**3. System Requirements**

**3.1. External Interface Requirements**

Information is picked from a pool of files and unique quizzes are generated. The input files contain the questions and answers in an XML format. The output files are pairs of HTMLs with questions and answers.

The program is developed using a specialized IDE (like Eclipse) and written in Java. This requires the user to installed a Java Runtime Environment.

Consequently, for the software interface are used specific xml and html packages like: java.xml.parsers and javax.swing.text.html. This, also, needs to be included in the existing JRE on the user's machine.

Having a command line interface, as hardware, a functional keyboard is required.

**3.2. Functional Requirements**

**3.2.1. Product Requirements**

In order to generate unique quizzes for each student, the user interface must allow the following operations:

**Extract questions from a pool**

The user enters the location of the question pool and the questions are extracted randomly generating the quizzes. The program must read from XML files and access any element from it.

The possibility to choose the directory from where to pick the questions allows the user to change the source file accordingly to the necessity of the person.

**Select the quiz format**

The user has the ability to select the quiz format. One quiz contains a number of questions chosen by the operator, this means that there are different subjects and from each subject the handler chooses a number of questions.

The user also introduces a number of students which represents the number different quizzes to generate.

**Select the output location**

After selecting number and format of quizzes, the location of the output files is passed to the application. The processing results in a number of HTML files that are used as questions and the counterpart answers.

**Generate the output file**

Processing the quizzes implies selecting random questions from each quiz and, in case of multiple choice responses, the possible answers are printed in random order in the output HTML file.

Faciliting the grading process, the quiz and its answers will be identified by a serial number printed at the top of the page.

**3.2.2. Use Case Description**

**3.2.2.1. Quizzes Generate**

|  |  |
| --- | --- |
| **Use Case Name:** | Quizzes generate |
| **Brief Description:** | This operation permits to:  - choose the xml files sources;  - select format;  - select quizzes;  - select output file. |
| **Priority:** | Essential |
| **Trigger:** | The user starts the application. |
| **Preconditions:** | The xml files must exist and be accessible. |
| **Basic Path:** | 1. Choose the xml file;  2. If the file is not in the correct format generate an error message and return to  point 1;  3. Choose the number of questions from that file;  4. If the number is greater than the total number of questions from that file, generate an error message and go to point 3;  5. If the input is correct, choose to continue or go to point 1;  6. Choose the number of quizzes;  7. Choose the output directory path, where the generated quizzes will be saved. |
| **Alternate Path:** | n/a |
| **Post condition:** | The files are saved in html format in the output directory. For each quiz there are two files: one with the questions and the other with answers. |
| **Exception Path:** | If the input xml files or the output directory suffer modifications during processing, the behaviour is unknown. |

**3.3. Performance Requirements**

Being a simple application which is not using a graphical user interface, the user will run the program from his computer. The program can be ran simultaneously from different computers. It only needs to have as input file a xml file and, at the end, the operator has to obtain as output file a html file.

The program must run every time it will be called and as estimated the application must complete in less than 10 seconds.

**3.4. Design Constraints**

Because there is no graphical user interface, the program must assure it is compliant with the rules described in chapter 2 – point 2.4.

**3.5. Software System Attributes**

Confidence

Maintenance

Tenacity

Portability

Fault tolerance

Easy of handling

**3.6. Other System Requirements**