



UNIVERSITAT DE  
BARCELONA

**Final degree project**

**Degree in Computer Engineering**

**Faculty of Mathematics and Computer Science  
University of Barcelona**

---

**Peer Evaluation System: A web application to  
provide peer evaluation for a workshop**

---

**Qijun Jin**

Director: Eloi Puertas Prats  
Realized at: Department of Mathematics  
and Computer Science  
Barcelona, 13 de June de 2022



## **Abstract**

For any academic organization, the link between student and professor is the main support on which academic knowledge is based. Students are the ones who take a set of subjects and will be evaluated along the course. In contrast, professors are the ones who will evaluate the students by a set of criteria for each assignment of the subject. This evaluation procedure is known as the standard way to provide a qualification to the student where the professor is the only one who can influence the qualification.

This project is focused on the workflow of the whole evaluation period of an activity where students not only participate in the task but also evaluate their peers to get scored after the task is finished. Its goal is to analyze the current workshop activity on the virtual campus of the University of Barcelona (based on Moodle) and then synthesize the core requirements of the system into the designed web application that serves as a third-party platform that is more intuitive for both students and professors to get into.

By this, a whole sequence of the evaluation will take place at a web application which is the platform where students can participate in the process of peer evaluation in the same task. While the professor is responsible for evaluating both the evaluations given by the students and the attached files.

**Keywords:** Distributed system; Web application; UX/UI; Flask; Python; Vue; CSS; HTML; Javascript; Peer evaluation.

## Resumen

Para cualquier organización académica, el vínculo entre estudiante y profesor es el principal soporte sobre el que se asienta el conocimiento académico. Los estudiantes son los que toman un conjunto de materias y serán evaluados a lo largo del curso. En cambio, los profesores son los que evaluarán a los alumnos mediante una serie de criterios para cada trabajo de la asignatura. Este procedimiento de evaluación se conoce como la forma estándar de otorgar una calificación al estudiante donde el profesor es el único que puede influir en la calificación.

Este proyecto se centra en el flujo de trabajo de todo el período de evaluación de una actividad en la que los estudiantes no solo participan en la tarea, sino que también evalúan a sus compañeros para obtener una puntuación una vez finalizada la tarea. Su objetivo es analizar la actividad actual del taller en el campus virtual de la Universidad de Barcelona (basado en Moodle) y luego sintetizar los requisitos básicos del sistema en la aplicación web diseñada que sirve como una plataforma de terceros más intuitiva para tanto estudiantes como profesores para entrar.

De esta manera, toda una secuencia de la evaluación se llevará a cabo en una aplicación web que es la plataforma donde los estudiantes pueden participar en el proceso de evaluación por pares en la misma tarea. Mientras que el profesor es el responsable de valorar tanto las valoraciones dadas por los alumnos como los archivos adjuntos.

**Palabras claves:** Sistema distribuido; Aplicación web; UX/UI; Flask; Python; Vue; CSS; HTML; Javascript; Evaluación por pares.

## Resum

Per a qualsevol organització acadèmica, la vinculació entre estudiant i professor és el principal suport en què es fonamenta el coneixement acadèmic. Els estudiants són els que cursen un conjunt d'assignatures i seran avaluats al llarg del curs. En canvi, els professors són els que avaluaran els estudiants mitjançant un conjunt de criteris per a cada treball de l'assignatura. Aquest procediment d'avaluació es coneix com la forma estàndard d'oferir una qualificació a l'estudiant on el professor és l'únic que pot influir en la qualificació.

Aquest projecte se centra en el flux de treball de tot el període d'avaluació d'una activitat on els estudiants no només participen en la tasca sinó que també avaluen els seus companys per obtenir una puntuació un cop finalitzada la tasca. El seu objectiu és analitzar l'activitat actual del taller al campus virtual de la Universitat de Barcelona (basat en Moodle) i després sintetitzar els requisits bàsics del sistema en l'aplicació web dissenyada que serveix com a plataforma de tercers més intuïtiva per tant estudiants com professors per entrar-hi.

D'aquesta manera, tota una seqüència de l'avaluació es durà a terme en una aplicació web que és la plataforma on els estudiants poden participar en el procés d'avaluació entre iguals en la mateixa tasca. Mentre que el professor és l'encarregat d'avaluar tant les avaluacions fetes pels estudiants com les fitxes adjuntes.

**Paraules claus:** Sistema distribuït; Aplicació web; UX/UI; Flask; Vue; CSS; HTML; Javascript; Avaluació per parells.



## **Acknowledgments**

I would like to thank Dr. Eloi Puertas Prats, the tutor of this fieldwork, for all the help and support he has given to me during the completion of my project, for encouraging me to be strong, and for the constant availability of the meetings which guided me through the project with good results.

To all the professors and department of the Faculty of Mathematics and Computer Science of the University of Barcelona who have made my training in the field of Computer Science possible and taught me to work in a team and think to adapt to any situation, thus making me a valuable piece in any situation future project.

Thank you very much to all of you.

## Table of Contents

1. Introduction .....	1
1.1. Motivation .....	1
1.2. Context .....	2
1.3. State of art .....	2
1.4. Objectives .....	4
1.4.1. Main objectives .....	4
1.4.2. Specific objectives .....	4
1.4.2.1. Front-end level .....	4
1.4.2.2. Back-end level .....	5
1.5. Document structure .....	5
2. Research .....	7
2.1. Analysis .....	7
2.2. Requirements .....	9
2.2.1. Administrator requirements .....	9
2.2.2. Professor requirements .....	9
2.2.3. Student requirements .....	9
2.3. Product backlog .....	10
3. Planning .....	13
3.1. Timing .....	13
3.2. Sprints .....	14
4. Technologies .....	17
4.1. Programming languages and other languages .....	17
4.1.1. HTML (Hypertext Markup Language) .....	17
4.1.2. CSS (Cascading Style Sheets) .....	17
4.1.3. JavaScript .....	17
4.1.4. Python .....	18
4.2. Frameworks .....	18
4.2.1. Vue .....	18
4.2.2. Vuex .....	19
4.2.3. Flask .....	20
4.3. Platform .....	21
4.3.1. Heroku .....	21
4.3.2. GitHub .....	21
5. Design .....	23
5.1. Distributed application architecture .....	23
5.2. Branding .....	24
5.3. Client .....	25



5.3.1.	Home page .....	25
5.3.2.	Auth page .....	26
5.3.3.	Student page .....	26
5.3.4.	Professor page .....	28
5.3.5.	Admin page.....	30
5.4.	Server .....	31
5.4.1.	Configuration class .....	31
5.4.2.	Resources.....	31
5.4.3.	RESTful API.....	32
5.4.4.	Models .....	33
5.4.5.	Database entities .....	35
5.4.6.	Database relationships .....	39
5.4.7.	Security metrics .....	40
6.	Implementation .....	41
6.1.	Vue Router.....	41
6.2.	Stateness on the selected component.....	43
6.3.	Pairing algorithm.....	44
6.4.	Grading sheet .....	45
7.	Deployment .....	47
8.	Testing and Results .....	49
8.1.	Testing .....	49
8.2.	Results.....	50
9.	Costs.....	51
10.	Future works .....	53
10.1.	Learning Tools Interoperability (LTI).....	53
10.2.	Extend session .....	54
10.3.	More security metrics.....	54
11.	Conclusion .....	55
12.	References .....	57
13.	Annex.....	61
13.1.	User stories.....	61
13.2.	API documentation .....	73
13.3.	Web page results.....	85
13.3.1.	App module.....	85
13.3.2.	Auth module.....	89
13.3.3.	Admin module.....	90
13.3.4.	Professor module.....	91
13.3.5.	Student module.....	95

## Figures

Figure 1 - Workshop activity (Source: <a href="http://www.moodle.org">www.moodle.org</a> ) .....	2
Figure 2 - Website of <a href="http://peerassessment.com">peerassessment.com</a> (Source: <a href="http://peerassessment.com">peerassessment.com</a> ) .....	3
Figure 3 - Website of <a href="http://peergrade.io">peergrade.io</a> (Source: <a href="http://www.peergrade.io">www.peergrade.io</a> ) .....	3
Figure 4 - Workflow of UB's workshop from the student side .....	7
Figure 5 - Workflow of UB's workshop from professor side .....	7
Figure 6 - Workflow of the <a href="http://peerassessment.com">peerassessment.com</a> application (Source: <a href="http://peerassessment.com">peerassessment.com</a> ) .....	8
Figure 7 - Workflow of the <a href="http://www.peergrade.io">www.peergrade.io</a> application (Source: <a href="http://www.peergrade.io">www.peergrade.io</a> ) .....	8
Figure 8 - Gantt diagram of the sprints .....	15
Figure 9 - Data-flow of Vuex (Source: <a href="http://vuex.vuejs.org">vuex.vuejs.org</a> ) .....	19
Figure 10 - State management of Vuex (Source: <a href="http://vuex.vuejs.org">vuex.vuejs.org</a> ) .....	20
Figure 11 - Diagram of a client-server architecture (Source: <a href="http://www.wikipedia.com">www.wikipedia.com</a> ) .....	23
Figure 12 - Logotype of the application .....	24
Figure 13 - Sketch proposal for the home page .....	25
Figure 14 - Sketch proposal for the auth page .....	26
Figure 15 - Sketch proposal for the attach file subpage .....	27
Figure 16 - Sketch proposal for the evaluate peers subpage .....	27
Figure 17 - Sketch proposal for the review evaluations subpage .....	27
Figure 18 - Sketch proposal for the conclude task page .....	28
Figure 19 - Sketch proposal for the original set up task subpage .....	29
Figure 20 - Sketch proposal for the pair peers subpage .....	29
Figure 21 - Sketch proposal for the qualify peers subpage .....	29
Figure 22 - Sketch proposal for the retrospect task subpage .....	30
Figure 23 - Sketch proposal for the admin page .....	30
Figure 24 - Diagram of the configuration classes .....	31
Figure 25 - Diagram of the resource classes .....	33
Figure 26 - Diagram of the model classes .....	34
Figure 27 - Diagram of the database tables and relationships .....	39
Figure 28 - Code of <code>router/index.js</code> .....	41
Figure 29 - Code of <code>auth/router/isAuthenticatedGuardAdmin.js</code> .....	42
Figure 30 - Code of <code>auth/login.vue</code> .....	42
Figure 31 - Code of <code>admin/router/index.js</code> .....	42
Figure 32 - Active state on selected component .....	43
Figure 33 - Code of <code>professor/components/Task.vue</code> .....	43
Figure 34 - Code of <code>professor/components/Task.vue</code> .....	43
Figure 35 - Code of <code>professor/components/Task.vue</code> .....	44
Figure 36 - Code of <code>professor/components/TaskList.vue</code> .....	44

Figure 37 - Code of professor/components/TaskList.vue.....	44
Figure 38 - Code of resources/task_pairing.py.....	45
Figure 39 - Code of resources/task_user_mark.py .....	46
Figure 40 - Homepage of the web application.....	47
Figure 41 - Home page of the web application.....	50
Figure 42 - LTI authentication sequence (Source: www.wikipedia.com).....	53

## Tables

Table 1 - User stories .....	11
Table 2 - Time estimation by user stories.....	14
Table 3 - Sprints duration .....	15
Table 4 - Resource classes .....	32
Table 5 - Account entity.....	35
Table 6 - User entity .....	35
Table 7 - Task entity .....	36
Table 8 - Aspect entity.....	36
Table 9 - Pairing entity.....	37
Table 10 - Evaluation entity .....	37
Table 11 - Attachment entity.....	38
Table 12 - Conclusion entity .....	38
Table 13 - Mark entity.....	38
Table 14 - Testing issues .....	49
Table 15 - Project costs.....	51



## **1. Introduction**

This project is about the process of research and analysis followed by the design and development of a web application that serves as an e-learning tool for peer evaluation [1] with the purpose to be used by any lecturer.

The process starts with research and analysis of the current workshop on the virtual campus of the University of Barcelona, followed by conceptualization and design proposal of the frontend and backend, and finally leads to the implementation of the website, meeting always the requirements established by the tutor of this project who is the product owner.

The project is structured in such a way that we first proceed to investigate the current market trends in this type of e-learning tool, define the product backlog of the project, and the frameworks used to develop it, as well as the technologies used behind together with the platform, to establish a well-structured information base.

With all this data, we proceed to make design proposals from the product backlog of the project, which include both frontend and backend. The frontend part begins by making branding for the project and is followed by making sketches of the website, from which we get a list of components that will be implemented. The backend part begins by defining the database of the system and is followed by defining the endpoints of the resources that the system consumes.

Once all these features are implemented, the project will be deployed on a platform for production.

### **1.1. Motivation**

The main reason that led me to choose this project is the large number of areas it covers. Starting with the visual part and logical part of the frontend design which users will interact with on the webpage, followed by the connection part for transferring data between client and server. On the server side, the design of the whole database architecture allows the storage of the data in an appropriate way. Finally, the deployment of the web application into production. Internally of the application, several algorithms must be implemented to pair students and calculate the final mark of each student.

A simple approach from the beginning of the implementation already requires a lot of knowledge in several fields such as frontend, backend, database, deployment, algorithms, and other technologies. Moreover, the analysis of the webpage of the current workshop on the virtual campus of the University of Barcelona, the research of the appropriate UX/UI design for an academic webpage, and the synthesis of the whole workflow that ease the evaluation sequence have been also an important part of this project.

The diversity of work involved in this project has made me wonder to understand how each element works, think about how I could connect each element, incorporate new technology and feature into the system, and deliver the best result for the project.

## 1.2. Context

The context of this application is based on the participative evaluation system of the Workshop activity on Moodle [2], where an evaluation is a systematic determination of a subject's merit, worth and significance, using criteria governed by a set of standards. The evaluation system that we are used to receiving at the educational institution is that the professor qualifies the submission of the student for an assignment. In the peer evaluation system, the point is to commit students to evaluate the submission of other students and provide a qualification of the activity, then this score is marked as a part of the final mark that the student will receive. While professor will also provide a mark for the review made by students, meaning that the student must provide a justified mark. The whole procedure of the peer evaluation system has been highly influenced by the peer review activity which is a method used in the field of software engineering to improve the quality of work and to learn from others.

## 1.3. State of art

There are a few contents related to the field of this project. The websites which are more interesting to be reviewed and inspired are:

- Workshop activity on the virtual campus of the University of Barcelona (based on Moodle): It is an integrated tool on Moodle that allows professors to manage the peer assessment task, and students to add submissions which are then distributed amongst their peers for assessment based on a grading scale specified by the professor. In figure 1, there is a snapshot of the activity.

Fase de configuració Canvia a la fase de configuració ○	Fase de tramesa Canvia a la fase de tramesa ○	Fase d'avaluació Canvia a la fase d'avaluació ○	Fase de qualificació de les avaluacions Canvia a la fase d'avaluació de les qualificacions ○	Tancament Fase actual ●
<ul style="list-style-type: none"> <li>✓ Establi la descripció del taller</li> <li>✓ Especifiqueu les instruccions per a la tramesa</li> <li>✗ Editeu el formulari d'avaluació</li> </ul>	<ul style="list-style-type: none"> <li>✗ Especifiqueu les instruccions per a l'avaluació</li> <li>✓ Assigneu les trameses esperades: 80 trameses: 63 per assignar: 0</li> <li>⌚ Hi ha com a mínim un autor que encara no ha tramès la seva feina.</li> <li>⌚ Inici de les trameses:dimecres, 24 de març 2021, 09:00 (Fa 446 dies)</li> <li>⌚ Data límit per trametre: dimecres, 24 de març 2021, 17:40 (Fa 446 dies)</li> <li>⌚ Les restriccions de temps no se us apliquen</li> </ul>	<ul style="list-style-type: none"> <li>⌚ Inici de les avaluacions des de dimecres, 24 de març 2021, 17:40 (Fa 446 dies)</li> <li>⌚ Data límit de l'avaluació: dimecres, 24 de març 2021, 23:59 (Fa 446 dies)</li> <li>⌚ Les restriccions de temps no se us apliquen</li> </ul>	<ul style="list-style-type: none"> <li>✗ Calculeu les qualificacions de la tramesa esperades: 80 calculades: 0</li> <li>✗ Calculeu les qualificacions de l'avaluació esperades: 80 calculades: 0</li> <li>✓ Proporcioneu una conclusió de l'activitat</li> </ul>	

Figure 1 - Workshop activity (Source: www.moodle.org)

- <https://peerassessment.com>: It is a platform that is aimed to automate the peer assessment process, making it easier to hold more frequent peer assessments that return real feedback to students so that students can improve their team skills. It helps the professor to prepare a peer assessment for your entire class in under 5 minutes. The platform takes care of the rest by collecting and distributing the feedback to students, plus preparing a grading sheet for the professor. In figure 2, there is a snapshot of the website.

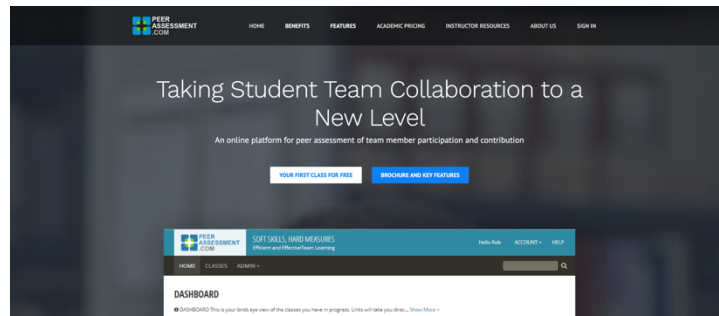


Figure 2 - Website of peerassessment.com (Source: peerassessment.com)

- <https://www.peergrade.io>: It is a platform for helping students give and receive written feedback on written work. First, teachers set up an account and tune the settings for how and when students can submit their work. Students can then log in and access their teacher's class, and they can interact with other students' submissions and add comments and edits. Teachers can then track their students' comments and their progress, and teachers can share feedback with students. In figure 3, there is a snapshot of the website.

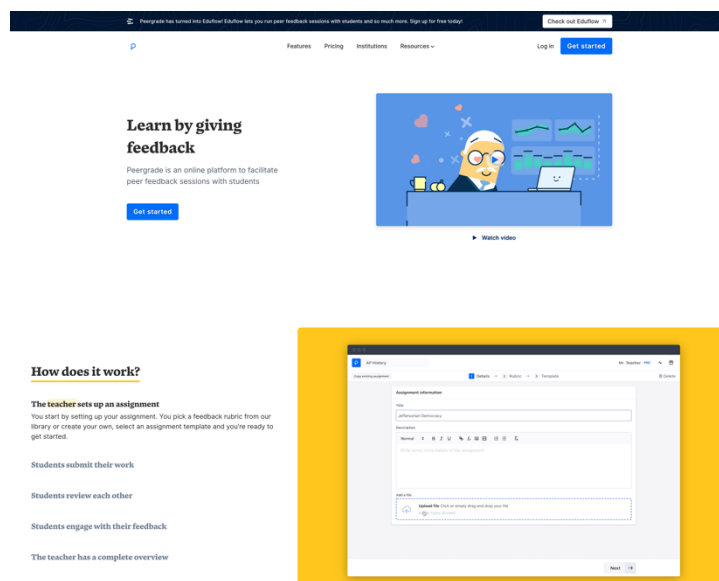


Figure 3 - Website of peergrade.io (Source: www.peergrade.io)



## **1.4. Objectives**

This section will describe the main objectives and the specific objectives that this project will address.

### **1.4.1. Main objectives**

The main objective of the project is to design a webpage for academic uses which can able the community of students to perform peer reviews and peer assessments of a task under a set of criteria determined by the professor, in the application, the professor is also responsible to determine the state of the workflow of the task. Both students and professors contribute to the scoring process of the individual task delivery and task review. It will consist of designing an architecture in which all the requirements of the end-users are satisfied and proposing a frontend design that will be intuitive for the user to interact with. It will also cover the ideation of different technologies that can be integrated into the software to improve the ease of using the webpage.

### **1.4.2. Specific objectives**

To carry out this project, the following specific objectives regarding the front-end level and back-end level must be met:

#### **1.4.2.1. Front-end level**

The front-end is the client-side of the application which includes everything that users experience directly, therefore the design of the webpage must satisfy the following points to provide a good user experience.

- UX/UI contents: The user interface must be clean, synthetic, and organized. While the user experience must be intuitive and user-friendly.
- SPA: The webpage must be a single-page application that dynamically rewrites the current webpage with the new data from the server. The aim is to make it feel like a native app by fast transitions of components.
- Multiple roles application: The webpage must route users by their role so that different users get to their corresponding space after logging in.
- Authentication: The webpage must be logged in via an authentication method connected with the server.
- File management: The webpage must allow to transfer and display the file managed during the assignment.

### **1.4.2.2. Back-end level**

The back-end is the server-side of the application which includes everything that the machine needs to process and provide an output of the system to the user, therefore the design of the architecture behind the system is required to meet the needs of the system.

- API: It will include the design of the endpoints that will be required for the system to manage the basic workflow of the evaluation system.
- SQL Database: The database must follow the design principles of the SQL.
- Authentication: The data manipulation in the database must be authenticated by token.
- File management: The server must allow to save and send the file managed during the assignment.
- Grading sheet: The grading sheet, which is the output of the system once the whole process of peer review and peer assessment is done, must be always generatable. By this, it means that for any uncertainty on the input of the evaluation made by the student, the system will always output the correct result under certain criteria.
- Pairing algorithm: The algorithm of pairing must be fast and efficient.

### **1.5. Document structure**

The content of this document is organized in the following chapters:

- Research: Analysis of the current UB virtual workshop, the definition of the requirements that each component that the system must fulfill, and breakdown of the product backlog.
- Planning: Explanation of the methodology applied in the project and the distribution of the different tasks in the development periods.
- Technologies: List of the technologies used in the system and justification of why they have been chosen.
- Design: Explanation of the frontend and backend designs.
- Implementation: Explanation of the technical aspects that implement the system internally.
- Deployment: Explanation of the platform used and how the project is deployed.
- Tests and results: Exposition of the tests carried out on the system and result of the project.
- Costs: Description of the economic costs involved in the commercial development of the system.
- Future works: List of possible and future lines of improvement.
- Conclusions: The summary of the project contextualized the initial problems of the project.
- References: Citations of the source of information used to develop the project.
- Annexes: Documentation and guidelines for the use and maintenance of the system.



## 2. Research

This section details the basic features that the application must have from analyzing the three websites mentioned at the state along with all the requirements that the project must satisfy, and the product backlog generated from user stories.

### 2.1. Analysis

First, we start with the workshop on the virtual campus of the University of Barcelona that serves as a baseline for this project. From the student's side, it begins with the authentication where the student is required to be authenticated to get into the system. Once the student gets into the system, by default, the student must get into the assignment to which the workshop belongs. After getting into the workshop, the student can see the timeline of the phases by which the workshop is set. Depending on the phases, the student will perform the following tasks: submit the task delivery, make the peer review and assessment, and review the peer feedback. In figure 4, we can see the workflow of the activity from student side.

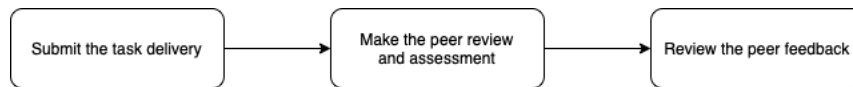


Figure 4 - Workflow of UB's workshop from the student side

From the professor's side, it also begins with the authentication where the professor is required to be authenticated to get into the system. Once the professor gets into the system, by default, the professor must create the Workshop activity which is in the Set up phase, here the professor must provide detailed grading criteria for the students to use. Since then, the students can submit work in the Submission phase where the professor must allocate submission to decide if you yourself want to choose which student assesses whose work (Manual allocation), or if you want Moodle to choose for you (Random allocation). The Assessment phase allows students to review each other's submissions and the professor can monitor their progress by looking at the grades underneath the phases screen. Finally, the Grading evaluation phase, Moodle calculates the final grades for submission and for assessment, here the professor can decide whether to maintain or delete some of the grades from the students. In figure 5, we can see the workflow of the activity from professor side.

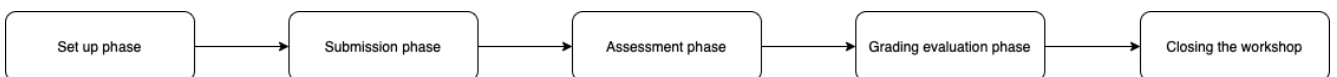


Figure 5 - Workflow of UB's workshop from professor side

Second, we are going to analyze the workflow of the <https://peerassessment.com> platform, at which the professor must perform the following steps:

- Create a class and load students
- Set up and schedule an assessment
- Run the assessment
- Distribute the student report and view the instructor report

In figure 6, we can see the workflow of the application.

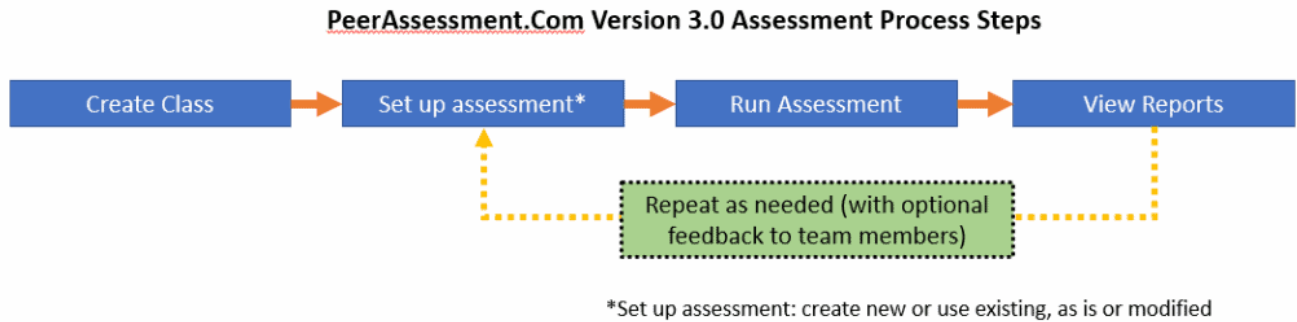


Figure 6 - Workflow of the peerassessment.com application (Source: peerassessment.com)

Third, we are going to analyze the workflow of the <https://www.peergrade.io> platform, at which both the students and the professor must perform the following steps:

- The professor sets up an assignment: The professor starts by setting up an assignment. Then, pick a feedback rubric from the library or create its own, select an assignment template and it is ready to get started.
- Students submit their work: Students submit their work to the assignment. They can submit anything such as text, files, videos, links, and even Google docs.
- Students review each other: Students give each other anonymous feedback through your rubric.
- Students engage with the feedback: Students receive feedback from their peers, and they react, discuss, and engage with their feedback.
- The professor has a complete overview: The professor gets a complete overview of everything that is happening in the assignment.

In figure 7, we can see the workflow of the application.

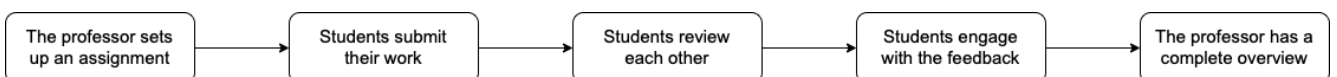


Figure 7 - Workflow of the www.peergrade.io application (Source: www.peergrade.io)

## **2.2. Requirements**

Considering the workflow of the peer evaluation system along with the management of the features that are composed at each step. The requirements of the project have been divided by the different roles of the users who are interacting with the system: administrator requirements, professor requirements, and student requirements.

### **2.2.1. Administrator requirements**

The administrator will manage the accounts that are registered in the database of the system. Therefore, it has the following requirement:

- Management of the accounts

### **2.2.2. Professor requirements**

The professor will manage the tasks that students will participate in for the peer evaluation. Therefore, it has the following requirements:

- Management of the tasks
- Management of the users that can participate in the task
- Management of the aspects that compose the task
- Management of the pairings that the user (giver of the evaluation) will be evaluating pairing\_user (receiver of the evaluation) in the task
- Management of the scoring for the students in the task
- Obtainment of the student's submitted task delivery
- Obtainment of the final grading sheet of the task

### **2.2.3. Student requirements**

The student will perform the peer evaluation of the task. Therefore, it has the following requirements:

- Submit the task delivery
- Obtainment of the peer's submitted task delivery
- Evaluate the peer's submitted task delivery
- Review the peer's evaluations
- Conclude the task

### 2.3. Product backlog

From the requirements listed in the previous section, the user stories to be implemented have been defined for the product backlog.

User stories are informal representations of the requirements of a software project. They are used in Agile development methodologies such as Scrum because of their ability to provide a quick response to changing requirements.

The below user stories will be expressed in the following sections: an identifier, a role, an action, and an outcome. First, we have the identifier that serves to unambiguously distinguish a user story. Then the set of role, action, and outcome constitutes the user story which is interpreted as follows: As a [role], I want to [action], so that I can [outcome]. The detailed version of the user stories with acceptance criteria is attached to the annex 13.1. of the project, where the acceptance criteria refer to a set of predefined requirements that must be met to mark a user story complete.

Before listing the user stories, it is necessary to define the roles that will be in the system. There will be three types: administrator, professor, and student. The user role will correspond to the client who requests the server data. The administrator has the aim to manage account data, the professor has the aim to manage all the phases of the evaluation of the tasks, and the student has the aim to perform the peer evaluation of the task.

In table 1, we can see the user stories of the project.

User Story	As a [role]	I want to [action]	So that I can [outcome]
US1	Professor, Student	Upload a form data of the sign-up account	Get access to the platform
US2	Administrator, Professor, Student	Upload a form data of the log in account	Get into the platform
US3	Administrator, Professor, Student	Click the log out button	Leave from the session
US4	Administrator	Upload a form data of the user	Create a user in the system
US5	Administrator	Upload a new form data of the user	Change the data of the user in the system
US6	Administrator	Click the remove user button	Remove the user from the system
US7	Administrator	Click the remove user's account button	Remove the account of the user from the system
US8	Professor	Upload a form data of the task	Create a task space for students

US9	Professor	Upload a new form data of the task	Change the data of the task in the system
US10	Professor	Click the remove task button	Remove the task from the list of tasks
US11	Professor	Upload a form of student	Create a student to the task
US12	Professor	Upload a CSV file of students	Create a list of students to the task
US13	Professor	Click the remove student button	Remove the student from the list of assigned students to the task
US14	Professor	Upload a form of the aspect	Set a criteria for evaluation
US15	Professor	Upload a new form data of the aspect	Change the data of the criteria for evaluation
US16	Professor	Click the remove aspect button	Remove the aspect from the list of assigned aspects to the task
US17	Professor	Upload a form of pairing	Create a pairing for evaluation
US18	Professor	Compute automatically the pairings	Create a list of pairings for evaluation
US19	Professor	Click the remove pairing button	Remove the pairing from the list of pairings
US20	Professor, Student	Click the attachment download button	Check the peer's task delivery
US21	Professor	Click see review qualification button	Check the result of the review qualification
US22	Professor	Upload a form of review qualification	Create a review qualification for the peer review
US23	Professor	Click see general qualification button	Check the result of the general qualification
US24	Professor	Upload a form of general qualification	Create a general qualification for the peer
US25	Professor	Click grading sheet download button	Check the grading sheet of the students
US26	Professor	Click increase / decrease state button	Manage the state of the task
US27	Student	Upload a file of attachment	Be evaluated by other peers
US28	Student	Upload a form of evaluation	Create an evaluation for the peer
US29	Student	Click see evaluation button	Check the result of the evaluations of the task
US30	Student	Upload a form of conclusion	Create a conclusion for the task

*Table 1 - User stories*





### 3. Planning

The development of this project has been carried out following the Scrum agile work methodology. This methodology involves applying a series of best practices focused on obtaining the best possible result from a project and based on the way highly productive teams work.

Scrum is frequently used in complex projects where the requirements are not fixed or are not very well defined and where it is necessary to obtain results quickly. For this same reason, in Scrum, partial deliveries of the product are made regularly, which also helps to resolve situations where the customer is not getting what he wants.

#### 3.1. Timing

Although in this case Scrum is not applied in full, since this project has not been done as a team, the advantages of this methodology have been equally reflected. Mainly, the achievement of results, the short testing periods, and a better compliance rate.

Another key point in the choice of this methodology has been its flexibility, which allows us to modify the thread of the facts at any time, thus achieving the resolution of conflicts on the fly and in an almost negligible way, in addition to avoiding unnecessary perfectionism.

In this case, the project has been divided into two large blocks: one dedicated to the development and the other to the memory. Considering them as separate projects, it has been possible to use different durations for the sprints of each one.

To distribute the user stories among all the sprints, we first estimated the approximate number of hours that their implementation would require. In table 2, we can see the time estimation for each user story:

User Story	Role	Action	Estimated time (hour)
US1	Professor, Student	Upload a form data of the sign-up account	10
US2	Administrator, Professor, Student	Upload a form data of the log in account	10
US3	Administrator, Professor, Student	Click the log out button	3
US4	Administrator	Upload a form data of the user	5
US5	Administrator	Upload a new form data of the user	5
US6	Administrator	Click the remove user button	3
US7	Administrator	Click the remove user's account button	3

US8	Professor	Upload a form data of the task	5
US9	Professor	Upload a new form data of the task	5
US10	Professor	Click the remove task button	3
US11	Professor	Upload a form of student	5
US12	Professor	Upload a CSV file of students	35
US13	Professor	Click the remove student button	3
US14	Professor	Upload a form of the aspect	5
US15	Professor	Upload a new form data of the aspect	5
US16	Professor	Click the remove aspect button	3
US17	Professor	Upload a form of pairing	5
US18	Professor	Compute automatically the pairings	40
US19	Professor	Click the remove pairing button	3
US20	Professor, Student	Click the attachment download button	40
US21	Professor	Click see review qualification button	5
US22	Professor	Upload a form of review qualification	5
US23	Professor	Click see general qualification button	5
US24	Professor	Upload a form of general qualification	5
US25	Professor	Click grading sheet download button	40
US26	Professor	Click increase / decrease state button	5
US27	Student	Upload a file of attachment	5
US28	Student	Upload a form of evaluation	5
US29	Student	Click see evaluation button	5
US30	Student	Upload a form of conclusion	5
Total estimated time (hours)			281

*Table 2 - Time estimation by user stories*

### 3.2. Sprints

After this, in table 3, we can see that the user stories are packaged into modules throughout the sprints as shown:

Sprint	Objective	Duration (weeks)
1	Initiation of the project	1

2	Research and analysis	1
3	Project definition and requirements	1
4	Front-end design: sketches and proposal	1
5	Back-end design: system and architecture	1
6	Implementation of App module	1
7	Implementation of Auth module	2
8	Implementation of Admin module	1
9	Implementation of Professor module	5
10	Implementation of Student module	4
11	Testing and bugs solutions	1
12	Documentation	2
Total duration (weeks)		21

Table 3 - Sprints duration

In figure 8, we can see the Gantt chart that illustrates the final planned distribution of the sprints:

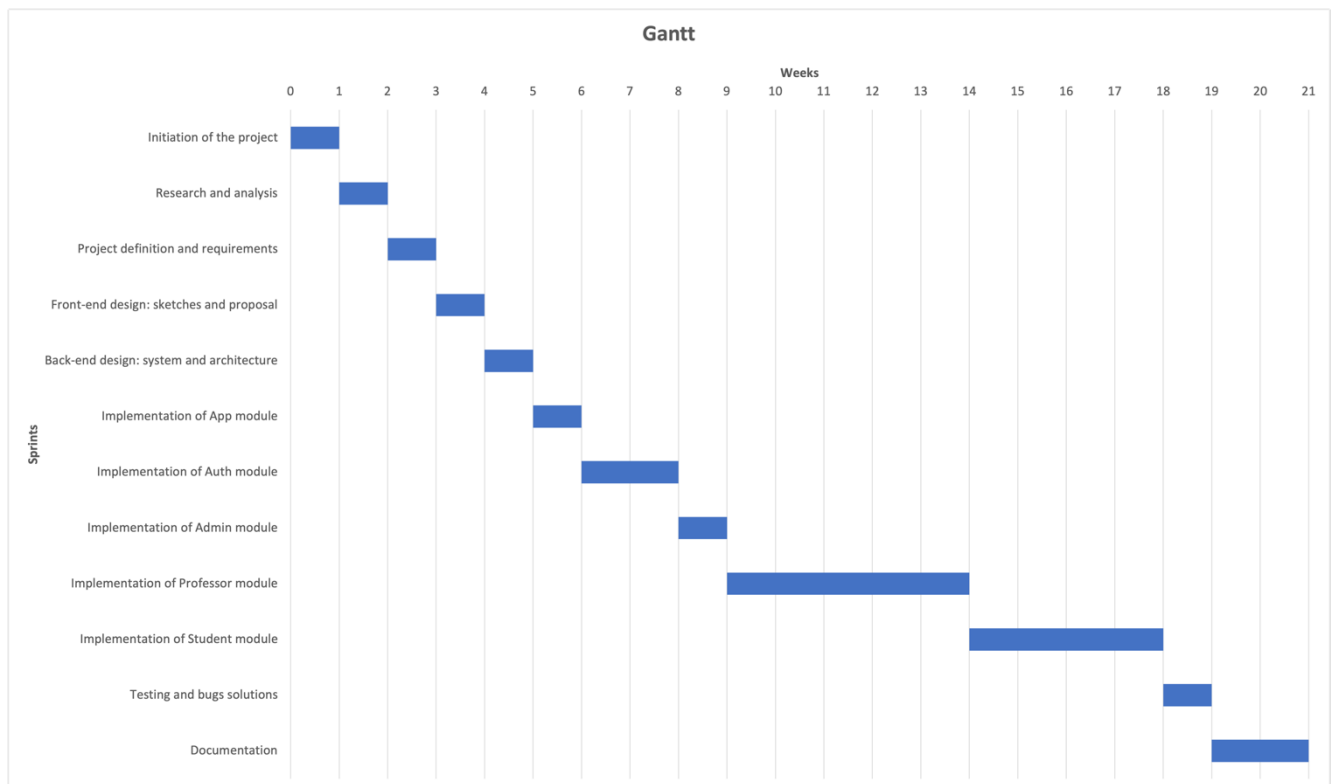


Figure 8 - Gantt diagram of the sprints



## **4. Technologies**

This section details all the languages, technologies, and platforms that are needed to elaborate on this project.

### **4.1. Programming languages and other languages**

#### **4.1.1. HTML (Hypertext Markup Language)**

HTML [3] is a language used to transfer information over the network using HTTP (Hyper Text Transfer Protocol), which defines the meaning and structure of web content. It allows web users to create and structure sections, paragraphs, and links using elements, tags, and attributes. Besides, other technologies are used to describe the appearance and behavior of the webpage.

The popular usage of HTML makes it a standard markup language that is recognized and interpreted by popular web browsers and its widespread acceptance as a language for web design.

#### **4.1.2. CSS (Cascading Style Sheets)**

CSS [4] is a design language that makes a webpage look more appealing than just plain or uninspiring pieces of text. Whereas HTML largely determines textual content, CSS determines the visual structure, layout, and aesthetics. It is defined as a style sheet language that is fundamental for web design, and it allows developers to adapt the presentation to different types of devices, such as small screens, and large screens. We have used CSS to add the appearance of the webpage.

#### **4.1.3. JavaScript**

JavaScript [5] is a lightweight object-oriented programming language that is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language, which enables dynamic interactivity on websites when it is applied to an HTML document.

It helps the developers to build modern web applications to interact directly without reloading the page every time, and it is commonly used to dynamically modify HTML and CSS to update a user interface. It is widely used for web applications on both client-side and server-side. We have used JavaScript to add the behavior of the webpage.

#### **4.1.4. Python**

Python[6] is designed to be a high-level and general-purpose interpreted language, in which instructions are executed directly without requiring that they have been previously compiled into machine language. Besides, it provides a strong level of abstraction as far as computer details are concerned. All these without restricting its fields of application.

Because of its simplified syntax and its emphasis on natural language, its popularity has increased in recent years. Thanks to this, a wide variety of libraries are available for almost any purpose. The latter, coupled with its portability, makes it the perfect language for the development of the system that is the subject of this project since portability to multiple platforms comes as a standard.

Like any other language, Python has its drawbacks. It presents a slow speed of execution, the usage of memory is relatively higher than in other languages, the storage of compiled code is also higher than in other languages and it is also prone to runtime errors.

The greatest part of Python is the support of libraries that can be integrated directly into any kind of project which makes software developers easier to implement new features.

## **4.2. Frameworks**

### **4.2.1. Vue**

Vue [7] is an open-source front-end framework written in JavaScript for designing user interfaces and web applications. Unlike other monolithic frameworks, this one is intended to be used incrementally.

Its core library is only focused on the visualization layer so integration with other libraries or existing projects are achieved with little hassle.

It is very lightweight, so not only is a high download and installation speed achieved, but also a strong positive impact can be achieved on SEO and user experience. Combined with its lightness, Vue.js stands out among other popular frameworks such as Angular or React for its performance.

Internally, each piece of the developed web application is structured into components, representing encapsulated interface elements. This division of the application into components is part of an approach called Component-Based Architecture that is also applied in Angular and React. The component structure makes it very easy to reuse components and to integrate HTML, CSS, and JavaScript code in a single file, making it easy to read and maintain.

Its developers also maintain very concise documentation, so its learning curve is flattened. The community around this framework is equally large, so there are plenty of libraries, tools, and solutions to design anything.

Being a relatively new framework, the number of plugins and components that Vue.js has does not overshadow those that both Angular and React support, for example. Its rapid evolution also complicates tracking for developers, who are often forced to relearn the framework to keep up.

However, these reasons have not been a drawback in choosing it for front-end development. It's easy portability between different devices and internal features such as conditional rendering of elements or loops, make it an ideal framework for the development of dynamic web pages and adaptive and spectacular web applications.

The reason for using this framework and not another for the interface has been purely personal. Having previously used it in other web applications and have obtained very good results, I have preferred it over others that might be more capable.

#### 4.2.2. Vuex

Vuex [8] is a state management pattern and library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable way.

In figure 9, we can see the state management pattern is a self-contained app with the following parts:

- State: the source of truth that drives our app.
- View: a declarative mapping of the state.
- Actions: the possible ways the state could change in reaction to user inputs from the view.

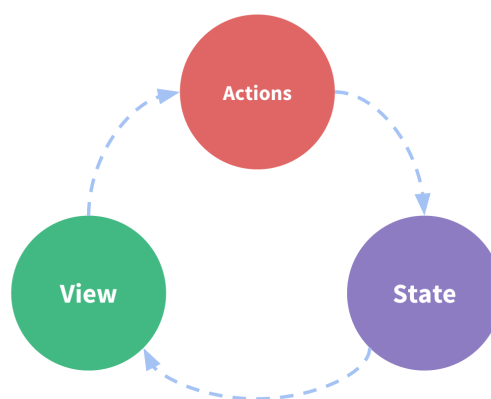


Figure 9 - Data-flow of Vuex (Source: vuex.vuejs.org)



In figure 10, we can see that Vuex is composed of state, mutations, actions, modules, and getters.

- State: the data that our components depend on and render.
- Mutations: synchronous methods to update the state in our Vuex store. These methods are used to commit and track state changes, it's a best practice to have actions to call mutations, which update our state directly.
- Actions: asynchronous information that comes from our API. These methods are used to fetch data from API and can process this data in two ways, by storing it in the state with a mutation method or by returning a response to the component that called the action.
- Getters: a method to get data stored from the Vuex store and often used to display the data on the webpage.

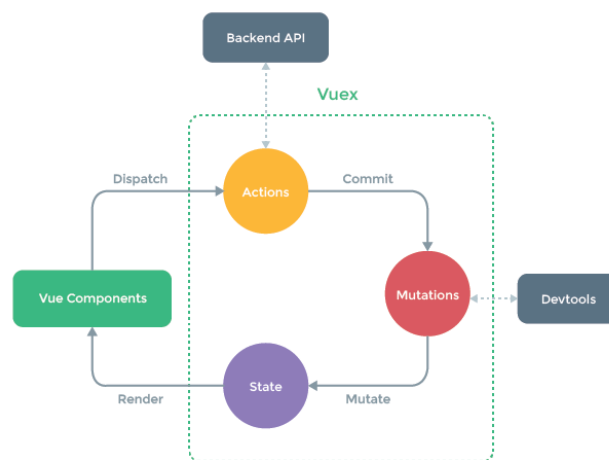


Figure 10 - State management of Vuex (Source: [vuex.vuejs.org](https://vuex.vuejs.org))

### 4.2.3. Flask

The server, an essential structure in this work, has been implemented with Flask [9], a framework focused on web application development.

Flask is a micro-framework written in Python. It's few dependencies with external libraries characterize it. On the one hand, is perfect because it makes it a lightweight and easy to update web framework. On the other hand, having no built-in libraries results in more work for the programmer, either by developing the features manually or by having to build the dependency list manually.

Also, although not by default, Flask has support for API development, which is a key element for the system being developed. Both the client and the interface will communicate with the server through a web API, so this support is one of the most important reasons for choosing Flask as a backend.

Other features include easy deployment, support for RESTful requests, extensive documentation, and high compatibility with many technologies.

In the long term, applications written in Flask can be easily developed, maintained, and scaled despite being based on a minimalist framework.

### **4.3. Platform**

#### **4.3.1. Heroku**

Containers are technologies that isolate applications along with their execution environment and dependencies, abstracting the application from the environment in which it is executed. This abstraction makes the deployment of applications much easier and factors such as the characteristics of the environment do not affect the application.

Heroku [10] is a cloud application container that uses Amazon Web Services cloud services. It saves the need to have your own servers to have the container and because it is developer-centric, the developer must spend time managing the container.

It's also easy to scale, use, and get started, making it ideal for small, big-picture projects. Heroku also includes a set of tools that allow you to integrate databases.

#### **4.3.2. GitHub**

It is a collaborative development platform dedicated to hosting projects using Git version control. It provides several useful features such as bug tracking, workflow management, and continuous integration.

During the development of a project, countless changes are made to the project's source code while deploying or delivering it to customers. Git version control keeps track of all modifications that have been made to the source code. Each record is generated by a commit. Each commit has a unique identifier that identifies a change or a set of changes. By keeping a record of all commits, developers are provided with the ability to revisit the source code before a given commit or even restore it.

GitHub [11], based on Git, incorporates some features for collaborative work. One of these features is branches. Essentially, they are forks of the main source code, with modifications concerning it. With this feature, multiple people can develop at the same time since the branches are independent of each other.

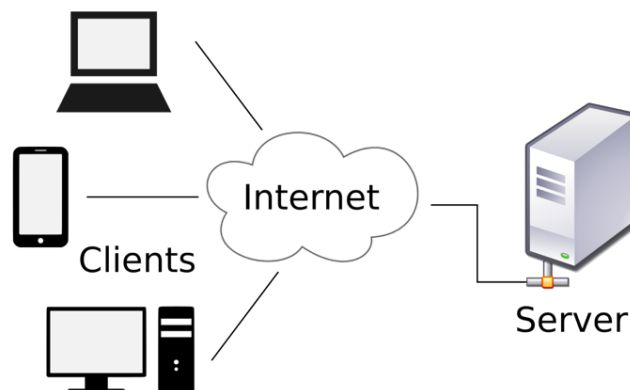


## 5. Design

This section details internal system design issues in terms of architecture, resources, database models, and security metrics.

### 5.1. Distributed application architecture

The model applied in this project is the so-called client-server model, as shown in the figure 11. This type of architecture partitions the tasks or workloads between the providers of a resource or service, called servers, and those who request the server, called clients.



*Figure 11 - Diagram of a client-server architecture (Source: [www.wikipedia.com](http://www.wikipedia.com))*

Communication between client and server is done using a request-response message pattern. The client sends a request to the server, which will return a response. For this to be possible, there must be a communication protocol known and used by both parties, which defines the language, syntax, and order of communications.

All communication protocols operate at the application layer. A server can implement an application programming interface or API. An API is a layer of abstraction in the access to a service. By abstracting access and restricting communications to a certain format, the exchange of information between different platforms is simplified.

In the context of the system, students, professors, and administrators, all take the role of clients against the implemented server. The server always responds to the requests of clients by returning data or processing it as needed via an API.

## 5.2. Branding

For any business, branding is an important part to create conscious and unconscious connections with the public to influence their usage decisions. In other words, branding focuses on making a brand known and desired and exerting a positive image in the minds and hearts of consumers. Therefore, creating a branding name, a branding domain and a branding logotype are essential to attract the user's attention.

The branding name needs to be simple and easy to remember. For this reason, we take the simplest naming approach as 'Peer Evaluation System', which means that the platform is a system where peers can evaluate between them. The word 'Peer' in the naming is taken from the idea of 'Peer Assessment', which is the evaluation of work by one or more people with similar competencies as the producers of the work.

The branding domain needs to be coherent with the branding name. Therefore, the approach to obtaining the branding domain is to synthesize the naming into shortened words that can retrieve the meaning of the branding name. The selection of the final branding domain leads to be: [peerevaluationsystem.herokuapp.com](https://peerevaluationsystem.herokuapp.com).

The branding logomark is the symbolic icon that represents a brand. The design of this logo is a process that has been taken for two main ideas. The center circle has the meaning of being a main peer. The surrounding circles represent other peers of the task, three of them are connected to the task meaning that they are chosen to evaluate between them, and three of them are not connected to the task meaning that they are not chosen to be evaluated. The selection of the final branding logomark is the following:



*Figure 12 - Logotype of the application*

### 5.3. Client

The client-side of the system is designed to have two types of roles. The first role is the student, who has the activity of evaluating other students. The second role is the professor, who controls all the cycles of an evaluation.

#### 5.3.1. Home page

As we can see in figure 13, the home page is designed to be the main entry point to a website, appearing when a user starts a session. It is divided into several sections. At the top of it, there is a navigation bar where users can see the branding of the application composed of a logo and a naming, the main routes of the application, and a log-in button. Then, it is followed by a set of different sections composed of vision and mission, main features, promotion area, and user opinions. At the bottom of it, there is a footer where users can see the description of the application, the follow us area, the main routes, and the contact information.

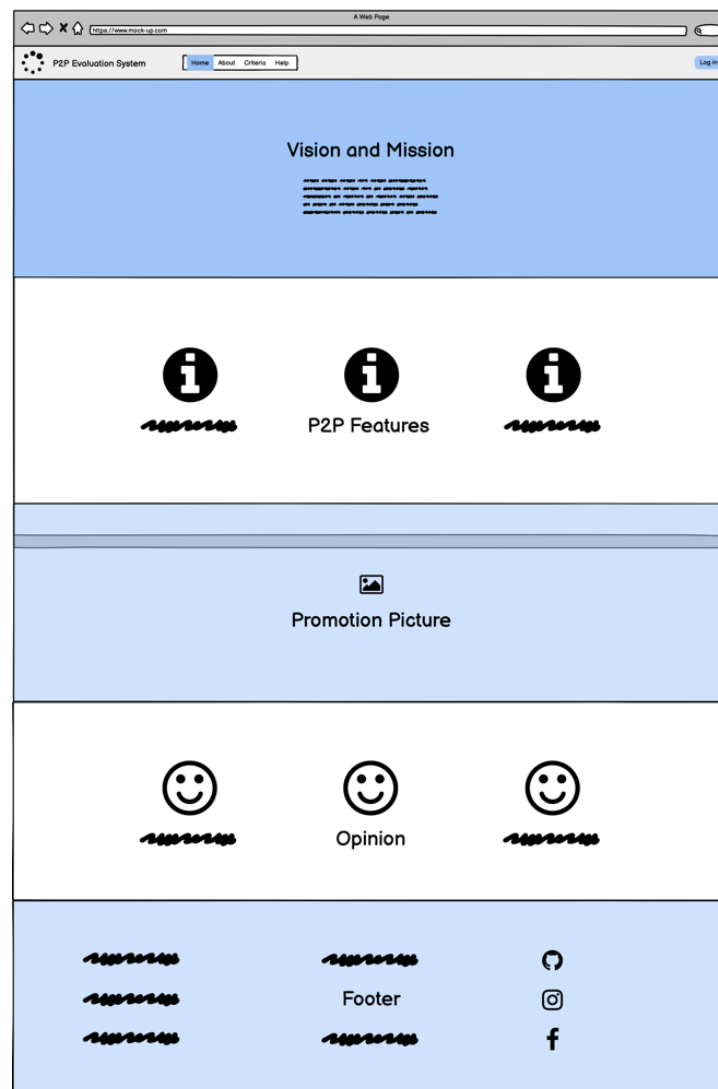


Figure 13 - Sketch proposal for the home page

### 5.3.2. Auth page

As we can see in figure 14, the auth page is designed to be the interface where users can get access to the system. This page is required to authenticate the user for security reasons and the need to routing the pages. The login form will require a valid email and password that are registered previously in the database via the Signup page. To sign up for an account, the introduced email requires to be previously added by the professor to the task. In this way, only those accounts' emails registered by professors are allowed to be signed up.



Figure 14 - Sketch proposal for the auth page

### 5.3.3. Student page

The student page is designed to be the interface once the student gets access to the system. The initial draft has divided the whole evaluation process into 4 phases for the student: attach, evaluate, review, and conclude. But throughout the process of creating a proper user-centered design interface, the involved phases are renamed into attach file, evaluate peers, review evaluations, and conclude task.

- As we can see in figure 15, the 'attach file' subpage is designed for the student to add a submission of a file that will be evaluated by the professor and other students.
- As we can see in figure 16, the 'evaluate peers' subpage is designed for the student to add evaluations of the attached file by other students.
- As we can see in figure 17, the 'review evaluation' subpage is designed for the student to review evaluations made by other students.
- As we can see in figure 18, the 'conclude task' subpage is designed for the student to write a conclusion of the overall evaluation of the task.

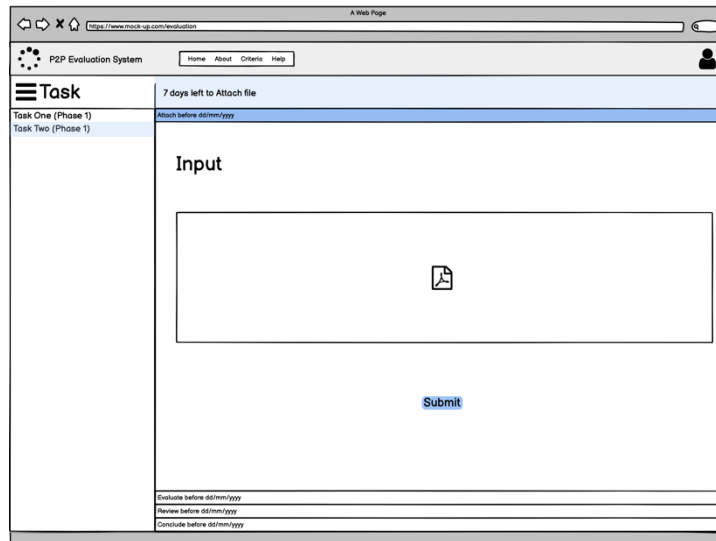


Figure 15 - Sketch proposal for the attach file subpage

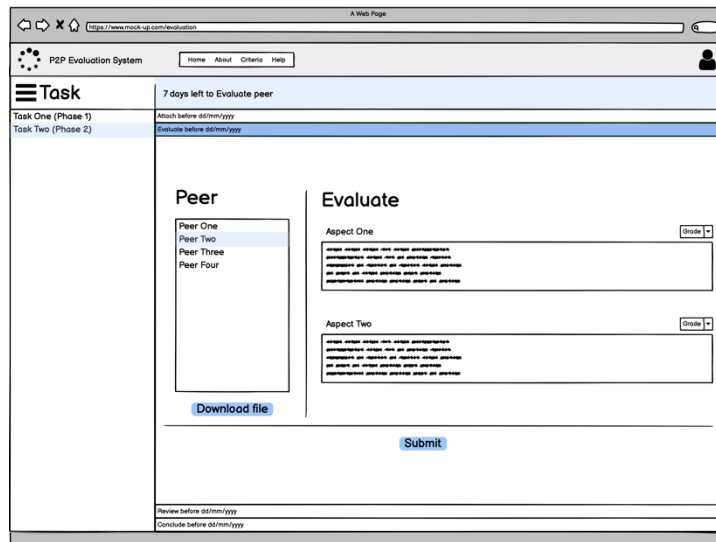


Figure 16 - Sketch proposal for the evaluate peers subpage

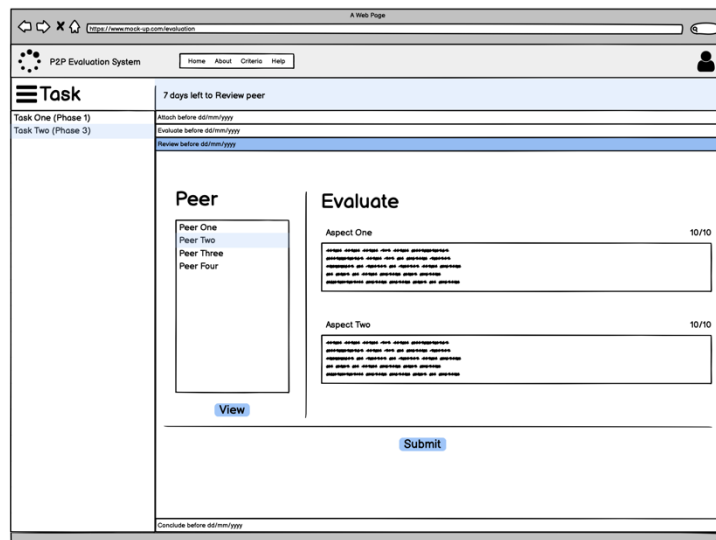


Figure 17 - Sketch proposal for the review evaluations subpage



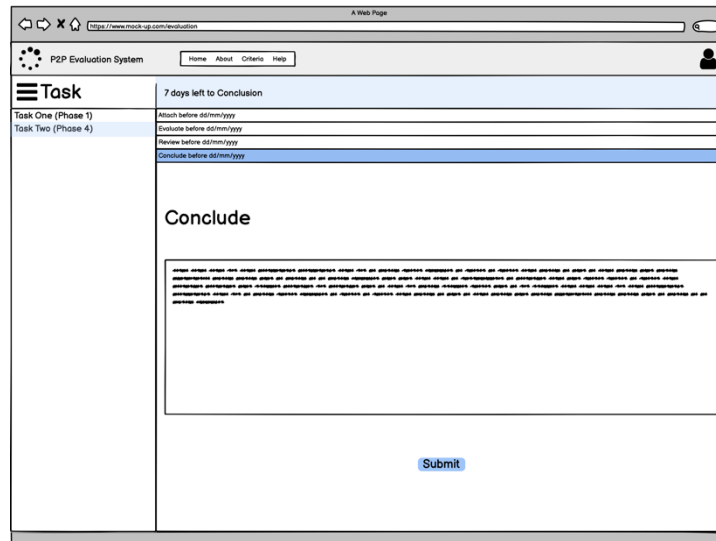


Figure 18 - Sketch proposal for the conclude task page

#### 5.3.4. Professor page

The professor page is designed to be the interface once the professor gets access to the system. The initial draft has divided the whole evaluation process into 4 phases for the professor: set up, pair, qualify, and retrospect. But throughout the process of creating a proper user-centered design interface, the involved phases are divided more concretely into set up task, add peers, add aspects, pair peers, qualify peers, and retrospect task.

- As we can see in figure 19, the original 'set up task' subpage is design to contain the following phases:
  - o The 'set up task' subpage is designed for the professor to add all primitive data that composes the task.
  - o The 'add peers' subpage is designed for the professor to add users that compose the task. It could be manually introduced or via a csv file.
  - o The 'add aspects' subpage is designed for the professor to add aspects that compose the task.
- As we can see in figure 20, the 'pair peers' subpage is designed for the professor to add a pairing of a student that will review and provide a commentary and score to the pairing student.
- As we can see in figure 21, the 'qualify peers' subpage is designed for the professor to add qualifications of the attached file and reviews that are made by the student.
- As we can see in figure 22, the 'retrospect task' subpage is designed for the professor to review all the conclusions that are left by the students after the evaluation session.

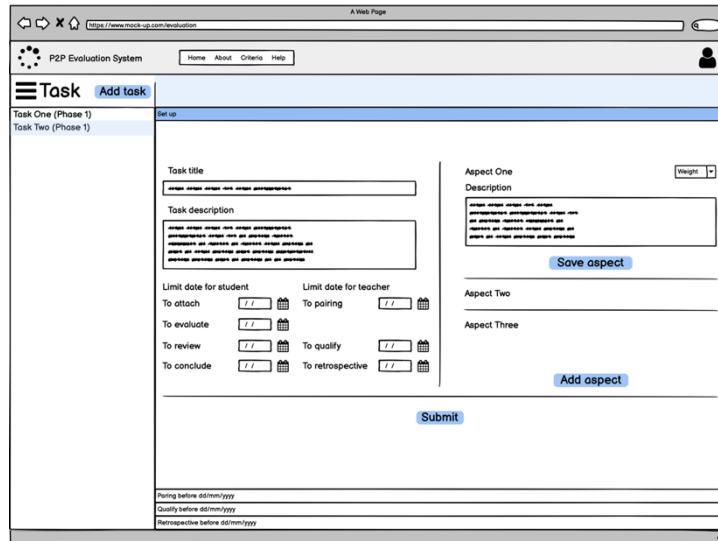


Figure 19 - Sketch proposal for the original set up task subpage

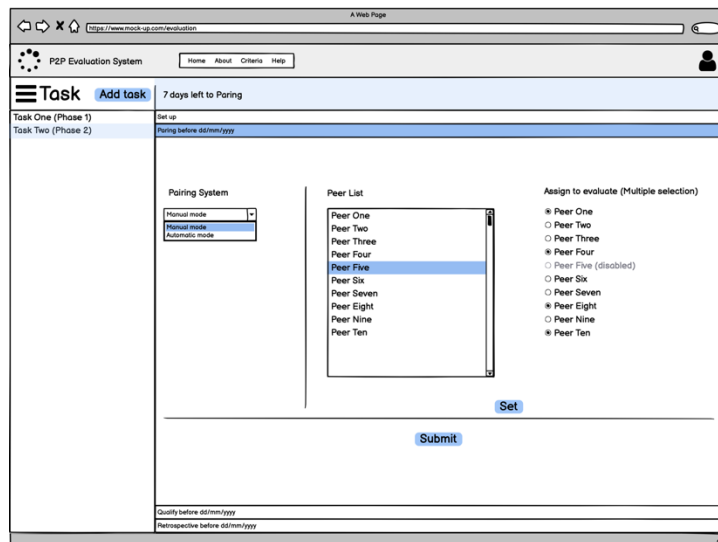


Figure 20 - Sketch proposal for the pair peers subpage

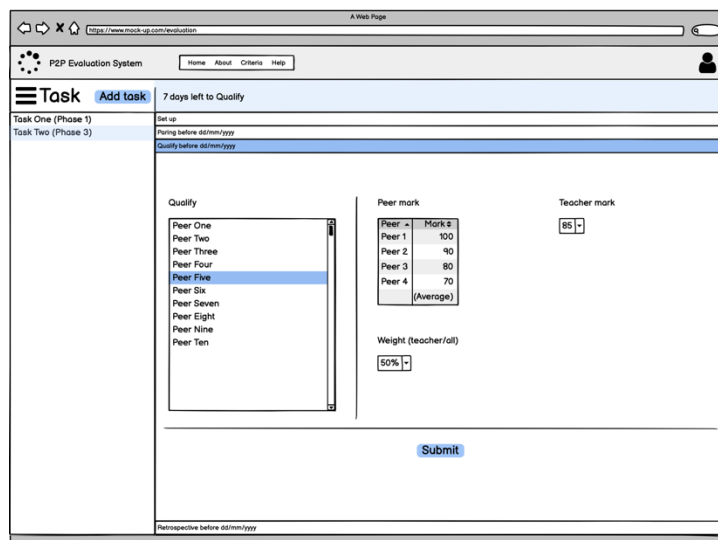


Figure 21 - Sketch proposal for the qualify peers subpage

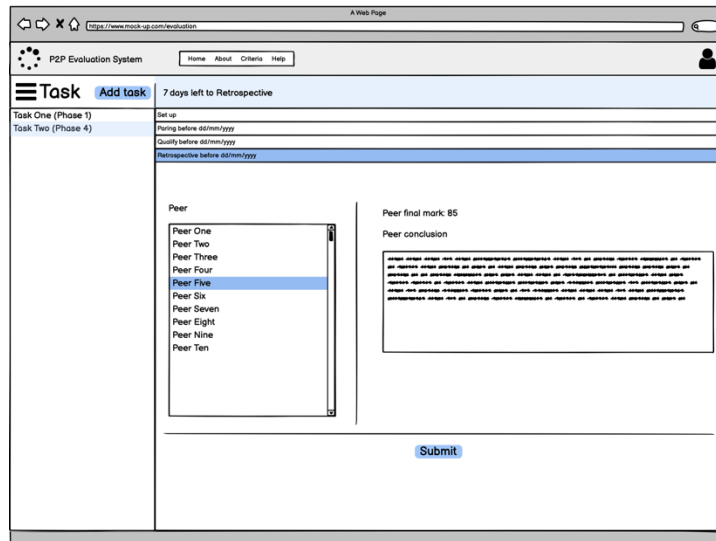


Figure 22 - Sketch proposal for the retrospect task subpage

### 5.3.5. Admin page

As we can see in figure 23, the admin page is designed to be the interface once the admin gets access to the system. The initial draft was proposed to manage users' accounts of the system, and throughout the process of creating a proper user-centered design interface, the initial proposal satisfies the final requirements.

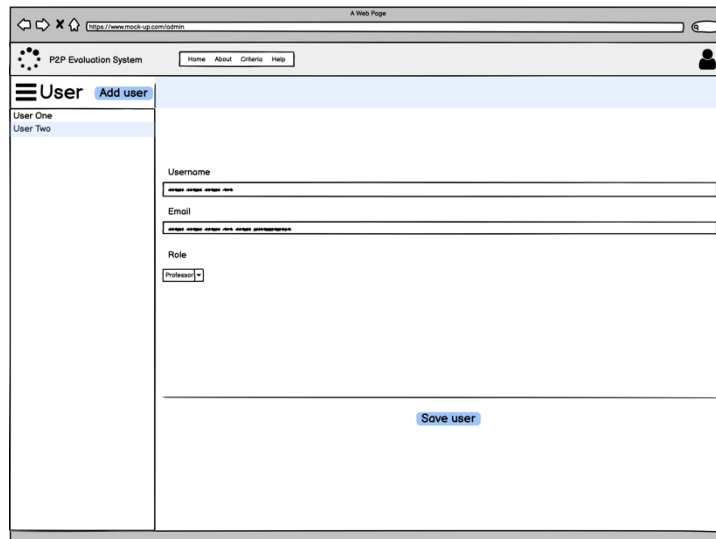


Figure 23 - Sketch proposal for the admin page

## 5.4. Server

The server-side of the system is designed to contain a configuration class along with two modules: resources and models. The resources are the package that contains the implementation of the RESTful API, and the models are the package that contains the implementation of the database.

### 5.4.1. Configuration class

As we can see in figure 24, the configuration class, namely Config class (config.py), is essential for loading the individual configuration of each environment. There are two classes that extend from this class, which are used to change the database used depending on the deployment environment, namely DevelopmentConfig and ProductionConfig. In detail, the variables are the mode of DEBUG, the path of SQLALCHEMY\_DATABASE\_URI, the mode of SQLALCHEMY\_TRACK\_MODIFICATIONS and the SECRET\_KEY of the application.

The rest of the files are in the root directory and have no special dependencies.

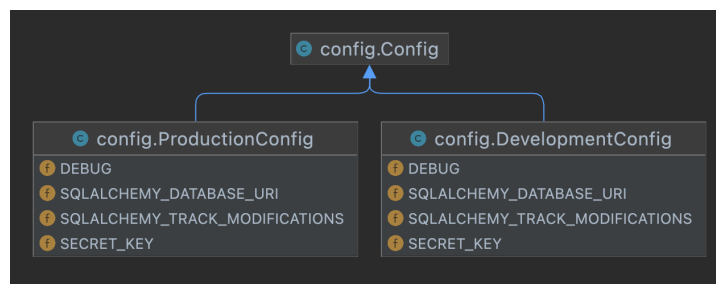


Figure 24 - Diagram of the configuration classes

### 5.4.2. Resources

The resource module contains all the RESTful classes that are related to the operations with the object from the database. All the classes are extended from Resources of Flask which allows us to implement a modular API and has its own function to realize, which is described in the following table 4:

Resource class	Description
Account	Management of an account
Signup	Register an account
Login	Authenticate a user by the registered account
TaskAspect	Management of an aspect of the task
TaskMarkList	Obtainment of a final grading sheet of the task

TaskPairingList	Compute a list of pairings of the task automatically by a seed number
TaskPairingUserPairingList	Obtainment a list of pairings assigned to the pairing_user of the task
TaskState	Management of the state of the task
TaskUser	Management of a user of the task
TaskUserList	Management of a list of users of the task
TaskUserAttachment	Management of an attachment of the user of the task
TaskUserConclusion	Management of a conclusion of the user of the task
TaskUserMark	Management of a mark of the user of the task
TaskUserMarkReview	Management of a review_mark of the user of the task
TaskUserPairingList	Obtainment of a list of pairings assigned to the user of the task
TaskUserPairingUserPairing	Management of a pairing of a user to a pairing_user of the task
Pairing	Management of a pairing of the task
PairingAspectEvaluation	Management of an evaluation of the aspect of the pairing
PairingEvaluationList	Obtainment of a list of evaluations of the pairing
User	Management of a user
UserList	Obtainment of a list of users
UserTask	Management of a task of the user
UserTaskList	Obtainment of a list of tasks of the user

*Table 4 - Resource classes*

### 5.4.3. RESTful API

The API of RESTful is an application programming interface that fits within the boundaries of the REST architecture and enables interaction with RESTful web services. It is composed of a set of architectural principles by which web services can be designed that are primarily focused on the resources of a system.

Each endpoint of the application obeys the REST constraints which are:

- Client-server architecture
- Statelessness
- Cacheability
- Layered system
- Code on demand
- Uniform interface

and we can implement all methods that contain each concrete Resource class in the figure 25:

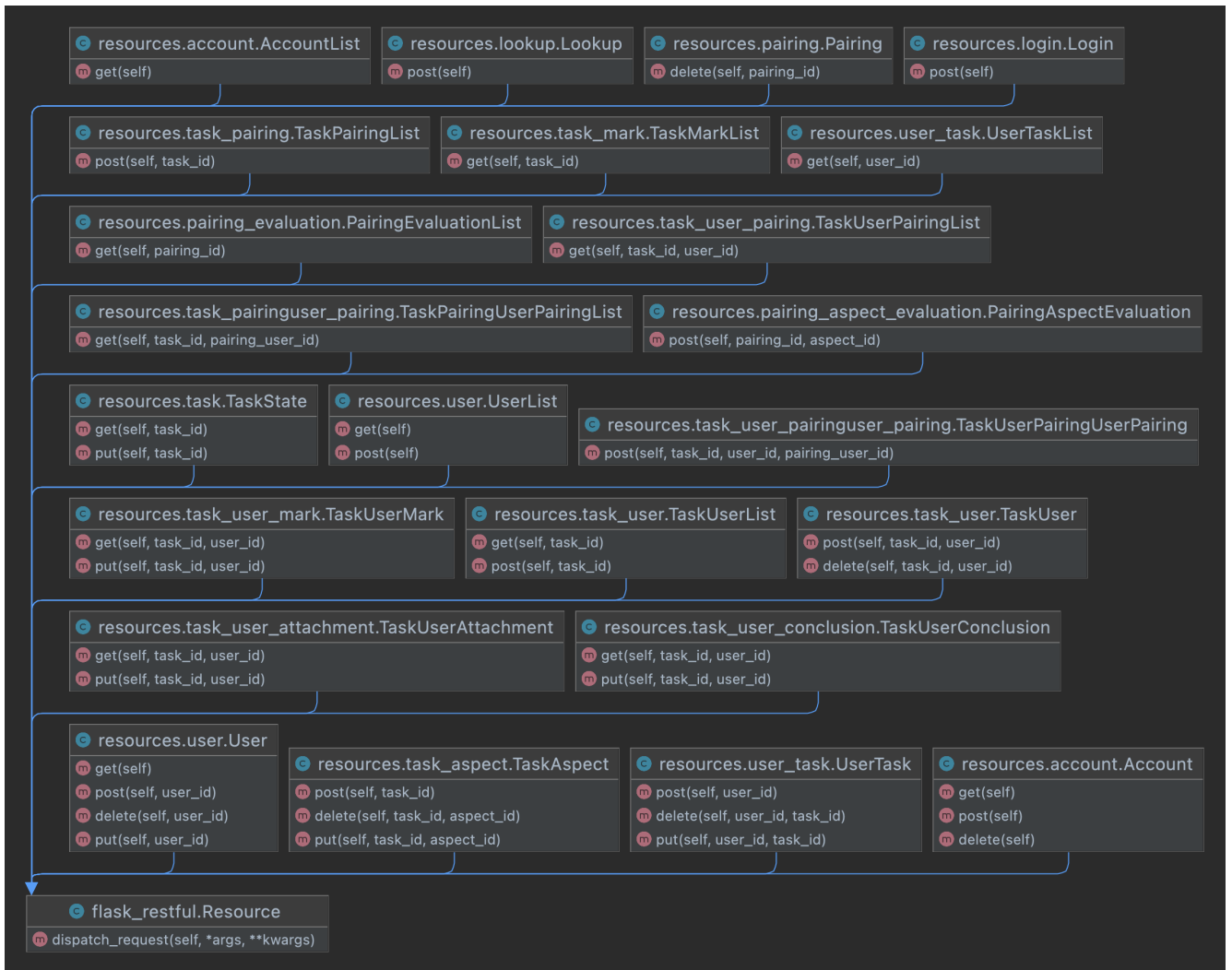


Figure 25 - Diagram of the resource classes

The concrete resources are extended from the `flask_restful.Resource` class and expose methods for each supported HTTP method. If a resource is invoked with an unsupported HTTP method, the API will return a response with the status `405 Method Not Allowed`. Otherwise, the appropriate method is called and passed all arguments from the URL rule used when adding the resource to an API instance. The detailed version of the RESTful API is attached to the annex 13.2. of the project.

#### 5.4.4. Models

The model module contains all the classes for creating a virtual object database where its methods can be added as the developer requires. In effect, it uses object-relational mapping, known as ORM, which is a programming technique for converting data between type systems using object-oriented

programming languages. For this project, we use SQLAlchemy, a Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

The concrete Model classes are detailed in the figure 26:

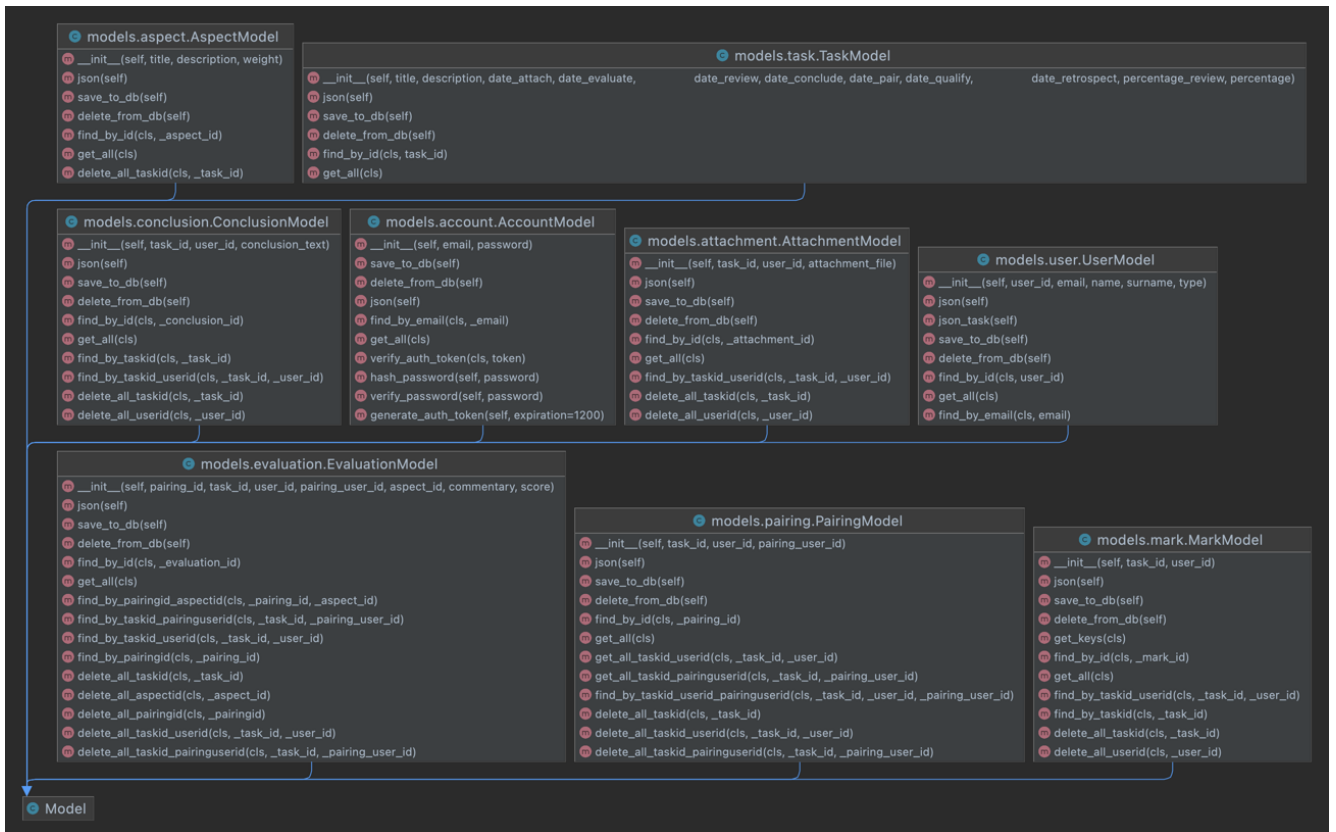


Figure 26 - Diagram of the model classes

The methods of each model are called from Resources and are used to manipulate the data in the database. Mainly, there are a few types of methods:

- json(): a representation of an entity object.
- save\_to\_db(): save an entity object to the database.
- delete\_from\_db(): delete an entity object from the database.
- find\_by\_id(<id>): find an entity object filtered by key element id.
- find\_by\_{<id>}{<id>}: find a set of entity objects filtered by a set of key element id.
- get\_all(): get all entity objects.
- get\_all\_{<id>}{<id>}: get a set of entity objects filtered by a set of key element id.
- delete\_all\_{<id>}{<id>}: delete a set of entity objects filtered by a set of key element id.

### 5.4.5. Database entities

We need to concrete the entities of the database before defining their relationships. Therefore, we start with the entities of the database by defining the attributes that belong to each entity and their type. A short description is included to justify the need for the attribute.

#### - Account

The *Account* has the purpose to authenticate the users, where all users are required to register on our system and logged in to the system to concrete the realization of the activities. For this reason, we need this specific database to save the data of authentication. It contains the following columns detailed in table 5:

Account		
Attribute	Type	Description
email	String	Mail address of the user
password	String	Password for the account. It is encrypted on the server-side

Table 5 - Account entity

#### - User

The *User* has the purpose to save all the users that are allowed to be registered and be the entity that will interact with other entities to concrete the realization of the activities. It serves as a separate entity from authentication data, and it has the user's personal information. It contains the following columns detailed in table 6:

User		
Attribute	Type	Description
user_id	Integer	Unique identifier of the user
name	String	Name of the user
surname	String	Surname of the user
email	String	Mail address of the user registered on the account database
type	Integer	Type of the user [0: student, 1: professor, 2:admin]

Table 6 - User entity

#### - Task

The *Task* has the purpose to save all the primitive data that compose a task. It contains the following columns detailed in table 7:



Task		
Attribute	Type	Description
task_id	Integer	Unique identifier of the task
title	String	Title of the task
description	String	Description of the task
date_attach	String	Deadline for students to attach file
date_evaluate	String	Deadline for students to evaluate others
date_review	String	Deadline for students to review evaluations
date_conclude	String	Deadline for students to conclude task
date_pair	String	Deadline for professor to pair students
date_qualify	String	Deadline for professor to qualify students
date_retrospect	String	Deadline for professor to retrospect task
state_value	Integer	The state of the task
percentage_review	Float	Percentage of the professor mark for review
percentage	Float	Percentage of the professor mark for task

Table 7 - Task entity

- **Aspect**

The *Aspect* has the purpose to save all the primitive data that compose an aspect of a task. It contains the following columns detailed in table 8:

Aspect		
Attribute	Type	Description
aspect_id	Integer	Unique identifier of the aspect
title	String	Title of the aspect
description	String	Description of the aspect
weight	Integer	Weight of the aspect for the evaluation

Table 8 - Aspect entity

- **Pairing**

The *Pairing* has the purpose to save the pairing that a user is paired to a pairing\_user for the evaluation of a task. Concretely, the pairing\_user will be evaluated by the user. It contains the following columns detailed in table 9:

Pairing		
Attribute	Type	Description
pairing_id	Integer	Unique identifier of the pairing
task_id	Integer	Foreign key that indicates where the pairing is belonged to
user_id	Integer	Foreign key that indicates the user who will give the evaluation
pairing_user_id	Integer	Unique identifier of the user who will be evaluated

Table 9 - Pairing entity

### - Evaluation

The *Evaluation* has the purpose to save all the data that compose an evaluation made by a user to a pairing\_user for each task. It contains the following columns detailed in table 10:

Evaluation		
Attribute	Type	Description
evaluation_id	Integer	Unique identifier for the evaluation
pairing_id	Integer	Foreign key that indicates the pairing of which is evaluated
task_id	Integer	Foreign key of pairing that indicates where the pairing is belonged to. (Operational reason)
user_id	Integer	Foreign key of pairing that indicates the user who will give the evaluation (Operational reason)
pairing_user_id	Integer	Foreign key of pairing that indicates the user who will be evaluated. (Operational reason)
aspect_id	Integer	Foreign key of aspect that indicates the aspect of which the user is evaluating
commentary	String	Commentary for each aspect evaluated
score	Integer	Score for each aspect evaluated

Table 10 - Evaluation entity

### - Attachment

The *Attachment* has the purpose to save the data of an attachment that a user will need to send for each task to get evaluated. It contains the following columns detailed in table 11:

Attachment		
Attribute	Type	Description
attachment_id	Integer	Unique identifier for the attachment

task_id	Integer	Foreign key that indicates where the pairing is belonged to
user_id	Integer	Foreign key that indicates the user who will give the evaluation
attachment_file	String	Name of the attachment file

*Table 11 - Attachment entity*

**- Conclusion**

The *Conclusion* has the purpose to save the data of a conclusion that a user will make by the end of each task. Through this, the professor can get feedback from students. It contains the following columns detailed in table 12:

Conclusion		
Attribute	Type	Description
conclusion_id	Integer	Unique identifier for the conclusion
task_id	Integer	Foreign key that indicates where the conclusion is belonged to
user_id	Integer	Foreign key that indicates the user who will give the conclusion
conclusion_text	String	Conclusion for the task

*Table 12 - Conclusion entity*

**- Mark**

The *Mark* has the purpose to save all the data that compose a mark of a user for each task. It contains the following columns detailed in table 13:

Mark		
Attribute	Type	Description
mark_id	Integer	Unique identifier for the mark
task_id	Integer	Foreign key that indicates where the mark is belonged to
user_id	Integer	Foreign key that indicates the user who will give the mark
review_mark	Float	Mark of the qualification from professor for review
student_mark	Float	Average mark of the evaluation from students
professor_mark	Float	Mark of the qualification from professor for task
final_mark	Float	Computed mark of review_mark, student_mark and professor_mark

*Table 13 - Mark entity*

### 5.4.6. Database relationships

All the database models are created from SQLAlchemy which generates a SQL table for a single model with all attributes that belong to the model. The general view of the created tables and their relationship is detailed in figure 27:

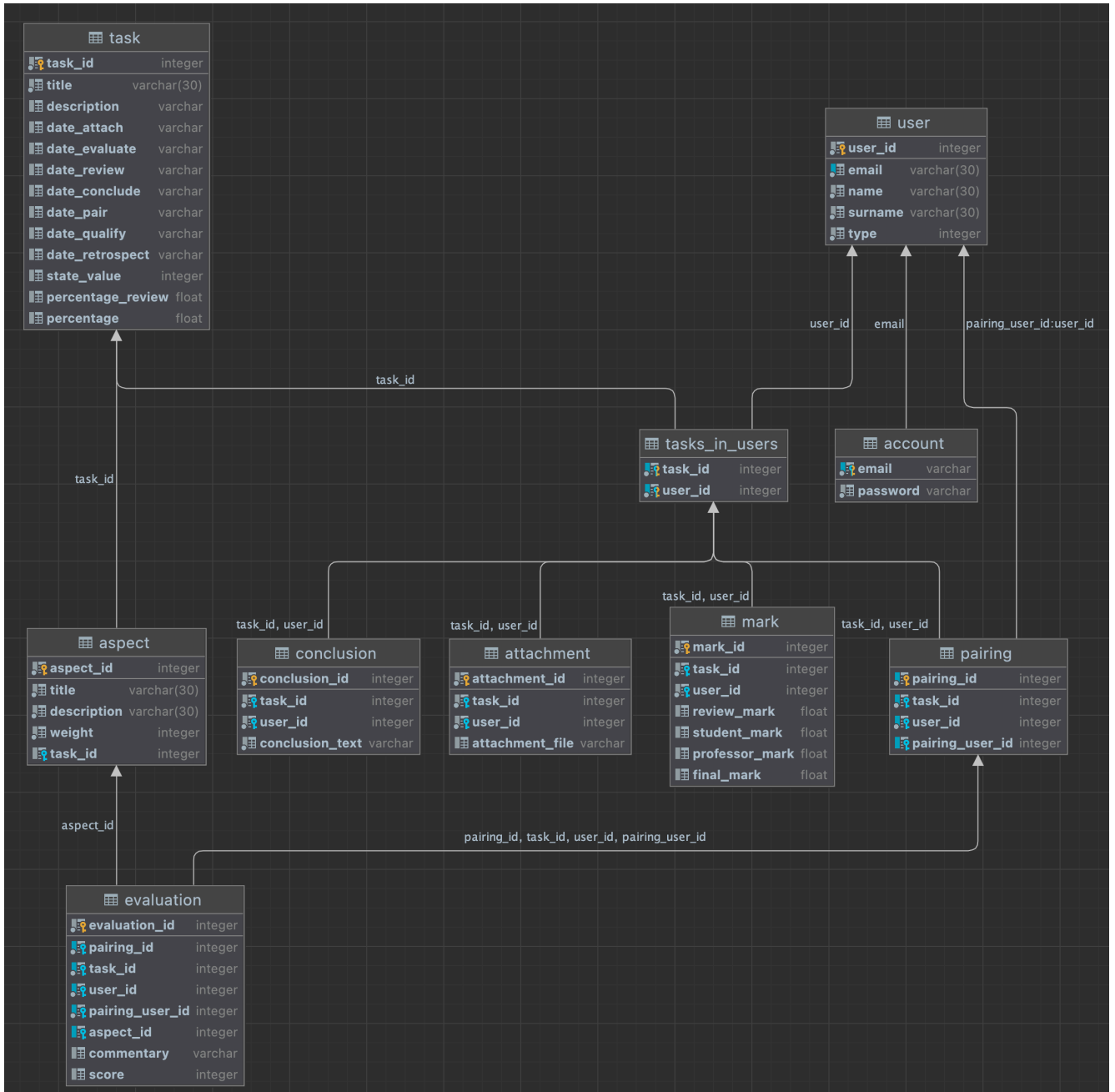


Figure 27 - Diagram of the database tables and relationships

As we can see from the image, the main tables are *user* and *task* which are related by a *tasks\_in\_users* for a many\_to\_many relationship with the Primary Key from both sides.

From the *account* table, the *account* has the attribute *email* as a Primary Key which is extended from *user* as a Foreign Key.

From the pairing table, the *pairing* has the attribute *pairing\_id* as a Primary Key. The set of attributes *task\_id*, *user\_id* is constrained as ForeignKeyConstraint which is extended from *tasks\_in\_users* as a set of Foreign Keys, and the attribute *pairing\_user\_id* is extended from *user* as a Foreign Key. Moreover, the set of attributes *task\_id*, *user\_id*, *pairing\_user\_id* is constrained as UniqueConstraint.

From the aspect table, the *aspect* has the attribute *aspect\_id* as a Primary Key. The attribute *task\_id* is extended from *task* as a Foreign Key.

From the evaluation table, the *evaluation* has the attribute *evaluation\_id* as a Primary Key. The set of attributes *task\_id*, *user\_id*, *pairing\_user\_id* is constrained as ForeignKeyConstraint which is extended from *pairing* as a set of Foreign Keys, and the attribute *aspect\_id* is extended from *aspect* as a Foreign Key. Moreover, the set of attributes *pairing\_id*, *task\_id*, *user\_id*, *pairing\_user\_id*, *aspect\_id* is constrained as UniqueConstraint.

From the attachment table, the *attachment* has the attribute *attachment\_id* as a Primary Key. The set of attributes *task\_id*, *user\_id* is constrained as ForeignKeyConstraint which is extended from *tasks\_in\_users* as a set of Foreign Keys.

From the conclusion table, the *conclusion* has the attribute *conclusion\_id* as a Primary Key. The set of attributes *task\_id*, *user\_id* is constrained as ForeignKeyConstraint which is extended from *tasks\_in\_users* as a set of Foreign Keys.

From the mark table, the *mark* has the attribute *mark\_id* as a Primary Key. The set of attributes *task\_id*, *user\_id* is constrained as ForeignKeyConstraint which is extended from *tasks\_in\_users* as a set of Foreign Keys.

#### **5.4.7. Security metrics**

The system implements several security measures to protect against unauthorized third-party intrusion. The most extensive is perhaps API authentication for all requests. Except for one endpoint, all others require authentication. When the user accesses the system, internally, the user receives a token that serves to authenticate requests in the API as if for each request the user authenticates with his username and password.

All tokens are associated with a user and have a limited lifetime. For user access to the system, the duration is two hours by default, due to that we want to avoid any loss of data during the class session which is at most two hours. After this time, they are no longer valid, and the server does not recognize them as authentication at the endpoints. The management of the tokens is entirely unrelated to the client-side. This security measure is implemented on the server-side.

## 6. Implementation

This section details the technical aspects behind the different features of the system. It will explain the implementation of the Vue Router, the stateness of the selected component as well as the pairing algorithm, and the calculation of the grading sheet of the task.

### 6.1. Vue Router

Vue Router allows the Vue application to transit from page to page on the client-side, without requesting the server. It is one of the most powerful features of modern single-page web applications. The router of the project is configured as shown in figure 28:

```
import {createRouter, createWebHashHistory} from 'vue-router'

import {appRouter} from "@modules/app/router";
import {authRouter} from "@modules/auth/router";
import {adminRouter} from "@modules/admin/router";
import {ProfessorRouter} from "@modules/professor/router"
import {StudentRouter} from "@modules/student/router";

import isAuthenticatedGuardAdmin from "@modules/auth/router/auth-guard-admin";
import isAuthenticatedGuardProfessor from "@modules/auth/router/auth-guard-professor";
import isAuthenticatedGuardStudent from "@modules/auth/router/auth-guard-student";

const routes = [
  {
    path: '/',
    ...appRouter
  },
  {
    path: '/auth',
    ...authRouter
  },
  {
    path: '/admin',
    beforeEnter: [isAuthenticatedGuardAdmin],
    ...adminRouter
  },
  {
    path: '/student',
    beforeEnter: [isAuthenticatedGuardStudent],
    ...StudentRouter
  },
  {
    path: '/professor',
    beforeEnter: [isAuthenticatedGuardProfessor],
    ...ProfessorRouter
  },
  {
    path: '/*',
    name: "notfoundpage",
    component: () => import(/* webpackChunkName: "notfound" */ '@modules/app/pages/NotFoundPage')
  }
]
```

Figure 28 - Code of router/index.js

Where the isAuthenticatedGuard<role>, as shown in figure 29, is the JavaScript file that enables the control of the transition from the *login page* to the <role> page by checking the authentication of the user. For example, in the case of the administrator, when the administrator requests to log in to the

system, it dispatches the action *checkAuthentication* from the *auth module*. The action uses the token saved on the *localStorage* to check if the token is valid on the server-side. If the token is not valid, then it will not let the user get into the system.

```
import store from "@/store";

const isAuthenticatedGuardAdmin = async (to, from, next) => {
  const {ok, type} = await store.dispatch('auth/checkAuthentication')
  if (ok && type === 2) next()
  else next({name: 'login'})
}

export default isAuthenticatedGuardAdmin
```

Figure 29 - Code of *auth/router/isAuthenticatedGuardAdmin.js*

Once the administrator gets into the system, by default, it will push the administrator to the *NoUserPage* since it is defined at the *loginAccount* method, as shown in figure 30.

```
loginAccount: async () => {
  if (userForm.value.email === '' || userForm.value.password === '') {
    new Swal({title: 'Please fill all the fields to log in', allowOutsideClick: false})
  } else {
    const {ok, data, type} = await loginUser(userForm.value)
    if (!ok) await Swal.fire('Error', data, 'error')
    else {
      if (type === 0) await router.push({name: 'no-task-student'})
      if (type === 1) await router.push({name: 'no-task-professor'})
      if (type === 2) await router.push({name: 'no-user'})
    }
  }
}
```

Figure 30 - Code of *auth/login.vue*

Only if the *id* is passed through the *route.params*, then it will push the administrator to the *UserPage* with its corresponding *id* of the user, as shown in the figure 31.

```
export const adminRouter = {
  path: '/admin',
  name: 'admin',
  component: () => import(/* webpackChunkName: "admin" */ '@modules/admin/pages/AdminPage'),
  children: [
    {
      path: '',
      name: 'no-user',
      component: () => import(/* webpackChunkName: "no-user" */ '@modules/admin/pages/NoUserPage')
    },
    {
      path: ':id',
      name: 'user',
      component: () => import(/* webpackChunkName: "user" */ '@modules/admin/pages/UserPage'),
      props: (route) => {
        const id = Number(route.params.id)
        return isNaN(id) ? {user_id: 0} : {user_id: id}
      }
    }
  ]
}
```

Figure 31 - Code of *admin/router/index.js*

## 6.2. Stateness on the selected component

To bring a better user experience when browsing the website, there are many identifiers that are saved in the *Store* provided by the Vuex framework. For example, in this case, when the professor clicked on the task, then the selected task component will be rendered with an active background color, as shown in the figure 32.

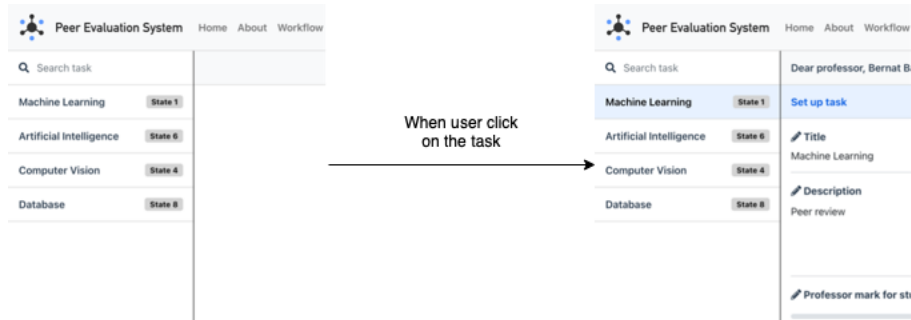


Figure 32 - Active state on selected component

### - Task component

The Task component gets two properties: the task itself and the id of the active task, as shown in figure 33.

```
name: "Task",
props: {
  task: {
    type: Object,
    required: true
  },
  activeTaskId: {
    type: Number,
    required: false
  },
},
```

Figure 33 - Code of professor/components/Task.vue

When it is rendered, there is a condition checking that the id of the Task component is equal that the id of the active task. If the condition is satisfied, then it will render the *activeBox* class, as shown in figure 34, which is defined in the *style.scss* file.

```
<template>
  <div class="task-container pointer rec-box-between"
    :class="{ 'activeBox': this.activeTaskId === this.task.task_id }"
    @click="navigateTask">
    <div class="fs-6 text-truncate ml-3 text-start ">
      <span class="fw-600"> {{ task.title }}</span>
    </div>

    <span class="badge badge-pill badge-primary bg-lightgrey m-3" style="color: black">
      State {{ this.taskState + 1 }}
    </span>
  </div>
</template>
```

Figure 34 - Code of professor/components/Task.vue



Once the user clicks on the Task component, the component will commit the mutation *setActiveTaskId* from the *professor module*, as shown in figure 35.

```
navigateTask: async () => {
  store.commit('professorModule/setActiveTaskId', props.task.task_id)
}
```

Figure 35 - Code of *professor/components/Task.vue*

### - TaskList component

The TaskList component contains the Task component which is defined as an asynchronous component, and it has an attribute that gets the state *getActiveTaskId* from the *professor module*, as shown in figure 36.

```
name: "TaskList",
components: {
  Task: defineAsyncComponent(() => import('@/modules/professor/components/Task'))
},
setup() {
  const store = useStore()
  return {
    activeTaskId: computed(() => store.getters['professorModule/getActiveTaskId']())
  }
}
```

Figure 36 - Code of *professor/components/TaskList.vue*

The Task component, which is rendered by the *v-for* directive taking a list of tasks, as shown in figure 37, has two properties assigned: the task itself and the id of the active task, which are mentioned in the above Task component.

```
<Task v-for="task in tasksByTerm"
  :key="task.task_id"
  :task="task"
  :active-task-id="this.activeTaskId">
</Task>
```

Figure 37 - Code of *professor/components/TaskList.vue*

## 6.3. Pairing algorithm

The pairing algorithm, used to automatically compute the pairings of users in the task, is designed to be simple and efficient. The approach is achieved by first getting an array of users in the task, then we apply the method *random.shuffle* to unsort this array and we extend the array with the *pairing\_number*,

which is a seed number that indicates the number of pairing\_user to be paired for each user. Finally, for each user, it will be paired to evaluate the next *pairing\_number* of pairing\_user, as shown in figure 38. This algorithm has the following features:

- The computation of the algorithm is cheap.
- The pairings created do not have any collision.
- The sequence is predictable, but it is supposed that no one shares his list of pairings.

```

users = [user.json_task() for user in UserModel.get_all() if task in user.tasks]
user_list = [u['user_id'] for u in users if u['type'] != 1]
random.shuffle(user_list)
pairing_number = data['pairing_number']

all_pairings = []
user_list.extend(user_list[:pairing_number])

for i in range(len(user_list) - pairing_number):
    for j in range(pairing_number):
        new_pairing = PairingModel(task_id, user_list[i], user_list[1 + i + j])
        new_pairing.save_to_db()

```

Figure 38 - Code of resources/task\_pairing.py

#### 6.4. Grading sheet

The grading of each student is composed of three types of marks: student\_mark, review\_mark, and professor\_mark. Both review\_mark and professor\_mark are the values provided by the professor, while the student\_mark must be computed by the evaluation marks provided by the peers.

Therefore, the student\_mark of each student is calculated by taking the mean of the following vector, as shown in equation 1.

$$Q = \frac{1}{\sum a_i} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}^T \times \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1m} \\ e_{21} & e_{22} & \cdots & e_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{nm} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix}^T$$

Equation 1 - Average mark from student's evaluation

Where the left-hand side is the vector of the weights of the aspects divided by the mean of its, and the right-hand side is the matrix of the evaluation marks provided by the peers for each aspect of the task.

Moreover, as shown in figure 39, if there is a set of missing values on the matrix of evaluation marks, then it will be replaced by the row-wise mean of the evaluation marks so that the result value will not be affected by the missing values.

```

aspect_dim = len(aspects)
aspect_vector = np.zeros(aspect_dim)

for i in range(aspect_dim):
    aspect_vector[i] = aspects[i]['weight']

user_dim = len(ps)
eval_matrix = np.zeros((aspect_dim, user_dim))

for i in range(len(evals)):
    for u in range(len(ps)):
        for a in range(len(aspects)):
            if evals[i]['aspect_id'] == aspects[a]['aspect_id'] \
                and evals[i]['user_id'] == ps[u]['user_id']:
                eval_matrix[a][u] = evals[i]['score']

aspect_vector = aspect_vector / np.sum(aspect_vector)

if np.count_nonzero(eval_matrix) != 0:
    mask_one = np.where(eval_matrix, 0, 1)
    avg_row = np.ma.array(eval_matrix, mask=eval_matrix == 0).mean(1)
    avg_fill_mark = mask_one * avg_row[:, np.newaxis]
    student_mark = eval_matrix + avg_fill_mark
    student_mark = np.mean(np.dot(aspect_vector, student_mark))
else:
    student_mark = 0

```

*Figure 39 - Code of resources/task\_user\_mark.py*

## 7. Deployment

During the development of the project, there are two environments that have been created: the development environment and the production environment. The development environment is part of a tiered structure of environments, where changes are deployed through different environments before reaching a live website. It is used to validate and test the changes during the sprints. At the end of the sprint and the release of the new version of the system, the changes are moved to the production environment.

Both environments are hosted on Heroku, the SaaS platform used for the project. Although both have the same operation and objectives, in terms of resources the production environment has more features, both in terms of availability and network traffic share.

The deployments have been automated using Automatic Deploys, a feature of Heroku where every push to the branch specified will deploy a new version of this app. In GitHub, there are using two main branches: the development branch and the production branch. Each branch corresponds to its environment.

During the development, the database used was SQLite, which despite depending on a single file, has very simple management. For the final testing and production phases, both environments have been migrated to a PostgreSQL database, which is simple to migrate by adding the add-on Heroku Postgres on the Heroku platform and it provides a better vertical scaling.

The home page of the application, as shown in figure 40, with the following domain name:

<https://peerevaluationsystem.herokuapp.com/>

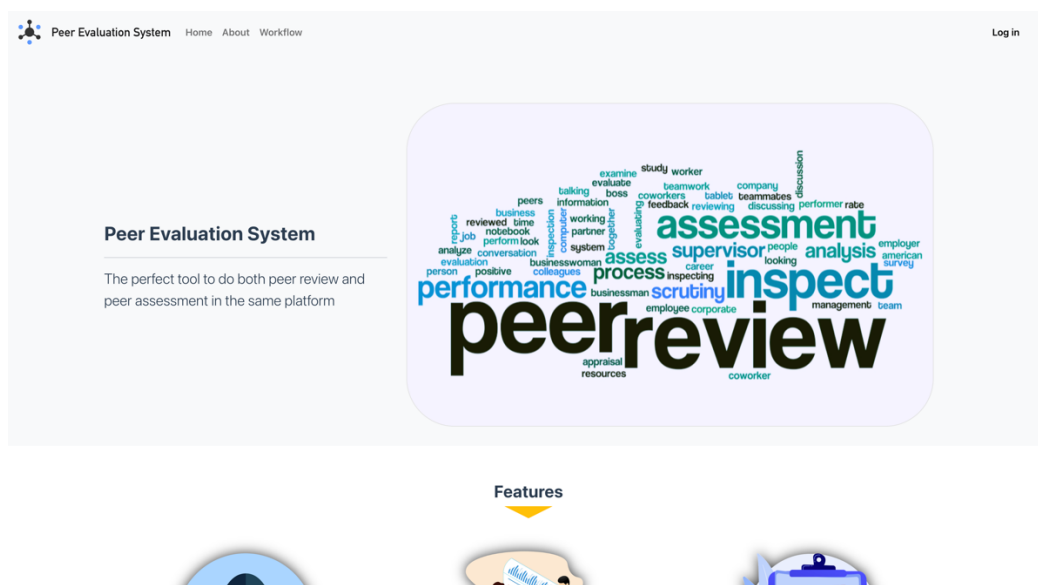


Figure 40 - Homepage of the web application



## 8. Testing and Results

### 8.1. Testing

By using the Scrum agile work methodology, a functional prototype of the system was available after each iteration. This has allowed us to perform small usability tests during the development stages, which, despite not being done with a finished product, have made it possible to identify errors in advance, thus avoiding the accumulation or aggravation of errors.

From the whole set of tests, several problems have been identified and solved, the most concrete ones are shown in the following table 14:

Issue	Description	Solution
The active state of the selected element	Once the user selects an element from the list such as a user or a task, when the webpage is refreshed, the active state of the selected element disappears	Implement an attribute that stores the unique identifier of the selected element and keep the session alive to avoid the disappearance after refresh
The active accordion between different task is maintained	When the user changes the task, the accordion component should reopen always the first one, because the state of the task blocks the locked accordion makes it unintuitive to use	Implement the emit function from mitt library that enables to toggle the first accordion of each task when the user changes the task
Too many distractions on the color of the buttons	There is different usage of color for each type of button such as add, update, remove an item	Unify the color of the action button so that it produces less distraction, and it looks more minimalistic
More awareness on the state of the task	The task displayed on both professor task page and student task page only contains the title of the task, which user cannot know the current state at first glance	Introduce a badge element from Bootstrap that shows the current state of the task
The range input for displaying non-interactive data confuses user	Some ranged data are shown with the range input from html that confuse user with the interactive property of the element	Change the read-only range data from input component to textual component
Disable the pre-requirement not satisfied button	Some buttons are display as interactable even if the pre-requirement is not satisfied	Make the buttons to be disabled when the pre-requirement is not satisfied
Update the state once students are added via csv file	When students are added via csv file, the state of the webpage is not actualized	Reload the pairings of the user once the automatic pairings are computed

Table 14 - Testing issues

## 8.2. Results

The result of the web pages is shown in annex 13.3., where the pages are sectioned by their containing module. Here will only show the home page of the application as a result which we can see in figure 41.

**Peer Evaluation System**

Home About Workflow Log in

**Peer Evaluation System**

The perfect tool to perform both **peer review** and **peer assessment** on the same platform

**peer review**

**assessment**

**inspect**

**performance**

**peer review**

**Features**

- Anonymity**  
The evaluation between peers is totally anonymized
- Peer assessment**  
Run multiple assessments to monitor team dynamics
- Quick feedback**  
Automatically obtain personalized peer and task feedback

**Vision**

Let collaboration and automation to improve the workflow of **Peer Evaluation System**

**Opinions**

- It is a really cool platform where I can manage easily my task and do the evaluations
- Wow! It is an amazing platform where I have done my peer assessment task in just 4 steps
- It was easy for me to handle my tasks during the course and get feedbacks of the task

**Peer Evaluation System**

This platform is aiming to provide a more accessible e-learning tool for both peer review and peer assessment

**Follow us**

**Directories**

- Home
- About
- Workflow
- Evaluation

**Contact**

- 695 47 64 46
- qqjun199@hotmail.com
- University of Barcelona
- peerevaluationssystem.herokuapp.com

Copyright ©2022 All rights reserved | Designed and developed by QJStudio | Demo Images: www.freepik.com

Figure 41 - Home page of the web application

## 9. Costs

The costs of the project should include both software expenditure and labor costs.

At the software level, we have the cost of the Heroku deployment. At the platform, we have used the add-on of Heroku, called Heroku Postgres, the database used in production. The fee of this add-on in the production environment (Heroku Hobby Basic plan) has a cost 9,00 dollars per month, 8,38 euros at the exchange rate. The fee of it in the development environment (Heroku Hobby Dev plan) is free, so it has not generated any expenses during the development of the project.

At the labor level, we have the cost of the development hours, to estimate the time spent, I have tracked the hours invested in programming and reading documentation from the beginning of the project by an external application. The total hours accounted for are 358 hours. Moreover, I have also taken an online course of 30 hours specialized in Vue which costs 109,00 euros.

For a more realistic approach, the development of this project would have been taken by a junior software engineer, considering working in a consulting firm, the estimated cost per hour would be around 18,00 euros. By the time that have being developing, the total estimated cost would be around 3.336,00 euros.

The estimation is intended to be generic, so they are based on data extracted from the internet. In total, the breakdown and total development cost of the project are shown in the following table 15:

Concept	Cost per unit	Amount of unit	Price
Heroku add-on subscription	8,38 € / month	6 month	50,28 €
Software development	18,00 € / hour	358 hours	6.444,00 €
Software specialization	12,00 € / hour	30 hours	360,00 €
Online course	109,00 € / course	1 course	109,00 €
Subtotal			5523,28 €
Total costs (tax included)			8425,57 €

*Table 15 - Project costs*





## 10. Future works

During the development of the system, some new ideas and improvements have emerged that could be implemented in future versions of the system. Some of these are as follows:

### 10.1. Learning Tools Interoperability (LTI)

In the latest years, IMS Global Learning Consortium, an international organization that emerged in 1995 and whose aim is to promote the growth and impact of learning technologies in education and corporate training globally, has developed an education technology specification called Learning Tools Interoperability. This specification has the purpose to invoke and communicating the internal system with external systems. The security of this communication has relied on an authentication mechanism by a token. Moreover, this tool can be extended to a Learning Management System which may use LTI to host all the contents and tools provided by the third-party systems, an external website. In this sense, the end-user does not need to log in to the external systems separately. All the information about the end-user and the learning context will be shared by the LMS with the external systems. The LTI technology was strongly inspired by Facebook Application Protocol and Blackboard Proxy Tools.

The flow of the launch sequence is shown in the figure 42:

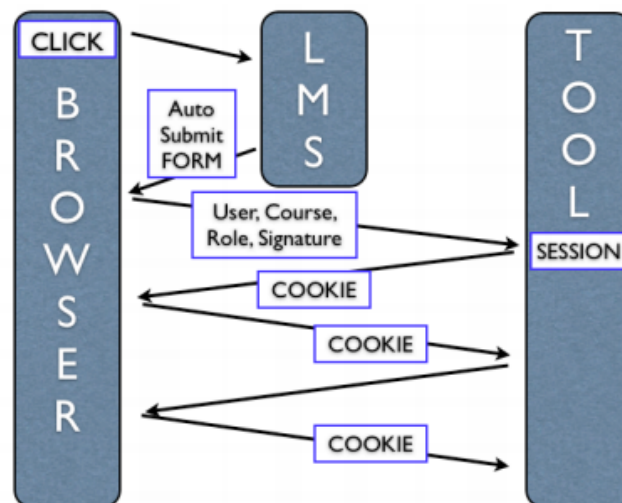


Figure 42 - LTI authentication sequence (Source: [www.wikipedia.com](http://www.wikipedia.com))

It uses the OAuth mechanism to ensure information security. The first step is to select the external learning tool using the browser of the LMS platform, once selected, the platform generates an HTML form and a JavaScript file to insert it into the external learning tool, from the moment of inserting the form, the external learning tool or system will establish bidirectional communication with the LMS

platform, the data transmission may include grades, student progress, completed tasks. In this way, LTI technology simplifies the registration mechanism between different e-learning tools.

For this reason, LTI has been adopted by many educational content providers and Learning Management Systems such as Moodle, which is used by the virtual campus of the University of Barcelona. Therefore, it would be nice to include it in this platform as a second method to authenticate the user.

### **10.2. Extend session**

Regarding the token expiration after two hours for each time after it is authenticated, the interface needs to incorporate measures against inactive sessions for a better user experience of navigation. For this reason, the client-side should implement a timer that will launch a notification 5 minutes before the session is expired and gives him the possibility to extend it. In the case that the user is absent or does not want to extend the session when the timer reaches its limit, the user should be logged out and redirected to the login page.

### **10.3. More security metrics**

As the approach to the security metrics has not been included in the main objectives of the project, there are several vulnerabilities that should consider.

First, the password of the account does not pass any strength metrics, but for security reasons, it is recommendable to apply strength on the password in terms of length, complexity, and unpredictability factors, to avoid any unintentional connections from guessing or brute-force attacks.

Second, the connection from any device to the system could be authenticated in two steps. That is a user will request to log in to the system and it will respond by sending an e-mail with a random six-digit code of one-time usage that serves as a verification code to launch the session, and only if the introduced code matches the one generated by the server, it will authorize the connection from the device.

Third, the retrieve of the password via e-mail verification. That is a user will request to retrieve the password from the system and it will respond by sending an e-mail with an URL of one-time usage with a timeout mechanism that serves as a verification method to identify the user, and only this user is authorized to retrieve the password.

## **11. Conclusion**

The main objective of the project was to design a webpage for academic uses which can able the community of students to perform peer reviews and peer assessments of a task under a set of criteria determined by the professor, where the professor is also responsible to determine the state of the workflow of the task, and both students and professors can contribute to the scoring process of the individual task delivery and task review. The result of the project not only meets the objectives but also exceeds all initial expectations with an intuitive design, multi-role system, and optimized algorithms.

It has been possible to design and develop a whole system including both the front-end and the back-end of the application along with its architecture. From the front-end of the application, it contains a clean, intuitive, modern, and adaptable interface as well as the initial approach of being a multi-role and single page application. From the back-end of the application, it contains the design of the SQL database and the API management as well as the authentication feature. All this without compromising the future lines of work along which the application can be extended.

It has been a project that has allowed me as a student to demonstrate and consolidate the capabilities to analyze, design, and develop applications. I have also been able to learn to work with many technologies that I have applied and integrated into the context of the application by applying architectures and resources seen throughout the degree.

Therefore, it can be concluded that the project has been very satisfactory and interesting in all its iterations and areas. From a personal point of view, it has given me experience and new knowledge that I can add to technological and professional learning. Finally, and after meeting the expectations, objectives, and requirements initially defined, this project can be considered completed with all the user stories covered which conclude the degree and mark the end of a stage.



## 12. References

- [1] J. M. Alzaid, "The Effect of Peer Assessment on the Evaluation Process of Students," 17 January 2017. [Online]. Available: <https://www.ccsenet.org/journal/index.php/ies/article/view/68537>.
- [2] "Workshop activity - MoodleDocs," [Online]. Available: [https://docs.moodle.org/400/en/Workshop\\_activity](https://docs.moodle.org/400/en/Workshop_activity).
- [3] "HTML: HyperText Markup Language | MDN," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML>.
- [4] "CSS: Cascading Style Sheets | MDN," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS>.
- [5] "JavaScript | MDN," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [6] "Python," [Online]. Available: <https://www.python.org>.
- [7] "Vue.js - The Progressive JavaScript Framework | Vue.js," [Online]. Available: <https://vuejs.org>.
- [8] "Vuex," [Online]. Available: <https://vuex.vuejs.org>.
- [9] "Flask," [Online]. Available: <https://flask.palletsprojects.com/en/2.1.x/>.
- [10] "Cloud Application Platform | Heroku," [Online]. Available: <https://www.heroku.com/home>.
- [11] "GitHub: Where the world builds software," [Online]. Available: <https://github.com/>.
- [12] Au-Yeung, J. "How to download a PDF file with Vue.js? The Web Dev." 2022, March 12. [Online]. Available: <https://thewebdev.info/2022/03/12/how-to-download-a-pdf-file-with-vue-js/>
- [13] Bootstrap. Bootstrap. <https://getbootstrap.com>
- [14] Ching, J. T. (2021, December 13). Manage your Python Virtual Environment with Conda - Towards Data Science. Medium. <https://towardsdatascience.com/manage-your-python-virtual-environment-with-conda-a0d2934d5195>
- [15] Data Structures — Werkzeug Documentation (2.1.x). Werkzeug. <https://werkzeug.palletsprojects.com/en/2.1.x/datastructures/>
- [16] Download file with Vue.js and Python Flask. (2019, October 8). Stack Overflow. <https://stackoverflow.com/questions/58279650/download-file-with-vue-js-and-python-flask>
- [17] Flask-RESTful - Upload image. (2015, March 11). Stack Overflow. <https://stackoverflow.com/questions/28982974/flask-restful-upload-image>
- [18] font-awesome - Libraries - cdnjs - The #1 free and open source CDN built to make life easier for developers. Cdnjs. <https://cdnjs.com/libraries/font-awesome/5.15.3>

- [19] Freepik | Graphic Resources for everyone. Freepik. <https://www.freepik.com>
- [20] GeeksforGeeks. (2019, April 22). Drop shadow for PNG image using CSS. <https://www.geeksforgeeks.org/drop-shadow-for-png-image-using-css/>
- [21] How may I upload file in Restful Flask? (2015, July 20). Stack Overflow. <https://stackoverflow.com/questions/31514568/how-may-i-upload-file-in-restful-flask>
- [22] How To Upload Files With Vue. Mastering JS. <https://masteringjs.io/tutorials/vue/file-upload>
- [23] IMS Global Learning Consortium |. LTI. <https://www.imsglobal.org>
- [24] json — JSON encoder and decoder — Python 3.10.5 documentation. Python JSON. <https://docs.python.org/3/library/json.html>
- [25] npm: font-awesome. Font-Awesome. <https://www.npmjs.com/package/font-awesome>
- [26] npm: mitt. Mitt. <https://www.npmjs.com/package/mitt>
- [27] npm: v-calendar. V-Calendar. <https://www.npmjs.com/package/v-calendar>
- [28] npm: vue3-simple-file-input. Vue3-Simple-File-Input. <https://www.npmjs.com/package/vue3-simple-file-input>
- [29] npm: vue-csv-import. Vue.Js Component to Handle CSV Uploads with Field Mapping. <https://www.npmjs.com/package/vue-csv-import>
- [30] npm: @vueuse/motion. (2022, March 16). @vueuse/Motion. <https://www.npmjs.com/package/@vueuse/motion>
- [31] NumPy. Numpy. <https://numpy.org>
- [32] Oberlehner, M. \$refs and the Vue 3 Composition API. Markus Oberlehner. <https://markus.oberlehner.net/blog/refs-and-the-vue-3-composition-api/>
- [33] Others-how to upload file using python flask-restful api ? (2021, March 16). Bswen. <https://bswen.com/2021/03/others-how-to-upload-file-using-flask-restful-api.html>
- [34] PostgreSQL: Documentation. The PostgreSQL Global Development Group. <https://www.postgresql.org/docs/>
- [35] Python Flask REST API File Upload Example. (2021, February 20). Roy Tutorials. <https://roytuts.com/python-flask-rest-api-file-upload/>
- [36] python-decouple. PyPI. <https://pypi.org/project/python-decouple/>
- [37] Sass: Syntactically Awesome Style Sheets. Sass. <https://sass-lang.com>

- [38] SQLAlchemy - The Database Toolkit for Python. SQLAlchemy. <https://www.sqlalchemy.org>
- [39] Stack Overflow - Where Developers Learn, Share, & Build Careers. Stack Overflow. <https://stackoverflow.com>
- [40] Twarog, A. (2020, September 21). How to Design a Website Prototype from a Wireframe. freeCodeCamp.Org. <https://www.freecodecamp.org/news/designing-a-website-ui-with-prototyping/>
- [41] Vue 3 Event Bus with Composition API. (2021, March 8). Stack Overflow. <https://stackoverflow.com/questions/66537320/vue-3-event-bus-with-composition-api>
- [42] Vue JS – Add Class To Element On Click Tutorial Example. (2022, May 30). Tuts Make. <https://www.tutsmake.com/vue-js-add-class-to-element-on-click-tutorial-example/>
- [43] Vue Js 2 Download File with Axios Example Tutorial. (2022, June 1). positronX.io. <https://www.positronx.io/vue-js-download-file-with-axios-example-tutorial/>
- [44] Vue Router. Vue Router. <https://router.vuejs.org>
- [45] Vue.js Download PDF with Axios. (2022, March 16). Shouts.Dev. <https://shouts.dev/articles/vuejs-download-pdf-with-axios>





## 13. Annex

### 13.1. User stories

The user stories are detailed with the following structure as a template.

User Story	As a [role]	I want to [action]	So that I can [outcome]
Route	<route>		
Functionalities	<RESTful API method>		
Pre-requirements	<Pre-requirements>		
AC	<Acceptance criteria>		
	<Launch error message>		
Ending	<Actions performed>		
	<Launch success message>		

US1	Professor, Student	Upload a form data of the sign-up account	Get access to the platform
Route	/register		
Functionalities	POST		
Pre-requirements	The user's email must be added previously by professor in the database		
AC1	If form fields are not completed		
	Launch error message: "Please fill all the fields to sign up"		
AC2	If the user is not found in the database		
	Launch error message: "User with ['email': {}] not found"		
AC3	If user already registered in the database		
	Launch error message: "Account with ['email': {}] already exists"		
Ending	The user gets registered in the database		
	Launch success message: "The account is saved successfully"		

US2	Administrator, Professor, Student	Upload a form data of the log in account	Get into the platform
Route	/login		
Functionalities	POST		
Pre-requirements	The user's account must be registered previously in the database		

AC1	If form fields are not completed
	Launch error message: "Please fill all the fields to log in"
AC2	If the account is not found in the database
	Launch error message: "Account with ['email': {}] not found"
AC3	If the password does not match to the account in the database
	Launch error message: "Password does not match to the account with ['email': {}]"
Ending	The user gets logged in the system and is routed to its corresponding page
	N/A

US3	Administrator, Professor, Student	Log out of the platform	Leave from the session
Route	/admin, /professor, /student		
Functionalities	N/A		
Pre-requirements	The user's account must be logged in previously in the platform		
Ending	The user's session data is cleaned and is routed to '/login'		
	N/A		

US4	Administrator	Upload a form data of the user	Create a user in the system
Route	/admin		
Functionalities	POST		
Pre-requirements	The administrator must be logged in to the system		
AC1	If form fields are not completed		
	Launch error message: "Please fill all the fields to save user"		
AC2	If the user already exists in the database		
	Launch error message: "User with ['user_id': {}] already exists"		
Ending	The user is added by administrator		
	Launch success message: "The user is added successfully"		

US5	Administrator	Upload a new form data of the user	Change the data of the user in the system
-----	---------------	------------------------------------	---

Route	/admin
Functionalities	PUT
Pre-requirements	The administrator must be logged in to the system
AC1	If form fields are not completed
	Launch error message: "Please fill all the fields to save user"
Ending	The user is updated by administrator
	Launch success message: "The user is saved successfully"

US6	Administrator	Click the remove user button	Remove the user from the system
Route	/admin		
Functionalities	DELETE		
Pre-requirements	The administrator must be logged in to the system		
	The system will request the administrator to ensure deletion: "Are you sure to delete?"		
AC1	If the user is not found in the database		
	Launch error message: "User with ['user_id': {}] not found"		
Ending	The user is removed from the list of users by administrator		
	Launch success message: "The user is deleted successfully"		

US7	Administrator	Click the remove user's account button	Remove the account of the user from the system
Route	/admin		
Functionalities	DELETE		
Pre-requirements	The administrator must be logged in to the system		
	The system will request the administrator to ensure deletion: "Are you sure to delete?"		
AC1	If the account is not found in the database		
	Launch error message: "Account with ['email': {}] not found"		
Ending	The account is removed from the list of accounts by administrator		
	Launch success message: "The account is deleted successfully"		

US8	Professor	Upload a form data of the task	Create a task space for students
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
AC1	If form fields are not completed		
	Launch error message: "Please fill all the fields to save task"		
AC2	If the user is not found in the database		
	Launch error message: "User with ['user_id': {}] not found"		
AC3	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	The task is added by professor		
	Launch success message: "The task is added successfully"		

US9	Professor	Upload a new form data of the task	Change the data of the task in the system
Route	/professor		
Functionalities	PUT		
Pre-requirements	The professor must be logged in to the system		
AC1	If form fields are not completed		
	Launch error message: "Please fill all the fields to save task"		
AC2	If the user is not found in the database		
	Launch error message: "User with ['user_id': {}] not found"		
AC3	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	The task is updated by professor		
	Launch success message: "The task is updated successfully"		

US10	Professor	Click the remove task button	Remove the task from the list of tasks
Route	/professor		
Functionalities	DELETE		

Pre-requirements	The professor must be logged in to the system
	The system will request the professor to ensure deletion: "Are you sure to delete?"
AC1	If the user is not found in the database
	Launch error message: "User with ['user_id': {}] not found"
AC2	If the task is not found in the database
	Launch error message: "Task with ['task_id': {}] not found"
Ending	The task is removed from the list of tasks by professor
	Launch success message: "The task is deleted successfully"

US11	Professor	Upload a form of student	Create a student to the task
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
AC1	If form fields are not completed		
	Launch error message: "Please fill all the fields to save user"		
AC2	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	The student is added to the task by professor		
	Launch success message: "The student is added to the task successfully"		

US12	Professor	Upload a CSV file of students	Create a list of students to the task
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
AC1	If input file is not chosen		
	Launch error message: "Please attach a csv to add students"		
AC2	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	The list of students is added to the task by professor		

	Launch success message: "The list of students is added to the task successfully"
--	--

US13	Professor	Click the remove student button	Remove the student from the list of assigned students to the task
Route	/professor		
Functionalities	DELETE		
Pre-requirements	The professor must be logged in to the system		
	The system will request the professor to ensure deletion: "Are you sure to delete?"		
AC1	If the user is not found in the database		
	Launch error message: "User with ['user_id': {}] not found"		
AC2	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	The student is deleted from task by the professor		
	Launch success message: "The student is deleted from the task successfully"		

US14	Professor	Upload a form of the aspect	Set a criteria for evaluation
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
AC1	If form fields are not complete		
	Launch error message: "Please fill all the fields to save aspect"		
AC2	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	The aspect is added to the task by professor		
	Launch success message: "The aspect is added to the task successfully"		

US15	Professor	Upload a new form data of the aspect	Change the data of the criteria for evaluation
Route	/professor		
Functionalities	PUT		

Pre-requirements	The professor must be logged in to the system
AC1	If aspect is not selected
	Launch error message: "Please select an aspect to save aspect"
AC2	If form fields are not complete
	Launch error message: "Please fill all the fields to save aspect"
AC3	If the task is not found in the database
	Launch error message: "Task with ['task_id': {}] not found"
Ending	The aspect is updated to the task by professor
	Launch success message: "The aspect is updated to the task successfully"

US16	Professor	Click the remove aspect button	Remove the aspect from the list of assigned aspects to the task
Route	/professor		
Functionalities	DELETE		
Pre-requirements	The professor must be logged in to the system		
	The system will request the professor to ensure deletion: "Are you sure to delete?"		
AC1	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
AC2	If the aspect is not found in the database		
	Launch error message: "Aspect with ['aspect_id': {}] not found"		
Ending	The aspect is removed from the task by professor		
	Launch success message: "The aspect is deleted from the task successfully"		

US17	Professor	Upload a form of pairing	Create a pairing for evaluation
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
	The student must be added previously by the professor to the task		
AC1	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		



Ending	The pairing is added to the task by professor
	Launch success message: "The pairing is added to the task successfully"

US18	Professor	Compute automatically the pairings	Create a list of pairings for evaluation
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
	The set of students must be added previously by the professor to the task		
AC1	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
Ending	A list of pairings is added to task by professor		
	Launch success message: "The list of pairings is added to the task successfully"		

US19	Professor	Click the remove pairing button	Remove the pairing from the list of pairings
Route	/professor		
Functionalities	DELETE		
Pre-requirements	The professor must be logged in to the system		
	The system will request the professor to ensure deletion: "Are you sure to delete?"		
AC1	If the pairing is not found in the database		
	Launch error message: "Pairing with ['pairing_id': {}] not found"		
Ending	The pairing is removed from task by professor		
	Launch success message: "The pairing is deleted from the task successfully"		

US20	Professor, Student	Click the attachment download button	Check the peer's task delivery
Route	/professor, /student		
Functionalities	GET		
Pre-requirements	The professor or the student must be logged in to the system		
	Attachment must be added previously by the paired student in the database		

AC1	If the attachment is not found in the database
	Launch error message: "Attachment with ['task_id': {}, 'user_id': {}] not found"
Ending	The attachment is downloaded by professor or student
	Launch success message: "The attachment is added to the task successfully"

US21	Professor	Click see review qualification button	Check the result of the review qualification
Route	/professor		
Functionalities	GET		
Pre-requirements	The professor must be logged in to the system		
AC1	If the review mark is not found in the database		
	Launch error message: "Mark with ['task_id': {}, 'user_id': {}] not found"		
Ending	The review mark is displayed on the website		
	N/A		

US22	Professor	Upload a form of review qualification	Create a review qualification for the peer review
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
Ending	The review mark is added by the professor		
	Launch success message: "The review qualification is added to the student successfully"		

US23	Professor	Click see general qualification button	Check the result of the general qualification
Route	/professor		
Functionalities	GET		
Pre-requirements	The professor must be logged in to the system		
AC1	If the mark is not found in the database		
	Launch error message: "Mark with ['task_id': {}, 'user_id': {}] not found"		
Ending	The mark is displayed on the website		

	N/A
--	-----

US24	Professor	Upload a form of general qualification	Create a general qualification for the peer
Route	/professor		
Functionalities	POST		
Pre-requirements	The professor must be logged in to the system		
Ending	The mark is added by the professor		
	Launch success message: "The qualification is added to the student successfully"		

US25	Professor	Click grading sheet download button	Check the grading sheet of the students
Route	/professor		
Functionalities	GET		
Pre-requirements	The professor must be logged in to the system		
AC1	If the marks are not found in the database		
	Launch error message: "Marks with ['task_id': {}] not found"		
Ending	The output of marks is downloaded by the professor		
	N/A		

US26	Professor	Click increase / decrease state button	Manage the state of the task
Route	/professor		
Functionalities	PUT		
Pre-requirements	The professor must be logged in to the system		
AC1	If the task is not found in the database		
	Launch error message: "Task with ['task_id': {}] not found"		
AC2	If the value of task's state is out of range		
	Launch error message: "Task with ['task_id': {}] state not changed"		
Ending	The state of the task is modified by the professor		
	Launch success message: "The state of task is updated successfully"		

US27	Student	Upload a file of attachment	Be evaluated by other peers
Route	/student		
Functionalities	POST		
Pre-requirements	The student must be logged in to the system		
	The state of the task must be matched to the phase		
	An attachment must be chosen previously by the student		
Ending	The attachment is added by student		
	Launch success message: "The attachment is added successfully"		

US28	Student	Upload a form of evaluation	Create an evaluation for the peer
Route	/student		
Functionalities	POST		
Pre-requirements	The student must be logged in to the system		
	The state of the task must be matched to the phase		
	At least one aspect must be added previously by the professor		
	At least one paired student must be assigned to the student previously by the professor		
AC1	If form fields are not complete		
	Launch error message: "Please fill all the fields to add evaluation"		
AC2	If the evaluation already exists in the database		
	Launch error message: "Evaluation with ['pairing_id': {}, 'aspect_id': {}] already exists"		
Ending	The evaluation is added by student		
	Launch success message: "The evaluation is added successfully"		

US29	Student	Click see evaluation button	Check the result of the evaluations of the task
Route	/student		
Functionalities	GET		
Pre-requirements	The student must be logged in to the system		
	The state of the task must be matched to the phase		
Ending	The evaluations by paired students are displayed on the website		

	N/A
--	-----

US30	Student	Upload a form of conclusion	Create a conclusion for the task
Route	/student		
Functionalities	PUT		
Pre-requirements	The student must be logged in to the system		
	The state of the task must be matched to the phase		
AC1	If form field is not complete		
	Launch error message: "Please fill the field to add conclusion"		
Ending	The conclusion is added by the student		
	Launch success message: "The conclusion is added successfully"		

### 13.2. API documentation

The API documentation is detailed with the following structure as a template.

<Resource Class>	
Endpoint	<URL>
Description	<Description>
<Method>	
Roles	<Roles>
Params	<Params>
<Status code>	<Resource>

Account	
Endpoint	/account
Description	Management of an account
DELETE	
Roles	administrator
Params	{"email": <str>}
200	{"account": {...}}
401	"Unauthorized access"
404	"Account with ['email': <str>] not found"
500	"An error occurred while deleting the account"

Signup	
Endpoint	/signup
Description	Register an account
POST	
Roles	administrator, professor, student
Params	{"email": <str>, "password": <str>}
201	{"account": {...}}
401	"Unauthorized access"

404	"User with ['email': <str>] not found"
409	"Account with ['email': <str>] already exists"
500	"An error occurred while inserting the account"

Login	
Endpoint	/login
Description	Authenticate a user by the registered account
POST	
Roles	administrator, professor, student
Params	{"email": <str>, "password": <str>}
200	{"account": {...}}
400	"Password does not match to the account with ['email': <str>]"
404	"Account with ['email': <str>] not found"

TaskAspect	
Endpoint	/task/<int>, /task/<int>/aspect/<int>
Description	Management of an aspect of the task
POST	
Roles	professor
Params	{"title": <str>, "description": <str>, "weight": <int>}
201	{"aspect": {...}}
401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
500	"An error occurred while inserting the aspect"
DELETE	
Roles	professor
200	{"aspect": {...}}
401	"Unauthorized access"
404	"Task with ['task_id': {}] not found"
404	"Aspect with ['aspect_id': {}] not found"
500	"An error occurred while deleting the aspect"

PUT	
Roles	professor
Params	{"title": <str>, "description": <str>, "weight": <int>}
200	{"aspect": {...}}
201	{"aspect": {...}}
401	"Unauthorized access"
404	"Task with ["task_id": <int>] not found"
500	"An error occurred while inserting the aspect"
500	"An error occurred while updating the aspect"

TaskMarkList	
Endpoint	/task/<int>/marks
Description	Obtainment of a final grading sheet of the task
GET	
Roles	professor
N/A	flask_csv.send_csv()
401	"Unauthorized access"
404	"Marks with ["task_id": <int>] not found"
500	"An error occurred while getting the marks"

TaskPairingList	
Endpoint	/task/<int>/pairings
Description	Compute a list of pairings of the task automatically by a seed number
POST	
Roles	professor
Params	{"pairing_number": <int>}
200	{"pairings": [...]}
401	"Unauthorized access"
404	"Task with ["task_id": <int>] not found"
500	"An error occurred while inserting the pairings"



TaskPairingUserPairingList	
Endpoint	/task/<int>/pairing_user/<int>/pairings
Description	Obtainment of a list of pairings assigned to the pairing_user of the task
GET	
Roles	student
200	{"pairings": [...]}
401	"Unauthorized access"

TaskState	
Endpoint	/task/<int>/state
Description	Management of the state of the task
GET	
Roles	professor
201	{"task": {...}}
401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
PUT	
Roles	professor
Params	{"state_value": <int>}
200	{"task": {...}}
400	"Task with ['task_id': <int>] state not changed"
401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
500	"An error occurred while updating the task"

TaskUser	
Endpoint	/task/<int>/user/<int>
Description	Management of a user of the task
DELETE	
Roles	professor
200	{"user": {...}}

401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
404	"User with ['user_id': <int>] not found"
500	"An error occurred while deleting the user"
PUT	
Roles	professor
Params	{"email": <str>, "name": <str>, "surname": <str>, "type": <int>}
200	{"user": {...}}
201	{"user": {...}}
401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
404	"User with ['user_id': <int>, 'email': <str>] not matched"
404	"User with ['user_id': <int>, 'email': <str>] not found"
500	"An error occurred while inserting the user"

TaskUserList	
Endpoint	/task/<int>/users
Description	Management of a list of users of the task
GET	
Roles	professor
200	{"users": [...]}
401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
PUT	
Roles	professor
Params	reqparse.request.data
201	{"users": [...]}
401	"Unauthorized access"
404	"Task with ['task_id': <int>] not found"
500	"An error occurred while inserting the users"

TaskUserAttachment	
Endpoint	/task/<int>/user/<int>/attachment
Description	Management of an attachment of the user of the task
GET	
Roles	professor, student
N/A	send_file()
401	"Unauthorized access"
404	"Attachment with ['task_id': <int>, 'user_id': <int>] not found"
PUT	
Roles	student
Params	{"attachment_file": <werkzeug.datastructures.FileStorage>}
200	{"attachment": {...}}
201	{"attachment": {...}}
401	"Unauthorized access"
500	"An error occurred while inserting the attachment"
500	"An error occurred while updating the attachment"

TaskUserConclusion	
Endpoint	/task/<int>/user/<int>/conclusion
Description	Management of a conclusion of the user of the task
GET	
Roles	professor, student
200	{"conclusion": {...}}
401	"Unauthorized access"
PUT	
Roles	student
Params	{"conclusion_text": <str>}
200	{"conclusion": {...}}
201	{"conclusion": {...}}
401	"Unauthorized access"
500	"An error occurred while inserting the conclusion"

500	"An error occurred while updating the conclusion"
TaskUserMark	
Endpoint	/task/<int>/user/<int>/mark
Description	Management of a mark of the user of the task
GET	
Roles	professor
200	{"mark": {...}}
401	"Unauthorized access"
404	"Mark with ['task_id': <int>, 'user_id': <int>] not found"
PUT	
Roles	professor
Params	{"professor_mark": <float>}
200	{"mark": {...}}
201	{"mark": {...}}
401	"Unauthorized access"
500	"An error occurred while inserting the evaluation"
500	"An error occurred while updating the evaluation"

TaskUserMarkReview	
Endpoint	/task/<int>/user/<int>/mark_review
Description	Management of a review_mark of the user of the task
PUT	
Roles	professor
Params	{"review_mark": <float>}
200	{"mark": {...}}
201	{"mark": {...}}
401	"Unauthorized access"
500	"An error occurred while inserting the evaluation"
500	"An error occurred while updating the evaluation"

TaskUserPairingList	
---------------------	--

Endpoint	/task/<int>/user/<int>/pairings
Description	Obtainment of a list of pairings assigned to the user of the task
GET	
Roles	professor, student
200	{"pairings": [...]}
401	"Unauthorized access"

TaskUserPairingUserPairing	
Endpoint	/task/<int>/user/<int>/pairing_user/<int>/pairing
Description	Management of a pairing of a user to a pairing_user of the task
POST	
Roles	professor
201	{"pairing": {...}}
401	"Unauthorized access"
409	"Pairing with ['task_id': <int>, 'user_id': <int>, 'pairing_user_id': <int>] already exists"
500	"An error occurred while inserting the pairing"

Pairing	
Endpoint	/pairing/<int>
Description	Management of a pairing of the task
DELETE	
Roles	professor
200	{"pairing": {...}}
401	"Unauthorized access"
404	"Pairing with ['pairing_id': <int>] not found"
500	"An error occurred while deleting the pairing"

PairingAspectEvaluation	
Endpoint	/pairing/<int>/aspect/<int>/evaluation
Description	Management of an evaluation of the aspect of the pairing

POST	
Roles	student
Params	{"commentary": <str>, "score": <int>}
200	{"evaluation": {...}}
401	"Unauthorized access"
409	"Evaluation with ['pairing_id': <int>, 'aspect_id': <int>] already exists"
500	"An error occurred while inserting the evaluation"

PairingEvaluationList	
Endpoint	/pairing/<int>/evaluations
Description	Obtainment of a list of evaluations of the pairing
GET	
Roles	professor, student
200	{"evaluations": [...]}
401	"Unauthorized access"

User	
Endpoint	/user/<int>
Description	Management of a user
POST	
Roles	admin
Params	{"email": <str>, "name": <str>, "surname": <str>, "type": <int>}
201	{"user": {...}}
401	"Unauthorized access"
409	"User with ['user_id': <int>] already exists"
500	"An error occurred while inserting the user"
DELETE	
Roles	admin
200	{"user": {...}}
401	"Unauthorized access"
404	"User with ['user_id': <int>] not found"

500	"An error occurred while deleting the user"
PUT	
Roles	admin
Params	{"email": <str>, "name": <str>, "surname": <str>, "type": <int>}
200	{"user": {...}}
201	{"user": {...}}
401	"Unauthorized access"
500	"An error occurred while inserting the user"
500	"An error occurred while updating the user"

UserList	
Endpoint	/users
Description	Obtainment of a list of users
GET	
Roles	admin
200	{"user": [...]}
401	"Unauthorized access"

UserTask	
Endpoint	/user/<int>/task/<int>
Description	Management of a task of the user
POST	
Roles	professor
Params	{"title": <str>, "description": <str>, "date_attach": <str>, "date_evaluate": <str>, "date_review": <str>, "date_conclude": <str>, "date_pair": <str>, "date_qualify": <str>, "date_retrospect": <str>, "percentage_review":<float>, "percentage":<float>}
201	{"task": {...}}
401	"Unauthorized access"
404	"User with ['user_id': <int>] not found"
500	"An error occurred while inserting the user"
DELETE	

Roles	professor
200	{"task": {...}}
401	"Unauthorized access"
404	"User with ['user_id': <int>] not found"
404	"Task with ['task_id': <int>] not found"
500	"An error occurred while deleting the user"
PUT	
Roles	professor
Params	{"title": <str>, "description": <str>, "date_attach": <str>, "date_evaluate": <str>, "date_review": <str>, "date_conclude": <str>, "date_pair": <str>, "date_qualify": <str>, "date_retrospect": <str>, "percentage_review":<float>, "percentage":<float>}
200	{"task": {...}}
201	{"task": {...}}
401	"Unauthorized access"
404	"User with ['user_id': <int>] not found"
404	"Task with ['task_id': <int>] not found"
500	"An error occurred while inserting the user"
500	"An error occurred while updating the user"

UserTaskList	
Endpoint	/user/<int>/tasks
Description	Obtainment of a list of tasks of the user
GET	
Roles	professor, student
200	{"tasks": [...]}
401	"Unauthorized access"






### 13.3. Web page results

#### 13.3.1. App module

- Home page

Home page of the application

## - About page

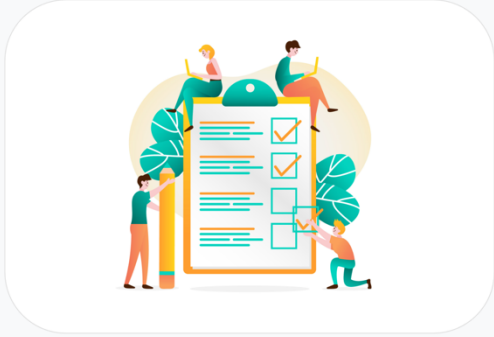
 Peer Evaluation System [Home](#) [About](#) [Workflow](#) [Log in](#)

---

### About Project

This project is focused on the workflow of the whole period of evaluation of the workshop from participating in the task to getting scored after the task is finished. Its goal is to analyze the current workshop on the virtual campus of the University of Barcelona and then synthesize the core requirements of the system into the designed web application that serves as a third-party platform that is more intuitive for both students and professors to get into.


By this, a whole sequence of the evaluation will take place at a web application which is the platform where students can participate in the process of evaluating other students in the same task. While the professor is responsible for evaluating both the evaluations given by the students and the attached files.



### Acknowledgements

I would like to thank Dr. Eloi Puertas, the tutor of this fieldwork, for all the help and support he has given to me during the completion of my project, for encouraging me to be strong, and for the constant availability of the meetings which guided me through the project with good results.





To all the professors and department of the Faculty of Mathematics and Computer Science of the University of Barcelona who have made my training in the field of Computer Science possible and taught me to work in a team and think to adapt to any situation, thus making me a valuable piece in any situation future project.



#### Peer Evaluation System

This platform is aiming to provide a more accessible learning tool for peer assessment task





**Follow us**

#### Directories

[Home](#)  
[About](#)  
[Workflow](#)  
[Evaluation](#)

#### Contact

 695 47 64 46  
 [qijunjin1996@hotmail.com](mailto:qijunjin1996@hotmail.com)  
 [University of Barcelona](#)  
 [peerevaluationsystem.herokuapp.com](#)

Copyright ©2022 All rights reserved | Designed and developed by QJStudio | Demo Images: [www.freepik.com](http://www.freepik.com)

## About page of the application

## - Workflow page

Peer Evaluation System

[Home](#)
[About](#)
[Workflow](#)

[Log in](#)

**Professor workflow**

**Student workflow**

**1 SET UP TASK**

is designed for the professor to add all primitive data that composes the task

**2 ADD PEERS**

is designed for the professor to add users that compose the task. It could be manually introduced or via csv file

**4 ASS ASPECTS**

is designed for the professor to add aspects that composes the task

**5 PAIR PEERS**

is designed for the professor to add a pairing of a student that will review and provide a commentary and score to the pairing student

**7 QUALIFY PEERS**

is designed for the professor to add qualifications of the attached file and reviews that are made by the student

**10 RETROSPECT TASK**

is designed for the professor to review all the conclusions that are left by the students after the evaluation session

**3 ATTACH FILE**

is designed for the student to add a submission of a file that will be evaluated by the professor and other students

**6 EVALUATE PEERS**

is designed for the student to add evaluations of the attached file by other students

**8 REVIEW EVALUATIONS**

is designed for the student to review evaluations made by other students

**9 CONCLUDE TASK**

is designed for the student to write a conclusion of the overall evaluation of the task

**Peer Evaluation System**

This platform is aiming to provide a more accessible learning tool for peer assessment task

**Follow us**

**Directories**

[Home](#)

[About](#)

[Workflow](#)

[Evaluation](#)

**Contact**

[695 47 64 46](#)

[qijunjin1996@hotmail.com](mailto:qijunjin1996@hotmail.com)

[University of Barcelona](#)

[peerevaluationsystem.herokuapp.com](https://peerevaluationsystem.herokuapp.com)

Copyright ©2022 All rights reserved | Designed and developed by QJStudio | Demo Images: www.freepik.com

Workflow page of the application

## - Not found page



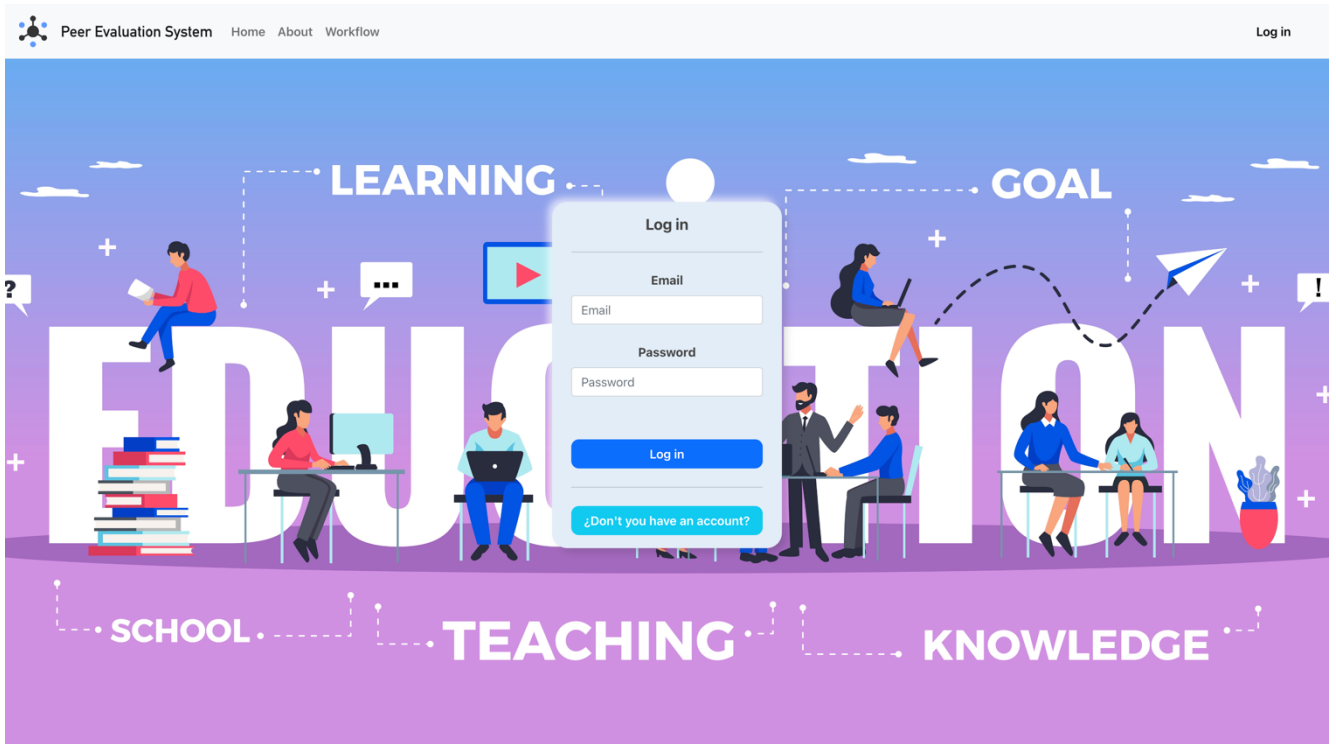
### 404 - Page Not Found

This page no longer exists or was moved to another location

Not found page of the application

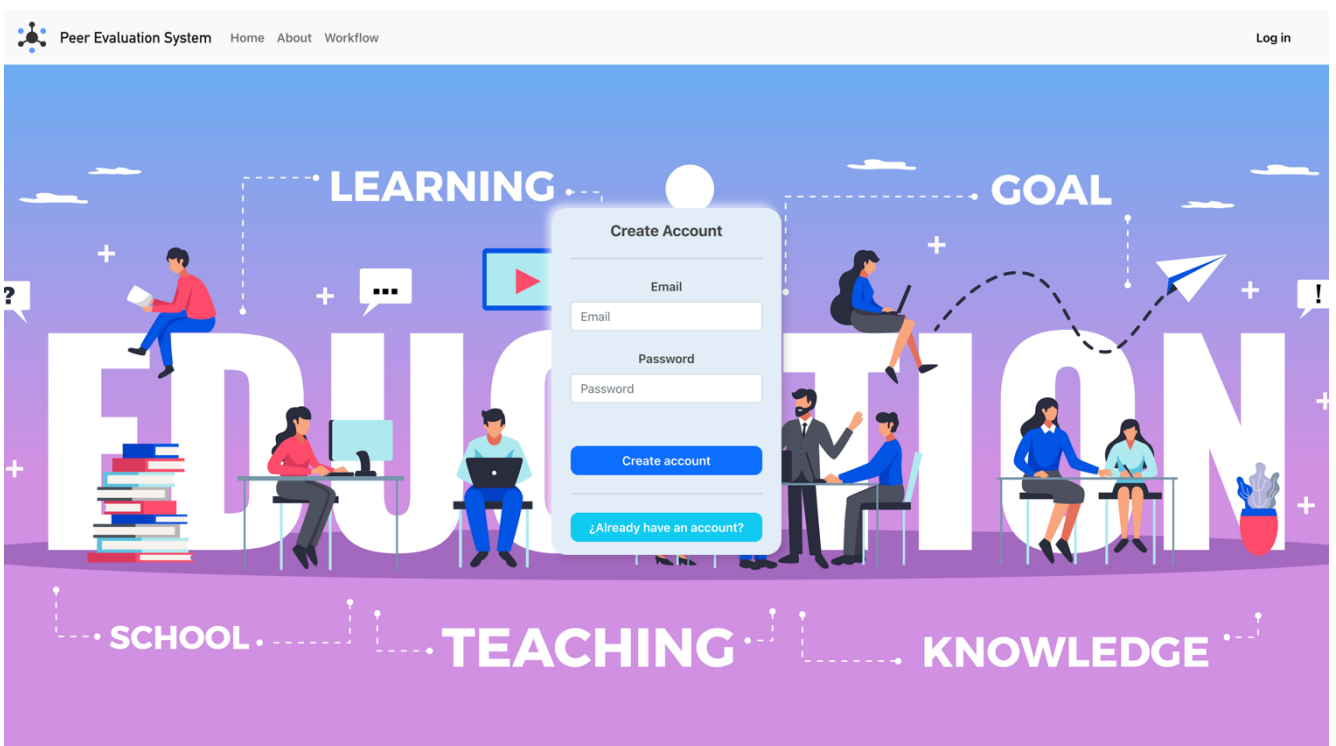
### 13.3.2. Auth module

- Auth page with log in component



Auth page of the application

- Auth page with create account component



Auth page of the application

### 13.3.3. Admin module

#### - Admin page (default)

The screenshot shows the Admin page of the Peer Evaluation System. The header includes the system name, navigation links (Home, About, Workflow), and buttons for 'Go to admin page' and 'Log out'. A search bar is present at the top left. A list of users is displayed on the left side, each with a name and a role (administrator or student). The main content area contains the message 'Please select one user on left bar'.

Search user	
Albert Abad	administrator
Bernat Bartomeu	professor
Elisabet Esteve	student
Francesc Fité	student
Glòria Guasch	student
Helena Herrera	student
Iván Ibáñez	student
Jordi Jin	student
Kàtia Klein	student
Lídia Llupià	student
Maria Moyà	student
Narcís Nadal	student
Oriol Ortiz	student
Pau Planes	student
Quim Quiròs	student

Admin page of the application

#### - Admin user subpage

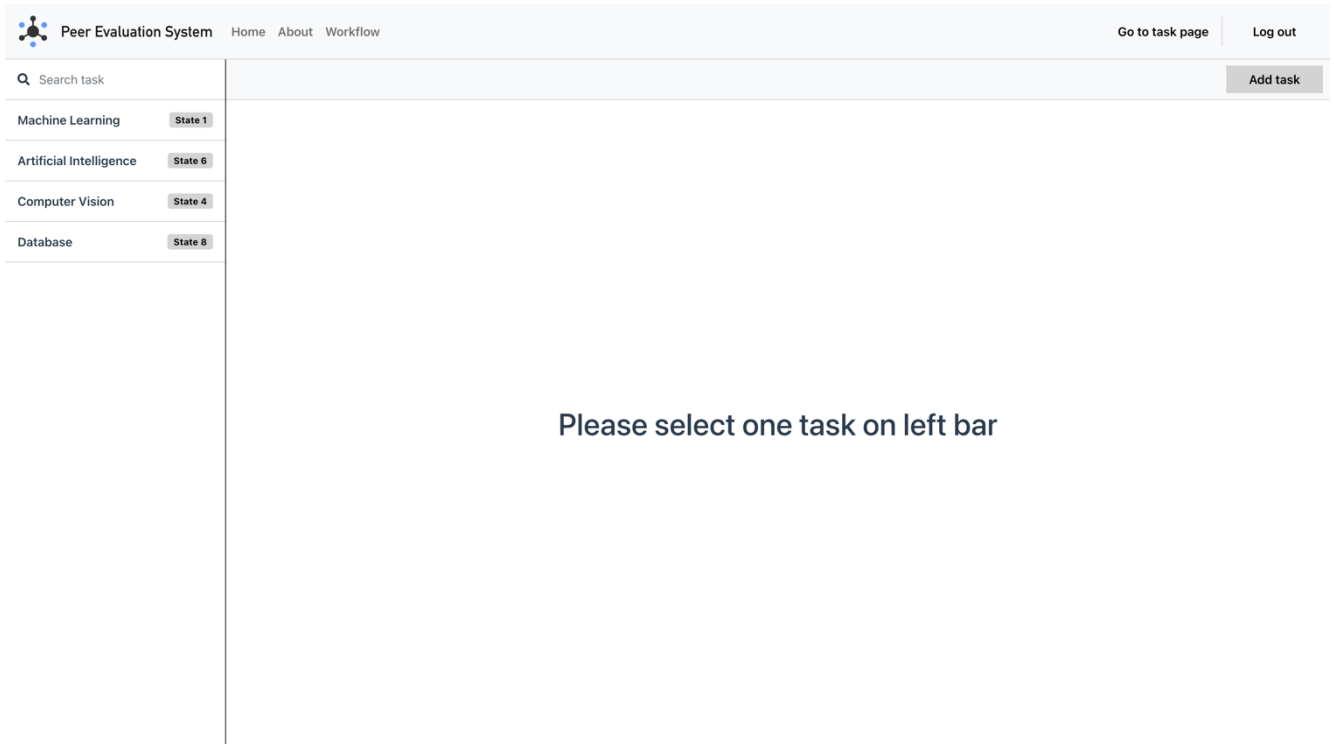
The screenshot shows the Admin user subpage for 'Albert Abad'. The header is the same as the previous page. The left sidebar shows the user list with 'Albert Abad' selected. The main content area displays a form for editing the user's details, including ID, Name, Surname, Email, and Role. The role is currently set to 'Administrator'. Buttons for 'Add user', 'Delete user', and 'Save user' are visible.

Search user	Dear administrator, Albert Abad!	Add user	Delete user
Albert Abad	ID		
Bernat Bartomeu	1		
Elisabet Esteve	Name		
Francesc Fité	Albert		
Glòria Guasch	Surname		
Helena Herrera	Abad		
Iván Ibáñez	Email		
Jordi Jin	a@a.com		
Kàtia Klein	Role		
Lídia Llupià	Administrator		
Maria Moyà			
Narcís Nadal			
Oriol Ortiz			
Pau Planes			
Quim Quiròs			

Admin user subpage of the application

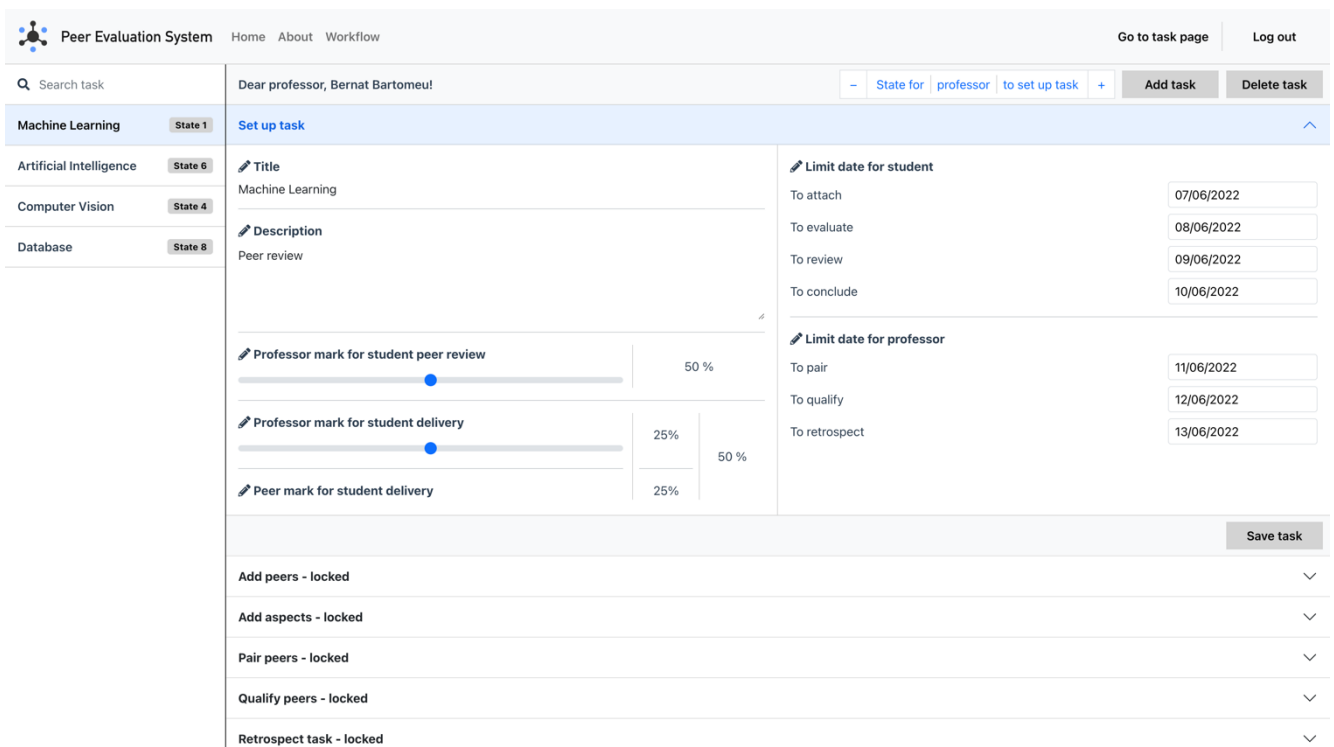
### 13.3.4. Professor module

#### - Professor task page (default)



Professor task page of the application

#### - Professor set up task subpage



Professor set up task subpage of the application



## - Professor add peers subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear professor, Bernat Bartomeu! - State for professor to add peers + Add task Delete task

Machine Learning State 2 Set up task ▼

Artificial Intelligence State 6 Add peers ▲

ID	Name	Surname	Role	Action
2	Bernat	Bartomeu	Professor	
1000	Elisabet	Esteve	Student	Delete
1001	Francesc	Fité	Student	Delete
1002	Glòria	Guasch	Student	Delete
1003	Helena	Herrera	Student	Delete
1004	Iván	Ibáñez	Student	Delete
1005	Jordi	Jin	Student	Delete
1006	Kàtia	Klein	Student	Delete
1007	Lídia	Llupià	Student	Delete
1008	María	Moyà	Student	Delete

Add user

Add aspects - locked ▼

Pair peers - locked ▼

Qualify peers - locked ▼

Retrospect task - locked ▼

Professor add peers subpage of the application

## - Professor add aspects subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear professor, Bernat Bartomeu! - State for professor to add aspects + Add task Delete task

Machine Learning State 4 Set up task ▼

Artificial Intelligence State 6 Add peers ▼

Computer Vision State 4 Add aspects ▲

Database State 8

Introduction Review introduction ✎ Title  
Introduction

Body paragraphs Review body paragraphs ✎ Description  
Review introduction

Conclusion Review conclusion ✎ Weight  
10 / 10

Add aspect Save aspect Delete aspect

Pair peers - locked ▼

Qualify peers - locked ▼

Retrospect task - locked ▼

Professor add aspects subpage of the application

## - Professor pair peers subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear professor, Bernat Bartomeu! - State for professor to pair peers + Add task Delete task

- Machine Learning State 5 Set up task ▼
- Artificial Intelligence State 6 Add peers ▼
- Computer Vision State 4 Add aspects ▼
- Database State 8 **Pair peers** ▲

User to be evaluated	Action
Elisabet Esteve <span style="float: right;">NIUB 1000</span>	
Francesc Fité <span style="float: right;">NIUB 1001</span>	1002 <span style="float: right;">Delete</span>
Glòria Guasch <span style="float: right;">NIUB 1002</span>	1003 <span style="float: right;">Delete</span>
Helena Herrera <span style="float: right;">NIUB 1003</span>	1008 <span style="float: right;">Delete</span>
Iván Ibáñez <span style="float: right;">NIUB 1004</span>	1009 <span style="float: right;">Delete</span>
Jordi Jin <span style="float: right;">NIUB 1005</span>	1011 <span style="float: right;">Delete</span>
Kàtia Klein <span style="float: right;">NIUB 1006</span>	
Lidia Llupià <span style="float: right;">NIUB 1007</span>	

Add pairing Compute automatic

Qualify peers - locked ▼

Retrospect task - locked ▼

Professor pair peers subpage of the application

## - Professor qualify peers subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear professor, Bernat Bartomeu! - State for student to review evaluations + Add task Delete task

- Machine Learning State 8 Set up task ▼
- Artificial Intelligence State 6 Add peers ▼
- Computer Vision State 4 Add aspects ▼
- Database State 8 **Qualify peers** ▲

User to be qualified	Aspect to check (1 / 3)	Weight
Elisabet Esteve <span style="float: right;">NIUB 1000</span>	Title	10 / 30
Francesc Fité <span style="float: right;">NIUB 1001</span>	Introduction	
Glòria Guasch <span style="float: right;">NIUB 1002</span>	Description	
Helena Herrera <span style="float: right;">NIUB 1003</span>	Review introduction	
Iván Ibáñez <span style="float: right;">NIUB 1004</span>	Commentary	Score: 10 / 10
Jordi Jin <span style="float: right;">NIUB 1005</span>	Nice introduction!	
Kàtia Klein <span style="float: right;">NIUB 1006</span>	Aspect to check (2 / 3)	
Lidia Llupià <span style="float: right;">NIUB 1007</span>	Title	Weight: 10 / 30
María Moyà <span style="float: right;">NIUB 1008</span>	Body paragraphs	

« Peer 1002 (download file) » Add qualification

Retrospect task - locked ▼

Professor qualify peers subpage of the application

- **Professor retrospect task subpage**

Peer Evaluation System [Home](#) [About](#) [Workflow](#) [Go to task page](#) [Log out](#)

Q Search task Dear professor, Bernat Bartomeu! [-](#) [State for](#) [professor](#) [to retrospect task](#) [+](#) [Add task](#) [Delete task](#)

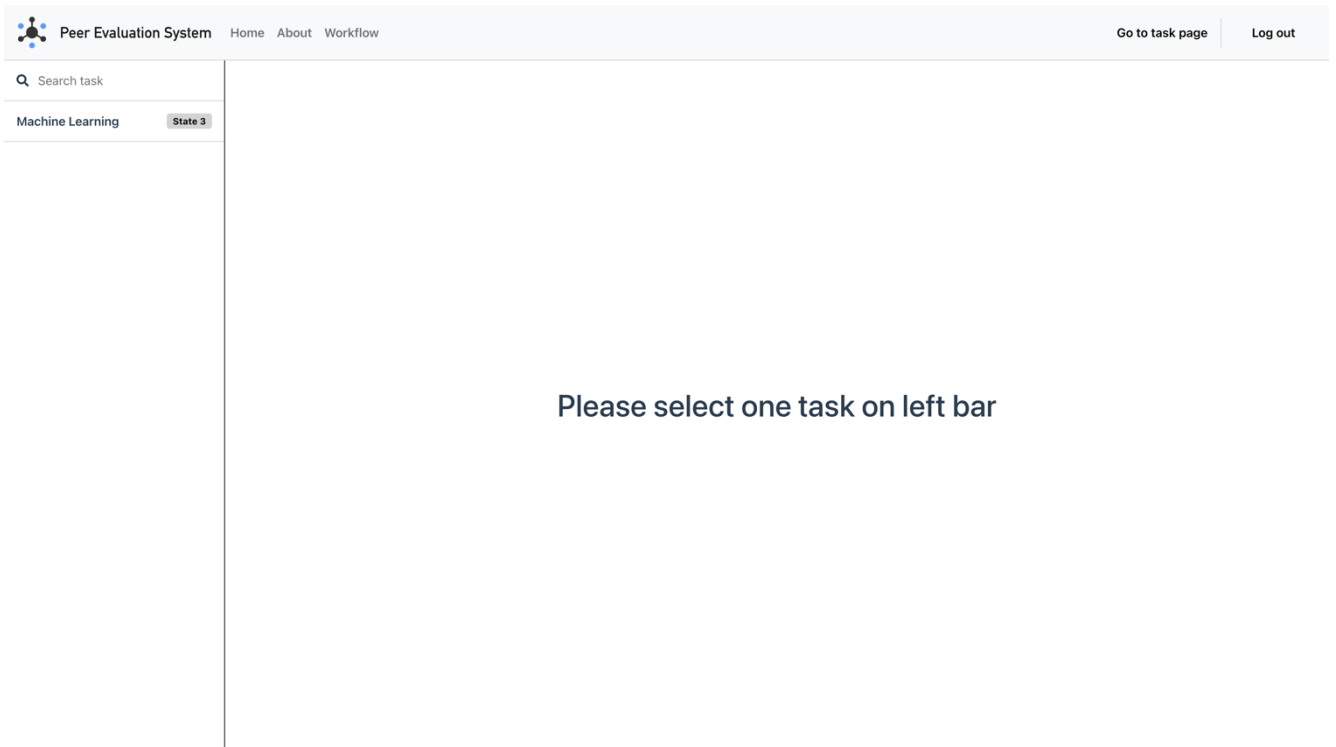
Machine Learning <b>State 10</b>	Set up task	▼
Artificial Intelligence <b>State 6</b>	Add peers	▼
Computer Vision <b>State 4</b>	Add aspects	▼
Database <b>State 8</b>	Pair peers	▼
	Qualify peers	▼
	<b>Retrospect task</b>	▲
Elisabet Esteve <b>NIUB 1000</b>	<b>Conclusion</b> I have enjoyed the peer review!	
Francesc Fité <b>NIUB 1001</b>		
Glòria Guasch <b>NIUB 1002</b>		
Helena Herrera <b>NIUB 1003</b>		
Iván Ibáñez <b>NIUB 1004</b>		
Jordi Jin <b>NIUB 1005</b>		
Kàtia Klein <b>NIUB 1006</b>		
Lidia Llupià <b>NIUB 1007</b>		
Maria Moyà <b>NIUB 1008</b>		

[Add qualification](#) [Download csv](#)

Professor retrospect task subpage of the application

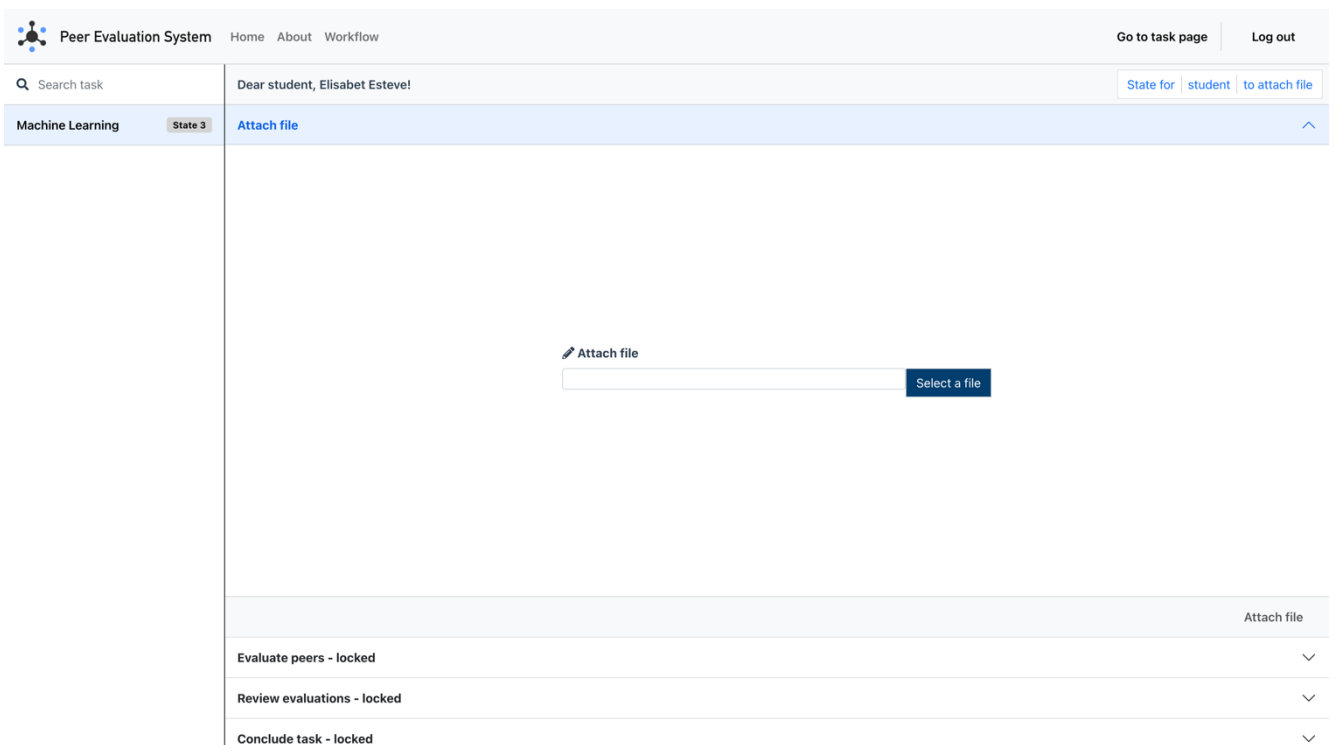
### 13.3.5. Student module

- **Student task page (default)**



Student task page of the application

- **Student attach file subpage**



Student attach file subpage of the application

- Student evaluate peers subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear student, Elisabet Esteve! State for student to evaluate peers

Machine Learning State 6 Attach file ▼

**Evaluate peers** ▲

Peer	ID	Aspect to check (1/3)
Peer 0	1002	Title
Peer 1	1003	Introduction
Peer 2	1008	Description
Peer 3	1009	Review introduction
Peer 4	1011	Commentary

Nice introduction

Score 10 / 10

Download file Add evaluation

Review evaluations - locked ▼

Conclude task - locked ▼

Student evaluate peers subpage of the application

- Student review evaluations subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear student, Elisabet Esteve! State for student to review evaluations

Machine Learning State 8 Attach file ▼

Evaluate peers ▼

**Review evaluations** ▲

Peer	ID	Aspect (1 / 3)	Weight
Peer 0	1004	Title	10 / 30
Peer 1	1005	Introduction	
Peer 2	1006	Description	
Peer 3	1007	Review introduction	
Peer 4	1010	Commentary	Score: 10 / 10

Nice introduction

« 1 »

Conclude task - locked ▼

Student review evaluations subpage of the application

## - Conclude task subpage

Peer Evaluation System Home About Workflow Go to task page Log out

Q Search task Dear student, Elisabet Esteve! State for student to conclude task

Machine Learning State 9

- Attach file
- Evaluate peers
- Review evaluations
- Conclude task

**Conclusion**  
I have enjoyed the peer review!

Submit

Student conclude task subpage of the application