CoursesManagementApp

Sprint Report

The 10x'ers

Koureas Athanasios, 4392

Athanasios Papapostolou, 4147

Konstantinos Georgiou, 4333

VERSIONS HISTORY

Date	Version	Description	Author
26/3/2022	v0.1	Tooling and basic requirements	Koureas Athanasios
2/4/2022	v1.0	---------	Koureas Athanasios

1 Introduction

This document provides information concerning the **5th** sprint of the project.

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	Koureas Athanasios
Scrum Master	Konstantinos Georgiou
Development Team	Konstantinos Georgiou, Papapostolou Athanasios, Koureas Athanasios

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**19/3/2022**	**26/3/2022**	**1**	**NF-1**
2	**26/3/2022**	**9/4/2022**	**2**	**US-1,US-2,US-3**
3	**9/4/2022**	**16/4/2022**	**2**	**US-4,US-5,US-6**
4	**16/4/2022**	**30/4/2022**	**2**	**US-7,US-8,US-9**
5	**30/4/2022**	**14/5/2022**	**2**	**US-10,US-11,US-12**

2.3 Sprint Backlog

Requirement Id	Which Sprint
NF1	**Sprint No1**
US-1,US-2,US-3	**Sprint No2**
US-4,US-5,US-6	**Sprint No3**
US-7,US-8,US-9	**Sprint No4**
US-10,US-11,US-12	**Sprint No5**

3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

3.1 <Use Case 1>

Use case ID	Browse List
Data	List of courses
Actors	Professor
Main flow of events	1. User presses the ‘Browse’ button 2. User scrolls through the list of courses
Post conditions	1. User is presented with a list of courses retrieved from a DBMS 2. New courses are presented dynamically in a lazy list fashion

3.2 <Use Case 2>

Use case ID	Add Course
Data	List of courses Course id Course name Syllabus Year Semester
Actors	Professor
Main flow of events	1. User presses the ‘Add Course’ button on top of the list view 2. User inputs textfields of new course data 3. User confirms data and presses the ‘Add’ button 4. List of courses is reloaded
Post conditions	1. User is presented with a new course view with empty textfields with placeholders 2. Textfields save data 3. Data is saved according to a DBMS Schema 4. List of courses previews the updated list with the newly added object

3.3 <Use Case 3>

Use case ID	Remove Course
Data	List of courses Course id
Actors	Professor
Main flow of events	1. User presses the 'Remove Course' button next to a course item 2. User presses the 'OK' button 3. List of courses is reloaded (reactively)
Post conditions	1. List item is sliced over and reveals the 'OK' button 2. Course id is sent to the DBMS to delete 3. List of courses previews the updated list without the deleted object

3.4 <Use Case 4>

Use case ID	Browse Student List
Data	<>
Actors	Professor
Pre Conditions	1. Should be viewing a particular course item
Main flow of events	1. Professor issues "Browse Students" command 2. System selects students list from the DBMS 3. User is presented with list-view of all students enrolled in the course
No Students Error	1. System delivers error message "Oops, no students enrolled yet to user"
Post conditions	<>

3.5 <Use Case 5>

Use case ID	Add Student Details
Data	Student id Student name Year of registration Semester
Actors	Professor
Pre Conditions	<>
Main flow of events	1. Professor issues “Add details” command 2. Professor fills in the data textfields accordingly 3. Professor presses the “OK” button 4. System repeats the “Browse Student List” use case
Field not filled	1. System notifies professor that a text field is not filled with data 2. System does not proceed and prompts professor to fill in the remaining fields
Exiting Student	1. System notifies the professor that the student attempting to add exists in the list 2. System repeats “Add Student Details”
Post conditions	1. List of students is updated with the new item

3.6 <Use Case 6>

Use case ID	Remove Student
Data	Student item
Actors	Professor
Pre Conditions	1. Should be viewing a particular course item 2. Should exist at least one student item
Main flow of events	1. Professor issues “Delete Student” command 2.System issues prompts professor with an “Are you Sure” alert 3. Professor confirms the deletion of an item 4. List of students removes the selected item
Post conditions	1. List of students is updated without the recently removed student item

3.7 <Use Case 7>

Use case ID	Update Student Details
Data	Student id Student name Year of Registration Semester
Actors	Professor
Pre Conditions	Student item we want to update should exist
Main flow of events	1. Professor issues “Update Student” command 2.System prompts professor with form 3. Professor fills in the data textfields accordingly 4. Professor presses the “OK” button 5. System repeats the “Browse Student List” use case
Post conditions	1. List of students is updated with the new student data

3.8 <Use Case 8>

Use case ID	Add Course Details
Data	Course id Course name Syllabus Year Semester
Actors	Professor
Pre Conditions	1. Professor should have issued the “Add” command
Main flow ofevents	1. Professor fills in the data textfields accordingly 2. Professor presses the “OK” button 3. System repeats the “Press Browse Button” use case
Field not filled	1. System notifies professor that a text field in not filled with data 2. System does not proceed and prompts professor to fill in theremaining fields
Exiting Course	1. System notifies professor that the course attempting to add exists in the list 2. System repeats the “Add Course Details”
Post conditions	1. List of courses is updated with the new item

3.9 <Use Case 9>

Use case ID	Press Browse Button
Data	<>
Actors	Professor
Pre Conditions	<>
Main flow of events	1. Professor issues “Browse” command 2. Systems selects course list from the DBMS 3. User is presented with a list-view of all courses
No Course Error	1. System delivers error message “Oops, no course found to user”
Post conditions	<>

3.10 <Use Case 10>

Use case ID	Press Remove Button
Data	Course item
Actors	Professor
Pre Conditions	Should exist at least one course item
Main flow of events	1. Professor issues “Delete” command 2.System issues prompts professor with an “Are you Sure” alert 3. Professor confirms the deletion of an item 4. List of courses reactively removes the selected item
Post conditions	1. List of courses is updated without the recently removed course item

3.11 <Use Case 11>

Use case ID	Update Course Description
Data	Course id Course name Syllabus Year Semester
Actors	Professor
Pre Conditions	Course item we want to update should exist
Main flow of events	1. Professor issues “Update” command 2.System issues prompts professor form 3. Professor fills in the data textfields accordingly 4. Professor presses the “OK” button 5. System repeats the “Press Browse Button” use case
Post conditions	1. List of courses is updated with the new course data

3.12 <Use Case 12>

Use case ID	Register Grades
Data	Grades id Project grade Exam grade Semester
Actors	Professor
Pre Conditions	1. Course object should exist 2. Student list should have at least one item
Main flow of events	1. Professor issues the "Add Grades" command 2. Professor fills in the data textfields accordingly 3. Professor presses the "OK" button 4. System refreshes the Student item view
Fields not filled	1. System notifies professor that a text field is not filled with data 2. System does not proceed and prompts professor to fill in the remaining fields
Post conditions	1. Grades of student should get added for the current semester

3.13 <Use Case 13>

Use case ID	Professor Register
Data	Username Password
Actors	Professor
Pre Conditions	<>
Main flow of events	1. Professor fills in the username and password textfields accordingly 2. Professor presses the “Register” button 3. System saves registration data in DBMS 4. System execute the “Login” use case with same data
Field not filled	1. System notifies professor that a text field is not filled with data 2. System does not proceed and prompts professor to fill in the remaining fields
Invalid Data Student	1. System finds an error in the provided data 2. System does not proceed and prompts professor to fill in the remaining fields
User Exists	1. System notifies professor that the registration he is trying to create already exists 2. System executes “Login” use case with same data
Post conditions	1. List of professor is updated with the new item

3.14 <Use Case 14>

Use case ID	Professor Login
Data	Username Password
Actors	Professor
Pre Conditions	1. Registration token with according data should exist
Main flow of events	1. Professor fills in the username and password textfields accordingly 2. Professor presses the “Login” button 3. System views the main page
Fields not filled	1. System notifies professor that a text field is not filled with data 2. System does not proceed and prompts professor to fill in the remaining fields
Post conditions	<>

4 Design

4.1 Architecture

<Specify the overall architecture for this release in terms of a **UML package diagram**.>

4.2 Design

<Specify the detailed design for this release in terms of **UML class diagrams**.>

<Document the classes that are included in this release in terms of CRC cards according to the template that is given below.>

Class Name:	
Responsibilities: ▪ ... ▪ .... ▪ ...	**Collaborations:** ▪ ... ▪ .... ▪ ....

Class Name:	
Responsibilities: ▪ ... ▪ .... ▪ ...	**Collaborations:** ▪ ... ▪ .... ▪ ....

....

....