Student: Popa Diana-Maria

**Group:30234**

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

‘Hospital Management System’ is an application designed for the management of an hospital. The application has three types of users (secretary, administrator, and doctor) . which have to provide a username and a password in order to use the application.

The clinic secretary can perform the following operations:

* Add/update patients (patient information: name, identity card number, personal numerical code, date of birth, address).
* Create/update/delete on patients’ consultations (e.g. scheduling a consultation, assigning a doctor to a patient based on the doctor’s availability).
* View all the schedules for all the doctor and patients
* Notify the doctor when a patient is arriving for a schedule.

The doctors can perform the following operations:

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

The administrator can perform the following operations:

* Create/update/delete an user accounts.
* View all the users

# Functional Requirements

Perform CRUD operation on patients, doctors and schedules.

# Non-functional Requirements

There are three applications, on for every type of user and including only the functionality for that type of user. The architecture is client-server and the data is stored in a database.

The client applications are Windows Forms applications. I have used a .Net Web API to expose the server functionality to the client applications.

2. Use-Case Model

Use case: add schedule for a doctor

Level: user-goal level

Primary actor: secretary user

Main success scenario:

Check the doctor’s availability

If the doctor is available in that day, add a new schedule

Extensions: if the doctor has too many schedules for that day, get an error message

Use case: get patient history

Level: user-goal level

Primary actor: doctor user

Main success scenario:

Get the patient id

Get all the schedules for the patient ID

3. System Architectural Design

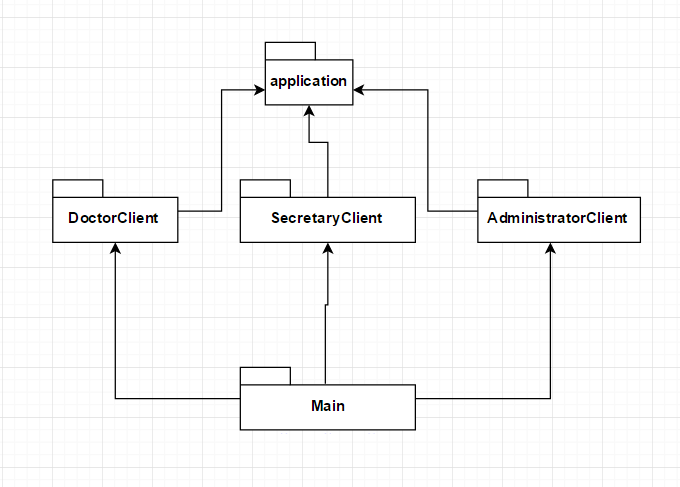
**3.1 Architectural Pattern Description**

This application implements a client-server functionality. The system is segregated into four application: one server – the database with application logic represented as stored procedures, and three clients: doctor, administrator, secretary which make requests to the server.

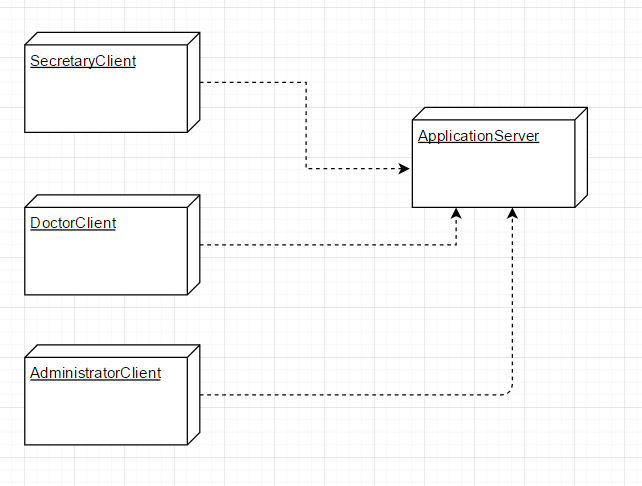
Using a .Net Web API to expose the server functionality to the client applications, there are basically two programs running at the same time: the code that lives on the server and responds to HTTP requests and the code that lives in the browser and responds to user input.

**3.2 Diagrams**

*Package diagram*

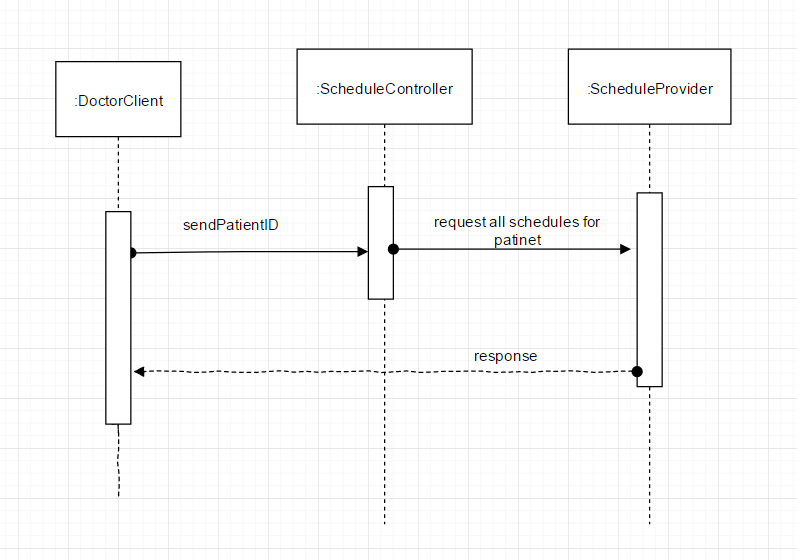
**

*Deployment diagram*

**

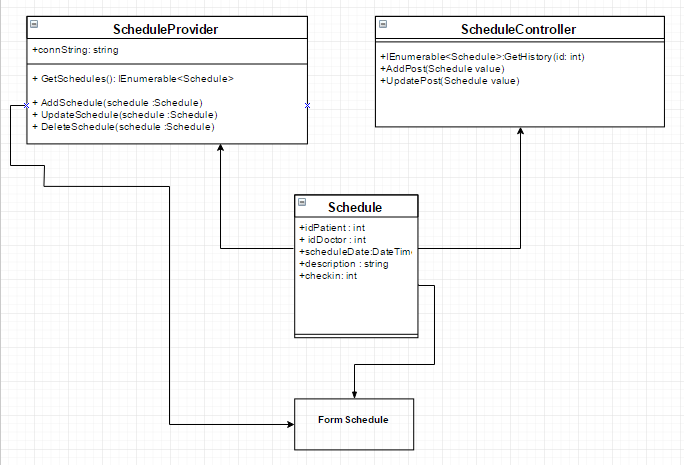
4. UML Sequence Diagrams

The sequence diagram for the use case: get all schedules for a patient

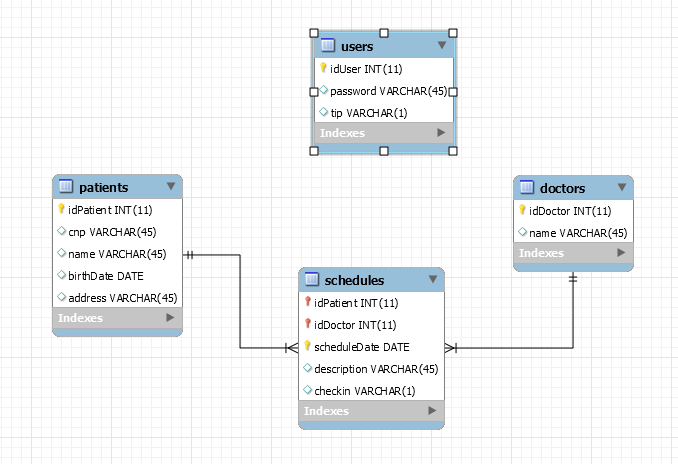


5. Class Design

**5.1 UML Class Diagram**



6. Data Model

**

7. Bibliography

[**http://www.codeproject.com/Articles/769671/Web-API-without-MVC**](http://www.codeproject.com/Articles/769671/Web-API-without-MVC)

<http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>