Manage Project Homework
Testing Document

Test phase

Alejandro Onatra Caro

Mónica María Lozano Romero

February 14th 2012

Table of contents

Test cases

Test results screenshots

1 Introduction

1.1 Purpose

This document contains and explains the testing process and the testing results of the MPH project development by one of the teams taking the course of software engineering two.

1.2 Test participants

1.2.1 Evaluation team

- Monica Maria Lozano Romero 780322
- Alejandro Onatra Caro 770643

1.2.2 Development team

- Mehrnoosh Askarpour 770385
- Ali Bahadory Bozchaloei 749922

1.3 Scope

This document will test the development of the MPH project by the assigned team, this testing will follow the approaches mentioned in the next chapter and studied during the lectures. The testing will include the following functional requirements identified by the development team on the RASD document:

Management of users:

[FR1] Access to profile.
[FR2] Login
[FR3] Logout
[FR4] Student registration
[FR5] Professor registration

Management of deliveries

[FR6] Upload file.
[FR7] Download file.
[FR8] Score checking.
[FR9] Use sharing.
[FR10] Evaluation
[FR11] Enable sharing[1]
[FR12] See files by type.

[1] As the team has only two integrants, this requirement wont be checked on the testing

Management of project teams

[FR13] See teams.
[FR14] See student list.
[FR15] See group list.

Management of projects

[FR16] Publish project

The non-functional requirements won't be tested given that there wasn't found any specification in the requirement, analysis and specification document.

1.4 Definition, acronyms, and abbreviations

The following acronyms will be used through the whole document:

- MPH: Manage project homework
- JUnit: Java unitary tests
- FR: Functional Requirements.
- NFR: Non-functional requirement.
- QA: Quality attribute.
- G: Goal.
- Test case: A functional requirement test.
- RASD: Requirement and analysis specification.

1.5 References

- ProjectMPHdescription.pdf
- mph-dn-en-bahadory-askarpour-planning.xlsx
- mph-dn-en-bahadory-askarpour-RASD.pdf
- mph-dn-en-bahadory-askarpour-designdocument.pdf
- mph-dn-en-bahadory-askarpour-testcase.pdf
- mph-dn-en-bahadory-askarpour-userguide.pdf
- Software formal inspections guidebook.pdf
- Java_checklist.pdf

1.6 Overview

This document specifies the testing process, approaches and results of the development of MPH project by the assigned team.

The document is organized in the following sections:

1. **Introduction**
   This section describes the purpose of the document including its scope, glossary and related documents used to do it.

2. **Test background**

   Contains a detail explanation of the testing environment, test cases and approaches taking into account for executing the development testing.

3. **Development validation**

   Contains a detail explanation of the analysis validation of the developed software based on the RASD document, design document, user manual and source code delivered on January 25th 2012.

4. **Development verification**

   Contains a detail explanation of the design test cases and the results obtained based on the outputs of the development and the project specification.

5. **Appendixes**

   Present a compilation of extra reference material for this document.

2 Test background

2.1 Test objectives

The present testing is for validating and verification the following goals identified by the tester group on the project specifications:

- **[G1]** For every project a professor (responsible) can find easily a delivery done in current and previous versions, i.e., for different dates.
- **[G2]** For the entire professors is easy and fast to share a delivery done by a project team to the other project teams.
- **[G3]** For every project a professor can classify the different types of project assignments, improving they search and accessibility.
- **[G4]** For the entire professors is easy to manage the execution parallels projects in different moments of time.
- **[G5]** For the entire professors is easy to manage and publish grades to each project team for a specific uploaded delivery assignment.
- **[G6]** For every project the final project grade for each project team is publish on time and with precise accuracy.
- **[G7]** For every assignment delivery a student can easily upload and update it.

2.2 Test environment information

Test environment information	
Software tested:	*Manage project Homework*
Type	Web application using JSP, servlets and EJBs
Version	1.0
Goal	The testing is for validating and verificating the development of the manage project homework software defined in the ProjectMPHdescription.pdf. The testing will be validated with the goals [G1] [G2] [G3] [G4] [G5] [G6] [G7]
Author	Alejandro Onatra, Mónica Lozano

2.3 Test tools

The test includes a project using the **JUnit framework** such that the functional requirements can be checked without the limitations of the web interface; this project includes the following source code and the functional requirements tested:

- *UserTest.java:*
  - Login.
  - Consult profile.
  - Logout.
- *UserStudentTest.java:*
  - Student registration
  - Student team creation
  - Student team subscription
  - Upload file
  - Score checking
- *UserProfessorTest.java*
  - Professor registration
  - Evaluation
  - Download file (by delivery type and team)
  - Consult teams for project
  - See teams
  - See student list
  - See group list
- *ProjectTest.java*
  - Publish project

Is important to mention that for each of functionality a set of test cases was executed, the test cases and it results are explained in the Development Verification chapter and in **¡Error! No se encuentra el origen de la referencia.** chapter.

2.4 Test – Requirement – Design relationship

2.4.1 User management

User profile access	
Test cases	- T-LM-AUP-001 - T-LN-AUP-001 - T-LN-UL-001
Functional requirements	[FR1]
Non-functional requirements	Doesn't apply

System components	**Web layer:** - StudentProfile.jsp
	Business layer: - Login Manager.
	Persistence layer: - User Entity
Relation explanation	When the user is logged in to the system, the first page displayed to the user is the profile page with the respective data of the logged user.

User login	
Test cases	- T-LM-UL-001 - T-LN-UL-001
Functional requirements	[FR2]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - Index.jsp - LoginServlet
	Business layer: - Login Manager.
	Persistence layer: - User Entity
Relation explanation	When the user fill the login form in the login page and sends it, the web tier calls the login manager to validate the information.

User logout	
Test cases	- T-LM-ULO-001 - T-LN-ULO-001
Functional requirements	[FR3]
Non-functional	Doesn't apply

requirements	
System components	**Web layer:** - LogOutServlet
	Business layer: - Login Manager.
	Persistence layer:
Relation explanation	When the user clicks on the logout link the system calls the login manager to erase the session information of the current user.

Student – user registration	
Test cases	- T-UN-RS-001 - T-UM-RS-001
Functional requirements	[FR4]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - StudentRegistrationServlet - student-registration.jsp
	Business layer: - StudentManager.
	Persistence layer: - Student
Relation explanation	When a new user wants to create a student profile, has to go to the register page, fill the information and send it. The student manager captures the information and sends it to the DB.

Professor – user registration	
Test cases	- T-AM-RP-001 - T-AN-RP-001 - T-AM-RPEE-002

	- T-AM-RPNEP-002
Functional requirements	[FR5]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - professor-registration.jsp - ProfessorRegistrationServlet
	Business layer: - AdminManager
	Persistence layer: - Professor
Relation explanation	When an admin user wants to create a user with the professor profile, has to go to log in, choose the register professor link, fill the information and send it. The admin manager captures the information and sends it to the DB.

2.4.2 Deliveries management

Upload file	
Test cases	- T-UM-UF-001 -
Functional requirements	[FR6]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - UploadFileServlet - UploadFile.jsp
	Business layer: - Doesn't apply
	Persistence layer: - Doesn't apply

Relation explanation	There's an upload page but cannot upload any file and the business tier doesn't have any functionality for these FR.

Download file	
Test cases	- T-UM-DF-001 - T-UM-DF-002
Functional requirements	[FR7]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - Doesn't apply
	Business layer: - Doesn't apply
	Persistence layer: - Doesn't apply
Relation explanation	There weren't any methods in any of the layers of the system to download a file by type or group.

Score checking	
Test cases	- T-UM-SC-001
Functional requirements	[FR8]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - PreScoreChekingServlet - ScoreChekingServlet - ScoreChekingServlet.jsp
	Business layer:

	- StudentManager
	Persistence layer: - Student - Project
Relation explanation	The student can got to the score checking web page after been logged and see the scores obtained per group and project.

Evaluation	
Test cases	- T-PM-E-001 - T-PM-E-001
Functional requirements	[FR10]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - SetScoreServlet - PreSetScoreServlet - SetScore.jsp
	Business layer: - ProfessorManager - UserSessionBean
	Persistence layer: - LevelFile
Relation explanation	The professor goes to the set score web page and selects the level file to be graded and fills the grading form, defining grade and penalty.

See files by type	
Test cases	- T-PN-SFT-001
Functional requirements	[FR12]

Non-functional requirements	Doesn't apply
System components	**Web layer:** - SeeFileServlet - PreSeeFilesServlet - SeeFiles.jsp
	Business layer: - Doesn't apply
	Persistence layer: - LevelFile
Relation explanation	The professor goes to the see files page and selects the type off file and project to see the list of LevelFile information about the selection.

2.4.3 Project teams management

See group list	
Test cases	- T-PN-SGL-001 - T-PM-SGL-001
Functional requirements	[FR15]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - PreSeeGroupsServlet - SeeGroups.jsp
	Business layer: - ProfessorManager
	Persistence layer: - Professor - Group
Relation explanation	The professor goes to the see groups page and can see the list of information about the groups that belong to the projects of the professor.

See student list	
Test cases	- T-PN-SSL-001 - T-PM-SSL-001
Functional requirements	[FR14]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - GroupManaging.jsp - PreSeeGroupsServlet - SeeMemberServlet
	Business layer: - ProfessorManager
	Persistence layer: - Professor - Student
Relation explanation	The professor goes to the see groups page and can see the list of information about the groups that belong to the projects of the professor. After that it can select the get members link to see the members of the group.

2.4.4 Project management

Publish project	
Test cases	- T-PM-PP-001 - T-PM-SSL-001
Functional requirements	[FR14]
Non-functional requirements	Doesn't apply
System components	**Web layer:** - CreateNewProject.jsp - PublishProject - CreateNewProjectServlet

	Business layer: - ProfessorManager
	Persistence layer: - Professor - Student
Relation explanation	The professor goes to the publish project link after logged in, fill the form for creating projects and then sends the information. The information is captured by the ProfessorManager to send it to the DB.

2.5 Aspects to be tested

Among the aspects to be tested besides the functional requirements mention the scope of the document (Scope) are the different documents created during the development process of the project by the development team, these documents also include the source code and the execution of the final software delivered on January 25th 2012.

The documents verified against the final software are:

- mph-dn-en-bahadory-askarpour-RASD.pdf
- mph-dn-en-bahadory-askarpour-designdocument.pdf
- mph-dn-en-bahadory-askarpour-testcase.pdf
- mph-dn-en-bahadory-askarpour-userguide.pdf

2.6 Approach

For validating and verifying the development project was used the analysis approach and the testing approach, in this way a complete view of the development is achieved. The depth of the verification and validation was limited by the testers available time, therefore the execution of the tests are oriented to the functional requirements specified on the project description document.

In the testing approach were executed black box and glass box tests, it means, the functional requirements were verified not only using automatic test tools (that allowed us to define the drivers, stubs and oracle needed) and executing the methods implemented by the development team in the business layer of the JEE architecture but also interacting with the web interface provided by the web client developed by the development team.

In the analysis approach was executed a static analysis on the RASD and design documents published by the development team against the source code delivery for the test phase.

2.7 Hardware and software requirements

2.7.1 Hardware requirements:

Hardware requirements
Processor
Core 2 duo
Primary memory
4GB
Secondary Memory
>=10 GB

Table 1 Hardware requirements

2.7.2 Software requirements:

For installing the software is necessary to have installed in the selected server the following items:

- Java virtual machine.
- The application server Jboss AG 5.1.
- The user will need only a compatible browser installed in his personal computer.

The tester should have also installed the software specified in the **¡Error! No se encuentra el origen de la referencia.**

Required software	
Database management server	
Name	MySQL
Mnemonic	MySQL
Specification number	Community edition
Version number	5.5.18
Source	http://www.mysql.com/downloads/
Development IDE	
Name	Eclipse Indigo JEE

Mnemonic	ECLIPSE
Specification number	3.7
Version number	3.7.1
Source	http://www.eclipse.org/downloads/
	JUnit Framework
Name	JUnit Framework
Mnemonic	JUnit
Specification number	4
Version number	4
Source	http://www.junit.org/

Table 2 Software requirements

3 Development Validation

This chapter contains the validation process and results done over the software development assigned. In the last chapter of the document the most important results are summarized.

3.1 Validation procedure

The validation of the development was based on the software formal inspections guidebook from the office of safety and mission assurance of NASA. Therefore, during the validation were review the requirement and analysis specification document, design document and the source, the detailed list is summarized in the next subsections.

3.1.1 Specification and design documents

- *mph-dn-en-bahadory-askarpour-RASD.pdf*
- *mph-dn-en-bahadory-askarpour-designdocument.pdf*

3.1.2 Source code

- Business logic file list
- Web file list

3.2 Requirements and Analysis specification document

In this section are summarized the findings of the validation review on the last version of the RASD document of the assigned development.

RASD Validation	
Clarity	• The system goals are not defined. • The terminology used is not consistent with the standards, for instance the use of the terms profile, project level and home page. • The functional and non-functional requirements of the system weren't specified in any section of the document. • The description of conceptual model is incomplete and ambiguous. For instance project level, file type, profile, among others. • There aren't hardware environment and software environment specifications for the software product. • For the different requirements there is no a clear explanation about all their possible inputs and outputs.
Completeness	• Not all the constraints for the program are listed, for instance the file transmission during an upload of a homework assignment can be done locally in the server or in a remote server. Additionally some requirement weren't taken into account during the use case specification, such as the download files by type or by group for a professor.

	• No priority was assigned to the different functional requirements. • Among all the functional requirements specified in the use cases there isn't a summary related to the ones that were going to be implemented. • No user specifications related to the educational level of the users and they possible training for using the application. • There isn't a technology, development and runtime environment specification.
Compliance	• The document doesn't compliant with a RASD document standard.
Consistency	• The functional requirement related to the download of file is not consistent with the project specifications, it should be only be done by professor users. • There are functional requirements that are not consistent with the project specification, such as the consultation of the different teams by a student user.
Correctness	• The requirements specified are feasible with respect to the schedule and technology. The developers are new with the JEE technology then this could impact the project development.
Data usage	• No specification or identification of file system requirements on the server side for accomplishing the most critical part of the project. • No specification or identification of the file transmission of the project homework assignments when uploaded or downloaded by a user of the system. Moreover, there isn't a specification on the accepted file formats. • No specification about the data integrity constraints related to the user passwords and student homework assignment deliveries, such that the system reliability is guarantee. Additionally
Functionality	• There isn't a complete covering of the system functionality by the use cases specified. For instance there are not use cases related to the final calculation of a grade, the subscription to a team, and download all the files delivered by a team. • There isn't a clear specification of the relationships among the entities identify on the analysis class diagram (the related sequence diagrams are too general). Moreover, the conceptual model doesn't satisfy all the system requirements, for instance a student can belongs only to a group on the system and as the group can be in several projects the student will need to work in every project with the same work team.
Interface	• There aren't user interface mockups for the different users, such that

	they have an idea of the possible difficulties during the development of the user interface of the system. • External interfaces (software and hardware) weren't studied and specified in the document.
Level of detail	• The use case flow diagrams don't show any detail related to the relationships among the conceptual entities identified during the requirements and analysis specification, so that the design phase can start straightforward. • The use cases are defined too general and doesn't allow to identify all the possible interactions between the system and the user during the execution of the involved use case. • There isn't a good understanding of the functional requirements.
Maintainability	• There isn't a specification on the maintenance requirements such as file system backups, database backups, and user management, among others. • There aren't maintainability requirement specifications such as the management of new requirements, or requirements modifications, among others.
Performance	There aren't performance requirement specifications over the system.
Reliability	There aren't reliability requirement specifications over the system, such as minimal quality measures.
Testability	All the use cases specified can be testable.
Traceability	All the use cases specified can be traceable, but the functional requirements cannot be traceable.

3.3 Design document

In this section are summarized the findings of the validation review on the last version of the design document of the assigned development.

Design document Validation	
Clarity	• There isn't a defined architecture of the system that shows the main components of the system such as their functional responsibilities and interconnections. A JEE architecture was assume as the system architecture but the document doesn't show how this architecture is used in the application domain, such as the needed session beans, the type of client application (web or stand alone), the needed entity beans, among others.

	• There isn't any explanation related to the web client subcomponents such as the set of pages needed, what functionalities they support, which kind of users can use them, among others. Moreover, what kind of technology was going to be used on the web client implementation and how it interconnects with the business layer of the JEE architecture. • As the system architecture is missing and only the business logic components are specified, such managers are oriented by user services and not as system services. In this way there is not a clear functionality division among the defined business logic manager. • There isn't detail specifications of the services that a manager offers such as it preconditions, it exit exceptions, the user constraints, among others.
Completeness	• The components definition on the business logic doesn't support any anticipated changes on the requirements, because such changes should involved user functionalities in other case it will be difficult to select among the proposed managers given that they are user oriented and not service oriented. • There isn't any description related to design decisions, assumptions or constraints of the system.
Compliance	The document doesn't compliant with a design document standard.
Consistency	• The last version of the design document is inconsistent with the developed components. • There are sequence diagrams that don't reflect the real system behavior. Actually some of the diagrams don't satisfy the architecture constraints, for instance there are sequence diagrams where entity components execute business logic, or components that executes user actions such as link selections. • The entities beans implemented are not consistent with the design document specifications. There are new attributes or new relationships among the proposed entities. For instance the type attribute type in project entity and project level entity, and the relationship between project entity and student entity.
Correctness	• The current design can be achieved with the schedule given if the developers have good level of expertise in web development technologies otherwise the learning curve will take parts of the available development time.
Data usage	• The database design is not as detailed as require for this phase, given that it stills in a conceptual model and not in a logical model.

	• Some entities were changed during implementation, for instance the project entity and the project level entity. • There isn't any detail design related to the file system management or the file transmission when uploading or downloading homework assignments.
Functionality	• The managers designed cover all functional requirements specified on the project specifications, but some of them are inconsistent with the RASD document. For instance, the final grade of a group in a project is not specified as a use case on the RASD document but it is included in the group manager.
Interface	• There aren't descriptions for the different methods offered by the managers such as their functionality and behavior can be easy to understand. • There isn't the design of the different user interfaces (web pages) such that its functionality and behavior can be easy to understand. • There isn't the web technology specification such as JSP, Java server faces, servlets, among others; such that during the development phase can be taken into account the time need in all the environment configurations and the learning curve of the developers.
Level of detail	• The design is not as detailed as need for the development phase.
Maintainability	• There isn't a specification on the maintenance requirements such as file system backups, database backups, and user management, among others. • The maintainability of the system is highly compromise given that its components are not well defined and have high coupling level.
Performance	There aren't any performance design specifications over the system.
Reliability	There aren't reliability design specifications over the system
Testability	All the components specified can be testable.
Traceability	The functionalities designed cannot be traceable given that there isn't a detail explanation of the methods offered by the different managers and there isn't any web client design. Moreover, the design has some inconsistencies with the RASD document (as mentioned before).

3.4 Source code

In this section are summarized the findings of the validation review on the last version of the source code of the assigned development.

Source Validation	
Clarity	• There source code is not well commented, such as possible assumptions inside a method, special validations and business logic, can be understood without checking the code in detail. • The methods and interfaces are not well commented, such that the method behavior can be known without checking the code. Moreover there isn't a javadoc of the services offered by the session beans. • The entities name is not clear for the application domain. For instance the project level entity and the level file. • The variables names on the servlets are not self descriptive. For instance, variables called ref1, ref2, ref3, lm or pm doesn't give any hint about their type (can be a manager, entity or a POJO). • The use of exceptions is not correct, given that on the source code there is a java class inheriting from Exception for every kind of exception thrown by the system, actually none of them does messaging management for reporting the exception description.
Compliance	The code doesn't follow well known coding standards, such as variable naming, code indentation, among others.
Data usage	• There aren't basic validations when entities are retrieved from the database; for instance if the searched entity was actually found. • The relationship project-student appears on the entity implementation, which violates the functional description of the system analysis and system description. • The database queries are hardcoded in the methods; this is classified, as a bad practice given that for doing modifications is mandatory to understand in witch one is the related method and the class where it belongs. • Some queries return a set of results but on the code is taken as a single result. • The methods have incomplete business logic, such as validations basics for the system. For instance in the grade assignment the grade cannot be grater than ten and lesser that one.
Functionality	• Several methods on the business layer are not completely implemented. • Several methods aren't call in any part of the code or can be tested. • The upload and download files are missing in the responsible managers. • There is repeated code in the different classes of the project (in EJB source code and in web client servlets). For instance, the look up of the session beans is repeated in each servlet.

	• Some methods are not located in the correct manager, such as getting the list of whole projects in the system is in the student manager.
Maintenance	• As the development doesn't follow strictly the JEE framework, the development maintenance is very difficult given that the services are not correctly distributed among the selected session beans, and the web client use JSP technology in this way the line between the presentation layer and the business logic is not very highlighted.

4 Development Verification

4.1 Functional requirements testing

In this section we verify that the functional requirements are correct according to the analysis and design document presented. For this we checked the functionalities using the business logic tier and in case there's no implementation available in this tier, we tested using the web tier if possible.

4.1.1 Login manager

4.1.1.1 User login

User Login	
Code	T-LM-UL-001
Purpose	Check that a client can login with an existent username and password.
Components	The managers used in this test case are the following: ▪ LoginManager
Inputs 1	neocloud, 1234
Inputs 2	password, name
Inputs 3	admin, admin
Expected output 1	The number "1" identifying that the user exists and it's of type student.
Expected output 2	The number "2" identifying that the user exists and it's of type professor.
Expected output 3	The number "3" identifying that the user exists and it's of type administrator.
Current output 1	Number "1"
Current output 2	Number "2"
Current output 3	Number "3"
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ User exists in the database. The following was checked: ▪ Log in the user if the password and username are correct.
Result	The user was recognized

Observations	The data validation is performed inside the web pages
Incidents	When test with null parameters, the component recognize it as valid ones.

Test case 1 User login

User Login with invalid parameters	
Code	T-LM-UL-002
Purpose	Check that a client can't login with an invalid username and password.
Components	The managers used in this test case are the following: ▪ LoginManager
Inputs	User1, password1
Expected output	The component sends an error message.
Current output	The system sends an exception.
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ User exists in the database. The following was checked: ▪ Log in the user if the password and username are correct.
Result	The system sends and error message.
Observations	The data validation is performed inside the web pages
Incidents	

Test case 2: User Login with invalid parameters

4.1.1.2 User logout

User Logout	
Code	T-LM-ULO-001
Purpose	Check that a client can logout from the system
Components	The managers used in this test case are the following: ▪ LoginManager
Inputs	*Empty*
Expected output	Successful logout
Current output	Successful logout
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ User is log in into the database. The following was checked: ▪ Log in the user if the password and username are correct.
Result	The user was correctly log out of the system.
Observations	

Incidents	

Test case 3: user logout

4.1.1.3 Access profile

Access user profile	
Code	T-LM-AUP-001
Purpose	Check that a user can check his/her profile.
Components	The managers used in this test case are the following: ▪ LoginManager
Inputs	User.
Expected output	The Profile of the input User.
Current output	The Profile of the input User.
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ User exists in the database. ▪ User profile exists in the database The following was checked: ▪ Log in the user if the password and username are correct.
Result	The profile was correctly obtained.
Observations	The data validation is performed inside the web pages
Incidents	When used a null user the manager doesn't catch the exception.

Test case 4: check profile

4.1.2 Student manager

4.1.2.1 Student registration

Register a student	
Code	T-UM-RS-001
Purpose	Check that a user can be register into the system with the student profile
Components	The managers used in this test case are the following: ▪ StudentManager The entities used in this test are the following: ▪ Student ▪ Profile
Inputs	name, surname, email@surname.com, photo, password
Expected output	The creation of a student with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = email@surname.com

	▪ Photo = photo The creation of a profile with the following parameters: ▪ Username = email@surname.com ▪ Password = password
Current output	The student and profile parameters are the same as the expected output.
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ User exists in the database. ▪ User profile exists in the database The following was checked: ▪ Log in the user if the password and username are correct.
Result	The test was successful
Observations	The data validation is performed inside the web pages.
Incidents	When used null parameters the manager doesn't recognize it as an invalid input.

Test case 5: Registering a student

4.1.2.2 Create group

Create a group	
Code	T-UM-CG-001
Purpose	Check that a student can create a group of 3 persons.
Components	The managers used in this test case are the following: ▪ StudentManager The entities used in this test are the following: ▪ Student ▪ Project
Inputs	student1, student2, student3, project
Expected output	The creation of a group with the associated students and project.
Current output	Creation of a group with the associated students and project.
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ Users exist in the database. ▪ There is an available project in the system The following was checked: ▪ The group is correctly created and associated to the project
Result	The test was successful
Observations	The data validation is performed inside the web pages.
Incidents	When used null parameters the manager doesn't recognize it as an invalid

	input, and for example, register a null valued project.

Test case 6: Create a group

4.1.2.3 Upload file

Upload file	
Code	T-UM-UF-001
Purpose	Check that a student can upload a file
Components	The components used in this test case are the following: ▪ UploadFileServlet ▪ UploadFile.jsp
Inputs	project, projectLevel, aa.zip, Student3, Project
Expected output	Confirmation of the successful upload of the file
Current output	The web component doesn't upload the file.
Execution environment	The following was used or required for the test: ▪ JUnit testing environment ▪ Users exist in the database. ▪ There is an available project in the system ▪ There is an available project level To check this functionality, only servlets and the web pages were used.
Result	Uploading a file to the system was not possible
Observations	Given that there wasn't any EJB method to analyse with JUnit regarding the upload of a file into the server system, the test was performed entirely with a browser.
Incidents	The system didn't show the project levels of the selected project and neither uploads the file when the path was specified. When press enter at the moment of uploading a file, an error was shown.

Test case 7: Upload file

4.1.2.4 Score checking

Score checking	
Code	T-UM-SC-001
Purpose	Check that a student can check the score obtained by the group given an specific project.
Components	The managers used in this test case are the following: ▪ StudentManager The entities used in this test are the following: ▪ Student

	▪ Project
Inputs	Student1
Expected output	Score obtained by the student.
Current output	Score obtained by the student.
Execution environment	
Result	The test was successful
Observations	The components assume that the group is contained already inside the user; in fact the function only takes the user and uses the object methods to find the score.
Incidents	When used null parameters the system doesn't recognize it as an invalid input but let the system crash when reach a null pointer exception.

Test case 8: Score checking

4.1.3 Professor manager

4.1.3.1 See student list

See student list	
Code	T-PM-SSL-001
Purpose	Check that a user with the professor profile can obtain the list of the students that have projects with him.
Components	The managers used in this test case are the following: ▪ ProfessorManager The entities used in this test are the following: ▪ Professor ▪ Student
Inputs	professor1 with id 9
Expected output	The id´s of the student list retrieve should be: ▪ 1,2,3
Current output	The current output were a list of students with id´s: 1,2,3.
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The professor exists and has a project. ▪ There is a group registered in the project previously mentioned. The following was checked: ▪ The list of the retrieve ids is coherent with the database information.
Result	The id´s were retrieved correctly.

Observations	
Incidents	The system doesn't capture the null pointer exception when the professor is null.

Test case 9: See student list

4.1.3.2 See group list

See group list	
Code	T-PM-SGL-001
Purpose	Check that a user with the professor profile can obtain the list of the groups that have projects with him.
Components	The managers used in this test case are the following: ▪ ProfessorManager The entities used in this test are the following: ▪ Professor ▪ Group
Inputs	professor1 with id 9
Expected output	The id´s of the group list retrieve should be: ▪ 1
Current output	The current output were a list of groups with id´s: 1.
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The professor exists and has a project. ▪ There is a group registered in the project previously mentioned. The following was checked: ▪ The list of the retrieve ids is coherent with the database information.
Result	The id´s were retrieved correctly.
Observations	The user with profile student id is linked to the id of a defined project, as well as the group, this restricts the system so, a user can only participate in only one project, and moreover, it's unnecessary to link a group to a project given that its participants are restricted to one and only one project.
Incidents	The system doesn't capture the null pointer exception when the professor is null.

Test case 10: See group list

4.1.3.3 Publish project

Publish project	
Code	T-PM-PP-001
Purpose	Check that a user with the professor profile can create a project

Components	The managers used in this test case are the following: ▪ ProfessorManager The entities used in this test are the following: ▪ Professor ▪ Project
Inputs	professor1 with id 12, project_name, 2014-12-10
Expected output	The creation of a project with the following parameters: ▪ Professor = professor1 ▪ Project name = project_name ▪ Deadline date = 2014-12-10
Current output	The current output is consistent with the expected one.
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The professor exists. The following was checked: ▪ The project was correctly created.
Result	The project was created successfully
Observations	The data validation is performed inside the web pages
Incidents	The manager doesn't capture the null pointer exception when the professor or the other data is null.

Test case 11: publish project

4.1.3.4 Download files by type

Download file by type	
Code	T-UM-DF-001
Purpose	Check that a professor can download the allowed files in the system by type.
Components	
Inputs	
Expected output	
Current output	
Execution environment	
Result	
Observations	
Incidents	There weren't any methods in any of the layers of the system to download a file by type.

Test case 12: Download file by type

4.1.3.5 Download files by group

Download file by group

Code	T-UM-DF-002
Purpose	Check that a professor can download the allowed files in the system by group.
Components	
Inputs	
Expected output	
Current output	
Execution environment	
Result	
Observations	
Incidents	There weren't any methods in any of the layers of the system to download a file by group.

Test case 13: Download file by group

4.1.3.6 Evaluation

Evaluation	
Code	T-PM-E-001
Purpose	Check that a user with the professor profile can evaluate a project correctly.
Components	The managers used in this test case are the following: ▪ ProfessorManager ▪ UserSessionBean The entities used in this test are the following: ▪ LevelFile
Inputs	LevelFile with id 1, 18
Expected output	Assign score 18 to the LevelFile with id 1.
Current output	The current output is consistent with the expected one.
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The LevelFile exist The following was checked: ▪ The score was correctly assigned to the LevelFile selected
Result	The score was successfully assigned
Observations	Through the GUI it wasn't possible to create the LevelFile but in the component UserSessionBean it's implemented.
Incidents	The manager doesn't capture the null pointer exception when the LevelFile is null.

Test case 14: Evaluation

4.1.4 Administrator manager

4.1.4.1 Register professor

Register professor	
Code	T- AM-RP-001
Purpose	Check that a user can be register into the system with the professor profile
Components	The managers used in this test case are the following: ▪ AdminManager The entities used in this test are the following: ▪ Professor ▪ Profile
Inputs	name, surname, email@surname.com, photo, password
Expected output	The creation of a professor with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = email@surname.com ▪ Photo = photo The creation of a profile with the following parameters: ▪ Username = email@surname.com ▪ Password = password
Current output	The professor parameters are the same as the expected output. The profile parameters are: ▪ Username = password ▪ Password = email@surname.com
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The administrator is registered in the system The following was checked: ▪ Professor not null and according to the creation parameters
Result	The system doesn't capture correctly the input
Observations	
Incidents	The test presents that at the moment of capturing the data for the profile the username and the password are in the inverse order.

Test case 15: Register professor

Register professor with email error	
Code	T- AM-RPEE-002

Purpose	Check that a user cannot be register into the system with the professor profile, using an invalid email structure
Components	The managers used in this test case are the following: ▪ AdminManager The entities used in this test are the following: ▪ Professor ▪ Profile
Inputs	name, surname, 1234, photo, password
Expected output	The creation of a professor with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = 1234 ▪ Photo = photo The creation of a profile with the following parameters: ▪ Username = email@surname.com ▪ Password = password
Current output	The professor parameters are the same as the expected output. The profile parameters are: ▪ Username = password ▪ Password = 1234
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The administrator is registered in the system The following was checked: ▪ Professor not null and according to the creation parameters
Result	The system doesn't identify the email structure.
Observations	The data validation is performed inside the web pages
Incidents	The test shows that the function doesn't identify if the structure of the email it's correct or not.

Test case 16: Register professor with email error

Register professor with null email and password	
Code	T-AM-RPNEP-002
Purpose	Check that a user cannot be register into the system with the null email and password.
Components	The managers used in this test case are the following: ▪ AdminManager The entities used in this test are the following: ▪ Professor

	▪ Profile
Inputs	name, surname, null, photo, null
Expected output	The creation of a professor with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = null ▪ Photo = photo The creation of a profile with the following parameters: ▪ Username =null ▪ Password = null
Current output	The professor parameters are the same as the expected output. The profile parameters are the same as the expected output.
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment ▪ The administrator is registered in the system The following was checked: ▪ Professor not null and according to the creation parameters
Result	The test shows that the function doesn't identify if any of the parameters are null.
Observations	The data validation is performed inside the web pages
Incidents	The test shows that the function doesn't identify if any of the parameters are null.

Test case 17: Register professor with null email and password

4.1.5 Group manager

4.1.5.1 Calculate total grade

Calculate total grade	
Code	T-GM-CG-001
Purpose	Check that a user can check the total grade of a group.
Components	The managers used in this test case are the following: ▪ GroupManager The entities used in this test are the following: ▪ Group
Inputs	group1 with id = 1
Expected output	The total grade = 18
Current output	
Execution environment	The following was used or required for the test: ▪ JUnit testing enviroment

	▪ The group is registered in the system and has at least one LevelFile The following was checked: ▪ The total grade is the sum of the grades of all the LevelFiles of the group.
Result	The score match the expected output
Observations	The data validation is performed inside the web pages
Incidents	The manager doesn't capture the null pointer exception when the LevelFile is null.

Test case 18: Calculate total grade

4.1.5.2 Calculate penalty

Calculate penalty	
Code	T-GM-CP-001
Purpose	Check that the system calculates the penalty of a LevelFile
Components	
Inputs	
Expected output	
Current output	
Execution environment	
Result	
Observations	This test case was planned according to the document of design and in this document this method was proposed.
Incidents	The method was implemented in neither the business logic nor the web tier.

Test case 19: Calculate penalty

4.1.5.3 Calculate total penalty

Calculate total penalty	
Code	T-GM-CP-001
Purpose	
Components	
Inputs	
Expected output	
Current output	
Execution environment	

Result	
Observations	This test case was planned according to the document of design and in this document this method was proposed.
Incidents	The method was implemented in neither the business logic nor the web tier.

Test case 20: Calculate total penalty

4.2 GUI (Graphic user interface) testing

In this section we verify that the navigability of the web tier satisfies the functional requirements and therefore the goals of the system.

4.2.1 Login navigation

4.2.1.1 User login

User Login	
Code	T-LN-UL-001
Purpose	Check that a client can login with an existent username and password and is redirected from the login page to the user menu page.
Components	The web components used in this test case are the following: ▪ Index ▪ LoginServlet
Inputs 1	neocloud, 1234
Inputs 2	password, name
Inputs 3	admin, admin
Expected output 1	Navigate to the student home menu
Expected output 2	Navigate to the professor home menu
Expected output 3	Navigate to the administrator home menu
Current output 1	Reach the student home menu
Current output 2	Reach the student home menu
Current output 3	Reach the student home menu
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ User exists in the database. The following was checked: ▪ Log in correctly and display of the user home menu.
Result	Display the user home menu
Observations	
Incidents	

4.2.1.2 User logout

User Logout	
Code	T-LN-ULO-001
Purpose	Check that a client can logout from the system
Components	The web components used in this test case are the following: ▪ LogOutServlet
Inputs	Click on the home button
Expected output	Successful logout
Current output	Successful logout
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ User exists in the database. The following was checked: ▪ Log out correctly.
Result	The user was correctly log out of the system.
Observations	The way of login out is misleading because in the breadcrumbs the “home” link doesn’t take you to the home menu but log you out of the system.
Incidents	

Test case 21: user logout

4.2.1.3 Access profile

Access user profile	
Code	T-LN-AUP-001
Purpose	Check that a user can check his/her profile.
Components	The web components used in this test case are the following: ▪ StudentProfile.jsp ▪ LoginServlet
Inputs	Log in to the system as a student with username: neocloud
Expected output	Show the user neocloud profile information
Current output	The Profile information of the user neocloud.
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ User exists in the database. The following was checked: ▪ Show the profile information of the user correctly.
Result	The profile was correctly shown.
Observations	The profile information is shown immediately after the login and is used as a

	home menu.
Incidents	

Test case 22: check profile

4.2.2 Student navigation

4.2.2.1 Student registration

Register a student	
Code	T-UN-RS-001
Purpose	Check that a user can be register into the system with the student profile.
Components	The managers used in this test case are the following: ▪ student-registration.jsp ▪ StudentRegistrationServlet
Inputs	name, surname, email, photo, password,1234
Expected output	The creation of a student with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = email ▪ StudentID = 1234 ▪ Password = password The creation of a profile with the following parameters: ▪ Username = email ▪ Password = password
Current output	The creation of a student with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = email ▪ StudentID = 1234 ▪ Password = password The creation of a profile with the following parameters: ▪ Username = password ▪ Password = email
Execution environment	The following was used or required for the test: ▪ JBoss server The following was checked: ▪ Register correctly in the system as a student
Result	
Observations	The data validation is performed by the web pages but doesn't specify the error to the user.
Incidents	The system allows repeated students id's

Test case 23: Registering a student

4.2.2.2 Create group

Create a group	
Code	T-UN-CG-001
Purpose	Check that a student can create a group of 3 persons from the web page.
Components	The managers used in this test case are the following: ▪ PreCreateGroupServlet ▪ CreateGroupServlet ▪ Creategroup.jsp
Inputs	Information of the students with the user id: 1,2,3
Expected output	Creation of a group with the following members id: 1,2,3
Current output	Creation of a group with the following members id: 1,2,3
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ Three users exist in the database. ▪ There is an available project in the system The following was checked: ▪ The group is correctly created and associated to the project
Result	The test was successful
Observations	When a user create a group, by default he is registered to the group if no other students are added, but there is no possible way to add a new student to the group
Incidents	

Test case 24: Create a group

4.2.2.3 Upload file

Upload file	
Code	T-UN-UF-001
Purpose	Check that a student can upload a file
Components	The components used in this test case are the following: ▪ UploadFileServlet ▪ UploadFile.jsp
Inputs	aa.zip
Expected output	Confirmation of the successful upload of the file
Current output	The web component doesn't upload the file.
Execution	The following was used or required for the test:

environment	▪ JBoss server ▪ Users exist in the database. ▪ There is an available project in the system ▪ There is an available project level To check this functionality, only servlets and the web pages were used.
Result	Uploading a file to the system was not possible
Observations	Given that there wasn't any EJB method to analyse with JUnit regarding the upload of a file into the server system, the test was performed entirely with a browser.
Incidents	The system didn't show the project levels of the selected project and neither uploads the file when the path was specified. When press enter at the moment of uploading a file, an error was shown.

Test case 25: Upload file

4.2.2.4 Score checking

Score checking	
Code	T-UN-SC-001
Purpose	Check that a student can check the score obtained by the group given a specific project.
Components	The components used in this test case are the following: ▪ PreScoreChekingServlet ▪ ScoreChekingServlet ▪ ScoreChekingServlet.jsp
Inputs	Click on the link for score checking
Expected output	Score obtained by the student per LevelFile name and score: rasd->18, rasd2->21
Current output	Score obtained by the student per LevelFile name and score: rasd->18, rasd2->21
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ Available LevelFiles for the user. The following was checked: ▪ The score in the LevelFiles is correct.
Result	The test was successful
Observations	
Incidents	When try to navigate from the page ScoreChecking.jsp to UploadFile.jsp the application throws a page not found exception, because the name was misspelled.

Test case 26: Score checking

4.2.3 Professor navigation

4.2.3.1 See student list

See student list	
Code	T-PN-SSL-001
Purpose	Check that a user with the professor profile can obtain the list of the students that have projects with him.
Components	The components used in this test case are the following: ▪ GroupManaging.jsp ▪ PreSeeGroupsServlet ▪ SeeMemberServlet
Inputs	Click in the group managing link and then in the get members link in the group with id=1.
Expected output	List the information of the students composing the group with ids: 1, 2 and 3.
Current output	List the information of the students composing the group with ids: 1, 2 and 3.
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ At least one student registered in a group belonging to the professor logged in. The following was checked: ▪ The score in the LevelFiles is correct.
Result	The id´s were retrieved correctly.
Observations	
Incidents	The system doesn't capture the null pointer exception when the professor is null.

Test case 27: See student list

4.2.3.2 See group list

See group list	
Code	T-PN-SGL-001
Purpose	Check that a user with the professor profile can obtain the list of the groups that have projects with him.
Components	The components used in this test case are the following: ▪ PreSeeGroupsServlet ▪ SeeGroups.jsp
Inputs	Click on the managing groups link while logged in with the professor with id=9
Expected output	The id´s of the group list retrieve should be: 1, 2 and 3.
Current output	The id´s of the group list retrieve is: 1, 2 and 3.
Execution	The following was used or required for the test:

environment	▪ JBoss server ▪ At least there's one group registered to one project of the professor. The following was checked: ▪ The group list belongs to one project published by the professor.
Result	The id´s were retrieved correctly.
Observations	
Incidents	

Test case 28: See group list

4.2.3.3 Publish project

Publish project	
Code	T-PN-PP-001
Purpose	Check that a user with the professor profile can create a project
Components	The components used in this test case are the following: ▪ CreateNewProject.jsp ▪ PublishProject ▪ CreateNewProjectServlet
Inputs	Project with name=project3 and dead line=2012-09-08
Expected output	The creation of a project with the following parameters: ▪ Project = project3 ▪ Deadline = 2012-09-08
Current output	The current output is consistent with the expected one.
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ Logged in as a professor The following was checked: ▪ Correct creation of the project with the input data.
Result	The project was created successfully
Observations	After creating the project there's an option to create ProjectLevels if this is not done immediately, there is no way to do it later; furthermore a project can be created without ProjectLevels.
Incidents	

Test case 29: publish project

4.2.3.4 Download files by type

Download file by type	
Code	T-UN-DF-001
Purpose	Check that a professor can download the allowed files in the system by type.

Components	
Inputs	
Expected output	
Current output	
Execution environment	
Result	
Observations	
Incidents	There weren't any methods in any of the layers of the system to download a file by type.

Test case 30: Download file by type

4.2.3.5 Download files by group

Download file by group	
Code	T-UN-DF-002
Purpose	Check that a professor can download the allowed files in the system by group.
Components	
Inputs	
Expected output	
Current output	
Execution environment	
Result	
Observations	
Incidents	There weren't any methods in any of the layers of the system to download a file by group.

Test case 31: Download file by group

4.2.3.6 See files by type

See files by type	
Code	T-PN-SFT-001
Purpose	Check that a user with the professor profile can check the see some information about the project deliveries by type.
Components	The components used in this test case are the following: ▪ SeeFileServlet ▪ PreSeeFilesServlet ▪ SeeFiles.jsp
Inputs	Click on the link to see files and select RASD type of document for the project 1
Expected output	See all the LevelFile information for one project from type RASD

Current output	See all the LevelFile information for one project from type RASD
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ Logged in as a professor ▪ A project belong to the professor and exist at least one LevelFile related to a group. The following was checked: ▪ See the list of LevelFIles data relevant to the type selected.
Result	The list was correctly displayed.
Observations	
Incidents	If the project is changed, the current files don't change after the first ones displayed.

4.2.3.7 Evaluation

Evaluation	
Code	T-PN-E-001
Purpose	Check that a user with the professor profile can evaluate a project correctly, using a score and a penalty.
Components	The components used in this test case are the following: ▪ SetScoreServlet ▪ PreSetScoreServlet ▪ SetScore.jsp
Inputs	18, 2
Expected output	Assign score 18 to the LevelFile with id 1.
Current output	The current output is consistent with the expected one.
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ Logged in as a professor ▪ A project belongs to the professor and exist at least one LevelFile related to a group. The following was checked: ▪ Correct grading of the project.
Result	The score was successfully assigned
Observations	Through the GUI it wasn't possible to create the LevelFile but in the component UserSessionBean it's implemented.
Incidents	

Test case 32: Evaluation

4.2.4 Administrator navigation

4.2.4.1 Register professor

Register professor	
Code	T-AN-RP-001
Purpose	Check that a user can be register into the system with the professor profile
Components	The components used in this test case are the following: ▪ professor-registration.jsp ▪ ProfessorRegistrationServlet
Inputs	name, surname, aaaa@aaaa.com, 1111
Expected output	The creation of a professor with the following parameters: ▪ Name = name ▪ Surname = surname ▪ Email = aaaa@aaaa.com The creation of a profile with the following parameters: ▪ Username = aaaa@aaaa.com ▪ Password = 1111
Current output	The professor parameters are the same as the expected output. The profile parameters are: ▪ Username = 1111 ▪ Password = aaaa@aaaa.com
Execution environment	The following was used or required for the test: ▪ JBoss server ▪ Logged in as a administrator The following was checked: ▪ Correct registering of the professor.
Result	The system doesn't capture correctly the input
Observations	The test presents some problems in the way of storing the email and the password.
Incidents	The test presents that at the moment of capturing the data for the profile the username and the password are in the inverse order.

Test case 33: Register professor

4.2.5 Group manager

This section cannot be tested with the GUI given that the functionalities related to it do not have an implementation that interacts directly with the user.

4.3 Incidence report

The next table presents a description of the test cases and its incidences and reports.

Table of incidences		
Functional requirements testing		
Test case code	Observation	Incidence
T-LM-UL-001	The data validation is performed inside the web pages	When test with null parameters, the component recognize it as valid ones.
T-LM-UL-002	The data validation is performed inside the web pages	
T-LM-AUP-001	The data validation is performed inside the web pages	When used a null user the manager doesn't catch the exception.
T-UM-RS-001	The data validation is performed inside the web pages.	When used null parameters the manager doesn't recognize it as an invalid input.
T-UM-CG-001	The data validation is performed inside the web pages.	When used null parameters the manager doesn't recognize it as an invalid input, and for example, register a null valued project.
T-UM-UF-001	Given that there wasn't any EJB method to analyse with JUnit regarding the upload of a file into the server system, the test was performed entirely with a browser.	- The system didn't show the project levels of the selected project and neither uploads the file when the path was specified. -When press enter at the moment of uploading a file, an error was shown.
T-UM-SC-001	The components assume that the group is contained already inside the user; in fact the function only takes the user and uses the object methods to find the score.	When used null parameters the system doesn't recognize it as an invalid input but let the system crash when reach a null pointer exception.
T-PM-SSL-001		The system doesn't capture the null pointer exception when the professor is null.

T-PM-SGL-001	The user with profile student id is linked to the id of a defined project, as well as the group, this restricts the system so, a user can only participate in only one project, and moreover, it's unnecessary to link a group to a project given that its participants are restricted to one and only one project.	The system doesn't capture the null pointer exception when the professor is null.
T-PM-PP-001	The data validation is performed inside the web pages	The manager doesn't capture the null pointer exception when the professor or the other data is null.
T-UM-DF-001		There weren't any methods in any of the layers of the system to download a file by type.
T-UM-DF-002		There weren't any methods in any of the layers of the system to download a file by group.
T-PM-E-001	Through the GUI it wasn't possible to create the LevelFile but in the component UserSessionBean it's implemented.	The manager doesn't capture the null pointer exception when the LevelFile is null.
T-AM-RP-001		The test presents that at the moment of capturing the data for the profile the username and the password are in the inverse order.
T- AM-RPEE-002	The data validation is performed inside the web pages	The test shows that the function doesn't identify if the structure of the email it's correct or not.
T- AM-RPNEP-	The data validation is	The test shows that the function doesn't

002	performed inside the web pages	identify if any of the parameters are null.
T-GM-CG-001	This test case was planned according to the document of design and in this document this method was proposed.	The method was implemented in neither the business logic nor the web tier.
T-GM-CP-001	This test case was planned according to the document of design and in this document this method was proposed.	The method was implemented in neither the business logic nor the web tier.
GUI navigation testing		
T-LN-ULO-001	The way of login out is misleading because in the breadcrumbs the “home” link doesn’t take you to the home menu but log you out of the system.	
T-LN-AUP-001	The profile information is shown immediately after the login and is used as a home menu.	
T-UN-RS-001	The data validation is performed by the web pages but doesn’t specify the error to the user.	The system allows repeated students id’s
T-UN-CG-001	When a user create a group, by default he is registered to the group if no other students are added, but there is no possible way to add a new student to the group	
T-PN-SFT-001		If the project is changed, the current files don’t change after the first ones displayed.
T-UN-SC-001		When try to navigate from the page

		ScoreChecking.jsp to UploadFile.jsp the application throws a page not found exception, because the name was misspelled.
T-PN-SSL-001		The system doesn't capture the null pointer exception when the professor is null.
T-PN-PP-001	After creating the project there's an option to create ProjectLevels if this is not done immediately, there is no way to do it later; furthermore a project can be created without ProjectLevels.	
T-PN-E-001	Through the GUI it wasn't possible to create the LevelFile but in the component UserSessionBean it's implemented.	
T-AN-RP-001	The test presents some problems in the way of storing the email and the password.	The test presents that at the moment of capturing the data for the profile the username and the password are in the inverse order.

Table 3: Incidence report table

5 Appendix

5.1 List of files reviewed:

5.1.1 Business logic file list

./controllers:

AdminManager.java
AdminManagerRemote.java
GroupManager.java
GroupManagerRemote.java
LoginManager.java
LoginManagerRemote.java
ProfessorManager.java
ProfessorManagerRemote.java
StudentManager.java
StudentManagerRemote.java

./entities:

Admin.java
Group.java
LevelFile.java
Professor.java
Profile.java
Project.java
ProjectLevel.java
Student.java
User.java

./exception:

DuplicateLevelException.java
DuplicateProjectException.java
DuplicateUserException.java
GroupFullException.java
NoInforException.java
NoUserException.java
NoUserFoundException.java

./session:

UserSessionBean.java
UserSessionBeanRemote.java

5.1.2 Web file list

./src/test:

AddNewLevelServlet.java
CreateGroupServlet.java
PreSeeGroupsServlet.java
PreSetScoreServlet.java

CreateNewProjectServlet.java

EvaluationServlet.java

LogOutServlet.java

LoginServlet.java

PeSeeFilesServlet.java

PreCreateGroupServlet.java

PreEvaluatianServlet.java

PreGroupManagingServlet.java

PreScoreChekingServlet.java

PreSeeFilesServlet.java

PreUploadFileServlet.java

ProfessorRegistrationServlet.java

ScoreCheckingServlet.java

SeeFileServlet.java

SeeGroupFileServlet.java

SeeMemberServlet.java

SetScoreServlet.java

StudentRegistrationServlet.java

UploadFileServlet.java

./WebContent:

AddNewLevel.jsp

AdminProfile.jsp

CreateNewProject.jsp

Creategroup.jsp

Evaluation.jsp

GroupManaging.jsp

Index.jsp

META-INF

professor-registration.jsp

student-registration.jsp

styles.css

styles1.css

SeeGroups.jsp

SeeMembers.jsp

SetScore.jsp

StudentProfile.jsp

UpdateExisting.jsp

UploadFile.jsp

WEB-INF

ProfessorProfile.jsp

PublishProject.jsp

ScoreChecking.jsp

SeeFiles.jsp

SeeGroupFiles.jsp

5.2 Testing Screenshots

5.2.1 Login manager screenshots

5.2.2 Student manager screenshots

Figure 1: The upload page, but doesn't show the project level list in spite of project2 having to project levels.

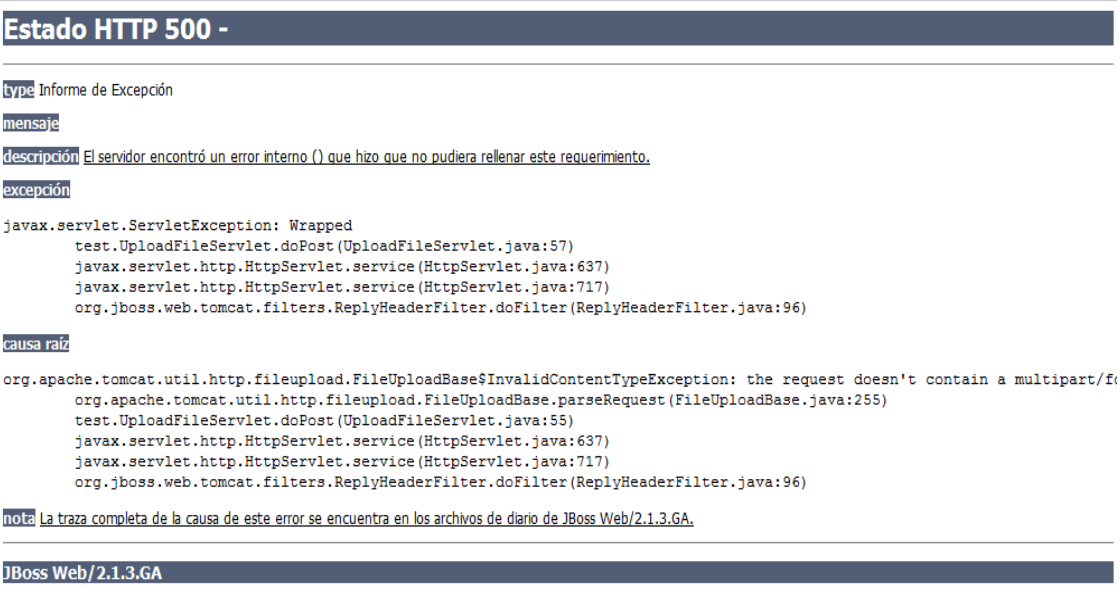

Figure 2: This error appeared when press enter in the download web page.

	id	score	project_id
▶	1	0	NULL
*	NULL	NULL	NULL

Figure 3: In the JUnit tests, a group can be created without a project because all the checking is done in the web tier.

	id	deadline	name	professor_id
▶	124	NULL	NULL	NULL
	125	2012-09-09 00:00:00	project1	9
	126	2012-09-09 00:00:00	project2	9
*	NULL	NULL	NULL	NULL

Figure 4: In the JUnit tests, a Project can be created with null parameters.

	id	deadline	name	type	project_id
▶	1	2012-09-08 00:00:00	level1	RASD	126
*	NULL	NULL	NULL	NULL	NULL

Figure 5: The Project with null parameters can have an associated ProjectLevel with a valid group LevelFile

Estado HTTP 404 - /testWebClient/uploasFile.jsp

type Informe de estado

mensaje /testWebClient/uploasFile.jsp

descripción El recurso requerido (/testWebClient/uploasFile.jsp) no está disponible.

JBoss Web/2.1.3.GA

Figure 6: Name error in a link to the web page: UploadFile.jsp

5.2.3 Professor manager screenshots

5.2.4 Administrator manager screenshots

	pr_id	password	username
	6	name@sumame.com	password
	7	name@sumame.com	password
	8	name@sumame.com	password
	9	1234	password
	10	name@sumame.com	password
	11	name@sumame.com	password
	12	NULL	password
	13	name@sumame.com	NULL
	14	NULL	password
	15	NULL	NULL
*	NULL	NULL	NULL

Figure 7: In the JUnit tests, a professor can be registered with null password and username, furthermore the registration process mix the password and email.