|  |  |
| --- | --- |
| Polytechnic University of Bucharest | |
| Software Engineering Quiz-generator Software Design Document | |
| Team:  Teodora Mohai, gr. 1262E  Salvador Canas Moreno , gr. 1262E  Irina Talmaciu, gr. 1262E  David Suarez Caro, gr. 1262E |  |
| Coordinator:  As. Lucian Pestritu  As. Alexandra Todiruta | Date created:  Wednesday, November 28, 2012 |

# Delivery Report

(will be delivered along with the project)

Name Group Project implementation [%, reason] Signature

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Delivery date:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Table of Contents

[Delivery Report 2](#_Toc254468674)

[Table of Contents 3](#_Toc254468675)

[System Design 4](#_Toc254468676)

[Document Change History 4](#_Toc254468677)

[1. Introduction 4](#_Toc254468678)

[1.1. Purpose 4](#_Toc254468679)

[1.2. Target Public 4](#_Toc254468680)

[1.3. Definitions, Acronyms and Abbreviations 4](#_Toc254468681)

[1.4. Document Structure 4](#_Toc254468682)

[2. References 5](#_Toc254468683)

[3. Decomposition Description 6](#_Toc254468684)

[3.1. Modules Description 6](#_Toc254468685)

[3.2. Description of Concurrent Processes 6](#_Toc254468686)

[3.3. Description of Data Modules 6](#_Toc254468687)

[4. Dependency Description 8](#_Toc254468688)

[4.1. Dependencies among modules 8](#_Toc254468689)

[4.2. Dependencies among processes 8](#_Toc254468690)

[4.3. Dependencies among data modules 8](#_Toc254468691)

[5. Interface Description 9](#_Toc254468692)

[5.1. Module Interfaces 9](#_Toc254468693)

[5.2. Processes Interfaces 9](#_Toc254468694)

[6. Detailed Design 10](#_Toc254468695)

[6.1. Modules detailed design 10](#_Toc254468696)

[6.2. Data Modules Detailed Design 10](#_Toc254468697)

[Appendices 11](#_Toc254468698)

[A1. Use cases diagrams 11](#_Toc254468699)

[A2. Class diagrams 11](#_Toc254468700)

[A3. Sequence diagrams 11](#_Toc254468701)

[A4. Document evolution 11](#_Toc254468702)

[A5. Conclusions on activity 11](#_Toc254468703)

# System Design

According to the IEEE STD-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*.

## Document Change History

Depending on the chosen type of development, retention of all changes made to this document may be useful. For example, for a project using the *Waterfall w/ milestones* methodology, all the changes made between two project evaluation moments (*milestones*) will be retained. A chronological sorting of the list of changes is indicated.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | The author/authors of the change | Details of changes |
| 1.00 | 28.11.2012 | Teodora Mohai  Salvador Canas Moreno Irina Talmaciu  David Suarez Caro | Basic version |

## 1. Introduction

### 1.1. Purpose

This Software Design Document provides the design details of University Politehnica of Bucharest Quiz-generator application. It means this is the document which will guide all the developers in their work. All the technical details have to be written clearly in this document in order to be understood by all the developers.

The expected audience is Polytehnic University of Bucharest, including all the teachers who work there, the student developers, the student assistants and people who will maintain the application.

### 1.2. Target Public

|  |  |  |
| --- | --- | --- |
| **Name** | **Last name** | **Role** |
| Teodora | Mohai | Student developer |
| Salvador | Moreno Canas | Student developer |
| Irina | Talmaciu | Student developer |
| David | Suarez Caro | Student developer |
| Lucian | Pestritu | Assistant |
| Alexandra | Todiruta | Assistant |
| Nicolae | Goga | Proffesor |

### 1.3. Definitions, Acronyms and Abbreviations

**Acronyms:**

**XML**- Extensible Markup Language

**SQL**-Structured Query Language

**UML**-Unified Modeling Language

**IEEE**-Institute of Electrical and Electronic Engineers

**SDD**-Software Design Document

**SRS**-Software Requirement Specification

**Tbn**-To be named

**Tbd**-To be decided

**Web Site**-A place on the world wide web

**Definitions:**

**Connection** **String** = string used to connect to a database, containing server name or IP, database name, database user name and password.

**Data** **Source** = data retrieved from the database based on an SQL query, used in displaying structured information.

**Query** = primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format. Many database management systems use the SQL standard query format.

**Java** = object-oriented [programming language](http://en.wikipedia.org/wiki/Programming_language) originally developed by [James Gosling](http://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) and released in 1995 as a core component of Sun Microsystems' [Java platform](http://en.wikipedia.org/wiki/Java_%28software_platform%29).

**Class =** a construct that is used to create [instances](http://en.wikipedia.org/wiki/Instance_%28computer_science%29) of itself – referred to as class instances, class objects, instance objects or simply [objects](http://en.wikipedia.org/wiki/Object_%28object-oriented_programming%29).

**Method** =[subroutine](http://en.wikipedia.org/wiki/Subroutine) (or procedure) associated with a [class](http://en.wikipedia.org/wiki/Class_%28computer_science%29). Methods define the behavior to be exhibited by [instances of the associated class](http://en.wikipedia.org/wiki/Instance_of_a_class) at program run time.

**Entity – Relationship model** (**ER model**) = an abstract way to describe a [database](http://en.wikipedia.org/wiki/Database). Diagrams created to design these entities and relationships are called entity–relationship diagrams or ER diagrams.

**Pattern** = software solutions that solve a recurring problem within a given context. The use of patterns has many advantages. Patterns encapsulate a design expert's time and expertise to solve a software problem.

### 

### 1.4. Document Structure

The remaining chapters and their contents are listed below.

Section 2 is a reference list which shows all the sources used to develop this document.

All the references are numbered and dated. The authors are also named with the corresponding reference.

This allows a clear explanation where you can obtain more details of some part of the design.

Section 3 is a Decomposition Description that specifies the different modules we described previously in the SRS document and all their characteristics detailed. This chapter also contains the description of the concurrent processes and the data modules with all their specifications.

Section 4 is a Dependency Description where you can find all the dependencies firstly named in the previous chapter and detailed in this chapter.

Section 5 is an Interface Description which presents the external characteristics of the user interface showing all the different actions the users can do.

Section 6 is Detailed Design which shows all the modules previously named and described. This part contains diagrams and examples of the code that developers can follow.

Section 7 is the Appendices where you will find all the diagrams which describe the behavior of the application.

## 2. References

[1] IEEE STD-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*

[2] [IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,”

2001 edition.

[3][XML] All the tutorials to understand the XML language are in <http://www.w3schools.com/xml/>.

[4][HTML]The standars HTML documents are described in <http://www.w3.org/>.

## 3. Decomposition Description

This article corresponds to chapter 6.2.1., Decomposition description, from [1].

### 3.1. Modules Description

Modules description according to 5.3.1 -5.3.10, from [1]. The modules can be identified on the use case diagram(s) from the Software Requirements Specification.

#### 3.1.1. Description of Module 1

|  |  |
| --- | --- |
| Name | Name of module 1 |
| Type | Subprogram *or* Code module *or* Procedure *or* Process *or* Data module |
| Purpose | Purpose of the module |
| Way of operating | Description of the operating way |
| Subordination | [optional] Identification of the subordinate modules |
| Dependencies | [optional] Identification of the module’s dependencies |
| Resources | Necessary resources for this module (e.g. MySQL, PHP) |

#### 3.1.2. Description of Module 2

...

### 3.2. Description of Concurrent Processes

Description of concurrent processes according to 5.3.1 – 5.3.10, from [1]. The processes can be identified on the sequence diagram(s) from the Software Requirements Specification.

#### 3.2.1. Description of Process 1

|  |  |
| --- | --- |
| Name | Name of process 1 |
| Type | Process |
| Purpose | Purpose of the process |
| Way of operating | Description of the operating way |
| Subordination | [optional] Identification of the subordinate processes |
| Dependencies | [optional] Identification of the process’ dependencies |
| Resources | Necessary resources for this process (e.g. global timer (NTP)) |

#### 3.2.2. Description of Process 2

...

### 3.3. Description of Data Modules

Description of data modules according to 5.3.1 – 5.3.10, from [1]. The data modules can be identified on the use case diagram(s) from the Software Requirements Specification.

#### 3.3.1. Description of Data Module 1

|  |  |
| --- | --- |
| Name | Name of data module 1 |
| Type | Data module |
| Purpose | Purpose of the data module |
| Way of operating | Description of the way of operating on data from this module |
| Subordination | [optional] Identification of subordinate modules |
| Dependencies | [optional] Identification of the data module’s dependencies |
| Resources | Necessary resources for this data module (e.g. relational data base, SQL compatible) |

#### 3.3.2. Description of Data Module 2

...

## 4. Dependency Description

This chapter corresponds to chapter 6.2.2, *Dependency description*, from [1].

### 4.1. Dependencies among modules

The dependencies identified in chapter 3.1 are detailed. Detailed diagrams are recommended.

### 4.2. Dependencies among processes

The dependencies identified in chapter 3.2 are detailed. Detailed diagrams are recommended.

### 4.3. Dependencies among data modules

The dependencies identified in chapter 3.3 are detailed. Detailed diagrams are recommended.

## 5. Interface Description

This chapter corresponds to chapter 6.2.3, *Interface description,* from [1].

### 5.1. Module Interfaces

The module interfaces described in chapter 3.1 are detailed. Detailed diagrams are recommended.

#### 5.1.1. Module 1 Interface

|  |  |
| --- | --- |
| Name | Name of module 1 (see 3.1.1) |
| Type | (see 3.1.1) |
| Purpose | (see 3.1.1) |
| Way of operating | (see 3.1.1) |
| Interface 1 | Description  E.g.: result Login(user\_name, password) |
| Input | E.g.:  user\_name: string – the user name, read from the keyboard  password: string – user access password, read from the keyboard |
| Output | E.g.:  result: string – OK or FAILED, depending on the search result of the (user\_name, password) pair in the user data base (see chapter X.Y) |
| Description | Description of the module in which the interface works  E.g.:  The registration operation is made by introducing a user name and a password from the keyboard; the existence of this pair is verified in the user data base; the data base search result will be returned as a *string*. |

#### 5.1.2. Module 2 Interface

...

### 5.2. Processes Interfaces

The processes interfaces described in chapter 3.2 are detailed. Detailed diagrams are recommended.

#### 5.2.1. Process 1 Interface

#### 5.2.2. Process 2 Interface

...

## 6. Detailed Design

This chapter corresponds to chapter 6.2.3, *Interface description*, from [1].

### 6.1. Modules detailed design

The modules described in chapter 3.1 are detailed. Detailed diagrams are recommended.

#### 6.1.1. Module 1

|  |  |
| --- | --- |
| Name | Name of module 1 (see 3.1.1) |
| Type | (see 3.1.1) |
| Purpose | (see 3.1.1) |
| Way of operating | (see 3.1.1) |
| Classes | Identification of constitutive classes |

**6.1.1.1. Module 1, class 1**

|  |  |
| --- | --- |
| Name | Name class 1 |
| Purpose | Description of the purpose of this class within the module |
| Members | Identification of the class’ members |
| Methods | Identification of the constitutive methods (functions) |

Class diagram (see class diagrams from the Software Requirements Document).

**6.1.1.1.1. Module 1, class 1, method 1**

|  |  |
| --- | --- |
| Name | Name of method 1 |
| Purpose | Description of the purpose of this method within the class |
| Prototype | Description of the method’s prototype |
| Input | Description of the method’s input |
| Output | Description of the method’s output |
| Caller | Routines that call this method |
| Calls | Other routines called by this method |
| Algorithm | Description of the algorithm/procedure of operating |

...

#### 6.1.2. Module 2

...

### 6.2. Data Modules Detailed Design

The data modules described in chapter 3.3. are detailed. Detailed diagrams are recommended.

#### 6.2.1. Data module 1

#### 6.2.2. Data module 2

...

# Appendices

## A1. Use cases diagrams

## A2. Class diagrams

## A3. Sequence diagrams

## A4. Document evolution

## A5. Conclusions regarding the activity