jamk

Supporting students' career planning and skills development

CASE: Development of an API service based on SFIA skills

Lauri Kuuva

Bachelor's Thesis
5/2025
Information and Communications Technology

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Cover page

Lauri Kuuva

Supporting students' career planning and skills development CASE: Developing a Sfia skills-based API service

Jyväskylä: JAMK University of Applied Sciences. April 2025, 40 pages.

Information and Communication Technology degree programme. Thesis AMK.

Language of publication: English

Publication permission on the open web: yes

Abstract

This project investigated the possibility of developing an API solution that retrieves data from the SKILL-db database regardless of the operating environment. The aim of the project was to determine how JavaScript technologies can be used to implement data retrieval systems. During the development process, the required components of the API functionality, compatibility with other platforms and presentation of information in a user-friendly and interactive way were considered. The results of this work prove that the initial solution is feasible, but requires further development, in particular in terms of database integration optimisation and scalability. This study provides a basis for the development and application of API solutions for different user environments and purposes.

Keywords (subject headings)

SFIA Framework, API Development, Unit Testing, Career Path Planning, REST API, JavaScript, Educational Technology, PostgreSQL, Frontend Development, Backend Integration, Software Testing, Interactive Learning Tools

Other information (confidential attachments)

-

Content

1 Introduction

In today's technological world, teaching is becoming increasingly electronic, which is why many different digital tools have been developed to support teaching. Yet many students change their learning path or drop out, raising questions about how to better support students' needs. This need led to the idea of developing a tool to help understand students' motivations and goals in their career choices and to link the skills available to students and the courses offered by institutions to widen their career planning.

The project integrates the international SFIA (Skills Framework for the Information Age) framework, which defines the skills and competences required for specialist roles in the information economy. This thesis designs and implements an API database service based on the SFIA framework, which allows users to search for information on the skills required for different career paths. The API will be used to test whether the SFIA skills search service can be implemented in such a way that it can be easily deployed in different environments and serve the needs of both students and institutions. The study will also explore how the visualization of SFIA skills could support students' skills development and guide them towards their career goals.

This project investigated the possibility of developing an API solution that retrieves data from the SKILL-db database regardless of the operating environment. The aim of the project was to determine how JavaScript technologies can be used to implement data retrieval systems. During the development process, the required components of the API functionality, compatibility with other platforms and presentation of information in a user-friendly and interactive way were considered. The results of this work prove that the initial solution is feasible but requires further development, in terms of optimization of database integration and scalability. This study provides a basis for the development and application of API solutions for different user environments and purposes.

2 Purpose, aims and objectives

SFIA API is one of Marko Rintamäki’s many ongoing projects, and in this assignment Rintamäki, who works at JAMK as a Senior Lecturer who is well versed in latest technologies and researching ways to implement these technologies into applications for students. He asked the author to investigate if such an API could be developed, and if it could work in an environment such as GitHub

Pages. Rintamäki's goals to bring more visuality to learning are part of the objectives, where the purpose of the API is meant to visualize the skills needed. This started a project to find out if it was possible to design an API solution that could be designed that would work in as many environments as possible, which would mean creating the lightest possible solution.

Significance

The development work of SFIA API is relevant from several perspectives. It should provide a solution to the need for students and teachers to have clear information on the skills required for professions and career paths. SFIA API should allow users to map the skills required for different occupations and tasks and to compare them with the learning content offered by education and training. This information would help students to plan their studies and goals in a more informed way, which can reduce dropouts and uncertainty in career choices.

SFIA API also should enable institutions and organisations to better link their educational content to the needs of the world of work. By knowing what skills are valued in the labour market, course contents can be developed to meet these demands. This will enable closer cooperation between education and the world of work and improve the graduates' employability

The SFIA API is to provide a technical solution that can be deployed in many different environments thanks to its scalability and extensibility. It serves as a powerful tool for managing and visualising career path information, making it a flexible and valuable tool for educational institutions and other organisations.

Objective of the project

The aim of the project is to see if it is possible to develop an API solution that retrieves and visualizes data from the SKILL-db database regardless of the operating environment. The implementation will use JavaScript technologies as a starting point for the solution.

3 Theoretical background

Reference Framework

This thesis is divided into two parts, of which the first part is theoretical focusing on the requirements and requirements elucidation in software development. The second section is the practical part, which focuses on the development of the prototype itself, and the tools and technologies used to develop it.

3.1 Concepts

WIMMA Lab was a two and half months long course provided by JAMK for information technology students which was run annually during the summer vacation, where students were divided into imaginary departments to simulate working in real-life company (WIMMA Lab, 2023). Each department had one or more projects to complete during the summer and teaching the agile development methods and project-based teamwork skills and value of collaboration.

The SKILL-db is a database programmed and developed by Wimma Lab 2023 IoTitude group, implemented according to the specifications of the eighth (8) version of SFIA. SFIA Foundation has already released the next version of SFIA, which is the ninth (9) version, and differs from eight in some key areas. It still retains the seven levels of responsibility but introduces new Level Essence Statements to further clarify descriptions of the SFIA levels and closer alignment with generic attributes (Moving to SFIA 9, 2024). It also introduces few new skills for business administration (*SFIA 9 Full Framework View*, 2024).

The SKILL-db API is the API interface of the Sfia database which allows reading data from the database. The SFIA API is an API now developed that will use SKILL-db API to retrieve exact information from the database for presentation on, for example, web pages.

JAMK, Jyväskylän ammattikorkeakoulu (Jamk University of Applied Sciences) is a university for applied sciences located in Jyväskylä. JAMK provides many different Bachelor's degree programs and many master's degree programs. All these programs can be studied in English. This study is focusing on this educational institute.

3.2 Data management and databases

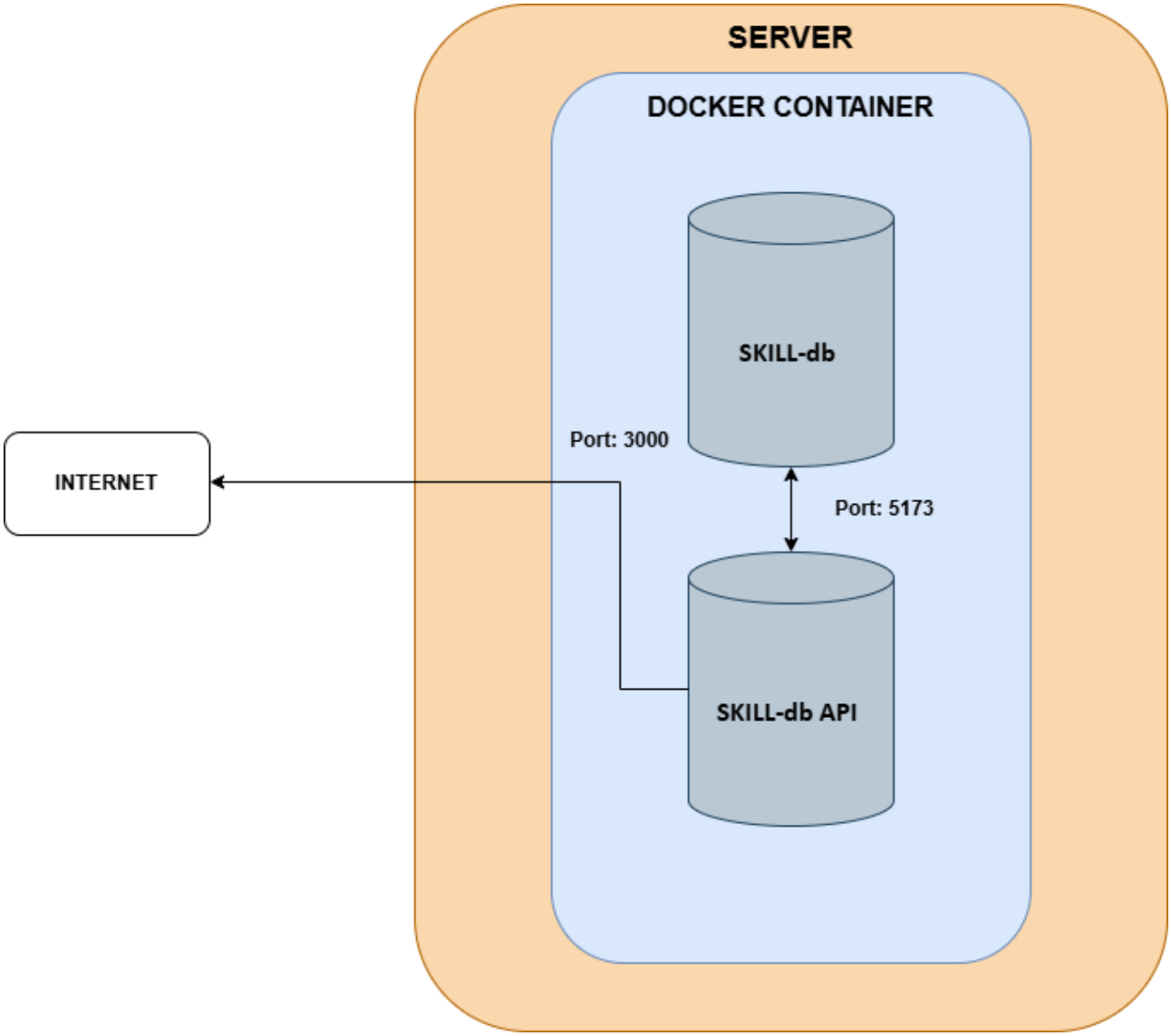

Figure 1. SKILL-db structure

The figure illustrates the structure of the SFIA database known as SKILL-db which was developed by IoTitude project in Wimma Lab using the accompanying SKILL-db API that will search for data by keywords and provides the skill data of them according to the definitions of the function used. For example, accessing path /skills returns the whole database, and /category can be used to return skills of specific category. As seen in the figure above, the SKILL-db and SKILL-db API images are served in one Docker Container and is ready to use after changing the default passwords and secrets, and composing the build with Docker.

3.3 Research questions

For this work, several different questions needed to be answered to produce a complete implementation that matches the goals set from the customer of this study Marko Rintamäki, who wants to see this kind of technology to be used in his projects like WIMMA Capstone and Future Factory.

The research questions for this study are listed below:

1. How relevant is the SFIA Framework to students' future career plans?
2. Do students see a connection between the SFIA framework and their professional skills development?
3. Would students consider SFIA to be a useful tool for evaluating and improving their skills?
4. Do students think their university should integrate SFIA into their curriculum or offer courses based on the framework?

For these questions to be answered, a method was needed to gain the answers. The methodology of the research is described in following chapter.

3.4 Methodology

For the SFIA API, technical testing methods such as unit testing and integration testing were employed to ensure the reliability, performance, and security of the API. Each development phase was evaluated using a combination of user feedback and technical metrics, including performance and usability tests. The ultimate goal was to create comprehensive and easy-to-understand documentation that would facilitate the deployment and integration of the SFIA API within educational institution systems and applications.

To assess the level of interest in the SFIA Framework, a quiz was created, targeting at least two student groups. The thesis author conducted a brief introductory presentation (Appendix 3) for these classes, explaining what the SFIA Framework is, its goals, and the importance of understanding student interest in such technology. This approach was deemed the most suitable for this study, as it was necessary to first gauge the students' interest and determine whether SFIA-based technologies could play a more prominent role within JAMK and similar educational environments.

The quiz was designed to be anonymous, with no personal information such as names or email addresses collected from respondents. It utilized a quantitative research method, allowing data to be gathered from a large group of participants through structured questions. The survey was composed of multiple-choice questions to streamline the data collection process. Microsoft Forms was used as the tool for the quiz, as it is user-friendly, accessible, and enables efficient data gathering. Additionally, it ensured that only JAMK ICT students had access to the quiz, maintaining the integrity and relevance of the data collected.

4 Implementation

This section focuses on the development process of the SFIA API and explores how the existing SKILL-db API could be further developed to achieve the objectives for which the SFIA API was created. The chapter outlines the key development phases: the design phase, the development phase, and an evaluation of the outcome—assessing whether the goals were met through the activities and design choices made, using the research methods selected for this project.

The development of the SFIA API began as part of the Career Scouter 2.0 project, where it was utilized to display the skills associated with various roles and the corresponding responsibilities of these professions. The SFIA API worked by transforming the data from the SKILL-db and displaying it within a hover element on the webpage. When a user hovered the mouse over a skill element, a separate informational box would appear, showing the skill's description.

At this stage of development, the implementation was a basic solution, consisting of a simple black box with white text, providing a description of the skill. Since the theme of the interface was customizable by the end user, the focus during this phase was placed on functionality rather than appearance. As a result, minimal time was spent on refining the visual design, with the primary goal being to ensure the core features worked as intended.

4.1 Quiz

In this chapter, we focus on the quiz held by the author. It details the participants, the questions of the quiz, who it was held to and the ethicality of the quiz.

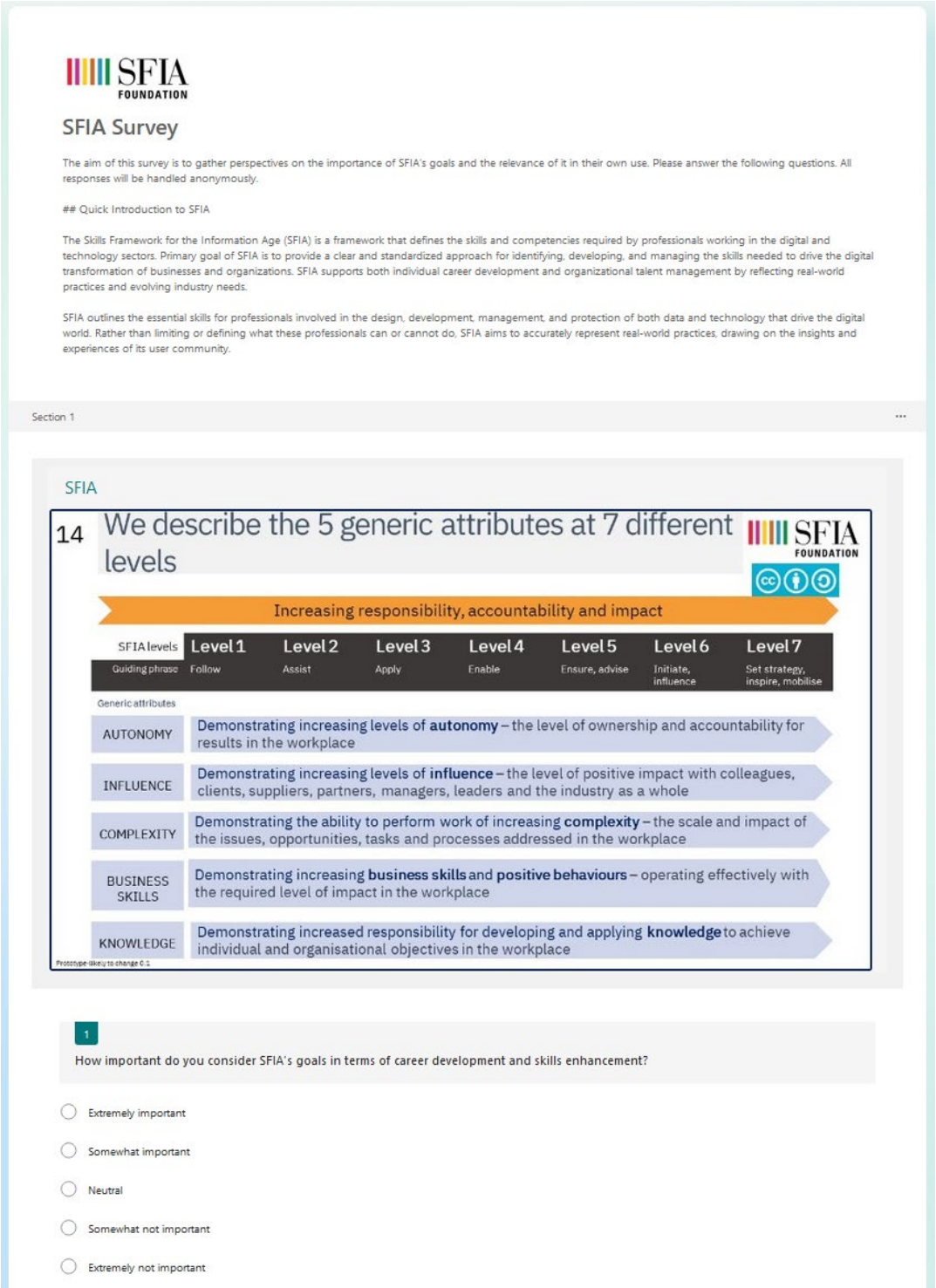

Figure 1. Quiz Layout

In figure above, which illustrates the structure of the quiz used to further chart the interest for SFIA based technologies and web services aimed to help students to plan their study paths and what skills their careers would need. For this anonymous quiz two first year student groups were selected from the Institute of Information Technology. These classes were TIC24S1, ZJATIC24S1 and TIC24S2. The multiple-choice quiz was presented as follows:

"The aim of this survey is to gather perspectives on the importance of SFIA's goals and the relevance of it in their own use. Please answer the following questions. All responses will be handled anonymously.

The Skills Framework for the Information Age (SFIA) is a framework that defines the skills and competencies required by professionals working in the digital and technology sectors. Primary goal of SFIA is to provide a clear and standardized approach for identifying, developing, and managing the skills needed to drive the digital transformation of businesses and organizations. SFIA supports both individual career development and organizational talent management by reflecting real-world practices and evolving industry needs.

SFIA outlines the essential skills for professionals involved in the design, development, management, and protection of both data and technology that drive the digital world. Rather than limiting or defining what these professionals can or cannot do, SFIA aims to accurately represent real-world practices, drawing on the insights and experiences of its user community."

The first question was *"How important do you consider SFIA's goals in terms of career development and skills enhancement?"* and options to answer were *"extremely important", "somewhat important", "neutral", "somewhat not important"* and *"extremely not important".*

The second question was *"How useful do you find SFIA for your learning and professional development?"* and the alternatives given were the same *"extremely important", "somewhat important", "neutral", "somewhat not important"* and *"extremely not important".*

The third question was *"Do you think there would be use for SFIA framework and technologies that use SFIA in your study path?"* where the acceptable answers were *"yes", "no"* and *"maybe".*

The fourth and final question was *"How would you rate the usability of SFIA in learning and skills management?"* and the acceptable answers were *"excellent", "good", "neutral", "fair"* and *"poor".*

4.2 Technology used

SKILL-db was implemented using Express.js, a lightweight and flexible Node.js-based framework. With Express, the API can provide an efficient way to create routes, handle HTTP requests and manage cache according to RESTful principles. The code itself was written in TypeScript.

SKILL-db was a Node.js-based API implemented in TypeScript. It was built using the Express.js framework, with PostgreSQL as the database. The API was containerized with Docker, allowing for easy deployment and scalability. The Nginx web server also runs in the background to receive incoming requests and improve performance (Skill-db Documentation 2023). SKILL-db uses a PostgreSQL database to store and manage skills data. This database supports extensive data management and provides advanced features such as in-description and ACID integrity. The API was containerized using Docker, which provides simplified deployment and scalability, and allows the service to run easily on-premises.

Nginx acts as a web server to receive incoming requests, and the Docker container allows the entire suite to run quickly, with Nginx directing traffic to the Express server, which uses a PostgreSQL database for data management.

Docker

Docker is an application deployment automation tool based on application deployment containers that can be deployed to different environments easily and efficiently using Docker. Docker works by packaging an application and all the dependencies, platforms, environments and connections between them needed (operating systems, programs, libraries, languages etc.) to run it into a compressed container, which can then be run in multiple locations, depending on the user/business need, including on-premises and in the cloud (Docker, 2018). The lightweight nature of Docker containers brings a possibility for a single server or virtual machine to run multiple Docker

containers at the same time, reducing the amount of maintenance required on creating, maintaining multiple virtual machines, which in turn reduces the time and resources spent on maintenance.

PostgreSQL

PostgreSQL was selected as the database management system for SKILL-db due to its extensible open-source architecture and its ability to accommodate modifications efficiently. As a relational database system, PostgreSQL adheres to ACID (Atomicity, Consistency, Isolation, Durability) principles, ensuring data integrity and reliable transaction handling. Additionally, PostgreSQL offers a comprehensive set of SQL functions and operators, supporting complex data management tasks while maintaining efficiency in handling large datasets.

Although PostgreSQL includes advanced features such as indexing, transactions, JSON support, and spatial data processing, many of these capabilities remain unused in SKILL-db. This is primarily due to the static nature of the database, which typically does not undergo modifications once initialized. As a result, while PostgreSQL's advanced functionality provides potential scalability, the SKILL-db implementation primarily benefits from its stability and efficient querying capabilities rather than its full range of features.

The current framework of SKILL-db is presented in Appendix 1.

Below is an example of the very early implementation used in the Career Scouter 2.0 prototype:

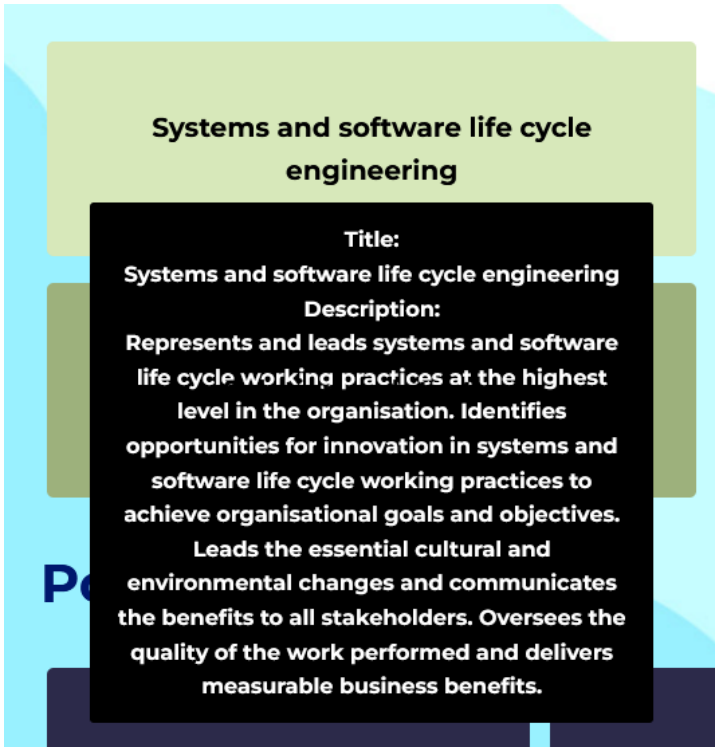

Figure 2. Testing SFIA API in the prototype version of Career Scouter

With this implementation, SFIA API can already be declared functional in the Node-based Vue.js environment and in some other environments based around JavaScript/HTML frameworks. What remains to be carried out is to make SFIA API work on GitHub Pages, which will fulfill the client's wishes for SFIA API to be deployable firstly on the platform of their choice and secondly to fulfill the need for the possibility of deployment in other environments.

4.3 API development

This development was carried out in phases, focusing on the scalability and interoperability of the SFIA API with different environments. In the first phase, the SFIA API was designed to support a database service related to SFIA skills, allowing users to search and visualize the skills and competences required for different career paths. The SFIA API provides access to this data, allowing both students, career tutors, teachers as well as study counsellors and degree programme coordinators to use it in their own planning and development.

The SFIA API was developed in a modular way so that it can be easily customized and extended in the future. The aim for the SFIA API is to be platform-independent and to integrate seamlessly with the various information systems that educational institutions and other organizations can use. The development work will also consider security and performance requirements to ensure that the SFIA API is reliable and fast even in large-scale deployments.

4.3.1 Restrictions

Developing the SFIA API for GitHub Pages presented certain challenges, primarily due to the limitations of GitHub Pages in handling external dependencies. Specifically, required modules such as **axios** could not be bundled with package managers like Vite or Webpack, as GitHub Pages does not support such bindings. This restriction left only two possible solutions: either importing axios via a **CDN (Content Delivery Network)** within a separate `<script>` block or downloading axios manually and including it within the API files.

To simplify development and minimize dependencies, the latter approach was avoided. Instead, the decision was made to use as few external packages as possible—ultimately eliminating the need for any additional modules. Rather than relying on axios for handling HTTP requests, the **native JavaScript `fetch` function** was utilized. Since `fetch` is built into modern JavaScript environments, it does not require external downloads or CDN imports, ensuring a more lightweight and easily deployable API.

4.3.2 Tools and technology

SFIA API is simple in its implementation and uses only JavaScript and the axios module to contact Sfia-db, the address of which must be determined by the end user and set to the correct format when SFIA API is deployed. Otherwise, SFIA API can be used either as a loadable module in the user's own project or directly as a function.

SFIA API consists of two parts SFIA API itself and SFIA Visualize. SFIA API handles communication with the SKILL-db and has four functions, which are as follows:

1: getSkills which retrieves all the skills in the SKILL-db.

2: getSkillsByCode retrieves a specific skill from SKILL-db and returns it and its accompanying information.

3: Function getResponsibilites retrieves the Sfia responsibilities and returns them.

4: Function getResponsibilitiesByCode returns the responsibility information retrieved by the SFIA code and returns it.

5: Test these functions with Unit Testing. (See Appendix 3)

SFIA Visualize, on the other hand, handles the presentation of this information and provides for this a function, which retrieves the given code for the skill or responsibility in question, and displays it as a box when the user hovers the mouse over the target gesture where SFIA Visualize set to activate.

5 Results

This research has demonstrated that it is possible to create lightweight REST API solutions, such as the SFIA API, which can be used to implement various data visualizations—particularly for educational purposes. These visualizations add value to learning environments where presenting complex information in an easy-to-understand, interactive format is essential.

The SFIA API can be hosted on a GitHub Page or any other environment capable of running JavaScript and connecting over the Internet to the SKILL-db service. This flexibility allows it to be integrated into a wide range of applications and websites without requiring complex server infrastructure. Its lightweight design ensures scalability, while the speed of SKILL-db's PostgreSQL database enables rapid data retrieval, making it well-suited for handling large datasets efficiently. This is particularly important for interactive visualizations and real-time data processing, where fast data access is critical for maintaining smooth user experience.

Up next, this paper will detail the two distinct plans for achieving these results: the further development of the SFIA API and the necessary optimizations to the SKILL-db and SKILL-db API. These plans outline the technical steps required to enhance system efficiency and usability.

Plan 1: SFIA API + SFIA VISUALIZER

Figure 4 below illustrates the structure of this method where SFIA API uses SKILL-db API to retrieve data and transforms it on the web service side to usable data.

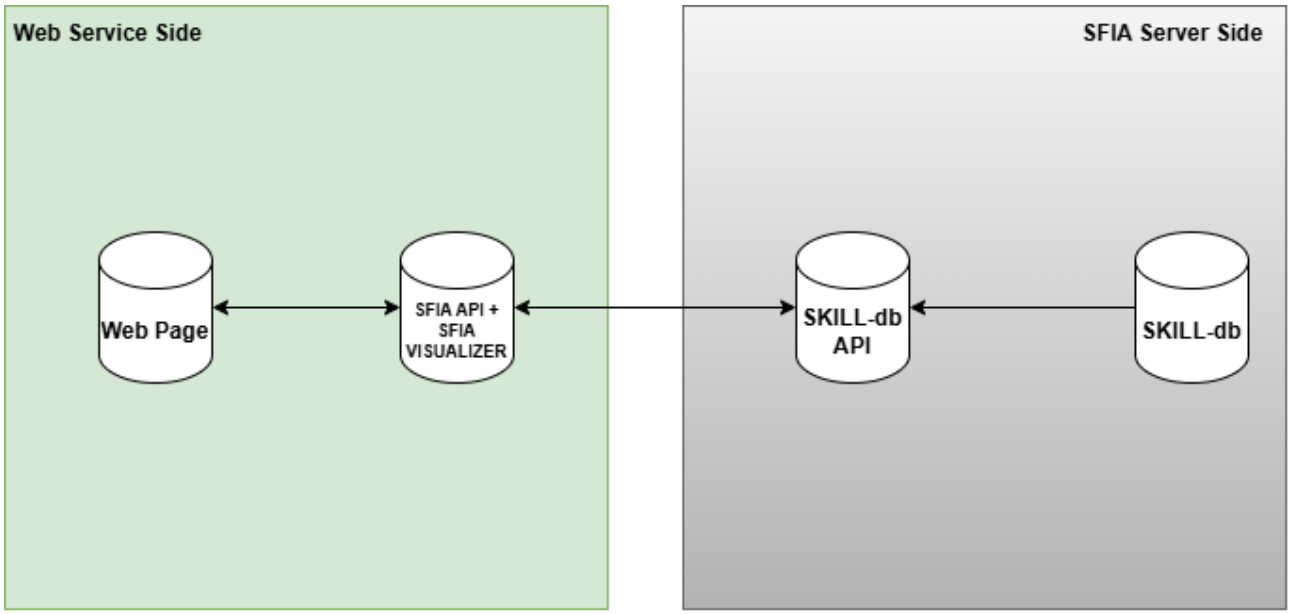

Figure 3. Plan 1: SFIA API + SKILL-db API

In this plan, the SFIA API initiates a fetch request to the SKILL-db API to retrieve the full contents of the SKILL-db. It then processes this data, filtering it down to the required role or roles before passing it forward to the SFIA Visualizer, which displays the relevant information to the user. However, this approach is resource-intensive for both the end user and the server hosting the SKILL-db. Each time a user interacts with a new element on the website, a fresh request is sent, and the server responds by transmitting the entire database table again, leading to unnecessary data transfers and increased server load.

To address these inefficiencies, the author proposes further development of both the SKILL-db and the SKILL-db API, as outlined in “Plan 2”. This approach aims to optimize data retrieval, reducing redundancy and improving overall system performance.

Plan 2: SKILL-db and SKILL-db API Rework

Plan 2, as illustrated in Figure 5, focuses on enhancing the existing SKILL-db API, which currently provides only basic access to the SFIA database. The primary goal of this development is to optimize the way skill data is retrieved by reducing redundant full-database queries. At present, every request results in the API returning the entire dataset, which is inefficient.

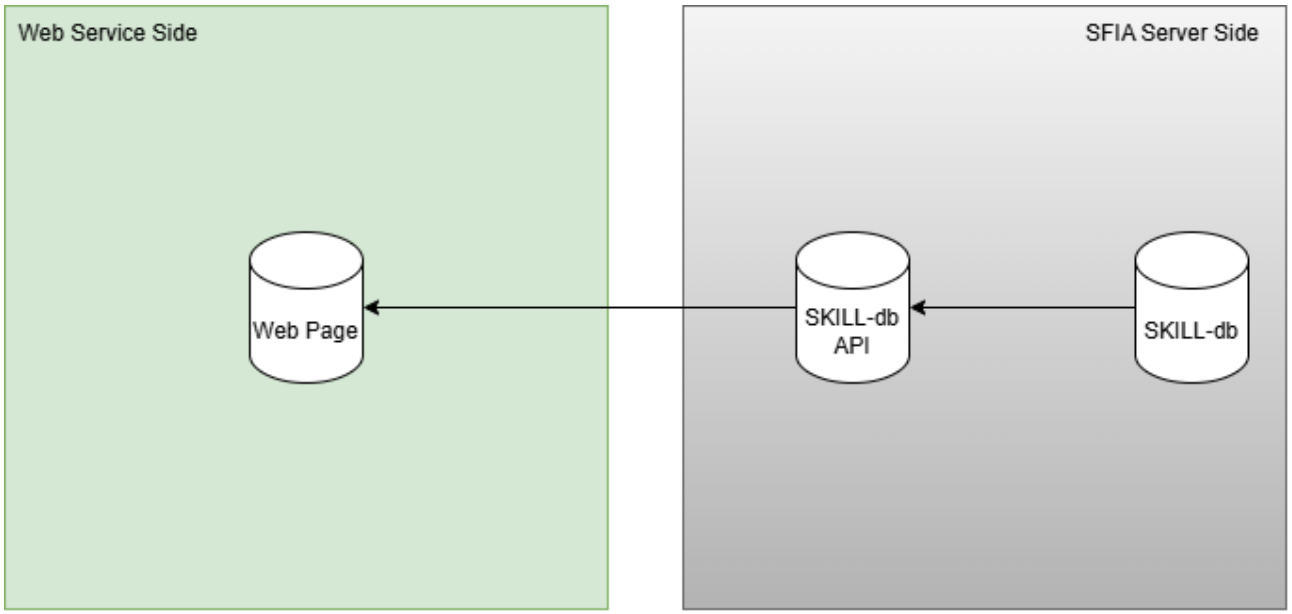

Figure 4. Plan 2: Web Services use SKILL-db API directly

To improve this, the SKILL-db API should be extended to allow requests for specific skills based on a given identifier, similar to how the SFIA API retrieves individual skills using their four-letter iden-

tification code. Additionally, the response should include SFIA's color coding system, which categorizes skills into different fields. This would enhance visualization on the web service side, making it easier to present skill-related information in a structured and meaningful way.

Extended functionality on the SFIA-db API side would make creating another API redundant, as SKILL-db API could return all the data requests fully. The visualization of this data then relies on the Web Service or Web Page developer to carry it out, and the SKILL-db API requests cannot be modified by the Web Service developer, whereas in method 1 the developer can edit the requests to suit their needs. In this method, the SFIA API is no longer needed, and SFIA Visualize sends the fetch request to SKILL-db API directly to construct the elements for the skill from returned data from the SKILL-db API.

Unit Test results

During the development, Unit Testing was used to reach the goals for desired API behavior and to ensure the robustness and reliability of the SFIA API. A Unit Testing Manifest was created to guide and automate the testing process by outlining specific test cases targeting the most critical functionalities of the API. These included tests for data retrieval, error handling for invalid requests, performance under large data loads, and response format validation. Testing was done using modern JavaScript testing frameworks Jest and Supertest, which allowed the simulation of HTTP requests to the API endpoints and helped ensure consistency between expected and actual responses. In addition, native JavaScript features such as the fetch() function were preferred over external packages like Axios, due to the technical limitations of GitHub Pages. This constraint led to a development decision to keep the number of dependencies minimal, further reinforcing the lightweight nature of the SFIA API. Each unit test was documented and included in the development process as part of continuous validation and iterative improvement. (See Appendix 3 and 4 for the full manifest and test implementation overview).

```
PASS tests/sfia-api.test.js
  SFIA API Unit Tests
    √ Should retrieve skill data successfully (76 ms)
    √ Should return 404 for invalid skill ID (22 ms)
    √ API should return response in less than 3 seconds for large dataset (13 ms)
    √ API should return valid JSON schema for skill data (71 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.758 s, estimated 1 s
Ran all test suites.
```

Figure 5: Final Unit Testing results

5.1 Results of the quiz

The results from the quiz are described in the pie chart below.

The quiz was answered by 45 first-year ICT students, which is about everyone from the two groups who participated in this questionnaire. The presentation slides that were shown to the students prior to the questionnaire itself can be found in Appendix 3. All slides were taken from the SFIA homepage, which served these slides for the explicit purpose of explaining what SFIA is and what it seeks to accomplish.

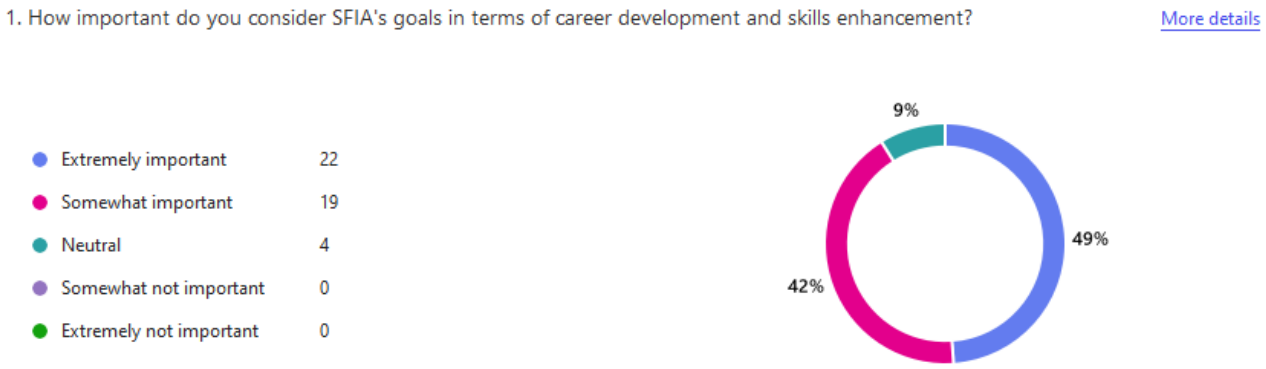

Figure 6. Question 1 answers

In figure six, we see the answers to first question, which was *"How important do you consider SFIA's goals in terms of career development and skills enhancement?".* The overwhelming majority of students answered positively.

Figure 7. Question 2 answers

In figure seven, we can see answers to question two, where question was *"How useful do you find SFIA for your learning and professional development?"* The answers were a little more toward neutrality but still maintained positivity.

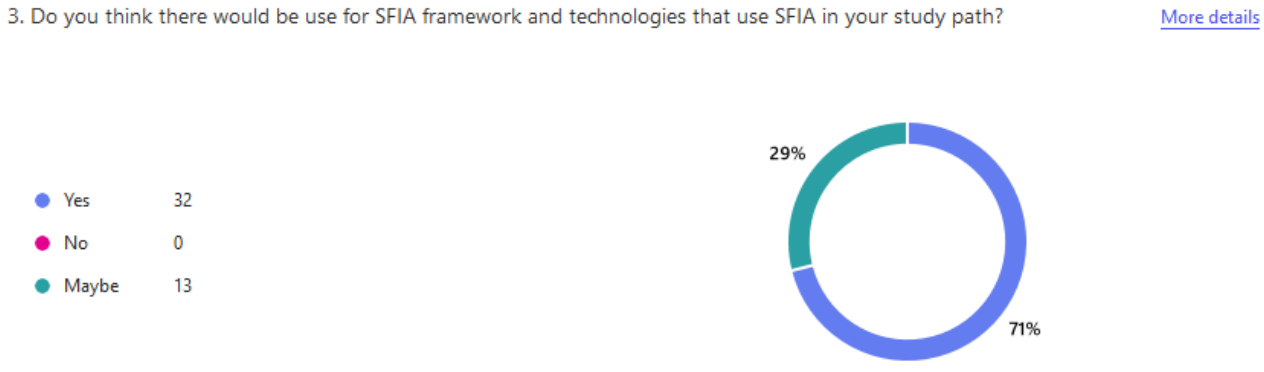

Figure 8. Question 3 answers

In question 3 as seen in Figure 8, the question “Do you think there would be use for SFIA framework and technologies that use SFIA in your study path?” was the most divisive one, where most answered yes and one third answered maybe.

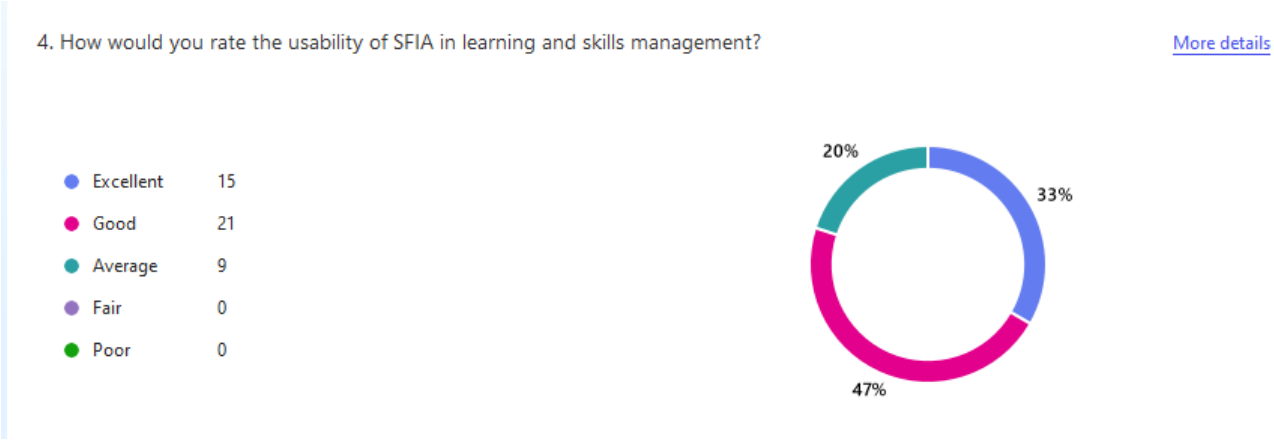

Figure 9. Question 4 answers

In figure nine we see answers for question four, which had more students aligned towards the average than in questions 1 or 2, but still only one fifth was those who answered average.

5.2 Data analysis of results

From The responses from the two groups that participated in the quiz clearly indicate a strong interest in SFIA-related technology and its application in both study path planning and career development. Not a single student selected “no” in response to question 3, which specifically assessed their interest in integrating the SFIA framework into their study paths. Furthermore, the overwhelming majority rated the usability of SFIA in learning as either good or excellent.

These findings suggest that students recognize the value of technologies designed to support study path planning and career guidance. The results also highlight a clear opportunity for further development and potential integration of the SFIA framework into the IT Institute’s educational materials, reinforcing its relevance in academic and professional growth.

6 Discussion

6.1 Main outcomes

The main results of this thesis were that the possibility of this kind of API system was deemed possible, and significant interest for SFIA Framework was found, which warrants further development for SFIA based technologies and possible implementation into JAMKs systems and structures.

6.2 Reliability and ethicality

To achieve full usability of SKILL-db, SKILL -db, the body of the SKILL -db needs to be developed to provide a path to the soft_skills section, which skills that SFIA describes as soft skills (SFIA, 2024), and also to work with the data in the database itself and fill in several missing descriptions so that SFIA-API can return a description of the same length for skills or responsibilities; currently some skills have l1_description (Level 1 Description) but not l7_description (Level 7 description) and vice versa, so if SFIA API tries to return a level 1 description for all, for example, this is not possible for every skill or responsibility because this data does not exist. Another extension to be developed would be to make the database at least bilingual, if not trilingual (Finnish, Swedish, English), so that SKILL-db could retrieve pre-translated matches of descriptions for sites that may have different language or even multilingual platforms and could with this extension make wider use of SKILL-db and SFIA API.

The third extension to be developed would be to include colour coding in the skills on the database side, so that API, which visualises them, could download their colour coding directly from the database. This would improve data presentation and make the visualizations more intuitive and clearer for users. Currently, colour coding can only be implemented through manual coding or configuration files, but moving the logic from colour coding to the database side will create a centralised solution that is easy to manage and customise.

In the database, these colour coding can be given, for example, a separate field that would contain the colour coding of each skill to which Sfia category they belong. The API could then also separately request a specific colour-code for a particular group of skills, although this is already a supported action through their category ID.

6.3 Suggestions for further development

In this work, it was determined that the desired results outlined by the customer can be technically achieved through two distinct approaches. The primary approach focuses on guiding the further development of the SFIA API, ensuring it meets the necessary requirements and functionalities. Meanwhile, the secondary, though still significant, approach involves reworking various features and mechanics of both SKILL-db and the SKILL-db API to enhance their compatibility and efficiency.

6.3.1 Further SFIA API development

If implementing plan for SFIA API, which was explained in Chapter 5, the SFIA API should be developed to include soft_skills in the future. This will be possible once a new version of SKILL-db is implemented with support for this functionality, ensuring that the API can properly utilize it. The current structure of the SFIA API should also be assessed for its adaptability to other environments, as maintaining its usability in different contexts is important for long-term sustainability. Additionally, its visual presentation should be further refined to effectively communicate the retrieved information. At the same time, ensuring API compatibility with various system requirements remains a priority. It should be evaluated how the SFIA API can be leveraged in future applications, whether in different domains or entirely new use cases, to maximize its flexibility and relevance.

From a long-term perspective, scalability and security must be key considerations. The API should be designed to handle potential expansions, accommodating large numbers of students and applications without compromising performance. Additionally, data security must be a priority, ensuring that all information passing through the API remains protected against unauthorized access and potential vulnerabilities.

6.3.2 Expanding SKILL-db API functionality

If or when developing the SKILL-db API more, which was also mentioned in Chapter 5, the functionality of the SKILL-db API must be expanded to accommodate soft skills. This enhancement should allow the API to handle requests based on a unique identifier, such as an ID or another distinguishing attribute, ensuring that only the relevant skills are retrieved. This optimization is cru-

cial, as it prevents the server from unnecessarily returning the entire skill dataset with each request, thereby improving efficiency and reducing data transfer overhead. The response from the SKILL-db API should contain essential details for each requested skill, including the skill name, an assigned color code, and descriptions tailored to proficiency levels ranging from 1 to 9. Additionally, a general description should be included to provide overarching context for each skill. This structured dataset is then utilized by the Web Service layer, where it can be processed as needed. Specifically, the color code can be employed to visually categorize skills, allowing the front-end interface to clearly indicate which category a particular skill belongs to.

A significant advantage of enabling targeted API requests is the reduction in bandwidth usage. Currently, when retrieving skill data for specific roles, the SFIA API fetches the entire skill database during each call, which is inefficient. By implementing more refined queries within SKILL-db, only the necessary data for the requested roles will be transmitted, thereby optimizing resource utilization and improving response times. Furthermore, SKILL-db must undergo modifications to align with the latest version of the SFIA framework, transitioning from SFIA 8 to SFIA 9. This update necessitates not only structural adjustments but also the completion of skill descriptions for all entries to ensure consistency and completeness in the dataset. These improvements will enhance the accuracy and usability of the system, ultimately leading to a more streamlined and effective skill management process.

6.4 Discussion

This study investigated, developed and monitored the development of a lightweight, easy-to-integrate API solution that aims to visualize database data interactively and efficiently. The development work focused on keeping the API simple in structure and aimed at miniaturizing the use of external packages, as in some environments this is not feasible. The results show that it is possible to create a scalable and adaptive solution that meets these requirements. This work provides a basis for further development of the API, in relation to improving the compatibility of the environment and visual features.

Already existing tools and systems should be developed further, as detailed in chapter

Great interest in SFIA Framework was distinguished and students felt that such tools as SFIA could be useful in their study paths and researching their career paths, and as such teachers at JAMK should research more if and how SFIA could be more involved in study materials and how the skills SFIA describes be used in reflecting real world understanding and competencies in JAMK courses.

Sources

Dillman, D. A., Smyth, J. D., & Christian, L. M. (2014). Internet, Phone, Mail, and Mixed-Mode Surveys: The Tailored Design Method (4th ed.). Wiley.

Docker. (2024). Enterprise Application Container Platform | Docker. Docker. https://www.docker.com/

Healy, K. (2018). Data Visualization. Google Books. https://books.google.com/books/about/Data_Visualization.html?id=3XOYDwAAQBAJ

MashaMSFT. (2024). Educational SQL resources - SQL Server. Microsoft.com. https://learn.microsoft.com/en-us/sql/sql-server/educational-sql-resources?view=sql-server-ver16

Moving to SFIA 9. (2024). SFIA. https://sfia-online.org/en/sfia-9/moving-to-sfia-9

Oppenheim, A. N. (2000). Questionnaire design, interviewing, and attitude measurement. Continuum.

Richardson, L., Amundsen, M., & Ruby, S. (2013, September). RESTful Web APIs. Www.oreilly.com. https://www.oreilly.com/library/view/restful-web-apis/9781449359713/

SFIA. (2024). The global skills and competency framework for a digital world. SFIA. https://sfia-online.org/en

SFIA 9 full framework view. (2024). SFIA. https://sfia-online.org/en/sfia-9/sfia-views/full-framework-view

Säännöt ja periaatteet. (2025). Jyväskylän Ammattikorkeakoulu - JAMK. https://www.jamk.fi/fi/opiskelijalle/tutkinto-opiskelija/saannot-ja-periaatteet

WIMMA Capstone. (2025). Jamk.fi. https://wimma-capstone.jamk.fi/

WIMMA Lab. (2023). SKILL-db Documentation. WIMMA Lab. https://wimma-lab-2023.pages.labra-net.jamk.fi/iotitude/core-skill-db

WIMMA Lab. (2023). Wimmalab.org. https://www.wimmalab.org/

Appendices

Appendix 1. SKILL-db PostgreSQL framework

Table: sfia8_skills		
Column Name	**Data Type**	**Description**
id	character(4)	Unique identifier
title	character varying(64)	Title of the skill
category	character varying(64)	Category of the skill
subcategory	character varying(64)	Subcategory of the skill
gen_descrip-tion	character varying(1024)	General description of the skill
l1_description	character varying(1024)	Level 1 description of the skill
l2_description	character varying(1024)	Level 2 description of the skill
l3_description	character varying(1024)	Level 3 description of the skill
l4_description	character varying(1024)	Level 4 description of the skill
l5_description	character varying(1024)	Level 5 description of the skill
l6_description	character varying(1024)	Level 6 description of the skill
l7_description	character varying(1024)	Level 7 description of the skill

Table: soft_skills		
Column Name	**Data Type**	**Description**
id	character(4)	Unique identifier
title	character varying(64)	Title of the soft skill
category	character varying(64)	Category of the soft skill
gen_descrip-tion	character varying(1024)	General description of the soft skill

Table: sfia8_responsibilities		
Column Name	**Data Type**	**Description**
id	character(4)	Unique identifier
title	character varying(16)	Title of the responsibility
l1_description	character varying(2048)	Level 1 description of the responsibility
l2_description	character varying(2048)	Level 2 description of the responsibility
l3_description	character varying(2048)	Level 3 description of the responsibility
l4_description	character varying(2048)	Level 4 description of the responsibility
l5_description	character varying(2048)	Level 5 description of the responsibility
l6_description	character varying(2048)	Level 6 description of the responsibility
l7_description	character varying(2048)	Level 7 description of the responsibility

Appendix 2: Presentation slides

Prior the quiz a short presentation was held by the thesis author to reach out to the students about what SFIA is, what its goals are and what how it works, so they could answer the quiz knowing what SFIA is about.

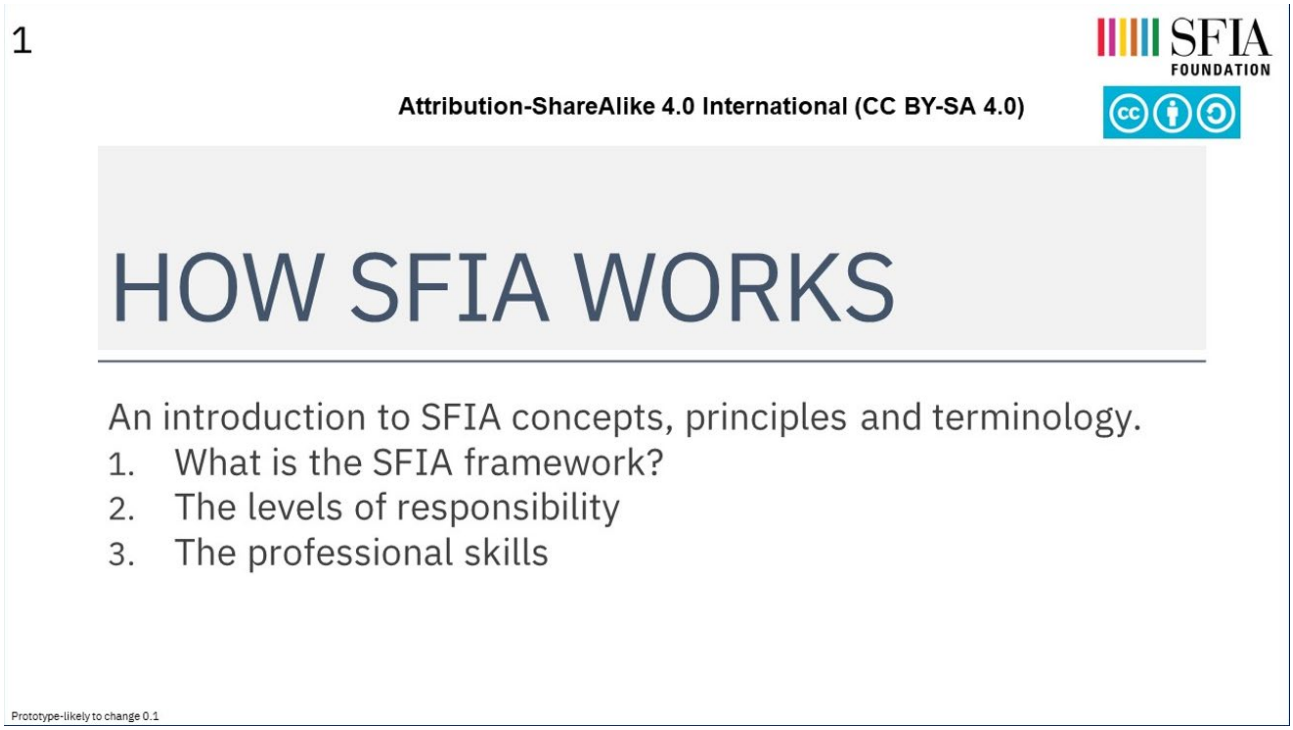

Figure 10: Presentation slide 1

Figure 11: Presentation slide 2

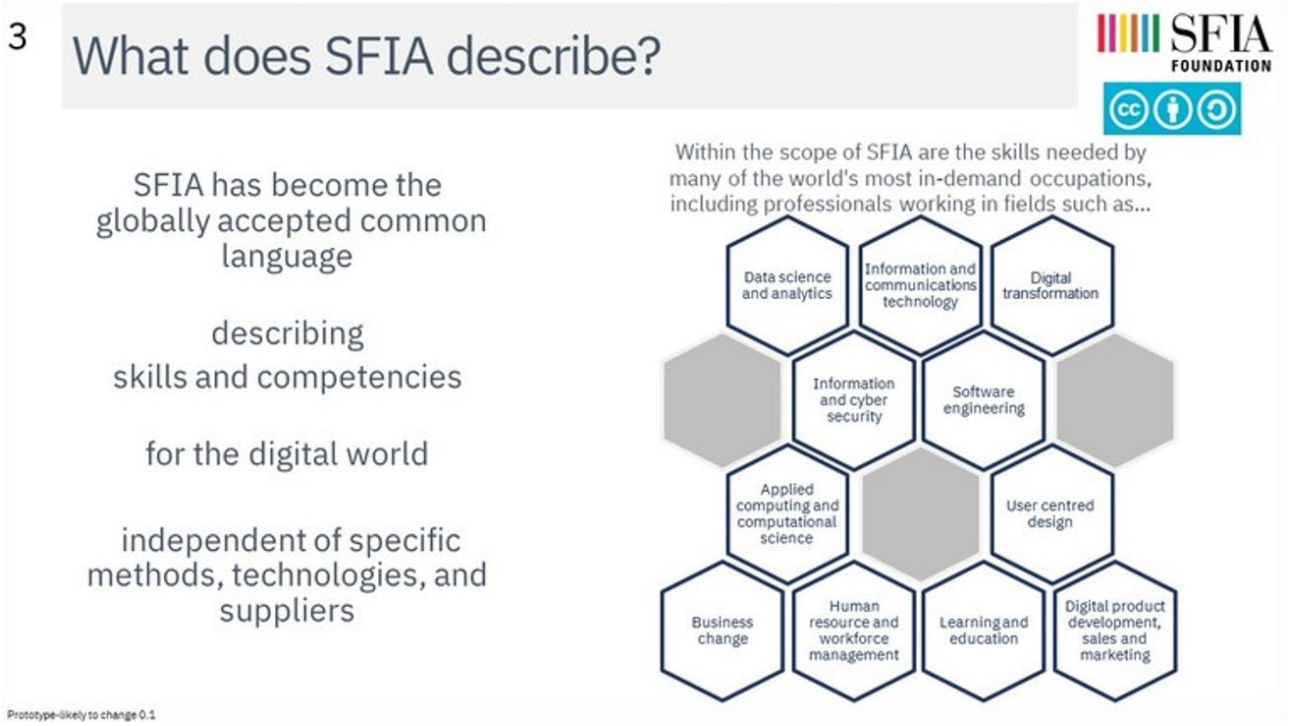

Figure 12: Presentation slide 3

Figure 13: Presentation slide 4

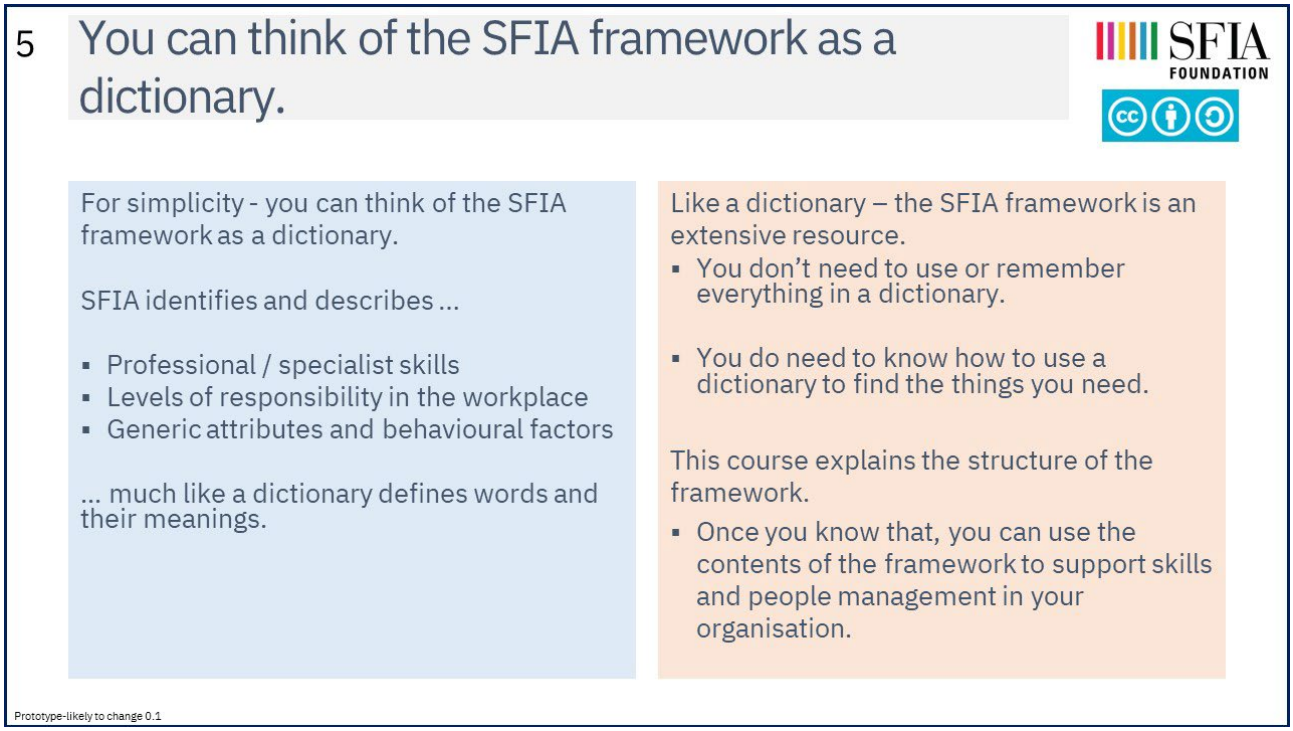

Figure 14: Presentation Slide 5

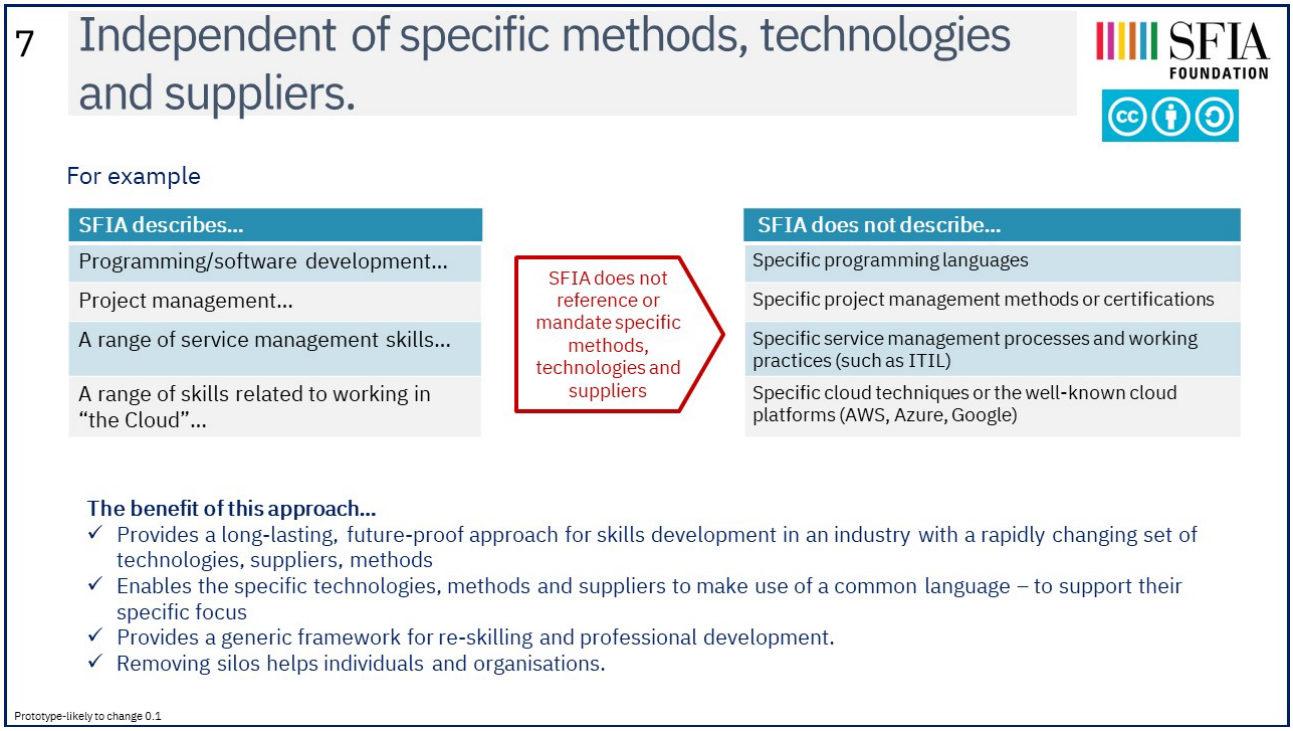

Figure 15: Presentation Slide 6

Figure 16: Presentation Slide 7

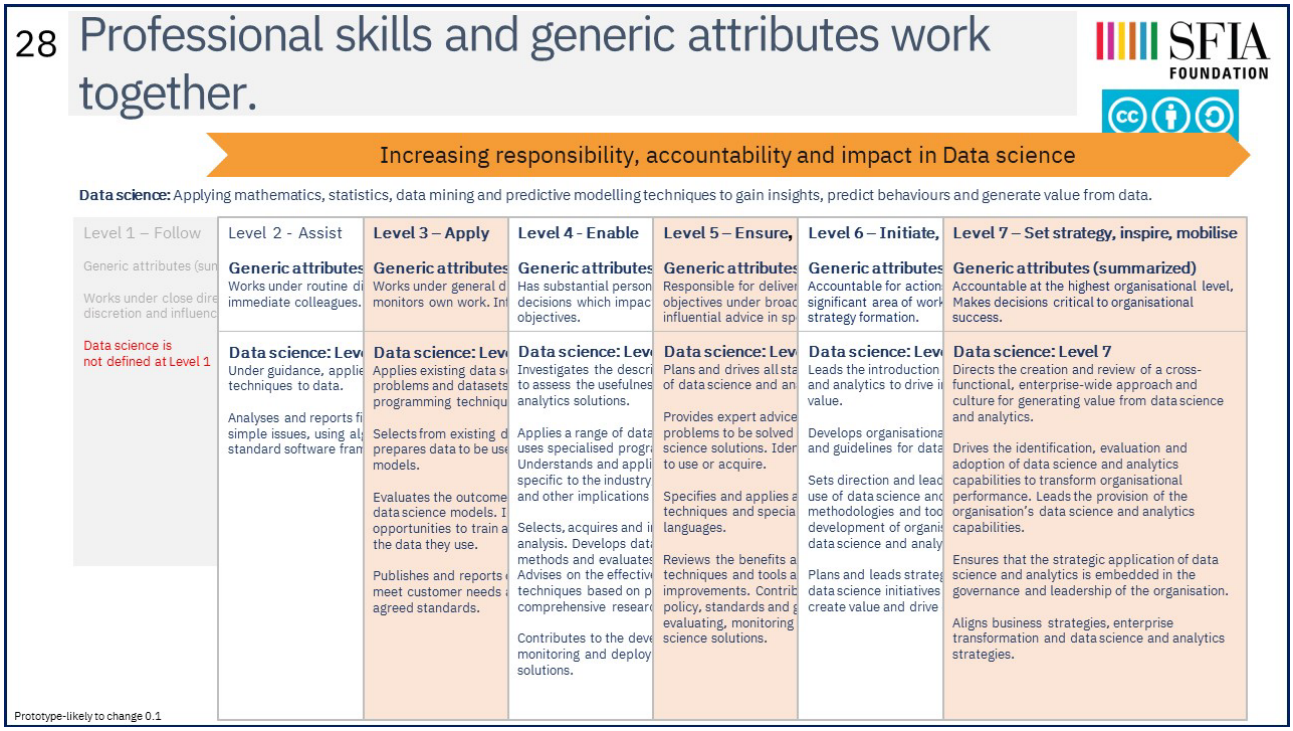

Figure 17: Presentation Slide 8

These slides that I used for these two presentations were freely given by SFIA Foundation at their homepage, and I took the most important ones to quickly teach the classes in question the best I could, so the students can answer the quiz.

Appendix 3: Unit testing manifest

Unit Testing Manifest for SFIA API

Test Suite Name: SFIA API Unit Tests

Test Framework: Jest, Supertest

Test Scope: To verify the correct functionality, performance, and error handling of the SFIA API endpoints

Unit tests will be executed using **Jest** and Supertest. Automated tests will run after each update to ensure that all parts of the SFIA API are functioning as expected, improving the overall reliability of the system.

Test 1: Verify Successful Retrieval of Skill Data

Test Name: Should retrieve skill data successfully

Test Description: Ensure that the API successfully retrieves data for a specific skill based on its identifier.

Test Steps:

- Send a GET request to the /skills/{skill_id} endpoint.

Check if the response status is 200 OK.

- Validate that the response body contains the skill's correct details (e.g., name, description, associated color coding).

Expected Result:

- The API should return a 200 OK response.
- The response body should match the expected data structure for the given skill ID.

Test 2: Validate Error Handling for Invalid Skill ID

Test Name: Should return 404 for invalid skill ID

Test Description: Test how the API handles requests for non-existent or invalid skill IDs.

Test Steps:

- Send a GET request to the /skills/{invalid_skill_id} endpoint (use an invalid skill ID). Ensure the response status is 404 Not Found.
- Check if the response body contains an error message indicating that the skill was not found.

Expected Result:

- The API should return a 404 Not Found response.
- The response should contain an error message "Skill not found".

Test 3: Check Response Time for Large Data Sets

Test Name: API should return response in less than 3 seconds for large dataset

Test Description: Verify that the API returns responses in a reasonable amount of time when querying a large dataset.

Test Steps:

- Send a GET request to the /skills endpoint to retrieve all skills (or a large set of data).
- Record the response time.
- Verify that the response time is within acceptable limits (e.g., less than 3 seconds for a large dataset).

Expected Result:

- The API should respond within the set performance threshold, indicating that it can efficiently handle large datasets.

Test 4: Validate Data Integrity and Format (JSON Schema Validation)

Test Name: API should return valid JSON schema for skill data

Test Description: Ensure that the data returned by the API adheres to the correct JSON schema format.

Test Steps:

- Send a GET request to the /skills/{skill_id} endpoint.
- Validate the response body against a predefined JSON schema for the skill data.

Expected Result:

- The API response should conform to the predefined JSON schema, ensuring that the data structure is consistent and correct.

Appendix 4: Unit testing code

```
const request = require('supertest');
const app = require('../server');

describe('SFIA API Unit Tests', () => {

  // Test 1: Verify Successful Retrieval of Skill Data
  test('Should retrieve skill data successfully', async () => {
    const skillId = 'stpl';
    const response = await request(app).get(`/skills/${skillId}`);

    expect(response.status).toBe(200);
    expect(response.body).toHaveProperty('id', skillId);
    expect(response.body).toHaveProperty('title');
    expect(response.body).toHaveProperty('category');
  });

    // Test 2: Validate Error Handling for Invalid Skill ID
    test('Should return 404 for invalid skill ID', async () => {
      const invalidSkillId = 'ffff';
      const response = await request(app).get(`/skills/${invalidSkillId}`);

      expect(response.status).toBe(404);
      expect(response.body).toHaveProperty('message', 'Skill not found')
    });

    // Test 3: Check Response Time for Large Data Sets
    test('API should return response in less than 3 seconds for large dataset',
async () => {
      const start = Date.now();
      const response = await request(app).get('/skills');
      const end = Date.now();

      expect(response.status).toBe(200);
      expect(end - start).toBeLessThan(3000); // result is in ms
    });

    // Test 4: Validate Data Integrity and Format (JSON Schema Validation)
    test('API should return valid JSON schema for skill data', async () => {
      const skillId = 'itsp';
      const response = await request(app).get(`/skills/${skillId}`);
      expect(response.status).toBe(200);
```

```
      const skillSchema = {
        type: 'object',
        properties: {
          id: { type: 'string' },
          title: { type: 'string' },
          category: { type: 'string' },
        },
        required: ['id', 'title', 'category']
      };

      const ajv = require('ajv');
      const validate = new ajv().compile(skillSchema);
      const isValid = validate(response.body);
      expect(isValid).toBe(true);
    });
});
```