Software Engineering

2022./2023.

LearningTogether

Documentation, Rev. 1

Group:

MANZANA

Leader: *Alexander Varling*

Date: 18. 11. 2022.

Teacher: *Nikolina Frid*

Contents

1. Documentation change log

Continuous updating

Rev.	**Change description**	**Authors**	**Date**
0.1	Defined functional requirements	Kamil Konefeld, Agnieszka Grzymska, Jakub Kubiak	17/10/2022
0.2	Described use cases	Kamil Konefeld, Jakub Kubiak	02/11/2022
0.6	Created use case diagram	Agnieszka Grzymska	06/11/2022
0.8	Described non-functional requirements	Kamil Konefeld	06/11/2022
0.10	Added and described sequence diagrams	Agnieszka Grzymska	10/11/2022
0.11	Added database diagram	Kamil Konefeld	14/11/2022
0.12	Added tables description	Jakub Kubiak	17/11/2022
0.13	Added project description	Claire Flori	17/11/2022
0.13	Added class diagram	Inès Zemri	17/11/2022
0.15	Description of system architecture	Jorge Milla	18/11/2022
1.0			

Continued on next page

Continued from previous page

Rev.	**Change description**	**Authors**	**Date**
1.1			
1.2			
1.3			
1.5			
1.5.1			
2.0			

2. Project assignment description

Learning Together is a new web platform for knowledge exchange. The idea of this platform is that anyone can create their own course on any topic and upload written, video and audio materials. In addition, this platform enables interactive communication with the person holding the course through comments and live chats.

An unregistered public user can view the courses that are available, but to access the course, it is necessary to register with an email address. Registered users can browse available courses and enroll in any available course. Some courses may require payment.

To create a new course, the following information is required: name, category (choose one of the existing ones in the system), an estimate of how much time the average user needs to master all the materials, price (optional) and the course content. The content of the course is divided into points (teaching units). Each unit, along with a mandatory short description, can contain written, audio and video materials that registered users can download (the materials are not available to users who are not course participants). The course owner can add or remove materials at any time. The owner of the course can enable the option of online chat with participants and indicate the dates and times when it will be available and the maximum number of participants. Interested course participants can register for the offered dates . Enrolled course participants can comment and rate the course. Comments and ratings are publicly visible (also available to non-registered users).

Registered users will have a private and a public profile. Private profile will contain users' personal information, payment details and information about courses taken and/or created. Also, on their private profile, users can select one or more categories of courses that they are interested to get recommendations and updates. A public profile is required only for users who create and offer courses to other users. The application is maintained by system administrators who can add or delete any user, change the category of one of the existing courses and delete any course.

Learning Together platform shall be implemented as a web or mobile application. The back end must be implemented using object-oriented programming languages such as Java, C, etc. HTML, CSS, and JavaScript can be used for front-end development. Python, PHP, and various JavaScript frameworks (e.g., Node.js) are only allowed if they are used respecting the OO paradigm.

The application must have a responsive user interface so that it can be displayed with the same quality on the computer screen as on the screens of smartphones and tablets. All personal data in the application must be stored in accordance with the GDPR regulation.

The goal of our project is to create the Learning Together platform, in this case a website, using the methods of software engineering seen in class. The project will be composed of a theoretical study of the subject, and the creation, using programming, of the web platform. The theoretical study is made of three different diagrams, the functional requirements, and the description of our system architecture, meaning a description of the database and of the programing language used to create the website. The project will be created using object-oriented programming language.

The diagrams used are the use case diagram, the sequence diagram, and the class diagram. To represent all the possibilities of our platform, there are several sequence diagrams. The database used is created with the MySQL server. The point of the database will be to store all the information about the different users of our platform, it will evolve with the platform when users and classes are added or deleted. The programming language chosen is Python 3. It was recommended and the most known language in the team.

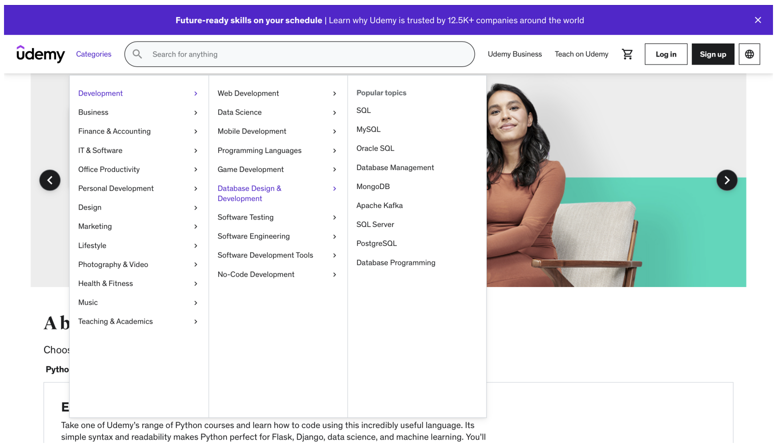
In the end the platform should be able to show unregistered users the classes and comments, the registered users should be able to create classes, and do everything linked to creating a course, and enroll in classes and comment and rate it. Finally, the administrators should be able to delete and chance courses and users. The platform would be created to help people learn something new online using different means of learning.

The benefits of such a project are that it helps people from all background to better themselves. It is a learning platform where the users can both enroll in a class and give on. Furthermore, given the different forms of learning available, the enrolled student can easily find a way of learning that suits them.

Learning together is like other learning platforms such as: Udemy, Coursea or any learning platform online. However, the difference with the project at hand is that the users can only enroll in a course, they can't teach a course as well.

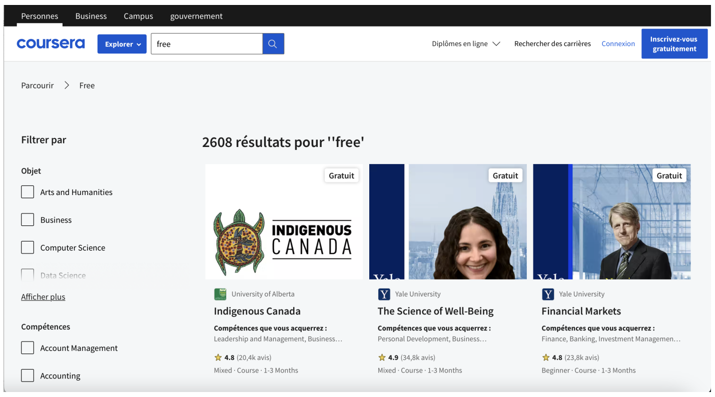
Generally, the users of the platform would be students wanting to learn more about some school subject or students that are home schooled. The other users that might be interested are the people how want to better their skills and be more educated so that they can have a promotion.

The possible customizations are that the users can have the courses they are taking on the front page and have the lecture material easily accessible and the graces easily viewable. Then they should be the recommendation bellow the courses followed by the user.



Take one of Udemy's range of Python courses and learn how to code using this incredibly useful language. Its simple syntax and readability makes Python perfect for Flask, Django, data science, and machine learning. You'll

Figure 2.1: Example of the Udemy Webpage



2608 résultats pour ''free'

Figure 2.2: Example of the Coursea Webpage

3. Software specification

3.1 Functional requirements

Stakeholders:

1. Students
2. Teachers
3. Companies
4. Developers
5. Administrators
6. Team Leaders
7. Google Calendar API Provider

Actors and their functional requirements:

1. Registered user can:
    - (a) browse the courses
    - (b) enroll
    - (c) pay for them
    - (d) create courses
    - (e) edit the profile
        - i. select interesting categories of courses - private
    - (f) see comments and rating of courses
        - i. select interesting categories of courses - private
2. Unregistered user can:
    - (a) browse the courses
    - (b) register
    - (c) see comments and rating of courses

3. Course Owner (registered user) can:
   (a) add materials to the course
   (b) remove materials from the course
   (c) enable online chat
4. Course Participant can:
   (a) comment and rate the course
   (b) access course materials
   (c) register for offered dates
5. System Administrator can:
   (a) add or delete any user
   (b) change category of existing course
   (c) delete any course

3.1.1 Use cases

UC1 - Register

- **Main participant:** Unregistered user
- **Goal:** Get acces to all possibilities of the web application
- **Participants:** Database
- **Prerequisites:** None
- **Description of the basic course:**
  1. User inputs name, surname, username, e-mail and password
     (a) E-mail gets verified if is unique
     (b) Password is checked if it passes security requirements
  2. Registered user is redirected to the main page
- **Description of possible deviations:**

  2.a User is being informed if there is already a user that has registered with that username or e-mail
    1. User has to change the username/e-mail

  2.b Password does not meet security requirements
    1. User has to choose diffrent password

UC2 - Login

- **Main participant:** Registered user
- **Goal:** Get access to all features intended for registered users
- **Participants:** Database
- **Description of the basic course:**
  1. User inputs username or e-mail and password
  2. Login info is validated
  3. Access to the functionalities intended for registered users is given
- **Description of possible deviations:**
  1. Main actor is being notified for invalid info which does not match any registered accounts data from database

     1.a Main actor tries again to provide with correct login information

     1.b Main actor abandons login window

  UC3 - Edit User Profile

  - **Main participant:** Registered user

- **Goal:** Manage user's profile
- **Participants:** Database
- **Prerequisites:** User should be registered and logged in
- **Description of the basic course:**
  1. Logged user selects edit profile option
  2. User's information is shown and ready to edit
  3. User can choose which information to edit
  4. User can save the changes
  5. After changes user is redirected to his profile
- **Description of possible deviations:**

  4.a User tries to leave without saving changes
    1. Application prompts request to save the changes
    2. User leaves without saving changes

<u>UC4 - Browse the courses</u>

- **Main participant:** Registered user, Unregistered user
- **Goal:** Browse for the courses
- **Participants:** Database
- **Prerequisites:** Any course should be available
- **Description of the basic course:**
  1. User selects the search tab
  2. Categories of courses are being displayed to the user
  3. User can choose interesting category
  4. Courses from this category are being displayed
  5. User can click on the course to see more information
- **Description of possible deviations:**

  4.a Category of courses is empty
    1. Information is being displayed
    2. User can go back to all categories

<u>UC5 - Enroll</u>

- **Main participant:** Registered user
- **Goal:** Get access to all contents available in the area of specific course
- **Participants:** Database
- **Prerequisites:** User should have already registered account
- **Description of the basic course:**

1. User begins the process of enrolling for the desired course
2. After paying predefined price for the course, user is already enrolled to the course

– **Description of possible deviations:**
1. Main actor failed to complete the payment what makes it impossible to gain access to the assets of the course

UC6 - Pay for the courses

– **Main participant:** Registered user
– **Goal:** Pay for the course
– **Participants:** Database
– **Prerequisites:** Any not free course should be availabe, Enroll option should be working
– **Description of the basic course:**
1. Price is being displayed next to enroll button
2. After clicking enroll user is being redirected to third-party paying website

– **Description of possible deviations:**
2.a Third-party website is not working properly
    1. Information is being displayed, enroll button is turned off

UC7 - Create the courses

– **Main participant:** Registered user
– **Goal:** Create the course
– **Participants:** Database
– **Prerequisites:** User must be logged into their account
– **Description of the basic course:**
1. Main actor is being redirected to the page providing him with the tools for creating the course
2. After filling all obligatory fields, main actor can provide the course with any file they are willing to post
3. Main actor can choose between posting the course for free or for the desired price they are willing to be paid
4. Main actor can abort the process of creating the course

– **Description of possible deviations:**
1. Main actor fails to fill all the obligatory fields

UC8 - Comment and rate the course

- **Main participant:** Enrolled user
- **Goal:** Rate the course and comment opinion
- **Participants:** Database
- **Prerequisites:** User needs to be enrolled into course (UC5)
- **Description of the basic course:**
  1. Rate button is displayed on the course page
  2. After clicking rate button, user is redirected to the rating page
  3. After answering questions user can click submit button
  4. After submitting user is redirected to course page
- **Description of possible deviations:**

  3.a User tries to leave without submitting form
    1. Application prompts request to submit the answers
    2. User leaves without submitting - rating and comment are gone

UC9 - See comments and ratings of the course

- **Main participant:** Registered and unregistered users
- **Goal:** Check the quality of the course
- **Participants:** Database
- **Prerequisites:**None
- **Description of the basic course:**
  1. User can browse the opinions of other users who already had access to the course
- **Description of possible deviations:**
  1. There are neither comments nor rating existing

UC10 - Add materials to the course

- **Main participant:** Course owner, Administrator
- **Goal:** Add materials to the course, update course with new stuff
- **Participants:** Database
- **Prerequisites:** User needs to be course owner/administrator (UC7 - Creating the courses)
- **Description of the basic course:**
  1. Course owner selects add materials button

2. Course owner is being redirected to form
3. Course owner selects type of new materials (for example flashcards)
4. Course owner is provided with special form for flashcards
5. Course owner selects save button
6. Course owner is redirected to course page

- **Description of possible deviations:**

  5.a Course owner tries to leave without submitting form

    1. Application prompts request to save the changes
    2. Course owner leaves without saving - all new materials are gone

<u>UC11 - Remove content from the course</u>

- **Main participant:** Admin and owner of the course
- **Goal:** Remove content of the course that is no longer valid or inappropriate
- **Participants:** Database
- **Prerequisites:** User needs to be the owner of the course (UC7)
- **Description of the basic course:**

  1. Main actor moves to the editing tool
  2. Main actor removes desired item from the course

- **Description of possible deviations:**
  1. The is no content to be removed, the course is empty

<u>UC12 - Enable online chat</u>

- **Main participant:** Owner of the course
- **Goal:** Allow participants of the course to contact the owner and discuss the desired content of the course
- **Participants:** Database
- **Prerequisites:** User needs to be enrolled to the course (UC5)
- **Description of the basic course:**

  1. Main actor enters the online chat intended for specific content of the course
  2. Main actor can add and see their thoughts and other participants of the course

- **Description of possible deviations:**
  1. The online chat is turned off

UC13 - Register for offered dates

- **Main participant:** Course Participant
- **Goal:** Register for offered dates for online chat
- **Participants:** Database
- **Prerequisites:** Online chat dates needs to be enabled (UC12)
- **Description of the basic course:**

  1. Course Participant select date from offered dates visible on course page
  2. Course Participant is redirected to third-party meeting application (for example discord, MS Teams)
  3. Participant data is signed into database

- **Description of possible deviations:**

  1.a Course Owner did not enable online chat

  1. Information is being displayed

UC14 - Add and remove users

- **Main participant:** Admin and owner of the account
- **Goal:** Remove no longer needed account or inappropriate user
- **Participants:** Database
- **Prerequisites:** Users need to be registered (UC1)
- **Description of the basic course:**

  1. Main actor proceeds to the user deleting procedure
  2. Main actor is being asked to confirm their decision
  3. The account is being deleted

  1.1 Admin removes desired account from the database

- **Description of possible deviations:**

Admin The account willing to be deleted no longer exists

UC15 - Change category of existing course

- **Main participant:** Administrator
- **Goal:** Change category of course
- **Participants:** Database
- **Prerequisites:** Course is created (UC7)
- **Description of the basic course:**

  1. Administrator selects course that he wants to change category of

2. Administrator selects new category out of droplist
3. Administrator saves the change

- **Description of possible deviations:**
  3.a Administrator wants to leave without saving changes
    1. Application prompts request to save the changes
    2. Administrator leaves without saving changes

UC16 - Delete any course

- **Main participant:** Administrator
- **Goal:** Delete any course
- **Participants:** Database
- **Prerequisites:** Course is created (UC7)
- **Description of the basic course:**
  1. Administrator selects course that he wants to delete
  2. Administrator clicks Delete button
- **Description of possible deviations:**
  2.a Administrator clicks delete button by mistake
    1. Application prompts request to confirm deletion

Use case diagrams

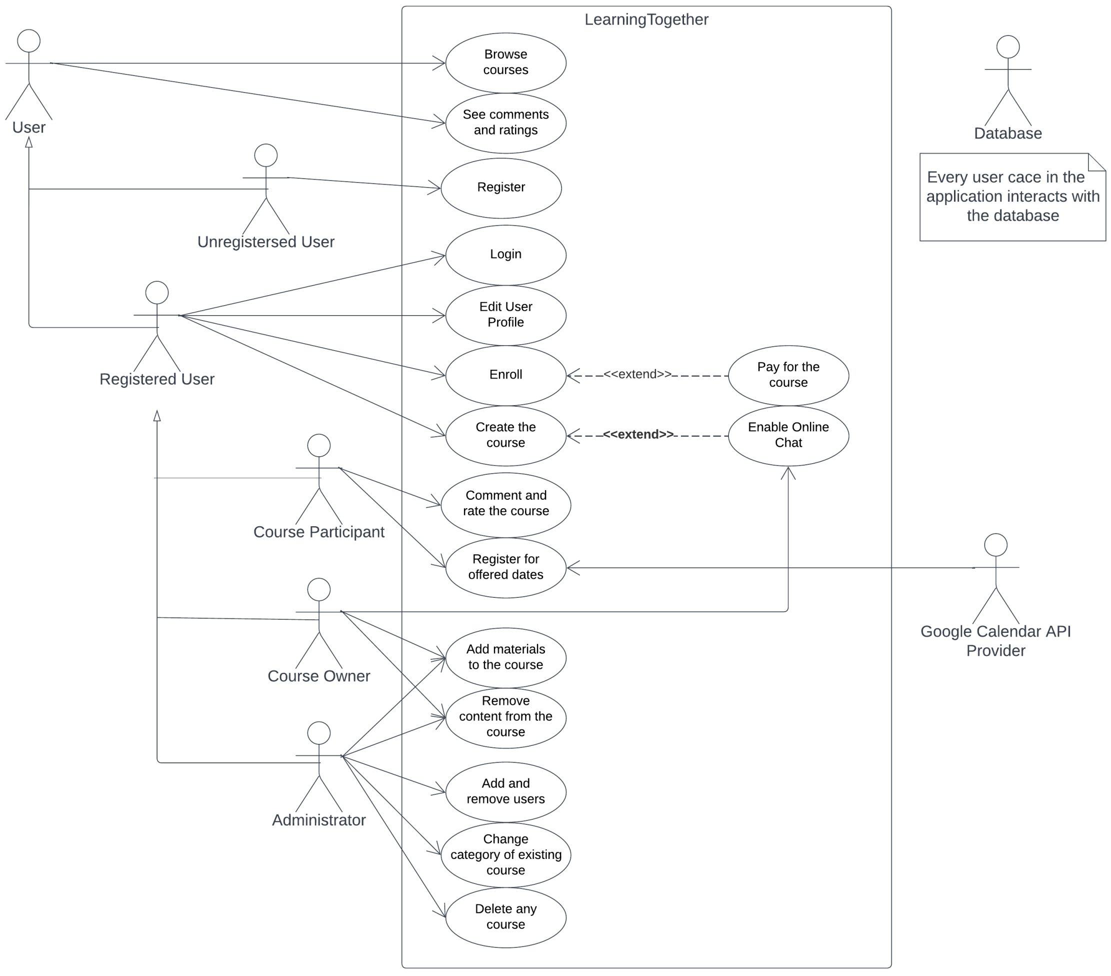


Figure 3.1: Use case diagram

3.1.2 Sequence diagrams

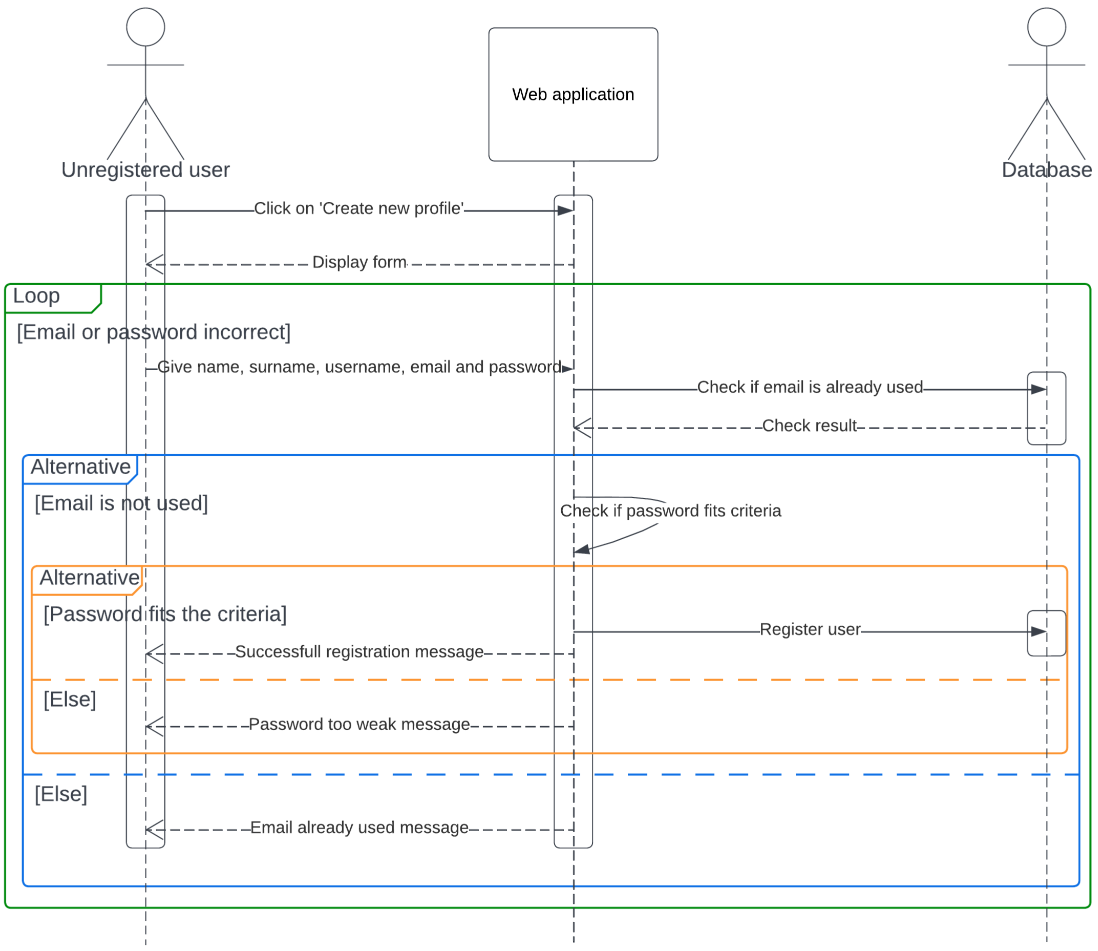

Unregistered user
Web application
Database
Click on 'Create new profile'
Display form
Loop
[Email or password incorrect]
Give name, surname, username, email and password
Check if email is already used
Check result
Alternative
[Email is not used]
Check if password fits criteria
Alternative
[Password fits the criteria]
Register user
Successfull registration message
[Else]
Password too weak message
[Else]
Email already used message

Figure 3.2: User registration diagram

The diagram above presents the process of registering new users. First the user clicks on a ”Create new profile” button. In response to that, application displays a form for them to fill out. User needs to specify their name, surname, username, email, and password. The system makes sure that the email has not been yet assigned to any profile and that the password fits safety criteria. Once both of this conditions are met, it puts the new user in the database.

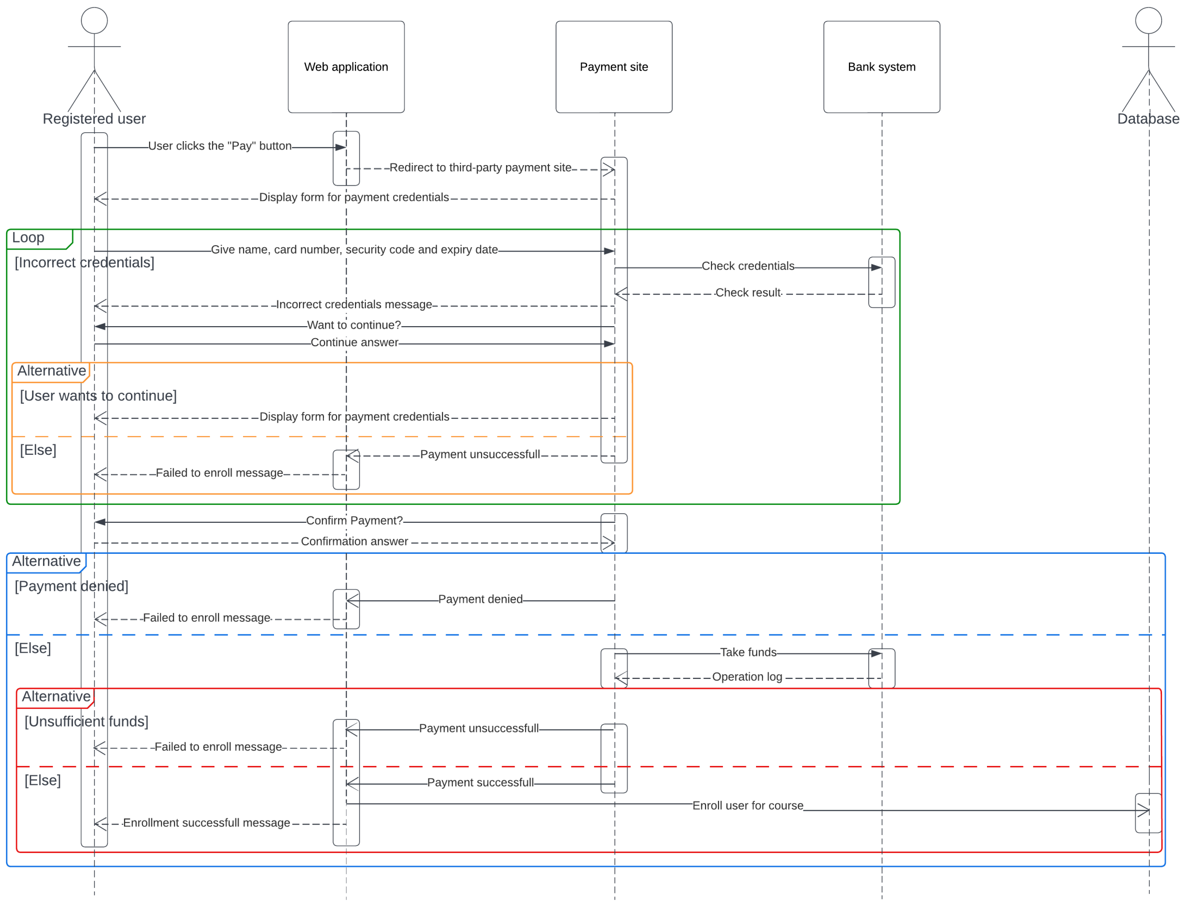


Figure 3.3: Course payment diagram

Next diagram shows how payment for a course proceeds. The actor in this process must be a registered (and logged in) user. Once they click on the ”Pay” button, our application redirects them to a third-party payment website. They need to enter their credentials (name, surname, credit card number, security number and card expiry date). If the information is confirmed to be correct by the bank system, there is a final prompt asking user if they want to proceed with the payment. If the credentials are not correct, user can try to enter them again or resign and therefore not be enrolled for the course. If the user chose to finalize the transaction, the site sends that request to the bank system. If there are sufficient funds on the user's account, the process ends successfully and the information about their enrollment is being saved in the database.

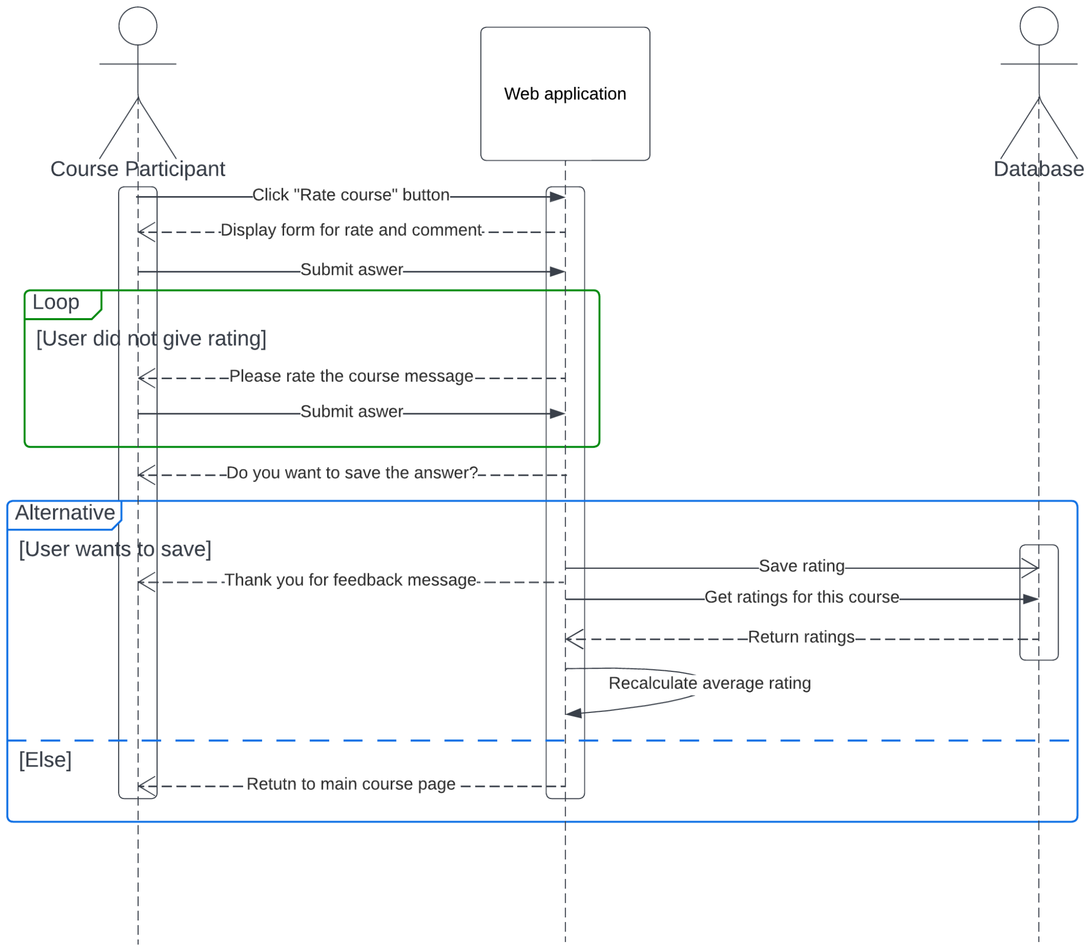

Course Participant
Web application
Database
Click "Rate course" button
Display form for rate and comment
Submit aswer
Loop
[User did not give rating]
Please rate the course message
Submit aswer
Do you want to save the answer?
Alternative
[User wants to save]
Save rating
Thank you for feedback message
Get ratings for this course
Return ratings
Recalculate average rating
[Else]
Retutn to main course page

Figure 3.4: Course rating diagram

The last diagram shows how users can rate and comment courses they took. User has to be logged in and enrolled to the given course. After clicking on the ”Rate course” button, the website displays a form asking for a 1 to 5 rating and optionally a comment. As long as the user tries to submit the answer without rating, they get a message asking to fill it out. Once they have done that, application saves their rating in the database and recalculates and updates average course rating.

3.2 Other requirements

The application will be only available in English. The system will not be hardly affected by increased number of users. The access to the system will be available through the most popular web browsers like : Chrome, Firefox, Safari, Brave or Opera. Application will also function on mobile counterparts of the abovementioned browsers. Application for mobile and computer use will be designed with the user's comfort in mind. It will also be very intuitive to work with.The application will be resistant to SQL-Injection, and sensitive data stored in the database will be properly encrypted. Application will also have e-mail registration authentication.

4. System architecture and design

In our web application backend and frontend are implemented independently of each other. For the client-side of the application, we use the Angular framework together with JavaScript. Whereas the backend is built using Python DjangoFramework. When it comes to the frontend, we use the architecture that is implemented in Angular. Angular Module declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities.

4.1 Database

We have created a relational database and used MySQL for the its implementation. Main tables that it consists of are User, UserPayment, Categories, Course and CourseGrade.

4.1.1 Table description

Administrator		
ID_USER	INT	ID of the admin
Moderator	VARCHAR	

User		
ID_USER	INT	ID of the user
email	VARCHAR	The email of the user
name	VARCHAR	The name of the user
surname	VARCHAR	The surname of the user

Continued on next page

Continued from previous page

User		
password	VARCHAR	The password of the user
Gender	VARCHAR	The sex of the user

UserPayment		
ID_Payment	INT	ID of the user's confirmed payment
ID_USER	INT	The user's ID
name_of_payment	VARCHAR	The title of the purchased item
data	VARCHAR	Data abou the payment, info about payment method etc.

FavCategory		
ID_CATEGORY	INT	ID of the category
ID_USER	INT	The user's ID

UserProfile		
ID_USER	INT	The user's ID
Picture	VARCHAR	The title of the purchased item
Description	VARCHAR	Data abou the payment, info about payment method etc.

Categories		
ID_CATEGORY	INT	ID of the category of a course
name	VARCHAR	The title of a course

Continued on next page

Continued from previous page

Categories		
description	VARCHAR	Description of the categorie

Course		
ID_COURSE	INT	ID of the course
ID_CATEGORY	INT	ID of the category
name	VARCHAR	Name of the course
description	VARCHAR	Description of the course
logo	VARCHAR	Logo image of the course
price	INT	Price of the course

OnlineChat		
ID_ONCH	INT	ID of the opened online chat
ID_COURSE	INT	ID of the course linked to online chat
date	DATE	Scheduled date of the online chat
link	VARCHAR	Link to the third-party(?) software providing online chat option
registerednumber	INT	

TeachingUnits		
ID_ TEACHINGUNIT	INT	ID of the teaching unit
ID_COURSE	INT	ID of the course
name	VARCHAR	Name of the teaching unit
description	VARCHAR	Description of the teaching unit

TUMaterials		
ID_MATERIAL	INT	ID of the teaching material
ID_TEACHINGUNIT	INT	ID of the teaching unit
material	VARCHAR	Name of the teaching material
explain	VARCHAR	Explanation of the teaching material

CourseOwners		
ID_COURSE	INT	ID of the course
ID_USER	INT	ID of the course owner's user
permission	VARCHAR	Permission for owning the course

CourseEnrolled		
ID_COURSE	INT	ID of the course
ID_USER	INT	ID of the enrolled user

CourseGrade		
ID_GRADE	INT	ID of the course's grade
ID_USER	INT	ID of the user enrolled to the course
ID_COURSE	INT	ID of the course
rate	INT	Rate given to the course
comment	VARCHAR	Comment left under the rate

4.2 Class diagram

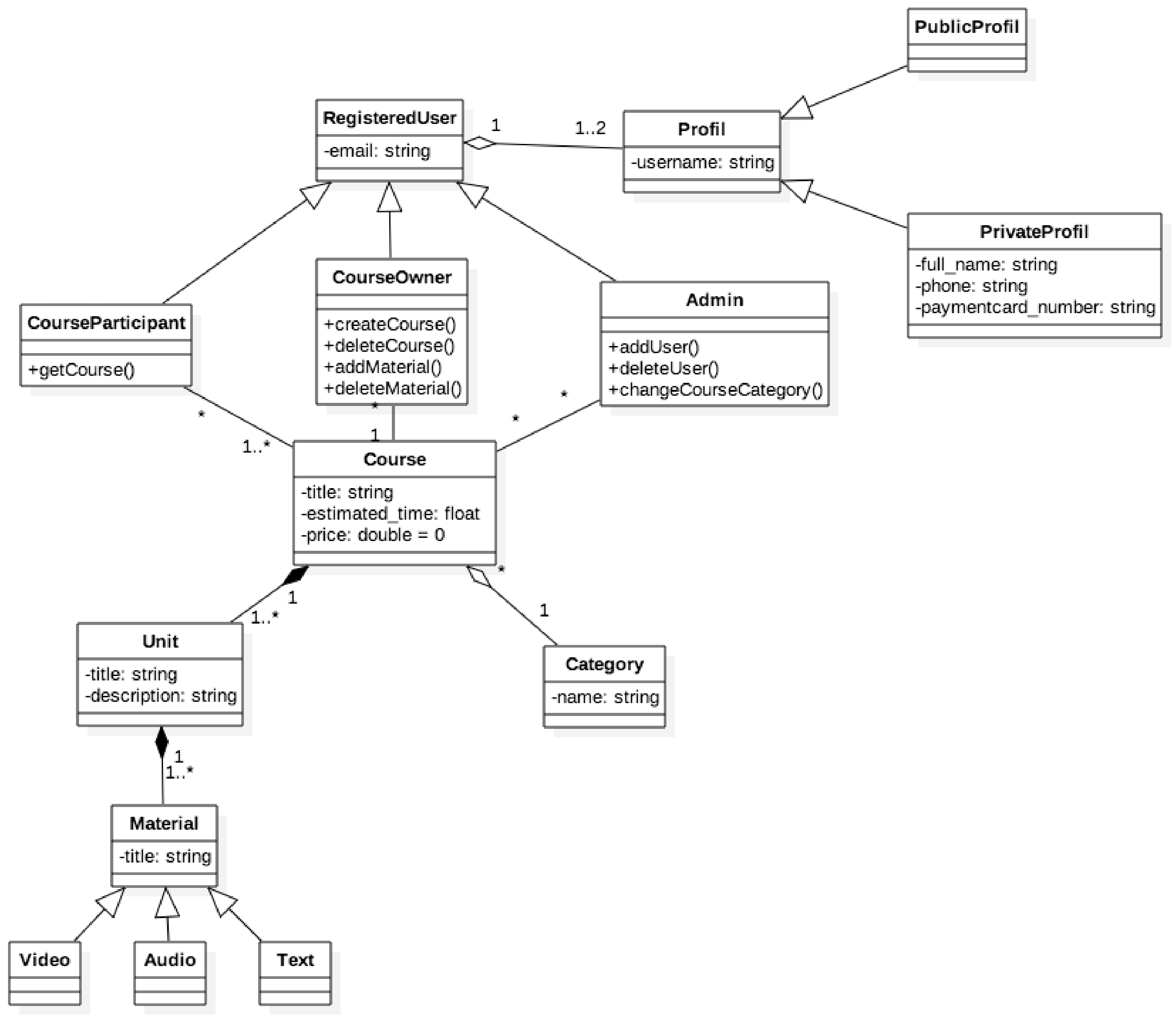


Figure 4.1: Class diagram

part of the 2nd revision

During the second submission of the project, the class diagram and descriptions must correspond to the actual state of implementation.

/newpage

4.3 State machine diagram

part of the 2nd revision

*It is necessary to attach a state diagram and describe it. One state diagram showing the **significant part of the functionality** of the system is sufficient. For example, user interface states and the flow of use of some key functionality are a significant part of the system, and registration and login are not.*

4.4 Activity diagram

part of the 2nd revision

It is necessary to enclose a attach of activities with a corresponding description. The activity diagram should show a significant part of the system.

4.5 Component diagram

part of the 2nd revision

A component diagram with the accompanying description must be attached. The component diagram should show the structure of the whole application.

5. Implementation and user interface

5.1 Tools and technologies

part of the 2nd revision

List in detail all technologies and tools used in the development of documentation and applications. Briefly describe them, and state their meaning and place of application. For each of the listed tools and technology it is necessary to ***specify internet link*** *where you can download or learn more about them.*

5.2 Software testing

part of the 2nd revision

In this chapter it is necessary to describe the implementation of testing of implemented functionalities at the level of components and at the level of the whole system with the presentation of selected test cases. Students should examine core functionality and boundary conditions.

5.2.1 Component testing

It is necessary to conduct unit testing on classes that implement basic functionalities. Develop a ***minimum 6 test cases*** *in which regular cases, boundary conditions, and exception throwing will be examined. It is also desirable to create a test case that uses functionalities that are not implemented. It is necessary to enclose the source code of all exam cases and a presentation of the results of the exam in the development environment (passing / failing the exam).*

5.2.2 System testing

The system test should be performed and described using the Selenium footnote `https://www.seleniumhq.org/` *framework. Develop a* ***minimum 4 test cases*** *that will examine regular cases, boundary conditions, and call functionality that is not implemented / cause an error to see how the system responds when something is not fully realized. The test case should consist of an input (eg username and password), the expected output or result, the test step and the output or result obtained.*

The creation of test cases using the Selenium framework can be performed using one of the following two tools:

- *browser extension* ***Selenium IDE*** *- recording user actions for automatic exam repetition*
- ***Selenium WebDriver*** *- support for writing exams in Java, PHP languages using a special programming interface*

Details on the use of the Selenium tool will be presented in a special lecture during the semester.

5.3 Deployment diagram

part of the 2nd revision

*You need to insert a **specification** layout diagram and describe it. It is possible to insert an instance layout diagram instead of a specification layout diagram, provided that this diagram better describes some important part of the system.*

5.4 Deployment instructions

part of the 2nd revision

In this chapter it is necessary to give instructions for deployment of the realized application. For example, for web applications, describe the process by which the source code leads to a fully set up database and server that responds to user queries. For a mobile application, the process by which the application is built and placed on one of the stores. For a desktop application, the process by which an application is installed on a computer. If mobile and desktop applications communicate with the server and / or database, describe the procedure for setting them up. When creating instructions, it is recommended that ***highlight installation steps using hints*** *and use* ***screenshots*** *as much as possible to make the instructions clear and easy to follow.*

The completed application must be running on a publicly available server. Students are encouraged to use one of the following free services: Amazon AWS, Microsoft Azure or Heroku. Mobile apps should be released on the F-Droid, Google Play or Amazon App Store.

6. Conclusion and future work

part of the 2nd revision

In this chapter it is necessary to write a review of the time of project assignment, what technical challenges have been identified, whether they have been solved or how they could be solved, what knowledge was acquired during project development, what knowledge would be especially needed for faster and better project implementation. and what would be the prospects for continuing work in the project team.

It is necessary to accurately list the functionalities that are not implemented in the realized application.

References

Continuous updating

List all references and literature that helped in the realization of the project.

1. Oblikovanje programske potpore, FER ZEMRIS, `http://www.fer.hr/predmet/opp`
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, `http://www.ece.rutgers.edu/~marsic/books/SE`
5. The Unified Modeling Language, `https://www.uml-diagrams.org/`
6. Astah Community, `http://astah.net/editions/uml-new`

Index of figures

Appendix: Group activity

Meeting log

Continuous updating

It's necessary to frequently update the meeting log according to the template.

1. meeting
    - Date: in this format: November 18, 2022
    - Attendees: J.Doe, ...
    - Meeting subjects:
        - subject description
        - subject description
2. meeting
    - Date: in this format: November 18, 2022
    - Attendees: J.Doe, ...
    - Meeting subjects:
        - subject description
        - subject description

Activity table

Continuous updating

Note: Activity contributions should be entered as hours contributed by person and activity.

	Team lead	**Team member**	**Ime Prezime**	**Team member**	**Team member**	**Team member**	**Team member**
Project management							
Project task description							
Functional requirements							
Individual patterns description							
Patterns diagram							
Sequence diagram							
Other requirements description							
System architecture and design							
Database							
Class diagram							
State diagram							
Activity diagram							
Components diagram							
Used technologies and tools							
Solution testing							
Layout diagram							
Deployment instructions							
Meeting log							

Continued on next page

Continued from previous page

	Team lead	**Team member**	**Ime Prezime**	**Team member**	**Team member**	**Team member**	**Team member**
Conclusion and future work							
References							
Additional task examples							
Welcome page creation							
Database creation							
Connecting to the database							
back end							

Change log diagrams

part of the second revision

Import generated change log diagrams from GitLab to this chapter. Diagrams can be reached at GitLab at Repository/Contributors.