Hospital Management

Analysis and Design Document

Student:Emiliana Popa

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 05/04/2017 | <1.0> |  | Emiliana Popa |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

I will design and implement an application for the front desk employees of a hospital. The application will have three types of users (a regular user represented by the front desk employee, an administrator user and a doctor) which have to provide a username and a password in order to use the application, except the patient who will just have to fill in a form.

The administrator user can perform the following operations:

* Add/update/view patient information (name, identity card number, personal numerical code, address, etc.).
* Create/update/delete/view doctor information
* Make appointments.
* Issue bills.

The patient user can perform the following operations:

* Request diagnosis
* Request appointment

The doctor user can perform the following operations:

* Give a diagnosis

# Elaboration – Iteration 1.1

# Domain Model

# *D:\Facultate\An 3\An 3\Semestrul 2\PS\Proiect\domainmodel.png*

# Architectural Design

## Conceptual Architecture

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

The MVC components are:



### Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

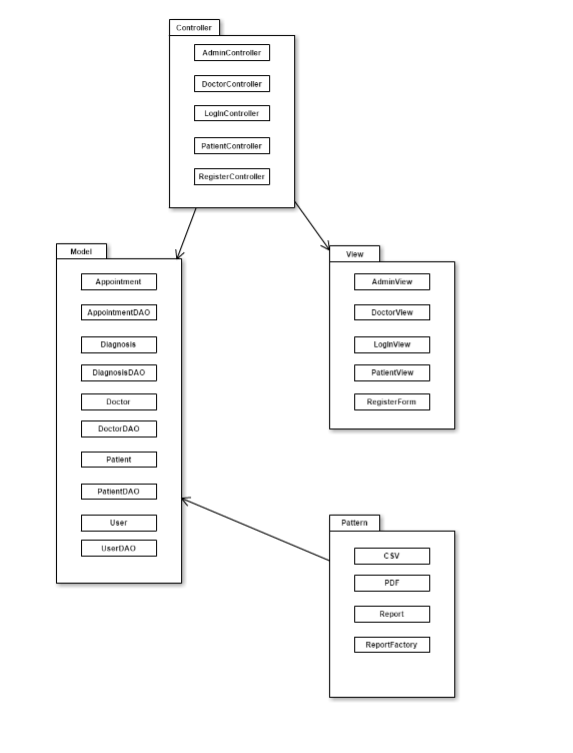
### View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

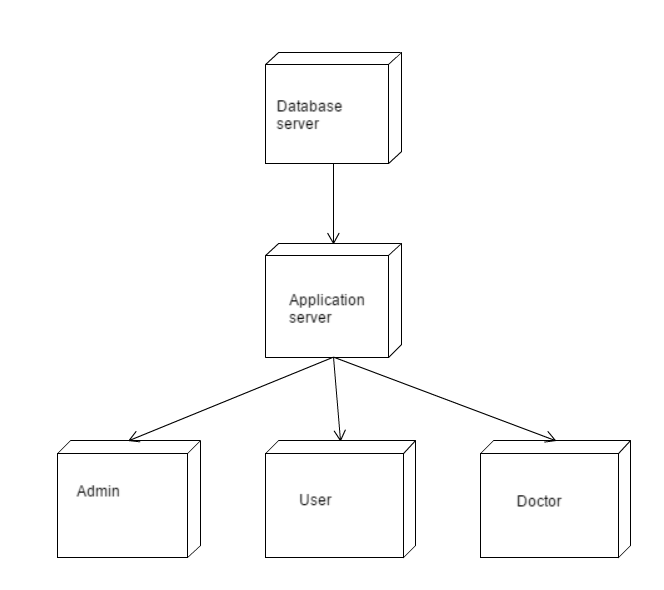
### Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

## Package Design



## Component and Deployment Diagrams

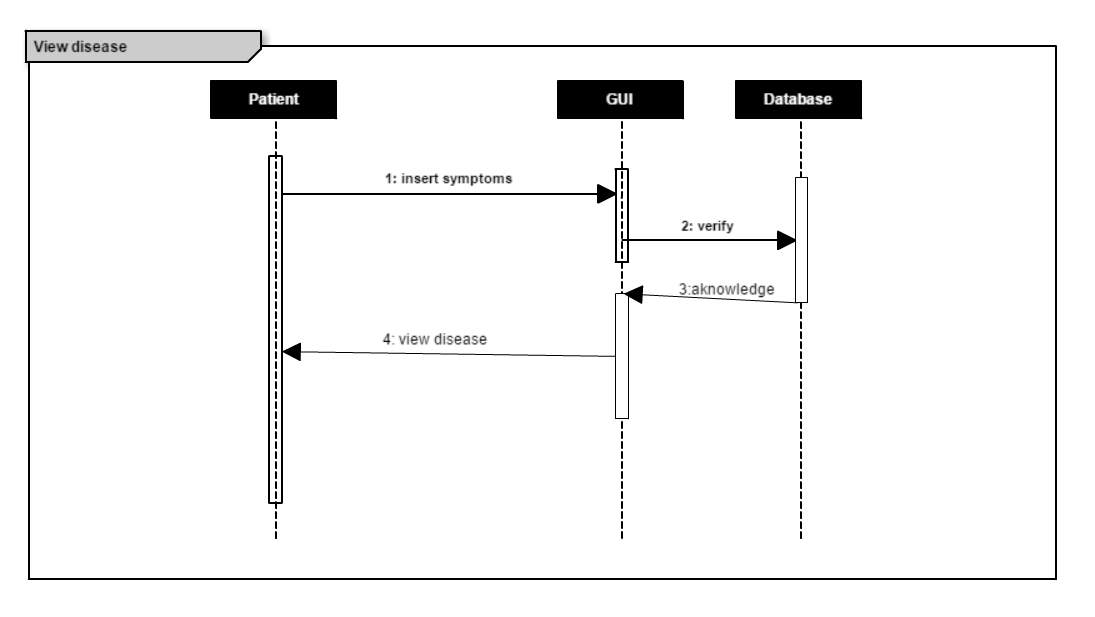


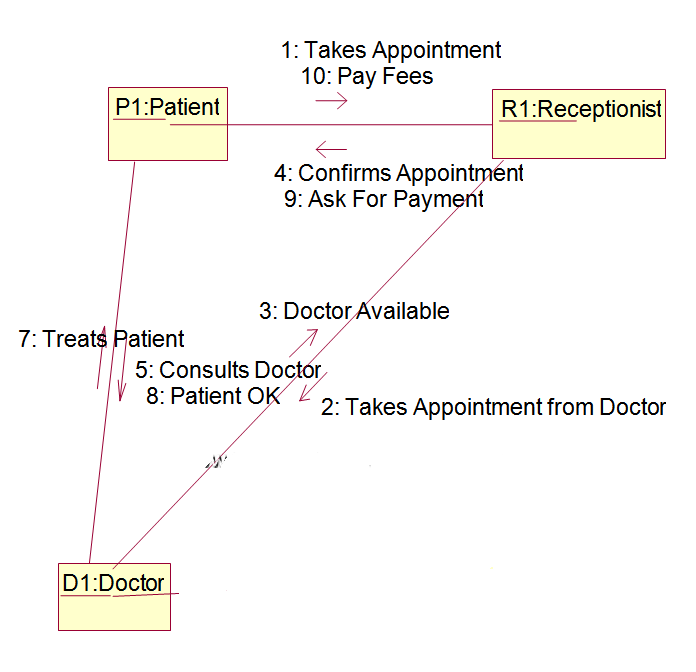
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

**

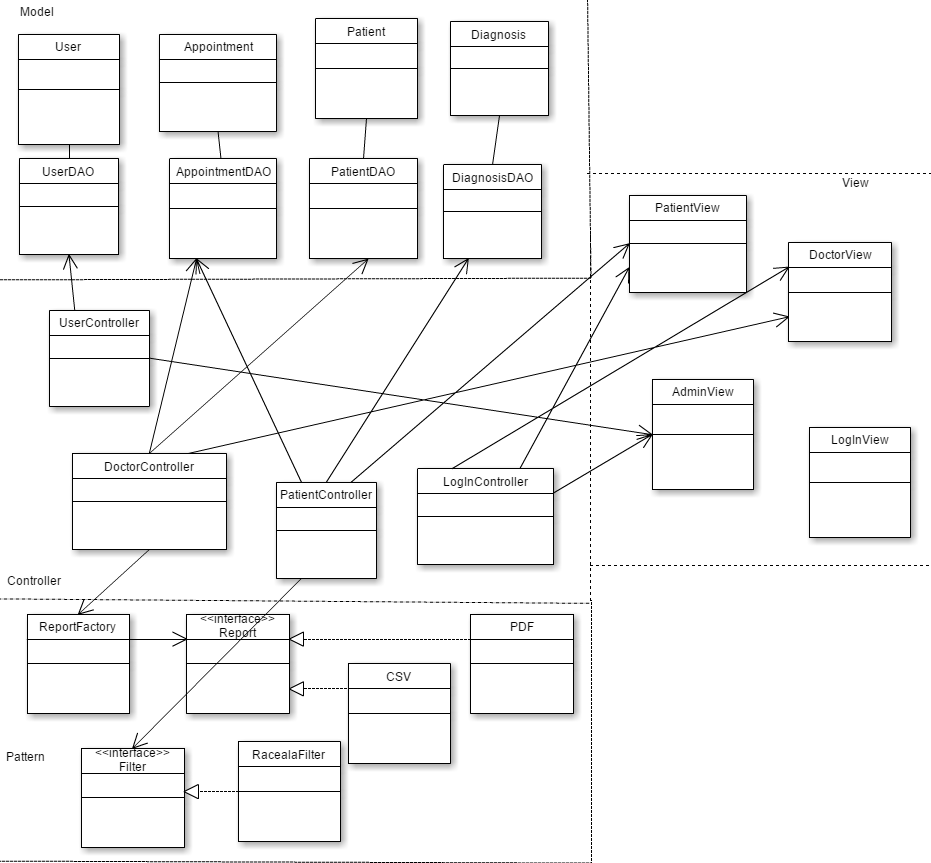
**

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

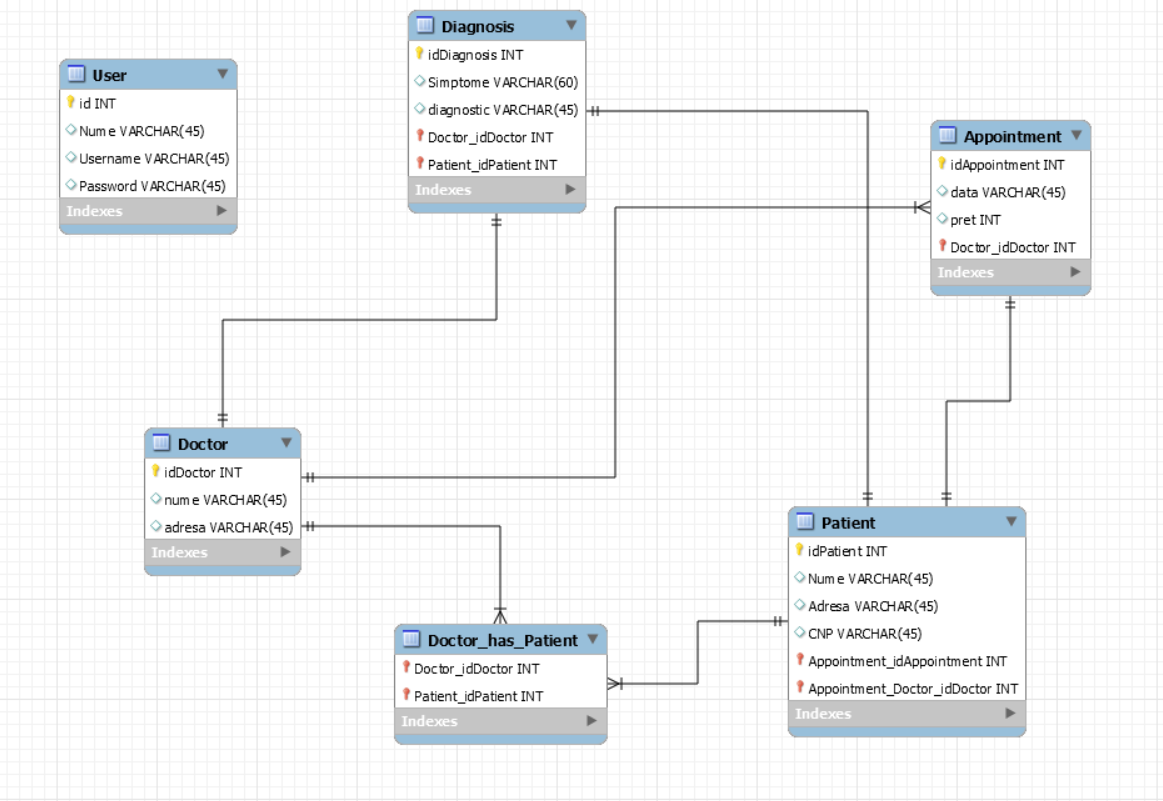
I will use Factory Pattern for exporting bills in PDF and CSV format.

Filter pattern or Criteria pattern is a design pattern that enables developers to filter a set of objects using different criteria and chaining them in a decoupled way through logical operations. This type of design pattern comes under structural pattern as this pattern combines multiple criteria to obtain single criteria. I used this pattern to take from my database all the diseases for a simptom

**

# Data Model

*[Create the data model for the system.]*

**

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

For testing I implemented a JUnit test named UserTest. This class will test all the methods in the UserDAO class. The first one is include method that add a user to the database using a User object. For this I gave some values for a random user and a used the assertEquals method to test if the doctor for example was add to the database. The second method is update and is almost the same as the previous one except that at the update method I will need a parameter integer that represent the id. The last test method is design to see if the delete method works.

After all the testing I encounter the results that I needed and so I realized that the user CRUD is correct.

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

In the package diagram I changed some parts of it like adding a new package for the exporting files called Pattern. The other ones were good implemented so I didn’t need to change anything. The sequence diagram is the same that was at the first time that I write it, and the other ones except the class diagram suffered no changes.

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

I didn’t make some important changes from the initial diagrams just by adding some classes. At the class diagram I make some new classes like Factory for using the factory design pattern ,or CSV and PDF classes that are used for exporting the database in the format that is wanted.

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

In this part I used the try and catch block for catching the exceptions. In every class that I used to put, update, get items from the database I added the instructions that I mentioned earlier.

# Future improvements

*[Present future improvements for the system]*

An important improvement for this project will be to transform it to web service so that all the users that have access to the internet can use the application. Another update for the project will be to have the possibility to print the bills in another form like JSON.

Using more design patterns is every time a good improvement for a project,and this is not the exception for this rule.

# Bibliography

1. <https://www.tutorialspoint.com>
2. <http://www.oodesign.com/factory-pattern.html>