certain department of a particular hospital
- Backup – the system creates text files to store the data of patients, if the connection with SQL server is not established, this file is generated as backup to reconcile the data of patients

Design Phase of the system

Actors: User, System, Admin of the hospital System

Use Case diagram of the system

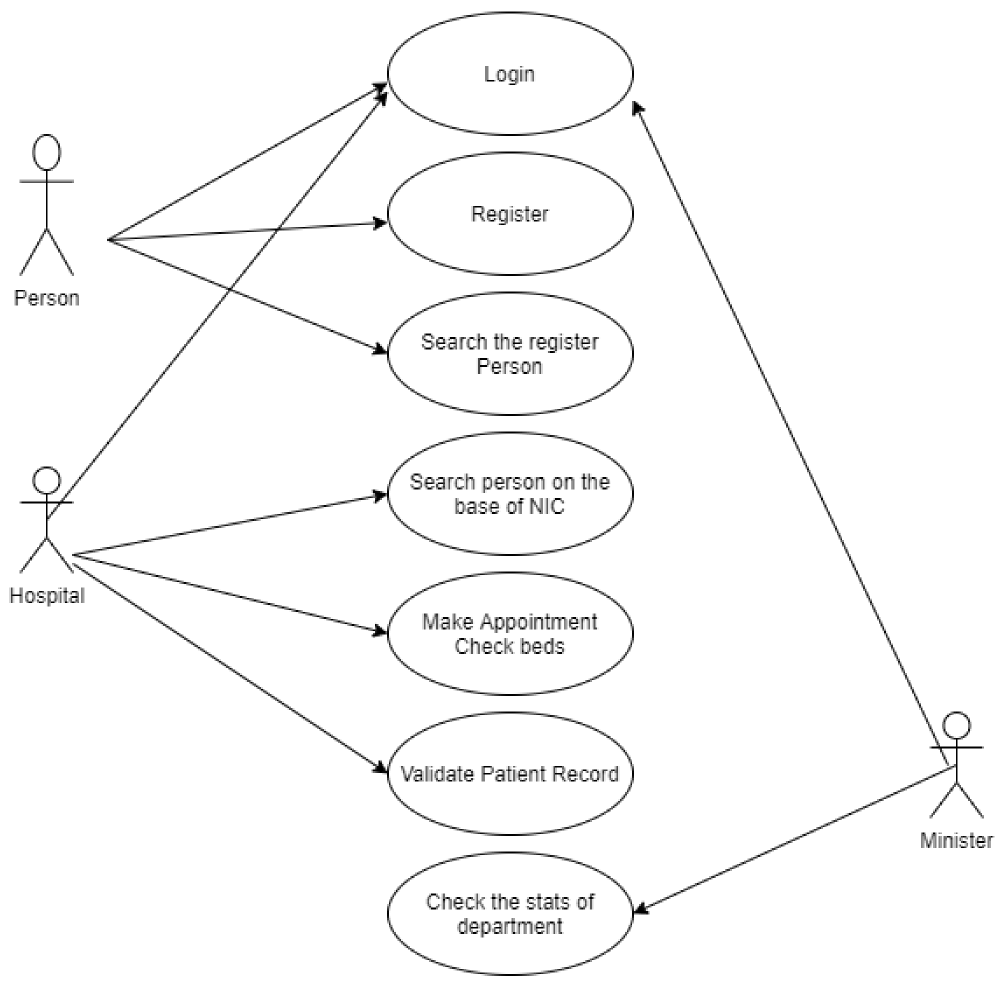

Login
Register
Search the register Person
Search person on the base of NIC
Make Appointment Check beds
Validate Patient Record
Check the stats of department
Person
Hospital
Minister

Figure 1: Use Case Diagram

Use cases for the person entity

Use cases for the person will be able to do following things:

- Person can login to NRC
- Person can add his record and save it in the system
- Person can Search himself
- Person can go back to dashboard by using the back buttons
- Population in the hospital is shown to the user

Uses cases for the Hospital entity

Hospital admin will be able to do following things:

- The hospital admin can login
- Hospital admin can search user on the basis of NIC
- Hospital admin can make appointments
- Hospital admin can search beds for the user
- Hospital admin can delete the appointment
- Hospital admin can go back to the dashboard

Use cases for the Ministry

- Ministry can view the stats in the departments of the hospitals

Entity Relationship of the system

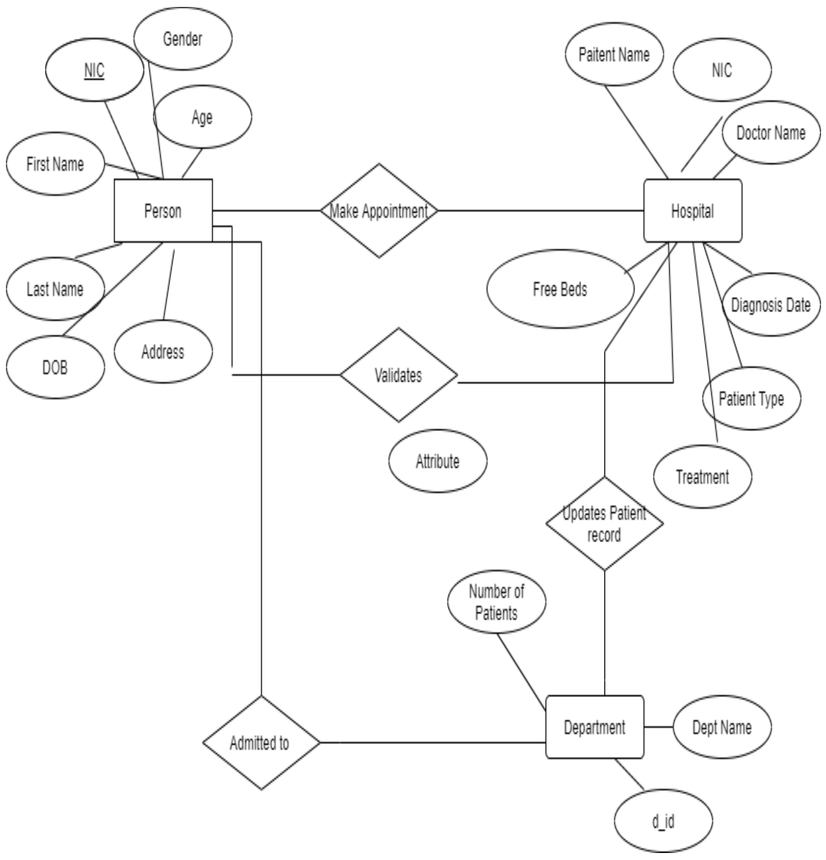
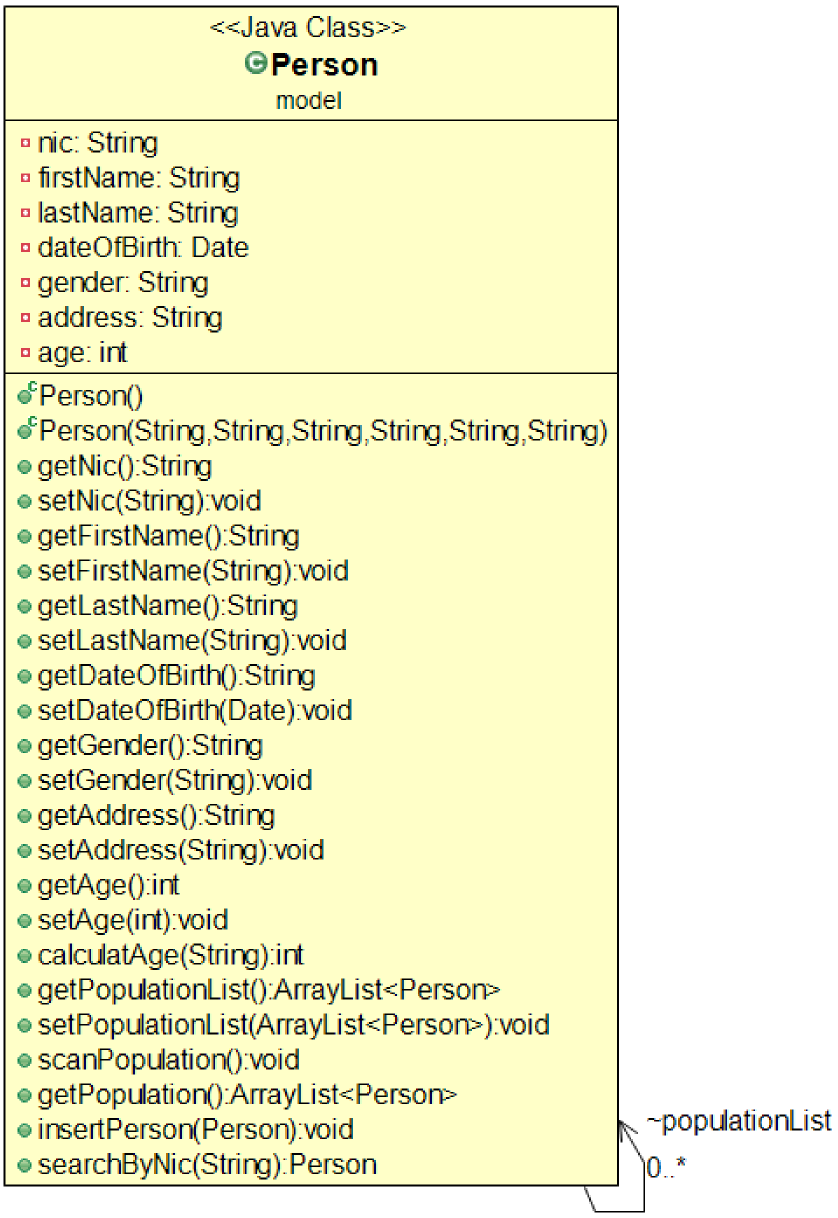


Figure 2: ER Diagram

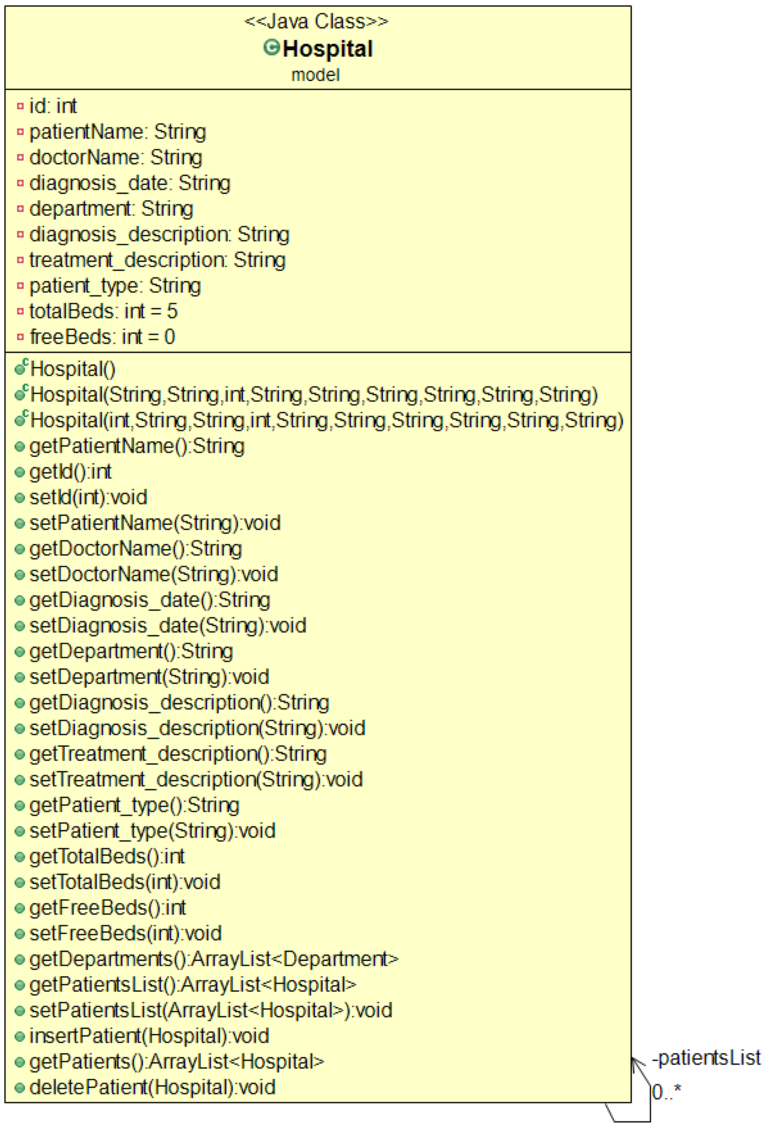
Class Diagram of the System

A Class diagram illustrates the main objects of an application system. Since Java is an object oriented programming language, therefore it incorporates classes, the following snippets explain all the classes designed in this program, such as:

Person Class – Model class importing Java collection files, and MySqlConnector class, along with public accessibility to classes, their methods and data members allowing open access to anyone intended to login or register in the hospital DBS.

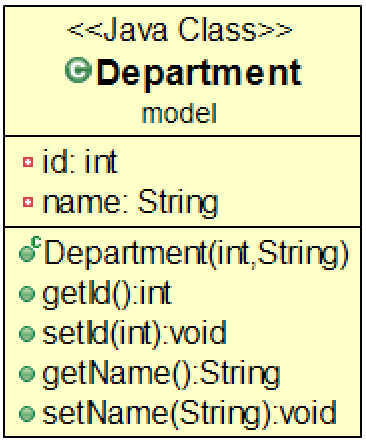


Hospital Class – model class importing util.File, database.MySqlcontroller, and generating record of registered patients by creating methods and data members, this class contains methods allowing to search the record of a particular patient in hospital, deleting the record from hospital, adding information of doctors attending the particular patient, the check for number of beds available if the patient is required to be admitted in the hospital, and such as:

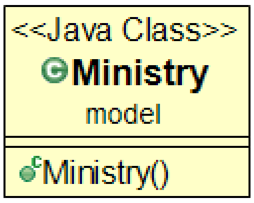


<<Java Class>>
Hospital
model
id: int
patientName: String
doctorName: String
diagnosis_date: String
department: String
diagnosis_description: String
treatment_description: String
patient_type: String
totalBeds: int = 5
freeBeds: int = 0
Hospital()
Hospital(String,String,int,String,String,String,String,String,String)
Hospital(int,String,String,int,String,String,String,String,String,String)
getPatientName():String
getId():int
setId(int):void
setPatientName(String):void
getDoctorName():String
setDoctorName(String):void
getDiagnosis_date():String
setDiagnosis_date(String):void
getDepartment():String
setDepartment(String):void
getDiagnosis_description():String
setDiagnosis_description(String):void
getTreatment_description():String
setTreatment_description(String):void
getPatient_type():String
setPatient_type(String):void
getTotalBeds():int
setTotalBeds(int):void
getFreeBeds():int
setFreeBeds(int):void
getDepartments():ArrayList<Department>
getPatientsList():ArrayList<Hospital>
setPatientsList(ArrayList<Hospital>):void
insertPatient(Hospital):void
getPatients():ArrayList<Hospital>
deletePatient(Hospital):void
-patientsList
0..*

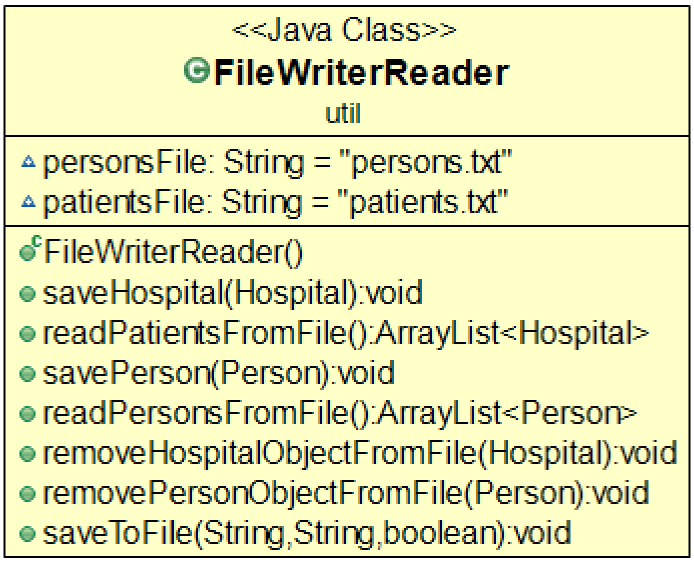
Department Class – this is a model class, containing the data members and methods for displaying the patients record in a particular department of a hospital – such as methods of this class are shown by the following figure:



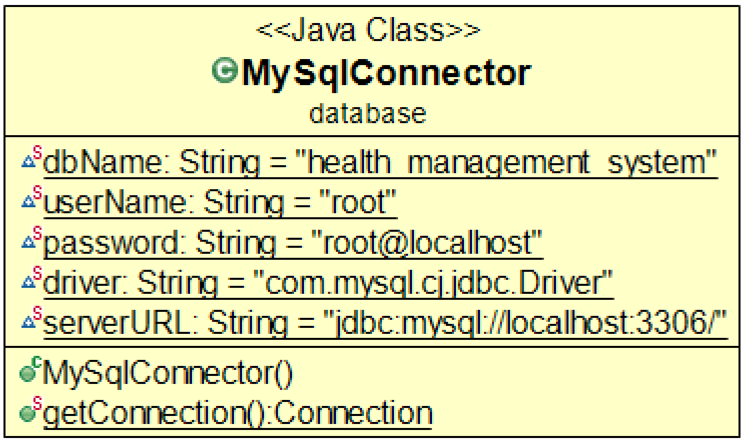
Ministry Class – model class containing only single method that is imported in main method for allowing the execution of assigned function for Ministry, such as viewing the hospital stats for number of patients admitted. The following snippet presents the visual structure of Ministry class:



FileWriterReader – collection class identified by util keyword, importing model classes, other utility classes, .io.Files, and model classes – the function of this class was to write down the user entered data from application into a text file when MySQL DBS is not connected. The methods and data members of this class are represented as:

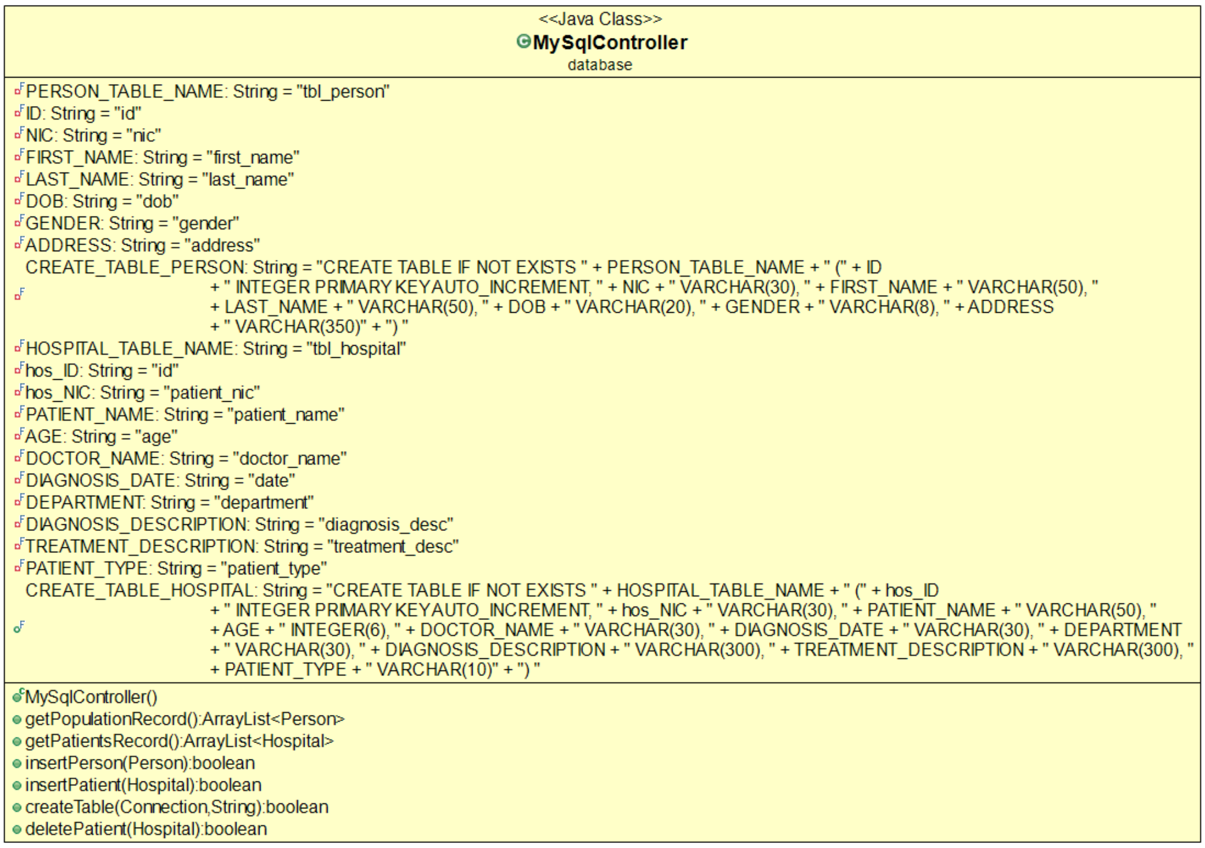

<<Java Class>>
FileWriterReader
util
personsFile: String = "persons.txt"
patientsFile: String = "patients.txt"
FileWriterReader()
saveHospital(Hospital):void
readPatientsFromFile():ArrayList<Hospital>
savePerson(Person):void
readPersonsFromFile():ArrayList<Person>
removeHospitalObjectFromFile(Hospital):void
removePersonObjectFromFile(Person):void
saveToFile(String,String,boolean):void

MySqlConnector – a database class, imports java.sql files, and model classes of person and hospital, allowing the record maintenance. The access modifiers for data members of this program are set as private, ensuring security and encapsulation of data. The MySqlConnector was a piece of code for establishing a connecting MySql Database with the application. The following snippet presents the methods and data members of MySqlConnector class:



<<Java Class>>
MySqlConnector
database
dbName: String = "health_management_system"
userName: String = "root"
password: String = "root@localhost"
driver: String = "com.mysql.cj.jdbc.Driver"
serverURL: String = "jdbc:mysql://localhost:3306/"
MySqlConnector()
getConnection():Connection

MySqlController – as the connection of MySql was established, the MySqlController provides a series of instructions for creating statement, executing queries, and processing results of queries. The following snippet presents a detailed description for all the imported classes, declared methods, and data members for MySqlController class:

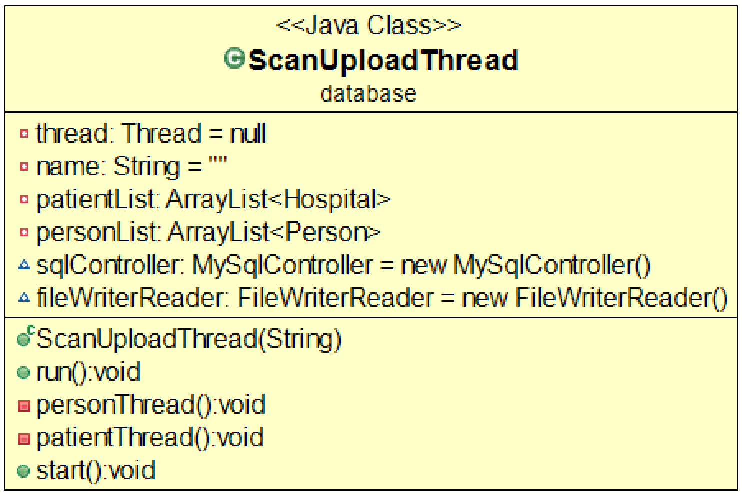


```
<<Java Class>>
MySqlController
database

PERSON_TABLE_NAME: String = "tbl_person"
ID: String = "id"
NIC: String = "nic"
FIRST_NAME: String = "first_name"
LAST_NAME: String = "last_name"
DOB: String = "dob"
GENDER: String = "gender"
ADDRESS: String = "address"
CREATE_TABLE_PERSON: String = "CREATE TABLE IF NOT EXISTS " + PERSON_TABLE_NAME + " (" + ID
                              + " INTEGER PRIMARY KEYAUTO_INCREMENT, " + NIC + " VARCHAR(30), " + FIRST_NAME + " VARCHAR(50), "
                              + LAST_NAME + " VARCHAR(50), " + DOB + " VARCHAR(20), " + GENDER + " VARCHAR(8), " + ADDRESS
                              + " VARCHAR(350)" + ") "
HOSPITAL_TABLE_NAME: String = "tbl_hospital"
hos_ID: String = "id"
hos_NIC: String = "patient_nic"
PATIENT_NAME: String = "patient_name"
AGE: String = "age"
DOCTOR_NAME: String = "doctor_name"
DIAGNOSIS_DATE: String = "date"
DEPARTMENT: String = "department"
DIAGNOSIS_DESCRIPTION: String = "diagnosis_desc"
TREATMENT_DESCRIPTION: String = "treatment_desc"
PATIENT_TYPE: String = "patient_type"
CREATE_TABLE_HOSPITAL: String = "CREATE TABLE IF NOT EXISTS " + HOSPITAL_TABLE_NAME + " (" + hos_ID
                              + " INTEGER PRIMARY KEYAUTO_INCREMENT, " + hos_NIC + " VARCHAR(30), " + PATIENT_NAME + " VARCHAR(50), "
                              + AGE + " INTEGER(6), " + DOCTOR_NAME + " VARCHAR(30), " + DIAGNOSIS_DATE + " VARCHAR(30), " + DEPARTMENT
                              + " VARCHAR(30), " + DIAGNOSIS_DESCRIPTION + " VARCHAR(300), " + TREATMENT_DESCRIPTION + " VARCHAR(300), "
                              + PATIENT_TYPE + " VARCHAR(10)" + ") "

MySqlController()
getPopulationRecord():ArrayList<Person>
getPatientsRecord():ArrayList<Hospital>
insertPerson(Person):boolean
insertPatient(Hospital):boolean
createTable(Connection,String):boolean
deletePatient(Hospital):boolean
```

ScanUploadThread Class – for allowing multithreading to avoid sequential execution. It includes the methods of person threading and patients threading allowing the simultaneous execution for generating person's data and patients' record in hospital. The following snippet represents a detailed view for the data members and methods for thread class:

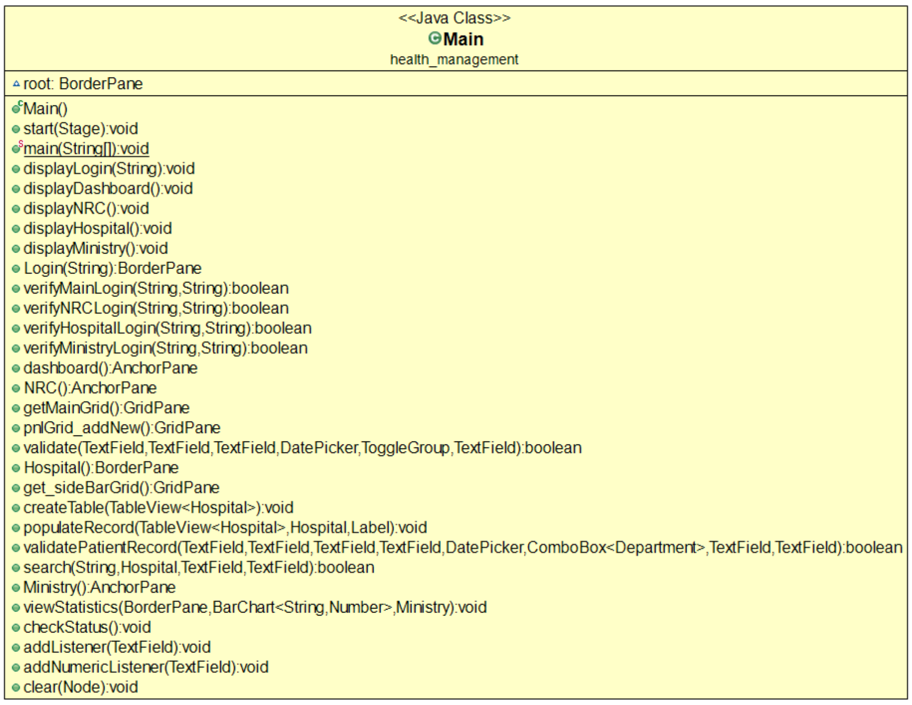


```
<<Java Class>>
ScanUploadThread
database

thread: Thread = null
name: String = ""
patientList: ArrayList<Hospital>
personList: ArrayList<Person>
sqlController: MySqlController = new MySqlController()
fileWriterReader: FileWriterReader = new FileWriterReader()

ScanUploadThread(String)
run():void
personThread():void
patientThread():void
start():void
```

Main – it is the main method of the program that interconnects all the functional activities of the program with each other allowing successful program execution. It includes the importing of all classes, and defining methods creating interconnectivity between the classes and sub-classes of the program, and called methods for executing the program.



```
<<Java Class>>
Main
health_management
root: BorderPane
Main()
start(Stage):void
main(String[]):void
displayLogin(String):void
displayDashboard():void
displayNRC():void
displayHospital():void
displayMinistry():void
Login(String):BorderPane
verifyMainLogin(String,String):boolean
verifyNRCLogin(String,String):boolean
verifyHospitalLogin(String,String):boolean
verifyMinistryLogin(String,String):boolean
dashboard():AnchorPane
NRC():AnchorPane
getMainGrid():GridPane
pnlGrid_addNew():GridPane
validate(TextField,TextField,TextField,DatePicker,ToggleGroup,TextField):boolean
Hospital():BorderPane
get_sideBarGrid():GridPane
createTable(TableView<Hospital>):void
populateRecord(TableView<Hospital>,Hospital,Label):void
validatePatientRecord(TextField,TextField,TextField,TextField,DatePicker,ComboBox<Department>,TextField,TextField):boolean
search(String,Hospital,TextField,TextField):boolean
Ministry():AnchorPane
viewStatistics(BorderPane,BarChart<String,Number>,Ministry):void
checkStatus():void
addListener(TextField):void
addNumericListener(TextField):void
clear(Node):void
```

Database of the System

Database includes the table for person having NIC as primary key and other attributes like first name, last name, address, gender and date of birth of the patient. Database has table for hospital which includes NIC as foreign key and doctor name, patient age, patient name, diagnosis date, department, patient type. Third table includes the department information contains NIC as foreign key and contains department name as attribute and department id as a primary key.

This is a relational database and hospital has one-to-many relationships with the person table and department has one-to-many relationships with the person and hospital entity.

Design Methodology

System has a relational database and developed using Java language and Eclipse IDE. Front end is developed using java Graphical user interface classes. Object oriented programming concepts are primarily use for the development of the system Backend is developed using SQL database. Offline data access is also given to the system. System is developed in the form of modules. Modules integrate to develop a full system.

The system design was based on MVC (Model View Controller) pattern. It incorporates the division of system in three main classes, such as model classes, view classes, and controller classes.

There are particular features of Java programming language that were utilized to optimize the functionality of software application, such as use of multithreading, java collection classes, JavaFX (for creating Pane class behaving as a base class for all layout panes), and package model functions, such as:

Multi-threading

The application was built by using Java programming language that is an object-oriented language. For incorporating an optimal approach in the design multithreading approach was used. Multithreading in an optimal feature of Java that provides the facility to carry out concurrent execution with simultaneous function calls to ensure CPU utilization is taken at maximum range. According to this approach, each part of program is termed as thread; therefore multiple threads were designed in this application. This brings technical efficiency and implies a light-weight process within the actual process for executing a set of commands. Threading is done with two mechanisms, such as:

- Implementing threading in the class
- Incorporating the implementation of the runnable interface

For instance, the methodology follows by creating a class that is used to implement an extension for the java.lang.Thread class. Accordingly this class works for overriding the run() method in the thread class. The life of thread is initiated from the run() method, where objects for creating the new class, and calling start() method are created, and thus starting the execution of thread.

Java Packages

- Java Packages are a feature of Java that allows the grouping of related classes. It is similar to a folder found in a file directory. The program was based on writing Java Packages for ascertaining the optimal features of Java Packages, such avoiding class name conflicts, and providing an optimal infrastructure of program design. It provides two kinds of packages, such as built-in packages and user defined packages, both ways were used in the program design.

Java Model Classes

- The Java Model classes provide a data-centric structure for classes to allow encapsulation of closely related items in classes. Moreover it supports the MVC infrastructure allowing a model files to act as a bridge for the views and controller files of Java program. Yet, these classes are independent of how the data components are rendered or viewed. Therefore Java Model classes were created for ensuring an optimal program structure is established.

Java Collection files

- the util.File feature of Java programming was deployed for using the file package to ensure the access for the Java virtual interface to access files in the system, specifying and evaluating file attributes, and file classes. This includes the method of Java collection files that provides support to several objects with supporting array declarations. This approach was used to create the code for FileWriterReader.Java codes.

JavaFX

- JavaFX referring to Java “special effects” is an optimal feature of Java programming that allows the design of application for making multiple platforms execution, and providing built-in functions for enhancing the optimality of the program. The program was first built on sceneBuilder, and then it was changed by keeping a complete focus on javaFX features.

Source Code Explanation

The program was based on class oriented infrastructure incorporating a number of methods related to event-driven programs

- Main.Java
  The main method in Java program includes the importing of all Java packages, providing a data-centered infrastructure to execute the program. The code of main method includes the catch block for code for exception handling, try block of code having the piece of code that may generate errors, the finally block that runs in any case regardless of the exception occurrence. Some part of the snippet is taken from the code to show the function of main method.

```
publicclass Main extends Application {
      BorderPane root;
      @Override
      publicvoid start(Stage primaryStage) {
            try {
                  root = new BorderPane();
                  Scene scene = new Scene(root, 1000, 600);
      scene.getStylesheets().add(getClass().getResource("applic
ation.css").toExternalForm());
                  primaryStage.setScene(scene);
                  primaryStage.setTitle("Health Management");
                  primaryStage.show();
                  displayLogin("main");
/*** These are the two threads which starts when application
start Responsible for* upload data to the MySql server from
textFiles There are two textFiles "persons.txt"
and"patients.txt" persons.txt file is used to store the data *
of Person class Object when Internet is down patients.txt file
is used to * store the data of Hospital class Object when
Internet is down This thread* scans the data from file line by
line and delete object data from file after * successful insert
to MySql server*/
```

```
ScanUploadThread personThread = new
ScanUploadThread("personThread");
                  personThread.start();
                  ScanUploadThread patientThread = new
ScanUploadThread("patientThread");
                  patientThread.start();

      /*** to stop child threads when main thread stops */

                  primaryStage.setOnCloseRequest(e -> {
                        Platform.exit();
                        System.exit(0);
                  });

            } catch (Exception e) {

                  e.printStackTrace();

            }
```

```
/** is called when Application starts to Display Dashboard*/

      publicvoid displayDashboard() {

            root.getChildren().clear();
            root.setCenter(dashboard());

      }

/** is called when "National Registration button clicked on
Dashboard" */

      publicvoid displayNRC() {
            root.getChildren().clear();
            root.setCenter(NRC());
      }
/** is called when "Hospital" button clicked on Dashboard    */

      publicvoid displayHospital() {

            root.getChildren().clear();
            root.setCenter(Hospital());
      }

/** is called when Ministry button clicked on Dashboard */

      publicvoid displayMinistry() {
            root.getChildren().clear();
            root.setCenter(Ministry());
            checkStatus();
      }
/***=======================Login===================================*/
      public BorderPane Login(String login_desc) {
            BorderPane borderPane = new BorderPane();
            borderPane.setMaxHeight(Double.MAX_VALUE);
            borderPane.setMaxWidth(Double.MAX_VALUE);

      .
      .
      .
      }
```

The main approach of main method was first built on the basis of sceneBuilder, but afterwards the JavaFX approach was deployed that caused the following changes in the program code:

```
package health_management; //Note:- This was first built on sceneBuilder
but then due to preference of just using javaFx was changed
import java.time.DayOfWeek; import java.time.LocalDate; import
java.time.format.DateTimeFormatter; import database.ScanUploadThread;
import model.Person; import model.Department; import model.Hospital;
import model.Ministry; import javafx.application.Application; import
javafx.application.Platform; import javafx.beans.value.ChangeListener;
import javafx.collections.FXCollections; import
javafx.collections.ObservableList; import javafx.geometry.HPos; import
javafx.geometry.Insets; import javafx.geometry.Pos; import
javafx.geometry.VPos; import javafx.stage.Stage; import
javafx.util.Callback; import javafx.util.StringConverter; import
javafx.scene.Node; import javafx.scene.Scene; import
javafx.scene.chart.BarChart; import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis; import javafx.scene.chart.XYChart;
import javafx.scene.control
```

- Person.Java

  This class consists of private data members to set NIC, name, date of birth of the person, age calculation method is written to calculate the age of person by counting the period from date of birth to the current date. Moreover the class contains the methods to show the population of persons, and option for searching the person record by entering the NIC value. The following piece of code presents some main features of this class:

```
public class Person {
      private String nic;
      private String firstName;
      private String lastName;
      private Date dateOfBirth;
      private String gender;
      private String address;
      private int age;
      private ArrayList<Person> populationList;
      .
      .
      .
public ArrayList<Person> getPopulationList() {
            return populationList;
      }
      public void setPopulationList(ArrayList<Person> populationList)
{
            this.populationList = populationList;
      }
      public void scanPopulation() {
            populationList = new ArrayList<Person>();
            populationList.addAll(getPopulation());
      }
      public ArrayList<Person> getPopulation() {
            ArrayList<Person> list = new ArrayList<>();
            MySqlController sqlController = new MySqlController();
            list = sqlController.getPopulationRecord();
            FileWriterReader obj=new FileWriterReader();
            list.addAll(obj.readPersonsFromFile());
            return list;
      }
      public void insertPerson(Person person) {
            populationList.add(person);
            MySqlController sqlController = new MySqlController();
            if(sqlController.insertPerson(person)) {
            }
            else {
                  System.out.println("============>dataBase error");
//write to local file if can't insert successfully to MySql database
                  FileWriterReader obj=new FileWriterReader();
                  obj.savePerson(person);
            }
      }
      public Person searchByNic(String _nic) {
            Person person = new Person();
            return person;
      }
}
```

- Department.Java

The department class consists of private data members such a name of patient admitted or associated with consultation for the department, and ID of the person associated with the department. The following code shows this functionality:

```
package model;
public class Department {
    private int id;
    private String name;
    public Department(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

- Hospital.Java

The hospital class consists of the patients' details, doctor details, department information, and the methods to connect the MySQL with hospital class for data retrieval, and with the text files to write data when the connection with MySQL is not established. The following code describes some of the methods and data members of this class:

```
public class Hospital extends Person {
      private ArrayList<Hospital> patientsList = null;
      private int id;
      private String patientName;
      private String doctorName;
      private String diagnosis_date;
      private String department;
      private String diagnosis_description;
      private String treatment_description;
      private String patient_type;
      private int totalBeds = 5;
      private int freeBeds = 0;
      public Hospital() {
            super();
            patientsList = new ArrayList<Hospital>();
            patientsList.addAll(getPatients());
      }
```

Codes describing the retrieval and insertion of data:

```
public ArrayList<Hospital> getPatientsList() {
            return patientsList;
      }
      public void setPatientsList(ArrayList<Hospital> patientsList)
{
            this.patientsList = patientsList;
      }
      // will insert data into database
      public void insertPatient(Hospital hospital) {
            this.patientsList.add(hospital);
            MySqlController sqlController = new MySqlController();
            if (sqlController.insertPatient(hospital)) {
            } else {
                  System.out.println("============>dataBase error");
                  // write to local file if can't insert
successfully to MySql database
                  FileWriterReader obj = new FileWriterReader();
                  obj.saveHospital(hospital);
            }
      }
```

- Ministry.Java

  This performs function for assessing hospital record and viewing the statistics, while a warning is displayed as the maximum limit for patients is reached:

```
package model;
public class Ministry extends Hospital{
public      Ministry(){

      }
}
```

- FileWriterReader.Java

  This codes was written to handle the data of persons and patients when MySQL server is not connected – this data handling was carried out by creating text files for persons and patients, by writing down the user entered data from application to text files (when MySQL database is not connected).

```
public class FileWriterReader {
      String personsFile = "persons.txt";
      String patientsFile = "patients.txt";
      /*** Converts the object of Hospital to String and insert into
patients.txt File ***/
      public void saveHospital(Hospital patient) {

            String objectString = patient.getNic() + "|" +
patient.getPatientName() + "|" + patient.getAge() + "|"
                        + patient.getDoctorName() + "|" +
patient.getDiagnosis_date() + "|" + patient.getDepartment() + "|"
                        + patient.getDiagnosis_description() + "|" +
patient.getTreatment_description();

            saveToFile(patientsFile, objectString, true);
      }
      /*** Converts the object of person to string and insert into
persons.txt File ***/
      public void savePerson(Person person) {
            String objectString = person.getNic() + "|" +
person.getFirstName() + "|" + person.getLastName() + "|"
                        + person.getDateOfBirth() + "|" +
person.getGender() + "|" + person.getAddress();

            saveToFile(personsFile, objectString, true);
      }


      }
}
```

```
// will retrieve data from database
      public ArrayList<Hospital> getPatients() {
            ArrayList<Hospital> list = new ArrayList<>();
            MySqlController sqlController = new MySqlController();
            FileWriterReader obj = new FileWriterReader();
            list = sqlController.getPatientsRecord();
            list.addAll(obj.readPatientsFromFile());
            return list;
      }
      public void deletePatient(Hospital hospital) {
            this.patientsList.remove(hospital);
            MySqlController sqlController = new MySqlController();
            if(sqlController.deletePatient(hospital)) {
                  System.out.println("==================>Patient
Record Deleted Successfully");
            }
      }
```

- MySqlConnector.Java

The MySqlConnector class contains codes describing the connection of MySQL server with the application data. The following codes describe this:

```
public class MySqlConnector {
      /*** Database Name* */
       static String dbName = "health_management_system";
       /*** Database User* */
       static String userName = "root";
       /** Database Password* */
       static String password = "root@localhost";
       /*** Driver*/
       static String driver="com.mysql.cj.jdbc.Driver";
       /**Server URL* */
       static String serverURL="jdbc:mysql://localhost:3306/";
       /***Returns the connection of database* */
       public static Connection getConnection() {
             try {
                  Class.forName(driver);
                  Connection  connection =
DriverManager.getConnection(serverURL+ dbName,…..}
```

- MySqlController.Java

This class works to control the queries for searching record, and deleting it. There were tables for hospital, person, patients, and departments created, allowing the query execution. The following codes describe this approach:

```
public class MySqlController {
      /**Creating Person Table **/
      private final String PERSON_TABLE_NAME = "tbl_person";
      private final String ID = "id";
      private final String NIC = "nic";
      private final String FIRST_NAME = "first_name";
      private final String LAST_NAME = "last_name";
      private final String DOB = "dob";
      private final String GENDER = "gender";
      private final String ADDRESS = "address";
      private final String CREATE_TABLE_PERSON = "CREATE TABLE IF NOT
EXISTS " + PERSON_TABLE_NAME + " (" + ID
                  + " INTEGER PRIMARY KEY AUTO_INCREMENT, " + NIC + "
VARCHAR(30), " + FIRST_NAME + " VARCHAR(50), "
                  + LAST_NAME + " VARCHAR(50), " + DOB + " VARCHAR(20),
" + GENDER + " VARCHAR(8), " + ADDRESS
                  + " VARCHAR(350)" + ") ";
………….}
```

- ScanUploadThread.Java

This supports the multithreading approach, there were two threads one for person, and the other for patient created allowing avoidance to sequential programming, and reading data from text file and writing it into the MySQL server when the connection with server is established. The following codes explain this approach:

```
public class ScanUploadThread implements Runnable{
      private Thread thread = null;
      private String name = "";
      private ArrayList<Hospital> patientList;
      private ArrayList<Person> personList;
      MySqlController sqlController = new MySqlController();
      FileWriterReader fileWriterReader = new FileWriterReader();
      public ScanUploadThread(String _name) {
            this.name=_name;
      }
      @Override
      public void run() {
            while(true) {
                  if(name.equals("personThread")) {
                        personThread();
                  }else if(name.equals("patientThread")) {
                        patientThread();
                  }
```

```
      private void personThread() {
            personList = fileWriterReader.readPersonsFromFile();
            if (personList != null && personList.size() > 0) {
                  for (Person person : personList) {
                        if (sqlController.insertPerson(person)) {
                  fileWriterReader.removePersonObjectFromFile(person);
                              //stop thread for three seconds after each
record inserted from file to database
                              try {
                                    Thread.sleep(3000);
                              } catch (InterruptedException e) {
                                    e.printStackTrace();
                              }
                        }
                  }
                  personList.clear();
            } else {
                  System.out.println("==========>No data Available in
personsFile to Upload");
            }
      }
```

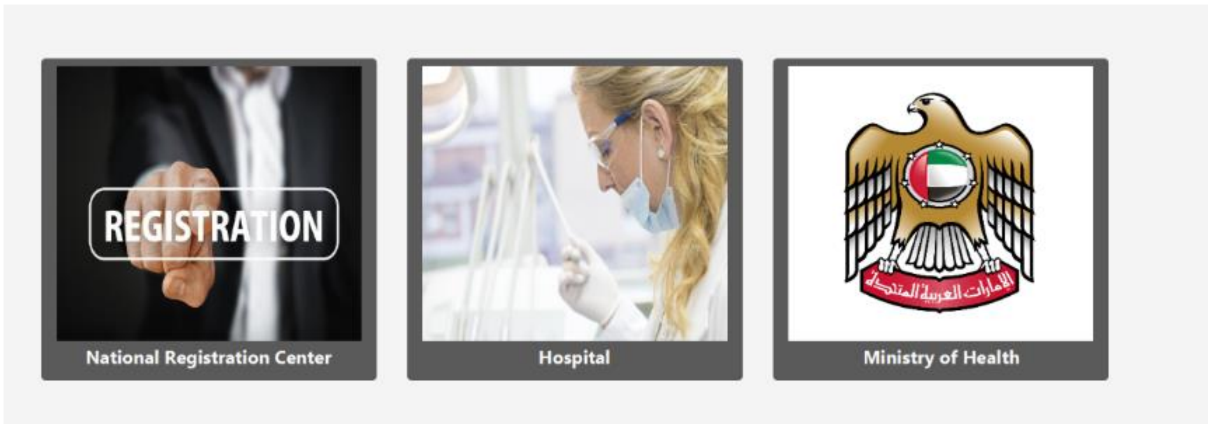
```
private void patientThread() {
            patientList = fileWriterReader.readPatientsFromFile();
            if (patientList != null && patientList.size() > 0) {
                  for (Hospital patient : patientList) {
                        if (sqlController.insertPatient(patient)) {

      fileWriterReader.removeHospitalObjectFromFile(patient);
                              //stop thread for three seconds after
each record inserted from file to database
                              try {
                                    Thread.sleep(3000);
                              } catch (InterruptedException e) {
                                    e.printStackTrace();
                              }
                        }
                  }
                  [illegible]
```

Results

Interface Design

The interface of the system was developed as:



The NRC Screen was designed as:

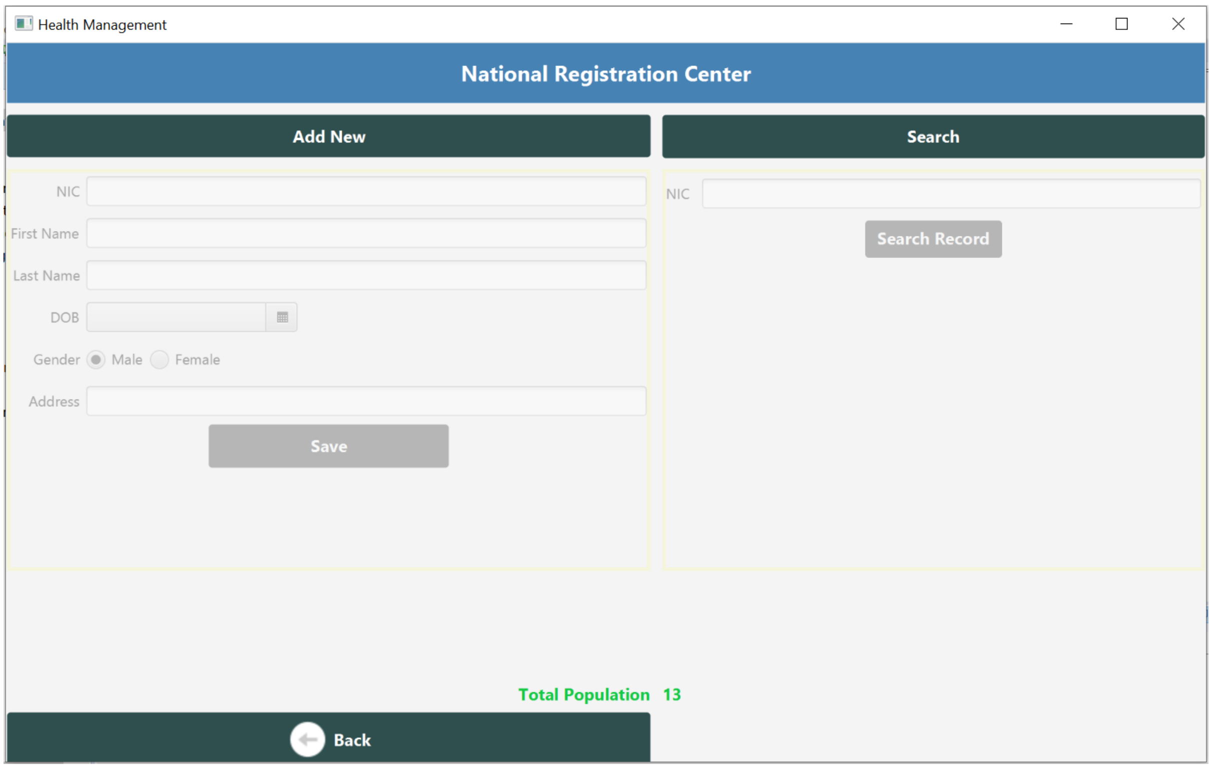


Figure 3: NRC Screen

Login page for every department (the snippet is taken from hospital login, but the same format is followed for every login page)

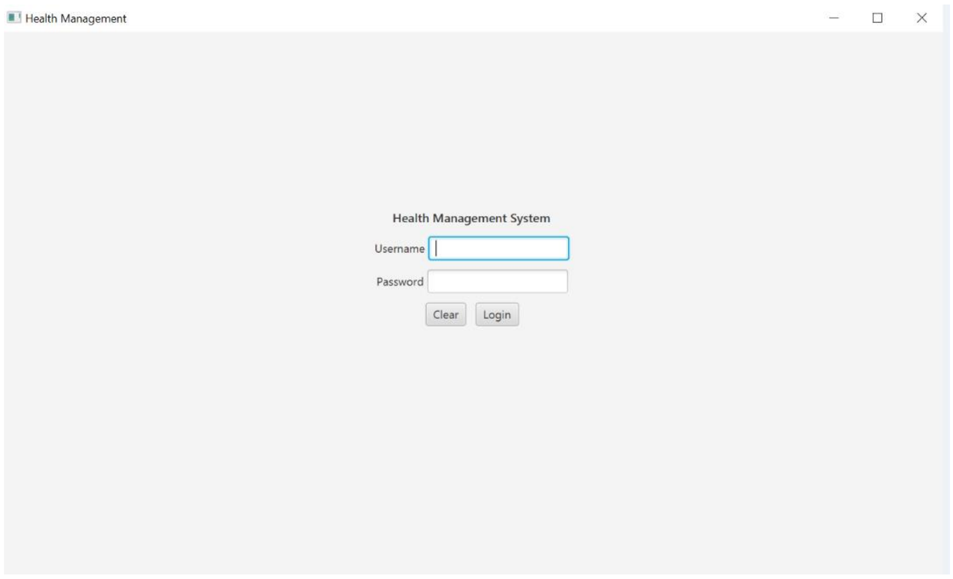


Figure 4: Login page for health management system

The statistics for the system

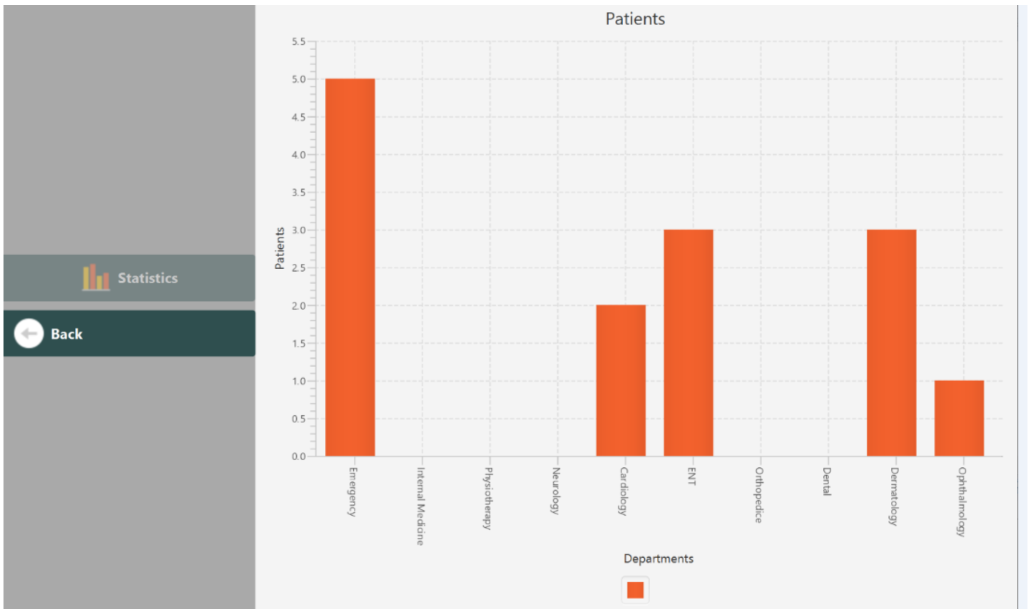
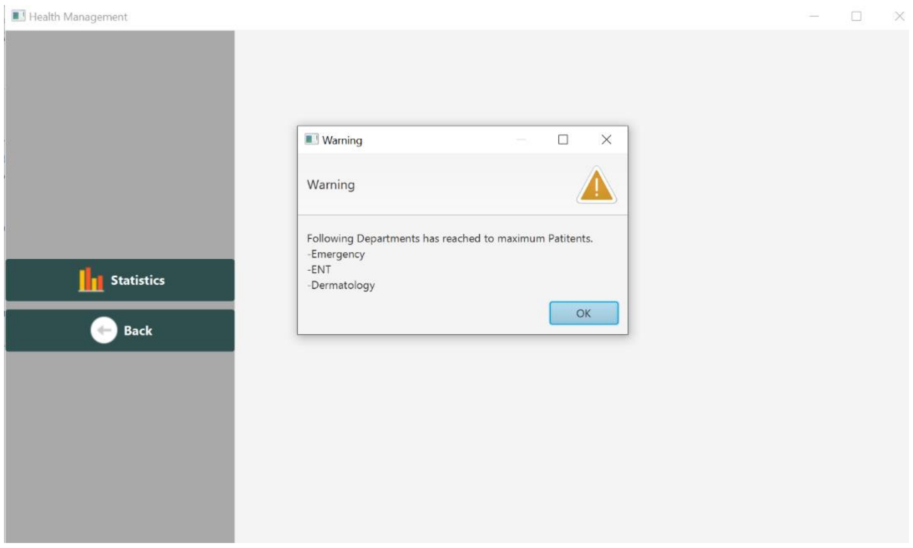


Figure 5: Statistical description of patient's record

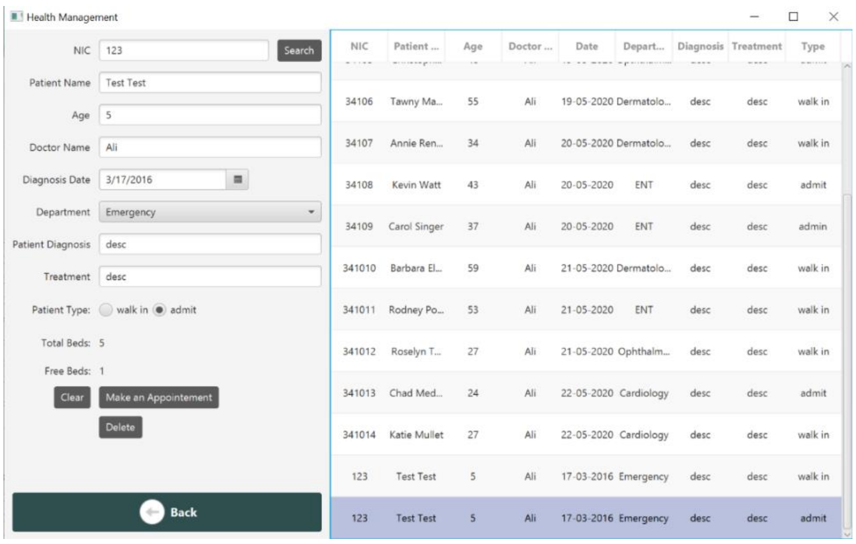


Warning

Following Departments has reached to maximum Patitents.
-Emergency
-ENT
-Dermatology

Figure 6: Warning message generated as soon as the limit for number of patients is reached

Description of patients' data to health management sector



NIC	123
Patient Name	Test Test
Age	5
Doctor Name	Ali
Diagnosis Date	3/17/2016
Department	Emergency
Patient Diagnosis	desc
Treatment	desc
Patient Type:	walk in [✓ admit]
Total Beds:	5
Free Beds:	1

NIC	Patient ...	Age	Doctor ...	Date	Depart...	Diagnosis	Treatment	Type
34106	Tawny Ma...	55	Ali	19-05-2020	Dermatolo...	desc	desc	walk in
34107	Annie Ren...	34	Ali	20-05-2020	Dermatolo...	desc	desc	walk in
34108	Kevin Watt	43	Ali	20-05-2020	ENT	desc	desc	admit
34109	Carol Singer	37	Ali	20-05-2020	ENT	desc	desc	admin
341010	Barbara El...	59	Ali	21-05-2020	Dermatolo...	desc	desc	walk in
341011	Rodney Po...	53	Ali	21-05-2020	ENT	desc	desc	walk in
341012	Roselyn T...	27	Ali	21-05-2020	Ophthalm...	desc	desc	walk in
341013	Chad Med...	24	Ali	22-05-2020	Cardiology	desc	desc	admit
341014	Katie Mullet	27	Ali	22-05-2020	Cardiology	desc	desc	walk in
123	Test Test	5	Ali	17-03-2016	Emergency	desc	desc	walk in
123	Test Test	5	Ali	17-03-2016	Emergency	desc	desc	admit

Figure 7: The description of record to health management system

Testing Phase

The testing methodologies as standard for testing an application program includes A/B test, Black box Testing, and White box testing, and User testing. Since the standard methods of testing application ensures the optimality of system design, yet due to time constraints and restriction of COVID-19 lockdown, these methodologies were become restricted, and the application was tested manually, by using White Box Testing method, ensuring the technical features of the program works accordingly, the MySQL connection was established with success and the intended functionality of the program was successfully achieved.

Conclusion and Recommendations

The application was built successfully yet, some of the future developments are required, as when the close() function is not called in the ScanUpload class, the loop for writing data into text file runs infinity times. Moreover the application requires appropriate implementation for checking duplicate data, so that record maintenance for patient readmission can be approached. Setting the server capacity to deal with a maximum number of user accesses is also required as recommendation. This ascertains the number of hospitals system can incorporate defining the memory efficiency of the system. Some of the features were not implemented due to time constraints, and COVID-19 lockdown restrictions. Therefore further development process for this application is expected before its practical implementation.

References

Harrison, J., & Palacio, C. (2006). The Role of Clinical Information Systems in Health Care Quality Improvement. *The Health Care Manager*, *25*(3), 206-212. https://doi.org/10.1097/00126450-200607000-00003

IMF. (2020). *International Monetary Fund*. Retrieved 6 June 2020, from https://www.imf.org/en/Topics/imf-and-covid19.

IT News. (2020). *COVID-19: UAE's Pure Health secures InterSystems TrakCare deal*. Healthcare IT News. Retrieved 6 June 2020, from https://www.healthcareitnews.com/news/europe/covid-19-uae-s-pure-health-secures-intersystems-trakcare-deal.

WHO. (2020). *Coronavirus*. Who.int. Retrieved 6 June 2020, from https://www.who.int/emergencies/diseases/novel-coronavirus-2019.