Ecole Supérieure Privée
d'Ingénierie et de Technologies

E-COLLAB

Project based learning platform

Elaborated By:

Iheb Chakroun

Dorsaf Ferchichi

<#ASHTAG>

Imene Tbini

Khaled Saidi

ESPRIT SCHOOL OF ENGINEERING

IN PARTNERSHIP WITH ALEER

Contents

List of Figures

INTRODUCTION

Nowadays learning takes different shapes and forms. One of these is what we call today "Digital Learning", a digitized approach to learn new skills or explore new subjects through taking online courses, bootcamps, workshops and much more. What we are interested in is what instantiates itself as E-learning platforms with Project-Based Learning as a teaching method.

The field of e-learning has gained increasing popularity and for the right reasons. Over the last few years, intuitive learning software has seen leaps in offering a complete learning experience that is fun, interactive, and ultimately engaging. So, what exactly is an E-Learning Platform? An online learning platform is an integrated set of interactive online services that provide trainers, learners, and others involved in education with information, tools and resources to support and enhance educational delivery and management.

Research on PBL supports an increase in learner engagement and achievement and helps learners develop the 21st-century skills they need to succeed in their future careers. So, what exactly is Project-Based Learning?

Learners work on a project over an extended period of time – from a week up to a semester – that engages them in solving a real-world problem or answering a complex question. They demonstrate their knowledge and skills by creating a public product or presentation for a real audience.

As a result, learners develop deep content knowledge as well as critical thinking, collaboration, creativity, and communication skills. Project Based Learning unleashes a contagious, creative energy among learners and trainers.

CHAPTER 1: PHASE I

The preliminary study is an important step for the realization of an application. Indeed, it allows analyzing, evaluating and criticizing the usual operation, while developing the list of possible solutions.

I. The Purpose of the Project

The old-school model of passively learning facts and reciting them out of context is no longer sufficient to prepare students to survive in today's world. Solving highly complex problems requires that students have both fundamental skills and 21st century skills (teamwork, problem solving, research gathering, time management, information synthesizing, utilizing high tech tools). With this combination of skills, students become directors and managers of their learning process, guided and mentored by a skilled teacher.

By bringing real-life context and technology to the curriculum through a PBL approach, the need for a PBL platform stands out. This platform makes it easier for a project participant to communicate, exchange ideas, and follow the project progress.

This system is needed in order to:

- Ensure communication and collaborative work through a shared workspace;
- Facilitate assessment and evaluations management (by group, individual);
- Facilitate documentation sharing and artefacts management
- Facilitate event management (e.g. meetings, presentations, project ceremonies)
- Provide calendar for events management, evaluations and coaching planning
- Facilitate sharing, communication and remote collaboration.

II. The Scope of the Work

A. The Current Situation

In this part we will present some examples of collaborative project-based learning platform such as OPENedX and Moodle.

OPENEDX

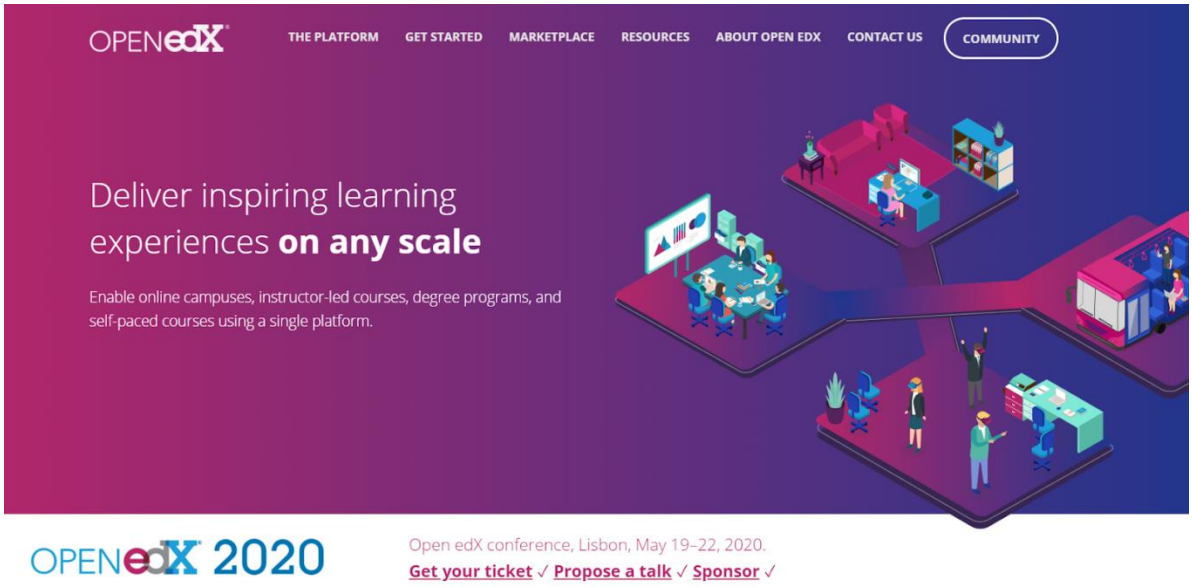


Figure 1: Screenshot "www.openedX.com"

OpenedX is the online learning destination founded by Harvard and MIT in 2012

- Increase access to high-quality education for everyone, everywhere
- Enhance teaching and learning on campus and online
- Advance teaching and learning through research
- Provides access to coursecontent and supporting infrastructure (schedules, discussion boards, collaboration tools, student administration, certificate generation, messaging, and more).
- Equally supports online courses, online campuses, and online degree programs.
- Customizable learner experience.

MOODLE

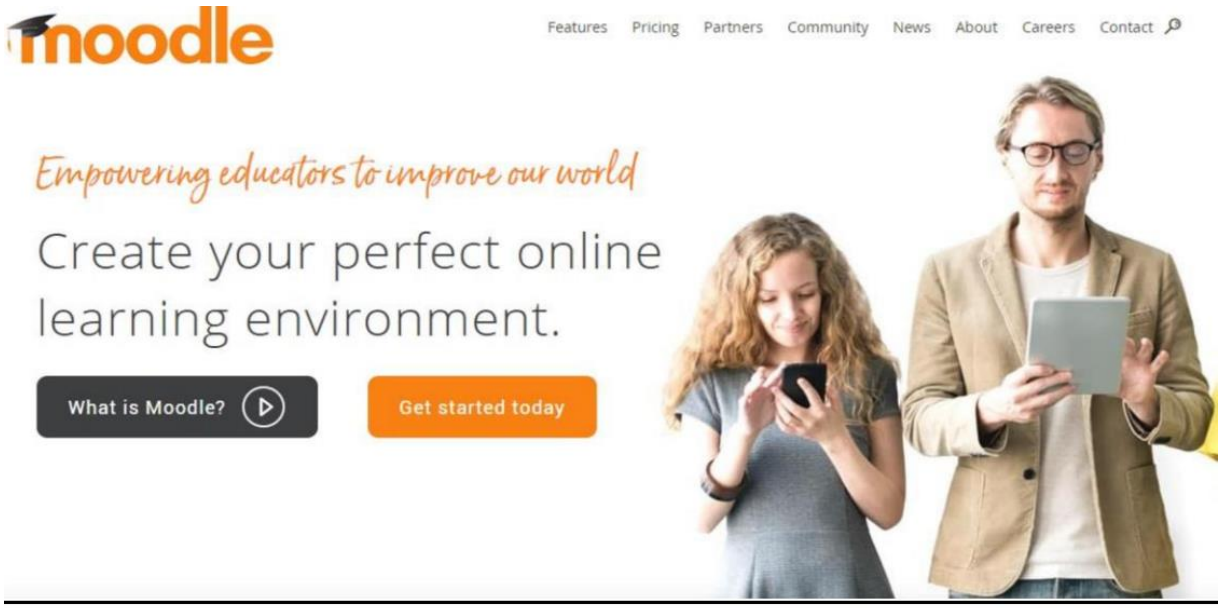


Figure 2: Screenshot "www.moodle.com"

Moodle is an online learning platform that provides services for learners and instructors such as:

- Access to all courses for all learners, everywhere
- Enhance teaching and learning.
- Evaluation and Exam taking system
- Communication via messages

B. Shortage and Solution

After analyzing the current state of the art. We figured out that there is no collaborative workspace to manage projects. Therefore, we propose a collaborative project-based learning platform in which our:

- Group formation based on the member's skills to provide balanced groups.
- Tutors assignment based on the project topic and tutor's profession.
- Avoid communication conflicts by providing a workspace to exchange Information and send files.
- Monitor the progress of each team and each member on the project.

- A calendar for planning all the events, meetings, coaching and evaluations.

C. Innovation

With the increasing use and emergence of advanced technologies, competition has increased. As a result, we need to think outside of the box to grab the user's attention. These are some of our proposed ideas which will help us engage a larger customer base, streamline and automate business processes, improve conversion rate, and boost revenue stream:

- A chatbot for helping learners and answer any of their questions at any time.
- Customizable role assignments to ensure accessibility.

III. Mandated Constraints

A. Financial Feasibility:

Being a web application, the platform will have an associated hosting cost.

At the initial stage the potential market space will be the local universities and higher educational institutes.

B. Technical Feasibility

The platform is a complete web-based application. The main technologies and tools that are associated with are:

- MongoDB
- Express JS
- React JS
- Node JS
- GraphQL

Each of the technologies are freely available and the technical skills required are manageable.

C. Resource Feasibility

Resources that are required for the platform includes:

- Programming device
- Programming tools
- Programming individuals

D. Risk Feasibility

Business impact risks

The platform can be implemented as an individual system. Since it automates some key features associated in the learning process, the users can increase the revenue.

Reasonableness of delivery deadline:

Being a 14 weeks project, the platform will have several deadlines and deliverables that are scheduled successively.

Sophistication of end users:

The platform is designed while maintaining the complexity at a very low level. Usability is highly improved by making GUI easy to use.

Customer related risks:

The initial version of the platform is designed for ESPRIT. Before implementing it in other environments, there will be some basic modifications required.

Technology risks

Are the technologies used new? All the technologies are very well established.

Do the system requirements demand the creation of new algorithms, input or output technology?

The platform will have several algorithms to generate groups, badges…

E. Social/Legal Feasibility

The platform uses freely available development tools, and provides the system as an open source system.

Since this new system reduces the amount of work, it will have a great impact in a university system.

IV. Prototyping:

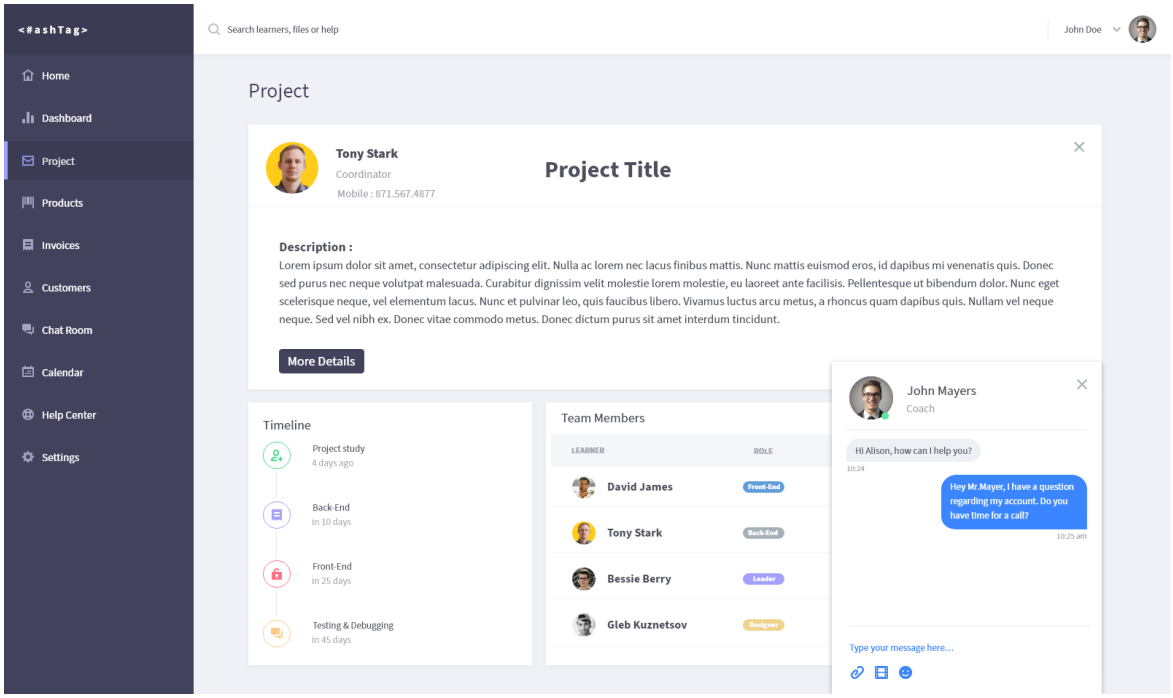


Figure 3: Learner accessing the project he/she was assigned to.

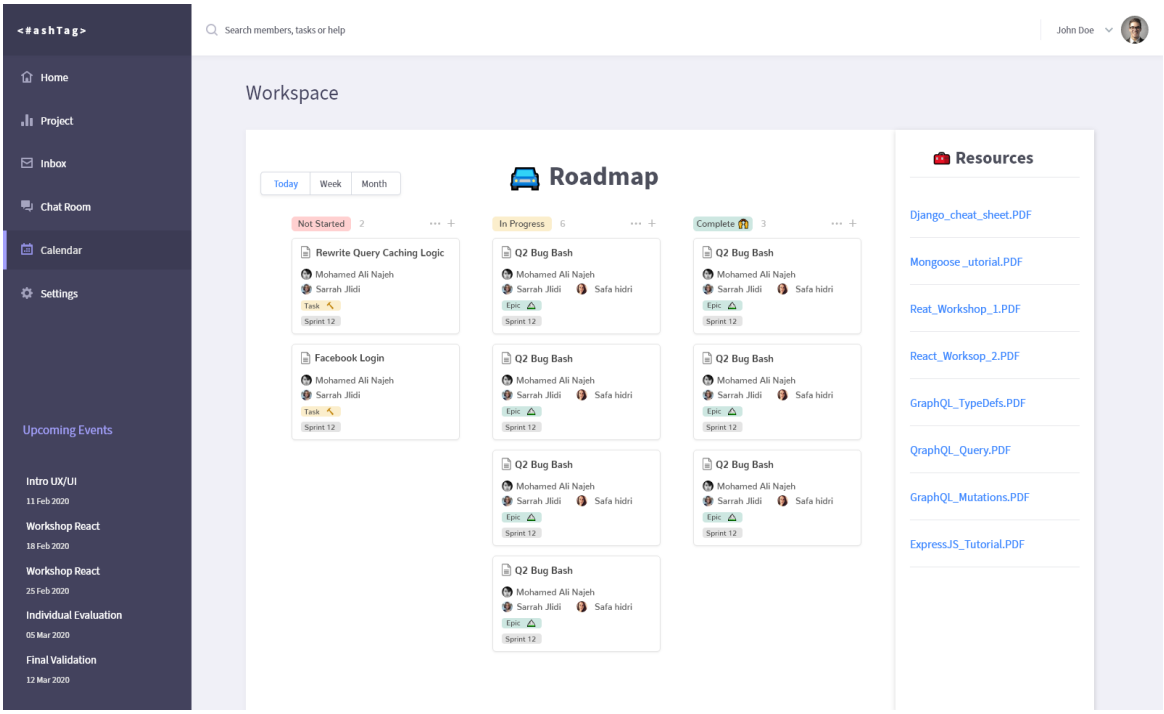


Figure 4: Learner accessing his team workspace.

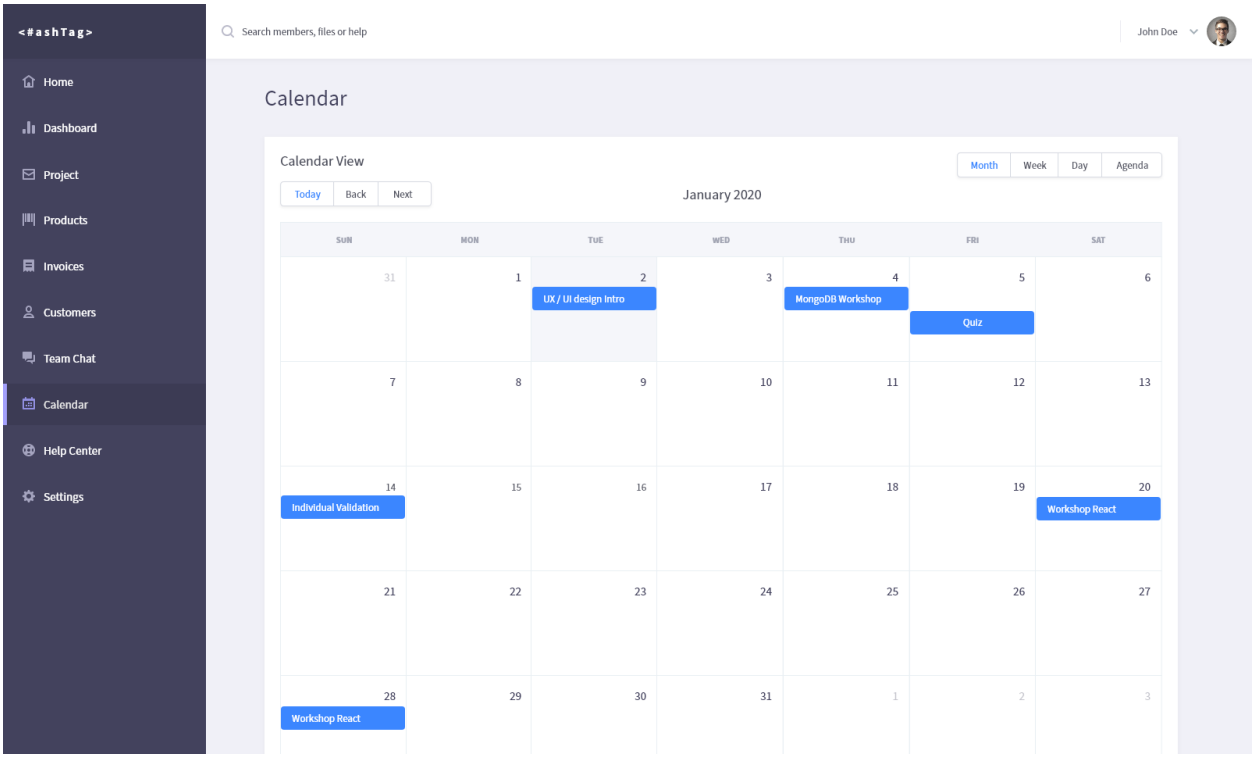


Figure 5: Learner checking his calendar.

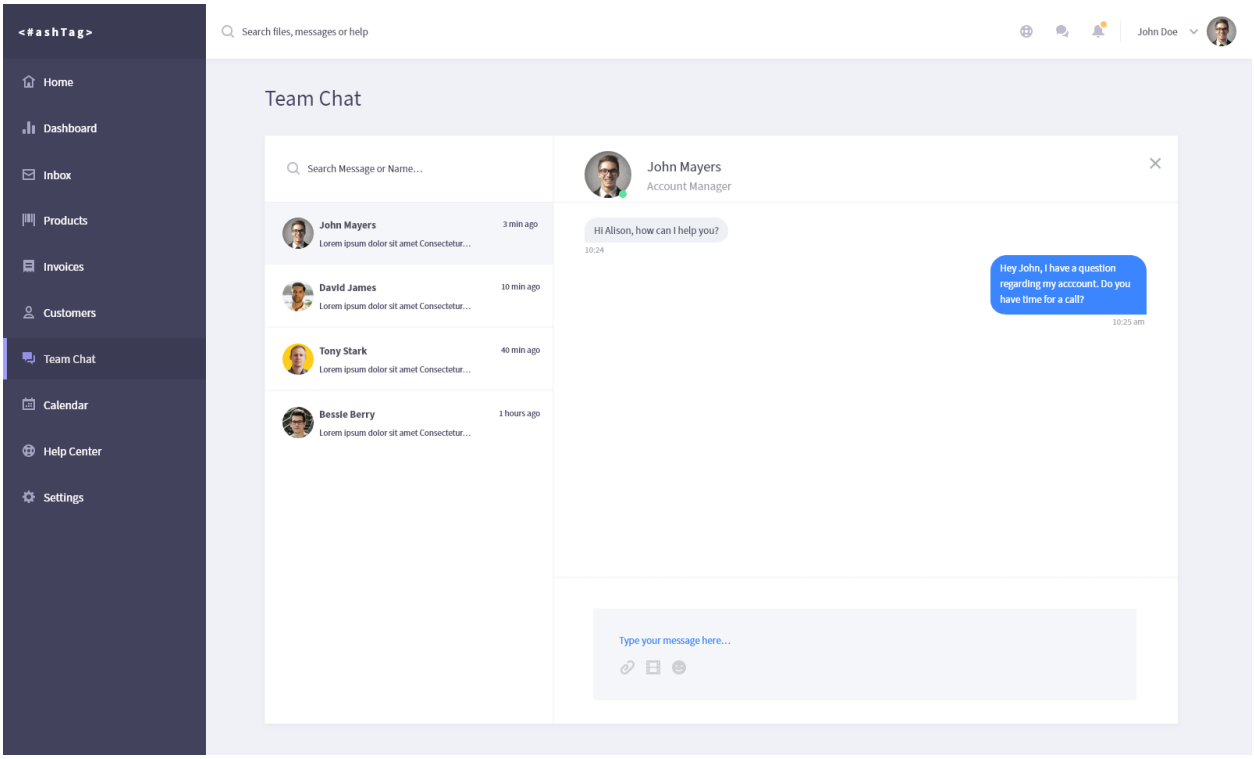


Figure 6: Learner having a team chat.

V. Business Model Canvas

In this section we will establish our Business Model Canvas which is a strategic management and lean startup template for developing new or documenting existing business models. It is going to help us visualize the elements describing our product's value proposition, infrastructure, customers, and finances.

Business Model Canvas

Designed for:	Designed by:	Date:	Version:
ESPRIT	#ASHTAG	2020	1.0

Key Partners	Key Activities	Value Propositions	Customer Relationshi	Customer Segments
Government Universitys High schools Educational organisations	Project based learning Networking Evaluation	Access project planning Manage assignments (teams, projects,tutors) Collaborative workspace Evaluation system	Online access to the services Community Profiling Evaluation by Competancy Quizes	Worldwide Learners Students Reaserchers Educational institutes
	Key Resources Projects /users data Non profit		**Channels** Internet Websites Mobile applications	

Cost Structure	Revenue Streams
Legal secure access Traffic acquisition It systems	Premium access

Figure 7: Business Model Canvas

CHAPTER 2: PHASE 2

In this chapter will talk about all the tools and technologies that we used during

I. Functional Requirements:

Admin space management:

Being in control of the platform, the admin can do:

- Access the list of all projects.
- Add a new project.
- Update/Delete a project.
- Access the list of all users.
- Add a new user.
- Update/Delete a user.
- Manage the access rights of all users: Access Rights are the permissions an individual user holds to read, write, modify, delete or otherwise access a file; change configurations or settings.

Coordinator space management:

Being directly related to a project a coordinator can:

- Access project planning.
- Modify project planning.
- Completely manage project subjects.
- Assign teams to subjects.
- Manage teams assigned to this project.
- Assign coaches to teams.
- Specify the work methodology.
- Treat team change requests.
- Assign stakeholders to their proposed subjects.

- Modify the team members.
- Planify validations dates.
- Assign the jury members of a validation.

Coach space management:

After being assigned to a specific team, a coach has the ability to:

- Create an evaluation test/quiz.
- Grade students after coaching session or evaluation.
- Communicate with students.
- Monitor the workspaces of teams related to him/she.

Project Carrier space management:

- After being assigned to his proposed subject, a project carrier has the ability to:
- Schedule a workshop.
- Schedule a meeting.
- Monitor his subject workspaces with limited access.
- Access the project calendar.

Former space management:

After being hired, a former has the ability to:

- Check his schedule on a calendar.
- Provide documentations and artefacts.
- Schedule a conference.
- Access the project calendar.

learner space management:

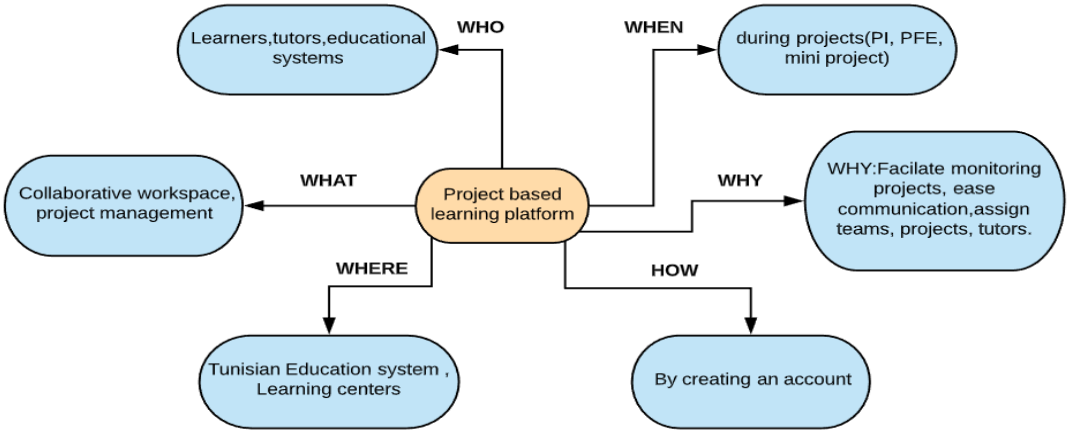
After enrolling in his/her learning organization, a learner has the ability to :

- Consult the subject he/she was assigned to.
- Consult calendar for detailed planning.

- Communicate with his team in a dedicated workspace.
- Update his progress on the tasks he/she was assigned to.
- Check his marks.
- Search for documentation or artefacts.
- Check the list of competencies he acquired.
- Update his profile with limited right accesses.
- Request a team change.
- Exchange files.

II. Non-Functional Requirements:

- Ergonomics: To provide meaningful and relevant experiences to users we used the User experience (UX) design. This involves the design of the entire process of acquiring and integrating the product, including aspects of branding, design, usability and function.



- Reliability: The system must always be able to function properly without the risk of error and failure.
- Security: securing the authorization and information exchange through defining a compact and self-contained way for securely transmitting information between parties as a JSON object which offers the minimum access to the database.

III. Advanced features specification

Chat:

This feature will allow learners to communicate with each other which will improve their ability to keep track of the project's evolution.

It will also allow communication between tutors and their designated group members

Chatbot:

The chatbot will give the learners using our application the help they need by answering any of their questions regarding the progress of the project or technical problems they may face.

Advanced grouping algorithm based on learner's competence:

We will develop an algorithm that will create groups based of the learners' competencies making the groups more balanced

IV. Product Use Cases:

A. Use Case Diagrams:

The global use case diagram will provide us with the global primary form of our project requirements.

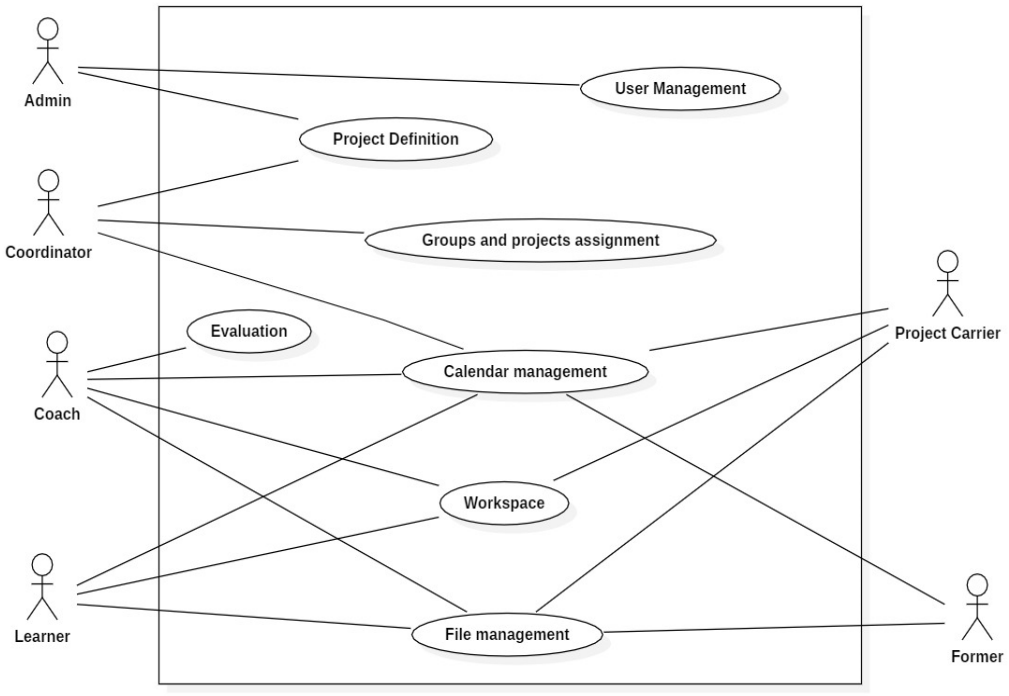


Figure 8: Global Use Case Diagram

The diagram bellow describes the main requirements of the functionality “workspace”.

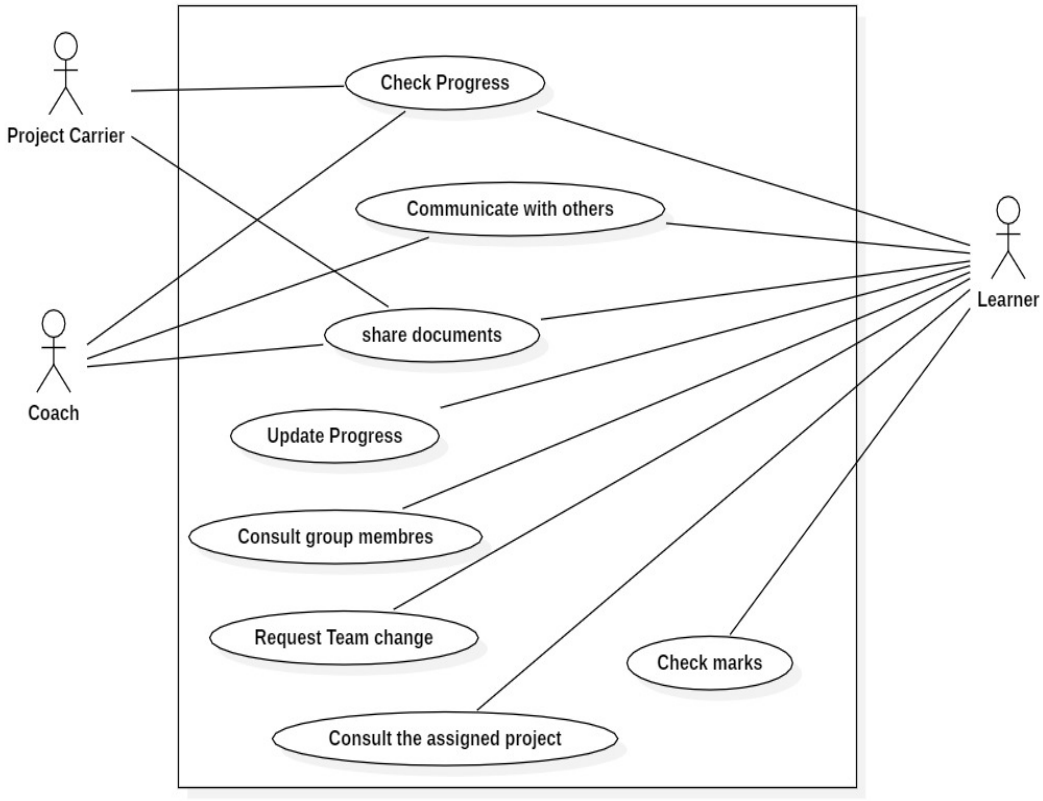


Figure 9: Detailed Use Case Diagram for the use case “Workspace”

B. System Sequence Diagrams:

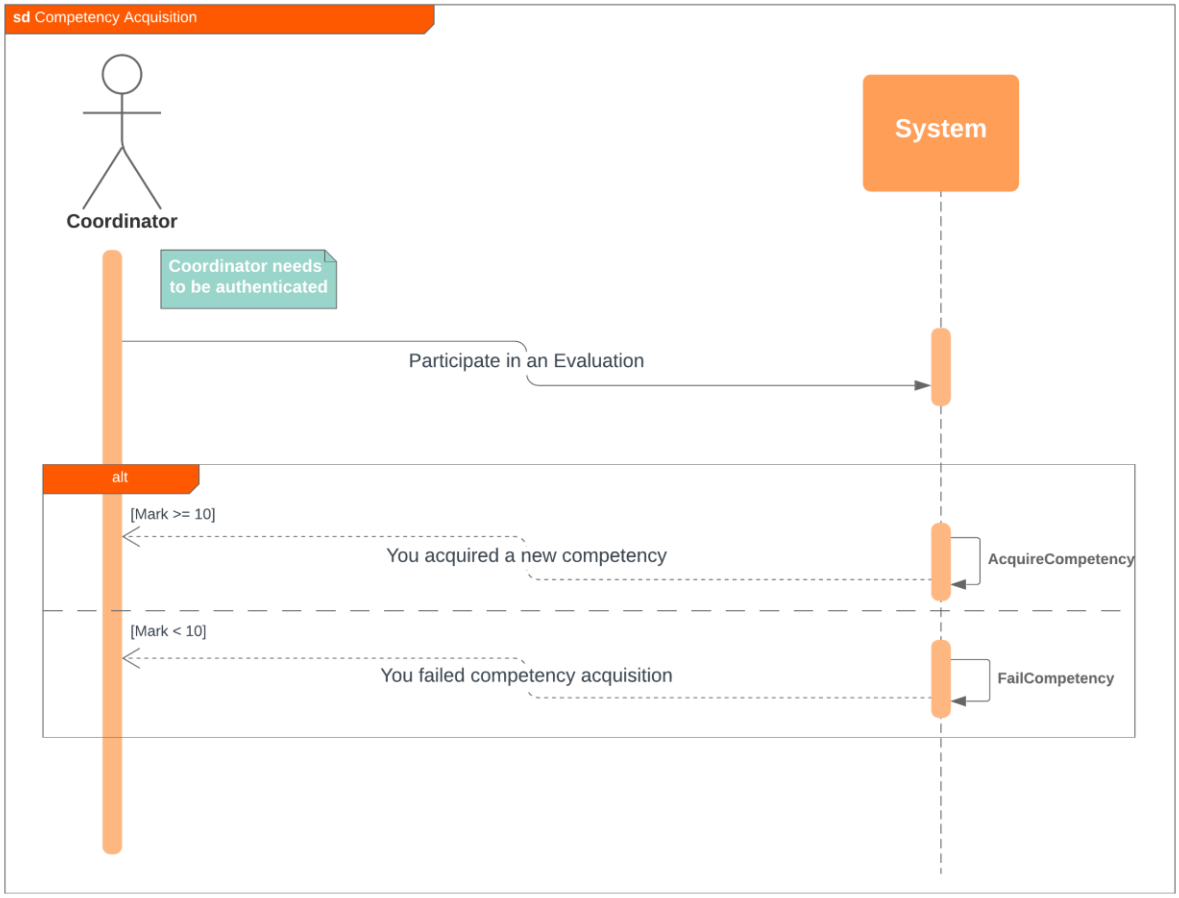


Figure 10: System sequence diagram for the "Competency Acquisition"

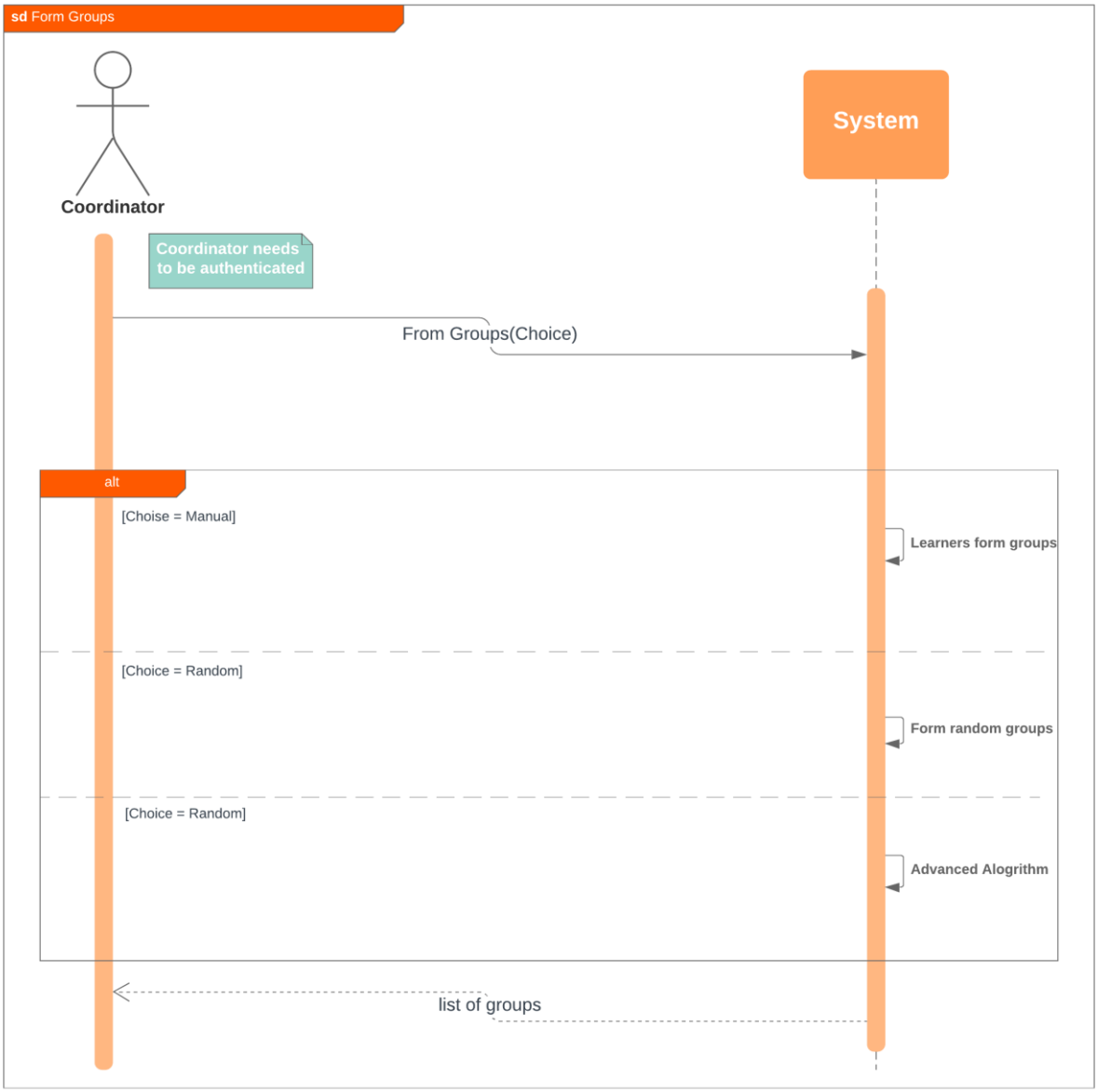


Figure 11: System sequence diagram for the "Form Groups"

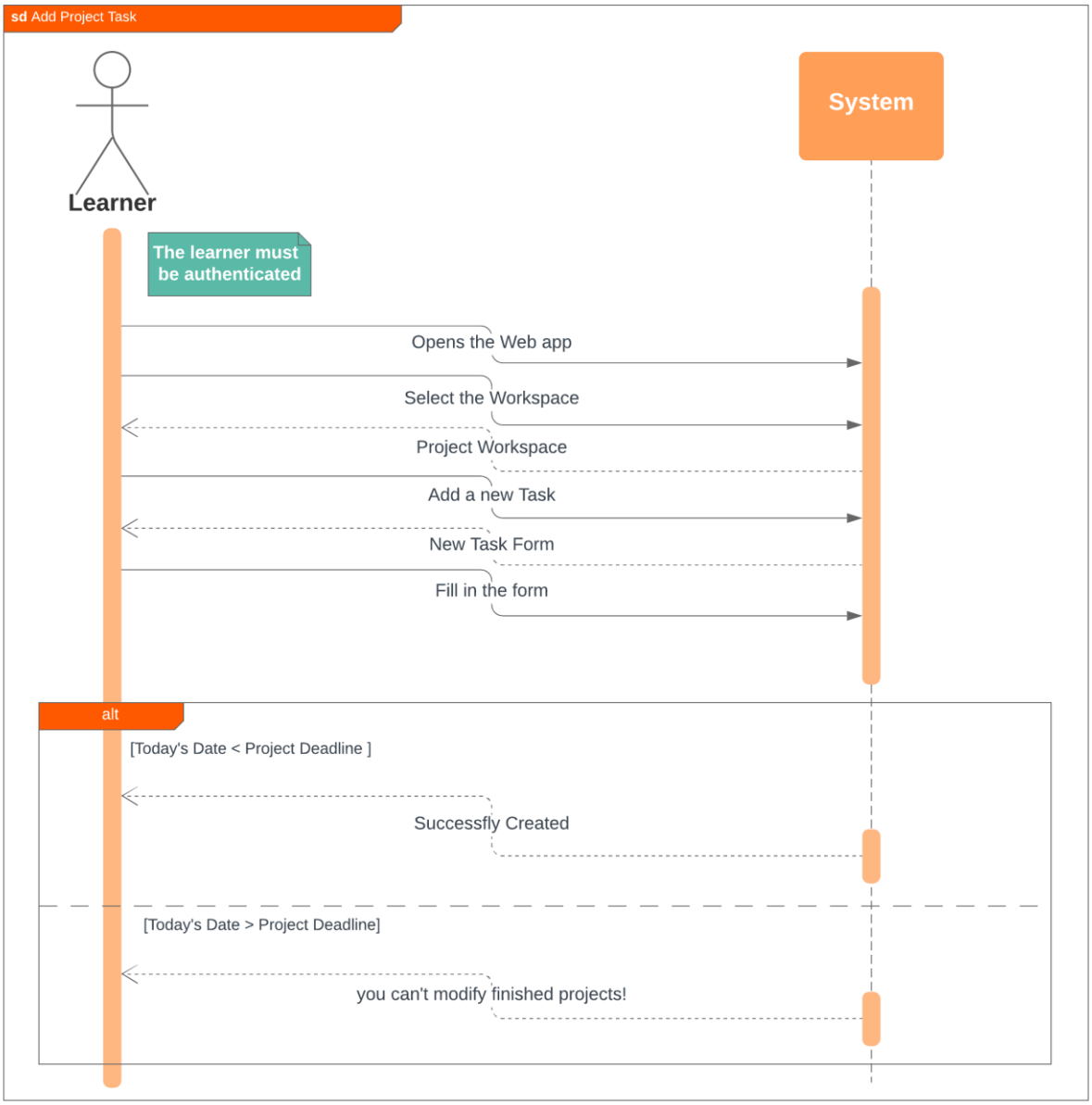


Figure 12: System sequence diagram for the "Add Project Task"

V. Software Architecture

A. Physical Architecture:

The overall architecture of the application will look like this during the development phase of our project:

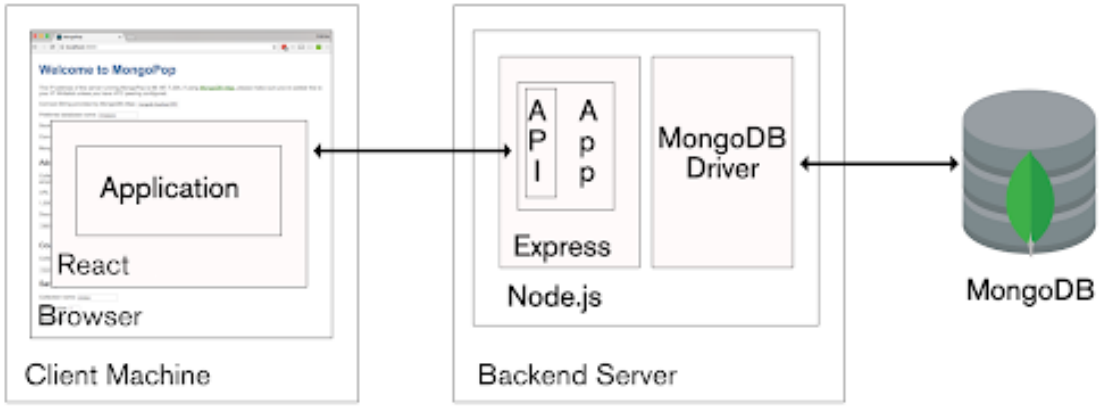


Figure 13: Physical architecture

B. Technical environments:

Since we will be working this project with the MERN stack, you will find below a brief explanation on the choice of technologies:

MongoDB

MongoDB is an object-oriented, simple, dynamic, and scalable NoSQL database. It is based on the NoSQL document store model. The data objects are stored as separate documents inside a collection, instead of storing the data into the columns and rows of a traditional relational database. The motivation of the MongoDB language is to implement a data store that provides high performance, high availability, and automatic scaling. MongoDB is extremely simple to install and implement. MongoDB uses JSON or BSON documents to store data.

Our choice is based on the following list of advantages:

- Flexible: field addition/deletion have less or no impact on the application.
- Heterogeneous Data.
- Dynamic: No rigid schema.
- High performance.

Node.JS

Figure 14: NodeJS logo

Node.js is a JS runtime environment which allows the infrastructure to build and run an application. It's a light, scalable, and cross platform way to execute code. It uses an event-driven I/O model which makes it extremely efficient and makes scalable network application possible.

The most important advantages of Node include:

- It makes building real-time apps (eg. chat or gaming) lightning fast.
- It makes coding in JavaScript for both the client and server side possible.
- The ever-growing NPM Node Package Manager) gives developers multiple tools and modules to use, thus further boosting their productivity.

Express.JS

Figure 15: Express JS logo

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

It's worth noting that we can cut down our programming time in half through ExpressJS. Also, since NodeJS and ExpressJS are written in JavaScript, a very easy language to learn and manipulate, the framework is highly scalable and flexible.

ReactJS

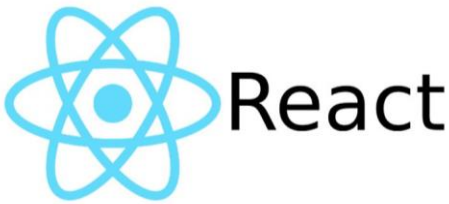


Figure 16: React logo

ReactJS is an open-source JavaScript library which is used for building user interfaces specifically for single page applications. It's used for handling view layer for web and mobile apps. React also allows us to create reusable UI components.

The most important advantages of React include:

- It facilitates the overall process of writing components.
- It boosts productivity and facilitates further maintenance.
- It ensures faster rendering.
- It is backed by a strong community.

GraphQL

Figure 17: GraphQL logo

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

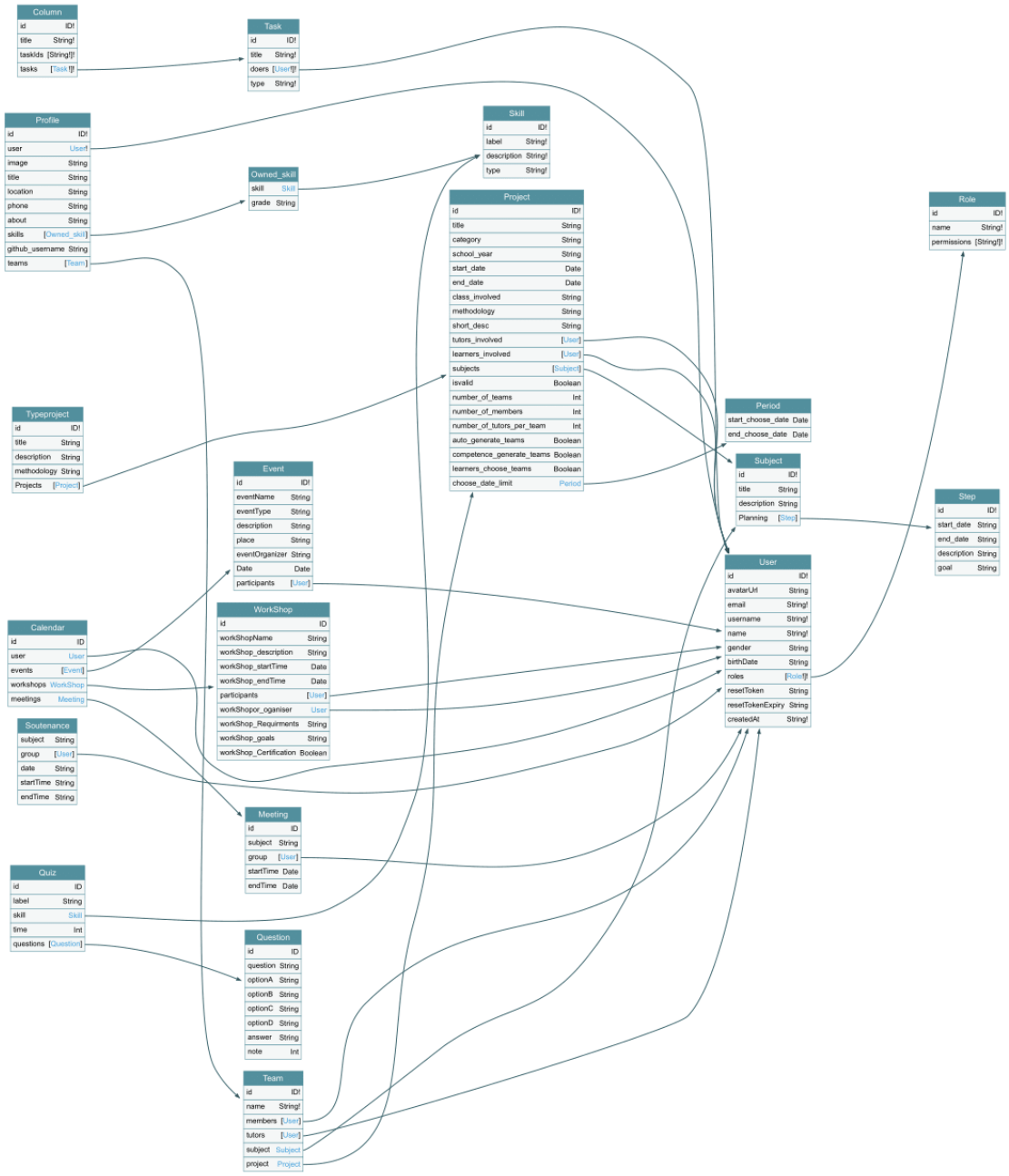
The most important advantages of GraphQL include:

- Fetching data with a single API call.
- No over- and under-fetching problems.
- Tailoring requests to our needs.
- API evolution without versioning.

MERNG Stack

The combination of ExpressJS and ReactJS, a complete JavaScript paradise! We can handle our front-end JavaScript coding with ReactJS library and leave all our backend worries for ExpressJS framework. And since we are already well versed in JavaScript which is a relatively easy language to learn, now we don't have to bother learning many languages for so many purposes!

VI. Data Model



Ps: for more clearer version, check this [pdf](#).

VII. Tools and libraries

Ant Design

Ant Design is a React UI library that has a plethora of easy-to-use components that are useful for building elegant user interfaces. Created by Chinese conglomerate Alibaba. Ant Design is used by several big names: Alibaba, Tencent, Baidu, and more.

GitHub API

The GitHub API is an interface provided by GitHub for developers that want to develop applications targeting GitHub. As example you can build an application with more functionality or better presentation layer on top of the api. The GitHub API makes it easy to connect applications and develop event driven workflows. Pipedream makes it easy to write and run serverless code and event-driven workflows. Simplify Data Delivery. Community Actions. Inline Observability.

Google Drive API

Google Drive API is a tool that allows users create apps leveraging Drive cloud storage. By means of this feature you can develop applications integrating with Google Drive and create powerful functionality in your applications.

Google Calendar API

The Google Calendar API enables developers to add full calendar data and functionality into their app using a REST interface, or through one of the client libraries Google offers for languages like Java, Python, PHP, JavaScript, and more.

Google-auth library

This is Google's officially supported node.js client library for using OAuth 2.0 authorization and authentication with Google APIs. This library provides an implementation of Application Default Credentials for Node.js. The Application Default Credentials provide a simple way to get authorization credentials for use in calling Google APIs.

Apollo

Apollo is a platform for building a data graph, a communication layer that seamlessly connects your application clients (such as React and iOS apps) to your back-end services.

- Apollo Server is an open-source, spec-compliant GraphQL server that's compatible with any GraphQL client. It's the best way to build a production-ready, self-documenting GraphQL API that can use data from any source.
- Apollo Client is a complete state management library for JavaScript apps. Simply write a GraphQL query, and Apollo Client will take care of requesting and caching your data, as well as updating your UI.

Bcrypt.js

Optimized bcrypt in JavaScript with zero dependencies. Compatible to the C++ bcrypt binding on node.js and also working in the browser.

bcrypt is a password-hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher and presented at USENIX in 1999. Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power.

Mongoose.js

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

Joi.js

Hapi Joi is an object schema description language and validator for JavaScript objects. With Hapi Joi, we create blueprints or schemas for JavaScript objects (an object that stores information) to ensure validation of key information. Hapi is a simple to use configuration-centric framework with built-in support for input validation, caching, authentication, and other essential facilities for

building web and services applications.

React-Redux

Redux is a lightweight state management tool for JavaScript applications, released in 2015 and created by Dan Abramov and Andrew Clark. Redux is the most popular state management solution, helping you write apps that behave in the same way, are easy to test, and can run the same in different environments (client, server, native). One of the key ways Redux does this is by making use of a redux store, such that the entire application is handled by one state object.

Chapter 3: Phase III

I. Deployment Diagram

Once our application is finished, for deployment purposes we decided to use: Heroku and MongoDB atlas.

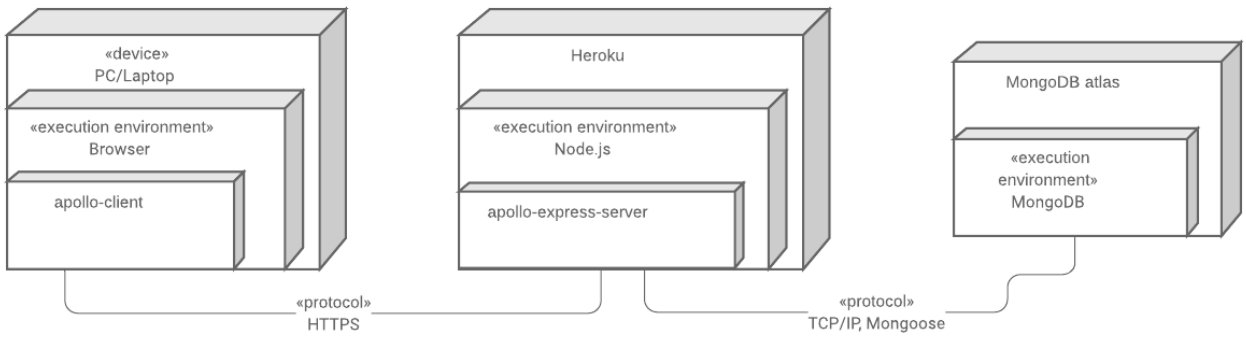


Figure 18: Deployment Diagram

Heroku

Figure 19: Heroku logo

Heroku is a container-based cloud Platform as a Service. Heroku is used to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

MongoDB atlas

Figure 20: MongoDB logo

MongoDB Atlas is the global cloud database service for modern applications. Deploy fully managed MongoDB across AWS, Azure, or GCP. Best-in-class automation and proven practices guarantee availability, scalability, and compliance with the most demanding data security and privacy standards.

II. GraphQL API description

A. Queries

Queries are used to request the data needed from the server. Unlike REST APIs where there's a clearly defined structure of information returned from each endpoint, GraphQL always exposes only one endpoint, allowing the client to decide what data it really needs from a predefined pattern.

Users:

```
me: User
user(id: ID!): User
users: [User!]!
roles: [Role!]!
```

Figure 21: Users Queries

Workspace:

```
column(id: ID!): Column
columns: [Column!]!
task(id: ID!): Task
```

Figure 22: Workspace Queries

Profile:

```
getMyProfile: Profile
getProfile(id: ID!): Profile
```

Figure 23: Profile Queries

Skills:

```
getSkills(): [Skill]
getSkillById(id: ID!): Skill
```

Figure 24: Skills Queries

Projects:

```
getproject(id: String): Project
getprojects(): [Project]
```

Figure 25: Project Queries

```
gettype_project(id: String): Typeproject
gettypes_project(): [Typeproject]
getSteps(): [Step]
```

Figure 26: Project types Queries

Subjects:

```
subject(id: String): Subject
subjects(): [Subject]
```

Figure 27: Subject Queries

Events

```
event(id: String): Event
allEvents(): [Event]
workShop(id: String): WorkShop
allworkShops(): [WorkShop]
calendars(): [Calendar]
soutenance(id: String): Soutenance
allSoutenances(): [Soutenance]
```

Figure 28: Events Queries

```
meeting(id: String): Meeting
allMeetings(): [Meeting]
```

Figure 29: Meeting Queries

Quiz:

```
quiz(id: String): Quiz
allQuizzes(): [Quiz]
question(id: String): Question
allQuestions(): [Question]
```

Figure 30: Quiz Queries

Teams

```
getTeams(): [Team]
getTeamById(id: ID!): Team
getTeamsByUserId(id: ID!): [Team]
```

Figure 31: Teams Queries

B. Mutations

Mutations are the key. In GraphQL, mutations are used to CUD:

- Create new data
- Update existing data
- Delete existing data

Users:

```
signUp(email: String! username: String! name: String! password: String!): User
google(code: String!): User
signIn(email: String!, password: String!): User
signOut(): Boolean
requestReset(email: String!): Boolean
resetPassword(email: String password: String confirmPassword: String
      resetToken: String): User
updateMe(email: String username: String name: String password: String): User
createUser( email: String! username: String! name: String! password: String!
          gender: String! birthDate: String! roles: [String!]!): User
updateUser(_id: String email: String username: String name: String gender: String
              birthDate: String roles: [String]): User
deleteUser(id: String!): User
addRole(name: String!, permissions: [String]): Role
updateRole(id: String!, name: String!): Role
deleteRole(id: String!): Role
```

Figure 32: Users Mutations

Workspace:

```
createColumn(title: String!, taskIds: [String], tasks:
      [String]): Column
updateColumn(_id: String title: String taskIds: [String]
      tasks: [String]): Column
deleteColumn(id: String!): Column
createTask(title: String!, type: String!, doers: [String!]!):
Task
updateTask(_id: String!, title: String, type: String,
      doers: [String]): Task
deleteTask(id: String!): Task
```

Figure 33: Workspace Mutations

Profile:

```
createProfile(image: String title: String location: String
      phone: String about: String github_username: String  user_id: ID!): Profile
updateMyProfile(image: String title: String location: String
      phone: String about: String github_username: String profile_id: String): Profile
```

Figure 34: Profile Mutations

Skills:

```
createSkill(label: String, description: String
  type: String): Skill
updateSkill(id: ID!, label: String,
  description: String type: String): Skill
deleteSkill(id: ID!): Skill
```

Figure 35: Skills Mutations

Projects:

```
addproject(title: String category: String school_year: String start_date: Date end_date: Date
              class_involved: String methodology: String number_of_teams: Int number_of_members: Int
              number_of_tutors_per_team: Int auto_generate_teams: Boolean competence_generate_teams: Boolean
              learners_choose_teams: Boolean Choose_date_limit: periode_input): Project
updateproject(id: String title: String category: String school_year: String start_date: Date end_date: Date
              class_involved: String methodology: String number_of_teams: Int number_of_members: Int
              number_of_tutors_per_team: Int): Project
deleteproject(id: String): Project
add_tutors_to_project(id: String, id_tutor: String): Project
add_learners_to_project(id: String): Project
add_subjects_to_project(id: String title: String description: String): Project
```

Figure 36: Projects Mutations

```
add_phase(id_phase: String): Subject
add_task_to_subject(id: String start_date: String end_date: String description: String
                    goal: String): Subject
add_type_project(title: String description: String methodology: String): Typeproject
update_type_project(id: String title: String description: String methodology: String): Typeproject
delete_type_project(id: String): Typeproject
add_project(id: String title: String category: String school_year: String start_date: String
              end_date: String class_involved: String methodology: String number_of_teams: Int
              number_of_members: Int number_of_tutors_per_team: Int auto_generate_teams: Boolean
              competence_generate_teams: Boolean learners_choose_teams: Boolean Choose_date_limit: periode_input
               learners_involved: [String]tutors_involved: [String]): Typeproject
createtask(start_date: String end_date: String description: String goal: String): Step
deletestep(id: String): Step
```

Figure 37: Projects types Mutations

Subjects:

```
addsubject(title: String, description: String): Subject
updatesubject(id: String, title: String, description: String):
Subjdeletesubject(id: String): Subject
validatesubject(id: String): Subject
addsubjecttoproject(title: String, description: String): Subject
```

Figure 38: Subject Mutations

Events:

```
addEvent(eventName: String eventType: String description: String place: String eventOrganizer: String
         Date: Date): Event
updateEvent(id: String eventName: String eventType: String description: String place: String
            eventOrganizer: String Date: Date): Event
deleteEvent(id: String): Event
participantsAssignment(participant: String, id: ID!): Event
addWorkShop(workShopName: String workShop_description: String workShop_startTime: Date
            workShop_endTime: Date  workShop_Requirments: String workShop_goals: String
            workShop_Certification: Boolean): WorkShop
updateWorkShop(id: String workShopName: String workShop_description: String  workShop_startTime: Date
               workShop_endTime: Date workShop_Requirments: String workShop_goals: String
               workShop_Certification: Boolean): WorkShop
deleteWorkShop(id: String): WorkShop
AssignmentGroupToWorkShops(id_workShop: String, emails: [String]): WorkShop
```

Figure 39: Events Mutations

```
updateCalendar(id: String, year: Int, month: Int, day: Int): Calendar
deleteCalendar(id: String): Calendar
eventAssignmentToCalendaer(id_event: ID!, id_user: ID!): Calendar
eventAssignmentToCalendaerApi(id_event: ID!, id_user: ID!): Event
getUsercalendar(id: String): Calendar
addSoutenance(id: String subject: String date: String startTime: String
              endTime: String): Soutenance
updateSoutenance(id: String subject: String date: String startTime: String
              endTime: String): Soutenance
deleteSoutenance(id: String): Soutenance
```

Figure 40: Calendar Mutations

```
addMeeting(subject: String, startTime: Date,
      endTime: Date): Meeting
updateMeeting(id: String
      subject: String
      startTime: Date
      endTime: Date): Meeting
AssignmentGroupToMetting(id_meeting: String,
      emails: [String]): Meeting
deleteMeeting(id: String): Meeting
```

Figure 41: Meetings Mutations

Quiz:

```
addQuiz(label: String, id_skill: String, time: Int): Quiz
updateQuiz(id: String, label: String): Quiz
AssignmentQuestionsToQuiz(id_quiz: String, id_questions: [String]): Quiz
deleteQuiz(id: String): Quiz
assignmetSkillToUser(id_user: String, id_skill: String, grade: Int): Profile
addQuestion(id: String question: String optionA: String optionB: String optionC: String
            optionD: String note: Int time: Int answer: String): Question
updateQuestion(id: String question: String answer: String optionA: String optionB: String
            optionC: String optionD: String note: Int): Question
deleteQuestion(id: String): Question
```

Figure 42: Quiz Mutations

Teams:

```
createTeam(name: String, members: [String], project_id: ID!): Team
changeName(id: ID!, name: String!): Team
transferMembers(id_team_1: ID! id_member_team_1: ID! id_team_2: ID!
          id_member_team_2: ID!): [Team]
generateRandomTeams(project_id: ID!): [Team]
assignOrChangeSubject(id: ID!, new_subject_id: ID!): Team
assignTutor(id_team: ID!, id_tutor: ID!): Team
changeTutors(id_team: ID!, id_old_tutor: ID!,
          id_new_tutor: ID!): Team
moveLearner(id_member: ID!, id_team_from: ID!,
              id_team_to: ID!): [Team]
addNewMember(id_team: ID!, id_member: ID!): Team
removeMember(id_team: ID!, id_member: ID!): Team
removeTutor(id_team: ID!, id_tutor: ID!): Team
```

Figure 43: Teams Mutations

C. Example:

Fetch all teams: (Query)

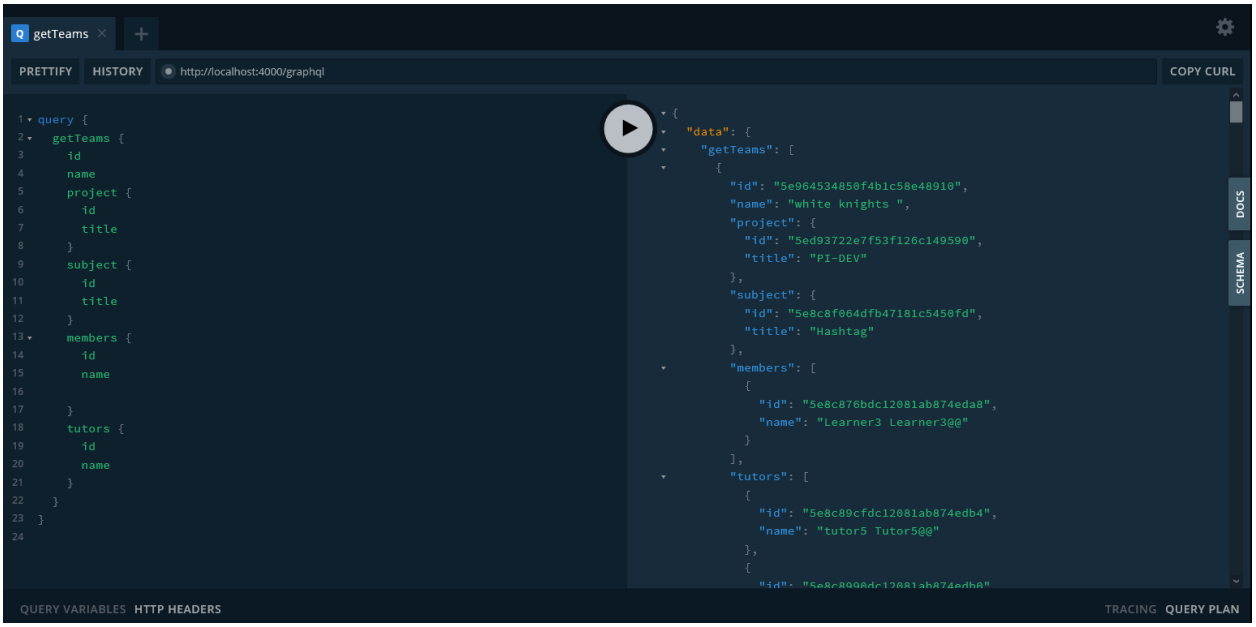


Figure 44: Fetch teams query example

Create new skill: (Mutation)

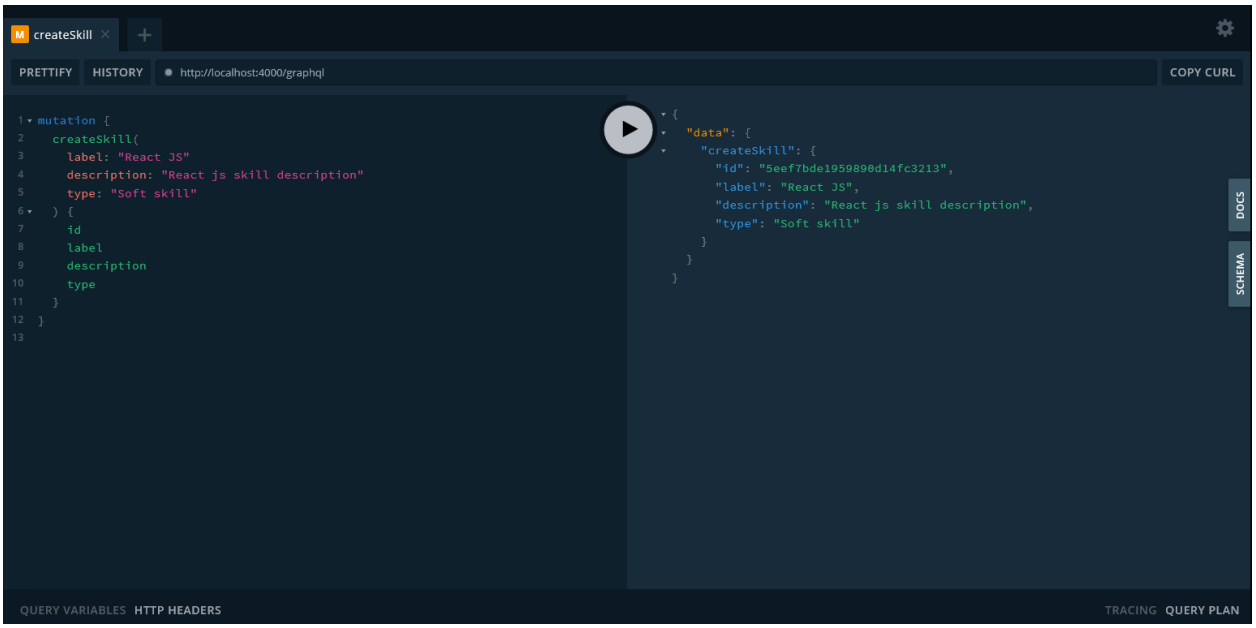


Figure 45: Create new skill mutation example

III. User interfaces

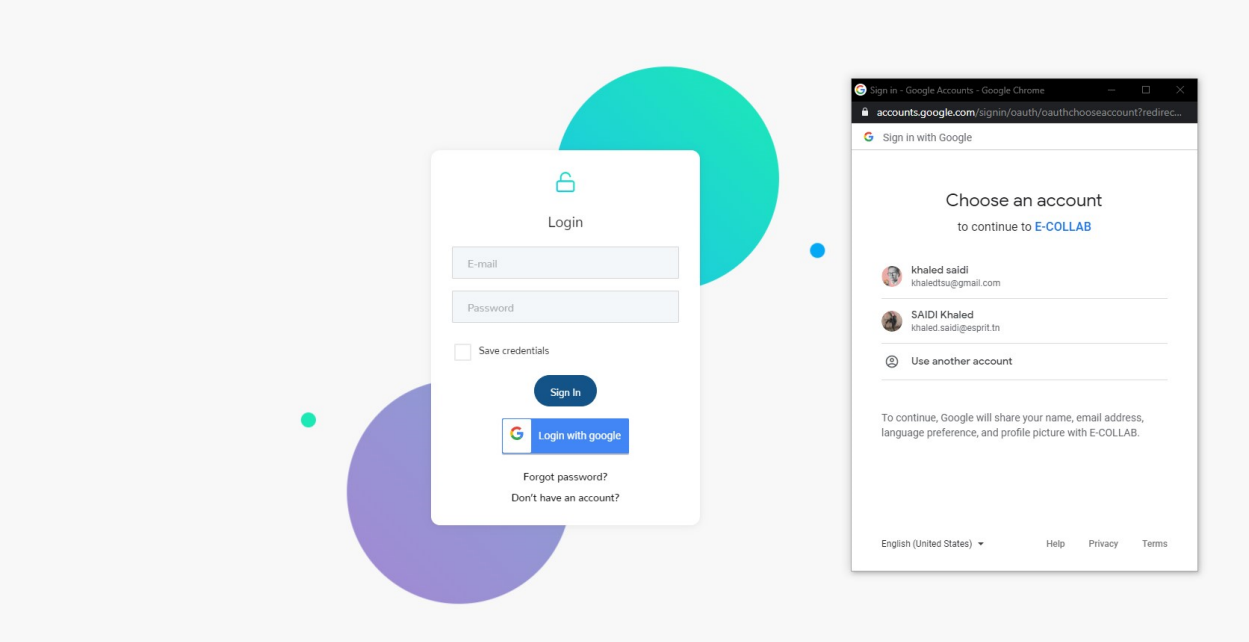


Figure 46: Login interface

We tried to provide various methods to access the E-COLLAB application, with a little bit of constraints, such as:

The classic method: A user can authenticate is via his credentials (After signing up for an account using an e-mail that contains the following domain: esprit.tn.).

A modern approach: Using google authentication to either signing to his account, if he/she doesn't have an account the system will automatically create for him/her.

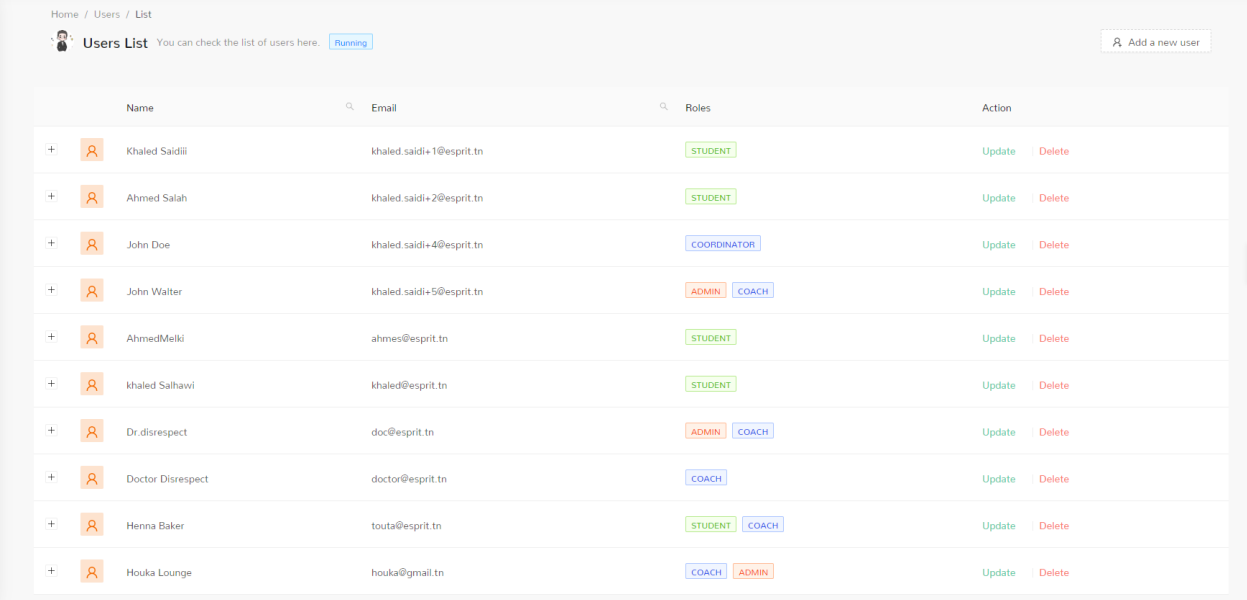


Figure 47: Users List interface

This UI gives the admin a general overview on list of users and their roles. The ability to create more users or simply update an existing one (as example giving him a new role or remove a role). Press the + sign for more details. Look for specific user by his name or e-mail. Finally remove a specific user by pressing the delete button and confirming the action.

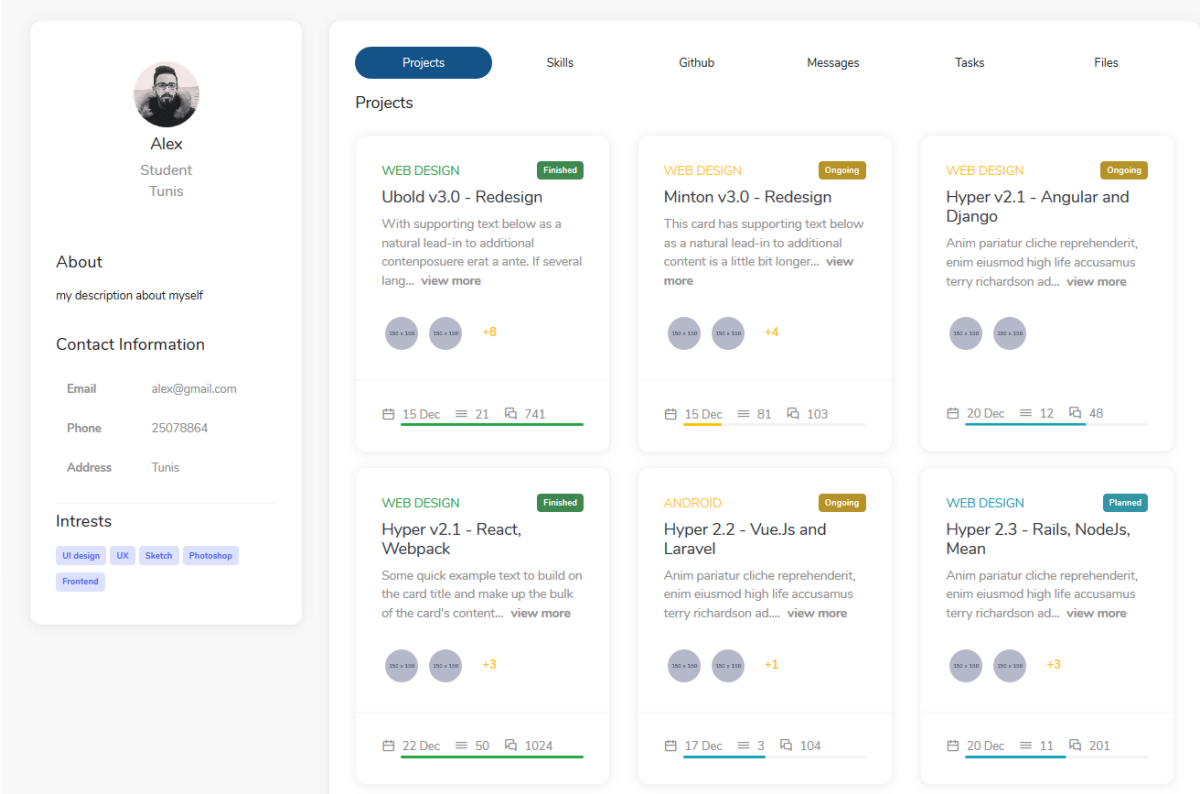


Figure 48: Profile interface

This UI contains all the projects that a student had/has been assigned to. Every project has a status either it's finished, ongoing or still on planning phase. Each project has a brief description alongside the list of members and the list of coaches assigned to it.

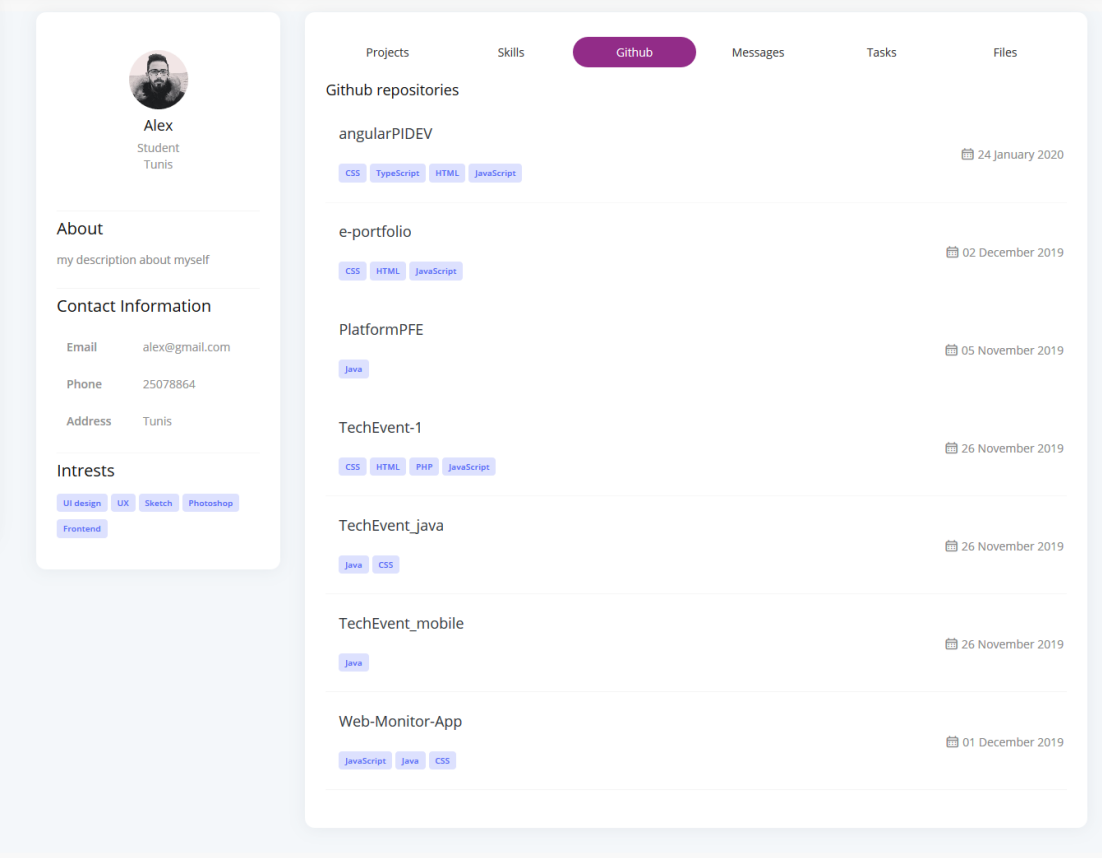


Figure 49: GitHub api profile

Thanks to GitHub API, we were able to fetch the student's repositories by providing his GitHub username. With each repository, we fetched the languages used on that specific repository and the date it was initiated.

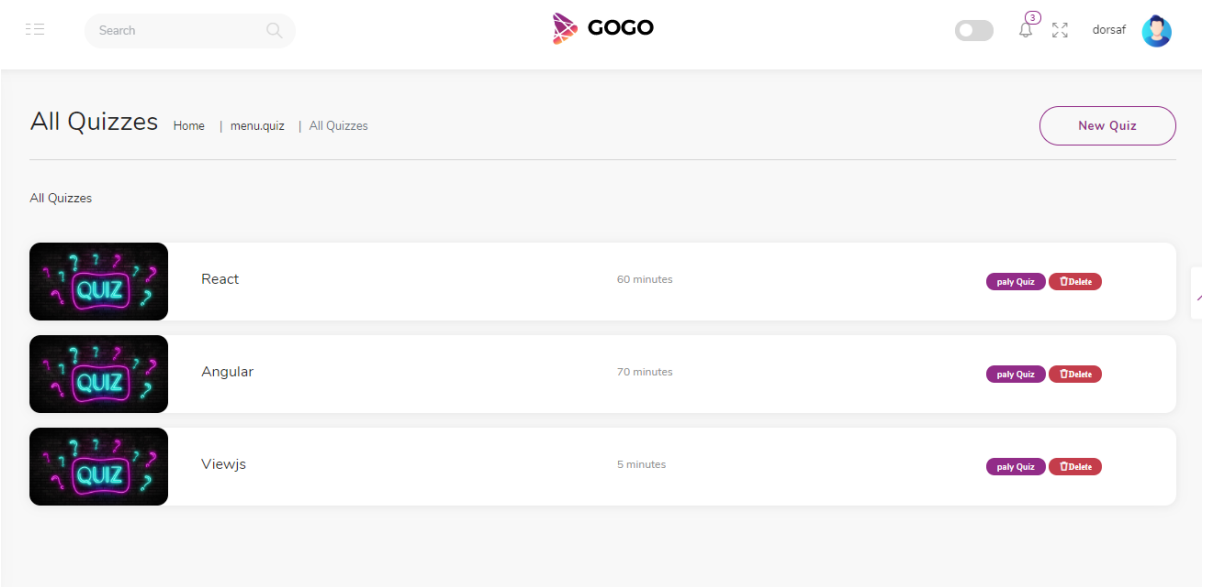


Figure 50: Quizzes Interface

This interface enables us to manage the different quizzes like adding, deleting them. In addition, it shows us Some information about it such as the name, the duration. By clicking the name of the quiz, you will be able to see the different questions with the correct answers, besides that the play quiz button guides us to play the quiz and submit our answers.

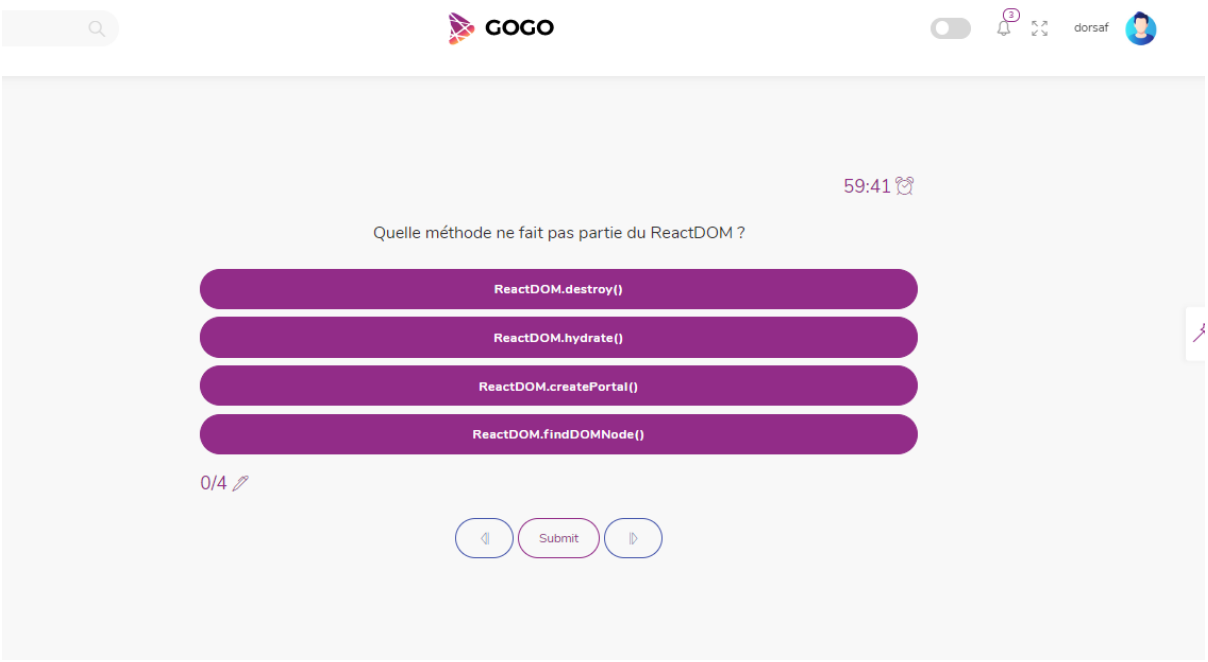


Figure 51: Quiz Play Interface

Through this interface, the student can pass the quiz. By choosing an answer the next question will be displayed, moreover, he can click the previous and the next buttons to get to the next or the previous question, besides that the Small pencil shows how much question he answered, next to it there is a countdown timer to remain him by the time left until the end of the quiz. Finally, he can click the submission button to end the quiz and submit his answers.

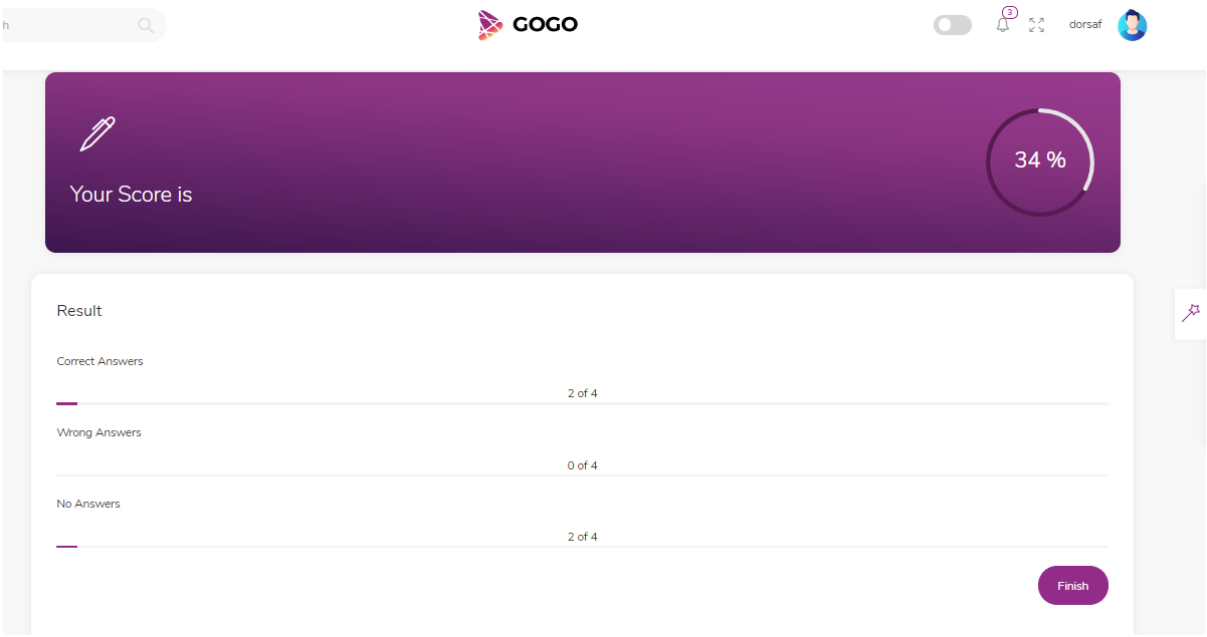


Figure 52: Summary Page Interface

The Summary page shows the result of the quiz, how many correct and wrong answers he submitted and how many questions he left with no answer, besides his score, in case he got more than 50% a new skill will be added to his skill list.

.

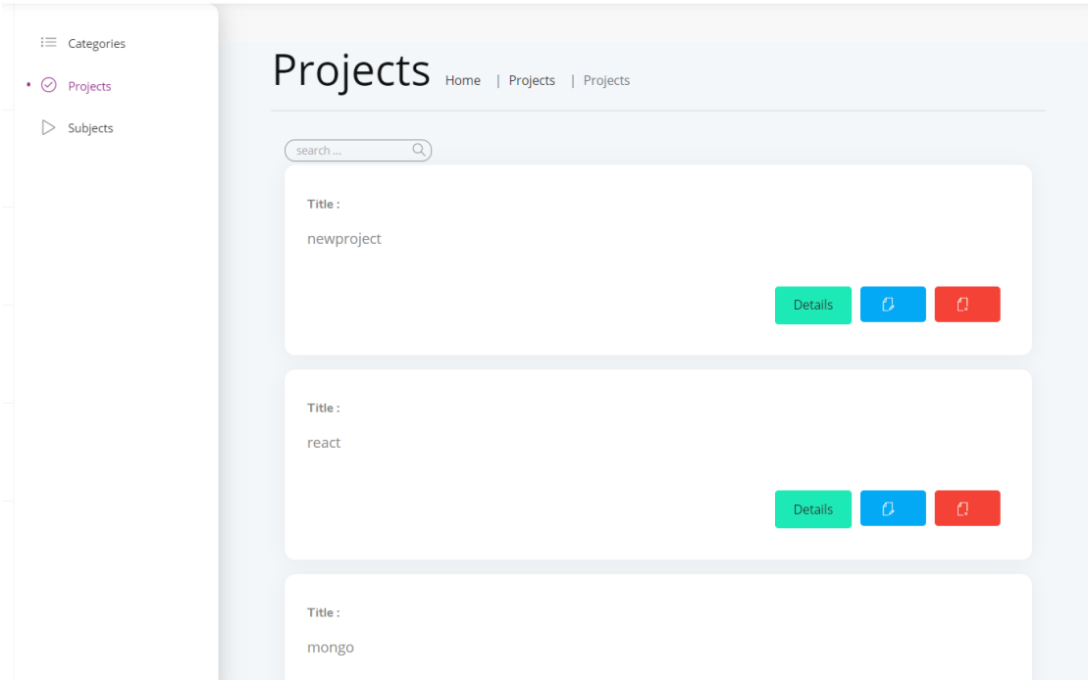


Figure 53: Project interface

This UI above allows the administrator to search update and delete projects and view their details.

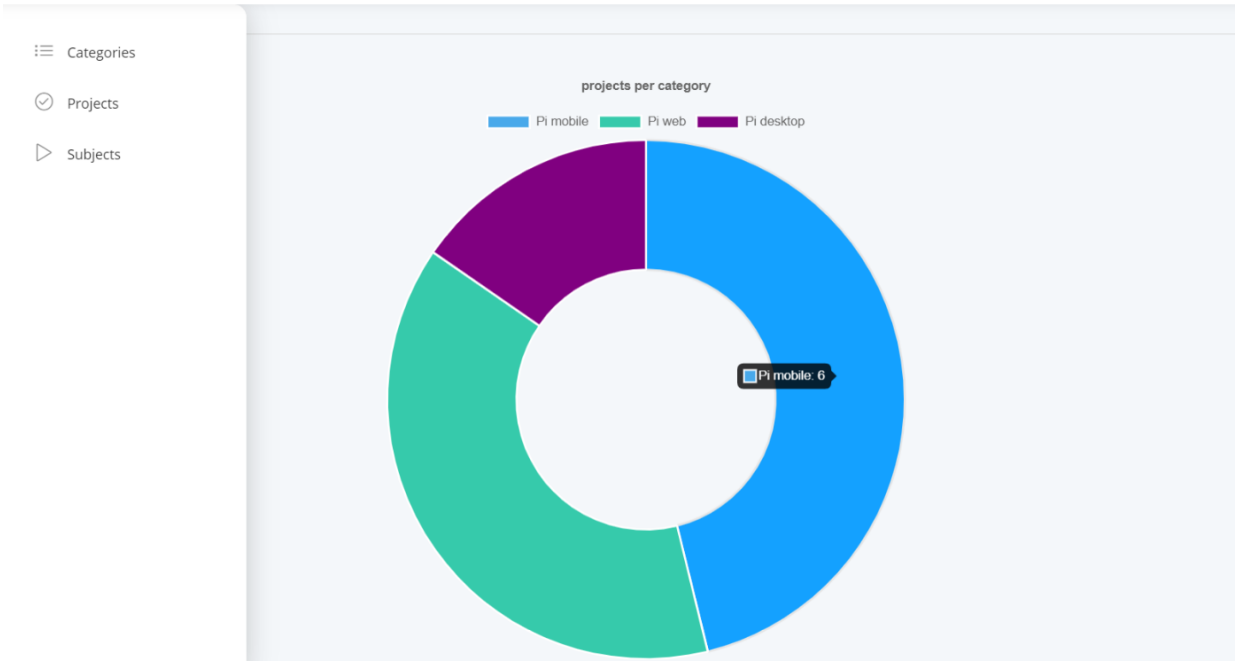


Figure 54: Project stats

This interface above calculates the projects in each category

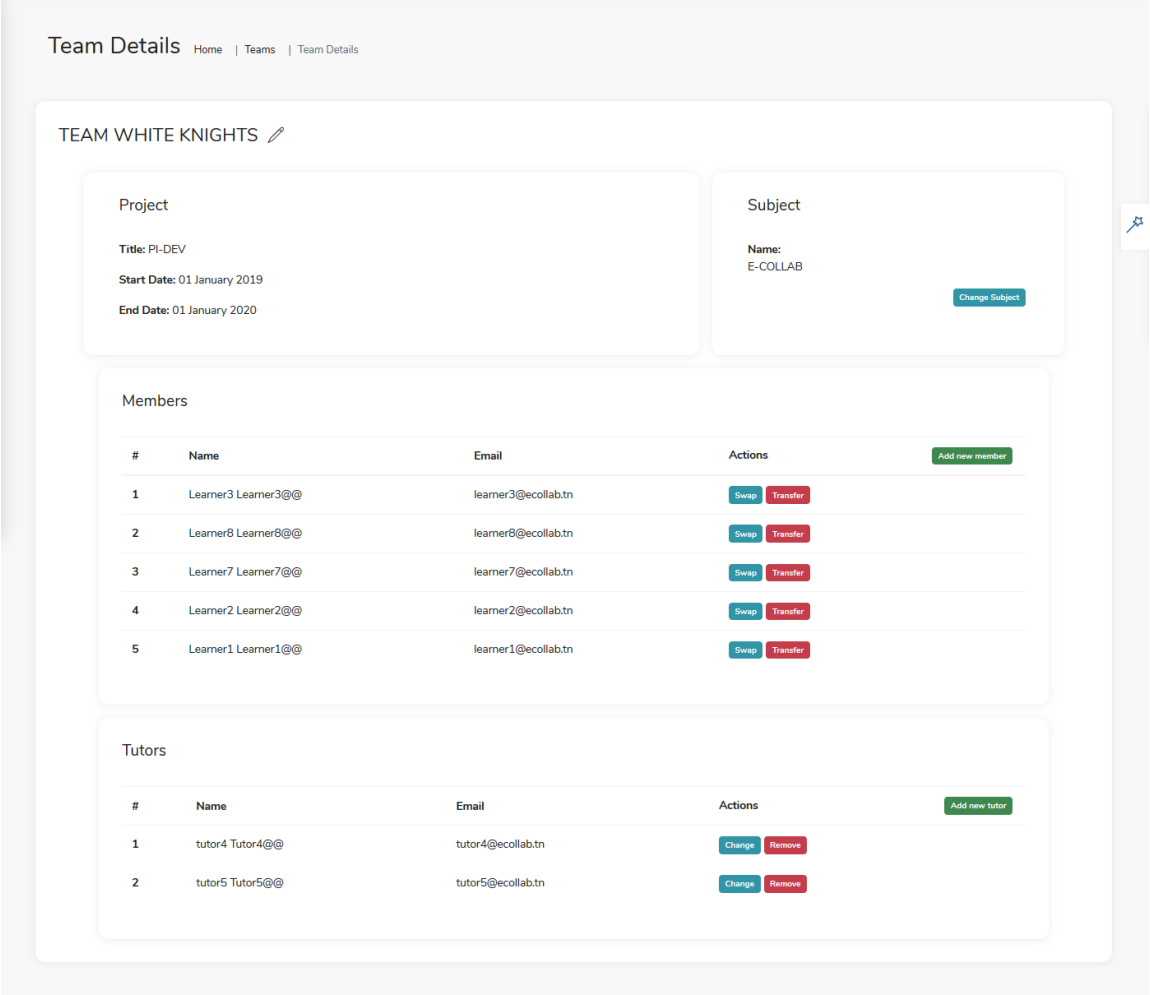


Figure 55: Team details interface

Each team is assigned to project's subject. Each team is formed by a certain number of members and assigned to certain number of coaches. A team could be either manually or generated randomly based on these factors: Number of members per team, Number of teams per project. Finally a coordinator has the ability to re-assign, swap team members and change the subject assigned to the team.

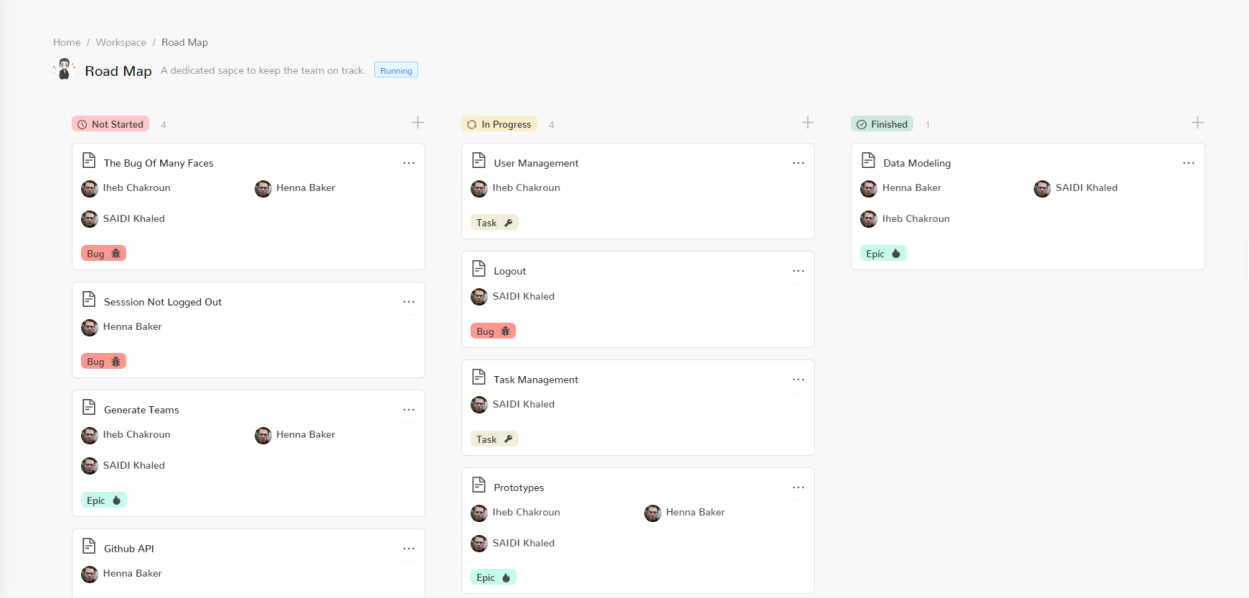


Figure 56: Workspace interface

A dedicated workspace to keep the team members on track and organized. Each member of the team can access it's team workspace and has the ability to add a new task, assign members to a task, change the task type (Task, Epic and Bug) and modify it. He/She could remove a task that is no longer maintained. Finally, he/she could drag and drop tasks from column to another according to their progress on the task add to the ability to reorder tasks on each column.

CONCLUSION

E-learning is constantly growing in popularity because of the convenience of exchanging information and ease of communication especially during the global pandemic, online learning has proven itself to be vital. In such case comes E-COLLAB the perfect application for online collaboration on projects which contains and connects all the advanced features to handle all learners needs and expectations in an online experience

To be able to work on a similar project is such a huge opportunity for us to learn how to interact with the new technologies and the programming languages frequently used nowadays.

REFERENCES

https://open.edx.org/

https://moodle.org/

https://www.definedstem.com/blog/what-is-project-based-learning/

https://www.litmos.com/platform/e-learning-platform-definition

https://www.edutopia.org/project-based-learning

https://www.pblworks.org/what-is-pbl?fbclid=IwAR3Jt4DymBAoWMTquqj33pcfKh4UQAyhv9X0BfQ7OZoMDN84vrKIg0ATbn8