Test Cases:

**<UD1> - <Upload\_Cancelled>: Ecore File to Upload == FALSE:**

Description:

This test case checks what is going to happen if user aborts uploading an UML diagram by clicking “Cancel” button in upload process.

Pre-conditions:

The user has logged in to ClubUML and selected to upload a diagram.

The popup window to browse for a file is displayed.

Post-conditions:

The user will end the use case by selecting Cancel.

No file will be uploaded and no new filename will appear in the list of files.

Data required:

None.

**<UD2> - <File\_Type>: file type is ecore == TRUE**

Description:

This test case checks if the uploading file meets the required file type (which is in .ecore extension).

If the file type is incorrect, the upload will failed and the user will be prompted with a error message.

Pre-conditions:

The user selected the upload option.

The user selected the file to be uploaded to ClubUML

Post-conditions:

The file type is validated and accepted by ClubUML.

Data required:

An UML diagram encoded in an Ecore file.

**<UD3> - <Access\_Database>: ClubUML can access the DatabaseServer == FALSE**

Description:

This test case checks if how the ClubUML handles the Database server is not available after the ecore file has been uploaded to its file system.

Pre-conditions:

An ecore file has been renamed and uploaded to its file system

Post-conditions:

The user will be given an error message that the server was unavailable and that the file path was not saved in the database

The uploaded file will be removed from file system

No new filename will appear in the list of files.

Data required:

None.

**<UD4> - <Convert\_to\_Image>: Ecore File can be converted to PNG file == TRUE**

Description:

This test case checks if the uploading file is able to be converted to a PNG file.

Pre-conditions:

The ecore file has been renamed and uploaded to file system.

The ecore file’s full path has been stored in database server.

Post-conditions:

The png file has been created with ecore file’s name and stored to file system.

The png file’s full path has been stored in database server.

Uploading diagram image is displayed.

Data required:

An EPackage/EObject from the ClubUML parser.

**<UD5> - <Comment\_Added>: content is null == FALSE**

Description:

This test case checks if the user input comments for the uploading diagram. Entered comments will have a non-empty content value in the data storage.

Pre-conditions:

Comment field is displayed during diagram upload.

Post-conditions:

Comment input is added for the uploading diagram (diagram\_Id).

Data required:

Comment input from the user.

**<Test Case 01> - <Registration Test Case >:**

**Description:**

This test case evaluates the process of registration in ClubUML. User need to fill in the username, password, confirm password and email.

Expected result: Registration success, user can login within the username and password.

**Pre-conditions:**

User goes to ClubUMLhomepage, and click ‘here’ to register.

**Post-conditions:**

1 User types in a username which already exists in database. There will be a red mark says ‘name exist!’

2 User types in confirm password, if it not match with the password above, there will be a red mark says ‘password not match’.

3 If any of these three text field remain blank, there will be an alert.

**Data required:**

String

**<Test Case 02> - <Login Test Case>:**

**Description:**

This test case is to verify the login process of ClubUML web application. The user will type in the username and password to login the application, and there are two expected results. One is the username password pair has verified by the server and the user successfully login. Other is the username or password was wrong and the user login failed.

**Pre-conditions:**

The user open the browser and reach the ClubUML homepage.

**Post-conditions:**

1. The user typed in the correct username and password, and successfully login.
2. The user typed in the wrong username or password, and the server denies the attempt and redirects the user to the login failed page.

**Data required:**

1. username string
2. password string

**<Test Case 03> - <Test Upload files >:**

**Description:**

User clicks the upload file button and chooses a file to upload. After clicks the confirm button, file upload successfully.

Expected result: file upload successfully and the file can be shown on the file list with upload time.

**Pre-conditions:**

User must log in.

**Post-conditions:**

File should be shown on the files list.

**Data required:**

File name, path, upload time.

**<Test Case 04> - <Show file diagram >:**

**Description:**

Users choose one of the file on file list. And click display button.

Expected result: the chosen file show on the left area of the same pages.

**Pre-conditions:**

User chooses one file and click display button.

**Post-conditions:**

A file displayed

**Data required:**

File, file path from database, file upload time.

**<UserD1> - <Register New User>:**

Description:

This test case checks if we could add new entry in DB User Table through UserDAO class by providing new user’s name, password, email, security question and security answer.

Pre-conditions:

The user opened the registration page and filled out all required information.

The servlet passed all the information/parameters to addUser() method in UserDAO class

Post-conditions:

A new entry/row is created in User Table

A new Id is created by MySQL and is returned to associate with this new user object.

Data required:

The new user’s name, password, email, security question and security answer

**<UserD2> - <Verify A User>**

Description:

This test case checks if we could find an existed user in DB by providing username and password

Pre-conditions:

The user opened the login page and input username and password

The servlet passed username and password to getUser() method in UserDAO class

Post-conditions:

An existed entry will be found in User Table.

Related data will be retrieved and set as a user object/bean to be used in application

Data required:

Username and password

**<UserD3> - <Verify A User by Security Question>**

Description:

This test case checks if we could find an existed user in DB by providing username, password, and security question answer

Pre-conditions:

The user opened the login page and input username, password, security question answer

The servlet passed username, password and security question answer to getUser() method in UserDAO class

Post-conditions:

An existed entry will be found in User Table.

Related data will be retrieved and set as a user object/bean used in application

Data required:

Username and password

**<UserD4> - <Update User Profile>:**

Description:

This test case checks if we could update a user’s profile

Pre-conditions:

The user opened the profile page and edited any field among username, password, security question answer

The servlet changed username, password, security question answer for current user.

The servlet passed this user object to updateUser() method in UserDAO class

Post-conditions:

The current user entry will be updated in User Table.

Data required:

Updated username, password, security question answer

**<ProjectD1> - <Create New Project>:**

Description:

This test case checks if we could add new entry in DB Project Table through ProjectDAO class.

Because for our purpose, only one project exists in this system and there is no approach for user to edit it. The system will create a project entry in DB first time launched and will use this only one project later on.

Pre-conditions:

The system has its first user.

Post-conditions:

A new entry/row is created in Project Table

A new Id is created by MySQL and is returned to associate with this project object.

Data required:

A default project Name.

**<ProjectD2> - <Retrieve Project>:**

Description:

This test case checks if we could get the only one entry in DB Project Table through ProjectDAO class.

Pre-conditions:

Every time the system has new user, servlet created a new user object.

Post-conditions:

An existed entry will be found in Project Table.

Related data will be retrieved and set as a project object/bean to be used in application

Data required:

A default project Name.

**<HistoryD1> - <Create New History Record>:**

Description:

This test case checks if we could add new entry in DB EditingHistory Table through EditingHistoryDAO class.

Pre-conditions:

The user uploaded a new diagram.

Or the user promoted a diagram after comparison.

The servlet created an EditingHistory object and set DiagramId, UserId related to it, then passed the object to addHistory() in EditingHistoryDAO

Post-conditions:

A new entry/row is created in EditingHistory Table

A new Id is created by MySQL and is returned to associate with this EditingHistory object.

Data required:

Diagram\_Id and User\_Id

**<HistoryD2> - <Retrieve the list of EditingHistory>**

Description:

This test case checks if we could find retrieve the list of EditingHistory ordered by recent editing time.

Pre-conditions:

The user opened the home page to see the list of all diagram.

The servlet call getPriorityList()in EditingHistoryDAO class.

Post-conditions:

A list of diagram will be retrieved from DabaBase and passed to servlet.

Data required: None.

The user opened the profile page and edited any field among username, password, security question answer

The servlet changed username, password, security question answer for current user.

The servlet passed this user object to updateUser() method in UserDAO class

Post-conditions:

The current user entry will be updated in User Table.

Data required:

Updated username, password, security question answer

**<Test Case ID> - <Test Case Name>:**

Description: [Describe the logical condition that the Test Case evaluates. Include the expected result.]

Pre-conditions: [List conditions that must be true before this Test Case can start.]

Post-conditions: [List conditions that should be true when this Test Case ends.]

Data required: [Identify the type of data required for this Test Case.]

**<CA01 > - <UML Diff\_CompareEPackage>:**

Description: To compare two EPackages from two ECore files and determine if they match or not. The test evaluates if it performs this and whether it gives the expected outcome when a match is found, ie call EClass method and call report or terminate the program when not matched.

Pre-conditions: UML Diff Algorithm has been initiated and two EPackages are available and given as input.

Post-conditions: If we get a match, EClass method is called and report method is called.

If a match is not found, it will report that no matches were found and end the program.

Data required: None.

**<CA02> - <UMLDiff\_CompareEClass>:**

Description: To compare two EClasses from two EPackages files and determine if they match or not.

A Match may or may not be found. The test evaluates whether it calls the EReference method, EMethod method and EAttribute method and a report method are called when a match happens or terminates the program when no match happens.

Pre-conditions: Two EPackage have been matched.

Post-conditions: If we get a match, EReference, EMethod and EAttribute is called and also report method is called.

If a match is not found, it will report that no matches were found and end the program.

Data required: None.

**<CA03> - <UMLDiff\_CompareEMethod>:**

Description: Main purpose is to compare two EMethods from two EClasses files and determine if they match or not.

A Match may or may not be found. The test evaluates whether it finds a match and then calls a report or it terminates without any match and reports this.

Pre-conditions: Two EClass objects have been matched.

Post-conditions: Test conclusions will be printed in the report.

i.e. if a match is found, the report will show that there is a match or it will show no match otherwise.

Data required: None.

**<CA04> - <UMLDiff\_CompareEAttribute>:**

Description: Main purpose is to compare two EAttributes from two EClasses files and determine if they match or not.

A Match may or may not be found. The test evaluates whether it finds an attribute match and then calls a report or it terminates without any match and reports this.

Pre-conditions: Two EClass objects have been matched.

Post-conditions: Test conclusions will be printed in the report.

i.e. if a match is found, the report will show that there is a match or it will show no match otherwise.

Data required: None.

**<CA05> - <UMLDiff\_CompareEReference>:**

Description: Main purpose is to compare two EReferences from two EClasses files and determine if they match or not.

A Match may or may not be found. The test evaluates whether it finds the right reference type match and then calls a report or it terminates without any match and reports this.

Pre-conditions: Two EClass objects have been matched.

Post-conditions: Test conclusions will be printed in the report.

i.e. if a match is found, the report will show that there is a match or it will show no match otherwise.

Data required: None.