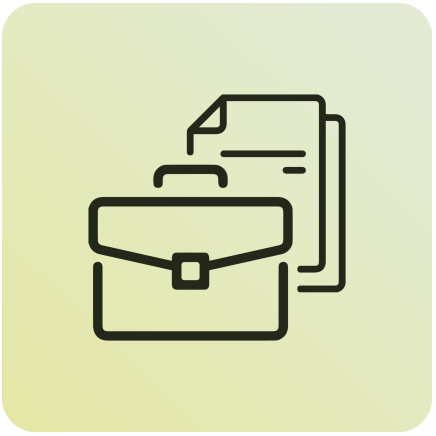
Τεχνολογία Λογισμικού
Τομέας Ηλεκτρονικής και Υπολογιστών
Τμήμα ΗΜΜΥ
Α.Π.Θ.

8th Semester
Spring 2022



Interview Now

From experts to experts

User Requirements and System Standards

Del.1.1

Version 0.1
(draft)

April 2022

Members of the Developer Team

Name	OA	Email
Α. Συμεωνίδης	*	asymeon@issel.ee.auth.gr
Ανδρέας Σαπουντζής	7	spandreas@ece.auth.gr
Γιώργος Γούναρης	7	ggounaris@ece.auth.gr
Χρήστος Χρυσικός	7	christosc@ece.auth.gr
Δημήτρης Φώλας Δεμίρης	7	folasded@ece.auth.gr

Table of Content

Figure & Diagram Appendix

1 System Requirements

1.1 Functional Requirements (Use case Scenarios)

<FR-1>
The user must be able to import his personal details.
Description: The user profile will contain all the necessary information required of the user. Upon account creation, the user will be asked to fill in their personal details in order to complete the registration process.
User Priority: 2/5 The impact to the user experience is small and sometime harmful if it consumes too much time
Technical Priority: 2/5 The personal details enable an easier identification of the user in contrast with having only the email of the user as a reference point. Therefore this requirement serves the purpose of convenient identification of the user and is not necessary for the function of the system.

<FR-2>
The user must be able to edit his personal details.
Description: The user will be able to alter the personal details of the account.
User Priority: 2/5 The user experience is almost not affected at all, just in order for the available information to be accurate.
Technical Priority: 3/5 Similar to the initial admission of the personal details this provides an easier way of user identification and allows the user and the system to adapt, should there be an alteration to this information.

<FR-3>
The user must be able to view the positions for which he filled out an interview assignment and the ones pending.
Description: The user will have access to a history of submissions allowing them to review the job positions they have previously applied to.
User Priority: 3/5 This requirement contributes to the user experience since it conveniently stores previous application saving the user from the trouble of keeping track of previous submissions
Technical Priority: 2/5 The core function of the system is not directly affected by this requirement which serves convenience purposes.

<FR-4>
The user must be logged in, in order to use the application.
Description: It is necessary for the user to be logged in, to fully extend the application's features, but not obligatory. Capabilities such as filling a submit, won't be possible if the user isn't logged in.

User Priority: 4/5 The user won't be able to access the app, if he is not logged in. Therefore no action will be available.
Technical Priority: 4/5 This is a high priority requirement, as the app won't be able to be functional without it.

<FR-5>
The user must be able to fulfill the assignment, through uploading a compressed file (zip or rar) and/or filling out the quiz on the online platform.
Description: The user will be submitting a compressed file, containing the answers/solutions to the Assignment, containing either source code, report files etc. depending on the nature of the job opening in question. The user should also be able to fill in a quiz-style skill assessment, if there is one created by the Manager and submit the answers through the UAL.
User Priority: 5/5 This defines the core way that the application functions. There must be a way for the user to submit the answer files specifically concerning each respective Assignment.
Technical Priority: 5/5

<FR-6>
The user must receive a notification when the U.A.L is available.
Description: The user will receive an email when the U,A,L. is created by the manager with the link for the assignment
User Priority:3/5 This enhances the experience of the manager since they do not have to send individual emails but they are all handled by the system
Technical Priority:1/5 This requirement serves convenience purposes and therefore is not of core importance

<FR-7>
The manager must be able to generate a U.A.L
Description: The manager will be able to create a unique link to assign a customized form to the job requirements, for the user.
User Priority: 3/5
Technical Priority: 4/5 This is a high priority requirement, as it is essential for the advancement of an Assignment.

<FR-8>
The manager will receive a notification regarding the status of the assignments.
Description: The manager must receive a notification when the assignment is fulfilled by the user or when the time frame has lapsed and the user did not fulfill the assignment.
User Priority: 3/5 This enables the manager to easily monitor the status of the pending and submitted assignments

Technical Priority: 3/5 This is a medium priority since it serves the purpose of the app which is fast and convenient hiring.

<FR-9>

The manager must be able to specify the time frame for which the U.A.L will be active.
Description: The manager will specify a time limit for which the link will be active. After the time limit, the link will expire.
User Priority: 4/5 This requirement ensures that the manager will receive candidates for a certain period of time.
Technical Priority: 3/5 It prevents scenarios where users might submit assignments after a significant amount of time since the job opening

<FR-10>

The manager must be able to upload the description of the assignment in a pdf file form and or a quiz style skill assessment for the user to fill.
Description: The management must be able to upload the assignment description in pdf format, as well as a quiz-style skill evaluation for the user to complete.
User Priority: 5/5 This is a very high user priority requirement since the whole user experience is centered around the assignment
Technical Priority: 5/5 This is the main functionality of the app is to assess the user skills with the aim of resulting in a hire.

1.2 Users and External Systems

1.2.1 <User>

He is one of the 2 categories of users in the System. The user-interviewee needs a personal account and needs to be logged in to his personal account in order for the application to be functional.

1.2.2 <Manager>

He is the second category between the 2 categories of users in the System. The manager has an account and in order to use the application he also has to be logged in to the system with his account. Manager type accounts have different access rights than normal users and they can create U.A.L 's and evaluate assignments

1.2.3 <Notification System>

It is an essential external system so the application is functional because it notifies the user when the U.A.L is available and also the manager when the assignment is fulfilled by the user in time or if the time ran out but the user did not upload any files.

1.2.4 <Database>

A Database is a required external system for the functionality of this particular application and the reason is that in order for the application to work there needs to be a place where personal information, the assignments that users uploaded, managers personal information, manager uploaded assignments and quizzes, U.A.L and every possible alteration of the above mentioned data are stored.

1.3 Important Non Functional Requirements.

1.3.1 Performance Requirements

<NFR-1>

The system must have a response time of 150 ms.

Description:

User Priority: 3 / 5

The user experience provided by the system should feel fluid and responsive.

Technical Priority: 4 / 5

It prevents scenarios where users might decrease efficiency and lose time because of the delayed responsiveness.

1.3.2 Usability Requirements

<NFR-2>

The system must be able to group the assignments for the same job opening in the manager dashboard.

User Priority: 4 / 5

It provides a more efficient workspace for the manager off of which the manager can work.

Technical Priority: 1 / 5

This will not change the way the system functions in a drastic way, it will only change the way the assignments are sorted in order to present them to the manager.

1.3.3 Reliability Requirements

<NFR-3>

The system must save the user's progress every time a user action is performed.

User Priority: 5 / 5

It will provide a safer and more reliable user experience.

Technical Priority: 5 / 5

This almost ensures that no work will be lost in case of a disconnection or a system malfunction.

<NFR-4>
The system must start the assignment countdown after the user has viewed the assignment.
User Priority: 5 / 5
This is very important, in order to maintain a fair experience for all users submitting an assignment through the application.
Technical Priority: 5 / 5
This is very important in order for the system to correctly and accurately calculate the remaining time for each assignment before it is due and allow the rest of the time sensitive operations to function as planned.

1.3.4 Policy Requirements (Policy - legal, corporate)

<NFR-5>
The system must be compliant with the privacy regulations of GDPR.
User Priority: 5 / 5
Very important in order to comply with data privacy protocols.
Technical Priority: 1 / 5
This does not however substantially change the behavior of the application.

1.4 Vocabulary determination

User: The interviewee, logged in user.

Manager: Corporate user, logged in linked with the company.

Personal Details: - Full name, Age, email, Personal Description(optional text box), CV (pdf file)

System: - It contains the full parts that affect the completion and smooth function of the application during the use.

U.A.L: Unique assignment link, a one-time use personalized request link for the user.

GDPR: General Data Protection Regulation

2 Use case Scenarios

2.1 Use case Scenarios Diagrams

2.1.1 Feature: Add Personal Details

```
Background:
        Given I am the user
        And I am logged in
Scenario: Successfully entering Personal Details
        Given that I am asked to enter my Personal Details
        When I enter my Personal Details and click Save
        Then I should see a message saying "Your Personal Information has been
        successfully saved"
        And I should be able to view the Personal Details
        And I should be able to go back (to the home page)
Scenario: Unsuccessfully entering Personal Details
        Given that I am asked to enter my Personal Details
        When I enter my Personal Details and click Save
        But the CV filetype is invalid
        Then I should see a message saying " Your Personal Details have not
        been saved, invalid CV filetype, please upload it again and be sure it
        is in a pdf file form"
        And I should be prompted to upload my CV again.
```

2.1.2 Feature: Edit Personal Details

```
Background:
        Given I am the user
        And I am logged in
        And I have already entered my Personal Details
Scenario: Successfully entering Personal Details
        Given that I click edit Personal Details
        When I enter my Personal Details and click Save
        Then I should see a message saying "Your Personal Details have been
        successfully edited"
        And I should be able to view the Personal Details
        And I should be able to go back (to the home page)
Scenario: Unsuccessfully editing Personal Details
        Given that I click edit Personal Details
        When I enter my Personal Details and click Save
        But the CV filetype is invalid
        Then I should see a message saying " Your information have not been
        edited, invalid CV filetype, please upload it again and be sure it is
        a pdf file form"
        And I should be prompted to upload my CV again.
```

2.1.3 Feature: Fulfill and submit an assignment

Background:

Given I am the user

And I am logged in

And I have already entered my Personal Details

And I have access to a valid U.A.L

Scenario: Successfully fulfilling an assignment

When I click on the U.A.L

Then I should see a message saying "The countdown has started"

And I should be able to view the requests of the assignment

And I should be able to fulfill the assignment

And i should be able to click submit the assignment

When I click submit the assignment and the countdown hasn't timed out

Then I should see a message saying "Your assignment has been submitted successfully"And I should be able to review my assignment

But I won't be able to alter my assignment

And I should be able to go back (to the home page)

Scenario: Successfully submitting an assignment

Given that I have already clicked on the U.A.L

And the countdown hasn't timed out

When I click submit my assignment

Then I should see a message saying "Your assignment has been submitted successfully"

And I should be able to review my assignment

But I won't be able to alter my assignment

And I should be able to go back (to the home page)

Scenario: Unsuccessfully submitting an assignment

Given that I have already clicked on the U.A.L

And the countdown has timed out

When I click submit my assignment

Then I should see a message saying "Your assignment wasn't submitted successfully because the countdown has timed out"

And I should be prompted to go back (to the home page)

And i should be prompted to directly contact the manager

2.1.4 Feature: Review submitted assignments

Background:

Given I am the user

And I am logged in

And I have already entered my Personal Details

And I have already submitted at least one assignment

Scenario: Successfully reviewing a submitted assignment

Given the assignment that I click is submitted ,received and reviewed by the manager

When I click on the submitted assignment

Then the assignment should open in a new window

```
        And I should be able to review the fulfilled assignment
        And I should be able to review all the comments made by the manager
        And I should be able to go back (to the home page)
```

2.1.5 Feature: Review pending assignments

```
    Background:
        Given I am the user
        And I am logged in
        And I have already entered my Personal Details
        And I have at least one pending assignment

    Scenario: Successfully reviewing a pending assignment
        When I click one of the my pending assignments
        Then the assignment should open in a new window
        And I should be able to review the fulfilled assignment
        And I should be able to go back (to the home page)
```

2.1.6 Feature: Create an assignment

```
    Background:
        Given I am the manager
        And I am logged in
    Scenario: Successfully creating assignment
        When I click create assignment
        Then I should be asked to create the assignment fields in either
closed form
        multiple choice quiz questions or open questions where the user can
        input text or upload a file in pdf or zip format
        Then I create all the required fields and click create
        Then I should see a message saying "Assignment successfully created"
        And I should be able to view the assignment information
        And I should be able to go back
```

2.1.7 Feature: Edit an assignment

```
    Background:
        Given I am the manager
        And I am logged in
        And at least 1 assignment has been created
    Scenario: Successfully editing assignment
        When I click edit assignment
        Then I should be able to add, remove or modify existing field
        Then I complete the changes and click save
        And I should see a message saying "Assignment successfully saved"
        And I should be able to view the assignment information
        And I should be able to go back
```

2.1.8 Feature: Notify User when UAL for an Assignment is available

```
Background:
	Given that I am the user
	And I am logged in
	And there is an external system called “Notification System”
Scenario: The system sends a notification to the user
	When the system sends a notification that the UAL is generated and
	mentions the link.
	Then I should be able to click the notification
	When I click the notification
	Then I should be redirected to the UAL
	And the system waits 30 minutes
	And the system sends the notification again to make sure that I have
	seen it.
```

2.1.9 Feature: Notify Manager when an Assignment is submitted or the time has lapsed

```
Background:
	Given that I am the manager
	And I am logged in
	And there is an external system called “Notification System”
Scenario: The system sends a notification to the Manager that an Assignment
was submitted
	When the system sends a notification that the assignment was submitted
	or not
	Then I should be able to click the notification and be prompted to
	review the Assignment
	And the system waits 30 minutes
	And the system sends the notification again to make sure that Manager
	have seen it.
Scenario: The system sends a notification to the Manager that an Assignment’s
time has lapsed
	When the system sends a notification that the assignment wasn’t
	submitted
	!! Then I should be able to click the notification and choose whether
	extend the time or not
	And the system waits 30 minutes
	And the system sends the notification again to make sure that I have
	seen it.
```

3 Demonstratively Graphical User Interface

3.1 User and Manager Home Page

GUI Mockup:

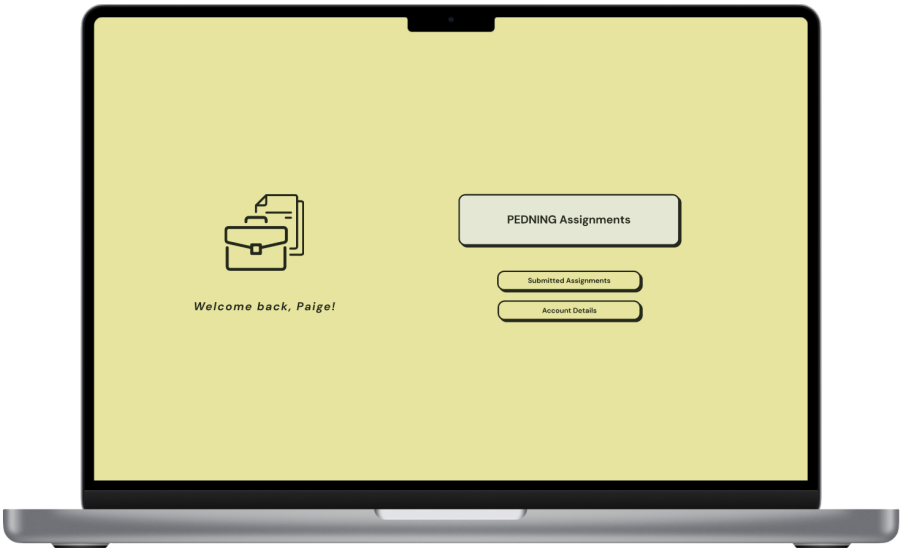


Figure 1. User Home Page

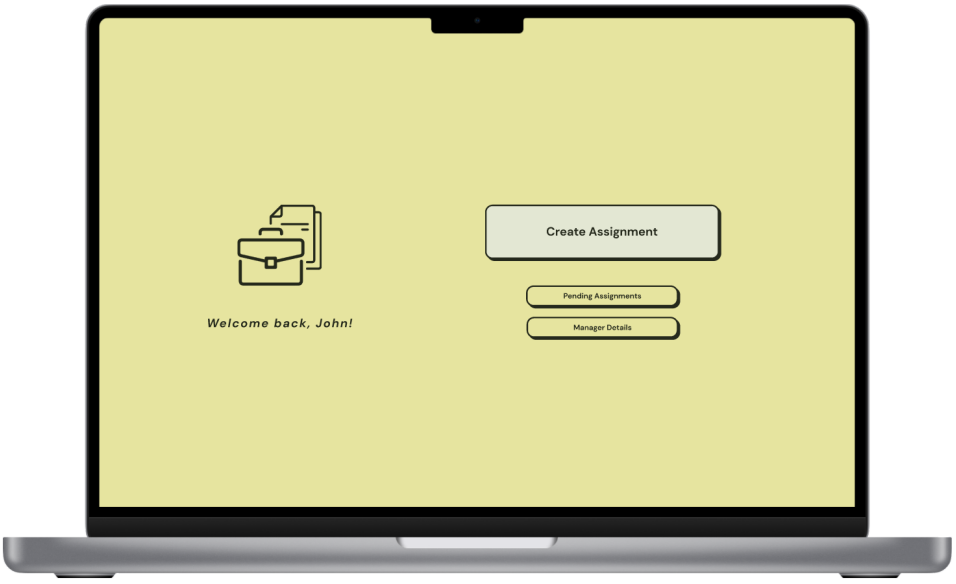


Figure 2. Manager Home Page

3.2 Pending Assignments Page

Figure 3. Pending Assignments Page

3.3 Submitted Assignments Page

Figure 4. Submitted Assignments Page

3.4 User and Manager Account Details Menus

FIgure 5. User Details Normal View

FIgure 6. User Details Edit View

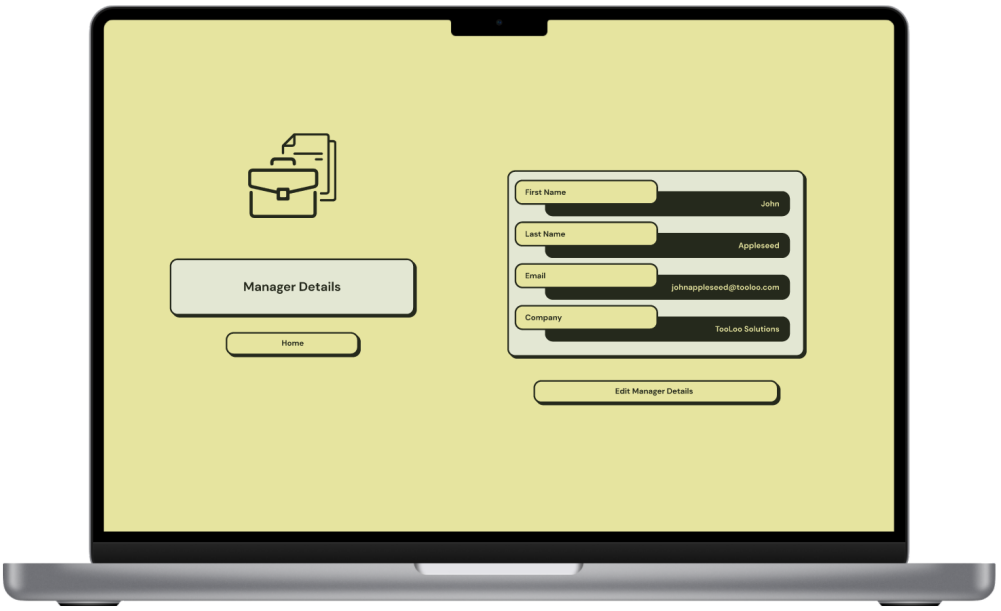


Figure 7. Manager Details Normal View

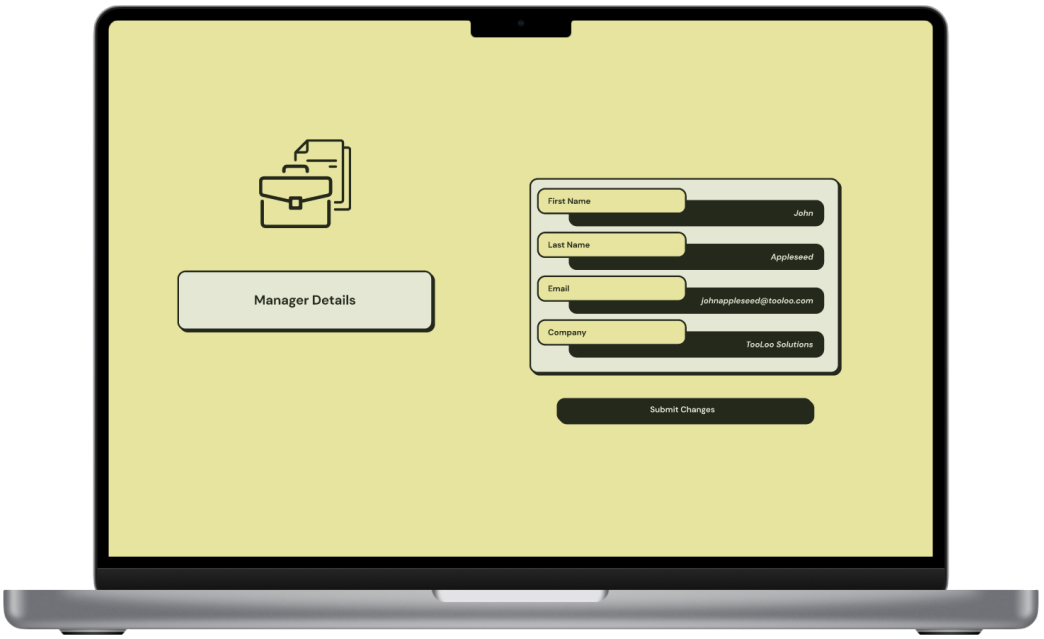


Figure 8. Manager Details Edit View

3.5 User Assignment Submission Page

Figure 9. Submit Assignment Page

4 Static Modelling

4.1 <Vocabulary package of High Priority Scenarios>

4.1.1 UserHandlePackage

This package contains the classes that connect to the welcome page, the Home page as well as to one controller which navigates the user to the display page that he chooses to.

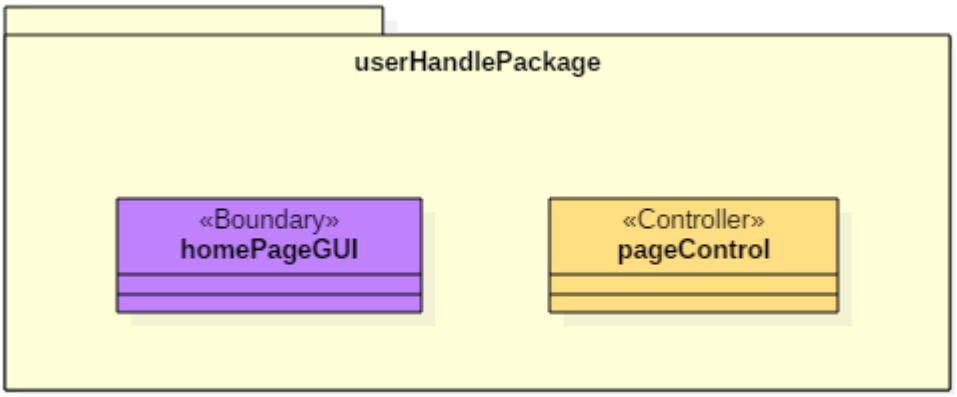


Diagram 1: userHandlePackage

***Controller pageControl*:**

The purpose of this controller is to provide to the user all the available functions so he can navigate to the system.

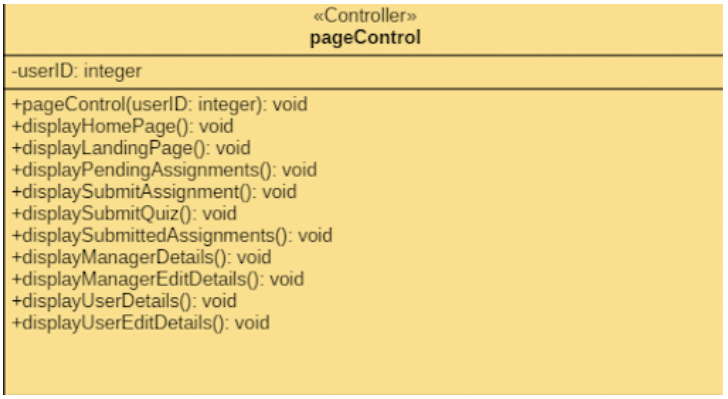


Diagram 2: pageControl

➔ **Class Attributes:**

- userID: This is the unique number linked to the user.

➔ **Class operations:**

- pageControl(userID : integer): This is a structure operation for the class.

- displayHomePage(): This operation is called when we want to display the home page.
- displayPendingAssignments(): This operation is called when we want to display the pending assignments.
- displaySubmitAssignment(): This operation is called when we want to display the contents of an assignment.
- displaySubmitQuiz(): This operation is called when we want to display the quiz contents.
- displaySubmittedAssignments(): This operation is called when we want to display the submitted assignments.
- displayManagerDetails(): This operation is called when we want to display the manager details.
- displayManagerEditDetails(): This operation is called when we want to display the manager's personal details edit page.
- displayUserDetails(): This operation is called when we want to display the user's personal details.
- displayUserEditDetails(): This operation is called when we want to display the user 's personal details edit page.

Boundary homePageGUI:

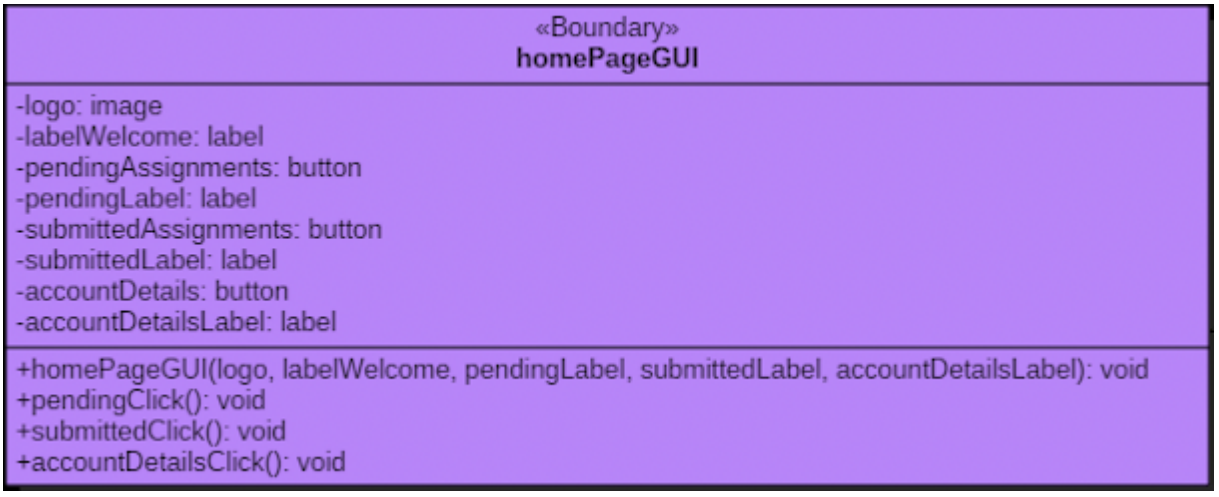


Diagram 3: homePageGUI

This class represents the homePage of the application and includes every available service of the application.

➔ **Class Attributes:**

- logo : Image logo of the application.
- labelWelcome : Sign of Welcome label.
- pendingAssignments : Button for the pending assignments page.
- pendingLabel : Sign of the interface of pending assignments page.

- submittedAssignments : Button for the submitted assignments page .
- submittedLabel : Sign of the interface of the submitted assignments page.
- accountDetails : Button for the interface of the account details page.
- accountDetailsLabel : Sign for the interface of account details page.

➔ **Class Operations:**

- homePageGUI(logo, labelWelcome, pendingaLabel, submittedLabel, accountDetailsLabel) : This is a structure operation for the class.
- pendingClick : This operation calls another operation displayPendingAssignments() in order to display the pending assignments.
- submittedClick : This operation calls another operation displaySubmittedAssignments() in order to display the submitted assignments.
- accountDetailsClick : This operation calls another operation displayUserDetails() in order to display the account details.

☐ **Class Diagrams:**

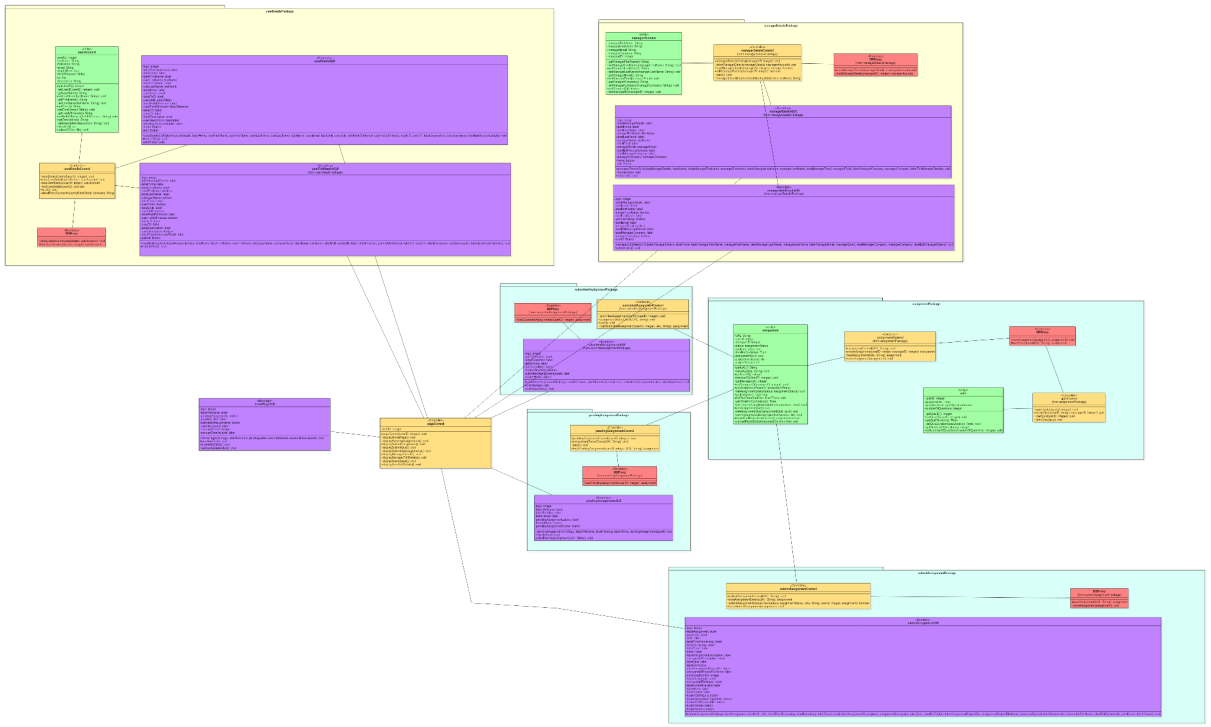


Diagram 4: userHandlePackage Full Class Diagram

4.1.2 assignmentPackage:

This package contains the classes for manager's customized assignments creation and provision to the users, as well as the ability to submit them.

Entity assignment:

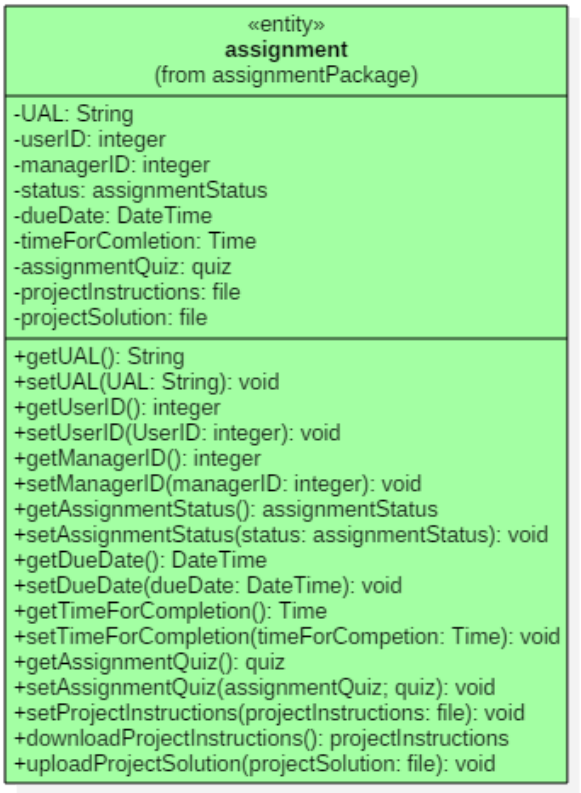


Diagram 5. Assignment Entity Diagram

➔ **Class attributes**

- UAL: The unique assignment link
- userID: The ID of the user that initiated the assignment
- managerID: The ID of the manager who created the assignment
- status: Whether the assignment is submitted or not
- dueDate: Deadline date.
- timeForCompletion: Time limit for assignment..
- assignmentQuiz: The assignment's quiz entity.
- projectInstructions: A file with the instructions for the project.
- projectSolution: A file with the project's solution.

➔ **Class Operations:**

- getUAL(): Returns the Unique Assignment Link.
- setUAL(UAL: String): Sets the Unique Assignment Link.
- getUserID(): Returns the user's ID.
- setUserID(UserID: integer): Sets the user's ID.
- getManagerID(): Returns the manager's ID.
- setManagerID(managerID: integer): void: Sets the manager's ID.
- getAssignmentStatus(): assignmentStatus: Returns the assignment status, whether submitted or pending.
- setAssignmentStatus(status: assignmentStatus): void: Set the assignment status to submitted or not.
- getDueDate(): DateTime: Returns the deadline date.
- setDueDate(dueDate: DateTime): void: Set the deadline date.
- getTimeForCompletion(): Time: Returns the time limit for the assignment.
- setTimeForCompletion(timeForCompletion: Time): void: Set the time limit for the assignment.
- getAssignmentQuiz(): quiz: Returns the quiz entity.
- setAssignmentQuiz(assignmentQuiz: quiz): void: Sets the new Assignment Quiz.
- setProjectInstructions(projectInstructions: file): void: Seth the file with the project's instruction.
- downloadProjectInstructions(): projectInstructions: Downloads the file with the project's instructions.
- uploadProjectSolution(projectSolution: file): void: Uploads the file with the project's solution.

Entity quiz:

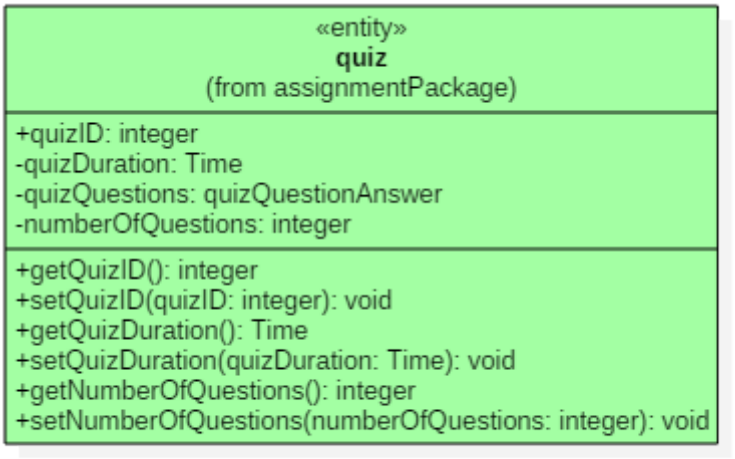


Diagram 6. Quiz Entity Diagram

➔ **Class attributes**

- quizID: The unique quiz id.
- quizDuration: The duration of the quiz.
- quizQuestions: The questions of the quiz.
- numberOfQuestions: Integer with the number of questions of the quiz.

➔ **Class Operations:**

- getQuizID(): integer: Returns the quiz ID.
- setQuizID(quizID: integer): void: Sets the quiz ID.
- getQuizDuration(): Time: Returns the time limit for the completion of the quiz.
- setQuizDuration(quizDuration: Time): void: Sets the time limit for the completion of the quiz.
- getNumberOfQuestions(): integer: Returns the number of quiz's questions.
- setNumberOfQuestions(numberOfQuestion: integer): void: Sets the number of quiz's questions.

Entity quizQuestionAnswer:

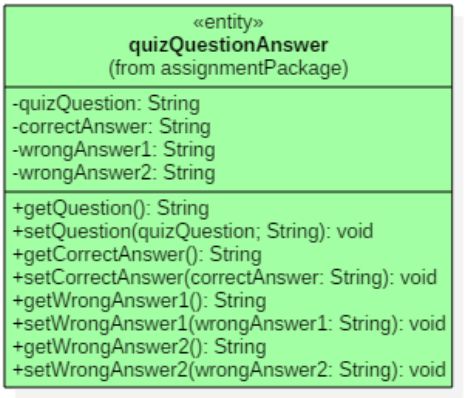


Diagram 7. Quiz Questions and Answers Entity Diagram

➔ **Class Attributes:**

- quizQuestion: The question of the quiz, to be answered
- correctAnswer: The correct answer of the question.
- wrongAnswer1: The approximately wrong answer.
- wrongAnswer2: The fully wrong answer.

➔ **Class Operations:**

- getQuestion(): String: Returns the question of the quiz.
- setQuestion(quizQuestion: String): void: Sets the question for the quiz.
- getCorrectAnswer(): String: Returns the correct answer of the question.
- setCorrectAnswer(correctAnswer: String): void: Sets the correct answer for the question.
- getWrongAnswer1(): String: Returns the approximately wrong answer of the question.
- setWrongAnswer1(wrongAnswer1: String): void: Sets the approximately wrong answer for the question.
- getWrongAnswer2(): String: Returns the fully wrong answer of the question.
- setWrongAnswer2(wrongAnswer2: String): void: Sets the fully wrong answer for the question.

Controller assignmentControl:

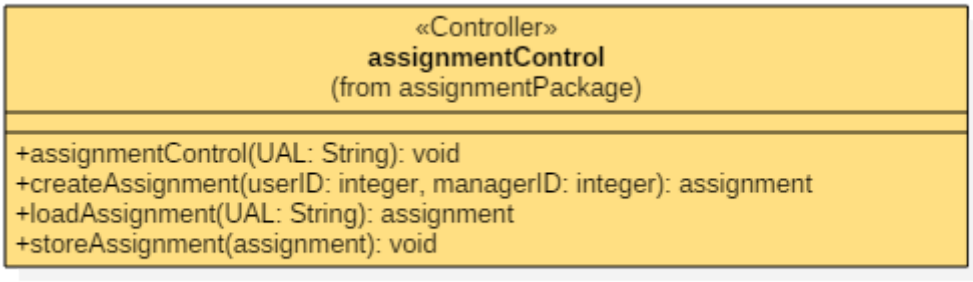


Diagram 8. Assignment Controller Control Class Diagram

➔ **Class Operations:**

- assignmentControl(UAL: String): void: This is a structure operation for the class.
- createAssignment(userID: integer, managerID: integer): assignment: This method creates a new assignment entity using userID and managerID in the database.
- loadAssignment(UAL: String): assignment: This method fetches and returns the requested assignment based on the UAL from the proxy of the database.
- storeAssignment(assignment): void: This method saves/replaces the new/edited assignment to the database.

Controller quizControl:

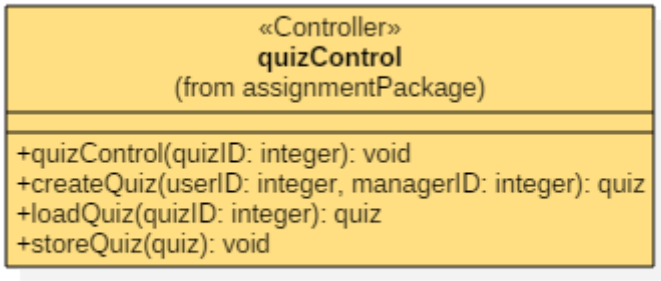


Diagram 9. Quiz Control Control Class Diagram

➔ **Class Operations:**

- quizControl(quizID: integer): void: This is a structure operation for the class.
- createQuiz(user: integer, managerID: integer): quiz:

- loadQuiz(quizID: integer): quiz: This method fetches and returns the requested quiz based on the quizID from the proxy of the database.
- storeQuiz(quiz): void: This method saves the quiz to the database.

Boundary DBProxy:

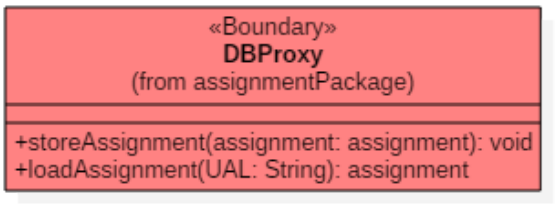
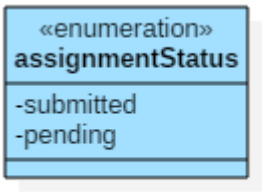


Diagram 10. Assignment Package Database Proxy

➔ **Class Operations:**

- storeAssignment(assignment: assignment): void: This operation stores the specific assignment in the database.
- loadAssignment(UAL: String): assignment: This operation loads and returns a specific assignment.

Enumeration assignmentStatus:



➔ **Class Attributes:**

- submitted: The status of the assignment when it is submitted.
- pending: The status of the assignment when it is pending.

☐ **Class Diagrams:**

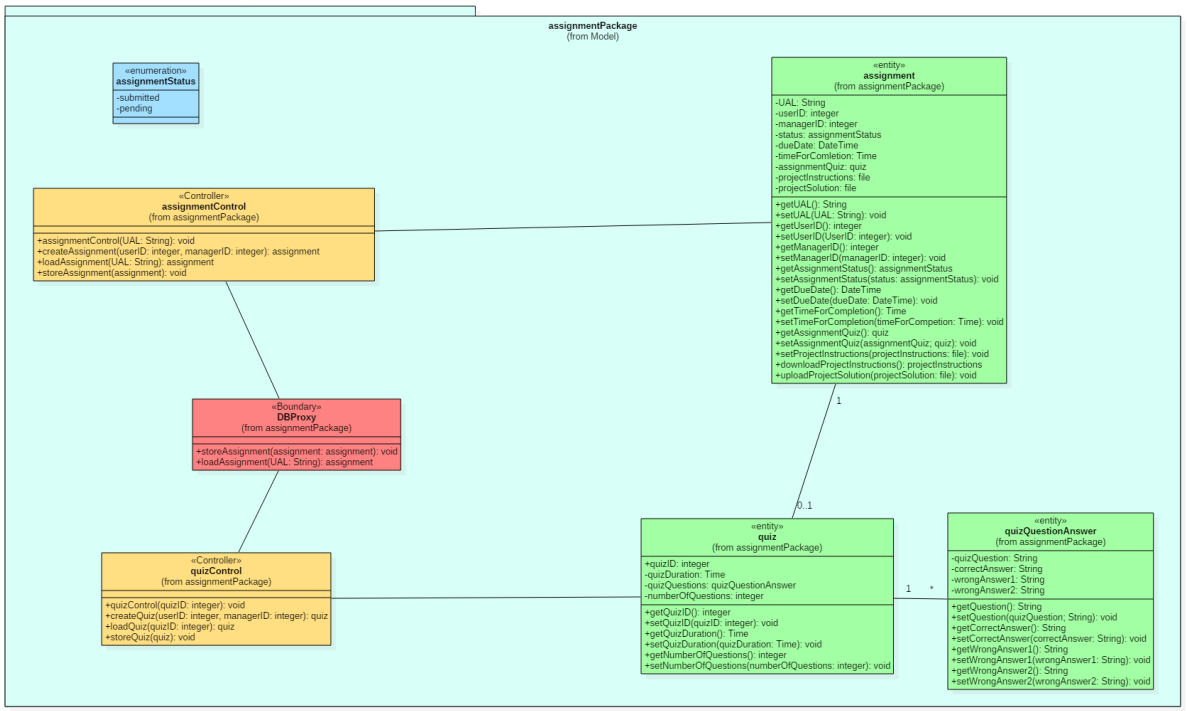


Diagram 11. Assignment Package Full Class Diagram

4.2 <Vocabulary package of Medium Priority Scenarios>

4.2.1 Notification Package

The purpose of this package is to manage notifications sent to the user and the manager respectively based on the status of the assignments that they are linked to.

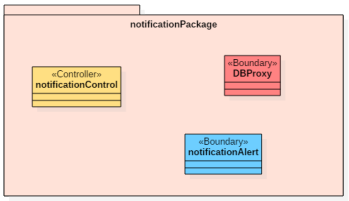


Diagram 12. Notification Package Short View Diagram

Boundary BDProxy:

This class connects to the Database and uses it to load data that has to do pending and submitted assignments.

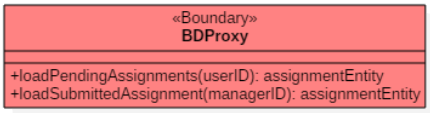


Diagram 13. Notification Package Database Proxy

➔ **Class Operations:**

- loadPendingAssignments(userID) : This method returns the assignments from the database that the user has started and are characterized as pending through his unique ID.
- loadSubmittedAssignments(managerID) : This method returns the assignments from the database that the user has started and are characterized as submitted through his unique ID.

Boundary notificationAlert:

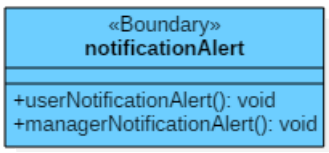


Diagram 14. Notification Alert Boundary Class

This class connects through the Notification Control with the DataBase for alerting the user or the manager respectively in time with a banner.

➔ **Class operations:**

- userNotifiacationAlert(): This method uses the system to announce to alert the user.
- managerNotifiacationAlert(): This method uses the system to alert the manager.

Controller notificationControl:

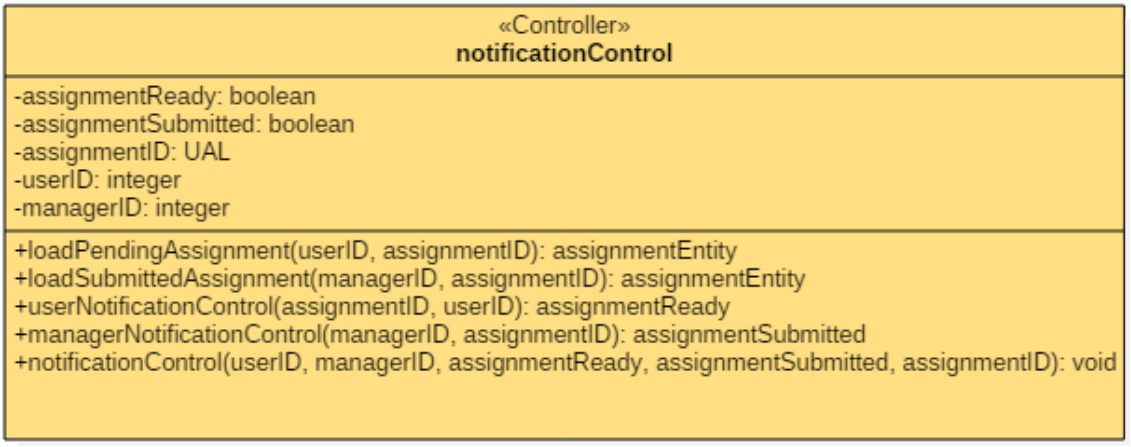


Diagram 15. Notification Controller Control Class

This class is the main controller to decide whether or not the user or the manager should be notified about a certain assignment.

➔ **Class Attributes:**

- assignmentReady: A boolean variable showing if the assignment meant for a user to complete is ready and thus the user should be notified.
- assignmentSubmitted: A boolean variable showing if the assignment's status has been changed to "submitted" and thus is ready and the manager responsible for it should be notified.
- assignmentID: Is the UAL for an assignment, used as so for simplicity's sake.
- userID: Is the unique identification number of a logged in user of the system.
- managerID: Is the unique identification number of a manager on the system.

➔ **Class Operations:**

- loadpendingAssignment: This method fetches and returns the requested assignment entity based on the UAL and the userID from the proxy of the Database.
- loadSubmittedAssignment: This method fetches and returns the requested assignment entity based on the UAL and the managerID from the proxy of the Database.
- userNotificationControl: This operation returns a boolean stating whether or not an assignment is ready and that its user should be notified respectively.

- managerNotificationControl: This operation returns a boolean stating whether or not an assignment is submitted and that its manager should be notified respectively.
- notificationControl: This operation is the constructor function for the Notification Controller.

☐ **Class Diagram:**

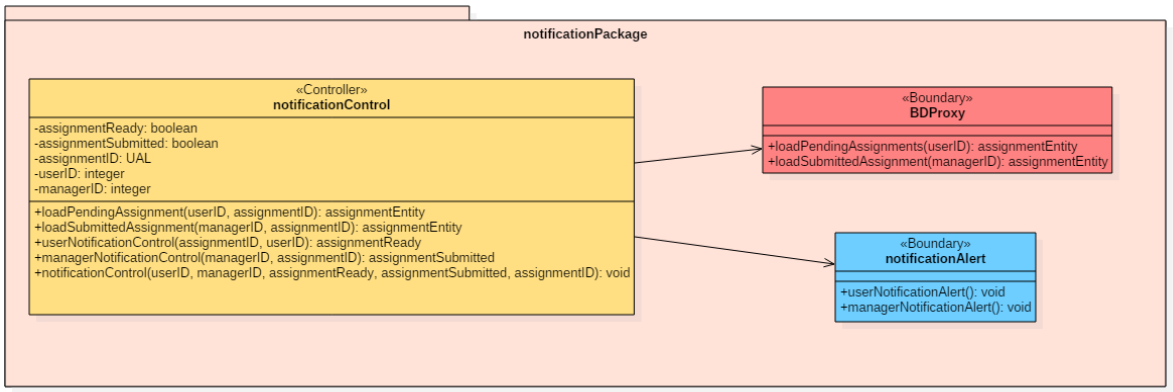
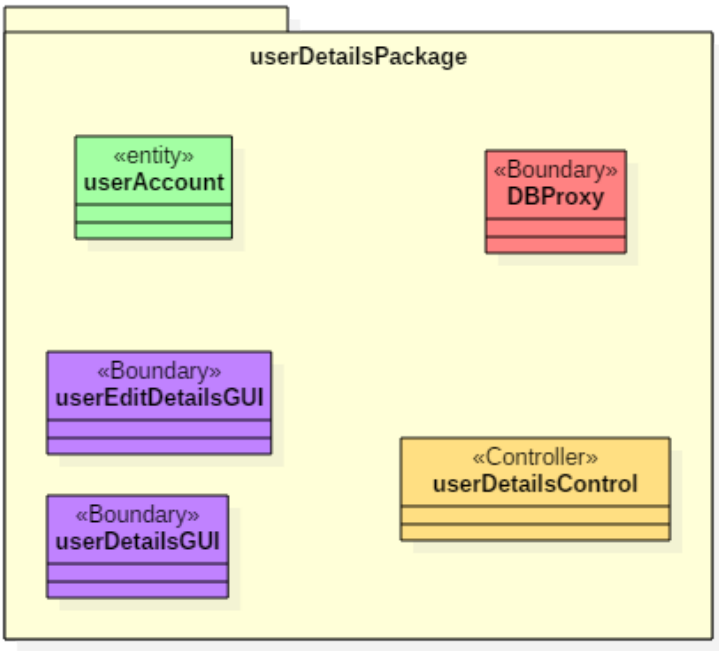


Diagram 16. Notification Package Full Diagram

4.3 <Vocabulary package of Low Priority Scenarios>

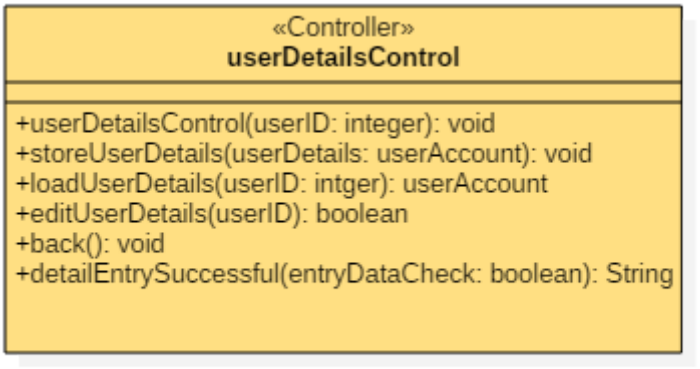
4.3.1 UserDetailsPackage:

This package contains the classes that connect to the welcome page, the Home page as well as to one controller which navigates the user to the display page that he chooses to.



Controller userDetailsControl

The purpose of this controller is to provide to the user all the available functions so he can view and alter all his personal details.

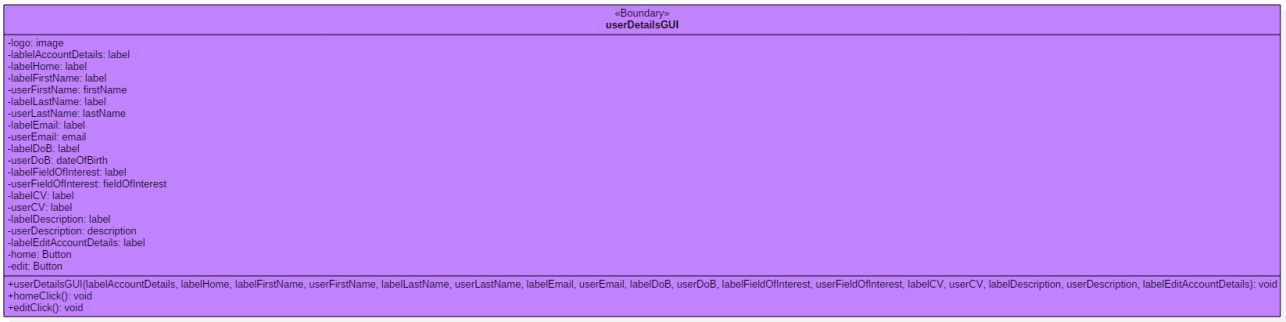


➔ **Class operations:**

- userDetailsControl(userID : integer): This is a structure operation for the class.
- storeUserDetails(userDetails : userAccount): This operation is called when we want to store user details.
- loadUserDetails(userDetails : userAccount): This operation is called when we want to load user details.
- editUserDetails(userDetails : userAccount): This operation is called when we want to edit user details.
- back(): This operation is called when we want to go back.
- detailEntrySuccessful(entryDataCheck : boolean): This operation is called when the entry of the user's personal details is successful and returns a message saying "Personal Details successfully entered"

userDetailsGUI:

This class represents the view of the user's details page.



// - userFirstName, userLastName, userEmail, userDoB, userFieldOfInterest, userDescription???

//Check new attributes and operations for the diagram

➔ **Class Attributes:**

- labelAccountDetails : Label "Account Details".

- labelUserID: Label with user's ID.
- labelFirstName : Label with user's first name.
- labelLastName : Label with user's last name.
- labelEmail : Label with user's email.
- labelDoB : Label with user's date of birth.
- labelFieldOfInterest : Label with the field of interest.
- labelCV : Label with attached user's CV.
- labelDescription : Label with additional Account Details.
- logo: Image logo of the application.
- labelHome: Label of Home.
- home: Button for letting the user return to home page.
- Label
- edit : Button for letting the user edit his personal details.

➔ **Class Operations:**

- userDetailsGUI(labelAccountDetails, labelUserID, labelFirstName, labelLastName, labelEmail, labelDoB, labelFieldofInterest, labelCV, labelDescription) : This is a structure operation for the class.
- homeClick : This operation calls another operation displayHomePage() in order to display the Home Page.
- editClick : This operation calls another operation displayUserEditDetails() in order to display edit user's personal details page.

userEditDetailsGUI:

This class represents the connection of the add/edit user details page with the user.

➔ **Class Attributes:**

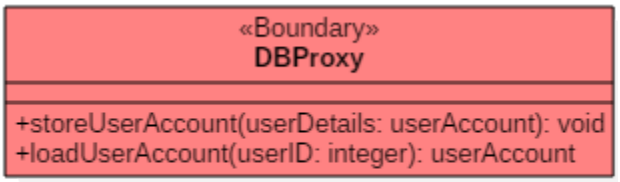
- labelHeader : Label "Edit User Details".
- labelUserID: Label with user's ID.
- labelFirstName : Label with user's first name.
- firstNameField: Text field to insert/edit the first name of the user.
- labelLastName: Label with user's last name.
- lastNameField: Text field to insert/edit the last name of the user.

- labelEmail : Label with user's email.
- emailField: Text field to insert/edit the email of the user.
- labelDoB : Label with user's date of birth.
- DoBField: Text field to insert/edit the DoB of the user.
- labelFieldOfInterest : Label with the field of interest.
- fieldOfInterestField: Text field to insert/edit the field of interest of the user.
- labelCV : Label with attached user's CV.
- CVField: Button to upload the CV of the user.
- labelDescription : Label with additional Account Details.
- descriptionField: Text field to insert/edit the account details of the user.
- logo: Image logo of the application.
- home : Button for returning the user to the home page.
- submit : Button that submits a new modified user's registration.

➔ **Class Operations:**

- userEditDetailsGUI(labelHeader, labelUserID, , labelFirstName, firstNameField, labelLastName, lastNameField, labelEmail, emailField, labelDoB, DobField, labelFieldOfInterest, fieldOfInterestField, labelCV, CVField, labelDescription, descriptionField, logo) : This is a structure operation for the class.
- homeClick : This operation calls another operation displayHomePage() in order to display the Home Page.
- submitClick : This operation calls operation storeUserDetails(userDetails: userAccount) from userDetailsControl in order to save the new modified user's personal details information. Then calls another operation detailEntrySuccessful(entryDataCheck : boolean) from userDetailControl and if the return value is 1, the system redirects the user to the userDetails page otherwise it prints an error message on the screen.

Boundary DBProxy:



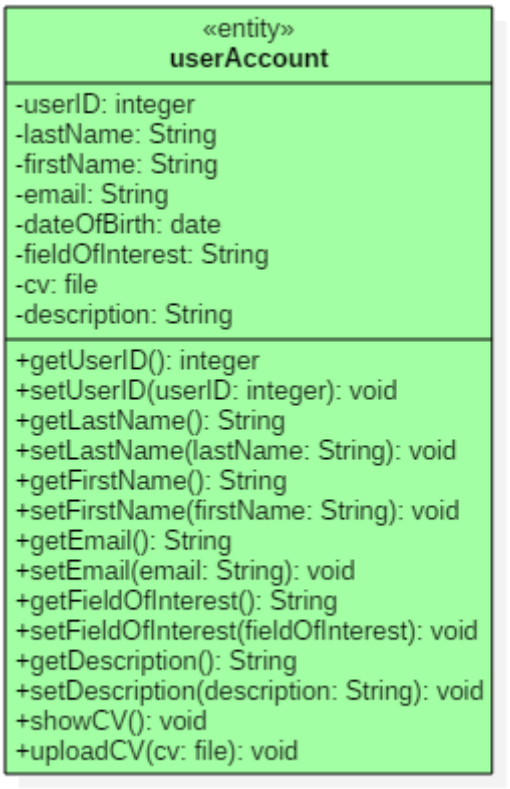
This class represents the connection of the application with the database and is responsible for storing and loading user details in and from the database to the application.

➔ **Class Operations:**

- storeUserAccount(userDetails : userAccount) : This operation is called when we need to store user details to the database.

- loadUserAccount(userDetails : userAccount) : This operation is called when we need to load user details from the database.

Entity userAccount:



This class represents the entity of the user with all his personal details inside the database.

➔ **Class Attributes:**

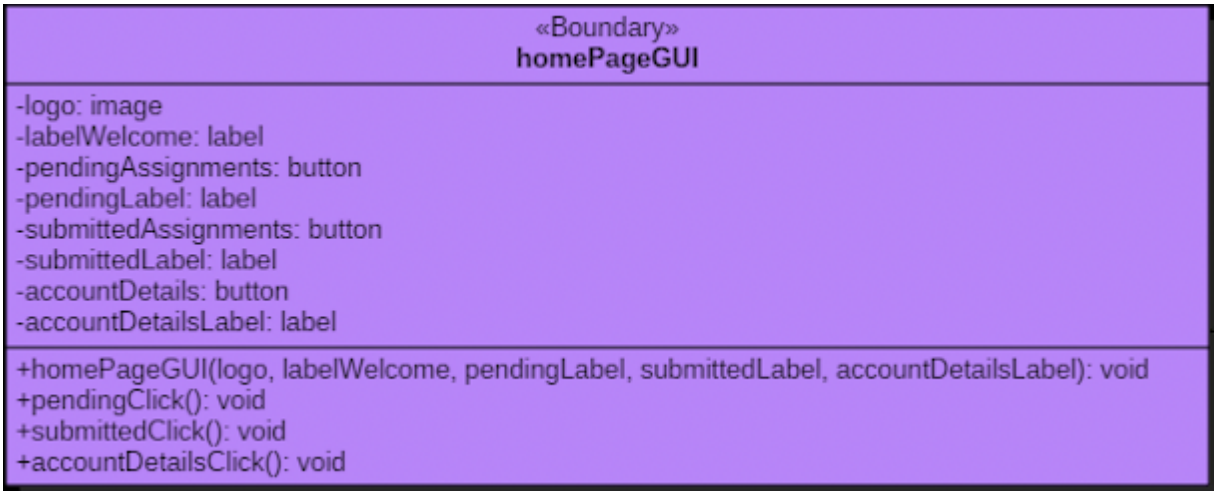
- userID : The unique ID of the user.
- firstName: The first name of the user.
- lastName : The last name of the user.
- email : The email of the user.
- dateOfBirth : The birth date of the user.
- fieldOfInterest : The field of interest of the user.
- cv : The CV of the user.
- description : The additional information of the user.

➔ **Class Operations:**

- getUserID() : Returns the user ID.
- setUserID(userID: integer) : Sets the user ID.
- getUserFirstName : Returns the first name of the user.
- setUserFirstName(firstName: String) : Sets the first name of the user.
- getUserLastName : Returns the last name of the user.

- setUserLastName (lastName: String) : Sets the last name of the user.
- getEmail Returns the email of the user.
- setEmail (email: String): Sets the email of the user.
- getDateOfBirth : Returns the date of birth.
- setDateOfBirth (DoB: integer) : Sets the date of birth of the user.
- getFieldOfInterest: Returns the field of interest.
- setFieldOfInterest (fieldOfInterst: fieldOfInterest): Sets the field of interest of the user.
- getDescription : Returns the additional information of the user.
- setDescription (description: String): Sets the additional information of the user.
- showCV : Return the CV of the user.
- uploadCV : Sets the CV of the user.

Boundary homePageGUI:



This class represents the homePage of the application and includes every available service of the application.

➔ **Class Attributes:**

- logo : Image logo of the application.
- labelWelcome : Sign of Welcome label.
- pendingAssignments : Button for the pending assignments page.
- pendingLabel : Sign of the interface of pending assignments page.
- submittedAssignments : Button for the submitted assignments page .
- submittedLabel : Sign of the interface of the submitted assignments page.
- accountDetails : Button for the interface of the account details page.
- accountDetailsLabel : Sign for the interface of account details page.

➔ **Class Operations:**

- homePageGUI(logo, labelWelcome, pendingaLabel, submittedLabel, accountDetailsLabel) : This is a structure operation for the class.

- pendingClick : This operation calls another operation displayPendingAssignments() in order to display the pending assignments.
- submittedClick : This operation calls another operation displaySubmittedAssignments() in order to display the submitted assignments.
- accountDetailsClick : This operation calls another operation displayUserDetails() in order to display the account details.

☐ **Class Diagram:**

4.3.2 ManagerDetailsPackage:

This package contains the classes that connect to the welcome page, the Home page as well as to one controller which navigates the manager to the display page that he chooses to.

Controller managerDetailsControl

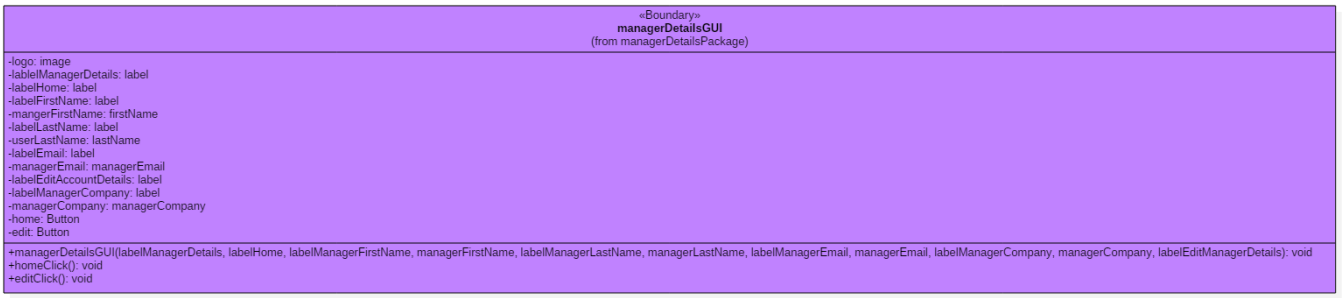
The purpose of this controller is to provide to the manager all the available functions so he can view and alter all his personal details.

➔ **Class operations:**

- managerDetailsControl(userID : integer): This is a structure operation for the class.
- storeManagerDetails(managerDetails : managerAccount): This operation is called when we want to store manager details.
- loadManagerDetails(managerDetails : managerAccount): This operation is called when we want to load manager details.
- editManagerDetails(managerDetails : managerAccount): This operation is called when we want to edit manager details.
- back(): This operation is called when we want to go back.
- detailEntrySuccessful(entryDataCheck : boolean): This operation is called when the entry of the manager's personal details is successful and returns a message saying "Personal Details successfully entered"

managerDetailsGUI:

This class represents the connection of the add/edit manager details page with the manager.



➔ **Class Attributes:**

- labelHeader : Label "Manager Details".
- labelManagerID: Label with manager's ID.
- labelFirstName : Label with manager's first name.

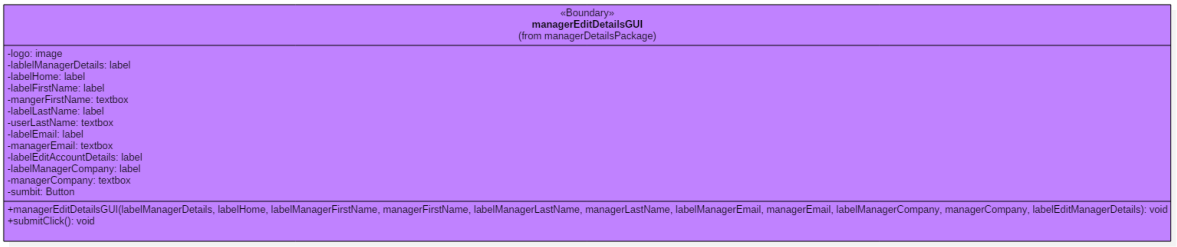
- labelLastName : Label with manager's last name.
- labelEmail : Label with manager's email.
- labelDoB : Label with manager's date of birth.
- labelCompany : Label with the company of the manager.
- labelAccountDetails : Label with additional Account Details.
- logo: Image logo of the application.
- edit : Button for letting the manager edit his personal details.

➔ **Class Operations:**

- managerDetailsGUI(labelHeader, labelUserID, labelFirstName, labelLastName, labelEmail, labelDoB, labelCompany, labelAccountDetails) : This is a structure operation for the class.
- homeClick : This operation calls another operation displayHomePage() in order to display the Home Page.
- editClick : This operation calls another operation displayManagerEditDetails() in order to display the edit manager's personal details page.

managerEditDetailsGUI:

This class represents the connection of the add/edit manager details page with the manager.



➔ **Class Attributes:**

- labelHeader : Label "Edit Manager Details".
- labelManagerID: Label with manager's ID.
- labelFirstName : Label with manager's first name.
- firstNameField: Text field to insert/edit the first name of the manager.
- labelLastName: Label with manager's last name.
- lastNameField: Text field to insert/edit the last name of the manager.
- labelEmail : Label with manager's email.
- emailField: Text field to insert/edit the email of the manager.
- labelDoB : Label with manager's date of birth.
- DoBField: Text field to insert/edit the DoB of the manager.

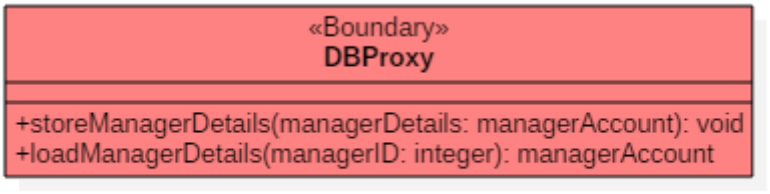
- labelCompany : Label with the company of the manager.
- companyField: Text field to insert/edit the field of the company of the manager.
- labelAccountDetails : Label with additional Account Details.
- accountDetailsField: Text field to insert/edit the account details of the manager.
- logo: Image logo of the application.
- home : Button for returning the manager to the home page.
- submit : Button that submits a new modified manager's registration.

➔ **Class Operations:**

- managerEditDetailsGUI(labelHeader, labelManagerID, , labelFirstName, firstNameField, labelLastName, lastNameField, labelEmail, emailField, labelDoB, DobField, labelCompany, companyField, labelAccountDetails, accountDetailsField, logo) : This is a structure operation for the class.
- homeClick : This operation calls another operation displayHomePage() in order to display the Home Page.
- submitClick : This operation calls operation storeManagerDetails(managerDetails: managerAccount) from managerDetailsControl in order to save the new modified manager's personal details information. Then calls another operation detailEntrySuccessful(entryDataCheck : boolean) from managerDetailControl and if the return value is 1, the system redirects the manager to the managerDetails page otherwise it prints an error message on the screen.

Boundary DBProxy:

This class represents the connection of the application with the database and is responsible for storing and loading manager details in and from the database to the application.



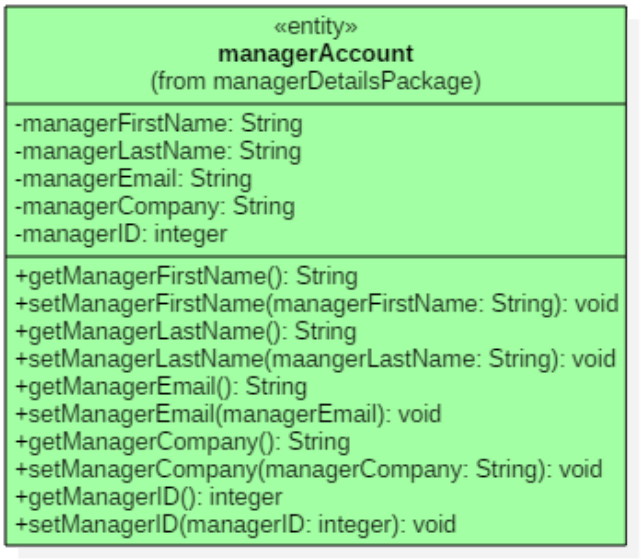
➔ **Class Operations:**

- storeManagerAccount(managerDetails : managerAccount) : This operation is called when we need to store manager details to the database.

- loadManagerAccount(managerDetails : managerAccount) : This operation is called when we need to load manager details from the database.

Entity managerAccount:

This class represents the entity of the manager with all his personal details inside the database.



➔ **Class Attributes:**

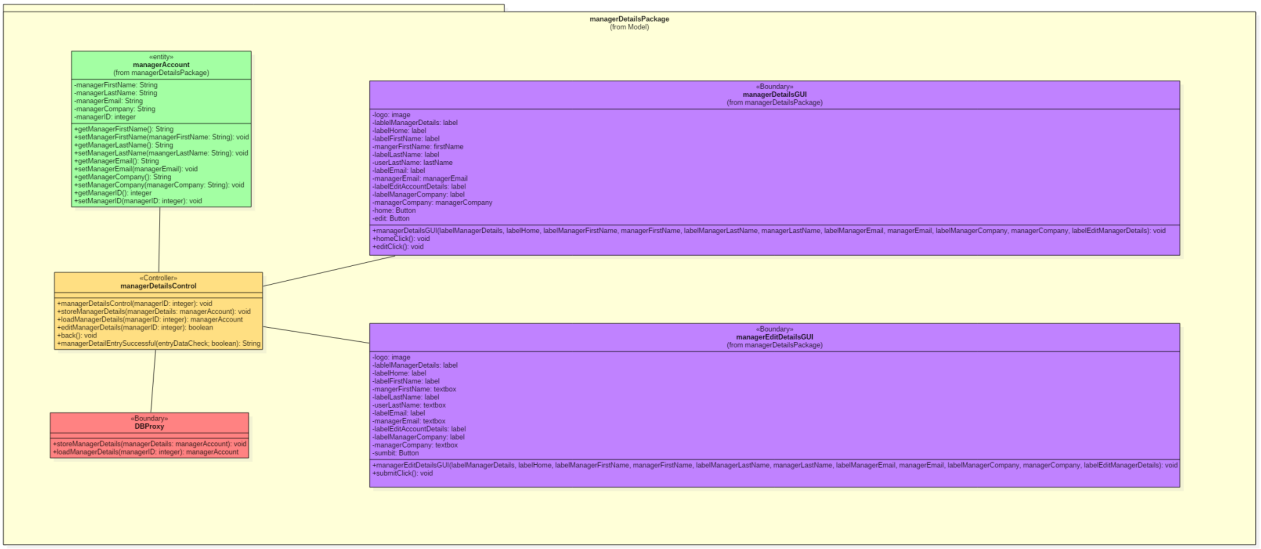
- managerID : The unique ID of the manager.
- firstName: The first name of the manager.
- lastName : The last name of the manager..
- email : The email of the manager.
- dateOfBirth : The birth date of the manager.
- company : The company of the manager.
- description : The additional information of the manager.

➔ **Class Operations:**

- getManagerID() : Returns the manager ID.
- setManagerID(managerID: integer) : Sets the manager ID.
- getManagerFirstName : Returns the first name of the manager.
- setManagerFirstName(firstName: String) : Sets the first name of the manager.
- getManagerLastName : Returns the last name of the manager.
- setManagerLastName (lastName: String) : Sets the last name of the manager.

- getEmail Returns the email of the manager.
- setEmail (email: String): Sets the email of the manager.
- getDateOfBirth : Returns the date of birth.
- setDateOfBirth (DoB: integer) : Sets the date of birth of the manager.
- getCompany: Returns the company of the manager.
- setCompany (company: String): Sets the company of the manager.
- getDescription : Returns the additional information of the manager.
- setDescription (description: String): Sets the additional information of the manager.

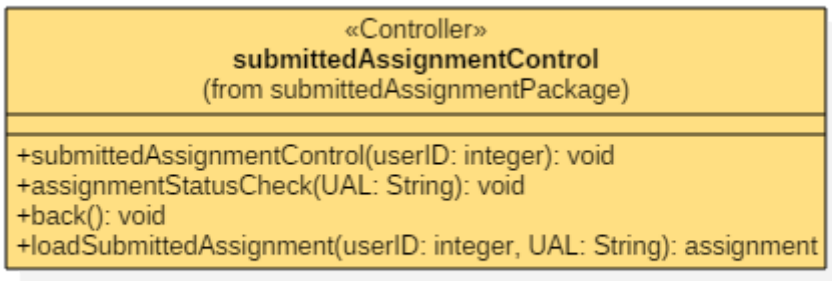
☐ **Class Diagram:**



4.3.3 submittedAssignmentPackage:

This package includes the classes that allow the user to browse and view all the assignments the user has previously submitted and the necessary controls to ensure that the assignments belong to the user and have been submitted.

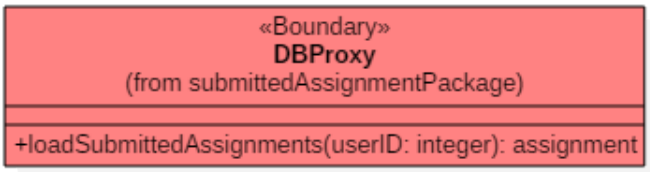
Controller submittedAssignmentControl:



➔ **Class Operations:**

- submittedAssignmentControl(userID: integer): void: This is a construction operation for the class
- assignmentStatusCheck(UAL: String): void: Checks the status of the assignment
- back(): void: Returns to the submittedAssignmentGUI page
- loadSubmittedAssignment(userID: integer, UAL: String): assignment: This operation loads and returns a specific assignment

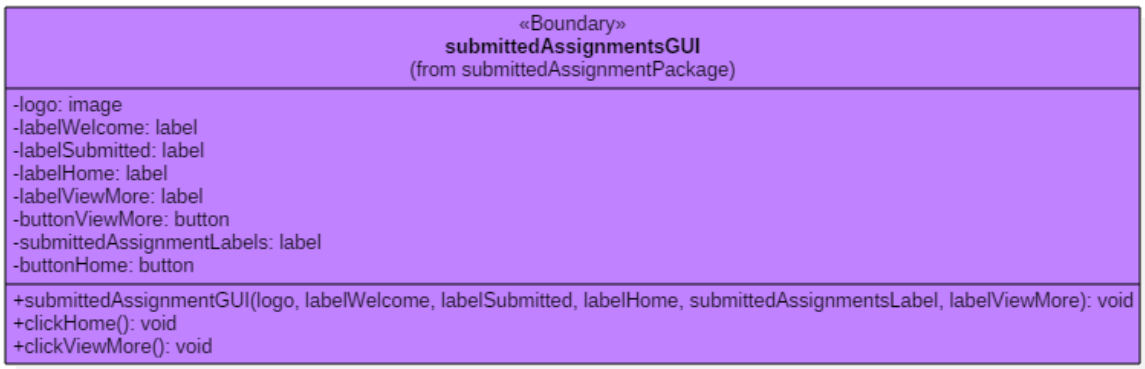
Boundary DBProxy:



➔ **Class Operations:**

- loadSubmittedAssignments(userID: integer): assignment: queries and returns the submitted assignments by a specific user

Boundary submittedAssignmentGUI:



➔ **Class Attributes:**

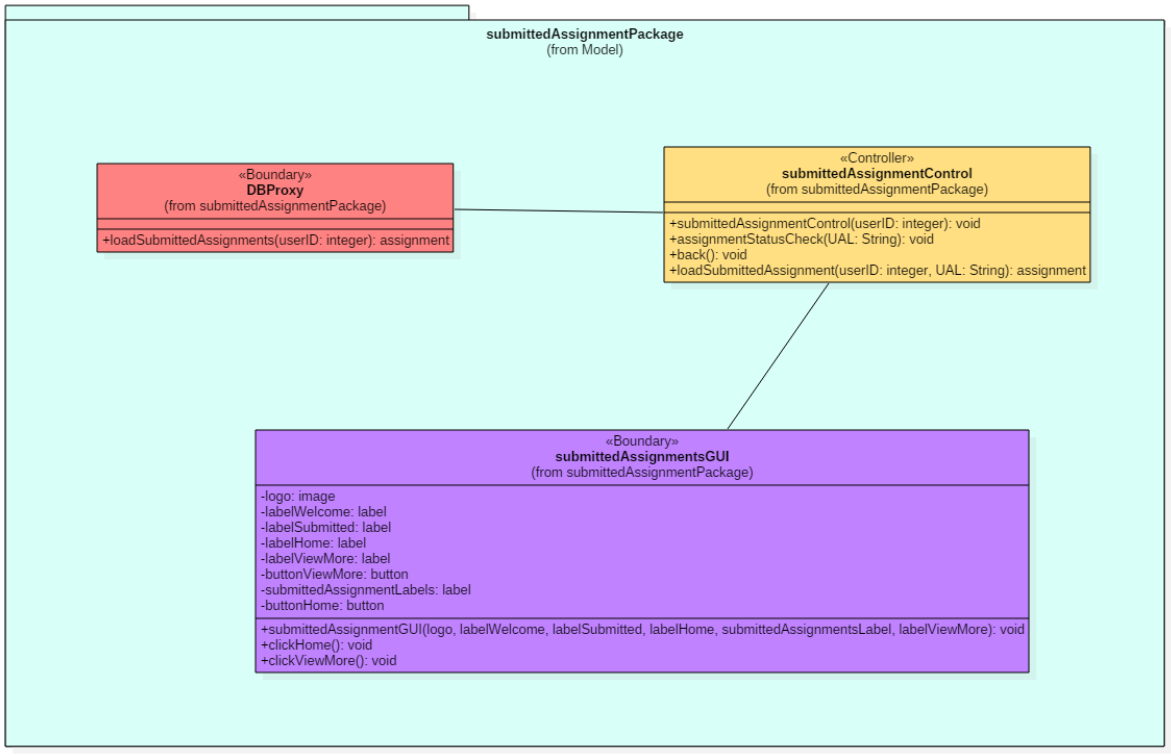
- logo: image of the logo of the app
- labelWelcome: label containing a welcome message
- labelSubmitted: label title of the submitted assignments list
- labelHome: homepage button label
- labelViewMore: load more assignments label
- buttonViewMore: load more assignments button

- submittedAssignmentLabels: list of labels containing submitted assignments
- buttonHome: homepage button

➔ **Class Operations:**

- submittedAssignmentGUI(logo, labelWelcome, labelSubmitted, labelHome, submittedAssignmentLabels, labelViewMore): void: This is a construction operation for the class
- clickHome(): void: Returns to the homepage by calling the displayHomePage method from the pageControl controller
- clickViewMore(): void: loads and displays more submittedAssignments

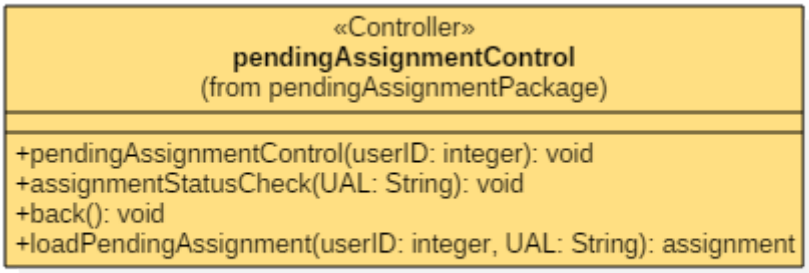
☐ **Class Diagram:**



4.3.4 pendingAssignmentPackage

This package includes the classes that allow the user to browse and view all the assignments the user has previously initiated and the necessary controls to ensure that the assignments belong to the user and are currently pending.

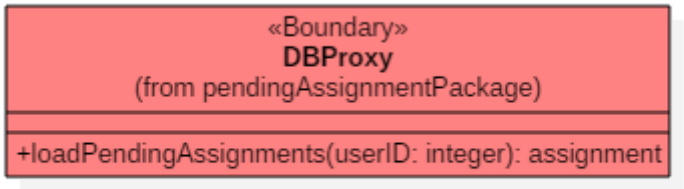
Controller pendingAssignmentControl



➔ **Class Operations:**

- pendingAssignmentControl(userID: integer): void: This is a construction operation for the class.
- AssigmentStatusCheck(UAL: String): void: Checks the status of an assignment.
- back(): void: returns to the submittedAssignmentGUI page.
- loadPendingAssignment(userID: integer, UAL:String): assignment: queries and returns the pending assignments by a specific user.

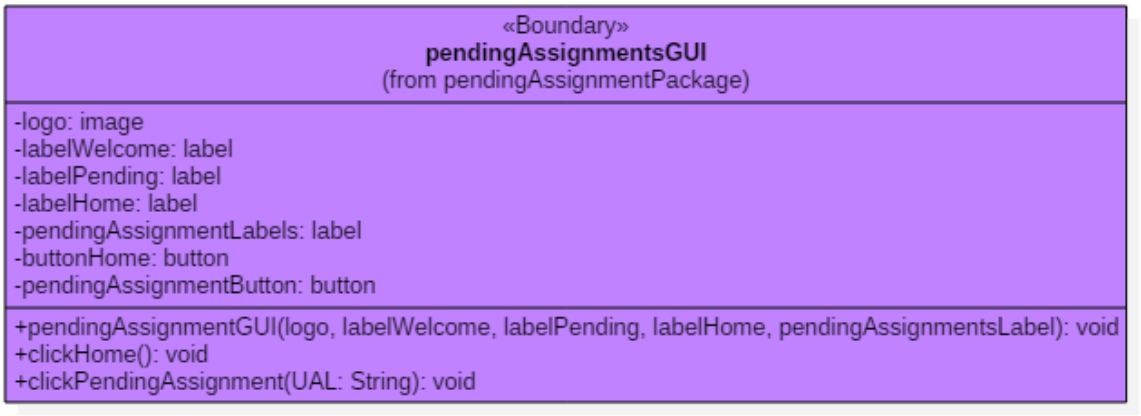
Boundary DBProxy



➔ **Class Operations:**

- loadPendingAssigments(userID: integer): assignment: This operation loads and returns a specific assignment

Boundary pendingAssignmentGUI



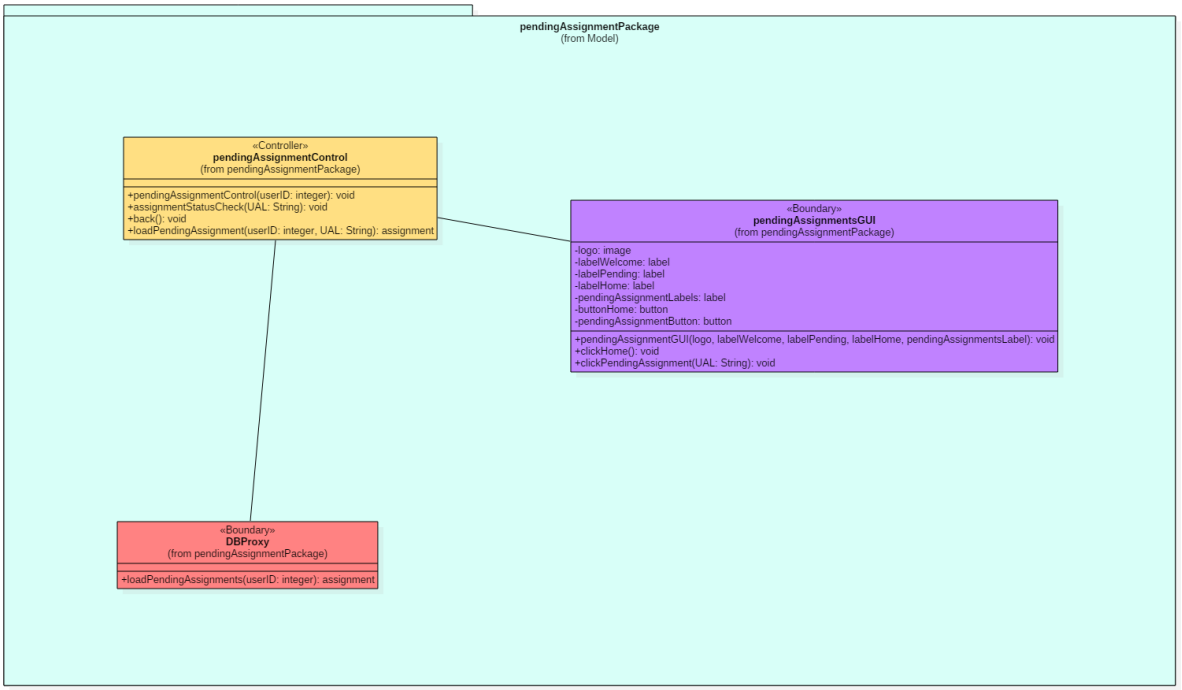
➔ **Class Attributes:**

- logo: image: image of the logo of the app
- labelWelcome: label: label containing a welcome message
- labelPeding: label: label title of the pending assignments list
- labelHome: label: homepage button label
- pendingAssignmentLabels: label: list of labels containing pending assignments
- buttonHome: button: homepage button
- pendingAssignmentButton: button to display a single pending assignment

➔ **Class Operations:**

- pendingAssignmentGUI(logo, labelWelcome, labelPending, labelHome, pendingAssignmentLabel): void. This is a construction operation for the class
- clickHome(): void: returns to the homepage by calling the diplayHomePage operation from the pageControl controller
- clickPendingAssignment(UAL: String): void: displays a single assignment

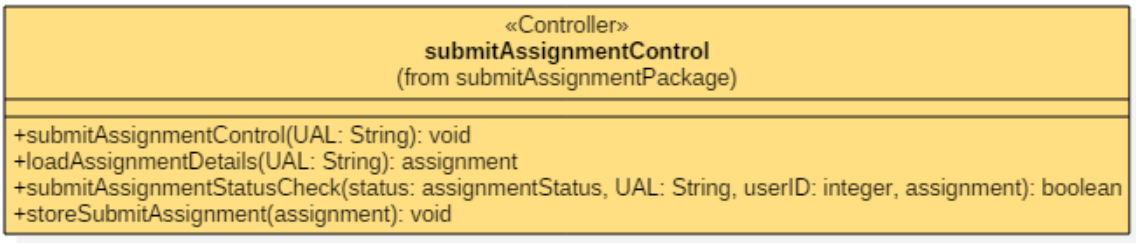
☐ **Class Diagram:**



4.3.5 submitAssignmentPackage

This package includes the classes that allow the user to view a single assignment the user has previously completed and the necessary controls to ensure that the assignments belong to the user and handle the submission of the assignment.

Controller submitAssignmentControl

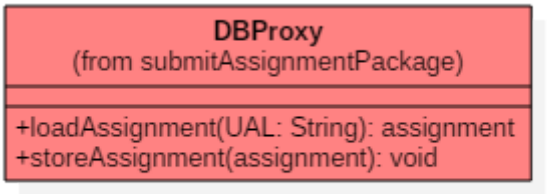


➔ **Class Operations:**

- submitAssignmentControl(UAL: String): void: This is a construction operation for the class
- loadAssignmentDetails(UAL: String): assignment: Load the assignments entity fields
- submitAssignmentStatusCheck(status:assignmentStatus, UAL: String, userID: integer, assignment): boolean: Checks whether the assignments is submitted

- storeSubmitAssignment(assignment): void: Stores the assignment

DBProxy



➔ **Class Operations:**

- loadAssignment(UAL: String): assignment: Queries and returns a specific assignment entity
- storeAssignment(assignment): void: Stores an assignments in the database

Boundary submitAssignmentGUI

➔ **Class Attributes:**

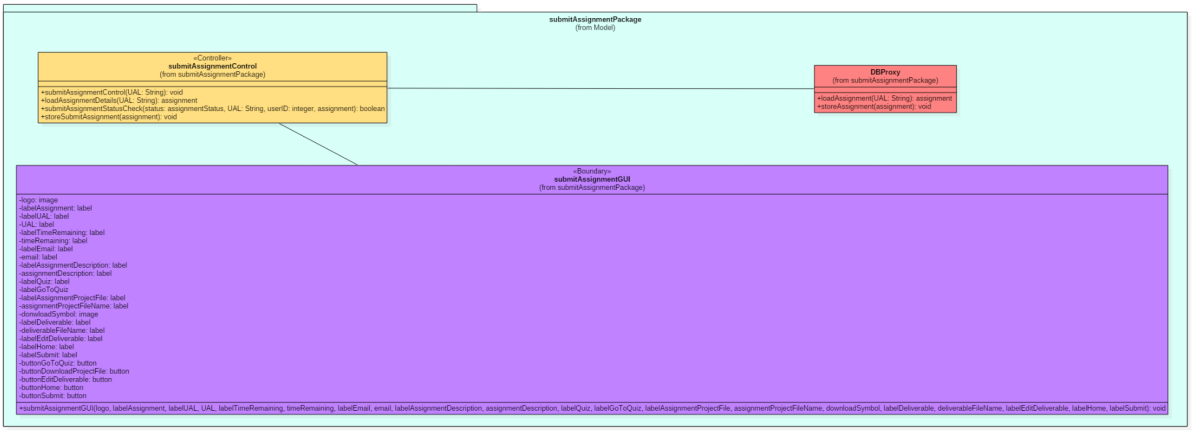
- logo: image: The logo of the app
- labelAssignment:label: label containing the title of the assignment
- labelUAL: label: Label containing the UAL field description
- UAL: label: Label containing the UAL of the assignment
- labelTimeRemaining: Label containing the timeRemaining field description
- timeRemaining: label: Label containing the remaining time left to submit the assignment
- labelEmail: label: Label containing the email field description
- email: label: Label containing the email of the manager
- labelAssignmentDescription: label: Label containing the description of the assignmentDescription field
- assignmentDescription: label: label containing the assignment description

- labelQuiz: label: Label containing the quiz field description
- labelGoToQuiz: Label containing the “Go To Quiz” string
- labelAssignmentProjectFile: label: Label containing the assignmentProjectFileName field description
- assignmentProjectFileName: label: Label containing the filename of the project assignment instructions file
- downloadSymbol: image: download icon next to the project instructions file
- labelDeliverable: label: Label containing the deliverableFileName field description
- deliverableFileName: label: Label containing the deliverable file name
- labelEditDeliverable: label of the buttonEditDeliverable
- labelHome: label: label of buttonHome
- labelSubmit: label: label of buttonSubmit
- buttonGoToQuiz: button: Button that launches the quiz of the assignment
- buttonDownloadProjectFile: button: Button that downloads the uploaded project description file
- buttonEditDeliverable: button: Button that allows the user to reupload a deliverable file
- buttonHome: button: Button that returns the user to the homepage
- buttonSubmit: button: Button that submits the assignment

➔ **Class Operations:**

- submitAssignmentGUI(logo, labelAssignment, labelUAL, UAL, labelTimeRemaining, timeRemaining, labelEmail, email, labelAssignmentDescription, assignmentDescription, labelQuiz, labelGoToQuiz, labelAssignmentProjectFile, assignmentProjectFileName, downloadSymbol, labelDeliverable, deliverableFileName, labelEditDeliverable, labelHome, labelSubmit): void: This is a construction operation for the class

☐ **Class Diagram:**


submitAssignmentPackage
(from Model)
«Controller»
submitAssignmentControl
(from submitAssignmentPackage)
+submitAssignmentControl(UAL: String): void
+loadAssignmentDetails(UAL: String): assignment
+submitAssignmentStatusCheck(status: assignmentStatus, UAL: String, userID: integer, assignment): boolean
+storeSubmitAssignment(assignment): void
DBProxy
(from submitAssignmentPackage)
+loadAssignment(UAL: String): assignment
+storeAssignment(assignment): void
«Boundary»
submitAssignmentGUI
(from submitAssignmentPackage)
-logo: image
-labelAssignment: label
-labelUAL: label
-UAL: label
-labelTimeRemaining: label
-timeRemaining: label
-labelEmail: label
-email: label
-labelAssignmentDescription: label
-assignmentDescription: label
-labelQuiz: label
-labelGoToQuiz
-labelAssignmentProjectFile: label
-assignmentProjectFileName: label
-donwloadSymbol: image
-labelDeliverable: label
-deliverableFileName: label
-labelEditDeliverable: label
-labelHome: label
-labelSubmit: label
-buttonGoToQuiz: button
-buttonDownloadProjectFile: button
-buttonEditDeliverable: button
-buttonHome: button
-buttonSubmit: button
+submitAssignmentGUI(logo, labelAssignment, labelUAL, UAL, labelTimeRemaining, timeRemaining, labelEmail, email, labelAssignmentDescription, assignmentDescription, labelQuiz, labelGoToQuiz, labelAssignmentProjectFile, assignmentProjectFileName, downloadSymbol, labelDeliverable, deliverableFileName, labelEditDeliverable, labelHome, labelSubmit): void

Annex I – Glossary

F.R: Functional requirement
N.F.R: Non Functional requirement
U.A.L: Unique assignment link, a one-time use personalized request link for the user
DoB: Date of Birth
GUI: Graphical User Interface

Annex II– Open Subjects

<Μια δυναμική λίστα με ανοιχτά θέματα απαιτήσεων, θα δημιουργηθεί στο παράρτημα Γ>