**SWE 205 – Introduction to Software Engineering**

Project – Hereditary Blood Disorder Center (HBDC)

*Al-Khobar Central Hospital*

PHASE 3

The software design document is prepared for programmers and future maintainers of the system, to specify the detailed architectural structure of the software. The document discusses the software design scope where we talk about major software functions, design constraints and other requirements. We further discuss the Object Oriented design of the software with high level package diagrams and components. The domain model of the software is further refined by the team by using sequence diagrams. Lastly, the team decides on whether to provide the planned interface with additional data display screens or input output screens.

1. **Software Design Scope**
2. *Major Software Functions*

This system helps the Al-Khobar Central Hospital to keep track of patients and their treatment history which will help to build up research and increase level of care for the patients. The *HDBC Tracker* is a very simple menu based system which allows usage to doctors and receptionist. This system is maintained by a system administrator who can add, modify or delete users from the system.

The receptionist can add new patients to the database and schedule appointments for them.

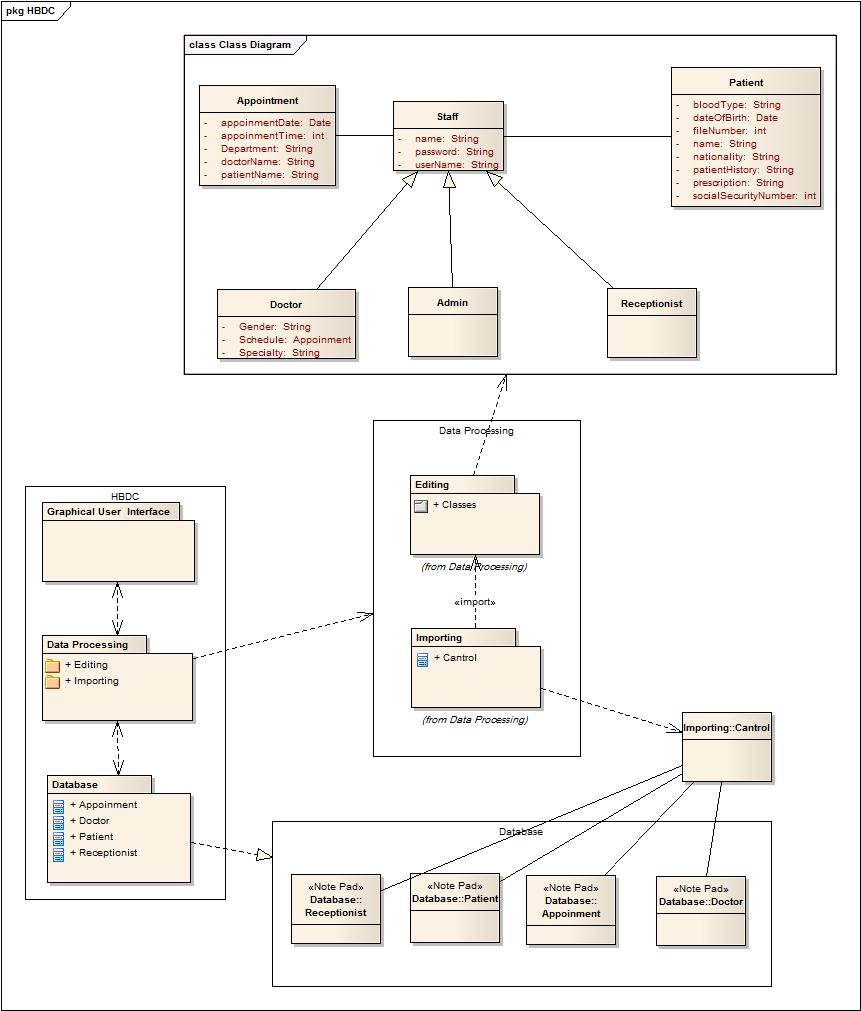
The doctors can retrieve patient information and the visit history of the patient as well as update the history of the patient.

1. *Major Design Constraints and other requirements*

Some of the major difficulties our team has faced over the weeks are listed as follows:

* Over the period of time as we started understanding the system requirements in a better way, we had to make changes in our documents and our planning.
* Initially, we were not completely sure of how we are going to design, plan and then implement all the things planned. But now in phase three, things seem to be very clear and now we have a complete understanding of the software.
* We had to learn a new software called NetBeans which helped us design better GUI interfaces for user friendliness which took up a lot of time.
* We had certain amount of difficulty in deciding the number of text files to be used as databases for the system.
* Lastly, our lack of experience in the software development field also proved to be a major constraint.

1. **Object Oriented Design**
2. *High Level package diagrams and components*

**

1. *Refinement of the Domain model*
2. Define any additional objects or classes, or new internal methods or attributes

**STAFF <interface>**

name : STRING

userName : STRING

password : STRING

accountType : STRING

**Accessors**

getName() : STRING

getUserName(): STRING

getPassword(): STRING

getAccountType() : STRING

**Mutators**

setName(name : STRING)

setUserName(userName : STRING)

setPassword(password : STRING)

setAccountType(accountType : STRING)

**PATIENT <class>**

nameOfPatient : STRING

dateOfBirth : STRING

fileNumber : INTEGER

nationality : STRING

bloodType : STRING

patientHistory : STRING

**Accessors**

getNameOfPatient() : STRING

getDateOfBirth() : STRING

getFileNumber() : INTEGER

getNationality(): STRING

getBloodType(): STRING

getPatientHistory(): STRING

**Mutators**

setNameOfPatient(nameOfPatient : STRING)

setDateOfBirth(dateOfBirth : STRING)

setFileNumber(fileNumber : INTEGER)

setNationality(nationality : STRING)

setBloodType(bloodType : STRING)

setPatientHistory(patientHistory : STRING)

**APPOINTMENT <class>**

appointmentDate : STRING

appointmentTime : STRING

department : STRING

doctorName : STRING

patientName : STRING

**Accessors**

getAppointmentDate() : STRING

getAppointmentTime() : STRING

getDepartment() : STRING

getDoctorName() : STRING

getPatientName() : STRING

**Mutators**

setAppointmentDate(appointmentDate : STRING)

setAppointmentTime(appointmentTime : STRING)

setDepartment(department : STRING)

setDoctorName(doctorName : STRING)

setPatientName(patientName : STRING)

**RECEPTIONIST implements <STAFF>**

**addToPatient (pat : ArrayList <Patient>)**

Function will add a new patient to the database.

**addAppointment(a : Appointment) : Appointment**

Function will create an appointment for a patient to a requested doctor.

**CONTROL CLASS <class>**

**importDoctorFromFile(d : ArrayList <Doctor>) : ArrayList**

**importReceptionistFromFile(r : ArrayList <Receptionist>) : ArrayList**

**importPatientDatabase() : ArrayList**

**writeDoctorToDatabase(d : ArrayList <Doctor>)**

**writeReceptionistToDatabase(r : ArrayList <Receptionist>)**

**writePatientTODatabase(p : ArrayList<Patient>)**

**DOCTOR implements <STAFF> extends <APPOINTMENT>**

Gender : STRING

schedule : APPOINTMENT

specialty : STRING

**Accessors**

getGender() : STRING

getSchedule() : APPOINTMENT

getSpeciality() : STRING

**Mutators**

setGender(gender : STRING)

setAppointment(schedule : APPOINTMENT)

setSpeciality(speciality : STRING)

**checkAppointment(appointmentDate : STRING) : APPOINTMENT**

This will help the doctor view his appointment chart for the day.

**checkPatient(fileNumber : INTEGER) : PATIENT**

This will allow the doctor to view and modify patient

**updatePatientHistory(ArrayList<Patient> pat) : ArrayList**

**ADMINISTRATOR implements <STAFF>**

**addReceptionistToDatabase(r : ArrayList<Receptionist>)**

This will allow administrator to add a new receptionist user.

**addDoctorToDatabase(d : ArrayList<Doctor>)**

This will allow administrator to add a new Doctor user.

**modifyUser(d : ArrayList<Doctor>,r : ArrayList<Receptionist>,index : INTEGER) : ArrayList**

**deleteUser(d : ArrayList<Doctor>,r : ArrayList<Receptionist>,index : INTEGER) : ArrayList**

**importDoctorDatabase() : ArrayList**

Imports the doctors’ database from text file through control class into an Array List.

**importReceptionistDatabase() : ArrayList**

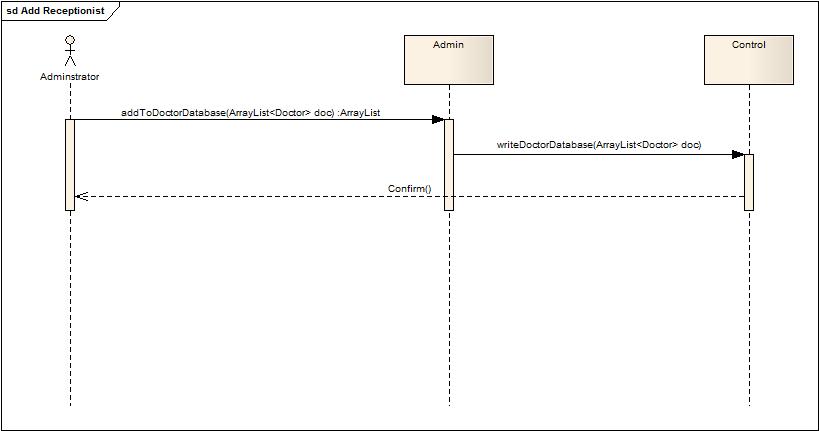
Imports the receptionists’ database from text file through control class into an ArrayList

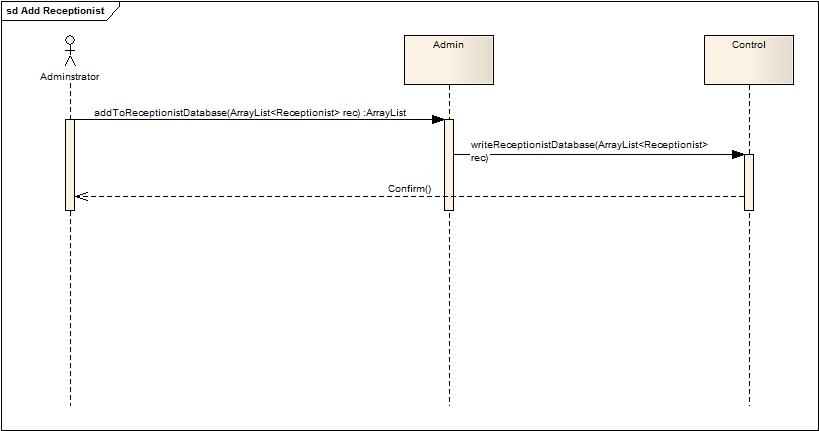
**searchForUser(d : ArrayList<Doctor> , r : ArrayList<Receptionist>) : INTEGER**

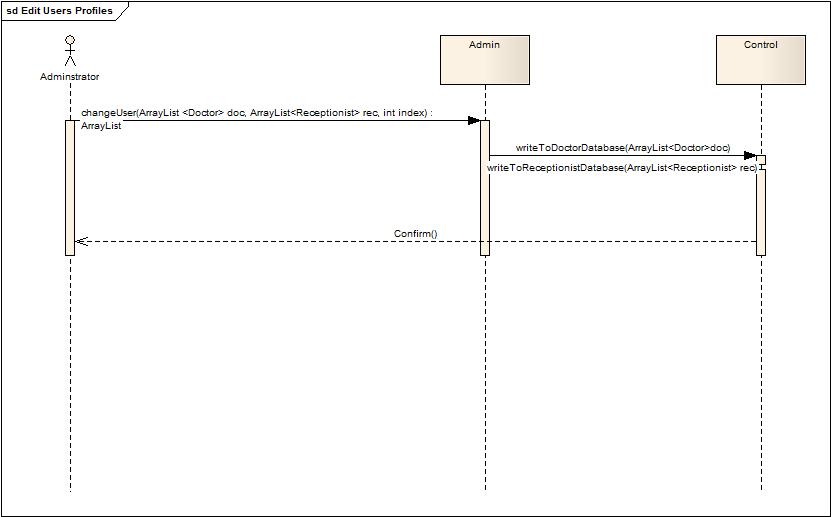
Searches for the specified user and returns the index of that user for modifications.

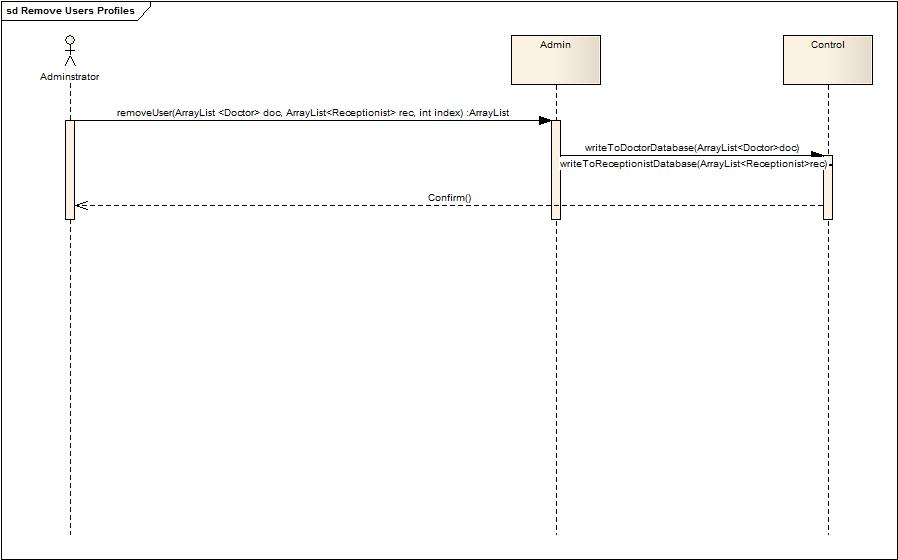
1. Develop the detailed design using Sequence diagram and/or Activity diagram.

**Administrator**

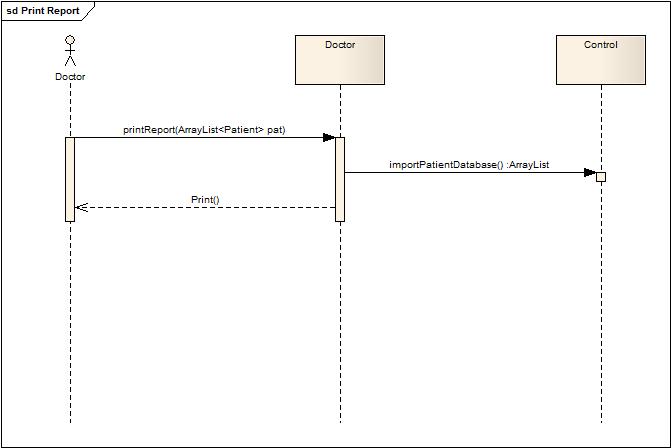


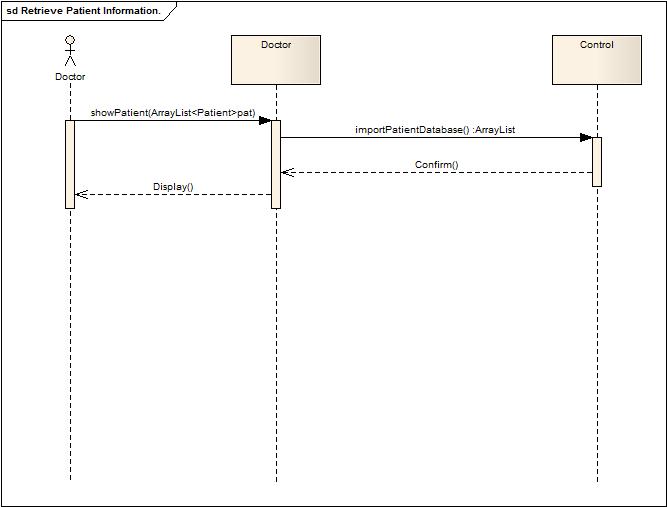


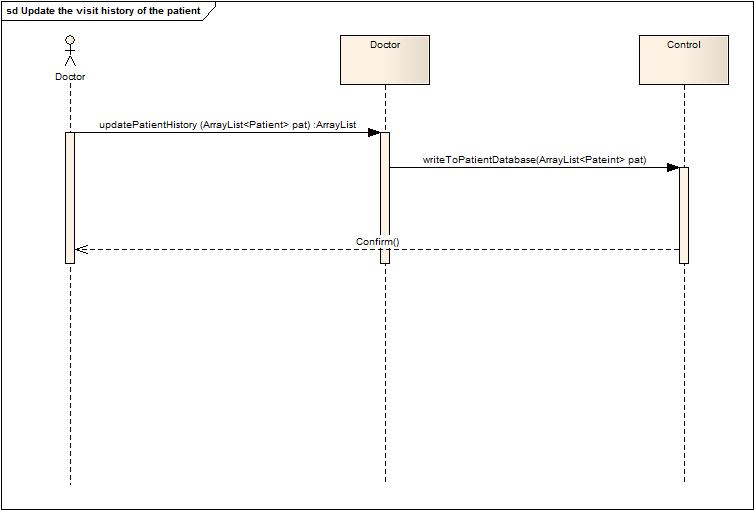




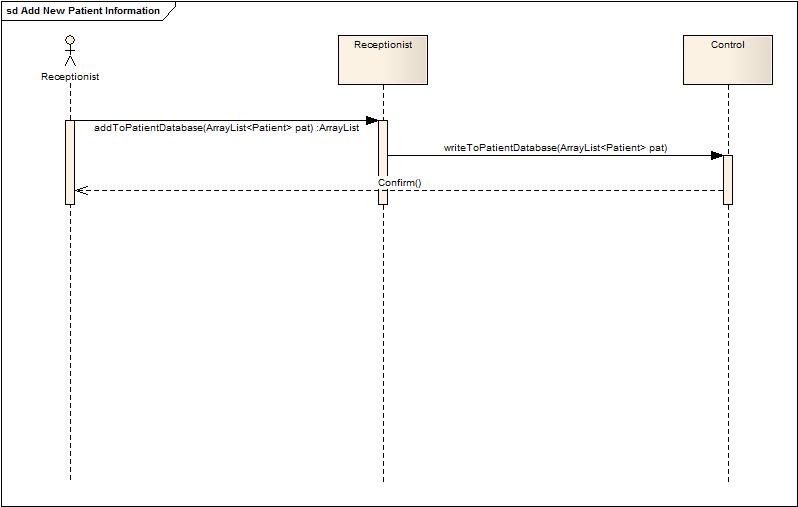
**Doctor**

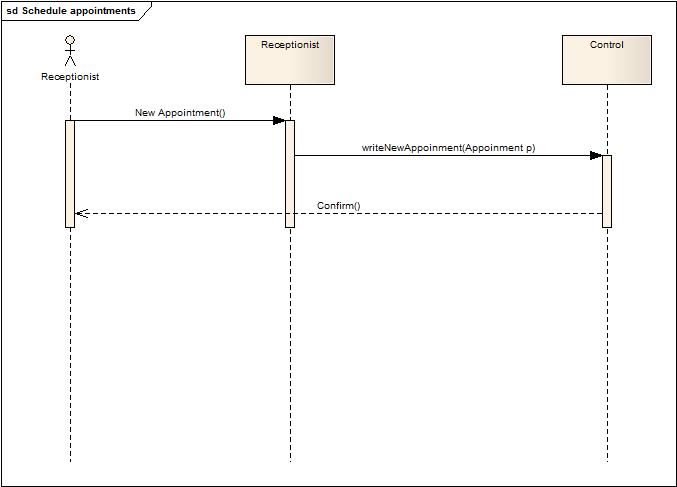
****

****

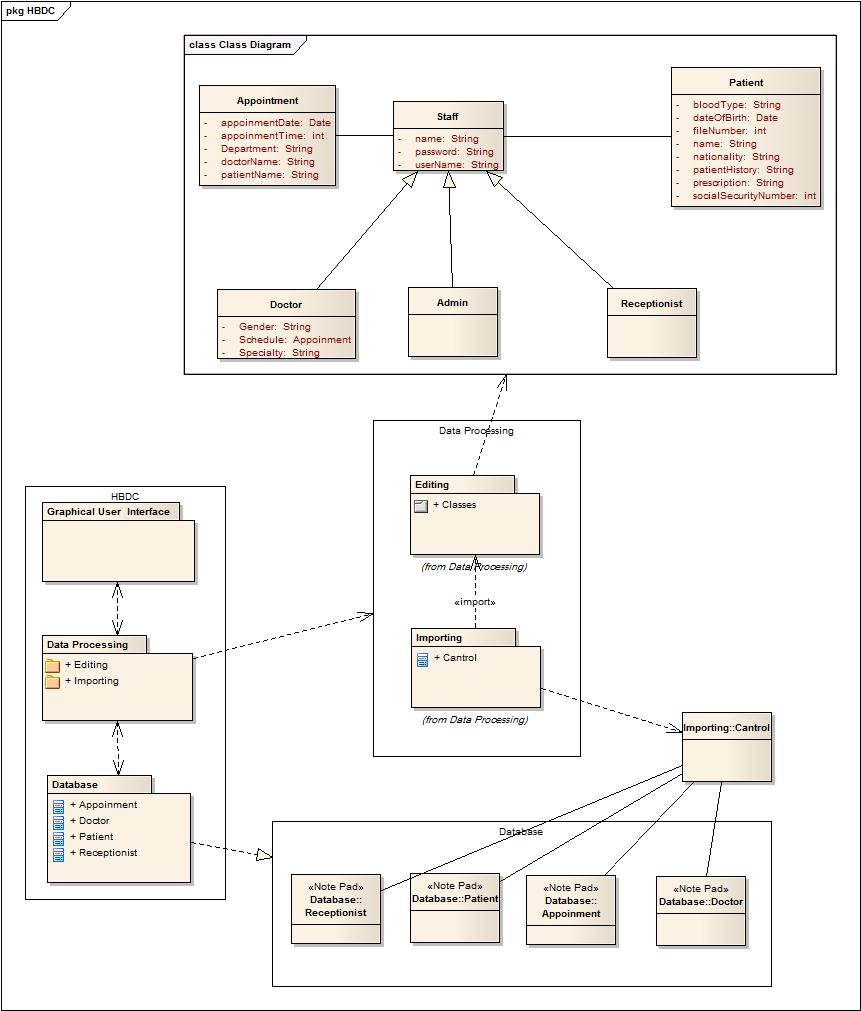
****

**Receptionist**

****

****

1. Integration with User Interface and Database.

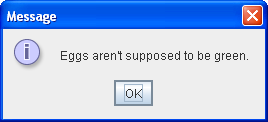


1. **Additional User Interface Design.**
   1. *Additional Input and Output screens.*

We don’t have any additional data input and output screens. We are planning to have the same number and type of screens as planned in phase 2.

* 1. *Additional data display screens.*

We will be using more update screens to show the user that they have successfully performed the work initiated. For this purpose we will use “JOptionPane” to use dialog boxes to update the user of the changes made or any other basic action performed on the graphic user interface.

****