

*Shaked Oz and Fredi Bulshtein*

*Project advisor: Dr. Naomi Unkelos Shpigel*

*Project Code: 24-2-D-28*

*Capstone project phase A*

**CORE SKILLS IN ENGINEERING EDUCATION**

[**Abstract** 3](#_Toc177833095)

[**1. Introduction** 4](#_Toc177833096)

[**2. Literature review** 4](#_Toc177833097)

[2.1 core skills in engineering education 4](#_Toc177833098)

[2.1.1 Challenges in embedding skills in engineering education 5](#_Toc177833099)

[2.2 User-centered design 6](#_Toc177833100)

[2.3 Technology background 7](#_Toc177833101)

[2.3.1 Client side 7](#_Toc177833102)

[2.3.1.1 React.js 7](#_Toc177833103)

[2.3.1.2 Next.js 7](#_Toc177833104)

[2.3.1.3 Angular 8](#_Toc177833105)

[2.3.1.4 React.js vs. Next.js 8](#_Toc177833106)

[2.3.2 Server-side 9](#_Toc177833107)

[2.3.2.1 Node.js 9](#_Toc177833108)

[3.3.2.2 Express.js 9](#_Toc177833109)

[2.3.2.3 SQL 9](#_Toc177833110)

[2.3.2.4 NoSQL 9](#_Toc177833111)

[2.3.2.4.1 MongoDB with mongoose 9](#_Toc177833112)

[2.3.2.4.2 Firebase 10](#_Toc177833113)

[2.3.2.5 SQL vs. NoSQL 10](#_Toc177833114)

[**3. Engineering process** 10](#_Toc177833115)

[3.1 Data collection 10](#_Toc177833116)

[3.2 Workflow 11](#_Toc177833117)

[3.3 Interviews 11](#_Toc177833118)

[3.3.1 Overview 11](#_Toc177833119)

[3.3.2 interviews goals: 12](#_Toc177833120)

[3.3.3 interviews insights 12](#_Toc177833121)

[3.4 Requirements 14](#_Toc177833122)

[3.4.1 Functional Requirements 14](#_Toc177833123)

[3.4.2 Non-Functional Requirements 16](#_Toc177833124)

[3.5 Use case diagram 16](#_Toc177833125)

[3.6 Class diagram 18](#_Toc177833126)

[3.7 Package Diagram 19](#_Toc177833127)

[3.8 Activity diagrams 20](#_Toc177833128)

[3.8.1 Add questionnaire activity diagram 20](#_Toc177833129)

[3.8.2 user uses core skills database 20](#_Toc177833130)

[3.9 Architecture Diagram 21](#_Toc177833131)

[3.10 Challenges 21](#_Toc177833132)

[3.11 Technologies review 22](#_Toc177833133)

[3.11.1 React Flow 22](#_Toc177833134)

[3.11.2 D3.js 22](#_Toc177833136)

[3.11.3 Sigma.js 22](#_Toc177833138)

[3.11.4 Cytoscape.js 22](#_Toc177833140)

[3.11.5 Graph library comparison 22](#_Toc177833142)

[3.12 Interface with the client during development 24](#_Toc177833143)

[3.14 Success Metrics for the Project 25](#_Toc177833144)

[3.15 platform screen sketches 26](#_Toc177833145)

[**4. Testing of the platform:** 28](#_Toc177833146)

[**5. AI usage:** 29](#_Toc177833147)

[**6. References:** 29](#_Toc177833148)

[**7. Appendices** 31](#_Toc177833149)

[7.1 Initial Requirements Form 31](#_Toc177833150)

[7.2 List of the skills and their sub-skills 32](#_Toc177833151)

# 

# 

# **Abstract**

The rapid evolution of the software engineering industry has highlighted the critical importance of core skills, such as communication, teamwork, and problem-solving, alongside technical expertise.

However, lecturers often need help to track, assess, and integrate these skills into the curriculum effectively, making it challenging to ensure students are equipped with the necessary competencies for the modern workforce.

This project focuses on developing a web-based platform that enables lecturers to manage and facilitate the addition of these core skills among software engineering students. The system provides a comprehensive environment where lecturers can design, categorize, and connect courses, activities, skills, and scaffolds to create an effective learning experience. With features like user authentication, skill categorization, activity management, and questionnaire creation, the platform aims to offer a structured and user-friendly interface that is accessible across multiple devices. The system is designed to grow and adapt to the evolving needs of educators and students. Through continuous interaction with various lecturers from different fields, the project has been developed to meet the specific needs of the academic environment, ensuring that it aligns with the project's objectives.

Keywords: Engineering education, core skills, user-centered design

# **1. Introduction**

In today’s competitive job market, the role of core skills in engineering education has become increasingly significant [1]. As employers seek graduates who not only possess strong technical abilities but also excel in interpersonal communication, teamwork, and leadership, the educational system must adapt to these evolving demands [1][2]. Traditional curricula, which primarily focus on technical skills, often fall short in preparing students for the collaborative and dynamic nature of the modern workplace [16][17].

This project addresses this gap by developing a web-based platform that enables lecturers in engineering programs to systematically manage and enhance the core skills development of their students. The platform is designed to be a comprehensive tool that supports the integration of core skills into the existing curriculum through various features, including skill tracking, activity management, and scaffolding.

The development of this platform is guided by extensive interaction with lecturers, ensuring that it meets the specific needs of educators and aligns with their vision [19]. The platform will feature a user-friendly interface that follows the ISO 9241-11 standards, offering an intuitive and efficient user experience, ensuring ease of use across various devices, and making it accessible to lecturers in different teaching environments [20]. Furthermore, the system is built with scalability in mind, allowing it to grow and evolve as the educational landscape changes.

This document provides a detailed overview of the project, including literature reviews on core skills and technical development, interviews, requirements, diagrams of the system, and the challenges encountered during development and testing. Through this project, we aim to contribute to the enhancement of engineering education by providing a tool that bridges the gap between technical proficiency and essential core skills [1][14].

# **2. Literature review**

## 2.1 core skills in engineering education

core skills in engineering education contain a wide range of non-technical abilities crucial for professional success. These include communication skills, enabling skills, and the ability to act professionally. Unlike technical skills, which are often tied to specific tasks or processes, core skills are more general and relate to how individuals interact with others and navigate complex social environments. Communication skills, for instance, involve conveying information clearly and effectively, both verbally and in writing, which is essential for collaboration within engineering teams and engaging with non-technical stakeholders [1].

Enabling skills, such as critical thinking, adaptability, and time management, allow engineers to solve problems creatively, manage their work efficiently, and adjust to changing circumstances in fast-paced environments. Interpersonal behaviours, including teamwork, leadership, and conflict resolution, are vital for working effectively within diverse teams and for leading projects to successful outcomes. Acting professionally, which involves ethical decision-making, responsibility, and a commitment to continuous learning, ensures that engineers uphold the standards of the profession and contribute positively to society [1].

The importance of these skills in engineering education is underscored by the recognition that technical expertise alone is insufficient for addressing the complex challenges of the modern world. As the engineering field becomes increasingly interdisciplinary and global, the ability to work across cultural and disciplinary boundaries, to innovate, and to adapt to new technologies and methodologies is becoming ever more critical. Therefore, integrating core skills into engineering curricula is not just an educational imperative but a necessity for preparing graduates who can thrive in diverse and dynamic professional environments [2].

Core skills in engineering education include communication skills, enabling skills, interpersonal behaviours, and acting professionally. These skills are distinct from technical skills and are essential for effective collaboration and problem-solving in complex social contexts. The growing emphasis on these skills reflects the evolving demands of the engineering profession, where technical expertise alone is no longer sufficient [1].

One significant example of the importance of core skills in engineering education is the recognition that these non-technical skills are critical for addressing societal needs and ensuring that engineering graduates are well-equipped to face future challenges. This insight highlights the necessity for engineering programs to integrate core skills into their syllabus, enabling graduates to engage in diverse and dynamic professional environments effectively [1].

Furthermore, the relationship between student engagement in extracurricular activities and the development of core skills has been explored in various studies. It is emphasized that such activities can significantly enhance communication, teamwork, leadership, and problem-solving skills. These skills are crucial for students' personal and professional growth. Through involvement in diverse activities, students gain practical experiences that complement their academic learning, thereby preparing them for future challenges in the workspace [14].

Moreover, recent research has highlighted that core skills such as adaptability, critical thinking, and emotional intelligence are not only beneficial for individual career success but are also essential for fostering innovation within engineering teams. As engineering projects become increasingly interdisciplinary and global in scope, the ability to communicate and collaborate across cultural and disciplinary boundaries becomes paramount. This shift necessitates a revaluation of traditional engineering curricula, with a stronger focus on embedding core skills training throughout the educational experience [15].

### 2.1.1 Challenges in embedding skills in engineering education

Integrating core skills into engineering curricula presents several significant challenges for educators and institutions. These challenges derive from various factors, including traditional teaching methods, curriculum constraints, and the complex nature of core skills themselves.

One of the primary obstacles is the resistance to change within established engineering programs. Many institutions have long-standing traditions of focusing primarily on technical skills, making it difficult to shift towards a more balanced approach that incorporates core skills. This resistance can come from faculty members who may feel unprepared to teach core skills or who believe that technical knowledge should remain the sole focus of engineering education [16].

Engineering curricula are already densely packed with technical courses, leaving little room for additional content. Integrating core skills without compromising the depth of technical knowledge is a delicate balancing act. Educators face the challenge of finding innovative ways to incorporate core skills training without extending program duration or reducing essential technical content [17].

Unlike technical skills, core skills are often subjective and challenging to measure quantitatively. Developing reliable and valid assessment methods for skills such as communication, teamwork, and leadership poses a significant challenge. Traditional evaluation methods may not be suitable for assessing these competencies, requiring educators to explore alternative assessment strategies [18].

There needs to be more consensus on which core skills are most critical for engineers and how they should be taught. More standardization is needed for institutions to develop comprehensive and consistent approaches to core skills education. The diversity of core skills required across different engineering disciplines further complicates the development of a unified framework [16].

## 2.2 User-centered design

User-Centered Design (UCD) is a design methodology that focuses on creating products that meet the actual needs, expectations, and preferences of users. This approach emphasizes the importance of involving users throughout the development process, from the initial research and ideation stages to prototyping and evaluation. According to Abras, Maloney-Krichmar, and Preece (2004), UCD is inherently iterative, requiring constant feedback loops where users are engaged at multiple stages to ensure the product aligns with their goals [19]. Rather than being driven solely by technological possibilities or business goals, UCD ensures that usability remains a key focus, leading to more intuitive and efficient systems.

UCD includes a range of practices, such as user research (e.g., surveys, interviews, and observational studies), usability testing, and iterative design adjustments based on user feedback. These practices help uncover the user's real needs, often beyond what is immediately apparent. By focusing on these needs, UCD enables designers to create interfaces and interactions that are easy to understand and use, reducing the learning curve and enhancing user satisfaction. For example, user testing often reveals usability problems that developers might not have anticipated, which can then be addressed through redesign or feature modifications.

Mao et al. (2005) highlight that while UCD principles are widely acknowledged in both industry and academia, their implementation often varies significantly across different organizations [20]. Organizational factors such as time pressure, budget constraints, and varying levels of understanding of UCD practices can impact the degree to which these principles are applied. Additionally, there is often a gap between recognizing the importance of UCD and having the resources or organizational buy-in to fully integrate it into the design and development process. Nevertheless, organizations that successfully adopt UCD tend to produce products that are more user-friendly, efficient, and adaptable to user needs, which in turn leads to higher user satisfaction and better market performance.

## 2.3 Technology background

### 2.3.1 Client side

### 2.3.1.1 React.js

ReactJS is a JavaScript library for building user interfaces, particularly for developing large and dynamic web applications. The research methodology includes a detailed exploration of ReactJS features such as the virtual DOM, JSX syntax, and unidirectional data flow. The article examines React's component-based architecture and its integration within the Model-View-Controller (MVC) framework. The conclusion highlights ReactJS's advantages in creating efficient, high-performance web applications, despite its limitations, positioning it as a transformative tool in modern web development [3].

### 2.3.1.2 Next.js

Next.js, a React framework that influences website performance and search engine optimization. Next.js offers significant performance enhancements due to its server-side rendering capabilities, which allow pages to load faster by pre-rendering content on the server before sending it to the client. This reduces the time to the first byte and improves the overall user experience. Next.js provides a comprehensive developer experience with features like automatic code splitting, hot module replacement, and a rich ecosystem of plugins and tools, which streamline the development process [4].

### 2.3.1.3 Angular

AngularJS is a powerful and innovative JavaScript MVC framework known for its novel templating and bi-directional data binding, which significantly reduces the lines of code needed for applications. It emphasizes testing and code quality, promoting best practices in the JavaScript ecosystem. Its popularity has led to a vibrant community, but as it is increasingly used in complex projects, developers may encounter challenges not addressed in standard documentation [5].

### 2.3.1.4 React.js vs. Next.js

The comparison between React.js and Next.js focuses on their popularity, documentation availability, and performance. The research involved analysing data from GitHub, Stack Overflow, NPM trends, and Google Trends to assess popularity, and reviewing official documentation from both frameworks' websites. Performance was compared using Google Chrome's Lighthouse metrics on applications built with each framework. The conclusion was that React.js outperformed Next.js based on these criteria, but Next.js offers server-side rendering, which enhances initial loading speed and user experience [6].

|  |  |  |
| --- | --- | --- |
| **Feature** | **React.js** | **Next.js** |
| **Routing** | Client-side routing using React Router or similar | Built-in file-based routing |
| **Rendering** | Client-side Rendering (CSR) only | Supports CSR, SSR (Server-side Rendering), and SSG (Static Site Generation) |
| **API Routes** | Requires external tools or services like Express | Built-in API routes with serverless functions |
| **Server-side Rendering** | Requires additional setup (like Next.js or similar) | Built-in support |
| **Use Case** | Best for single-page applications (SPA) and client-side rendering apps | Best for server-rendered applications, static sites, and hybrid applications |

Table 1. React Vs. Next

We chose to use Next.js for our final project because it offers several advantages that align with our project's needs. Next.js builds upon React.js by providing server-side rendering, which significantly improves page load times and enhances the user experience. This is particularly beneficial for our project, which involves dynamic content and requires optimal performance. Additionally, Next.js supports automatic code splitting, which helps streamline the development process and improve the scalability of our application. These features make Next.js an ideal choice for creating a high-performance, efficient web application.

### 2.3.2 Server-side

#### 2.3.2.1 Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment designed for building scalable network applications. It is built on Chrome’s V8 JavaScript engine and uses an event-driven, non-blocking I/O model, which makes it lightweight and efficient. Node.js can handle multiple connections concurrently without the need for multiple threads, which is beneficial for performance and scalability. Node.js enables JavaScript developers to write server-side code, allowing for a unified language across both the client and server sides of web applications. The runtime environment is known for its ability to handle data-intensive real-time applications efficiently, making it a popular choice for projects that require fast and scalable network solutions [7].

#### 3.3.2.2 Express.js

Express.js offers a wide range of features through middleware, enabling customization and flexibility. Additionally, its integration with Node.js makes it an ideal choice for projects that require high performance and asynchronous handling of requests. Express.js is particularly beneficial for projects that need a lightweight and fast framework capable of handling numerous connections simultaneously [8].

#### 2.3.2.3 SQL

SQL is known for its simplicity, powerful operators, and productivity benefits. However, it has its significant shortcomings, such as SQL's lack of orthogonality, misalignment with host programming languages, missing functions, incomplete support for the relational model, and limitations in handling nested expressions [9].

#### 2.3.2.4 NoSQL

##### 2.3.2.4.1 MongoDB with mongoose

MongoDB is highlighted for its flexibility in handling unstructured data, allowing for dynamic schema design, which is particularly beneficial for applications requiring rapid iteration and scalability. The database's ability to store data in JSON-like documents makes it easy to integrate with modern applications. Additionally, MongoDB's horizontal scalability through sharding and its robust querying capabilities are noted as key benefits, making it suitable for large-scale data processing and real-time analytics. Other benefits of MongoDB are support for high availability and redundancy, ensuring data reliability and performance in distributed environments [10].

mongoose is an Object Data Modelling library for MongoDB and Node.js. The fundamentals of Mongoose, including schema design, model creation, and data validation. The advantages of using Mongoose are its ability to provide a straightforward schema-based solution to model application data, enforce data structure consistency, and simplify the interaction with MongoDB databases. Mongoose is particularly useful for managing relationships between data, providing hooks for business logic, and offering built-in type casting, validation, and query building [11].

##### 2.3.2.4.2 Firebase

Firebase, Google's platform for mobile and web app development introduces a range of tools, including a real-time database, authentication, cloud storage, hosting, cloud functions, and machine learning. These features help developers manage backend tasks efficiently and focus on creating user-friendly interfaces. Firebase's benefits are ease of integration, scalability, and its ability to speed up app development, making it a valuable resource for developers [12].

#### 2.3.2.5 SQL vs. NoSQL

This text summarizes a comparison between SQL and NoSQL databases, particularly their performance in online transaction processing (OLTP). Stonebraker highlights that NoSQL databases are preferred for their flexibility and performance, especially in update- and lookup-intensive tasks. He attributes performance issues in traditional SQL databases to specific overheads like logging and locking. Stonebraker suggests that these overheads can be addressed to achieve high performance in SQL systems without abandoning SQL or ACID transactions, predicting the emergence of high-speed, open-source SQL engines that maintain ACID properties while offering scalability [13].

# **3. Engineering process**

## 3.1 Data collection

At the beginning of the semester, we received a comprehensive document detailing the requirements that our customers expect from the website. This document was a crucial foundation for our project, guiding us through the development process by outlining the necessary features, functionality, and user experience considerations.

The document is organized into various sections that address different aspects of the website's functionality and design:

* **User Interface requirements:** The document emphasizes a user-friendly, cross-device interface with the ability to add text and images, aligning with usability and flexibility requirements.
* **Categorization:** The document highlights the need for categories like scaffolds, skills, and activities. It also addresses the importance of linking and connecting these categories, aligning with our functional bidirectional linking requirements.
* **Functionality:** The document outlines the need for the website to support adding and managing various elements such as activities, and categorizations. This is directly related to the system’s functional requirements.
* **Strategic Vision:** The document provides a strategic vision for the website, mentioning tools like ResearchRabbit and suggesting the integration of academic sources. This vision aligns with our goal of making the website a comprehensive resource for lecturers to manage and enhance core skills training.

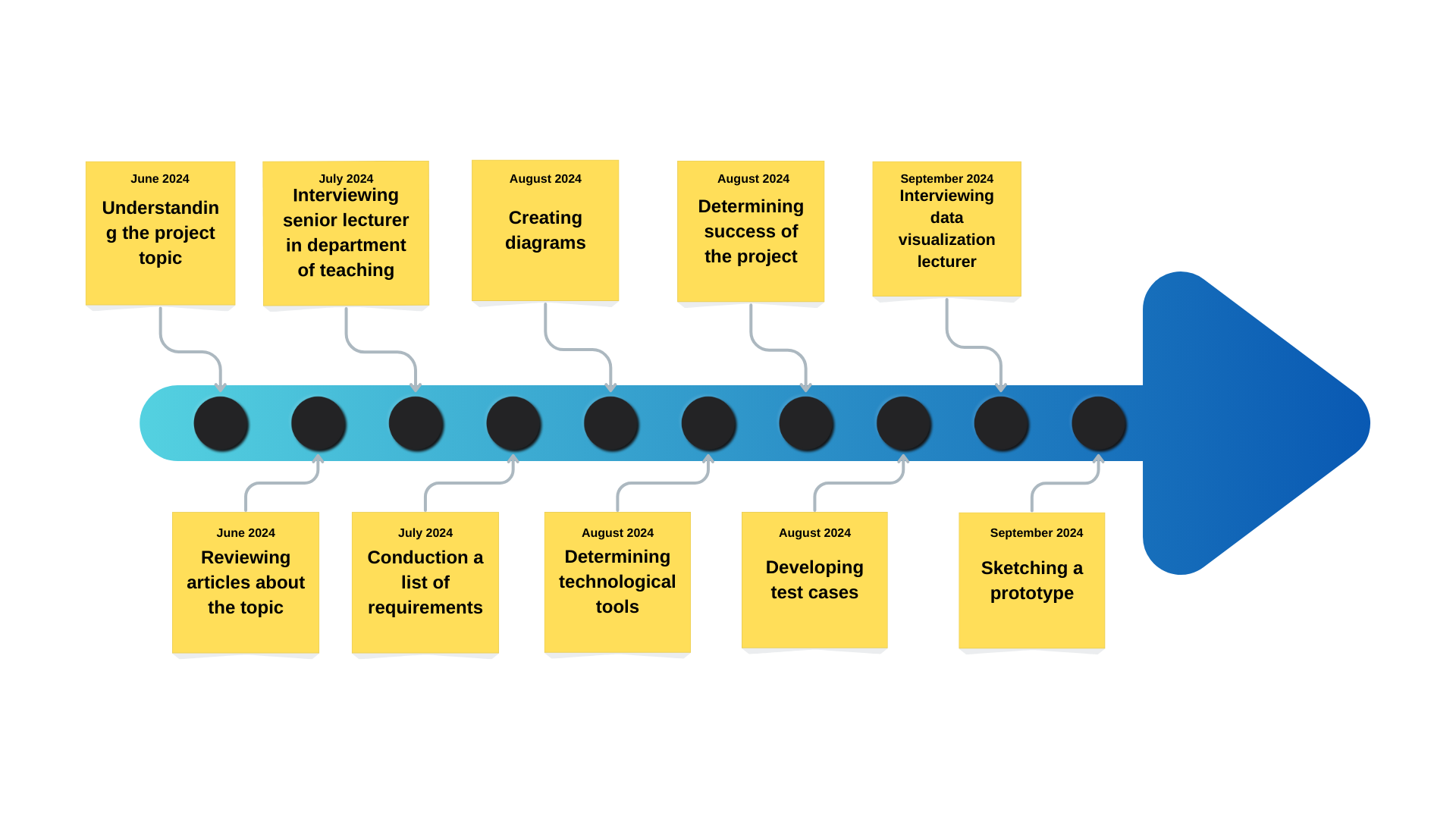
Key considerations included:

* Ensuring that all functional and non-functional requirements were met.
* Incorporating customer feedback to refine the UI and UX.
* Planning for scalability.

This document was a crucial part of shaping the final product, and its detailed analysis helped us anticipate challenges and design a solution that is both robust and user-friendly.

## 3.2 Workflow

The first part of our project involved thoroughly understanding the project topic and conducting a literature review to gather relevant information. We interviewed Dr. Yael Furman-Shaharabani, a senior lecturer from the education department to inform our approach and compiled a list of requirements. Next, we created UML diagrams to represent different aspects of the project visually and determined the technological tools to be used. Additionally, we sought guidance from a data visualization lecturer, sketched a prototype, and developed test cases to ensure the project met all requirements. Finally, we defined metrics to measure the project's success, focusing on user engagement and satisfaction.

Fig 1. Workflow of the project

## 3.3 Interviews

### 3.3.1 Overview

We had an interview with Dr. Yael Furman-Shaharabani, a Senior Lecturer in the Department of Education and General Studies. At a later stage, another interview with Dr. Julia Sheidin a lecturer in the Software Engineering Department, on the topic of Information visualization and Human-Computer Interaction. Lastly, Dr. Naomi Unkelos Shpigel, our supervisor, offered practical feedback based on her experience in practicing core skills and activities in her courses.

### 3.3.2 interviews goals:

* **Interview with Dr. Yael Furman-Shaharabani**: The primary goal was to understand the needs of system administrators, focusing on essential features, and what success would look like for a skill management platform.
* **Interview with Dr. Julia Sheidin**: We aimed to gather insights into how to effectively visualize relationships between core skills, activities, and scaffolds. This involved reviewing graph visualization techniques and designing a user-friendly interface for lecturers to manage and track skills development.
* **Interview with Dr. Naomi Unkelos Shpigel**: As our mentor, she shared her hands-on experience of using skill-based activities in her courses. The goal was to incorporate her feedback to ensure the platform meets practical teaching needs.

### 3.3.3 interviews insights

* **Dr. Yael Furman-Shaharabani**: The project aims to develop effective working methods with students and ensure they acquire the necessary skills. A platform specifically designed for managing and tracking skill development is needed, as there is currently no consistent methodology for managing the process. The platform should integrate all lecturers, making it easier to track and manage skill development. It should be flexible, allowing lecturers to select skills based on their course needs and see examples of how others utilize these tools in similar activities.

Accessibility via mobile devices is important. Ease of use and user-friendly interface are key areas of focus for improving the system, ensuring it is user-friendly for all.

* **Dr. Naomi Unkelos Shpigel**: The system should include a skill menu with filtering options for key elements. The system needs to include various visual aids such as a node and edge graph or tables according to user preferences, enabling a better understanding of skill development progress.

The system needs to be dynamic, allowing administrators to edit and update the list of skills and questionnaires according to the evolving needs of the field. It is important to give lecturers the ability to track and save progress data conveniently and efficiently.

The importance of flexibility in editing open-ended questions, while maintaining consistency in closed-ended questions, was emphasized by the lecturers.

Fig 2. Example of a categorizing activity

A collage of food items

Description automatically generated

* **Dr. Julia Sheidin:** Reviewed several graph visualization techniques suitable for representing relationships on the website.

Gained a clear understanding of the primary goal of the graph, including what key information it needs to convey to the lecturers.

Through the discussion, we evaluated and refined the needs and requirements for the graph, ensuring it effectively supports the intended use cases. Obtained valuable feedback on how the website pages should be structured and designed to best support the graph visualization and overall user experience. The interview highlighted the importance of making the graph and the website user-friendly and intuitive for lecturers, ensuring ease of navigation and understanding.

These insights helped shape the design and functionality of the website, highlighting the need for an advanced and flexible tracking system that meets user expectations while

streamlining the skill development process and providing lecturers with convenient tools for their work.

Fig 3. Core skills categorized

## 3.4 Requirements

### 3.4.1 Functional Requirements

|  |  |
| --- | --- |
| No. | Requirement |
| 1 | **The system shall allow user authentication.** |
| 1.1 | The system shall allow lecturers to sign up. |
| 1.2 | The system shall allow lecturers to log in. |
| 1.3 | The system shall support different user departments. |
| 2 | **The system shall allow search by categories.** |
| 2.1 | The system shall allow search by course. |
| 2.2 | The system shall allow search by activity. |
| 2.3 | The system shall allow search by skill. |
| 2.4 | The system shall allow search by scaffold. |
| 3 | **The system shall show course information.** |
| 4 | **The system shall show skills information.** |
| 4.1 | The system shall provide bidirectional linking between skills and categories. |
| 4.2 | The system shall connect a skill to relevant activities. |
| 4.3 | The system shall connect skills to relevant scaffolds. |
| 4.3.1 | The system shall allow connecting a scaffold for skill building. |
| 4.3.2 | The system shall allow connecting a scaffold for skill development. |
| 4.3.3 | The system shall allow connecting a scaffold for skill reinforcement. |
| 5 | **The system shall show scaffold information.** |
| 5.1 | The system shall connect scaffolds to the relevant skills. |
| 5.2 | The system shall connect scaffolds to relevant activities. |
| 6 | **The system shall present a gallery of activities.** |
| 7 | **The system shall allow activity management.** |
| 7.1 | The system shall allow lecturers to create activities. |
| 7.2 | The system shall allow clicking on an activity to show a detailed view. |
| 7.3 | The system shall allow connecting skills and activities. |
| 7.4 | The system shall allow connecting scaffolds and activities. |
| 8 | **The system shall allow questionnaire management.** |
| 8.1 | The system shall allow adding a new questionnaire. |
| 8.2 | The system shall allow editing of the questionnaires. |

Table 2. Functional requirements

### 3.4.2 Non-Functional Requirements

| No. | Requirement | Type |
| --- | --- | --- |
| 1 | The system shall ensure a user-friendly interface that follows the **ISO 9241-11** standard. | Usability |
| 2 | The system shall load any user action within 2 seconds under normal conditions. | performance |
| 3 | The system shall provide high availability and reliability | reliability |
| 4 | The system shall support adding new courses, skills, and users without requiring significant reconfiguration. | Scalability |
| 4.1 | The system shall be scalable to support an increasing number of users, courses, skills, and activities. | Scalability |
| 5 | The system shall include comprehensive documentation for developers and administrators. | maintainability |
| 6 | The system shall support data export to Excel files | Interoperability |
| 7 | The system shall be accessible on various devices, including desktops, tablets, and smartphones. | interoperability |

Table 3. Non-Functional requirements

## 3.5 Use case diagram

We have created a use case diagram to illustrate the interactions between three user types – lecturer, student, admin, and the system. The system provides a skills graph as the main feature of our platform, enabling users to view and manage various course-related activities, skills, scaffolds, and questionnaires.

* **Lecturer** – Can sign up and log in, view course, skills, and scaffold information. Also, create activities and questionnaires, and connect them to the relevant skills. Lastly, search for information by course, scaffold, activity, or skill.
* **Student** – Interact with the system by filling out questionnaires assigned to them by their lecturers.
* **Admin** – Responsible for managing skills and questions, in addition to having all the lecturers’ privileges.

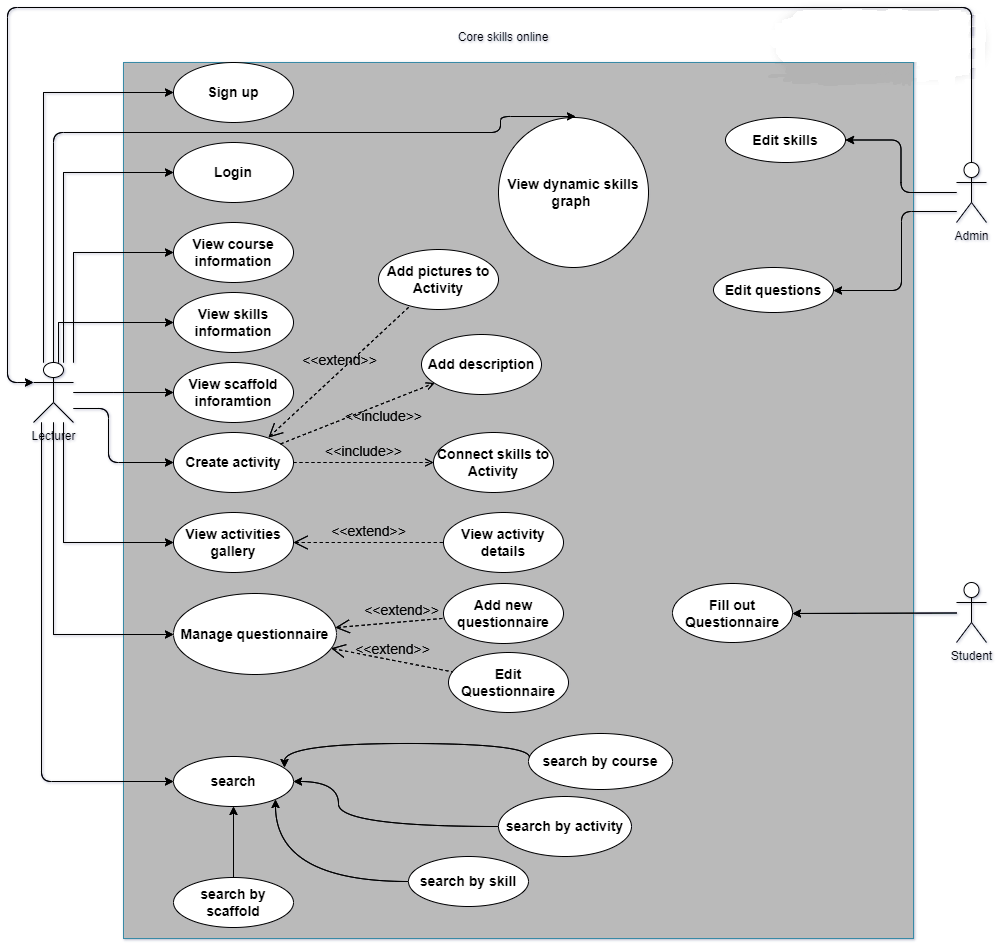


fig 4. Use case diagram

## 

## 

## 3.6 Class diagram

This class diagram outlines the various entities involved in our system, detailing their attributes and methods and showing their relations to each other.

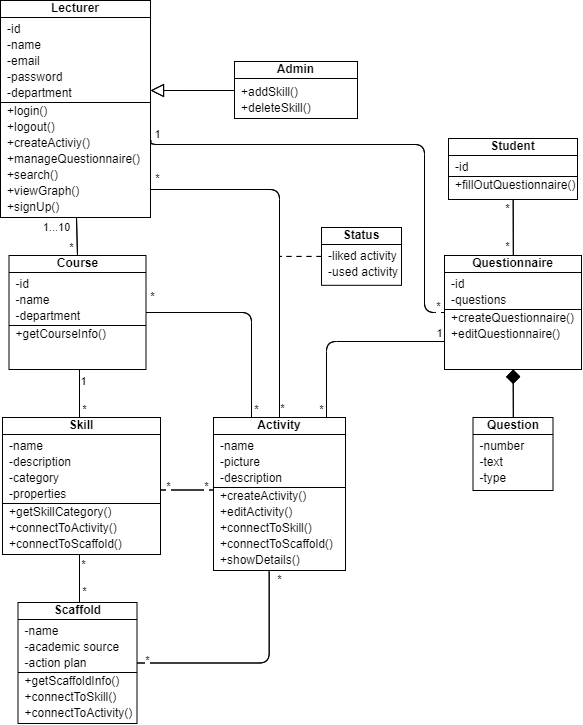


Fig 5. Class diagram

## 3.7 Package Diagram

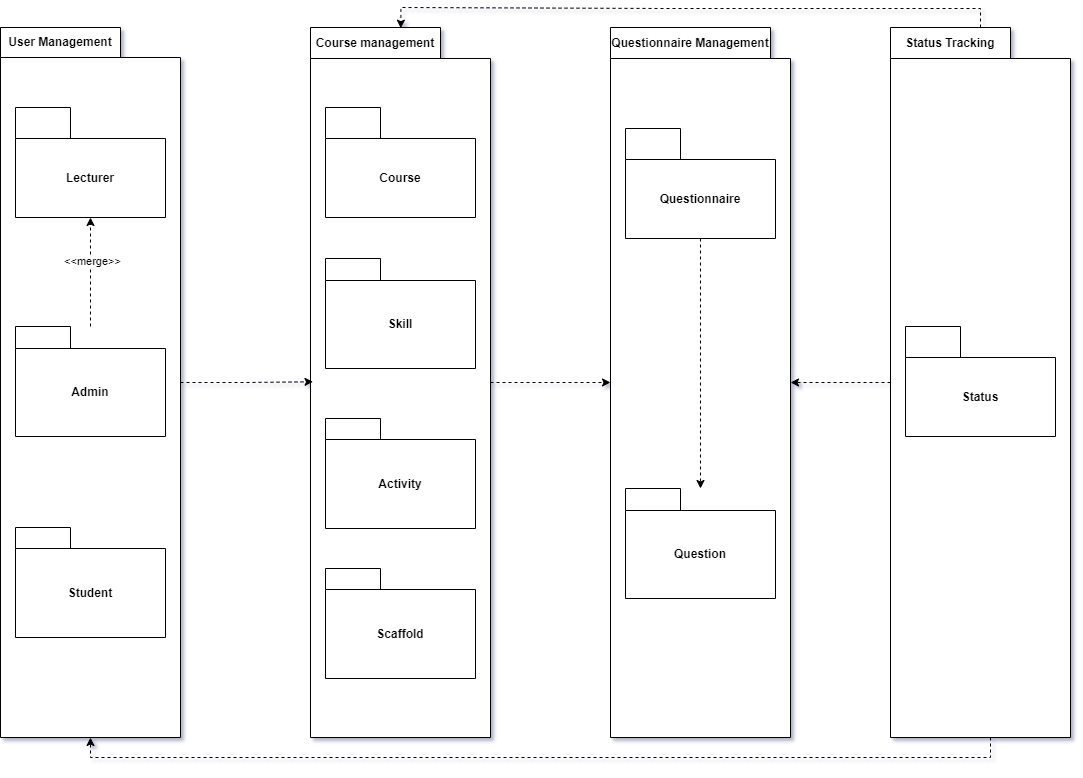
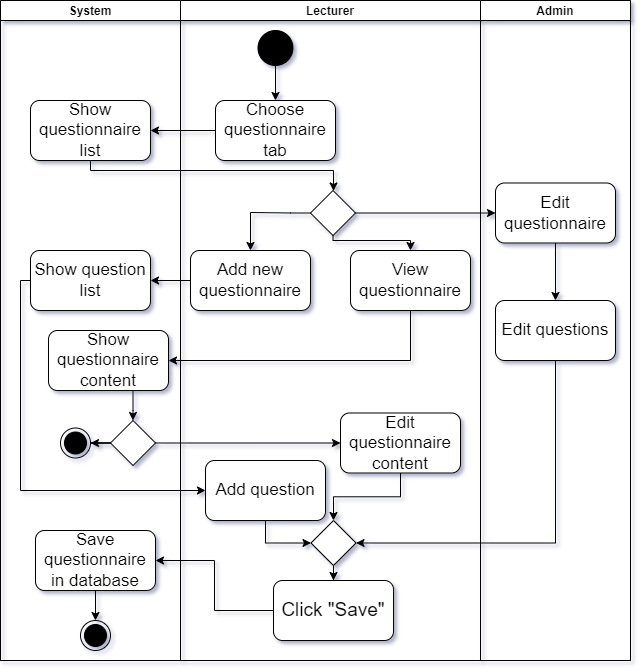
The package diagram shows the module structure of our website. The diagram splits into four main modules: User management, Course management, Questionnaire Management, and Status Tracking. The diagram shows the relationships between the modules, for example, lecturers can manage courses and status tracking monitors the activities and questionnaires.

Fig 6. Package diagram

## 3.8 Activity diagrams

### 3.8.1 Add questionnaire activity diagram

Fig 7. Activity diagram 1

### 3.8.2 user uses core skills database

## 

## 

Fig 8. Activity diagram 2

## 3.9 Architecture Diagram

Fig 9. Architecture diagram

Our platform’s architecture diagram illustrates the separation of the front-end and back-end of the platform. The client side represents the user interaction part of the website, accessible through all sized devices. technologies included are **HTML5**, and **CSS3** to provide the structure and styling of the site while using JavaScript with the help of **Next.js** (a react-based framework) to enable interactive features.

On the server side, we manage the site’s logic, data processing, and interaction with the database. We use **Express.js** - a node.js framework that easily interacts with our server - joined with Mongoose, an Object Data Modelling library for MongoDB providing a schema-based solution for modelling the Database.

For the database, we chose **MongoDB**, a No-SQL database service that offers high performance while being easy to use.

## 3.10 Challenges

* User Authentication and Security: Implementing a secure authentication system that prevents unauthorized access can be complex. One solution that we are leaning towards is using NextAuth as our authentication service.
* Search Functionality: Developing a robust search system that efficiently handles various categories, such as courses, activities, skills, and scaffolds, requires careful planning and possibly advanced indexing techniques.
* Data Modelling and Relationships: Managing the complex relationships between courses, skills, scaffolds, and activities can lead to difficulties in database design and data integrity.
* Scalability: Scaling the system to support 200 users as lecturers, courses, and activities without degrading performance.
* Documentation and Maintainability: Creating comprehensive documentation that is useful for both developers and administrators, can be time-consuming but is essential for the system’s long-term success.

## 3.11 Technologies review

To assess the library's suitability for our purposes to create a skill-activity graph, we created a comprehensive comparison table highlighting the strengths and weaknesses of each library. We employed a color-coding system to indicate the relative performance of each library against specific criteria, with dark green symbolizing the most favourable outcome and dark red representing the least desirable or potentially limiting factor.

### 3.11.1 React Flow

### React Flow is a library for building interactive node-based UIs. It’s highly customizable and easy to integrate with React applications, making it a great choice for creating dynamic, visually appealing graphs that allow users to interact with nodes and edges.

### 3.11.2 D3.js

### D3.js is a powerful JavaScript library for producing dynamic, interactive data visualizations in web browsers. It uses HTML, SVG, and CSS to bring data to life. While it has a steeper learning curve, it's incredibly flexible and widely used for custom visualizations.

### 3.11.3 Sigma.js

### Sigma.js is a JavaScript library dedicated to graph drawing. It's optimized for handling large-scale graphs and is particularly useful for visualizing complex networks. It offers a simple API and can be easily integrated into existing projects.

### 3.11.4 Cytoscape.js

### Cytoscape.js is a graph theory library that allows for graph analysis and visualization. It’s designed for performance and ease of use, supporting a wide range of graph layouts and styles. It’s ideal for both small and large datasets.

### 3.11.5 Graph library comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **React Flow** | **D3.js** | **Sigma.js** | **Cytoscape.js** |
| **Ease of Use** | Easy to use with React (hence with next), intuitive API | Steeper learning curve, requires deep JS/HTML/CSS knowledge | Moderate, with good documentation | Moderate, with a focus on graph theory |
| **Integration with React** | Seamless integration, built for React | Requires additional setup for integration | Possible but not as seamless | Requires wrapper for React integration |
| **Customization** | Highly customizable UI components | Extremely customizable, but more complex | Limited customization options | A high degree of customization, especially in graph styling |
| **Performance** | Good for small to medium-sized graphs | Excellent performance, even for large data sets | Optimized for large-scale graphs | Highly optimized for large graphs |
| **Interactivity** | Strong support for interactive UIs | Excellent interactivity options | Supports basic interactions | Supports complex interactions and graph manipulation |
| **Graph Complexity Handling** | Suitable for simple to moderately complex graphs | Handles any level of complexity | Best for complex, large-scale graphs | Excellent for both simple and complex graphs |
| **Graph Layouts** | Basic layouts available | Extensive, custom layouts | Predefined layouts, not as flexible | Wide range of built-in layouts, customizable |
| **Learning Curve** | Low to moderate, especially for React devs | High, requires an understanding of data-driven concepts | Moderate | Moderate to high, depending on usage |
| **Documentation & Community Support** | Good documentation, active community | Extensive documentation, large community | Decent documentation, smaller community | Well-documented, strong community |
| **Scalability** | Scalable, but performance may degrade with very large graphs | Highly scalable, can handle large datasets | Highly scalable, designed for large graphs | Highly scalable, designed for network analysis |
| **Visualization Types** | Primarily node-based UIs | Wide range of visualization types | Focused on network graphs | Primarily network graphs, with rich analysis features |
| **Popularity** | Moderately popular, growing in React community | Highly popular, widely adopted in data visualization | Less popular, niche but respected | Moderately popular, particularly in research and bioinformatics |

Table 4. Graph library comparison

Based on the comparison table, we chose to work with D3.js due to its exceptional performance, extensive customization options, and ability to handle complex graph visualizations, which aligns well with the needs of our project.

## 3.12 Interface with the client during development

Communication with the client was essential throughout the development process to ensure that the project aligned with their expectations and requirements. This interaction allowed us to gather feedback, make necessary adjustments, and refine the functionalities to meet the needs of the end-users.

The pictures added to the document are a prime example of how we used visual aids to understand the relationship between various skills and their application in real-world scenarios, much like the activity illustrated in the images.

This approach helped us understand the importance of skill identification and categorization in the project, ensuring that each element's role was clear and that the system could provide meaningful insights. Full collaboration with the clients will help us with the development process and ultimately will result in a product that will be aligned with their requirements and vision.

Fig 10. Sunburst chart of the core skills - <https://finalproject19994.github.io/finalProject/>

## 3.14 Success Metrics for the Project

The success of the website will be determined by:

* The number of users who engage with the platform will be a key metric of success, as it reflects the platform's ability to attract attention and interest within the target audience.
* Attracting new users is a crucial first step, as it signifies that the platform’s marketing, outreach, and relevance to educators are effective.
* Without initial user engagement, the platform's features and benefits cannot be fully realized or evaluated, making this the foundational step for the project’s success.
* By focusing on drawing in lecturers and educational professionals, the platform positions itself as a valuable tool in addressing their needs related to core skills management and development.
* **Their level of continued use over time:** The platform must not only capture initial interest but also provide a valuable and intuitive experience that keeps users engaged long-term.
  + Sustained use will depend on the platform offering features and tools that lecturers find helpful and easy to use.
  + A seamless user experience and ongoing relevance are key factors in maintaining engagement over time.
* **Feedback and participation from sceptical lecturers:** A key success factor will be the feedback and participation of lecturers who may initially be sceptical or unfamiliar with the platform and the idea of integrating core skills into engineering education.
  + These individuals are crucial because they represent a segment of the target audience that needs convincing.
  + Their willingness to step outside their comfort zones and engage with the platform, despite initial doubts, will be a significant indicator of the platform's effectiveness.
  + If the platform can address their needs and provide value, it will demonstrate its relevance and utility, even to users who were initially hesitant.
  + Successfully attracting, engaging, and retaining these users will be a strong measure of the platform’s overall success in enhancing the integration of core skills into engineering education.

## 3.15 platform screen sketches

Fig 11. Homepage

**Homepage -** The site's homepage will feature a nodes-edges graph connecting skills and activities. On the left side, the lecturer will have their course menu. The graph will be interactive, meaning that clicking on a node will display the information of the clicked skill or activity. Additionally, the graph will have the option to be filtered according to the lecturer's choice.

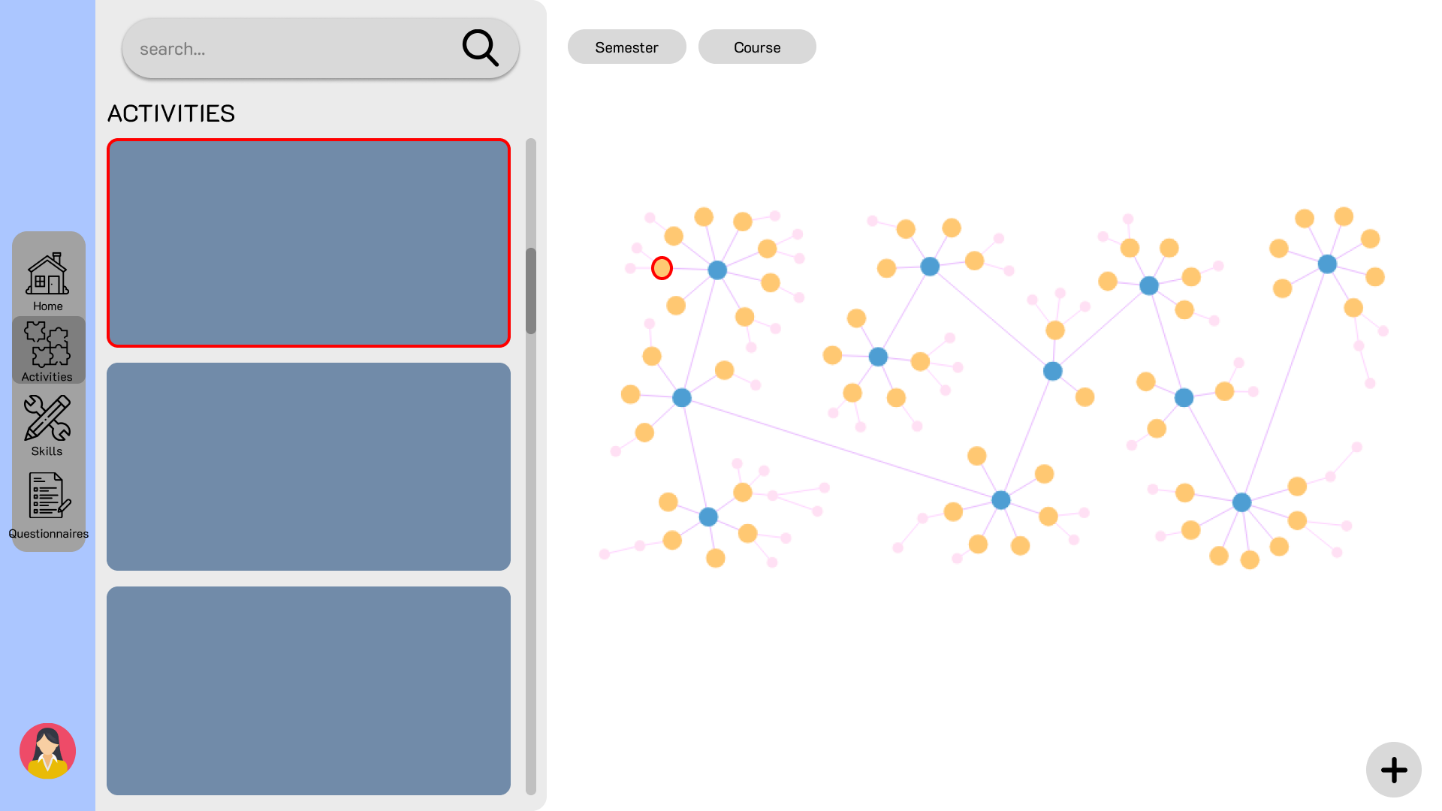


Fig 12. Activities page

**Activities page** - To maintain the site's consistency, this page will also incorporate a nodes-edges graph, which will primarily emphasize activities rather than skills. On the left-hand side of the screen, a list of activities featured in the graph is presented. Selecting an activity will prompt the display of a detailed activity window.

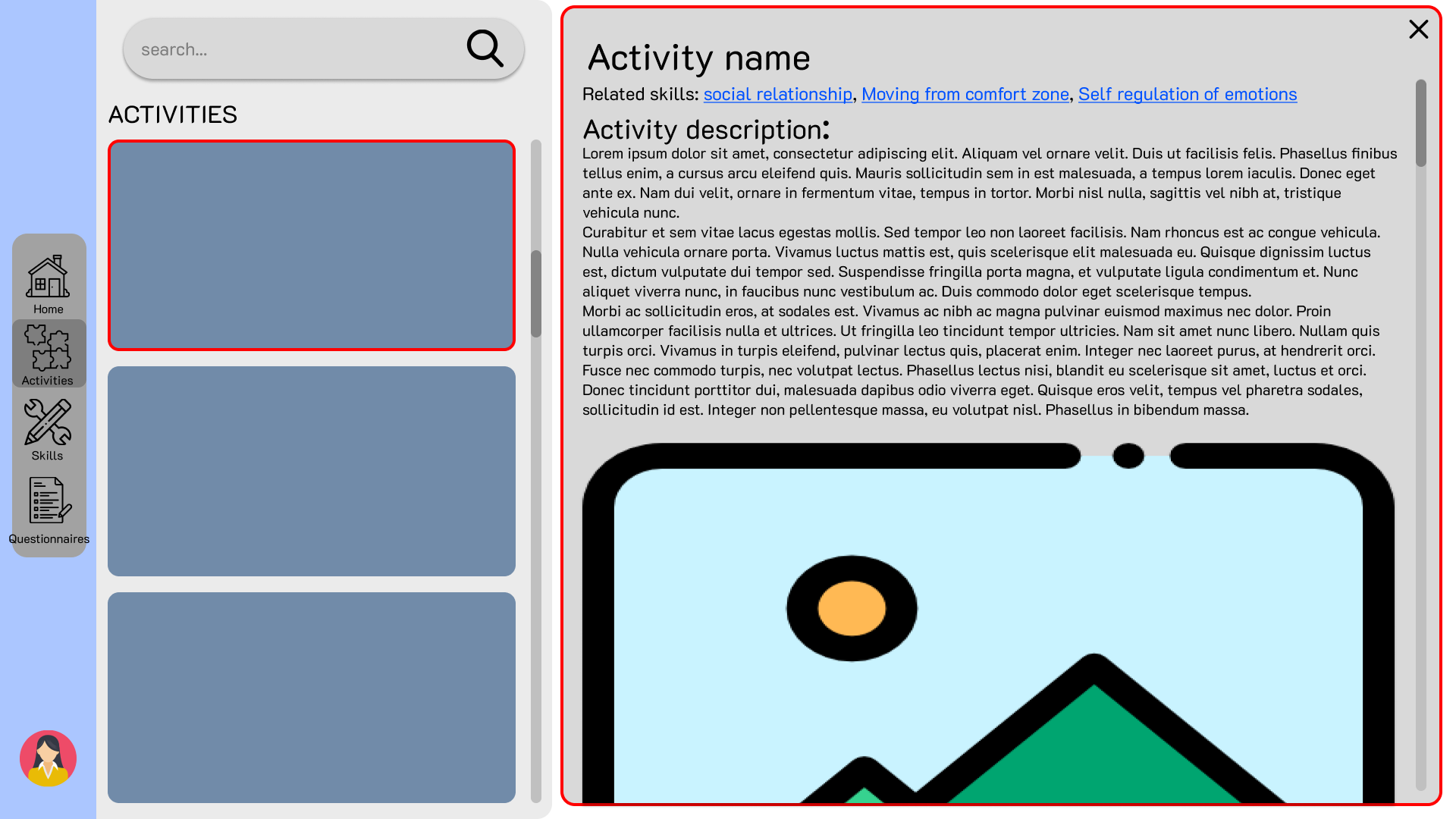
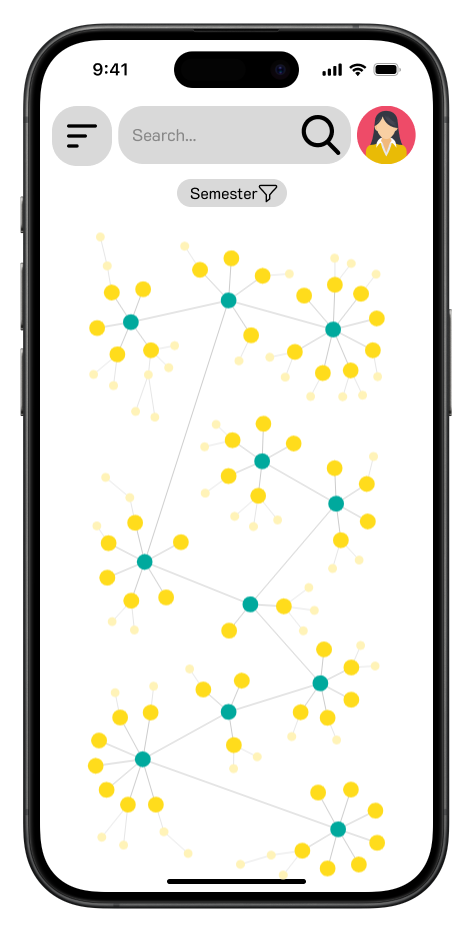


Fig 13. Activity page

**Activity page** -This page will show expended data of the selected activity, still maintaining the site’s flow with the option to access all other activities and hide the data to show the graph.



# 

# 

# 

# 

# 

# 

# 

Fig 14. Mobile - Menu

Fig 15. Mobile - Homepage

# 

# **4. Testing of the platform**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Area** | **Test Name** | **Description** | **Procedure** | **Expected Result** |
| **User Authentication** | Successful User Login | Verify that a lecturer can log in with the correct credentials. | Enter username: lecturer1 and password: correctpassword. Press "Login". | The user is successfully logged in. |
| Incorrect Password Login | Check the system's response to an incorrect password. | Enter username: lecturer1 and password: wrongpassword. Press "Login". | An error message "Incorrect username or password" is displayed. |
| Re-login Attempt | Ensure the system correctly handles attempts to log in by an already logged-in user. | Login as lecturer1 with password correctpassword. Attempt to log in again in another browser tab. | An error message "User is already logged in" appears. |
| **Course Management** | Add new Course | Verify that a lecturer can add a new course. | Login as admin1. Go to the "Add Course" page, fill in the course details, and submit. | The course is successfully |
| **Activity Management** | Add New Activity | Verify that a lecturer can add a new activity to a skill. | Log in as lecturer1. Select a course, go to the "Add Activity" page, fill in the activity details, and submit. | The activity is successfully added and appears under the selected skill. |
| Edit Existing Activity | Ensure that a lecturer can edit an existing activity. | Log in as lecturer1. Select a course, navigate to an activity, click "Edit", modify details, and save. | The activity details are updated and saved successfully. |
| Delete Activity | Verify that a lecturer can delete an activity from a skill. | Log in as lecturer1. Select a course, go to the activity list, click "Delete" on an activity, and confirm. | The activity is successfully deleted and removed from the skill list. |
| **Skill Management** | Add New Skill | Verify that an admin can add a new skill. | Login as admin1. go to the "Add Skill" page, fill in the skill details, and submit. | The skill is successfully added and appears in the skill list. |
| Edit existing skill | Verify that an admin can edit existing skills. | Login as admin1. go to a skill page. Click the "Edit Skill", fill in the skill details, and submit. | The skill is successfully edited. |
| Delete skill | Verify that an admin can delete existing skills. | Login as admin1. go to the "delete Skill" and delete skill. | The skill is successfully deleted and no longer appears in the skill list. |
| **Graph Visualization** | View Skill-Activity Graph | Verify that the skill-activity graph displays correctly. | Login as lecturer1 and see the Skill-Activity graph. | The skill-activity graph is displayed with nodes and edges correctly representing skills and activities. |
| Filter by Skill-Activity | Filter the graph according to user choice | Click on the filter by the semester B button | The skill-activity graph is displayed with nodes and edges only relevant to the user's choice |
| **Search Functionality** | Search for Courses | Test the ability to search for courses by name. | Login as lecturer1. Use the search bar to search for a specific course by name. | The search results display courses matching the search query |
| **Questionnaire Management** | Add New Questionnaire | Verify that a lecturer can add a new questionnaire. | Login as lecturer1. go to the "Add Questionnaire" page, fill in the details, and submit. | The questionnaire is successfully added and available for students. |
| **Data Validation** | Incomplete Course Creation | Check the system's response to incomplete course creation. | Log in as admin1. Go to the "Add Course" page, fill in only part of the required fields, and submit. | An error message indicating the missing fields is displayed. |

Table 5. Tests

# **5. AI usage**

<https://chatgpt.com/share/7d35b9a6-7404-4a3c-babc-6270178a72e3> - A conversation about which visualization library of the graph will be the best considering our project.

<https://chatgpt.com/share/71361c5e-f36d-414e-a708-86d2ccb68b9b> - A conversation to create a graph generation pseudo code and algorithm summary.

# **6. References**

[1] Beagon, U., & Bowe, B. (2023). Understanding professional skills in engineering education: A phenomenographic study of faculty conceptions. Journal of Engineering Education, 112(4), 1109-1144.

[2] Schulz, B. (2008). The importance of core skills: Education beyond academic knowledge.

[3] Aggarwal, S. (2018). Modern web-development using reactjs. *International Journal of Recent Research Aspects*, *5*(1), 133-137.

[4] Patel, V. (2023). Analyzing the Impact of Next. JS on Site Performance and SEO. *International Journal of Computer Applications Technology and Research*, *12*, 24-27.

[5] Kozlowski, P., & Darwin, P. B. (2013). Mastering Web Application Development with AngularJS. Packt Pub.

[6] Dinku, Z. (2022). React. js vs. Next. js.

[7] Syed, B. (2014). *Beginning Node. js*. Apress.

[8] Mardan, A. (2014). *Express. js Guide: The Comprehensive Book on Express. js*. Azat Mardan.

[9] Date, C. J. (1984). A critique of the SQL database language. ACM Sigmod Record, 14(3), 8-54.

[10] Bradshaw, S., Brazil, E., & Chodorow, K. (2019). *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media.

[11] Holmes, S. (2013). *Mongoose for Application Development*. Packt Publishing Ltd.

[12] Chougale, P., Yadav, V., Gaikwad, A., & Vidyapeeth, B. (2021). Firebase-overview and usage. International Research Journal of Modernization in Engineering Technology and Science, 3(12), 1178-1183.

[13] Stonebraker, M. (2010). SQL databases v. NoSQL databases. Communications of the ACM, 53(4), 10-11.

[14] Shcheglova, I. A. (2019). An student engagement in extracurricular activities facilitate the development of their soft skills?. *Мониторинг общественного мнения: экономические и социальные перемены*, (6 (154)), 111-121.

[15] Arat, M. (2014). Acquiring soft skills at university. Journal of educational and instructional studies in the world, 4(3), 46-51.

[16] CHAIBATE, H., & BAKKALI, S. (2017). Skills for employability: Identification of the soft skills required in engineering education. *The Journal of Quality in Education*, *7*(9), 12-12.

[17] Tang, K. N. (2019). Beyond Employability: Embedding soft skills in Higher Education. *Turkish Online Journal of Educational Technology-TOJET*, *18*(2), 1-9.

[18] Caeiro-Rodríguez, M., Manso-Vázquez, M., Mikic-Fonte, F. A., Llamas-Nistal, M., Fernández-Iglesias, M. J., Tsalapatas, H., ... & Sørensen, L. T. (2021). Teaching core skills in engineering education: An European perspective. *IEEE Access*, *9*, 29222-29242.

[19] Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-centered design. Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications, 37(4), 445-456.

[20] Mao, J. Y., Vredenburg, K., Smith, P. W., & Carey, T. (2005). The state of user-centered design practice. Communications of the ACM, 48(3), 105-109.

# **7. Appendices**

## 7.1 [Initial Requirements Form](#_heading=h.2zdcp4jgvbhx)

**מאגר מיומנויות**

* מתאים גם למחשב וגם לטלפון הנייד
* מרצה יכול להוסיף תמונה וכמה מילים של פעילות שעשה

**מאפיינים**

רשתות שמאפשרות גישה מנקודות מוצא שונות: מיומנויות, פיגומים, פעילויות

**מרצים**

שם משתמש

ססמה

מחלקה

האם אדמין הוא בהכרח מרצה או שהוא יכול להיות גם איש חיצוני?

**קורסים**

מס קורס

שם קורס

מחלקה

**מיומנויות**

כל מיומנות: משפט, מאפיינים, קטגוריה

(קישור קטגוריה - מיומנויות דו צדדי)

שילוב המיומנות בפעילויות שונות.

קישור לפיגומים לבניית/פיתוח/חיזוק המיומנות (לשאול מה המשמעות של המשפט?)

קישור לפעילויות (בקישור, האם הכוונה היא בconnection או בlink?)

**פיגומים - אסטרטגיה לבניית / פיתוח / חיזוק מיומנות**

שם פיגום

מקורות אקדמיים - researchrabbit

דרך פעולה

קישור למיומנויות

קישור לפעילויות

(האם כל מרצה יכול להוסיף לאתר פיגומים באופן עצמאי ואינדיבידואלי או רק דרך הפעילויות המתאימות?) כן!

**פעילויות**

גלרית פעילויות.

לכל פעילות - שם, דוגמא/ות, התאמה ל- תרגול / פעילות בקבוצות / הסבר הנושא / דיון, תמונה/ות. לחיצה מובילה להסבר הפעילות (כמו אצל יעל) - מה מאפשרת? תורמת? שלבים, דגשים.

שילוב מיומנויות בפעילות.

קישור לפיגומים

**גישה לסטודנטים**

כניסה לשאלונים עם ת.ז. - שאלונים בגישה משחקית.

מיוצא לאקסל.

**במהלך הזמן להוסיף**

* מרצים שהשתמשו ויכולים להמליץ
* להרחיב פעילויות
* אפשרות להמלצת המערכת על פעילויות נוספות בעקבות סימון "אהבתי" על פעילות מסוימת - שילוב AI
* לחשוב אם אפשרי לעשות דבר דומה לגבי המיומנויות

## 7.2 List of the skills and their sub-skills