# Software Requirements Specification

for

# <Project iTest>

**Requirements for Version 1.3.0** 

Prepared by <Δεσπούδης Θεοφάνης>

<Τεχνολογία Λογισμικού, ΑΠΘ>

**AEM <890>** 

<24-1-2008>

# **Table of Contents**

1. Introduction	
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Project Scope	2
1.5 References	
2. Overall Description	5
2.1 Product Perspective	
2.2 Product Features	6
2.3 User Classes and Characteristics.	
2.4 Operating Environment	10
2.5 Design and Implementation Constraints	10
2.6 User Documentation	10
2.7 Assumptions and Dependencies	
3. System Features	
Section A	
<ul><li>3.1 System Feature Execute iTestClient::REQ1</li></ul>	11
3.3 System Feature Connect to a server(online mode)::REQ2	11 12
3.4 System Feature Change save answers::REQ4	12 12
3.5 System Feature Hide question names::REQ5	12
3.6 System Feature Hide current answers at the end of the test::REQ6	13
3.7 System Feature Show iTest "About" information::REQ7	14
3.8 System Feature Quit iTestClient::REQ8	14
3.9 System Feature Finish test Session::REQ9	14
3.10 System Feature View Test Information::REQ10	15
3.11 System Feature View Test Information::REQ10	15
3.12 System Feature Student enters Login name::REQ12	16
3.13 System Feature Student enters Login name::REQ12	16
3.14 System Feature Run Test Session::REQ14	17
3.15 System Feature Test Time expires::REQ15	
Section B	
4. External Interface Requirements	
4.1 User Interfaces	
4.2 Hardware Interfaces	
4.3 Communications Interfaces	
5. Other Nonfunctional Requirements	34
5.1 Performance Requirements	
5.2 Safety Requirements	
5.3 Security Requirements	35
5.4 Software Quality Attributes	
Appendix A: Glossary	
Appendix B: Analysis Models (Intro)	
Appendix C: Issues List - ChangeLog	37

# **Revision History (may not be needed)**

Name	Date	Reason For Changes	Version

## 1.Introduction

## 1.1 Purpose

The purpose of this document is to specify the requirements and preview some elements of the analysis model of the program iTest.

iTest is a simple program which consists of two programs:

- iTestServer question/answer database editor and exam server
- iTestClient the program to install onto each client computer

**iTestServer:**offers an easy way to organize a database of questions and answers. It makes use of colors in a way that makes this even easier. It features a simple rich text editor that allows you to format your questions just the way you like.

iTestServer is also used as the server, to which client computers then connect in order to retrieve a test. It can save test sessions for later viewing and allows you to view and print statistics.

iTestClient: is used as a client program. It loads the test created by the server and randomly chooses the questions to form a unique test for the student. It can either connect to the server over a network or load the test from a file exported by the server. When the test is over, the results are saved into a file and if the client is connected over network, they are automatically sent to the server. In case of connection breakdown, the backup file can be used - it can be loaded back into the server.

This requirements Specification is based on the version **1.3.0** of the program.

#### 1.2 Document Conventions

In general this document prioritizes in writing the schema of the client-server mechanism that takes hold in this project and then analyzing in detail the tools that are available in the iTest editor and question database system. Therefore there are lots of abstractions to represent in a more convenient way the objects and their behavior on the system. Every requirement statement is assumed to have its own priority as to define in most appropriate way the system behavior. In addition there are various figures that represent the described system ,where it is needed, and serve only for better understanding of the deployment. Please refer to the official documentation of the program at <a href="itest.sourceforge.net">itest.sourceforge.net</a> if you have specific questions based on your system.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for any individual user, developer, tester, project manager or documentation writer that needs to understand the basic system architecture and its specifications. Here are the potential uses for each one of the reader types:

- **Developer**: The developer who wants to read, change, modify or add new requirements into the existing program, must firstly consult this document and update the requirements with appropriate manner so as to not destroy the actual meaning of them and pass the information correctly to the next phases of the development process.
- User: The user of this program reviews the diagrams and the specifications presented in this document and determines if the software has all the suitable requirements and if the software developer has implemented all of them.
- **Tester**: The tester needs this document to validate that the initial requirements of this programs actually corresponds to the executable program correctly.

For each one of the reader types to better understand this document, here is a suggestion of the chapters to read in this document:

- **Developer**:(1.1,1.3,2.2,2.3,2.5,2.7,3,4,5 and rest)
- User:(1,2.1,2.2,2.3,4.1,5.5)
- Tester:(1.1, 2.1, 2.4, 2.5, 2.7, 3, 4, 5)

This document contains the necessary requirement and some aspects of the analysis of the requirements and is organized based on the IEEE Standard for Software Requirements Specification (IEEE 830-1993).

#### Overview

- **1.Introduction**:Provide an overview of the application, describe the document structure and point the individual objectives.
- **2.Overall Description:**Provide the specification of the system model, the classes model, the main constraints and the list any assumed factors that used within this document.
  - **3.System Features**: Provide the analysis of the requirements by feature.
- **4.External Interface Requirements**: Provide the visualization of the program and the requirements that are related with hardware, software and networking.
- **5. Other Nonfunctional Requirements**: Provide some other constraints that apply to factors such as performance, safety and security.

## 1.4 Project Scope

This program iTest offers a complete and easy way to create and organize different sets of questions categorizing them with flags or by groups so as to use them in an appropriate way to make tests. The benefits of this is that the traditional testing methods (like written tests) are upgraded with online interactive support using computers as a tool to take tests.

Its main objective is to set up an remote connection test environment that allows users like teachers to make tests using a server that students can connect and answer the questions and print their success results. The tests can be different based on the server settings.

The goal is to make possible to teachers to provide tests via a server that anyone can connect using the port settings via LAN or wider network, provided that they have already installed the client and the server is running.

Here are some usefull screenshots that represent the program:

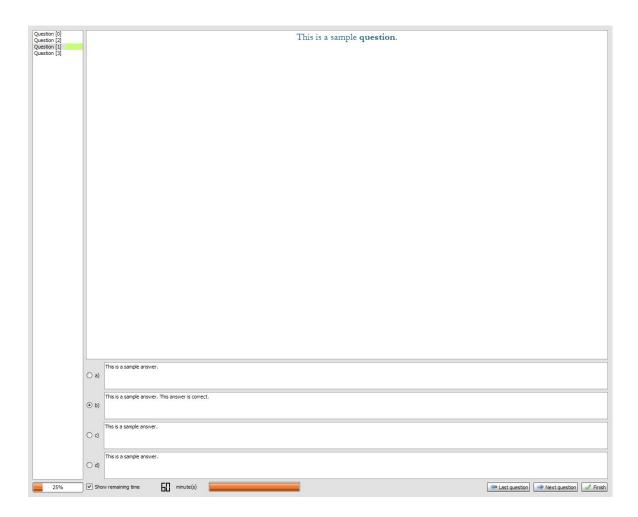


Figure 1.1 -iTestClient

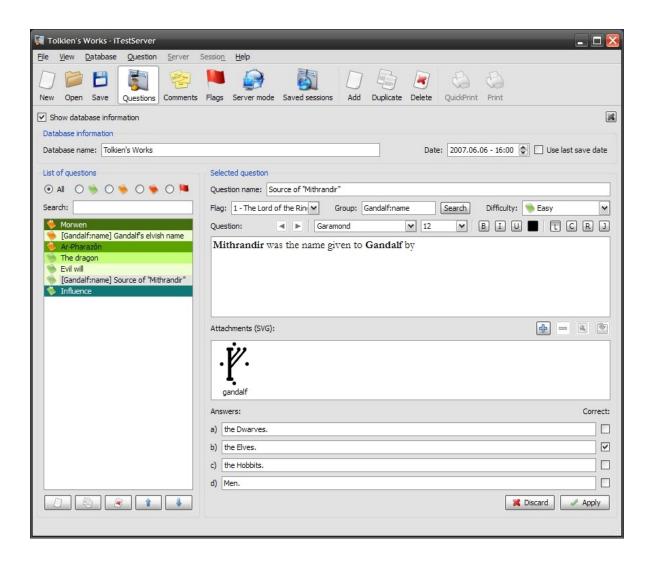


Figure 1.2 -iTestServer

## 1.5 References

This citation is used as a model of reference:

[IEEE Std 830-1998]

# 2.Overall Description

## 2.1 Product Perspective

Since this is a open source program it is under the **GNU General Public License (GPL).** so the source code is free to download. There are various reasons why should anyone use this program. First its a framework where you can create, edit and store multiple choice questions. Second is a easy and reliable testing program that is very unique in its category where you can use the questions you have created in a way that represents test sessions. And third due to its open source nature you can modify it according to your needs.

The major components of the system as noted in the previous sections is the iTest client window and the iTestServer window.

- The iTestClient is a simple client-window form that communicates with the active iTest Server and its purpose is to run the test sessions that were provided and prints the results. There can be multiple clients that can connect to a single server.
- The iTestServer is a question database editor that it can be enabled in server mode. As long as is in that state it opens ports to accept iTestClient connections or offline connections record the session statistics such as who is connected at the present time and how many questions has answered correctly, saves this information in log files and prints the results.

Bellow in figure 1.3 you can see the schema of the client-server relationship of the system.

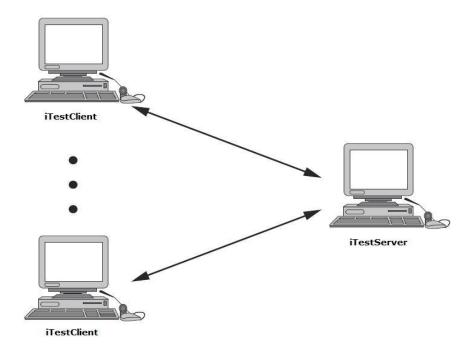


Figure 2.1 Client-Server Mechanism

Because there aren't many similar programs that offer a complete, adjustable and user-friendly environment for setting up a multiple choice online or offline session tests this software is very useful for individual users who want to use automated methods and tools to make tests.

#### 2.2 Product Features

The major features this program contains are the following:

- Cross platform support: Offers operating support for most of the known and commercial operating systems
- Language support: Offers multiple language support for global use.
- **Tcp or Connectionless support:** Clients can connect to the server using an tcp connection or connectionless using a removable disk that loads the test from it.
- Extended text editor: Offers an extended text editor options for writing comments or modifying questions using colored text to outline most useful phrases.
- Statistic tool: Questions can also be rated based on how many times they have been answered correctly in a statistics sheet that displays this information. Furthermore overrated or underrated question difficulty can be adjusted accordingly.
- **Flags:** Questions can be categorized using flags so that they can be manipulated more effectively and to show questions on a particular domain only.
- **Printing:** Offers printing support. Test session results can now be printed both from the client and server side point of view.
- **Saved sessions:** Every test session can be saved in a log file and studied further. Particular session of interest can be archived.
- **Sorting questions:** Sort questions from A-Z or Z-A.
- Filter questions: Show questions based on their difficulty or flag category.
- **Hidden questions:** Questions can be removed but not deleted from the database using the hidden option .They can be revealed later using the show hidden questions option.

Bellow in *figure 2.1* you can see the class diagram of some of those features such as the connectionless support and the attachments support.

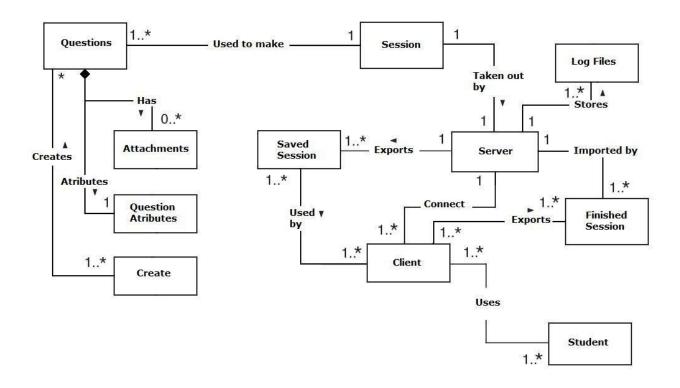


Figure 2.1 iTest class Diagram

#### 2.3 User Classes and Characteristics

For conventional reason we name each of the user classes-actors with this format:

#### **Physical Actors:**

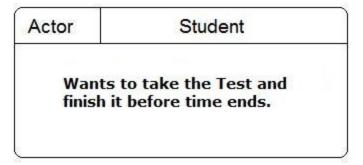
- **Student**: The student is the one that takes the test using the iTestClient interface. Multiple students can connect to the server and take the test. It is convenient to think that every student represents every user that takes the test and that he connects to one client.
- **Teacher**: The Teacher is the one that uses the iTestServer as an editor to create a database of questions and define the test options. It is convenient to think that every Teacher represents every user that uses the Database Editor to make tests.

#### **System Actors:**

- Client: The client is the system that connects to the server and handles the tests based on the session and finally submits the information back to the server.
- **Server**: The server is the system that accepts multiple connections from clients and saves the results.

• **Database Editor:** The database editor handles all the question database tools used to construct, modify and save questions and make different tests for the students. Also it is responsible for logging the session info and restore them for better study.

The primary actor is the client that connects to the server and takes the test. The term student is to be used as a physical actor to describe the physical user of the client. The same term is used for the Teacher who is the physical user of the database editor and the server and is responsible for the good behavior of the system and to make sure everything goes well. Bellow are the main goals of each of the actor list:



Wants to make a database of questions, build test sessions, and test his students.

Also he wants to make sure everyone is not cheating.



Must connect to Server and keep connection alive while students take the tests.

Also must be able to print results and offer a variety of options.

# Actor Server

Must set up a test session correctly, accept client connections and log the whole proccess. Also must be able to export the tests to a file so off-line clients can take the tests.



Must offer to teachers a complete set of tools to manipulate, store and prepare test sessions.

Figure 2.3 Actors roles

## 2.4 Operating Environment

This program will operate in the following operating environment for the client and the server GUI:

- Apple Mac OS X (Universal)
- Linux/Unix (Source code)
- Microsoft Windows (Installer)

## 2.5 Design and Implementation Constraints

This program is created using C++ programming language and uses the <u>Qt4 libraries</u> for the main iTestClient and iTestServer modules. So a minimum PC having at least 64mb of RAM and CPU over 400mhz is required to run the program with good speed. Also the program uses at least 15 megabytes of hard disk space to store the program libraries.

An installer is used to unpack them all in an install wizard.

For language support except from the basic English language pack there is also a Turkish,Russian,Portuguese,Spanish and Slovak language packs that can be enabled within the program.

For the connection stream Tcp-Ip is used as its the common gateway for internet applications. The program is provided AS IS with ABSOLUTELY NO WARRANTY OF ANY KIND as noted in the official project website.

#### 2.6 User Documentation

Here are the official links of the project where you can retrieve more information about it and download the latest version:

#### Online Documentation

-http://itest.sourceforge.net/documentation/1.3/en/

#### Official Webpages

- -http://itest.sourceforge.net/
- -http://sourceforge.net/projects/itest/

## 2.7 Assumptions and Dependencies

For creating the windows forms and setting up the core program Trolltech Qt framework and classes was used that offers a complete set of tools for developing cross platform projects and a language editor for additional translations. For better understanding the know-what to do rather than the know-how to do it we assume that the reader is not interested for knowing how to create and design windows forms and how the program is coded at that time.

# 3. System Features

Using the actor list we begin the analysis of the client-server features that takes place in Section A and then for each tool that is being reported so far in Section B we analyze its requirement basis in the Database Editor program. All these features require that the server is on.

#### Section A

#### 3.1 System Feature Execute iTestClient::REQ1

#### 3.1.1 Description and Priority

A student executes the iTestClient and the start up client screen appears. Priority level=low

#### 3.1.2 Stimulus/Response Sequences

Preconditions:None
1.Student double clicks iTest.exe.
2.iTestClient start up interface shows up.
3.Include REQ10.
Postconditions:None

#### 3.1.3 Functional Requirements

Includes REQ10 Specializes:REQ2,REQ3,REQ4,REQ5,REQ6 Extends:REQ10 Connects:REQ12,REQ7,REQ8 Connected by:REQ13

## 3.2 System Feature Connect to a server(online mode)::REQ2

#### 3.2.1 Description and Priority

The Teacher sets Server name and Server Port and connects to the server. Priority level=low

#### 3.2.2 Stimulus/Response Sequences

Preconditions: Teacher knows servers name and port. Server mode is on.

- 1. Teacher enters Servers name.
- 2 Teacher enters Server Port

- 3. Teacher elects to Connect to the server.
- 4. Teacher elects to view the test information.

Postconditions: Teacher is connected to the server. The test information is displayed. Abnormal Paths: If the servers name and port is incorrect iTest informs the student that which one of the two is incorrect.

#### 3.2.3 **Functional Requirements**

Specializes:REQ10 Specialized by:REQ1

## 3.3 System Feature Connect to a server(offline mode)::REQ3

#### 3.3.1 **Description and Priority**

The Teacher browses exported test session file with the .itos extension and and loads sessions test information.

Priority level=low

#### 3.3.2 Stimulus/Response Sequences

Preconditions: Teacher has the appropriate .itos exported file in his computer.

- 1. Teacher browses his computer storage devices.
  2. Teacher loads the appropriate itos file.
- 3. Teacher elects to view the test information.

Postconditions: Teacher loads successfully the file and the tests information and comments is displayed in the form.

Abnormal Paths: If the .itos file is corrupted then a message informs the student that the file is wrong.

#### 3.3.3 **Functional Requirements**

Specializes:REQ10 Specialized by:REQ1

## 3.4 System Feature Change save answers::REQ4

#### 3.4.1 **Description and Priority**

The Teacher choses to save the answers to a selected storage folder. Priority level=low

#### 3.4.2 Stimulus/Response Sequences

Preconditions:None

1. Teacher unchecks the default save path of the clients log file with the extension

.itcl.

2. Teacher browses his file system and chooses the save path of the log file

3. Teacher may choose to rename the log file.

4. iTestClient shows up with the selected saved path.

Postconditions: The default clients log file is prompted to be saved at the specific path with the specific name.

#### 3.4.3 **Functional Requirements**

Specialized by:REQ1

## 3.5 System Feature Hide question names::REQ5

#### 3.5.1 **Description and Priority**

The Teacher choses to hide the question names.

Priority level=low

#### 3.5.2 Stimulus/Response Sequences

Preconditions:None

1. Teacher checks the "Hide questions name and show numbers only" checkbox under the other settings title.

Postconditions: The name of the answers are not shown at the test.

#### 3.5.3 **Functional Requirements**

Specialized by:REQ1 Extends: REQ14

# 3.6 System Feature Hide current answers at the end of the test::REQ6

#### 3.6.1 **Description and Priority**

The Teacher chooses not to show the correct answers at the test results. Priority level=low

#### Stimulus/Response Sequences 3.6.2

Preconditions:None

1. Teacher checks the "Do not show correct answers at the end of the test" checkbox under the other settings title.

Postconditions: The correct answers are not shown at the test results screen.

#### 3.6.3 **Functional Requirements**

Specialized by:REQ1

Extends:REQ13

## 3.7 System Feature Show iTest "About" information::REQ7

#### 3.7.1 Description and Priority

The Teacher choses to view the iTest "About" information.

Priority level=low

#### 3.7.2 Stimulus/Response Sequences

Preconditions:None

1. Teacher clicks the About button.

Postconditions: The About information box is displayed showing some information about the program

#### 3.7.3 Functional Requirements

Connected by:REQ1

## 3.8 System Feature Quit iTestClient::REQ8

#### 3.8.1 Description and Priority

The student choses to quit the iTestClient. Priority level=low

#### 3.8.2 Stimulus/Response Sequences

Preconditions:None

1.Student clicks the Quit button.

Postconditions: The iTestClient closes.

#### 3.8.3 Functional Requirements

Connected by:REQ1,REQ13

## 3.9 System Feature Finish test Session::REQ9

#### 3.9.1 Description and Priority

The student after he answered the questions finishes the test session Priority level=high

#### 3.9.2 Stimulus/Response Sequences

Preconditions: Student in question screen.

- 1. Student clicks the Finish button.
- 2.A pop up window appears to validate the choice
- 3.iTest client checks the student questions and corrects them.
- 4. Student is eager to view his test results.

Postconditions: The test results screen is being showed. An offline log file with a filetype .itcl that contains the all the test results is created in a default folder inside the system.

Abnormal Paths: If the student chooses not to finish the test when the message box appears then he returns to the test session and no further actions are performed.

#### 3.9.3 Functional Requirements

Included by:REQ14 Specializes:REQ11 Connects:REQ13

## 3.10 System Feature View Test Information::REQ10

#### 3.10.1 Description and Priority

The sessions test information such as test name,database name,flags,test date, time and pass mark is displayed Priority level=low

#### 3.10.2 Stimulus/Response Sequences

Preconditions: Student connects successfully to the server or loads successfully the appropriate .itos file

- 1. Student elects to view the test information.
- 2.iTest displays the test information to the appropriate data diagrams.

Postconditions: The test information is displayed.

#### 3.10.3 Functional Requirements

Specialized by:REQ2,REQ3 Included by:REQ1

## 3.11System Feature View Test Information::REQ10

#### 3.11.1 Description and Priority

The test results are being forwarded to a printer

Priority level=low

#### 3.11.2 Stimulus/Response Sequences

Preconditions: Student has finished the test and is viewing the test results screen. Postconditions: Test information is printed using the default printer settings.

#### 3.11.3 Functional Requirements

Specialized by:REQ9 Extends:REQ13

## 3.12 System Feature Student enters Login name::REQ12

#### 3.12.1 Description and Priority

The student is obliged to choose a login name after he has connected to the server and clicked the ready button. Priority level=medium

#### 3.12.2 Stimulus/Response Sequences

Preconditions: Student clicked the ready button from the iTest start up screen

- 1.Student enters his name.
- 2. Student clicks the Start button.
- 3. Student elects to answer the questions.

Postconditions: Name is registered to the server and the Test screen is displayed. Abnormal Paths: If the students name is black or its already registered a message is displayed on the screen that informs the student to correct it.

#### 3.12.3 Functional Requirements

Connected by:REQ1 Connects:REQ14

## 3.13 System Feature Student enters Login name::REQ12

#### 3.13.1 Description and Priority

The test results are displayed. Priority level=medium

#### 3.13.2 Stimulus/Response Sequences

Preconditions: Student clicked the finish button from the test session screen.

- 1.Student is eager to view his test results.
- 2.iTest client shows the test result details(test name, score, results).
- 3.If the student has checked the "Do not show correct answers at the end of the test" option then the correct answers are not displayed.

Postconditions: The Test result screen is filled with the appropriate information based on the students options.

#### 3.13.3 Functional Requirements

Connects:REQ1,REQ8 Connected by:REQ9,REQ15 Extended by:REQ5

## 3.14 System Feature Run Test Session::REQ14

#### 3.14.1 Description and Priority

The main test screen appears when the student can answer the questions Priority level:low

#### 3.14.2 Stimulus/Response Sequences

Preconditions: Student is connected to the server and has loaded the questions

- 1.Student elects to answer the questions.
- 2.iTest client shows the main test screen with the questions
- 3.If a question comes with a scalable picture then the picture is displayed for that question
- 4.If the student has checked the "Do not show Question names show numbers instead" option then the correct answers are not displayed.
- 5.Include REQ9
- 6.If time for test expires then include REQ15

Postconditions: The test screen is filled with the questions

#### 3.14.3 Functional Requirements

Includes REQ9,REQ15 Connected by:REQ12 Extended by:REQ6

## 3.15 System Feature Test Time expires::REQ15

#### 3.15.1 Description and Priority

The time of the test session expires Priority level:medium

#### 3.15.2 Stimulus/Response Sequences

- 1. Time counter expires for the given test
- 2.REQ15 is included.
- 3.iTestClient immediately stops current session and goes to the test results screen. Postconditions:Same as when the student clicks the Finish button.

#### 3.15.3 Functional Requirements

Included by:REQ14 Connects:REQ13

Bellow in *figure 3.1* you can see the corresponded analysis state machine diagram of the iTestClient system:

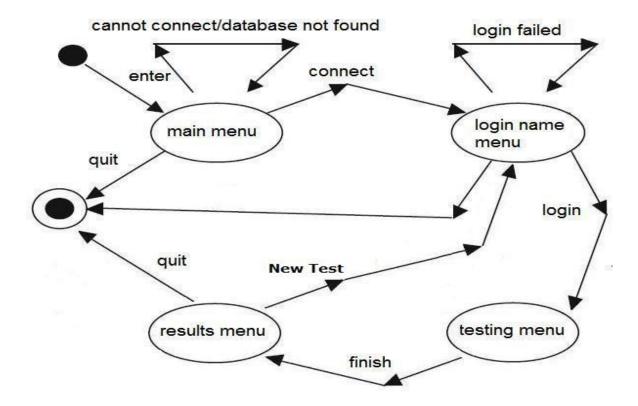


Figure 3.1 State machine diagram of iTestClient

#### **Section B**

#### **iTestServer**

In this section we describe the most common tools that the iTestServer uses in order to create test sessions. Almost every tool will have a use case figure or a screenshot to help.

Bellow in *figure 3.2* you can see an abstract overview of the methods the iTest Server Question Database Editor uses.

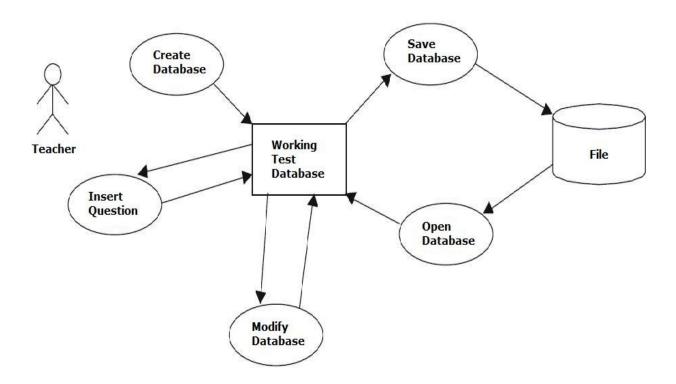


Figure 3.2 Database Editor Overview of the methods

When you first execute the iTestServer program you see a window that prompts the teacher or the user to open an existing recent database,browse for another database in the file system or create a new one.

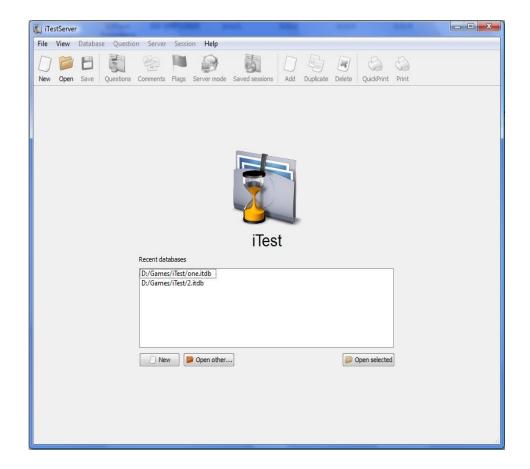


Figure 3.3 Open Database Screen

Below in Figure 3.4 you can see the state machine for opening a database

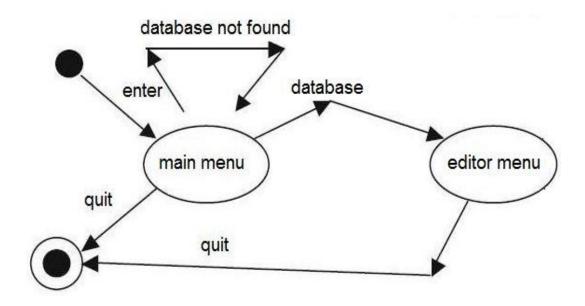


Figure 3.4 Open database state machine

When you create or open a database then the main editor menu appears as *Figure 3.5 shows* Initially there are no questions in the database so you can't do anything useful.

To create a new question simply click the lower left button bellow the **List of questions** list or from the Question dialog or by pressing Ctrl+Alt+A. Next write the name in the text space next to the **Question name** title and then next write the question itself bellow in the open space. The question potential answers can be filled bellow the **Answers** header witch you can check multiple correct answers. If you want to add an attachment picture the click on the plus (+) button next to the **Attachments(SVG)** header or using the **Question->Attachments** option in the menu dialog but make sure its a **Scalable Vector Graphics** picture. You can also **remove,edit** and **export** a SVG item after you have attached it. When you have finished all these preparations don't forget to press the **Apply** button bellow to actually store the question in the database. If you want to discard all your current progress then press the **Discard** button. Note that this only works with the unsaved changes you have made to that question. If you save a question the discard button cannot clear the fields you have entered

If you want to **delete** the selected question simply use the button bellow the **List of questions** list and next to the Add button or from the Question Dialog click the **delete** option or press the Ctrl +Alt+D combination.

If you want to **duplicate** the selected question ,simply use the button bellow the **List of questions** list and next to the delete button or from the Question Dialog click the **Duplicate** option or press the Ctrl+Alt+C combination.

If you check the **Show Database Information** checkbox you can rename the existing database you use and change the database save date to a fixed timestamp.

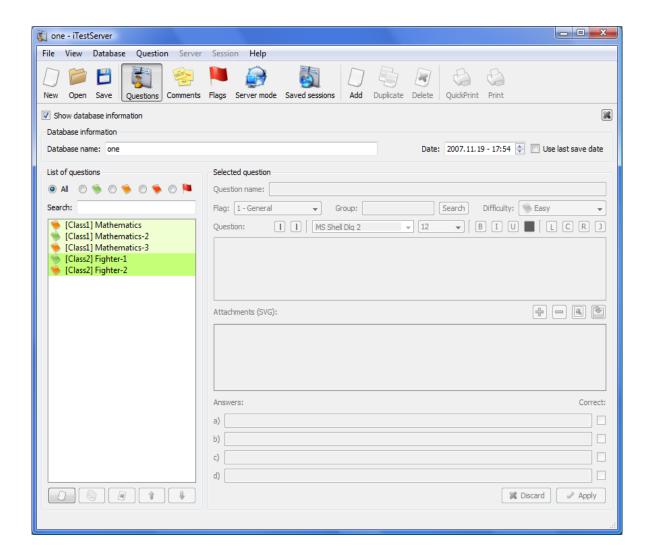


Figure 3.5 Main Database Editor Interface for creating questions

The text format of the question itself can be modified. For this the program offers an extended text editor where you can choose a font, alter the font size align the text itself and change the color of some characters. This option applies also to the **Comments** panel where you can write comments

about the current database and these will be transferred and shown later in the client main start up screen when they login to the server. The various options of the text editor are:

Redo/Undo:Redo or Undo last action.

**Select Font:**Change text font using the known fonts of the system.

Select Text Size: Change text size.

**Text Color:**Color your text.

Align left/right/center/justify: Align the text with various methods

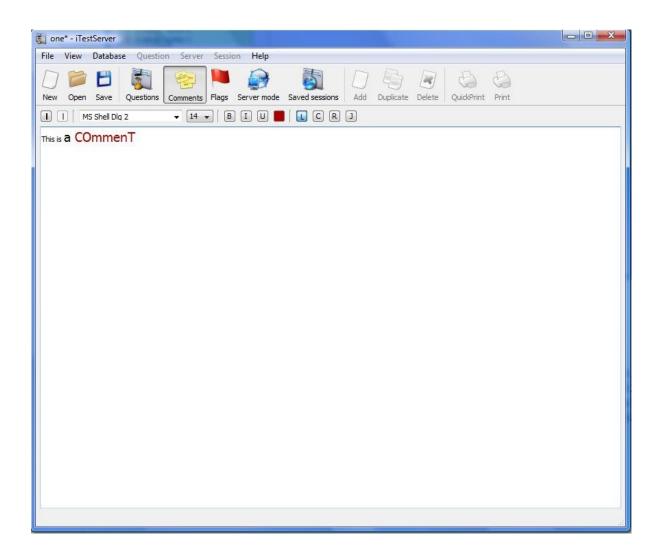


Figure 3.6 Comment Panel

You can further classify the questions you have created using **flags**. Flags are extra information that applies to every question that helps when you want to put questions in different buckets based on their knowledge domain or whatever domain you like. You can edit flags by clicking to the **Flags** panel above(with the Red Flag icon). There you see a new window that shows multiple checkboxes and Flag titles. Initially there are no flags checked so if you want to enable one simply check a box next to a flag to enable it. When you have created at least one question then it will be added to that flag automatically. Therefore you cannot destroy the first flag unless you delete all the questions of the database. There are 20 different flags you can use and each one of them has a unique color to make it more obvious and discrete. You can set up a name to a flag(the name can be blank)and it is also permitted 2 or more flags to have the same name because **every question can have only one flag.** When you have enables some flags press the **Apply** button to save your changes.

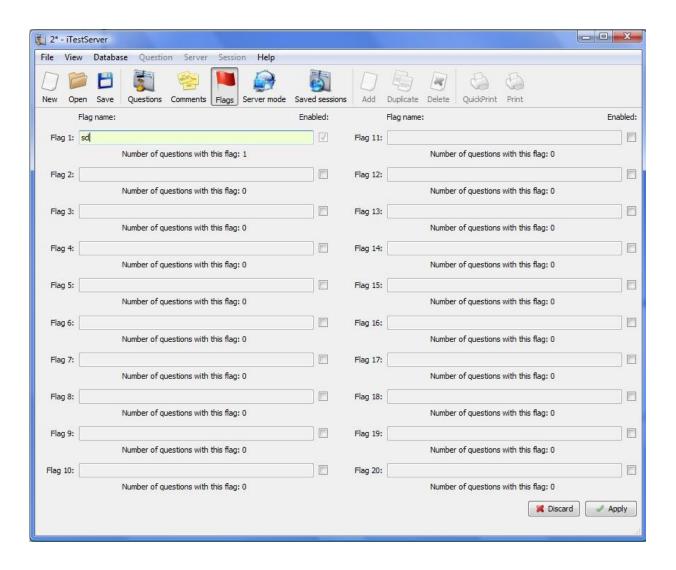


Figure 3.7 Flags Panel

The different questions you create come with a difficulty level that can either be set when you first create a question or can be adjusted automatically using a simple formula when the question has been used to tests. The are 3 different difficulties: Easy, Medium, Difficult and all questions have only one difficulty level at a given time. This also helps as a method of categorizing them based on their overall hardness. In the question panel in the left where the List of questions section is, there are selectable bullets that filter the questions based on their difficulty and their flags. You can also use the search option down bellow to search for a particular question. Also all these questionscan be sorted from A to Z or Z to A by using the Sort option in the Database->Sort Questions menu dialog. Also using the List that shows the question you can move up/down/to top/to bottom in the represented order the selected question using the Move Up/Down/to top/to bottom buttons bellow or using the Question Dialog or by pressing the appropriate shortcuts Ctrl +Up/Ctrl+Down/Ctrl+Shift+Up/Ctrl+Shift+Up.

Questions can also be hided. **Hidden questions** don't show in the list unless you chose to **show hidden questions** option in the Database menu dialog. In the Question dialog you can hide a question by ticking the **Hide** option. Thus the question will disappear(not deleted)from the list offering more space to view other questions.

Before configuring the server and start running sessions you must first save the database. There are several options here. You can just **Save** the database into the current location, **Save as** into something else or into a different folder, **Save a copy** of the database or **Save a backup** of the database. All of these options are available from the **File** menu dialog which also has the option of creating a **new** database or **open/close** an existing one.

Now we can set up a new test session using the server. From whatever panel click to the **Server Mode** Panel(next to the flags panel). There you see 2 data boxes **Available** and **Used** which are gray for the moment. To see more details check the **Advanced** checkbox. The **Available** label currently shows all the different flags you have created and the **Used** contains nothing. You can either show the questions themselves by selecting the **Questions bullet.** You can transfer some questions or whole flags into the **Used** section by clicking the right arrows that they are between or vice versa with the left arrow. These will be the questions that come along with the session test. If you select flags you can adjust the pass mark of the flag category itself.

If you want to customize more you can change the test name by checking the **Custom** checkbox next to the test name label and write a name.

Also you can specify if you want to set up 1 question per group. A group is a set of questions that can have different flags and difficulties. You can specify a Group for a question in the question Panel next to the flag dropdown box. The default value is empty.

You can modify the total questions and the pass mark by modifying the values next to the **Number of questions** label. Also you must set up the time limit for the test. You can set this up for the whole test or per one question only. Lastly if you want to change the default server port that opens to accept connections just modify the value next to the **Server Port** label. When you have done setting these up click the **Start Server** button in the right corner. This starts up the server mode.

Before the server starts a window appears to check the default printer settings. After you have configured the printer the server opens its port and waits to accept new connections from the clients. Note that **only in server mode can the clients connect to the server.** Thus as a safety measure you cannot close this window by normal means(you can close it with the help of an other program though) to ensure that the server is up and running. The window changes and shows some other panels. There is a label that shows the **Server information** and running port. There is a list of clients area in the left section where it records every clients name that have appeared to login in this session. Down there is the server log where it records every event such as Client logins, Client

disconnects ,Client submits results to server,Client prints results with each event having also a timestamp.

When a client submits its results they immediately appear in the selected client results where it shows all the test results and the status of the client. The user of this program can also have the option to export the session to a .itos file in order to be used from offline clients that have trouble connecting to the server. This is easily done from the **Server->Export** menu option.

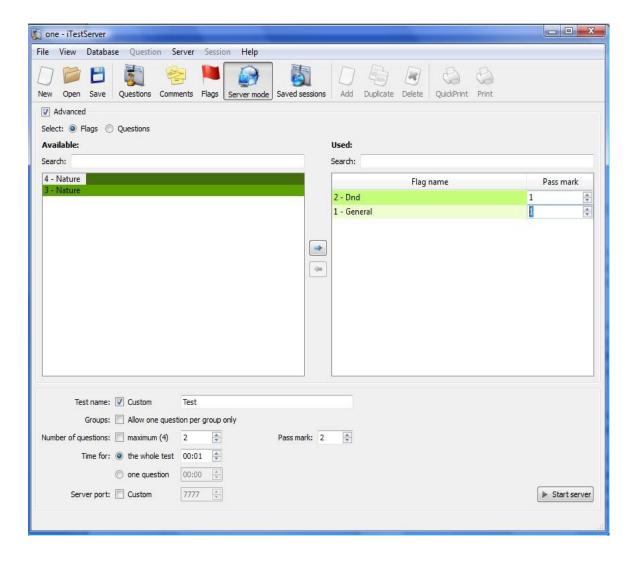


Figure 3.8 Server Mode Panel

When some clients finishes their tests then the user of the server can add one or more offline client logs (.itcl files)that are being created from the iTestClients. The server recognizes this files because they have specific format and loads the results into the screen. Note that there is no protection or encryption of these files. The teacher/users of the Server is mainly responsible for collecting these files before evil doers. You can add one or more offline client logs from the Server->Add Client/Add Clients menu option. Also you have the option to test yourself the session by clicking the Server->Run iTestClient option where you login as a new client. Every event that happens can be recorded and exported into a log file that have the HTML format. This is done with the option Server->Export Log.Later you can print this results with the quickprint or print options in the Server dialog.

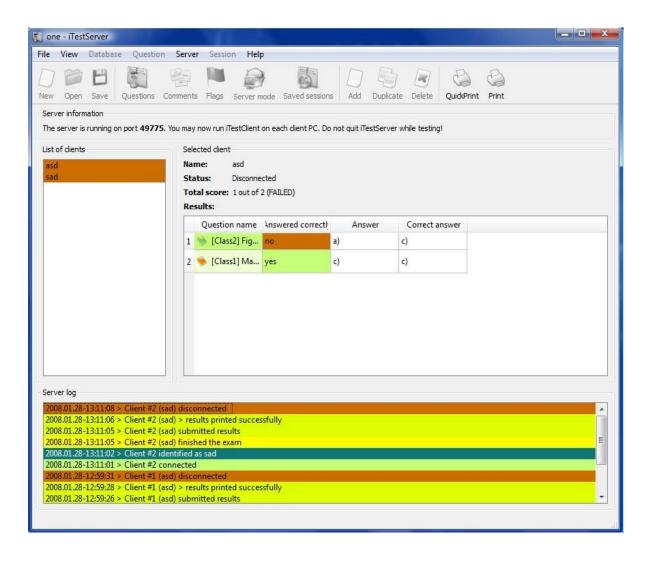


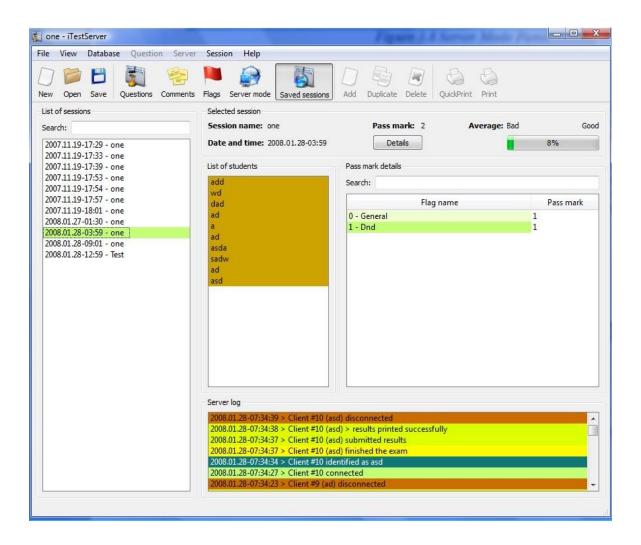
Figure 3.9 Server Mode Panel Server is on and Awaits connections

One important feature that iTestServer is the ability to analyze Saved sessions. Basically every time the server mode is on it saves all the activity of the sessions in the saved sessions panel.

You can access this panel through the **Saved sessions** panel which is located next to the **Server mode** panel. There you see a **list of sessions** section where it lists all the sessions that have been recorded from this database. Each session contains the **list of Students** section-where you see who had logged and took a test and what were his results-and the **server log** where you see the connection activity with the clients.

Every selected session also has a little bit more info such as a name and the Date which was created. The **Details** shows information(if any) about the pass mark and in the right there is a bar that shows visual representation of the success rate of the test. Basically it shows the percentage of the success ratio based on all questions taken and all questions answered correctly.

If you want to save a selected session for further study you can archive it. Simply click in the Session->Archive Session option or the Session->Copy to Archive and the selected session will be archived. You can access archived sessions in the Archived sessions list bellow the List of sessions. If you want to restore a session from archive by removing it from the list of Archived sessions click the Session->Restore from Archive or else if you want to copy and thus not removing it from the list click the Session->Copy from Archive option. In every other way you can print a selected session with the option Session->Print Session Summary or print all sessions in the list with the Session->Print All option.



#### Figure 3.10 Saved sessions panel

iTestsServer functionality doesn't end here. It offers a usefull automated statistics tool that comes in handy if you want to review questions based on how many times have been answered correctly and how many times not. You can access it in the **Database->Overall Statistics** option.

There you see a window that lists all the non hidden questions of the database, the current difficulty the calculated difficulty the number of correct answers and the number of incorrect answers. If a particular question has low ratio of correct answers to incorrect answers and vice verca then the adjust difficulty button right becomes clickable and you can adjust the difficulty of the question based on this algorithm:

```
if (Number of correct answers==0 && Number of incorrect answers==0) then difficulty =Medium:
```

else if (Number of correct answers==0 && Number of incorrect answers>0) then difficulty =Difficult;

else ratio= (Number of correct answers / Number of incorrect answers)

if  $(ratio \le 0.5)$  then difficulty=Easy

if (ratio > 0.5) && (ratio <= 2) then difficulty=Medium

*if* (ratio > 2) then difficulty=Difficult

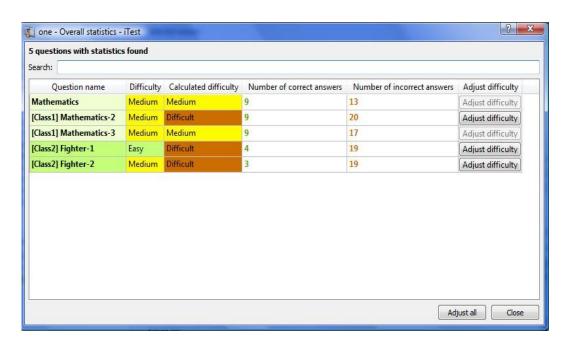


Figure 3.11 Overall Statistics Window

You can save time by clicking the **Adjust all** button where it Adjusts automatically all questions that are present. Also the adjust difficulty option is available in the Questions Panel just bellow the **Answers** section in a line that begins with **Statistics**. There by pressing the **Adjust difficulty** word automatically adjusts that question difficulty based on the previous algorithm.

Now we have concluded the overall behavior of the tools of the iTest server. These were all the available options that help organize the database.

# **4.**External Interface Requirements

#### 4.1 User Interfaces

Here you can see the iTestClient interface forms:

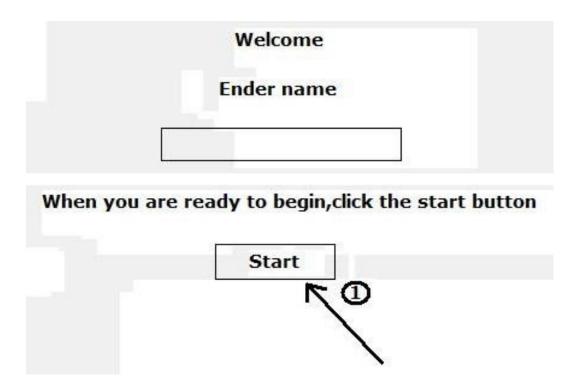


Fig 4.1 Client login name window Option 1 starts the test

In figure 4.2 you can see the first form that the iTestClient sees that manages the connection and test options.

Connect to server	ъ.		1	6
Server	Port			Connec
<ul> <li>Use Database File</li> </ul>				
File		Browse	Load	
Save answers to				
□ Default Path				
г Тв	rowse	Ī		
	101150	1		
Settings	200			
<ul><li>□ Hide question Nan</li><li>□ Do not show corre</li></ul>		worc.		
Do not snow corre	Ct alisv	wers		
Test Information				
Test Name				
Database date				
Test Date				
Test Time				
No. of Questions				
No. of Flags				
Flag names				
Pass mark				
Comments				
Quit About				Ready

Fig 4.2 Client start up Choice 1 exits Client and choice 2 Goes to Welcome Screen

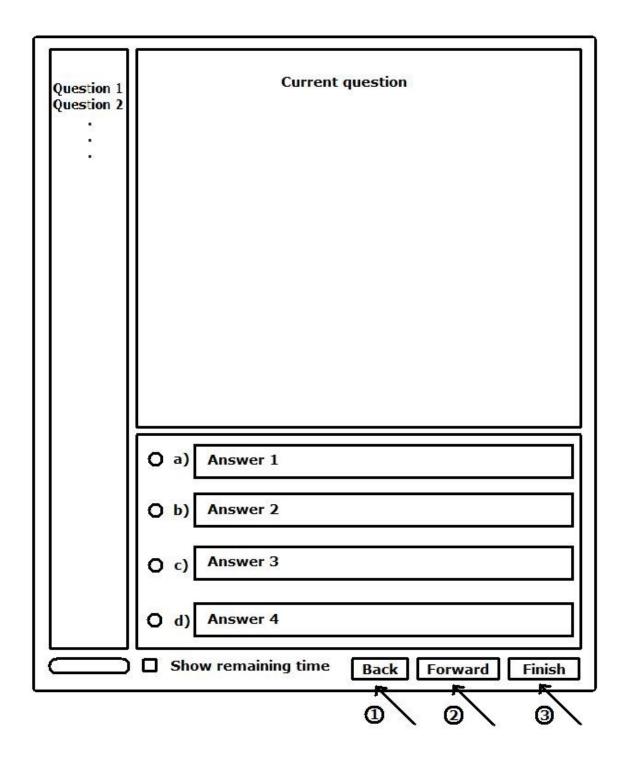


Fig 4.3 Test session screen. Option 1 goes 1 question back, option 2 goes 1 question forward and option 3 finishes the test.

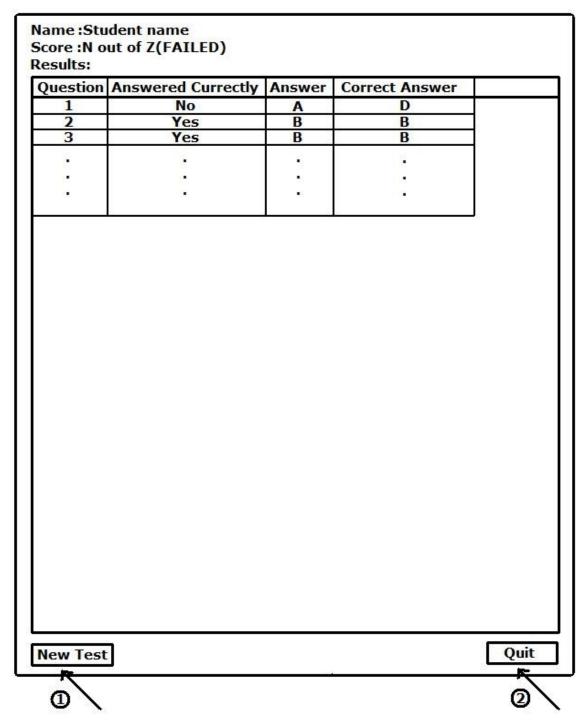


Fig 4.4 Option 1 goes to the iTest start up screen and option 2 Quits the iTestClient

#### 4.2 Hardware Interfaces

For the communication protocol the program needs these protocols to be installed:

- Tcp for the client to connect to the server in online mode.
- Storing devices(flash,optical disks etc.) for the client to take a test in offline mode.

Also in case that the iTestServer runs behind a firewall the appropriate ports must be port forwarded or port triggered for the clients to connect.

#### 4.3 Communications Interfaces

Setting up the server into server mode requires that there will be open ports for accepting connections from the clients. The connection between the client and the server uses Connection-oriented communication, via TCP/IP—Transfer Control Protocol/Internet Protocol, implements reliable delivery of messages. Connection-oriented communication makes programming easier because the protocol includes mechanisms for detecting and handling errors and an acknowledgment mechanism between client and service. Again we assume that the reader is not interested of how that is implemented into the program at this phase so there will be no more details of this now.

# **5.Other Nonfunctional Requirements**

Here we specify some nonfunctional constraints that the program satisfies in order to be more concrete and stable.

## **5.1 Performance Requirements**

**Performance:** checking the fact that the system must perform as what every user expects .So in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, In case of opening databases, sorting questions and computing there are no delays and the operation is performed in less than 2 seconds for opening, sorting, computing > 95% of the files. Also when connecting to the server the delay is based on the distance of the 2 systems and the configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds

## **5.2 Safety Requirements**

**Consistency:** checking the fact that all clients must be attachable to one server, so there would be appropriate control of the test statistics and information.

Also in case of a potential loss of connection between the client and the server the clients test progress so far is lost. When the client finishes its test(by pressing the finish button) then its progress is sent to the server and be logged. In case of a potential server breakdown only the so far finished tests are saved to the log file.

#### **5.3 Security Requirements**

This program uses object oriented mechanisms to protect its data passed using methods Also there is no currently a security schema of this program. Thus the log files that are being created are readable using a simple text reader.

## **5.4 Software Quality Attributes**

**Availability**: Checking that the system always has something to funtion and always pop up error messages in case of component failure. In that case the error messages appear when something goes wrong so to prevail availability problems.

**Usability**: Checking that the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

**Functionality:** Checking that the system provide the right tools for editing question databases, creating session tests and analyzing the test sessions. In that case the tools that the Database editor provide are the ones that provide that attribute.

# **Appendix A: Glossary**

Here we list all the terminology that is used throughout the document:

SRS:Software Requirements Specification

**Connects:**Links this requirement with another

**Includes:** Has the appropriate constraint in it

**Extends:** Shows or cancels a constraint effect if the conditions are met.

# **Appendix B: Analysis Models (Intro)**

Here we give an intro for the next phase of the specification that is the analysis of the requirements where we specify in analytical way all the modules and classes of the program.

Based on the previous requirements specification we can define the most useful attributes of the Questions.

#### **Object** Question

#### **Attributes:**

String name
int flag
String group
int difficulty
image attachments
String body(text and answers)
int correct answers
boolean status(hidden or not hidden)

Based on that there are also methods to set and get these attributes, constructors and destructors and methods to compute the difficulty.

Also there is a class named Student that stores the individual information of the client. Each student must have these Attributes:

#### **Object** Student

#### Attributes:

String Name boolean Score int number boolean Status(Ready or not) String Results boolean HasPassed

And has the appropriate methods of set and get, construction, destruction and saving these student data.

# **Appendix C: Issues List - ChangeLog**

Here is the program changelog up to the latest version

#### 5.5iTest 1.3.0

- added Spanish translation
- added SVG (scalable vector graphics) support, which means questions can now contain graphics
- added multiple correct answers support
- new database format older versions of iTest cannot open the new itdb 1.3 files, iTest 1.3 can still open older databases
- upgraded from Qt 4.3.1 to Qt 4.3.2

#### 5.5.1iTestServer (formerly Database Editor)

- added the ability to print questions and tests (with or without graphics)
- added search to the "overall statistics" dialogue
- made the ui more flexible
- many improvements in translations
- numerous bug-fixes and more...

#### 5.6iTest 1.2.0

#### **5.6.1iTestServer (formerly Database Editor)**

- renamed to iTestServer
- added Portuguese translation
- added the ability to set a pass mark for each flag separately
- this required some changes in the saved sessions view and in printing
- added the ability to group similar questions in order to be able to tell the client to choose one question of the group at most (useful when knowing the answer for one question would suffice to guess the answers for other questions)
- server setup (advanced): double clicking adds/removes items
- improved the "overall statistics" dialogue; question names now coloured according to their flag
- improved the "change language" dialogue
- improved the comments editor (undo/redo)
- changed flag 1 colour to light green
- row height now calculated correctly in the "overall statistics" dialogue
- fixed the "adjust difficulty" button now the icon in the list of questions changes when difficulty changed
- cleaned up the menus, improved their behaviour
  - code clean-up
  - new database format older versions of iTest cannot open the new itdb 1.2 files, iTest 1.2 can still open older databases
  - upgraded from Qt 4.3.0 to Qt 4.3.1
  - and more...

#### **5.6.2**iTestClient (formerly Test Writer)

- renamed to iTestClient
- added Portuguese translation
- added the ability to start a new test
- more advanced test generation, support for groups added
- upgraded from Qt 4.3.0 to Qt 4.3.1

#### 5.7iTest 1.1.1

#### 5.7.1Database Editor & Test Writer

- added Turkish translation
- if available, translation to the system language loaded by default
- NEW ENCODING: UTF-8 adds support for more languages and special characters
- iTest 1.1.1 can still open old CP 1250 databases from older versions of iTest
- older versions of iTest cannot open the new UTF-8 databases from iTest 1.1.1
- upgraded from Qt 4.2.2 to Qt 4.3.0

#### 5.7.2Database Editorprinting: support for the & sign

#### 5.8iTest 1.1.0

#### 5.8.1Database Editor

- added Russian translation
- removed the ability to delete a session
- added the ability to archive, restore and copy sessions
- added the ability to save a copy and a backup copy
- changes made to questions auto-applied when saving
- the selected session now green in the list of saved sessions
- decreased font size of printed exam results
- items in the list of students now green if passed and orange if failed
- warning when changing the name of the database or a question, occurrences in sessions and archived sessions auto-updated

#### 5.8.2Test Writer

- -added Russian translation
- -added the ability to hide question names