Care & Cure Hospital

Object Design

Version 1.0

29.12.2017

Begüm Mina BİLGİN

Ekin Naz EREN

Gamze SARAÇ

Süleyman Murat YİĞİT

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc436772639)

[1.1. Object Design Trade-offs 1](#_Toc436772640)

[1.2. Interface Documentation Guidelines 1](#_Toc436772641)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc436772642)

[1.4. References 1](#_Toc436772643)

[2. Packages 1](#_Toc436772644)

[3. Class Interfaces 1](#_Toc436772645)

OBJECT DESIGN DOCUMENT

# Introduction

## Object Design Trade-offs

In the ODD, we will define the trade-offs made by our team, interface documentation guidelines we have followed, packages, file organizations of the code, classes and interfaces used in the code. We had to do some certain trade-offs in our project, since a project cannot be “Fast, Good and Cheap” all together. In our project, we have done the trade-off as being “Good” and “Cheap”. Being cheap is a must, since this is a student project. Being “Good” is as all the functionalities described in RAD must work properly. Other trade-offs we have made is

* Buy vs Build : Our software project is built from scratch, and we are coding and developing the code together with our team.
* Memory space vs response time : Response time focused to be not take a very long time, and more memory space can be used for faster response time.

## Interface Documentation Guidelines

There are a list of rules we have applied in our developer team so we can understand and develop the code written (individually) together easier, and make it more readable.

These rules are:

* Buttons are labeled as “btn…\_Click” for onClick events
* Classes are named with singular nouns.
* Methods are named with verb phrases, fields, and parameters with noun phrases.
* Start and End of certain functions are shown in comment lines.
* Error or exceptional statuses are handled accordingly by try/catch or return respectfully.

## Definitions, Acronyms, and Abbreviations

ODD – Object Design Documentation

RAD – Requirement Analysis Documentation

SDD - System Design Document

VS – Visual Studio

UI – User Interface

SQL – Structured Query Language

## References

RAD and SDD are taken as a reference.

Since there is no pre-existing system, we don't take any references from an existing system.

# Packages

Describes the decomposition of subsystems into packages and the **file organization of the code.** This includes an overview of each package, its dependencies with other packages, and its expected usage.

Being a VS 2017 web forms project, our project classes include of an html-css-javascript(also bootstrap as framework) side code ending with aspx, which is the “view” and “UI” side, and a C# code ending with aspx.cs , which is the “controller” side of this code. The “model” side is the database, which is build-in in the project, in the “App\_Data” folder, as “CareandCure.mdf” SQL server database. In the file organization, each number shows a folder inside the project. Some folders such as bootstrap and jquery built-in script and css are skipped.

The file organization is as follows :

**CareandCureFull.sln**

1. **App\_data**

**1.a)CareandCure.mdf (database)**

1. **Images**

Contains the images to be used in the web project.

1. **UserFunctions**

Includes register(patient), login(all users), and to be included are loginviews of specific users.

Other classes include aspx and aspx.cs web pages which are Site.Master, Default page(Home page), About page and Contact page.

# Class Interfaces

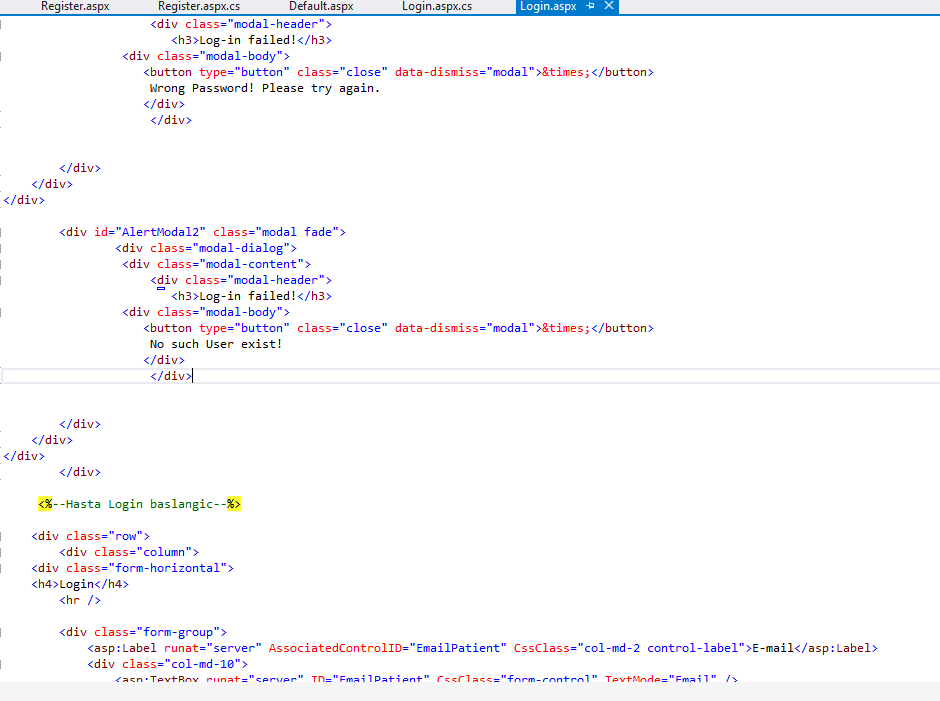
The classes in our project are:

* Site.Master : The master page of our site. Includes a fixed to top navbar, which enables the users to browse through the site and do functions such as login and register if they’re not logged in. If they’re logged in, they can logout or view their profile information. It also consists of a footer which shows the copyright text.
* Register : Registration form for the patients. The patients can login after they register successfully to the site. Several fields are required in order to register, and an already registered user cannot register again.
* Login : Enables every user type(doctor, patient, admin, laborant) to login.
* Default: The main page of the hospital. Every user, logged in or not, can see this page. It includes of a carousel as hospital information and some other informal text.
* About and Contact : The default aspects of a bootstrap themed web page. They show the “about” section and information of the hospital, and the contact info.

**Login**

**Asp.net code**

****

****

****

****

**C# Code**

****

****

The login form provides side-to-side two login forms. One is for patients, and the other is for the hospital staff (admin, doctor and laborant). The form checks if the user exists, if the user exists and the e-mail and password match, they are successfully logged in. If the e-mail exists in the database but the password is wrong, it gives a “wrong password” notification via popup. If the e-mail doesn’t exist in the database, it gives “no such user exists” notification. Only patient login is implemented yet.

**Register**

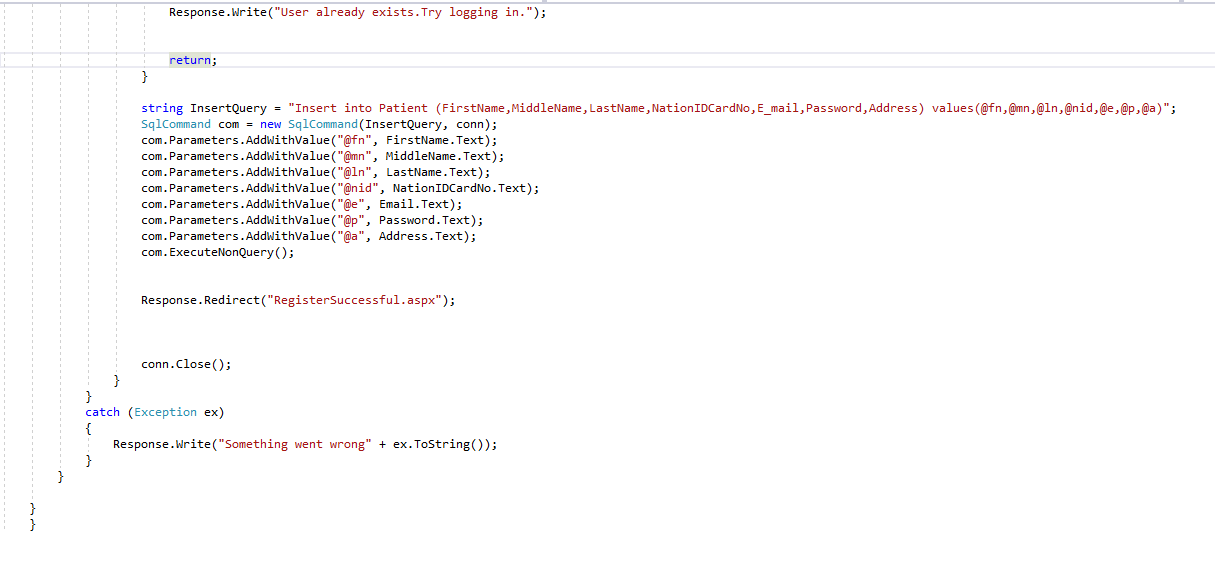
**Asp.net code**

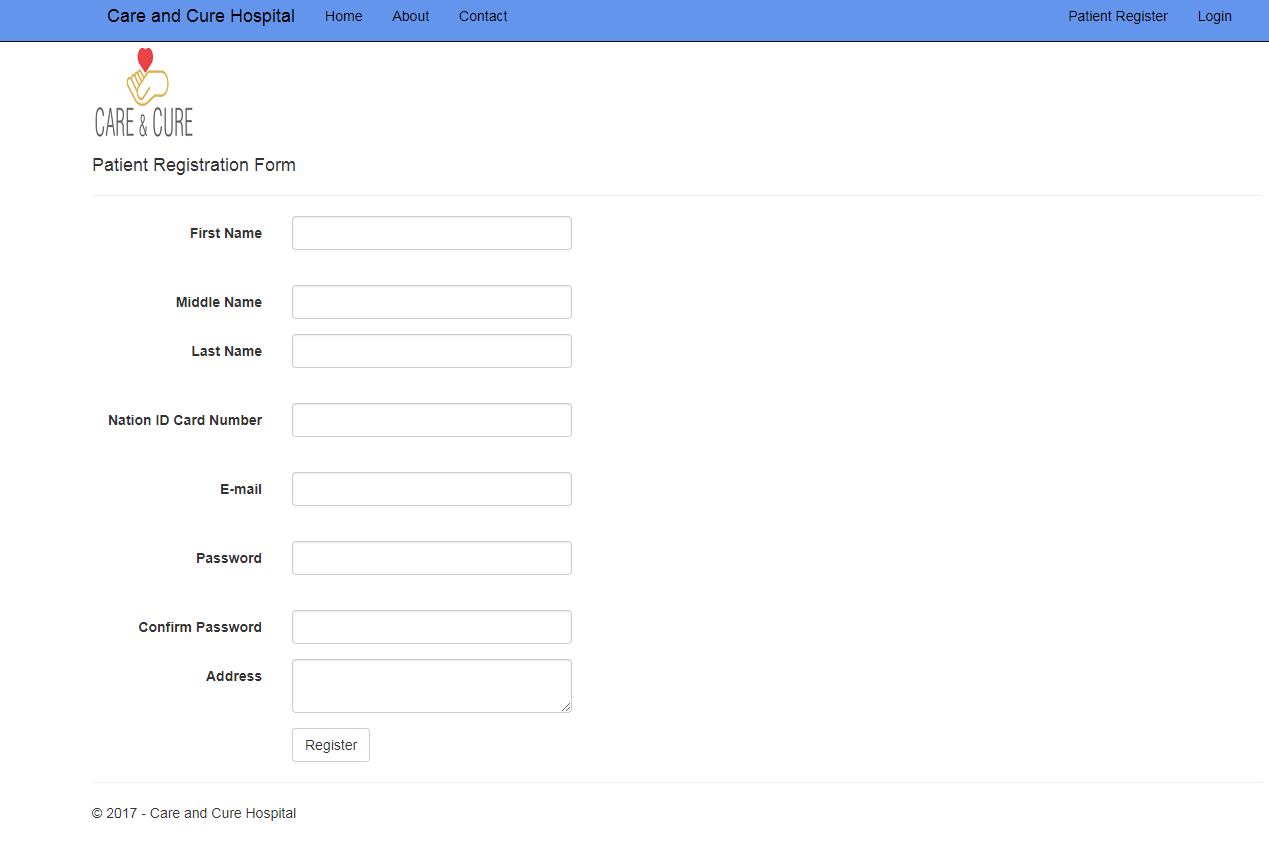
****

****

**C# Code**

****

****

****

Enables the patients to register in order to login. Patients have to fill out the first and last name, Nation ID card number, e-mail , password and address fields. Password and Confirm password fields must be the same. If one of this fields is empty, and register button is pressed, it gives “cannot be empty” exception via a red label underneath them. In the c# code, by the database it is checked if the user is already registered. If a user tries to register with the same e-mail or nation id card number, “User already exists. Try logging in” message is shown and a duplicate registration is not allowed.