WolfHospital Management System

For WolfHospital

CSC 440 Database Management Systems

Project 3

Team A

Renata Zeitler, Andy Zhang, Nicole Hlebak, Alex Sawyer

September 25th, 2017

Assumptions:

1. We are not going to check duplicate SSN for patients.
2. No patient will start multiple treatments on the same day- only the day/month/year is recorded.
3. Patients can reserve a specific ward/bed, but they must tell the receptionist at check-in.
4. To check into a ward, SSN is not required since it is optional patient data; however, patientID is.
5. Each type of fee is the same for every patient (for example, a registration fee is always the same amount).
6. The nurse in charge of each ward attends to the patients in the ward. Nurses do not attend to or treat patients outside of the ward they are assigned to.
7. Only one doctor will be performing a test.
8. Patients are limited to one ward/bed per visit; this will not change throughout their visit.
9. Medical records will be filled out by doctors which will record them “treating” a patient. Doctors will not treat a patient outside of recording it in a medical record (unless they are a specialist performing a test).
10. A test’s name will not determine its cost, as variations from test to test could cause a fluctuation in the test’s price.
11. A reserved ward for a patient will be given a start date that is in the future, for when the patient is supposed to arrive.
12. The ward usage percentage represents the current percentage of the word that is occupied.
13. The operation “report the medical history for a given patient and for a certain time period (month/year)” was interpreted as give a patient's medical history for a certain amount of time.

Transaction 1: Assign Patient to a ward

```java
DBManager.beginTransaction(); //The start of the transaction

int MRID = Input.getInt("Enter medical record ID: ");
int wardNumber = Input.getInt("Enter Ward Number: ");
int bedNumber = Input.getInt("Enter Bed Number: ");

String sqlRemove = "DELETE FROM PatientResidesIn WHERE MRID=" + MRID;
String sqlInsert = "INSERT INTO PatientResidesIn VALUES(%d, %d, %d)";
sqlInsert = String.format(sqlInsert, MRID, wardNumber, bedNumber);

//If the patient is currently in another ward, remove them
if(!DBManager.execute(sqlRemove)) {
  System.out.println("Error removing patient from current ward");
  DBManager.rollbackTransaction();
  //If there is an error removing the patient from their current ward
  //Roll the transaction back to the beginning
  return;
}

//Insert the patient into the specified ward
if(!DBManager.execute(sqlInsert)) {
  System.out.println("Error adding patient to new ward");
  DBManager.rollbackTransaction();
  // If there is an error adding the patient to a ward (for
  // the PatientResidesIn table), then roll the transacion
  // back to the beginning
  return;
}
DBManager.commitTransaction(); //Commit the transaction
```

Transaction 2: Insert Staff member

```
   DBManager.beginTransaction(); //Start the transaction

   String sql = "INSERT INTO StaffMembers VALUES(%d, '%s', %d, '%s', '%s',
'%s', '%s', '%s', '%s')";
   String sqlJob = "INSERT INTO %s VALUES(%s)";

   //Get the staff member info
   int staffID = Input.getInt("Enter staff member ID: ");
   String name = Input.getString("Enter name: ");
   int age = Input.getInt("Enter age: ");
   String gender = Input.getString("Enter gender: ");
   String jobTitle = Input.getString("Enter job title: ");
   String professionalTitle = Input.getString("Enter professional title:
");
   String department = Input.getString("Enter department: ");
   String phone = Input.getString("Enter phone number: ");
   String address = Input.getString("Enter address: ");

   sql = String.format(sql, staffID, name, age, jobTitle,
professionalTitle, gender, department, phone, address);
   //Attempt to insert staff member into StaffMembers
   if(!DBManager.execute(sql)) {
     //Rollback if it failed
     System.out.println("Couldn't add staff member");
     DBManager.rollbackTransaction();
     return false;
   }

   //Attempt to insert staff member into corresponding job table
   //Like Doctors, Nurses, Receptionists
   jobTitle = getJobTable(jobTitle);
   if(jobTitle != null) {
     sqlJob = String.format(sqlJob, jobTitle, staffID);
     if(!DBManager.execute(sqlJob)) {
       //Rollback if it failed
```

```
      System.out.println("Couldn't add staff to the table: " + jobTitle);
      DBManager.rollbackTransaction(); //Rollback if failed
      return false;
    }
  }

  DBManager.commitTransaction(); //Commit the transaction
```

Design Decisions:

The system has a main menu that gives the user the following options: “Exit”, “Patients”, “Staff”, “Medical Record”, “Wards”, “Billing”, and “Reports”. When prompted, the user enters a number 0-6 corresponding with what kind of action they would like to take. For each menu option there is a submenu displayed that has specific operations listed. Our system has a main class (Main.java) that facilitates switching between the main menu options and separate classes classes for each related group of operations. This was done to break the application into more manageable and maintainable pieces of code.

Our program also contains two helper classes. One helper class is DBManager, which, through the use of a shared database connection, provides the ability to issue SQL queries and perform transactions through various utility methods. The other class is Input, which is a class that attempts to abstract the details of getting different types of input from the user, and includes a method for menu formatting, as well as a helper function that assists with constructing SQL UPDATE statements.

Functional Roles:

Part 1:

- **Software Engineer:** Renata (Prime), Nicole (Backup)
- **Database Designer/Administrator:** Nicole (Prime), Andy(Backup)
- **Application Programmer:** Andy (Prime), Alex(Backup)
- **Test Plan Engineer:** Alex(Prime), Renata(Backup)

Part 2:

- **Software Engineer:** Alex (Prime), Andy (Backup)
- **Database Designer/Administrator:** Andy (Prime), Renata (Backup)
- **Application Programmer:** Renata (Prime), Nicole (Backup)
- **Test Plan Engineer:** Nicole (Prime), Alex (Backup)

Part 3:

- **Software Engineer:** Andy (Prime), Renata(Backup)
- **Database Designer/Administrator:** Alex(Prime), Andy(Backup)
- **Application Programmer:** Nicole (Prime), Alex(Backup)
- **Test Plan Engineer:** Renata (Prime), Nicole(Backup)