EFFICIENT DOCTOR PATIENT PORTAL

PROJECT REPORT

OF

BACHELOR OF TECHNOLOGY

(3rd year project)

(Computer science & Engineering)

SUBMITTED BY

Name of Students	Roll No.
Abhishek Kumar	1401004
Kumar Gaurav	1401025

COURSE: CS331 (Software Engineering Lab)

UNDER THE GUIDENCE OF

Dr. FERDOUS A. BARBHUIYA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, GUWAHATI

APRIL, 2017

ACKNOWLEDGEMENT

The Project was jointly undertaken by Abhishek Kumar and Kumar Gaurav as their Software Engineering Course Project, under the able guidance and supervision of DR FERDOUS AHMED BARBHUIYA. Our primary thanks goes to him, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It's our privilege to acknowledge our deepest sense of gratitude to him for his inspiration which has helped us immensely. We are extremely grateful to him for his unstilted support and encouragement in the preparation of this project. We also take this opportunity to express a deep sense of gratitude to our Lab Instructors Sir NAZATUL HAQUE SULTAN and Ma'am RICHA SARMA for their cordial support, valuable suggestions and guidance.

INDEX

Chapter 1

Introduction

We here propose a Doctor patient handling, managing system that helps doctors in their work and also patients to book doctor appointments and view medical progress. The system allows doctors to manage their booking slots online. Patients are allowed to book empty slots online and those slots are reserved in their name. The system manages the appointment data for multiple doctors for various date and times. Each time a user visits a doctor his/her medical entry is stored in the database by doctor. Next time a user logs in he may view his/her entire medical history as and when needed. At the same time a doctor may view patients medical history even bore the patient visits him. This allows for an automated patient doctor handling system through an online interface. Our system also consists of organ donor module. This module allows for organ donation registration as well as organ search. The module is designed to help urgent organ requirements through easy/instant searches.

Figure1.1

1.1 Problem Statement

As the statistics shows that the percentage of patient increases with time and that why most of patient die due to lack of doctor facility. The manual work for booking a doctor took more human power as well as money power. Manual booking consumes almost ½-1 hrs (approx.) of every patient which is surely a headache.

1.2 Process Model

A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

The model is used to build the “Efficient Doctor Patient Portal” software is “The Prototyping Model”. The prototyping paradigm is: - “Water fall model”

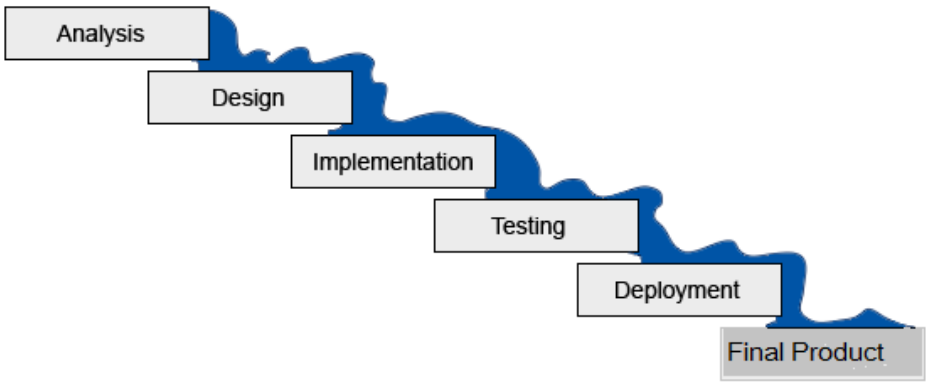

Figure1.2: Water *fall Model*

The water fall model is a software development model in which a systems development is viewed as flowing downwards through the phases of the system development process. The waterfall methodology is powerful, précised, and thorough. It has a number of phases that have to be implemented in a sequential manner.

The phases which come under the waterfall model are as follows:-

1.Requirement Analysis
2.Design
3.Implementation
4.Testing
5.Maintenance

Advantages:

1.Good for large projects
2.Waterfall suits a principled approach to design
3.Waterfall divides the project into manageable areas
4.Waterfall separates the logical and physical

Chapter 2

REQUIREMENT ANALYSIS

The basic function of requirement analysis is that it translates the ideas in the mind of the clients into a formal document. Thus the output of this phase is a set of precisely specified requirements which are complete and consistent. This document is called **Software Requirement Specification**.

In order to provide the user with a feeling of community, the following requirement should be taken care:

- Each user will have to create their own profile that they can log into each time they visit the site.
- If the user does not create or log in to an account they will only be able to browse questions on the site, they will not be able to use any of the sites other functionalities.
- In order to create an account the user must have a unique identity.
- Once they create an account the user will be able to Log in and out of the system, Upload profile.

User can view Doctor Profile as well which will help them to choose right Doctor wisely.

2.1 Use Case Diagram

UML Use Case Diagrams. Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

2.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

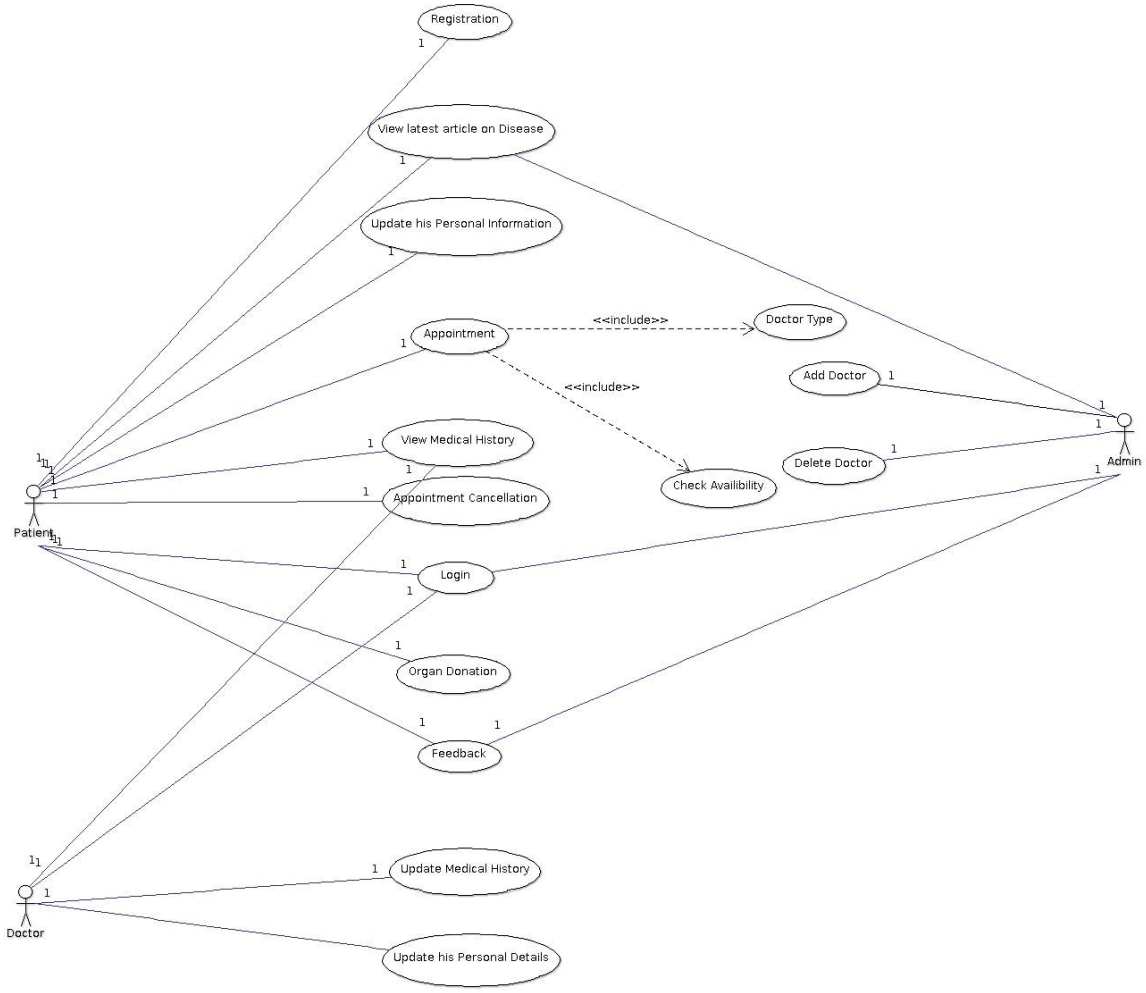

Fig 2.1: Use Case Diagram

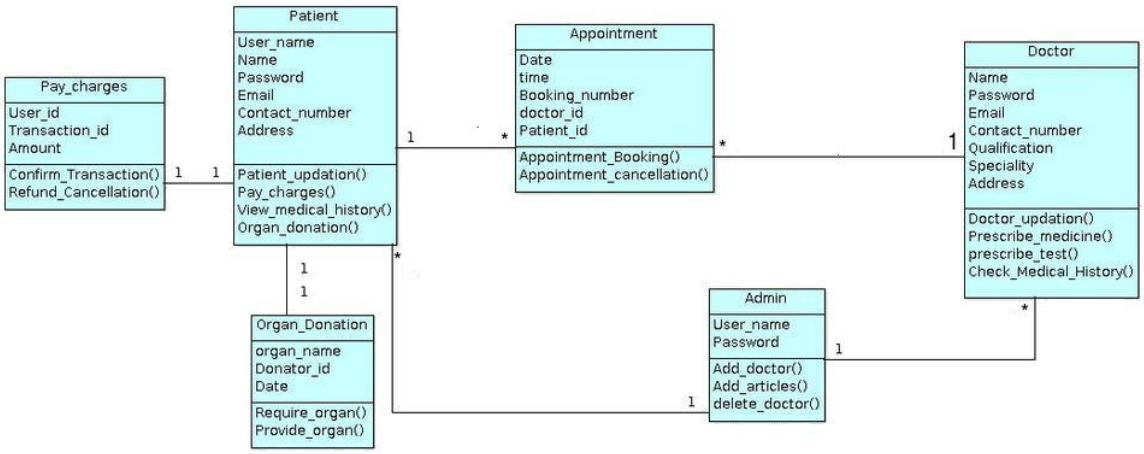

Fig 2.2: Class Diagram

2.3 State Diagram

A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

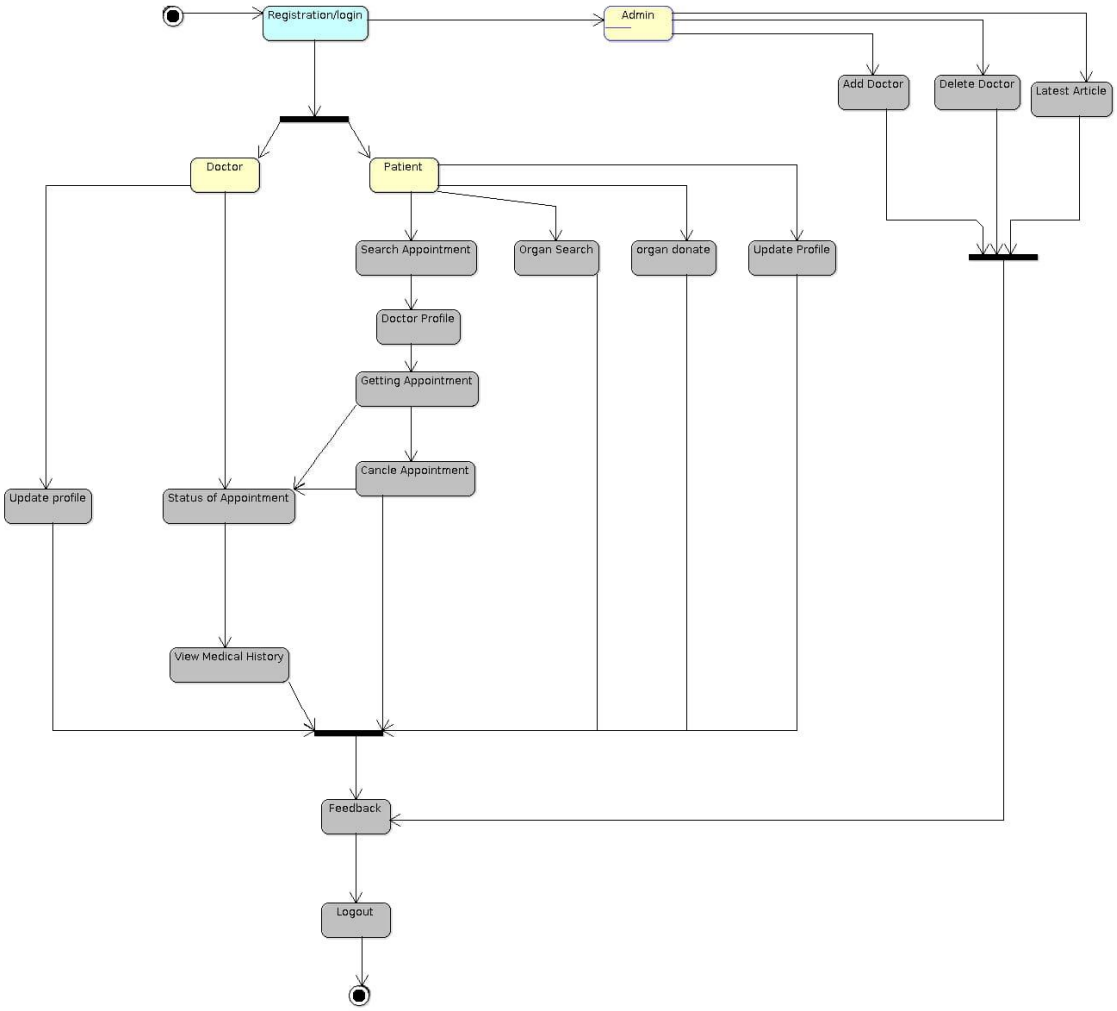

Fig 2.3: State Diagram

2.4 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another.

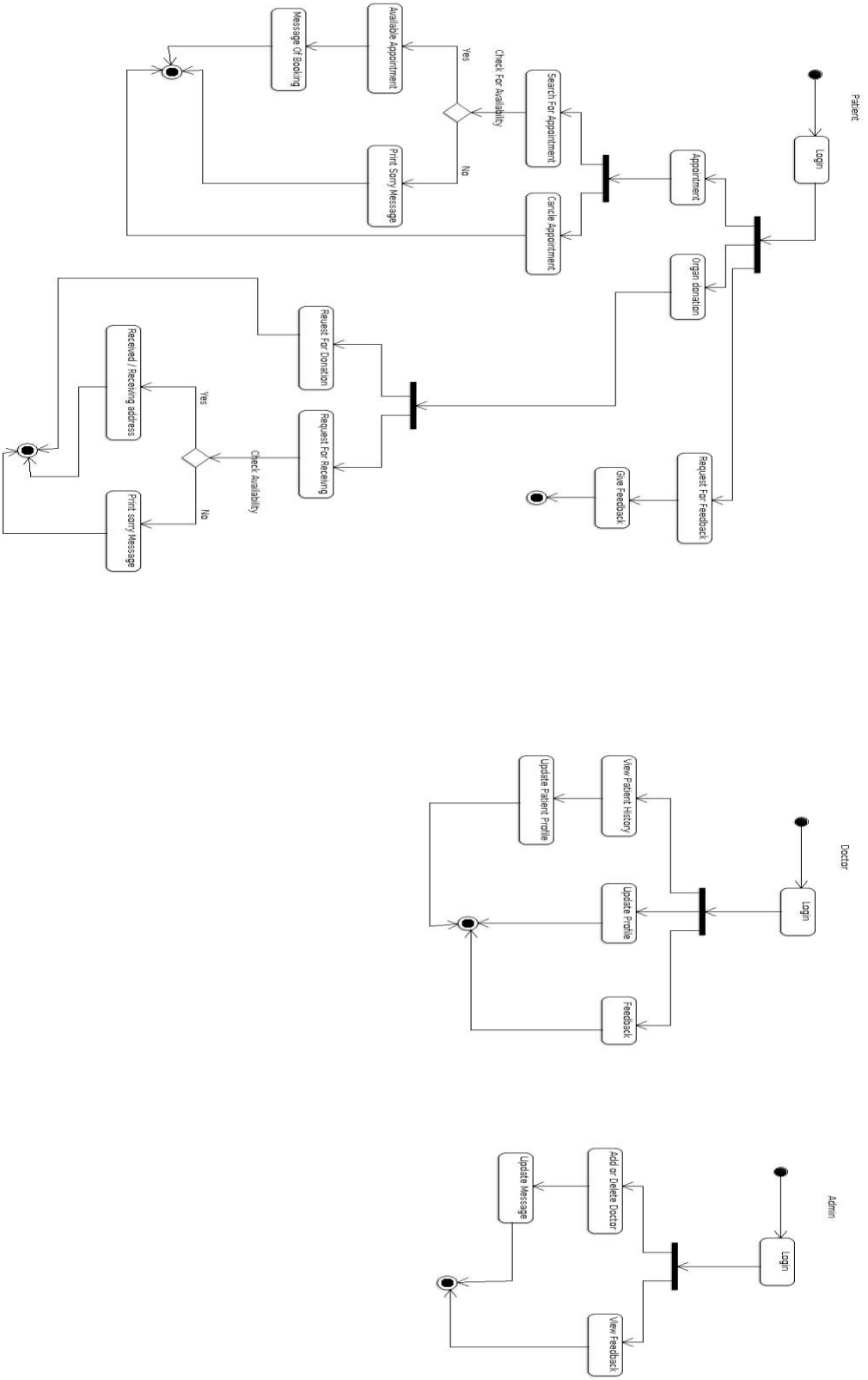

Fig 2.4: Activity Diagram

2.5 Swimlane Diagram

A swim lane (or swimlane diagram) is a visual element used in process flow diagrams, or flowcharts that visually distinguishes job sharing and responsibilities for sub-processes of a business process. Swim lanes may be arranged either horizontally or vertically.

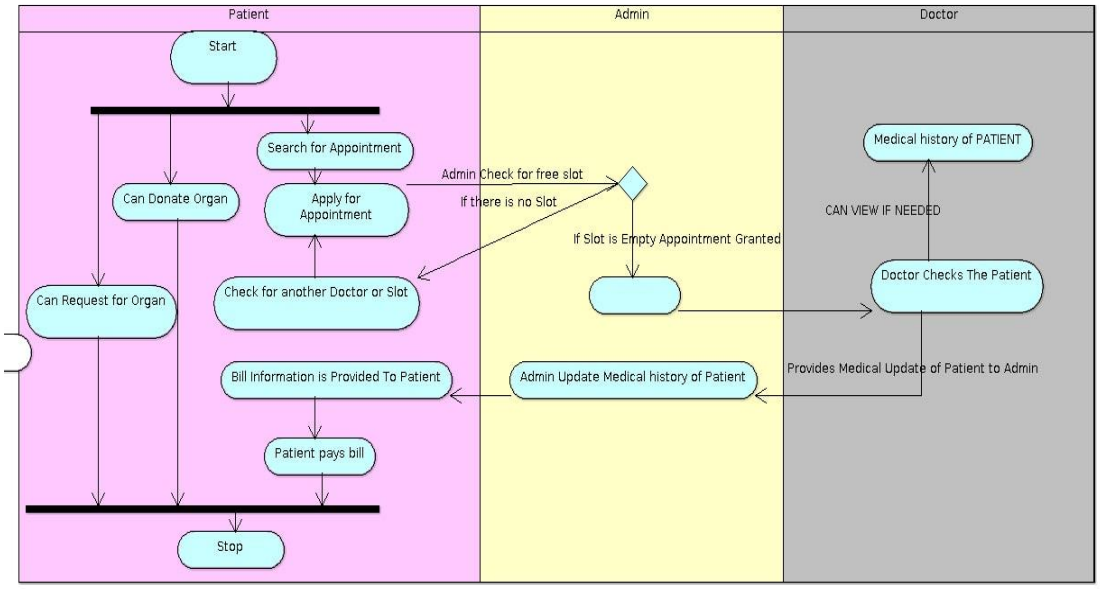

Fig 2.5: Swimlane Diagram

2.6 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

2.7 ER Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

2.8 Data Flow Diagram (DFD)

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

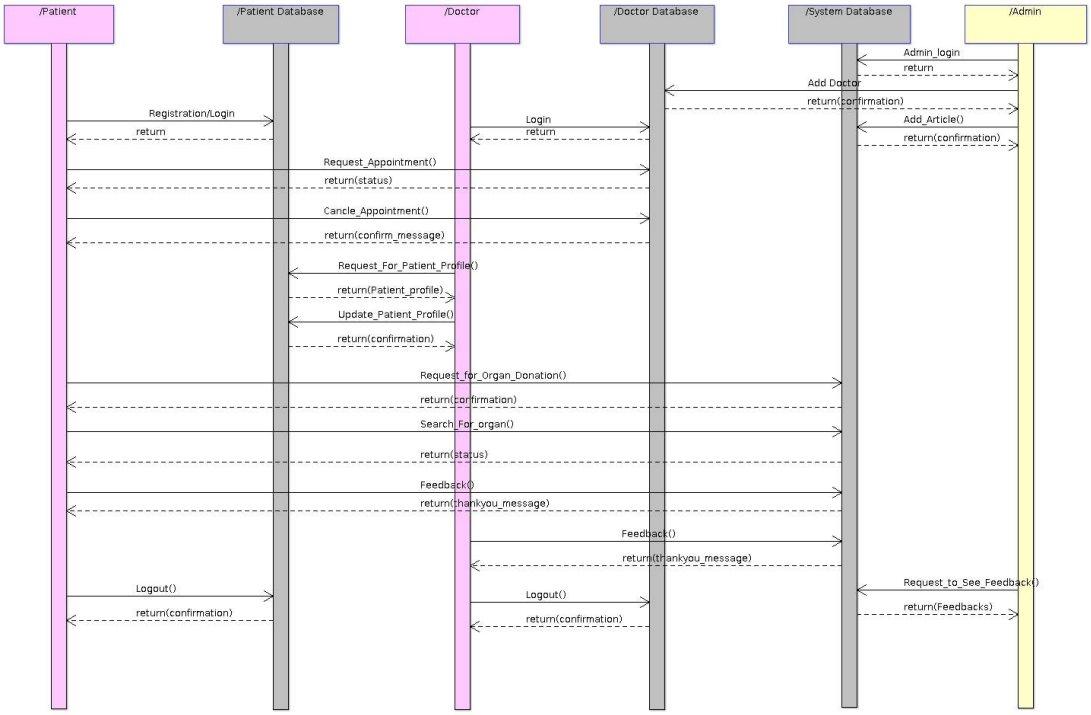

Fig 2.6: Sequence Diagram

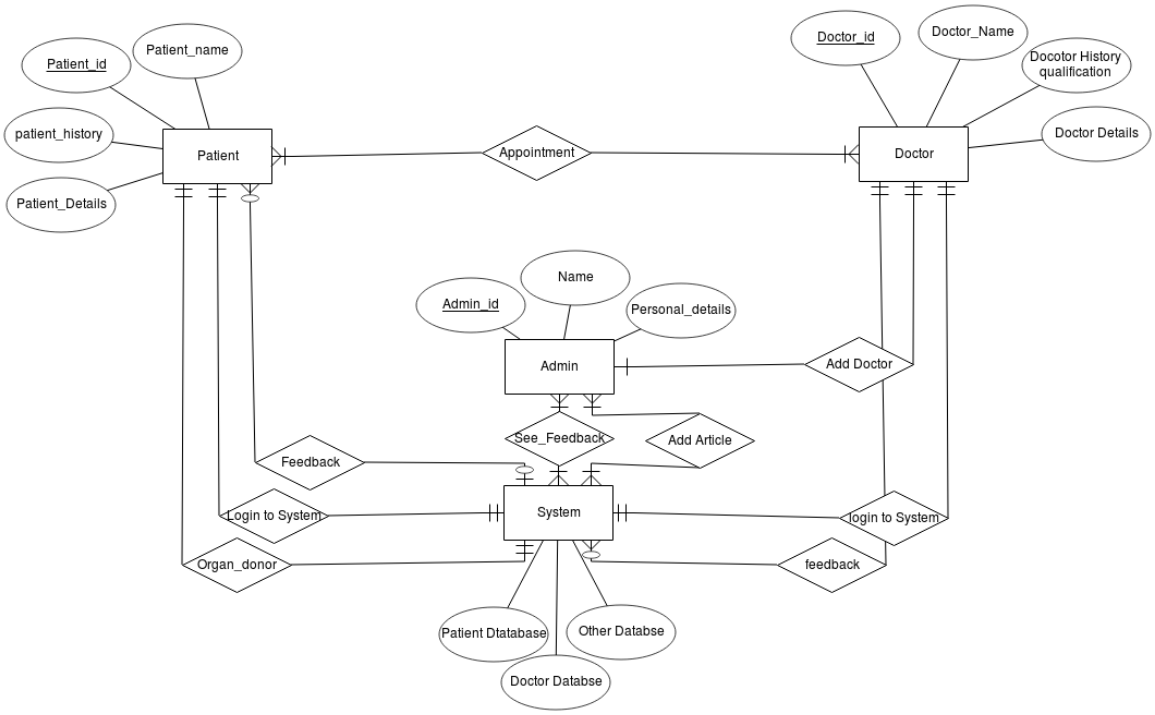

Fig 2.7: ER Diagram

DFDs may be partitioned into levels that represent increasing information flow and functional detail. Therefore, the DFD provides a mechanism for functional modeling as well as information flow modeling. DFDs are very useful in understanding a system and can be effectively used during analysis.

DFDs can be hierarchically organized, which helps in progressively partitioning and analyzing large systems. Such DFDs are called ***leveled DFDs***.

Levelled Data Flow Diagram (DFD)

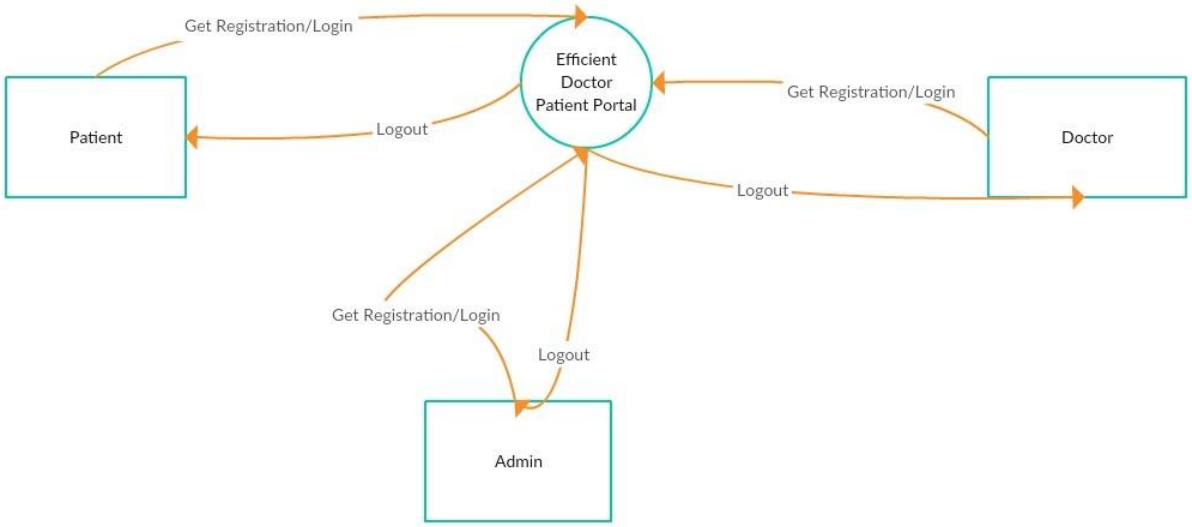

Fig 2.8.1: Level 0 DFD

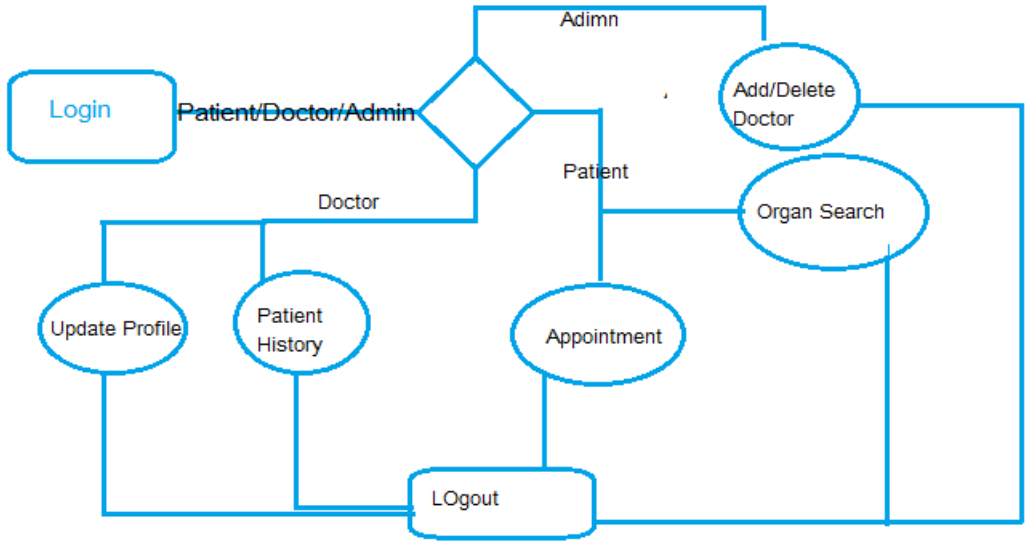

Fig 2.8.2 Level 1 DFD

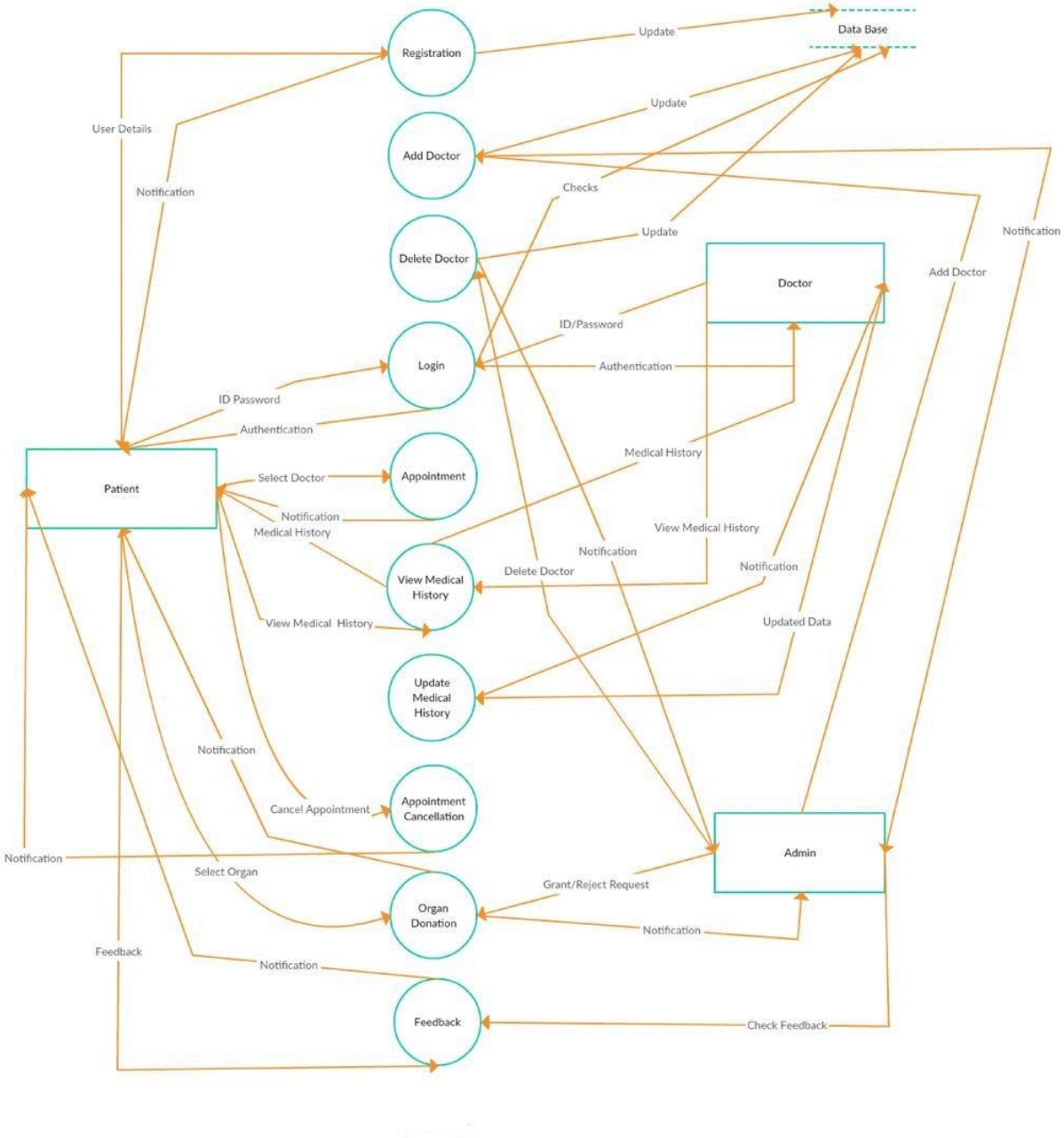

Fig 2.8.3 Level 3 DFD

Chapter 3

DESIGN ENGINEERING

The design of a system is essentially a blueprint or a plan for a solution for the system. A design methodology is a systematic approach to creating a design approach, a system is viewed as a transformation function, transforming the inputs to the desired outputs.

The design process for software systems often has two levels. At the first level the focus is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

3.1 Architectural Design

The architecture design process focuses on the decomposition of a system into different components and their interactions to satisfy functional and non-functional requirements.

For a function-oriented design, the design can be represented graphically by structure charts. The structure of a program is made up of the modules of that program together with the interconnections between modules. The structure chart of a program is a graphic representation of its structure. In a structure chart a module is represented by a box with the module name written in the box.

3.2 Component Level Design

The component-level design can be represented by using different approaches. One approach is to use a programming language while other is to use some intermediate design notation such as graphical (DFD, flowchart, or structure chart), tabular (decision table), or text-based (program design language) whichever is easier to be translated into source code.

The component-level design provides a way to determine whether the defined algorithms, data structures, and interfaces will work properly. Note that a component (also known as **module)** can be defined as a modular building block for the software. However, the meaning of component differs according to how software engineers use it. The modular design of the software should exhibit the following sets of properties.

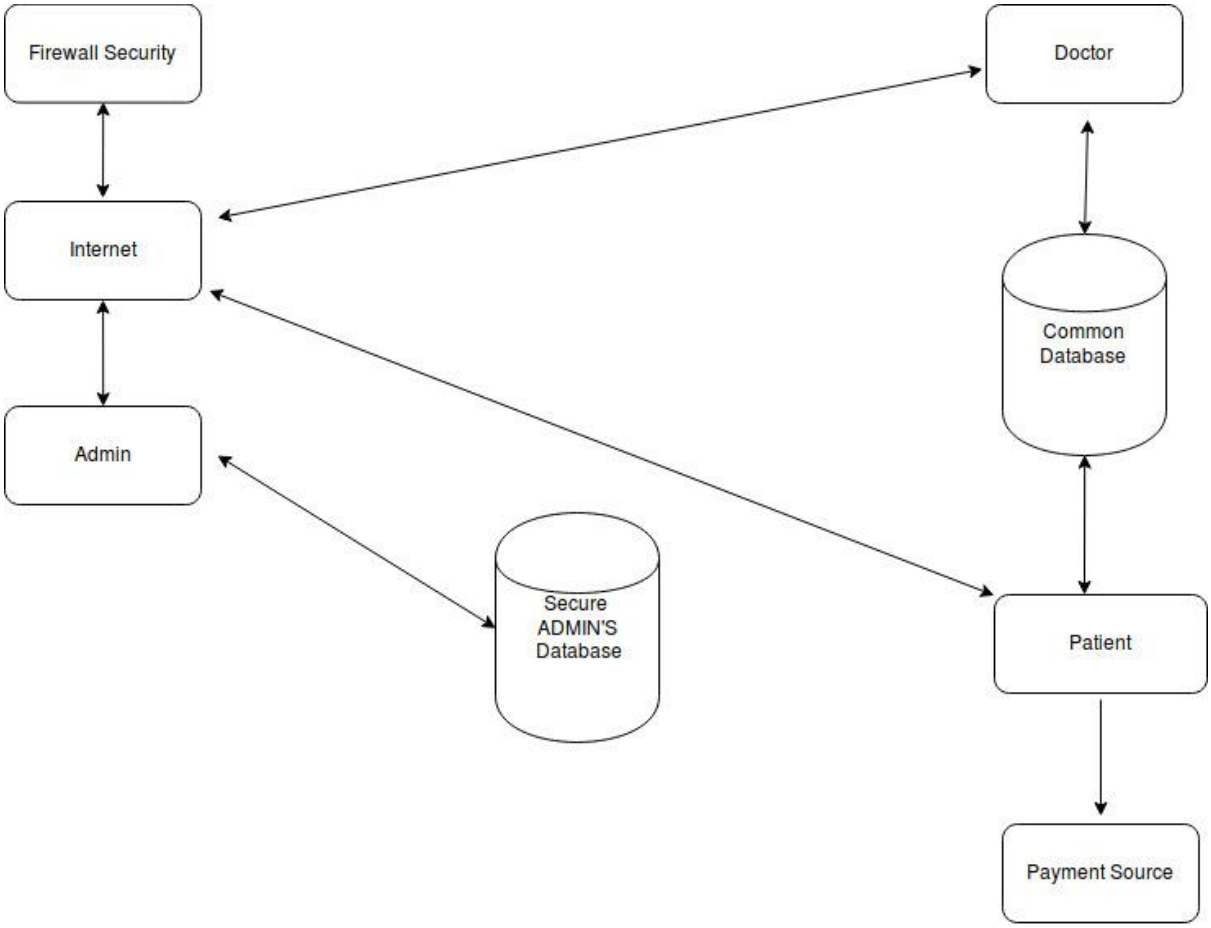

Fig 3.1 Architecture Design

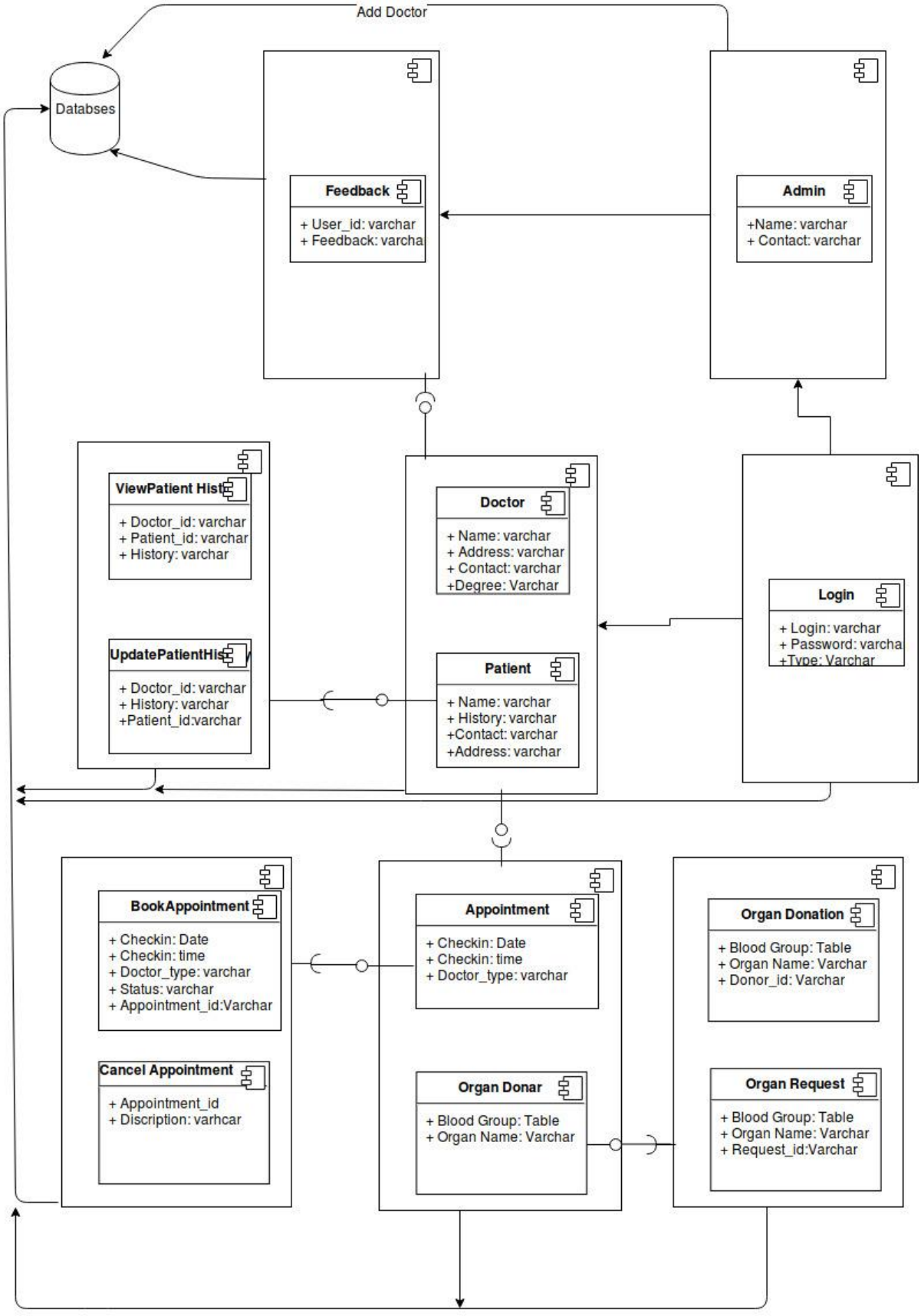

Fig 3.2 Component Level Design

Chapter 4

SOFTWARE TESTING

The main purpose of testing is to detect errors and error prone areas in a system. Testing must be thorough and well-planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.

4.1 Unit Testing

Unit testing is undertaken when a module has been created and successfully reviewed .In order to test a single module we need to provide a complete environment i.e. besides the module we would require

- The procedures belonging to other modules that the module under test calls
- Non local data structures that module accesses
- A procedure to call the functions of the module under test with appropriate Parameters.

Chapter 5

REFERENCES

• *Software Engineering- A practitioner's Approach by Roger S. Pressman: 6th edition McGraw Hill, 2005*

● *An Integrated Approach to Software Engineering by Pankaj Jalote: 3rd edition Springer, 2005*