**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**



* **Architectural Design**
* **Conceptual Architecture**

The **Layers** architectural pattern structures applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

In the data access layers will perform CRUD (Create, Read, Update, Delete) operations.

Business logic or domain logic is the part of the program that encodes the real-world [business rules](https://en.wikipedia.org/wiki/Business_rule) that determine how data can be [created, stored, and changed](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete).

The presentation layer is responsible for the delivery and formatting of information to the application layer for further processing or display.

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

View - View represents the visualization of the data that model contains.

Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

* **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]*

* **Data Model**

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

* **Unit Testing**

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

* **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.]*

* **Design Model Refinement**
* *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*
* **Construction and Transition**
* **System Testing**

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

* **Future improvements**

*[Present future improvements for the system]*

* **Bibliography**