<Hospital Application>

Analysis and Design Document

Student:Roman Bogdan -Ovidiu

**Group:30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Use Java/C# API to design and implement a client-server application for managing the consultations of doctors in a clinic. The application has three types of users: the clinic secretary, the doctors and an administrator.

# Functional Requirements

The clinic secretary can perform the following operations:

Add/update patients (patient information: name, identity card number, personal numerical code, date of birth, address).

CRUD on patients’ consultations (e.g. scheduling a consultation, assigning a doctor to a patient based on the doctor’s availability).

The doctors can perform the following operations:

Add/view the details of a patient’s (past) consultation.

The administrator can perform the following operations:

CRUD on user accounts.

# Non-functional Requirements

1. Performance – 1 sec response time for every operation from database like add remove view list . Aprox 13 seconds to run the application on server.
2. Availability- 24 h/day , 353 days/year, 98.9%
3. Reliability-99.9% correct given outputs from the system.
4. Security- Spring security with log in and admin user
5. Manageability- depends from case to case, depends on how the team works and how big the bug is.

2. Use-Case Model

*Use case: <Delete patient>*

*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <Admin>*

*Main success scenario: <1. Open the application*

*2. Log in into your user account as doctor or secretary*

*3. Logic successfully*

*4. Go to Patient Page*

*5. See the list of patient, choose one , the one you want to delete*

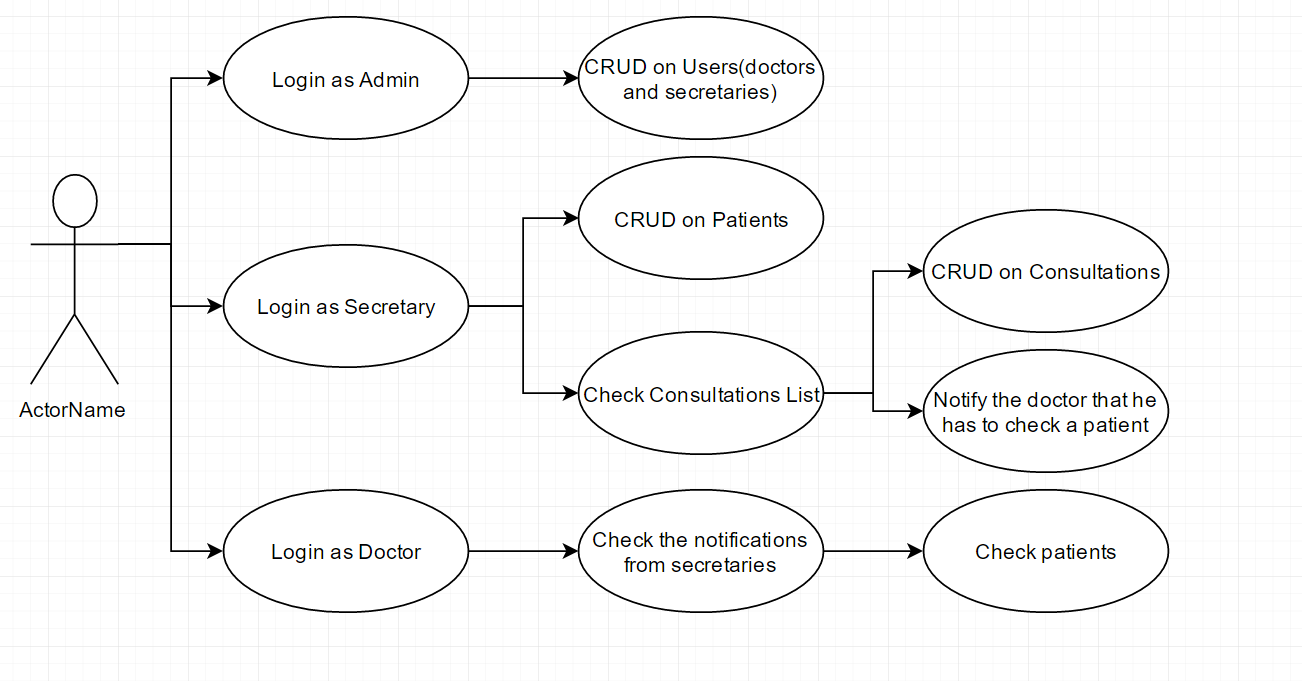
*6. Press delete button*

*7. Client is not in the table anymore: Succesfully deleted!*

*8. Log Out from the application>*

*Extensions: <2.Step 2 failed to log in, try again*

*3. The book you wanted to remove is not in the table, check again.*



3. System Architectural Design

**3.1 Architectural Pattern Description**

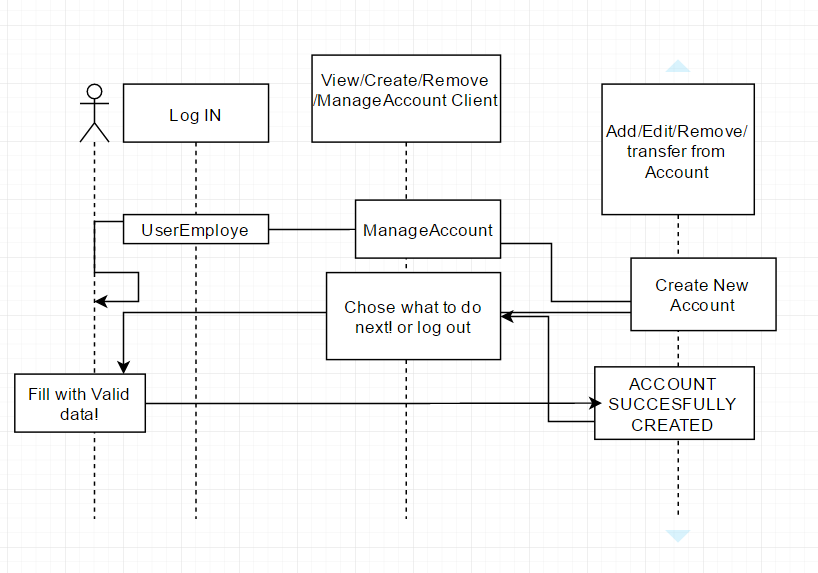
The **client–server model** is a [distributed application](https://en.wikipedia.org/wiki/Distributed_application) structure that partitions tasks or workloads between the providers of a resource or service, called [servers](https://en.wikipedia.org/wiki/Server_(computing)), and service requesters, called [clients](https://en.wikipedia.org/wiki/Client_(computing)).Often clients and servers communicate over a [computer network](https://en.wikipedia.org/wiki/Computer_network) on separate hardware, but both client and server may reside in the same system. A server [host](https://en.wikipedia.org/wiki/Host_(network)) runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are [Email](https://en.wikipedia.org/wiki/Email), [network printing](https://en.wikipedia.org/wiki/Network_printing), and the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. Observer pattern falls under behavioral pattern category.

**3.2 Diagrams**

*Systems conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagram.*

4. UML Sequence Diagrams

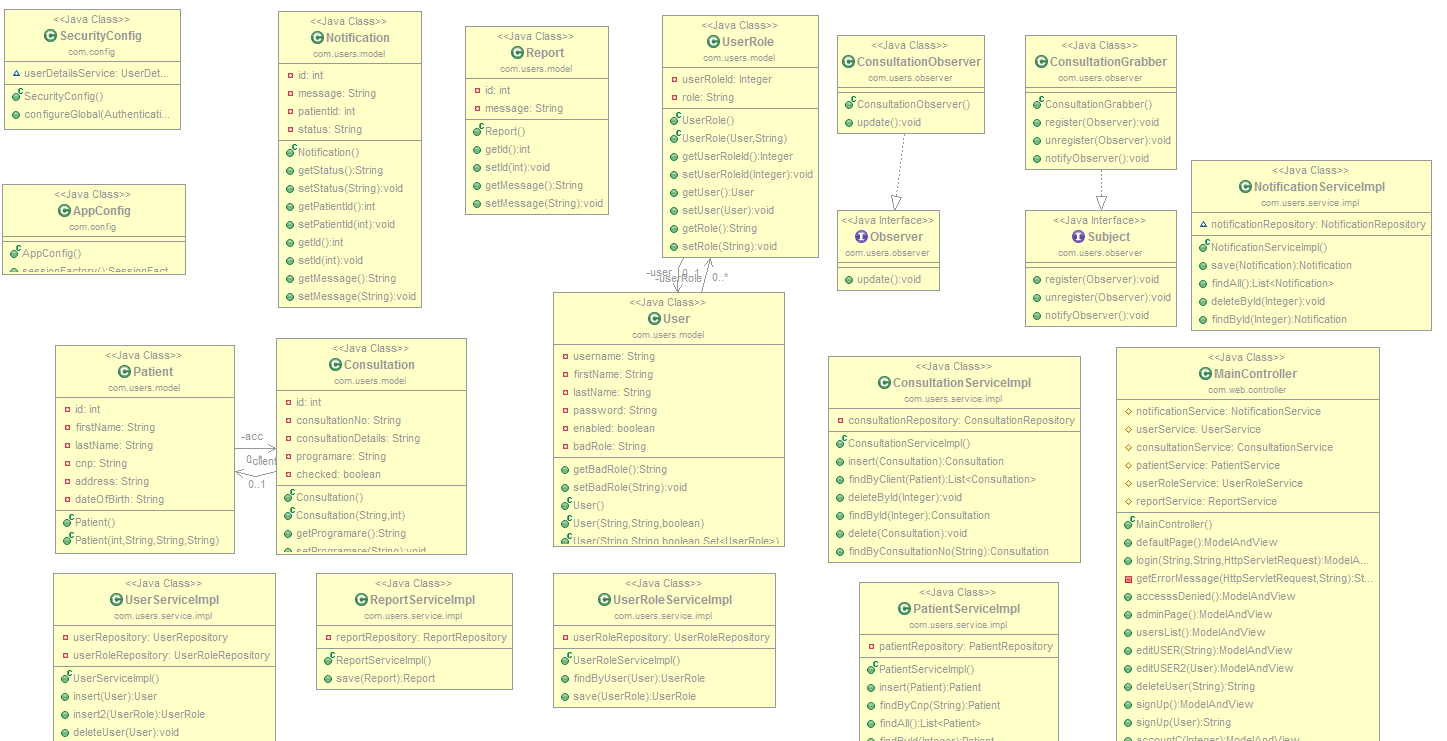


5. Class Design

**5.1 Design Patterns Description**

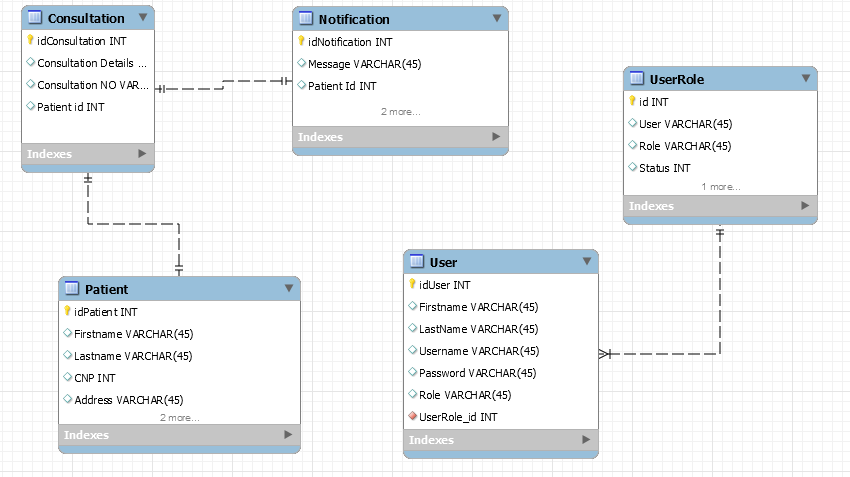
Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. Observer pattern falls under behavioral pattern category.

**5.2 UML Class Diagram**



6. Data Model

*The data models are : Consultation, Patient , User, UserRole and Notification tables.*

**

7. System Testing

*The system was tested manually.*

8. Bibliography

<http://www.tutorialspoint.com/design_pattern/observer_pattern.htm>

<https://en.wikipedia.org/wiki/Client%E2%80%93server_model>

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

<https://msdn.microsoft.com/en-us/library/54xbah2z(v=vs.110).aspx>