AAMS
(Automatic Assignment Management System)

Group 21

Ajanth Thangavelu
Roberto Castañeda Lozano
Tony Karlsson
Love Jädergård

REQUIREMENTS AND USE CASES

USER REQUIREMENTS

1. Definition and management of the courses

1.1. Creation of a course

1.1.1. Creation of a course

The system shall be able to provide a mean of creating a course which will contain the list of students and all the necessary information about assignments. There can exist many different courses at the same time.

Rationale: This is needed because there can exist assignments from a range of courses and students associated with different courses. This will help in organizing the courses and the homeworks.

1.2. Importing students list

1.2.1. Importing students list

There shall be a way to import a list of students into a created course.

Rationale: The amount of students attending a course can be very large, adding all of these students by hand into the system can be very time consuming. Adding a way of importing a list of students will help.

1.3. Defining an assignment

1.3.1. Defining an assignment

The system shall provide a mean of defining an assignment that will be associated with an existing course. There shall be a way to assign rules to each defined assignment. The rules that can be defined are the number of files attached in the e-mail, the format of these files and the expected e-mail subject associated with the assignment.

Rationale: Since there can be many assignments in a course there need to be a way of representing this in the system.

2. Collection and organization of assignments

2.1. Importing assignments

2.1.1. Input from students

The system shall allow students to submit their assignments by e-mail.

Rationale: e-mail is a widely spread and convenient way of communication. All students registered on a course should have access to a computer with e-mail capability. Most universities provide computers for the students to use at school.

2.1.2. Downloading assignments

The system shall download only those e-mails that relate to defined assignments.

Rationale: e-mails that do not relate to an assignment can be handled using the teachers normal e-mail client.

2.1.3. Identifying e-mails

The system shall use the e-mail subject line to decide if an e-mail is related to an assignment or not. The system should specify a format that e-mail containing assignments must follow.

Rationale: e-mail subjects can be downloaded from the e-mail server without downloading the whole message. The subject line is a convenient way for students to specify that an e-mail is related to an assignment.

2.1.4. Manual handling of assignments

The system shall provide a way to manually import assignments from students.

Rationale: Sometimes students will submit their work not using e-mail. There might be a good reason for this so the system must have some flexibility.

2.2. Organization of the assignments

2.2.1. Deleting assignment

The system shall provide a method to delete assignments that have been already stored in it.

Rationale: a teacher can be, for some reason, interested in removing the assignment of a specific student, so this can be asked to send the assignment again.

2.2.2. Organizing assignments

Downloaded e-mail with attachments (assignments) shall be organized into a folder hierarchy.

Rationale: A simple folder hierarchy is a convenient way to organize files on a computer. It is convenient to create a backup on CD-R from a folder hierarchy.

3. Editing and grading of assignments

3.1 Opening files

3.1.1. Opening files

The user should be able to open any documents fetched by the system. The user shall inform the people sending documents what file format to use.

Rationale: To be able to open any kind of documents the system should not put any constrains on specific file formats. There will be problem if the user receives a document which the users system does not have support for, thus informing solves the problem.

3.2 Commenting files

3.2.1. Commenting files

The user shall be able to make comments on any documents fetched by the system.

Rationale: Commenting documents excludes specific comments inside the document itself. Instead, the comments will be stored and sent as an email.

3.3. Grading assignments

3.3.1. Grading assignments

The system shall provide a mean of grading the assignments associated with the students.

Rationale: Since homework assignments usually are graded this needs to be represented in the system as well.

3.3.2. Product requirement

The grading format shall be general enough to accept grades in a very different formats.

Rationale: There are a lot of different grading formats in different countries, and the system should not be limited to a specific one.

4. Answer to the assignments

4.1. Delivery confirmation

4.1.1. OK confirmation

The system shall notify the sender of an assignment of the reception of this. The system shall automatically send an e-mail to the sender of the assignment as a delivery confirmation at the moment his assignment is received by the system.

Rationale: Delivery confirmations help the students that have sent their assignments to verify that these have really arrived to their destination, and can be used as a receipt in case there is some problem with the delivery.

4.1.2. Error notification

The system shall notify the sender of an assignment of possible errors when the e-mail does not fit the expected pattern. In this case, the system shall send automatically an e-mail to the sender of the assignment warning him that the received e-mail either does not contain the expected number of attached files or the format of some of these is not the expected one.

Rationale: This basic error notification can save a lot of work to the user of the system, that otherwise should notify the errors manually, and gives a chance to the sender to realize early of his error and fix them.

4.1.3. Product requirement

The delivery confirmation and error notification e-mails shall be encoded as plain text, using the standard UTF-8.

4.2. Sending feedback to students

4.2.1. Sending feedback to students

The system shall provide a method to send feedback to students by e-mail.

Rationale: Students submit their assignments by e-mail. It is reasonable to send feedback using the same way of communication.

4.2.2. Composition of feedback.

The system shall automatically compose the feedback e-mails. The feedback will be composed from teachers comments and grading.

Rationale: The information is already in the system. If the teacher would have to manually compose feedback e-mails it would be unnecessarily time consuming, and a possible source of errors.

4.2.3. When will feedback be sent.

The system shall only send feedback on request from user.

Rationale: There is no way for the system to tell if a user has finished work on an assignment.

5. Reporting

5.1. General state of a specific course

5.1.1. General state of a specific course

The user shall be able to retrieve a general status of a specific course.

Rationale: A general status would give the user an overview of the course and some statical data which could be used in conjunction with evaluation of the course.

5.2. State of a specific student

5.2.1. State of a specific student

The user shall be able to retrieve the status of a specific student attending a course.

Rationale: Status of a student is needed to justify a students grade at the end of the course.

5.3. State of a specific assignment

5.3.1. State of a specific assignment

The user shall be able to retrieve the status of specific assignments.

Rationale: The user needs this functionality to manage assignments at an individual level.

6. Res system link

6.1. Importing from the Res system

6.1.1. List of students import

The system shall be able to create a list with the needed data of all the students registered in a course from the Res system. The system shall connect to the Res system, obtain the data from the students that are registered in the desired course, and create a file with this information that, finally, can be used to define a course.

Rationale: In a large course, with tens or even hundreds of students, the task of introducing manually their names, e-mails, etc. can be so tedious that it could even discourage the potential users of the system. As in almost every university there is an internal system as Res where teachers can extract the list of students from, it is important to provide a way to automate this task and coordinate both systems.

6.1.2. Product requirement

The imported list of students must be stored as an XML (Extensible Markup Language) file that can be directly recognized and used in the definition of a course.

6.2. Exporting to the Res system

6.2.1. List of grades export

The system shall be able to update the grades of the students in the Res system with the information obtained from a specific course.

Rationale: It is essential to avoid the need to repeat the task of manually setting all the assignment grades twice, for the same reason as given in the requirement 6.1.1.

6.2.2. Product requirement

The exported list of grades must be stored as an XML file.

7. General non-functional requirements

7.1. Product requirements

7.1.1.

The files created by the system shall be implemented using XML.

7.1.2.

The system shall compile and run, at least, in the next operative systems: Windows XP, GNU/Linux (Ubuntu 7.x distribution) and FreeBSD 6.x.

7.2. Organizational requirements

7.2.1.

A first stable version of the system shall be ready to use in April 2008.

7.3. External requirements

7.3.1.

The system shall be licensed under a GPL (General Public License).

USE CASES

1. Definition and management of the courses

1.1. Creation of a course

Primary Actor: Teacher
Stakeholders and interests:
Teacher - to successfully create a course in the system.
Precondition: None
Minimal Guarantee: None
Success Guarantee: The course in mind is created.
Trigger: The user trigger this process.

Main Success Scenario:

1. The teacher tells the system to create a course.
2. The teacher gets to name the course.
3. The system checks so that the course name is valid.
4. The course is created.

Extensions:

2a. The course creation is aborted.
2a1. No course is created.
3a. The name for the course is invalid.
3a1. No course is created.

1.2. Importing students list

Primary Actor: Teacher
Stakeholders and interests:
Teacher - to successfully import the student list
Precondition: A course have been created
Minimal Guarantee: None
Success Guarantee: The list of students is imported.
Trigger: The user trigger this process.

Main Success Scenario:

1. The teacher tells the system to import a list of students.
2. The teacher selects which course the students in the list should be added to.
3. The teacher selects the location of the list of students.
4. The system checks if the list of students is in a valid format.
5. The students in the list are added to the selected course.

Extensions:

3a. The location of the list of students is invalid.
3a1. The importing of the student list is aborted.
4a. The format of the list of students is invalid.
4a1. The importing of the student list is aborted.

1.3. Defining an assignment

Primary Actor: Teacher
Stakeholders and interests:
Teacher - to successfully define an assignment in the system
Precondition: A course have been created
Minimal Guarantee: None
Success Guarantee: The assignment in mind is defined.
Trigger: The user trigger this process.

Main Success Scenario:

1. The teacher tells the system to define an assignment.
2. The teacher gets to select a existing course.
3. The teacher gets to name the assignment.
4. The system checks so that the assignment name is valid.
5. The teacher gets the option to set up rules about the assignment.

6. The system adds the assignment to each student in the course.

Extensions:

2a. The teacher aborts the defining of an assignment.

2a1. The defining of an assignment is aborted.

2. Collection and organization of assignments

2.1. Import assignments manually

Primary Actor: Teacher

Level: Summary

Stakeholders and Interests:

Teacher – to receive students homework assignments.

Student – to submit homework assignment.

University – to see that all guidelines are followed.

Precondition: At least one course defined with students and assignments.

Minimal Guarantees: An assignment is marked.

Success Guarantees: An assignment is marked.

Trigger: Teacher selects “Import assignment”.

Main Success Scenario:

1.Teacher selects “Import assignment”.

2. Teacher selects student.

3. Assignment is marked.

4. Teacher selects files to import.

1. Files are copied to students assignment directory.

Extensions:

4a. There are no files to import

2a1. Terminate import process.

2.2. Send feedback to student by e-mail

Primary Actor: Teacher

Level: Summary

Stakeholders and Interests:

Teacher – to send feedback on student homework assignment.

Student – to receive feedback on homework assignment.

University – to see that all guidelines are followed.

Precondition:

A selected student.

A selected assignment with comments and a grade.

Connection to the Internet.

E-mail account with smtp support.

Minimal Guarantees:

An attempt to connect to the e-mail account.

A e-mail is composed and saved for later sending.

Success Guarantees: A e-mail is composed and sent to the student.

Trigger: Teacher selects “send feedback”.

Main Success Scenario:

1. Teacher selects “send feedback”.

2. A e-mail message is composed from comments and grade.

3. A connection to the e-mail account is established.

4. The e-mail message is sent.

Extensions:

3a. A connection to the e-mail account could not be established.

2a1. Save message for later sending and terminate process.

3. Edition and grading of assignments

3.1. Opening files

Primary actor: Teacher

Stakeholders and interests:

-Teacher: Needs to review the assignment, which is in the file.

Preconditions: The filename must be specified.

Minimal guarantee: None
Success guarantee: The specified file opens
Trigger: The user triggers the process

Main success scenario:
1. The teacher choose one of the documents attached in the email.
2. The system reads the file format of the document.
3. The system executes the appropriate application to view the file.

Extensions:
3a. The file format is not a known to the users system.
3a1. The system informs the user of the problem.

3.2. Commenting files

Primary actor: Teacher
Stakeholders and interests:
-Teacher: Wants to give his or her opinion of the documents.
-Student: Wants to get his or her documents reviewed.
Preconditions: The email must be opened.
Minimal guarantee: None
Success guarantee: The document is commented.
Trigger: The user triggers the process

Main success scenario:
1. The teacher has a document opened.
2. The teacher has read through the document and comments it.
3. The system sends away the comment through email to the student.
4. The system saves the attached documents, email sender and assignment name.

Extensions:
3a. There is no comments to a document.
3a1. The system won't send away an email, instead continue to step 4.

3.3. Grading assignments

Primary Actor: Teacher
Stakeholders and interests:
Teacher - to grade the assignment that belong to the student in mind
Precondition: A course have been created. An assignment have been created. The student who is going to be graded have to exist in the system.
Minimal Guarantee: None
Success Guarantee: The student's assignment is graded.
Trigger: The user trigger this process.

Main Success Scenario:
1. The teacher decides to grade an assignment.
2. The teacher gets to select a course.
3. The teacher gets to select a student.
4. The teacher gets to select an assignment.
5. The teacher gets to input the grade.
6. The system checks that the grade is valid.
7. The assignment is graded.

Extensions:
2a. The user aborts the process.
3a. The user aborts the process.
4a. The user aborts the process.
6a. The grade is invalid:

4. Answer to the assignments

4.1. Send an assignment

Primary actor: Student
Stakeholders and Interests:

– Student: to ensure that his assignment will be received.
– Teacher: to ensure that all assignments are correctly built before reviewing them.

Preconditions:
– The student knows the e-mail address where he must send his assignment.
– The student knows the expected format (number of files and type of these) of the e-mail.

Success Guarantees:
– System registers the assignment of the student.
– Student receives a receipt confirming that the assignment has been registered.

Main success scenario:
1.Student sends his assignment (e-mail with some attached files) to Teacher's e- mail address via his usual e-mail client.
2 System verifies that the e-mail corresponds to some defined assignment.
3 System verifies that the e-mail comes from some defined student.
4 System verifies that the e-mail respects the expected format.
5 System verifies that the e-mail has been sent in time.
6 System stores the assignment and replies Student with a receipt confirming that the assignment has been registered.

Extensions:
2a The e-mail cannot be identified as a defined assignment delivery:
2a1 The e-mail is ignored by System and no reply is given to Student.
3a The sender cannot be identified as a registered student:
3a1 The e-mail is ignored by System and a warning e-mail is sent to Student.
4a The number of attached files in the e-mail or the expected format of these does not fit the expected pattern:
4a1 The e-mail is ignored by System and a warning e-mail is sent to Student.
5a The e-mail has been sent after the set deadline for the assignment:
5a1 A warning e-mail is sent to Student, the delay is registered by System and the sequence continues.

4.2. Import assignments from e-mail

Primary Actor: Teacher
Level: Summary
Stakeholders and Interests:
Teacher – to receive students homework assignments.
Student – to submit homework assignment.
University – to see that all guidelines are followed.
Precondition:
At least one course defined with students and assignments.
Connection to the Internet.
E-mail account with POP3 support.
Minimal Guarantees: An attempt to connect to the e-mail account.
Success Guarantees: A connection to the e-mail account is established and the account is checked for unread messages.
Trigger: Teacher selects "Update".
Main Success Scenario:
1. Teacher selects "Update".
2. A connection to the e-mail account is established.
3. The account is checked for unread messages.
4. Headers for unread messages are validated.
5. Messages with valid header are downloaded.
6. Downloaded messages and attachments are copied to students assignment directories.
7. Assignments are marked.
8. Confirmation e-mails are sent to students.

Extensions:

2a. A connection to the e-mail account could not be established.
2a1. Notify teacher and terminate import process.
3a. No unread messages in the e-mail account
3a.1 Terminate import process.
4a. Header is not valid.
4a.1 Mark header as non valid.
4a.2 Continue validating headers.
8a A message does not have the expected files attached
8a.1 A error report is appended to the confirmation e-mail
6a1. The process is aborted.

5. Reporting

5.1. General state of a specific course

Primary actor: Teacher
Stakeholders and interests:
-Teacher:Wants to read relevant information of an ongoing course.
Preconditions: None
Minimal guarantee: None
Success guarantee: The information of the specified course is retrieved.
Trigger: The user triggers the process.

Main success scenario:
1. The teacher specifies a course code or name.
2. The system verifies if the course exist in the database.
3. The system retrieves the information and displays it for the user.
Extensions:
2a. The system fails to find the course in the database.
2a1. The system notifies the user of the problem.

5.2. State of a specific student

Primary actor: Teacher
Stakeholders and interests:
-Teacher:Wants to read relevant information about an student.
Preconditions: None
Minimal guarantee: None
Success guarantee: The information about the specified student is retrieved.
Trigger: The user triggers the process

Main success scenario:
1. The teacher specifies a student name or code.
2. The system verifies if the student exist in the database.
3. The system retrieves the information and displays it for the user.
Extensions:
2a. The system fails to find the student in the database.
2a1. The system notifies the user of the problem

5.3. State of a specific assignment

Primary actor: Teacher:
Stakeholders and interests:
-Teacher:Wants to read relevant information about an assignment.
Preconditions: None
Minimal guarantee: None
Success guarantee: The information about the specified assignment is retrieved.
Trigger:The user triggers the process

Main success scenario:
1. The teacher specifies an assignment name or code.
2. The system verifies if the assignment exist in the database.
3. The system retrieves the information and displays it for the user.
Extensions:

2a. The system fails to find the assignment in the database.
2a1. The system notifies the user of the problem.

6. Res-system communication "plug-in"

6.1. Import list of students of a course from the Res system

Primary actor: Teacher

Stakeholders and Interests:

– Teacher: to get the updated list of students for his course, so he does not need
to introduce it manually in System.

– Res system managers: to ensure that the information stored in the Res system
remains consistent after the operation.

Preconditions:

– The course have been previously defined in the RES system and the students are registered there.

– Teacher has authorization to access to the RES system.

Success Guarantees:

– Teacher gets a file with a list of information about students that have registered n the RES system for the desired course.

Main success scenario:

1. Teacher starts the application that communicates with the RES system and provides the needed information about the course whose students list he wants to get.
2 The application gets the information from the RES system and creates a file with the list of students that can be later imported by System.

Extensions:

1a The RES system does not contain the required course:
1a1 The application warns the user and asks for the information about the course again.
1a2 Teacher supplies the needed information.

6.2. Export grades of a course to the Res system

Primary actor: Teacher

Stakeholders and Interests:

– Teacher: to copy the grades from System to the RES system, so he does not need to duplicate them manually.

– Res system managers: to ensure that the information stored in the Res System remains consistent after the operation.

Preconditions:

– The course have been previously defined in the RES system and the students are registered there.

– The graded assignments have been previously defined in the RES system.

– Teacher has authorization to access to the RES system.

Success Guarantees:

– The application stores the grades information, exported from System to a file, in the RES system.

Main success scenario:

1 Teacher starts System and opens a created course.
2 Teacher tells System to export the grades of the course to a file and specifies the file name and path.
3 System creates the file with all the necessary information in order to be able to store it in the RES system.
4 Teacher starts the application that communicates with the RES system, providing the grades file to it.
5 The application stores all the information about grades that the file contained in the RES system.

Extensions:

2a System detects that there are not assigned grades for the opened course:

2a1 System warns Teacher and aborts the export.
4a The RES system does not contain the course specified in the given file:

4a1 The application warns Teacher and aborts the process.

SYSTEM REQUIREMENTS

1. Definition and management of the courses

1.1. Creation of a course

Function: Creation of a course
Description: Create a course in the system.
Inputs: A course name.
Source: Course name is a input from the user.
Outputs: A created course in the system.
Action: If the course name is valid a course is created from the input name.
Requires: Nothing
Pre-condition: None
Side-effects: None

1.2. Importing students list

Function: Importing students list
Description: Import a student list from the local computer.
Inputs: Location of the student list. A course.
Source: Location of the student list is an input from the user. The course is selected by the user.
Outputs: Name and e-mail of the students.
Action: If the location of the list of students is valid and the format of the list is valid the students are loaded into the selected course.
Requires: The course in mind have to exist.
Pre-condition: None
Side-effects: None

1.3. Defining an assignment

Function: Defining an assignment
Description: Define an assignment in the system and the possibility to set up rules for the assignment.
Inputs: An assignment name. A course. Expected subject of the e-mails. Number of files and file format.
Source: All inputs from the user.
Outputs: A defined assignment.
Action: Define an assignment in the input course for all the students in the course. The assignment is associated with the input rules (number of files and file format) and the expected subject line in the e-mail.
Requires: The course in mind have to exist.
Pre-condition: None
Side-effects: None

2. Collection and organization of assignments

2.1. Importing assignments

2.1.1. From the e-mail account

Function: Import an assignment by e-mail
Description: Imports an assignment sent to the e-mail account of the teacher.
Inputs: An e-mail
Source: E-mail server
Outputs: Assignment extracted from the e-mail
Destination: Organize e-mails.
Action: Connect to e-mail server. Download header from server. Validate e-mail. If email is valid download e-mail and call Organize e-mails.
Requires: Connection to the Internet. E-mail account with POP3 support.
Pre-condition: None
Post-condition: None
Side effects: None

2.1.2. Manually

Function: Manual importation of an assignment
Description: Import assignments manually.
Inputs: Course, student, assignment and possibly some files.
Source: User input
Outputs: None
Action: Mark assignment and copy files to the folder hierarchy. The folder is determined from course, student and assignment.
Requires: A defined course with students and assignments.
Pre-condition: None
Post-condition: None
Side effects: None

2.2. Organization of the assignment

2.2.1. Organization of the assignment

Function: Deleting an assignment
Description: Deletes files and messages accidentally associated with an assignment.
Inputs: Assignment
Source: User input
Outputs: None
Action: Delete files and messages associated with an assignment.
Requires: A defined course with students and marked assignments.
Pre-condition: None
Post-condition: The selected files and messages are deleted.
Side effects: None

2.2.2 Organization of the assignment

Function: Organize assignments.
Description: Saves files and messages from the assignment in a folder hierarchy.
Inputs: A received assignment
Source: Import assignments by e-mail
Outputs: None
Action: Save the message and attached files in folder hierarchy. The folder is determined from course, student and assignment. Mark the assignment.
Requires: A defined course with students and assignments.
Pre-condition: None
Post-condition: Files are saved to disk.
Side effects: None

3. Edition and grading of assignments

3.1. Opening files

Function: Opening files
Description: Opens the file attached in the given email.
Inputs: The filename for the attached file
Source: E-mail client
Outputs: None
Destination: None
Action: The appropriate application to be used to open the file is launched.
Requires: That the users system has support to view the file.
Pre-condition: The user must have informed the email senders of which file format to use.
Post-condition: None
Side effects: File format constrains are dependent on the users system.

3.2. Commenting files

Function: Commenting files
Description: The system gives the user the ability to write comments for the opened file.
Inputs: Users comment.
Source: The user.
Outputs: None
Action: The user writes a comment about the file. The file is saved and closed.
Requires: the relevant information.
Pre-condition: A file has been opened.
Post-condition: The written comment is saved in the storage system with relevant information of who the comment was intended to, to which file and assignment.
Side effects: Only general comments suits best.

3.3. Grading assignments

Function: Grading assignments
Description: To grade a assignment.
Inputs: A grade. A course, a student in the course and a assignment from the student.
Source: All are inputs from the user.
Outputs: A graded assignment.
Action: Grade the selected assignment (that belong to a student, who belong to a course) with the input grade.
Requires: The course in mind have to exist in the system, the student have to exist in the system and the assignment have to exist in the system.
Pre-condition: None
Side-effects: None

4. Answer to the assignments

4.1. Delivery confirmation

Function: Assignment delivery validation.
Description: Validates a given e-mail as a correct assignment delivery, and notifies the sender either of the correct validation and reception of this or of the errors that it has.
Inputs: An e-mail e.
Source: The mailbox of the e-mail account associated with the user.
Outputs: Validity of e as a correct assignment delivery.
Action: e is validated following the steps reflected in the table:

	Validation	If true	If false
1	e corresponds to a defined assignment	Next step	Ignore e, return false
2	e comes from a defined student	Next step	Ignore e, reply a warning, return false
3	e respects the expected format	Next step	Ignore e, reply a warning, return false
4	e has been sent in time	reply a receipt, return true	Reply a receipt with a warning, return true

The test that the function applies to e in the step 3 is to check if the number of attached files and the format of these corresponds to what is expected to be delivered in the assignment (pattern previously defined by the user).
Requires: Internet connection, in order to automatically reply to e if necessary.
Pre-condition: None.
Post-condition: None.
Side effects: A receipt e-mail or a warning e-mail can be replied to the sender of e.
Special requirements: The receipt and warning e-mails shall be encoded as plain text, using the standard UTF-8.

4.2. Sending feedback

4.2.1. Send feedback to student

Function: Send feedback to student.
Description: Sends feedback on an assignment to the student.
Inputs: Assignment, student, comments and grades.
Source: User input
Outputs: e-mail
Destination: e-mail server
Action: Compose feedback e-mail and send to student.
Requires: Connection to the Internet. E-mail account with SMTP support.
Pre-condition: None
Post-condition: None
Side effects: None

4.2.2. Composition of feedback.

Function: Composition of feedback.
Description: Puts together a feedback e-mail from comments and grade.
Inputs: Comments, grade
Source: User input
Outputs: E-mail
Destination: Send feedback to an student.
Action: Insert comments and grade into predefined template.
Requires: None
Pre-condition: None

Post-condition: None
Side effects: None

5. Reporting

5.1. General state of a course

Function: Show the general state of a specific course
Description: Provides an overview of the course, assignments and students basic statistical data to the user.
Inputs: A course code or name
Source: Retrieves relevant information from the storage system
Outputs: An overview of the specified course
Destination: None
Action: Either the course exist in the storage system and information can be retrieved or the user is prompted with a 'failed to identify course' message.
Requires: A course identifier
Pre-condition: None
Post-condition: None
Side effects: None

5.2. State of a specific student

Function: Show the state of a specific student
Description: Retrieves information about a specific student.
Inputs: A student code or name.
Source: Retrieves relevant information from the storage system
Outputs: Information about the specified student
Action: Either the student exist in the storage system and information can be retrieved or the user is prompted with a 'failed to identify student' message.
Requires: A student identifier
Pre-condition: None
Post-condition: None
Side effects: None

5.3. State of a specific assignment

Function: Show the state of a specific assignment
Description: Retrieves information about the specified assignment
Inputs: An assignment code or name
Source: Retrieves relevant information from storage system
Outputs: Information about the specified assignment
Action: Either the assignment exist in the storage system and information can be retrieved or the user is prompted with a 'failed to identify assignment' message.
Requires: A assignment identifier
Pre-condition: None
Post-condition: None
Side effects: None

6. Res-system communication

6.1. Importing list of students from the Res system

Function: Get list of students from the RES system.
Description: Gets the list of students and associated information of a specific course from the RES system, storing this in a new file that the system is able to recognize.
Inputs: A course identifier.
Source: The course identifier is an input from the user.
Outputs: A list of students (names, identifiers and associated information).
Destination: Independent application that interacts with the RES system.
Action: If there exists a course defined with the given identifier, a list of students that have registered in the RES system for the specified course is obtained.
Requires: Authorization to access to the RES system.
Pre-condition: None.
Post-condition: None.
Side effects: None.
Special requirements: The list of students must be stored as an XML (Extensible Markup Language) file that can be directly recognized and used in the definition of a course.

6.2. Export grades from the system

Function: Export grades from the system.
Description: Exports the grades stored for an specific assignment and course into a list of grades that can be used later to introduce them in the RES system.
Inputs: An existing course, an existing assignment.
Source: All inputs from the user.
Outputs: A list of grades for a specific assignment (names of students and grades obtained).
Action: A list with the grades of all the students in the given assignment is created.
Requires: Nothing.
Pre-condition: None.
Post-condition: None.
Side effects: None.
Special requirements: The exported list of grades must be stored as an XML file that the application that interacts with the RES system can recognize.

6.3. Store grades in the RES system

Function: Store grades in the RES system.
Description: Stores the given grades of a list of students in a specific assignment.
Inputs: A list with the grades of all the students in a specific assignment.
Source: A file provided by the user.
Outputs: The RES system is updated with the given information.
Destination: Independent application that interacts with the RES system.
Action: If the given list is correctly formatted and the given course, assignment and students are defined in the RES system, the given grades of a list of students in a specific assignment are stored in it.
Requires: Authorization to access to the RES system.
Pre-condition: None.
Post-condition: None.
Side effects: None.
Special requirements: The input (list of grades) must be stored as an XML file.