WEB SYSTEM WITH GRAPH UI FOR EXPLORATORY LEARNING

ВЕБ-СИСТЕМА З ГРАФОВИМ ІНТЕРФЕЙСОМ КОРИСТУВАЧА ДЛЯ ДОСЛІДНИЦЬКОГО НАВЧАННЯ

by Andrii Tytenko

Presented in Partial Fulfillment of the Requirements for the Degree

Master of Software Engineering

American University Kyiv

2024

APPROVED BY:

Sergiy Tytenko, Ph.D., Faculty Mentor

Abstract

This capstone project report details the development of a Graph User Interface application for Exploratory Learning. The report focuses first on the background overview of literature about graphs, exploratory learning, and existing web systems.

Then it describes the development of a web-based graph visualization system where it highlights the selection of technologies like Next.js, TypeScript, and Tailwind CSS, and delves into the challenges encountered and UI features. The choice of specific library for graph visualization is emphasized for its performance and customization capabilities.

The report also addresses user interaction, particularly the implementation of keyboard navigation to improve accessibility and user experience. It acknowledges the system's limitations, such as handling large data sets and extends into recommendations for future enhancements, including interface optimization and customization.

Contents

Chapter 1. Introduction

The domain of digital education is continuously evolving, offering new platforms and tools to enhance the learning experience. The master's capstone project presented in this report introduces a modern approach to exploratory learning through the development of a web system integrated with a Graph User Interface. This project, developed as part of the master's program in Software Engineering, emphasizes the importance of intuitive design and user interaction in educational technology in the field of human-computer interfaces.. The web system was conceptualized and created in 2023-2024, with a focus on providing a dynamic and interactive learning environment for users.

1.1 Overview

Educational institutions and educators globally are increasingly adopting digital tools to facilitate effective learning. In this context, the development of a web system with a Graph UI aligns with the current trends in educational technology. The project team embarked on this venture to address the specific needs of exploratory learning. The work allowed to bring the team's unique skills and insights, contributing to the complex nature of the project.

The core of this project revolves around the creation of a web-based platform that leverages graphical interfaces to simplify complex concepts and promote interactive learning. The system is designed to be accessible and easy-to-use. This thesis documents the entire process of developing the web system, from initial conception to final implementation, including the challenges faced and the innovative solutions employed.

As the author of this paper, my role was to delve into the intricacies of Graph UI design, exploring various approaches to enhance user interaction and learning outcomes.

The significance of this project extends beyond its immediate educational applications. It may serve as an example for future development in the field of educational technology, particularly in the integration of graphical interfaces in

web-based learning environments. The documentation and analysis presented in this report aim to provide valuable insights and guidelines for similar projects in the future.

Furthermore, the project's emphasis on user-centered design and interactive learning modules may set a precedent for future educational tools. The adoption of Graph UI in the web system demonstrates the potential of graphical interfaces in simplifying complex information and making learning more accessible and engaging. This report explores the various aspects of this integration, trying to assess its possible impact on the learning process and user engagement.

1.2 Report Organisation

The report is organized into four comprehensive chapters, followed by a bibliography, each addressing a critical aspect of the Graph UI application project:

Chapter 1: Introduction. This chapter introduces the project, offering an overview and laying out the foundational aspects of the research. It covers the initial conceptualization, the motivation behind choosing this specific area of study, and the overarching goals the project aims to achieve.

Chapter 2: Background Overview. Diving into the essential background, this chapter provides the necessary context and theoretical knowledge crucial for understanding the project. It encompasses an exploration of exploratory learning, the significance of graphs in information visualization, the role of graph user interfaces in learning systems, and the inherent challenges of designing effective graph UIs.

Chapter 3: Design and Development of the Graph UI Web System for Exploratory Learning. The heart of the report, this chapter details the entire process of designing and developing the web system. It chronicles the journey from the initial conceptual framework, through the nuances of system requirements and technical specifications, to the selection and implementation of the technology stack. Further, it delves into the API considerations, the choice and application of the graph library, the architectural decisions, the intricacies of UI and graphical elements, the user flow design, and the integration of user interaction features.

Chapter 4: Conclusions. Concluding the report, this chapter synthesizes the key findings, challenges encountered, and limitations recognized during the project. It not only provides a comprehensive results list but also looks forward, offering

recommendations for future development and final thoughts that reflect on the project's implications, learnings, and potential impact.

Bibliography. The report culminates with a bibliography, listing all the academic and technical references utilized throughout the course of the project.

Chapter 2. Background Overview

Exploratory learning, driven by our natural curiosity, grows in a changing world. As said in [1], exploratory learning often leads to unexpected discoveries. In today's digital world, Graph User Interfaces (Graph UI) have become a key tool, linking interactive, graph-like data presentations with self-driven learning. By changing complex data into visual, easy-to-follow structures, Graph UI boosts exploratory learning, creating a space where learners can easily dive into connected information [16]. This overview starts from the basics of Graph UI to its big impact on how we learn, highlighting the promise and the urgent need to dig deeper to unlock its full potential for a strong, insightful learning experience.

2.1 Exploratory Learning

Exploratory learning is an essential approach for humans for learning new concepts. In terms of human-computer interaction, exploratory learning is a common approach for learning the systems as it is natural for human beings to explore something and try to understand concepts usually on their own [2]. While the process can be supervised, it often unfolds as an independent journey of discovery, making it a preferred method for learning in various technological contexts. The intrinsic nature of exploratory learning in HCI facilitates a deeper engagement with the system, as individuals are not merely following instructions but are actively involved in the learning process, experimenting, and discovering functionalities on their own. This active involvement often leads to a more profound and personal understanding of the system [16].

In data analysis, exploratory data analysis would be an important phase of the conducted research, as it allows us to start to build hypotheses and get familiar with the data specifics [4]. At that phase, the primary goal is to explore, investigate, and learn from the data set, rather than immediately attempting to confirm specific hypotheses [31]. This approach is inherently investigative and akin to detective work. It involves using summary statistics and graphical tools to familiarize oneself with the

data, identify potential relationships between variables, spot anomalies like outliers, and form initial hypotheses.

The effectiveness of exploratory learning is significantly enhanced by the visual representation of information [3]. Visual tools facilitate a more intuitive understanding of complex data sets and relationships. Graphs, charts, and other visual aids make abstract data more concrete, allowing learners to quickly grasp patterns, trends, and anomalies that might be less apparent in textual or numerical formats. This emphasis on visual representation in exploratory data analysis reflects a broader principle in exploratory learning across various fields: the power of visuals in aiding human understanding and memory. By engaging multiple senses and providing a more holistic view of the information, visual tools make exploratory learning a more immersive and effective experience.

2.2 Graphs and Information Visualization

The concept of information being either structured or unstructured is a fundamental aspect of data management and analysis. Structured information, which is often the focus when learning a new subject, is characterized by its high level of organization and format consistency, making it easier to analyze and visually represent. This contrasts with unstructured information, which is more amorphous and less easily categorized, such as text, images, or video content. [5].

When delving into a new topic, structured information becomes invaluable due to its clarity and ease of manipulation. One of the most effective ways to visually represent this type of information is through graphs [5]. Graphs, in this context, are not just charts or plots, but structured representations of data that consist of nodes (or vertices) and the relationships (or edges) between them. They are particularly useful in illustrating relationships and hierarchies in data, making complex information more digestible [16].

Several common types of graphs include trees, organizational charts, data flow diagrams, and entity-relationship diagrams [5]. Trees, for instance, are a type of graph where each node is connected to others in a hierarchical manner, with no cycles, making them ideal for representing structures like family trees or organizational hierarchies (as shown in Figure 1). Organizational charts are another form of graph, used to illustrate the structure of an organization, showing relationships between different positions or departments. Data flow diagrams depict the flow of information within a system, highlighting how data is processed and transferred between different

entities. Lastly, entity-relationship diagrams are used in database design to show the relationships between different data entities.

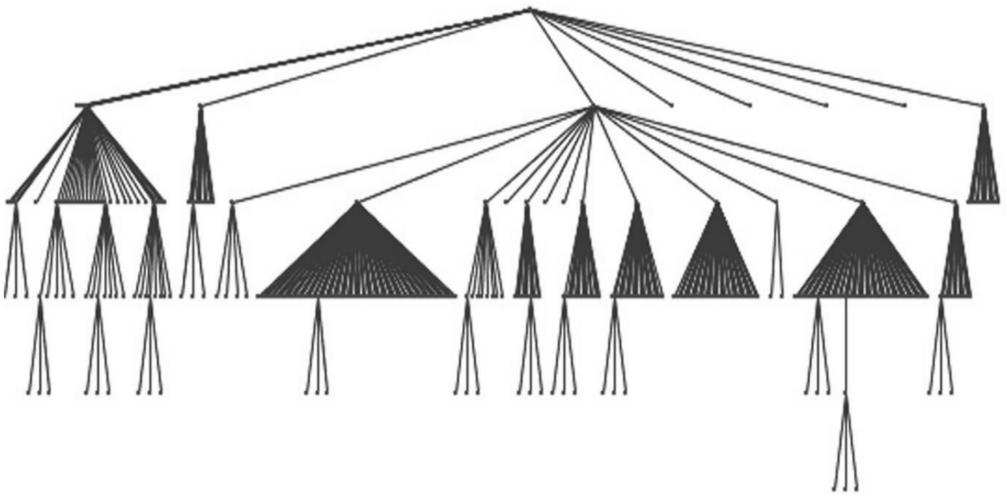

Figure 1: A tree layout for a moderately large graph [5]

The fundamental challenge of graph drawing involves determining node positions and edge curves given a set of nodes and their relations (or edges). This issue has persistently existed because graph definitions often rely on their drawings, as demonstrated by Euler's 1736 resolution of the “Königsberg Bridge Problem” using a visual representation [5]. The problem revolved around the city of Königsberg in Prussia, where seven bridges connected various land masses [6]. The challenge was to devise a walk through the city that would cross each bridge once and only once. Euler proved that such a walk was impossible, introducing the concept of an Eulerian path in the process. His innovative approach, which involved representing the land masses as nodes and the bridges as edges between them, created a new, abstracted way of thinking about the problem and marked the genesis of graph theory [7]. In fact, the solution to the problem did not rely on bridges, islands, or people, as well as the whole geography of the city didn’t influence the solutions, making the only thing that was important the connection itself [8].

Using Euler's ideas, mathematicians have looked at many ways to show connections between points, or nodes, with links, called relationships. Some allow the links to have a start and end point, while others just connect points without direction. Some types, like hypergraphs, let links join several points at once. However, there’s no single good solution for all the cases of how the structured information can be represented in graphs, so different models have to be used [8].

The research compiled by Battista et al. addresses the complexities in defining effective graph drawings, highlighting the importance of establishing properties and classification criteria for different graph types [9]. They explore a range of graph-drawing algorithms, each with unique computational complexities and suited to various graph types. These algorithms are crucial for handling large-scale graphs and adhering to specific aesthetic criteria, which are essential for making graphs readable and intuitive. Additionally, their work delves into various heuristics and optimizations to enhance graph drawings. These methods are important in managing the trade-offs between computational efficiency and the quality of the visual representation.

A key insight from Battista et al. is the role of user studies in determining what constitutes a "good" graph drawing. They suggest that understanding which layouts are more intuitive or readable for humans is critical [9]. This approach emphasizes the importance of user-centered design in graph drawing, ensuring that the visualizations are not only theoretically sound but also practically useful and accessible to the intended audience.

Graphs have a wide range of applications across multiple domains such as computer science, bioinformatics, social network analysis, and education. Their versatility is demonstrated in practical applications like concept maps. Concept maps are visual tools designed to organize and illustrate the relationships between different ideas. In these maps, two concepts are connected by a line, and words placed on this line serve as connection phrases, explicitly defining the nature of the relationship between the connected concepts.

This approach allows for a clear and structured visualization of complex relationships and hierarchies within a given domain, making it easier for users to understand and analyze them. In education, for instance, concept maps are used to help students grasp the connections between different concepts, fostering a deeper understanding of the subject matter. In fields like bioinformatics or social network analysis, they can illustrate intricate networks of interactions, providing valuable insights into the underlying structures of biological systems or social connections [10].

The development of concept maps originated from a fundamental need in educational psychology to establish a framework for meaningful learning. This concept was first described by David Ausubel, an educational psychologist. Ausubel's work emphasized the importance of understanding the relationships and connections between concepts in learning, rather than merely acquiring discrete facts. Concept maps became a tool to visually represent and organize knowledge, illustrating how different concepts are interconnected and building a more holistic understanding of a subject area.

This approach aligns with Ausubel's theory, which advocates for integrative learning, where new knowledge is assimilated into existing cognitive frameworks, thereby fostering deeper and more meaningful learning experiences [11]. It emphasizes the concept that effective learning involves not just the acquisition of new facts or data, but more importantly, the integration of this new information into the learner's existing knowledge base. This integration facilitates a deeper understanding and retention of new concepts, as it allows learners to build upon what they already know, creating a more cohesive and comprehensive understanding of the subject matter. This approach is fundamental in educational strategies that aim for meaningful and lasting learning outcomes [12].

Another similar visual tool that can be used in learning is a mind map. It centers around a key concept and radiates outwards like a tree. Mind maps typically display a clear hierarchy from the central idea, branching out to cover related topics or subtopics. They can also include additional links among various elements, allowing for the representation of more complex relationships. The radial layout of mind maps is particularly effective for showing how information is structured, making them a popular choice for organizing and visualizing ideas and concepts [12].

2.3 Graph User Interfaces and Learning Systems

The rise of accessible digital devices empowers users to interact with data in new ways, enabling a many user interfaces (UI) to facilitate unique experiences. A distinctive approach to this endeavor is represented by Graph User Interfaces (Graph UI), where information is modeled and interacted with through graph-structured representations. As was discussed earlier, graph UI embodies a visualization method where data points (nodes) and their interrelationships (edges) are visibly represented, offering users an interface to navigate, manipulate, and explore interconnected information, thus facilitating exploratory learning.

There are experiments providing evidence that information presented with the graphs can enhance cognitive engagements. Puntambekar et al. examined the navigation patterns of students and discovered that the maps assisted students in maintaining focus on their objectives, compared to a version of the system that used an index [13]. Feedback from students via an attitude survey revealed that they perceived the maps as beneficial for locating information pertinent to their objectives. The concept maps acted as a visual aid, facilitating students in making logical transitions between concepts. The map's connections and visualization of concepts pointed towards

additional information relevant to their ongoing reading, and students appeared to exhibit a positive attitude towards these elements.

There have been approaches to constitute a system that could build graphs specifically for educational purposes. It brings its own unique challenges which have to be solved and connected with ontological modeling. Chen et. al. [14]. discesses such systems as well as describe practical approaches for it. Their system, developed in response to the growing demands in the educational sector, automatically constructs educational knowledge graphs utilizing heterogeneous data like pedagogical and learning assessment data. Through neural sequence labeling algorithms, it extracts educational concepts from pedagogical data and employs probabilistic association rule mining on assessment data to ascertain crucial educational relations between concepts.

Another approach is discussed in [15] where it has been highlighted the significance and methodology of employing graphical visualizations, specifically concept maps, in the context of ontologically oriented information and educational web systems. In an effort to enhance the educational process and knowledge acquisition, the article underscores the need to integrate visualizations like interactive concept maps, which depict the structure of concepts and their interconnections into educational web environments. These maps, based on didactic ontology and a concept-thesis model, not only aim to improve navigation and facilitate a better understanding of subject areas but also to boost user engagement and learning effectiveness by presenting data in a visually intuitive manner [15].

The main goal is to develop a method for the automated construction of interactive concept maps within web systems and to enhance the functionality of the information and educational web environment, so a Graph UI will be created at the end [15]. The approach involves formalizing content based on the concept-thesis model, analyzing types of conceptual maps, and adapting models and algorithms according to identified needs and shortcomings. The system encapsulates server modules and client components for generating graph data and interfacing components, while also allowing for user interaction and convenient data display [15].

While the method presented promises clarity and interactivity in presenting educational content, ongoing research and development in the system seek to further optimize algorithms and minimize server and client load when building and utilizing interactive concept maps [15]. The automated creation of these knowledge graphs might enhance online education platforms by providing personalized teaching and adaptive learning solutions.

Exploratory learning greatly benefits from Graph UI advancements, which turn complex data into visual, interactive models, fueling curiosity and innovative problem-solving. Many researchers have explored and explained the benefits that graphs can provide [15]. These aren't merely visual tools but catalysts for cognitive engagement, creating an intuitive and efficient learning environment. Automating educational knowledge graph creation may significantly boost learning engagement and innovative application of information. While its integration with educational tech continues evolving, future innovations are poised to further enrich interactive learning, with a multidisciplinary approach ensuring it meets both tech and pedagogical demands [16].

The discussed case studies underline the transformative potential of Graph UI on educational technologies, particularly in automating the creation of educational knowledge graphs, thereby enhancing exploratory learning. This shift not only reshapes how information is absorbed and engaged with but also how it's innovatively applied. The process of refining Graph UI and aligning it with exploratory learning is ongoing. The automated creation of these knowledge graphs could potentially improve the process of learning in digital environments.

2.4 Challenges of Graph User Interfaces

Graph User Interface (Graph UI) design is a critical aspect of data visualization, involving complex challenges that span across various dimensions of technology and design. These challenges include optimizing user interfaces, and ensuring efficient performance, especially with large-scale graphs.

In Graph User Interface (Graph UI) design, optimizing user interfaces involves distinguishing between User Experience (UX) and User Interface (UI) to understand and deliver what users want, through intuitive interactions and appealing styling [17]. Effective graph visualization starts with a well-structured data model, consisting of nodes, links, and properties, and a visual model tailored to user needs. It's important to maintain consistency and intuition in the design, while avoiding overwhelming users with too much information. Selecting the right color palette also plays a key role in enhancing user experience [17].

For performance with big graphs, an effective graph visualization engine is vital [18]. Furthermore, providing users with decluttering tools such as filters and pruning options, and utilizing automated layouts, are essential for revealing insights and

patterns in large datasets. In cases of extremely large graphs, adaptive styling ensures clarity and usability at various scales. These integrated approaches in Graph UI design collectively address the challenges, ensuring a balance between technical solutions and design strategies for an effective, efficient, and user-friendly experience.

2.5 Chapter Summary

This chapter is the groundwork for understanding the vital role of exploratory learning and Graph User Interfaces (Graph UI) in educational technology, setting the stage for the next phase of this report: the Design and Development of the Graph UI Web System for Exploratory Learning. The insights collected here form a crucial foundation for the technical decisions and implementation strategies that will be detailed in the next chapter. It also portrays the actuality of the topic by highlighting the rising need in the digital learning systems with effective User Interfaces.

It was discussed how exploratory learning, fueled by natural curiosity, leads to deeper engagement and unexpected discoveries. This understanding will directly influence the design of the web system, where fostering user curiosity and enabling self-driven exploration will be central objectives. The upcoming chapters will demonstrate how this theoretical basis is translated into practical design elements that encourage exploration and discovery within the UI.

The discussion on the efficacy of visual tools in enhancing understanding explains the importance of intuitive and visually appealing Graph UIs. This will be reflected in the design choices made in the web system, where emphasis will be placed on creating clear, engaging, and easy-to-navigate graph structures. The system aims to transform complex data into visual formats that not only simplify information but also make it more accessible and engaging for learners.

Moreover, the challenges identified in Graph UI design, such as ensuring efficient performance with large-scale graphs and optimizing user interfaces, will be addressed in the development process. The next chapters will detail how these challenges are tackled through technical solutions and design strategies, ensuring a balance between functionality, aesthetics, and user experience.

The upcoming report chapters will also provide a comprehensive look at the creation of the Graph UI web system, highlighting how the theoretical foundations discussed in this chapter are brought to life. The project aim is to create a system that not only

aligns with the educational objectives set out in this chapter but also overcomes the practical challenges in Graph UI design.

Chapter 3. Design and Development of the Graph UI Web System for Exploratory Learning

Designing and developing a web system with a Graph User Interface for exploratory learning presents a unique set of challenges and opportunities, especially in the rapidly evolving domain of educational technology. The described project, realized through a Next.js application, focuses on leveraging the power of graph-based visualizations to enhance the learning experience.

3.1 Conceptual Framework of the Web System with Graph User Interface

The development of the Graph UI Web System requires a well-defined conceptual framework that integrates insights from the field of educational technology and the specific needs identified in the project's background overview. This framework is not merely a continuation of existing web-based educational systems; it represents a leap in how graph-based user interfaces are applied in exploratory learning contexts.

Central to this framework is the recognition of the limitations inherent in some current educational tools, particularly their lack of dynamic, interactive, yet simple and straight-forward learning experiences. This gap in educational technology is what the project seeks to address.

Drawing from a comprehensive review of educational technology literature, the project acknowledges the potential of graph-based interfaces. While many existing web systems utilize graphical elements, few harness them as the primary medium for delivering educational content. This project takes inspiration from such systems but moves towards a more focused use of Graph UI to facilitate an interactive and

immersive learning environment, which could provide users benefit in the context of the exploratory learning.

The system's design should revolve around the dual principles of simplicity and effectiveness. It will be built to cater to a diverse user base, including those without advanced technical skills, but eager to engage in learning through intuitive visualizations. This inclusivity should be central to the system's conceptual design, ensuring that it is not only user-friendly but also capable of delivering complex information in an accessible format.

The conceptual framework for the Graph UI Web System is a thoughtful combination of educational theory, user-centric design principles, and innovative application of graph-based interfaces. It sets the stage for the detailed system requirements and specifications outlined in the next subchapter, providing a clear roadmap for the development of a web system that is both technologically advanced and pedagogically sound.

3.2 Web System Requirements and Specifications

Category	**Description**
Functional Requirements	● Advanced interactivity and graph visualization. ● Dynamic user interaction with graphs. ● Enhanced exploratory learning experience.
Non-Functional Requirements	● High performance and quick load times. ● Reliability and continuous availability. ● Seamless user experience across platforms and devices.
UI Design Principles	● Clear presentation of complex information. ● Minimal distraction, focus on key elements. ● Easy navigation and comprehension.
Aesthetic and Usability Focus	● Color schemes and typography for readability. ● Intuitive navigation structures. ● Balance of visual appeal and functional simplicity.
User Interaction Emphasis	● Intuitive design for exploration and interaction. ● User-centric navigation and interaction patterns. ● Fluid learning journey through graphical data.

Design Philosophy	● Minimalism to reduce cognitive overload. ● Focus on learning aspects. ● Great user experience.
Strategic Feature Omissions	● Omission of features like SEO optimization. ● Focus on key functionalities. ● Enhanced educational value without interface complexity.

Table 1. System Requirements and Specifications Overview

Table 1 presents a well-structured and detailed breakdown of the project's core components. It's organized into specific categories, each one offering a snapshot of different aspects through concise bullet points. This format makes the information accessible and easily digestible.

In the Functional Requirements category, the focus is placed on the application's interactivity and robust graph visualization capabilities, with a special emphasis on how these features enhance the exploratory learning experience. This section underscores the application's primary function as an educational tool, designed to engage users interactively with complex data.

The Non-Functional Requirements are equally critical, highlighting the importance of performance aspects such as speed, reliability, and the provision of a seamless experience across various devices. This indicates a commitment to creating an application that is not only functionally robust but also consistently reliable.

The UI Design Principles category delves into the application's approach to making complex information accessible and engaging. This section emphasizes the importance of a clear and distraction-free layout, easy navigation, and the overall comprehension of the interface, reflecting a user-centric design philosophy.

In terms of Aesthetic and Usability Focus, the table points out the strategic selection of visual elements like color schemes and typography. This careful consideration ensures that the application is not only visually appealing but also enhances readability and user comfort, striking a balance between aesthetics and functionality.

The User Interaction Emphasis highlights the intuitive design. This approach is key to providing a fluid and engaging user experience, enabling users to navigate and interact with the graphical data in a natural and instinctive manner.

The Design Philosophy of the project is rooted in minimalism. This approach aims to improve the user experience and reduce cognitive overload, allowing users to focus

more on the learning aspect of the application. It’s a strategic decision to keep the interface simple yet effective.

Lastly, the Strategic Feature Omissions section explains the rationale behind excluding certain features, such as SEO optimization, to keep the project focused on its primary educational objectives. This decision reflects a deliberate effort to prioritize features that directly contribute to the project's goals, ensuring the core functionality remains uncompromised.

3.3 Graph UI Web System Technology Stack

In the development of the capstone project – a web system focused on Graph UI for Exploratory Learning – the selection of an appropriate and efficient technology stack was crucial. The core of the web system is built on Next.js, effectively harnessing the strengths of both React and Node.js, and further supported by TypeScript and Tailwind CSS. This combination of technologies was chosen for their synergistic properties, each contributing significantly to the project's functionality and user experience.

Next.js stands as the backbone of the application, chosen for its ability to merge server-side rendering (SSR) with client-side rendering (CSR) [19]. This integration is crucial for optimizing the performance of the application. The SSR mechanism enhances the initial load time, while CSR ensures a dynamic and responsive user interface. The use of React within Next.js is especially beneficial for its modular and component-based architecture. This allows for efficient development and maintenance of the user interface, which is essential for the graph-based design of the system, where interactivity and data visualization are central.

Node.js, integral to Next.js, underpins the server-side processes. Its non-blocking I/O model makes it an ideal choice for handling the application’s operations, ensuring smooth and efficient performance with great development experience [20]. This capability is vital in managing complex graph-related computations and user interactions without compromising the system's responsiveness.

TypeScript was selected over JavaScript for its robustness in type safety and error handling [21]. The static typing feature of TypeScript offers a more structured approach to coding, significantly reducing the likelihood of runtime errors and enhancing the overall maintainability of the codebase. This feature is particularly valuable in a project where the data types and interfaces can be complex.

Tailwind CSS increases the speed of the project's front-end development. Its utility-first approach facilitates rapid UI development, allowing for a high degree of customization and control over the application's design [22]. Unlike traditional CSS frameworks that come with predefined components, Tailwind CSS provides the flexibility to craft a unique and optimized user interface that aligns perfectly with the project's design requirements.

In considering alternatives, frameworks like Angular or Vue.js for front-end development and Django or Ruby on Rails for server-side processes were evaluated. Regarding the project development team experience, the combination of React's vast ecosystem and community support, along with Node.js's performance efficiency, positioned Next.js as the superior choice for this project. Similarly, while traditional JavaScript could have sufficed, TypeScript's advanced features offer greater benefits. For CSS styling, frameworks like Bootstrap or Materialize were considered, but Tailwind's flexibility and design control were more aligned with the project's needs. The selected technologies provide a balanced blend of performance, scalability, and developer ergonomics.

3.4 Semantic Portal API

In the development of the web system for Graph UI in Exploratory Learning, the choice of a reliable and comprehensive data source was crucial. To do so, the public API provided by Semantic Portal was utilized. The API is available at "http://semantic-portal.net/api". It is a platform that aligns seamlessly with the educational and informational needs of the project. Semantic Portal, as described on their website, is an educational project dedicated to facilitating quick and effective learning for students and professionals [23]. Their focus on knowledge modeling, ontologies, and intelligent educational web systems makes them an ideal source for the kind of data the project requires.

Semantic Portal's modern ontology-oriented approach offers a rich repository of IT-related topics and information, which is integral to the graph-based visualizations in the application. The portal's API provides access to a wide array of well-structured data, describing various IT subjects that are essential for creating informative and interactive graphs. This accessibility to a diverse range of topics ensures that the graphs generated in my application are not only relevant but also comprehensive in their coverage of IT knowledge.

Key Semantic Portal public API endpoints that can be used in the project are:

1. Branch endpoint. This section of the API provides comprehensive data about various knowledge branches, including their concepts, relations, and didactic relations. The endpoints such as “branch/[branch_name]” deliver all relevant data for a specific branch, which is crucial in constructing the initial structure of the graph. An example of that kind of branch is “Python”, which would consist of various information about the topic such as Python functions, modules, variables, data types, etc.
2. Concept endpoint. With an endpoint like “concept/[concept_id]”, the API allows the fetching of detailed information about specific concepts by their IDs. This is particularly useful when a user interacts with a node in the graph, as it enables the application to retrieve and display detailed information about that concept. It consists of theses, which could be of different types, like a subject definition, or some statement about the subject.

The integration of these API endpoints into the web system is well-planned. For the initial graph load, the application leverages the endpoint “branch/[branch_name]”. This endpoint is essential in constructing the primary structure of the graph, providing a comprehensive overview of a particular branch and its interlinked concepts and relations. It has the data necessary for defining nodes and links between them. For fetching more specific information related to the nodes, the endpoint “concept/[concept_id]” is utilized. This allows the system to present detailed and relevant information for each selected concept on node selection, thereby enriching the user’s learning experience.


```
{
    "id": "6377",
    "concept_id": "110",
    "view": "python",
    "thesis": "",
    "class": "c2v-created-here",
    "rating": "0",
    "count": null,
    "parsed": null,
    "hasConcept": "0",
    "to_concept_id": null,
    "to_thesis_id": null,
    "to_thesis_caption": null
},
```


Image 1. Example of the Semantic Portal data payload

As can be seen in the Image 1, the Semantic Portal API provides the clients with the data in the JSON format. In order to build the graphs, this data still has to be filtered and transformed to be successfully consumed by the graph library.

Opting for Semantic Portal's API was a strategic decision to avoid the time-intensive and potentially error-prone process of data collection. By leveraging their API, the development team was able to focus more on the development aspects of the project, ensuring the robustness and efficiency of the application. The API's quick response time and the breadth of content available significantly expedited the development process, making it possible to concentrate on enhancing the user experience and interactivity of the graph UI.

3.5 Graph Library for Graph UI

In the process of developing the Next.js application, a critical component was the selection of an appropriate graph library, a decision that would significantly influence the application's performance and the efficacy of the user interface. After a thorough evaluation of several leading libraries, including Vis.js, Sigma.js, and Cytoscape.js, 'react-force-graph-2d' was chosen for its superior performance and customization capabilities. The summarized findings are collected in the Table 2 below.

Graph Library	**Pros**	**Cons**
Vis.js	● Dynamic visualization. ● Offers a range of graph visualization possibilities.	Limited performance with large datasets or complex structures.
Sigma.js	● Efficient rendering of large data networks. ● Handles large-scale visualizations well	● Requires extensive knowledge of graph theory. ● Complex, steep learning curve. ● Restrictive customization options
Cytoscape.js	● Powerful for rendering complex network structures. ● High degree of interactivity	● Performance concerns with large datasets. ● Customization not fully aligned with project needs.

		• Integration challenges with React
react-force-graph-2d	• Exceptional performance. • Seamless integration with React. • High customization capability using JavaScript Canvas	No significant cons noted for the project's requirements.

Table 2. Graph Libraries Overview.

Vis.js, a dynamic visualization library known for its browser-based versatility, was one of the considered options [24]. It offers a range of graph visualization possibilities, making it an attractive choice initially. However, a notable limitation emerged in terms of performance, particularly when handling large datasets or complex graph structures. This shortcoming could potentially lead to slower rendering times and a diminished user experience, especially in an application where real-time data interaction and responsiveness are paramount.

Sigma.js, another candidate, is acclaimed for its efficient rendering of extensive data networks [25]. Its capability to handle large-scale visualizations was a significant draw. However, the complexity associated with Sigma.js became apparent. It requires a robust understanding of graph theory and sophisticated data visualization techniques, posing a steep learning curve. For example, the library was not able to provide the coordinates of the nodes of the graph. Some additional modules should be used to calculate the optimal positions of the elements of the graphs, which included iterative calculations. Moreover, its customization options, though present, were somewhat restrictive, potentially hindering the creative aspect of the application's design.

Cytoscape.js, similar to the others, is a powerful graph theory library used for data visualization [26]. While it excels in rendering complex network structures and offers a high degree of interactivity, the library's performance can be a concern with extensive and intricate datasets. Additionally, the level of customization, though adequate, did not fully align with the project's specific requirements, particularly in terms of integrating seamlessly with the React-based architecture of the application.

Consequently, 'react-force-graph-2d' emerged as the optimal choice. This library stood out for its exceptional performance, crucial for an application prioritizing real-time data interaction and visualization. Its compatibility with React further ensured

seamless integration into the existing application structure. A defining feature that influenced this decision was the library's customization capacity, particularly its use of JavaScript Canvas for element design.

JavaScript Canvas, a technology integral to 'react-force-graph-2d', makes it possible to render two-dimensional shapes and images. This capability is particularly advantageous in a graph-based UI, as it enables the creation of highly customized and interactive visualizations. The flexibility offered by JavaScript Canvas in designing and manipulating graphical elements played a pivotal role in enhancing the overall user experience and the application's visual appeal. At the same time, JS Canvas is an imperative browser API, which means that working with it can be tedious and complicated. Common in the front-end development CSS technology has no effect on JS Canvas, so some specific knowledge and techniques are required to be able to create visual elements with this tool.

3.6 Architecture of Graph UI Web System for Exploratory Learning

The architectural diagram for the capstone project below (Figure 2) illustrates the structure and flow of data within the web system, detailing how the various components interact to provide a seamless user experience.

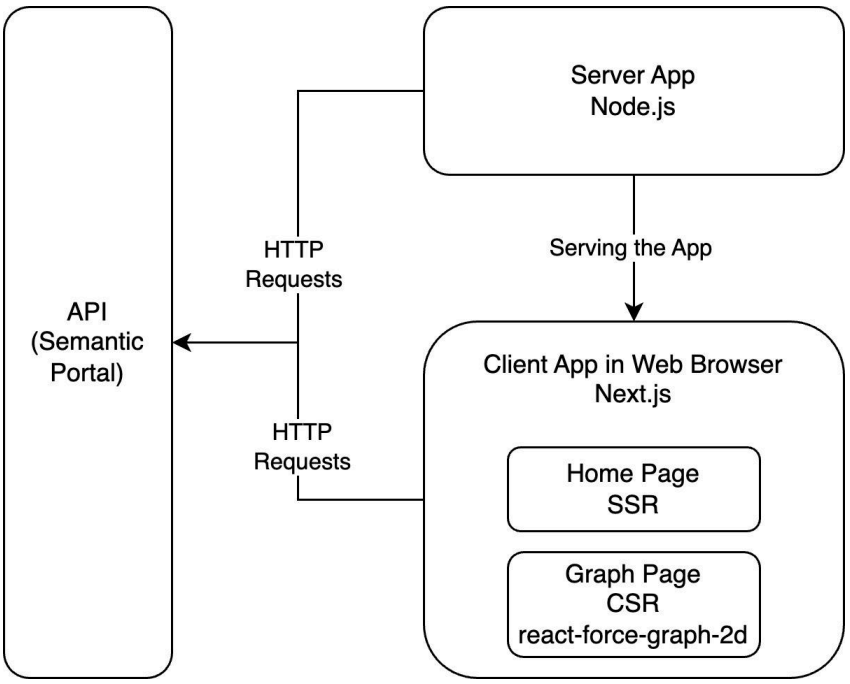

Figure 2: Architectural Diagram

At the core of the architecture is the Server Application, running on Node.js, which is responsible for serving the web application. The server interacts with the Client Application via HTTP requests, which are the standard protocol for the exchange of information on the web.

The system's data is sourced from an external API provided by Semantic Portal. HTTP requests are made to this API to fetch data related to IT topics, which are then used to draw the informative graphs within the application. The data flow is unidirectional, which simplifies the interactions between the Server App, the Semantic Portal API, and the Client App. This design is deliberate to ensure clarity in the data handling process and to maintain a clean separation of concerns within the application's architecture.

The Graph Page employs Client-Side Rendering, where the browser downloads a minimal HTML page, fetches JavaScript, and then renders the content dynamically. This part of the application leverages 'react-force-graph-2d', a React library that was described in details above. The use of CSR here allows for rich interactions and animations within the graph as users engage with the data, providing an interactive and engaging experience. As this part of the application is very dynamic and involves requesting new pieces of data upon the user interaction, the content of the graph can't be fully provided beforehand, so the CSR is very effective for the Graph UI web system.

3.7 Technical Decisions and Implementation Details of Graph UI Web System

The construction of our exploratory learning web system depends significantly on efficiently fetching and processing data from the Semantic Portal. In pursuit of this goal, the functional programming paradigm was adopted, focusing on creating pure functions dedicated to data filtering and transformation. This approach was instrumental in ensuring both the consistency and reliability of our data operations, and also aligns with the modern approach to the React applications building. In the Image 2 below one can see the example of the pure function for filtering the data implemented in the discussed web system.

```
const SUPPORTED_RELATIONS_CLASSES = ['c2c-part-of', 'c2c-is-a'];

export const filterData = ({ concepts, relations, didactic }: SemanticPortalData) => ({
  concepts: concepts,
  relations: relations.filter((relation: SemanticPortalNode) =>
    SUPPORTED_RELATIONS_CLASSES.includes(relation.class),
  ),
  didactic,
});
```

Image 2. A pure function for filtering the data from Semantic Portal

Data retrieval from the Semantic Portal is executed through a dual strategy, encompassing both server-side and client-side operations. On the server, we employed Next.js Server Components to implement Server Side Rendering (SSR). That Server Component with the asynchronous fallback can be seen in the Image 3 below.

```
async function loadBranch(branch: string) {
  const res = await fetch(`${SEMANTIC_PORTAL_API}/api/branch/${branch}`);
  return res.json();
}

export default async function Branch({ params: { branch } }: { params: { branch: string } }) {
  const branchData = await loadBranch(branch);

  return (
    <div>
      <Suspense fallback={<span className="□text-white">Loading...</span>}>
        <DynamicGraph branchData={transformData(filterData(branchData))} branch={branch} />
      </Suspense>
    </div>
  );
}
```

Image 3. Next.js Server Component with data fetching

On the client side, our application directly interacts with the public API of the Semantic Portal. This interaction is done through a custom React Hook, which encapsulates the logic for data fetching. The React Hook is intricately designed to leverage the component lifecycle methods provided by React, ensuring efficient management of side effects and state changes within the application.

```
const useFetchConceptData = (node: SemanticPortalNode) => {
  const [conceptData, setConceptData] = useState<SemanticPortalNode[] | null>(null);

  useEffect(() => {
    const fetchConceptData = async () => {
      if (!node?.id) return;

      const res = await fetch(`${SEMANTIC_PORTAL_API}/concept/${node.id}/thesis`);
      const parsedData = await res.json();

      setConceptData(filterThesis(parsedData));
    };

    fetchConceptData();
  }, [node]);

  return conceptData;
};
```

Image 4. React Hook for data fetching and state management on the client

Navigation within the web system was addressed through the implementation of Next.js App Routing. This powerful feature of Next.js made possible the construction of dynamic routes that integrated knowledge topics into the URL paths by organizing and naming the project directories in the proper way. Such arrangement enabled direct

access to various graph pages within the application, improving the user journey and link sharing.

The integration of 'react-force-graph-2d' was aligned with the React paradigm. The library was embedded within a React Functional Component, allowing to configure and manipulate the library's features to meet our project's specific requirements.

```
return (
  <div className="graph-container">
    <ForceGraph2D
      ref={fgRef}
      graphData={graphData}
      onNodeClick={handleClick}
      backgroundColor={backgroundColor}
      nodeCanvasObject={nodeCanvasObject}
      linkColor={linkColorFunction}
      linkWidth={1}
      linkDirectionalArrowLength={3}
      linkDirectionalArrowRelPos={0.33}
    />
  </div>
);
```

Image 5. The 'react-force-graph-2d' graph library integration and configuration

JavaScript Canvas was extensively used to design the graph nodes. It is important to mention that this API is imperative and requires manual design and configuration of the visual elements. Such things as text alignment and positioning could be very difficult within this browser API compared to doing so with CSS. Below is an example from the project source code of how a single element (the colored base of the node circle) is built.

```
ctx.beginPath();
ctx.arc(node.x, node.y, nodeSize / 2, 0, 2 * Math.PI, false);
ctx.fillStyle = nodeColor;
ctx.fill();
```

Image 6. Creating an element with JavaScript Canvas

3.8 User Interface of Exploratory Learning Web System and Graphical Elements

Moving to the user-faced part of the application, the graph is undoubtedly the main point of the Graph UI application, a central feature around which the entire user experience is crafted. In designing the interface, the principle of minimalism was used, which is reflected in every aspect of the user interface to avoid unnecessary complications that could detract from the user's engagement with the graphs.

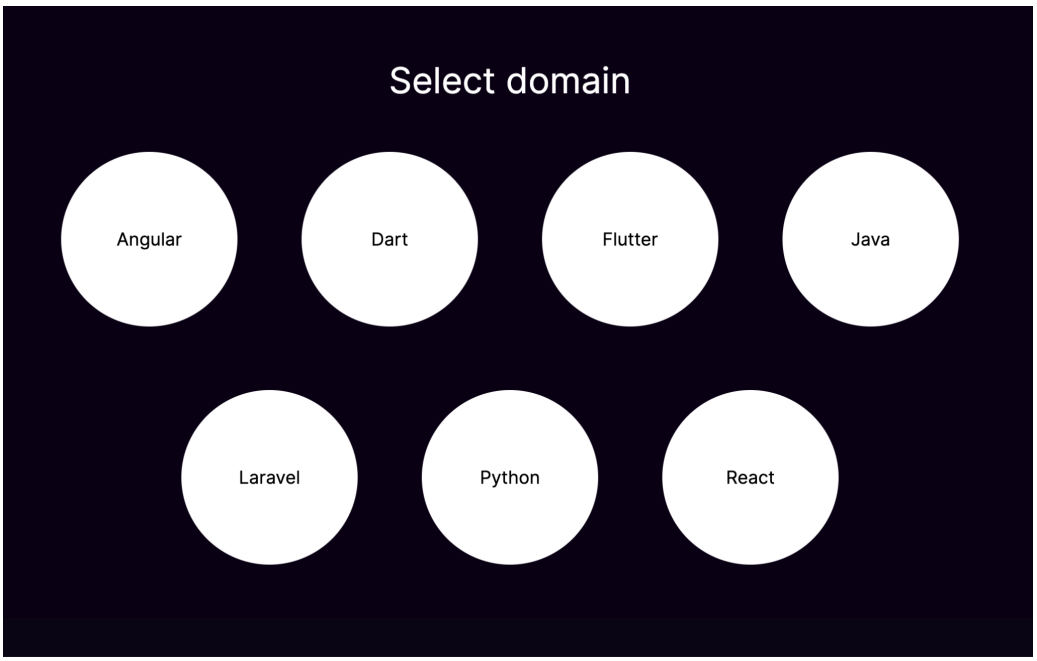

Image 7. Home page on Desktops

Starting with the home page (displayed in the Image 7), simplicity is key. It hosts the title of the application and presents users with a series of buttons. These buttons represent the various graph options available to the user and are styled as graph nodes to maintain a consistent look and feel across the application. This stylistic choice not only unifies the design but also subtly reinforces the graph-centric nature of the application. Moreover, as can be seen in the Image 7 below, the page is responsive, so it can adapt to different clients' devices by changing font size and element spacing.

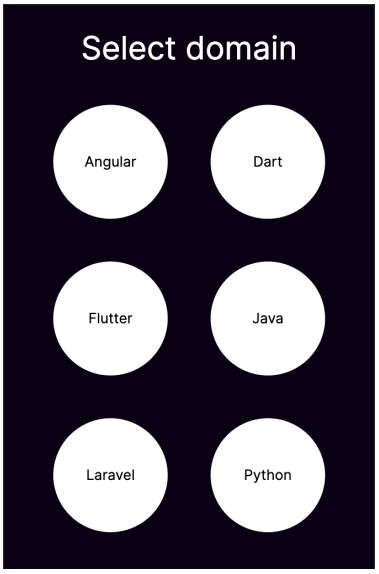

Image 8. Home page on Mobile Devices

Upon navigating to the Graph page, users see the graph itself (Image 9). The graph occupies merely all the available space. It is positioned front and center, inviting users to interact with its nodes and connections. It is designed to be immediately accessible, displaying smaller graphs in full without the need for adjustment, while larger graphs offer intuitive navigation options, such as panning and zooming, allowing users to explore the data in detail.

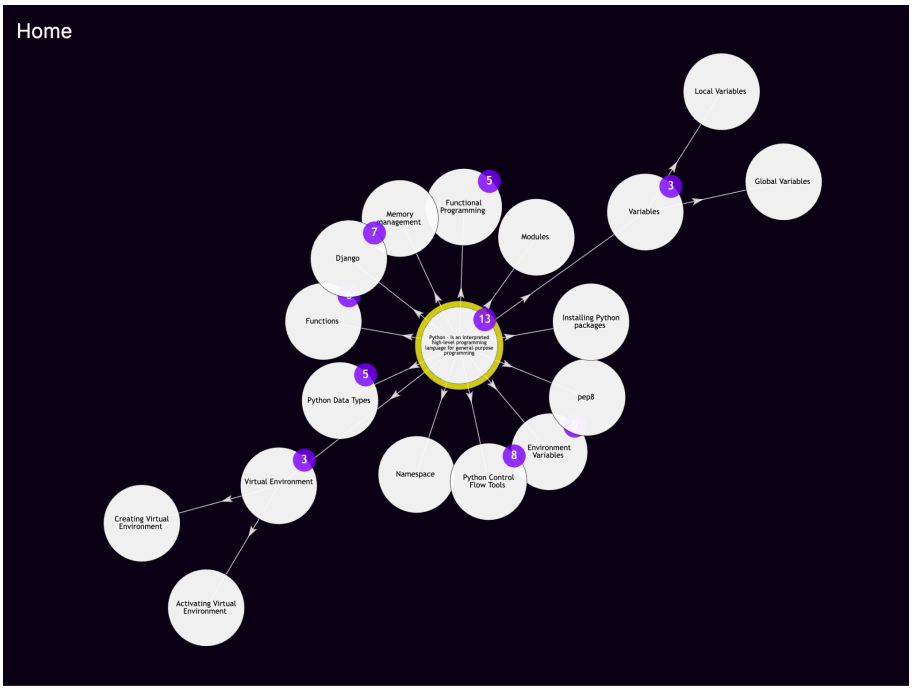

Image 9. Graph Page of the Web System application for Exploratory Learning

The application's color scheme is not just a design choice but a strategic decision informed by current trends and usability standards [27]. The dark background serves a dual purpose. It creates a visually appealing canvas that reduces eye strain and accentuates the graph elements. Purple, the primary color, is chosen for its depth, while white is used to achieve stark contrast, ensuring that text and node elements stand out crisply against the darker backdrop. This high-contrast color palette is both aesthetically pleasing and functional, enhancing the overall user experience.

In the general interface, it was decided not to use any animations specifically. This was not due to a lack of options or capabilities but rather from a commitment to maintaining a clean and distraction-free environment. Nevertheless, the application's interactivity is embodied within the graph itself, which is rich with motion and responsive to user interactions. The animations, from nodes popping to the smooth transitions as users interact with the data, were primarily functionalities provided out of the box by the 'react-force-graph-2d' library. These dynamic features in the graph were actually extended, integrating additional animations to ensure a cohesive and immersive experience. The library's default animations served as a foundation, enhancing the visual feedback and interactivity of the graph without overwhelming the user.

That use of animation serves a functional purpose as well, making the interaction with the graph not only informative but also engaging. The movements draw users into the data exploration process, making for a more memorable and interactive process, which can be crucial with exploratory learning.

Arial was chosen as the main font for its simplicity and readability, which complements the minimalistic design of the Graph UI application. Its clean and modern appearance ensures that all text, from titles to node labels, is easily legible against the application's dark background.

3.9 Graph UI Web System User Flow and Exploratory Learning Aspects

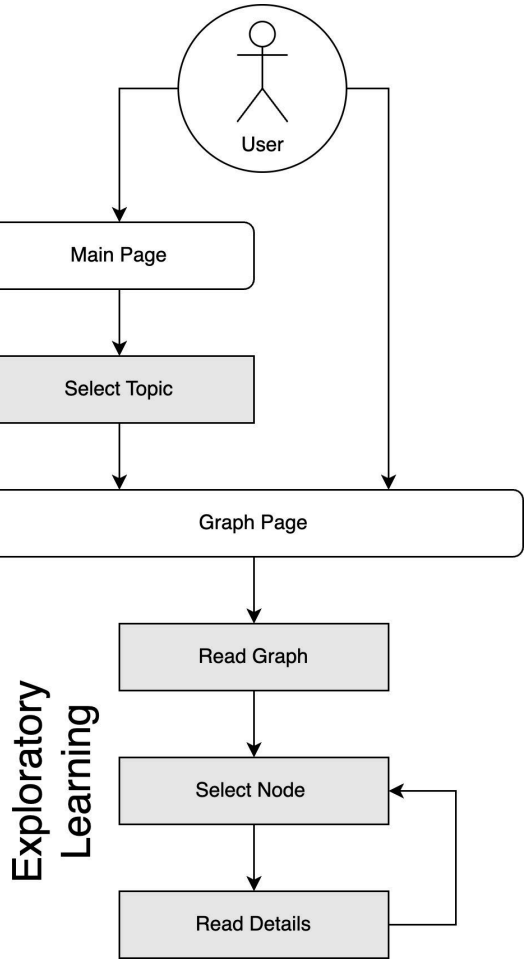

Figure 3: User Flow Diagram

Incorporating user flows in application development is an essential practice that leverages user-centered techniques to enhance software usability and acceptance [28]. User flows, which include personas, scenarios, and interaction models, are instrumental in understanding user requirements and designing interactions that cater to these requirements. The user flow diagram above (Figure 3) maps out the user's journey within the Graph UI application, detailing the sequential steps a user undertakes from initial engagement to the learning process.

Upon commencing the application, the user is directed to the Main Page. As mentioned earlier, this primary interface is intentionally designed to be intuitive,

presenting users with a curated selection of topics that guide them into the exploratory environment.

As the user selects a topic, they transition smoothly to the Graph Page. This page is the interactive heart of the application, where the visual representation of data through graph nodes and edges takes precedence. It is on this canvas that the user begins the active learning phase. There’s also an alternative way to get to the specific graph – a direct link with the graph name encoded into the URL.

Upon opening the graph, the user starts to investigate the graph. It is the stage where the visual data is interpreted and connections between concepts are formed. This stage is fundamental to the exploratory learning model, as it encourages the user to engage with the graphical representation in a meaningful way, building a cognitive map of the information presented.

The exploration deepens as the user selects a node. This is a step that shows the user's engagement with specific areas of interest within the broader topic. This action reveals a deeper layer of content, showing more children nodes if any available in the graph. Following the selection of a node, the user is presented with the opportunity to delve into more details. This function provides a focused view where granular information is conveyed, allowing the user to gain a comprehensive understanding of the particular subject matter.

This user flow diagram shows the educational philosophy of the Graph UI application, emphasizing a guided yet flexible approach to learning. It illustrates how the application's design — rooted in interactive graph visualizations — creates a conducive learning environment that encourages users to explore, interact, and engage deeply with the content. The iterative nature of this process affirms the application's role as a facilitator of learning through exploration and discovery.

3.10 Exploratory Learning Web System User Interaction Features

In the design of the Graph UI application, the unveiling of information is carefully planned to ensure an optimal user experience. Upon the initial launch of the graph, not all nodes are immediately visible to the user. This design decision was made to mitigate the visual complexity that can often be overwhelming at the outset of interaction with a data-rich graph. Presenting too much information simultaneously can not only be cognitively taxing but may also inadvertently stifle the user's curiosity

and desire to explore further. Therefore, a gradual disclosure strategy is employed, revealing the structure of the graph progressively, node by node, as the user navigates the data landscape. This stepwise exploration is intended to enhance the user's engagement and understanding by allowing them to focus on manageable segments of information at a time.

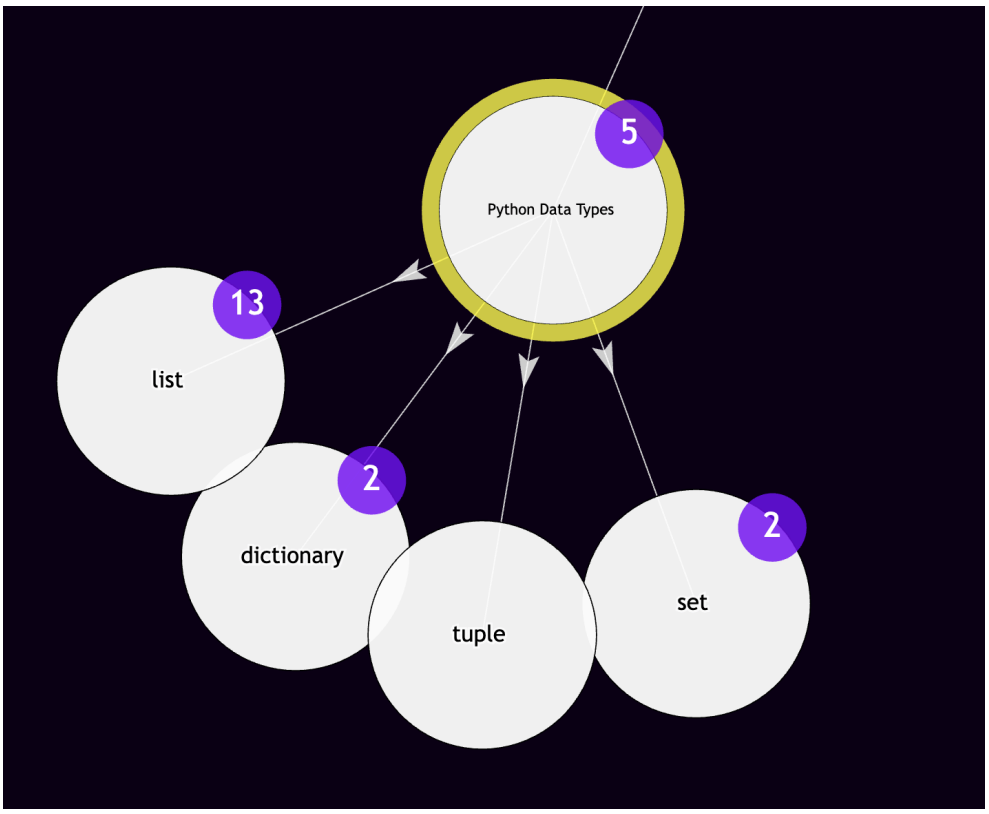

Image 9. Nodes with Links

To aid the user in navigating this incremental complexity, each node is equipped with a small bubble in the top-right corner, signaling the number of child nodes it contains. This subtle visual cue provides the user with insights into the node's connectivity and the breadth of information that awaits them, priming their exploration with context and anticipation. In the Image 9, one can see a closer look at the nodes and their linking in the UI.

Navigational controls within the application are intuitive and diverse, catering to the user's preferences. Users have the ability to zoom in and out, providing them with a mechanism to adjust their view of the graph to either a more detailed or more holistic perspective. This zoom functionality is complemented by the ability to click and drag the graph, allowing users to reposition their view seamlessly and explore different areas of the graph with ease.

A unique feature of the application is the incorporation of keyboard navigation, offering an alternative and efficient method for traversing the graph. Upon the graph's initiation, the main node is automatically highlighted and centered on the screen, distinguished by an outline of a contrasting color. This visual distinction marks the starting point of the user's journey through the graph's hierarchy. From this main node, the user can navigate the tree structure using the keyboard. Pressing the 'Arrow Down' key takes the user deeper into the tree, bringing child nodes into focus. To expand a node's branches and reveal its child nodes, the user can simply press the 'Enter' key, which imitates clicking on it. Horizontal navigation through sibling nodes is enabled by the 'Arrow Left' and 'Arrow Right' keys, allowing users to move laterally across the tree's breadth. Should the user wish to retract their steps or view higher-level nodes, the 'Arrow Up' key provides the means to ascend the tree.

As the user navigates using the keyboard, the camera automatically repositions to center the newly focused node, ensuring that the user's point of interest is always optimally displayed. This method of navigation not only accelerates the interaction process but also resonates with users who may find keyboard controls a more natural and engaging way to interact with digital content.

The additional benefits of keyboard navigation in web applications are significant, particularly in enhancing web accessibility. Keyboard navigation is crucial for many users, especially those with motor disabilities. People with tremors, limited hand use, or those who use modified keyboards rely heavily on keyboard navigation. Additionally, blind users also depend on keyboards for web navigation. Beyond accessibility, some users prefer keyboard navigation for efficiency or personal preference [30]. Implementing keyboard navigation is not just a compliance issue but a journey towards making the web more inclusive. By enabling keyboard navigation, web applications become accessible to a broader range of users, improving overall usability and user experience [29]. Therefore, adding the keyboard navigation to the Graph UI exploratory web app could significantly enhance the user's ability to interact with the system, allowing for a more inclusive experience.

The deliberate incorporation of keyboard navigation caters to an efficient workflow, reducing the time and effort required to navigate through the application. The familiar use of arrow keys and the enter button aligns with standard keyboard operations, enabling users to quickly adapt to the application's navigation schema. This responsiveness is not merely about ease of use but is also about reducing cognitive load and learning curve, which can be particularly beneficial for new or less tech-savvy users.

The implementation of keyboard navigation within the Graph UI application also opens the door to enhanced interactivity. As users engage with the graphical data using their keyboard, they become active participants in their learning journey. Each keystroke brings a new piece of information into focus, creating a rhythmic and tactile interaction with the content. This active engagement is essential in exploratory learning, as it fosters a deeper connection between the user and the material being studied.

3.11 Chapter Summary

Chapter 3 of the report on the "Design and Development of the Graph UI Web System for Exploratory Learning" culminates in several key conclusions about the process and outcomes of creating the Graph User Interface for exploratory learning. The chapter illustrates how theoretical aspects of UI/UX design in educational technology were effectively translated into a practical, user-centric application.

A notable conclusion is the strategic selection and utilization of the technology stack – Next.js, TypeScript, and Tailwind CSS. This choice was pivotal to the project's success, ensuring that the final product was not only high-performing but also resonated with the users in terms of ease of use and learning effectiveness. The use of the Semantic Portal's API also stands out as a smart decision, enabling the team to efficiently leverage existing data sources and focus on refining the user experience and system functionality.

The project's approach to balancing functionality with aesthetics in the UI design is another key takeaway. The minimalistic design, coupled with carefully chosen visual elements, underscores the project's commitment to creating an intuitive and appealing learning environment. This approach effectively reduces cognitive load and enhances user engagement, a critical factor in educational applications.

Moreover, the development of an adaptive and interactive learning environment, highlighted by features such as keyboard navigation, underlines the project's dedication to accessibility and user engagement. These features not only enhance the learning experience but also cater to a wider range of user needs.

Chapter 4. Conclusions

As we progress into the final sections of this report, we will explore several critical facets of the Graph UI application project. These include the challenges faced during development, the inherent limitations of the project's current scope, a comprehensive summary of the project's achievements and learning outcomes, recommendations for future enhancements, and concluding reflections that encapsulate the project's journey and its broader implications. This multi-dimensional analysis aims to provide a holistic understanding of the project's trajectory, from its conception to its culmination, and the roadmap for its future evolution.

4.1 Project Challenges

In the creation of the Graph UI application for Exploratory Learning, the development team encountered several challenges that tested the limits of their technical acumen and creative problem-solving abilities. These challenges spanned across various aspects of development, from selecting the optimal library to managing complex data models and customizing the user interface.

One of the primary hurdles was identifying a suitable graph library that could meet the performance requirements and offer the necessary features for an interactive graph-based UI. The selection process was exhaustive, as it required extensive research and comparison of multiple libraries. Each candidate library — Vis.js, Sigma.js, Cytoscape.js — was evaluated for its capabilities and limitations. The need for a library that could handle intricate data visualizations with efficient performance and provide customizable options was very important. The decision to use 'react-force-graph-2d' was not arrived at lightly, but only after careful consideration of the specific needs of the project and the potential for scalability and user engagement.

Another challenge was preparing specific data formats. The application aimed to visualize intricate relationships between various data points, which required some data normalization and filtering of the data provided by the Semantic Portal. Ensuring that the data model was both comprehensive and efficient enough to support real-time interactions posed a considerable challenge. It involved not only understanding of

graph theory but also the ability to implement this understanding in a way that was both computationally efficient and easily navigable for the user.

Customizing the user interface to create a minimalistic yet functional design also presented its challenges. Striking a balance between simplicity and functionality required a nuanced approach. The UI needed to be intuitive enough for users to engage with without extensive instructions while still providing all the necessary tools for data exploration. This was achieved through iterative design and testing, ensuring that each UI element was both aesthetically pleasing and served a clear purpose.

The exploration of new UI features, such as the keyboard navigation system, was a venture into uncharted territory. It required not only the implementation of these features but also the integration of them into the existing UI without disrupting the user experience. This experimentation was crucial in enhancing the accessibility and efficiency of the application but brought with it the need for rigorous user testing and refinement to achieve an optimal result.

Finally, the use of JS Canvas for custom element design was a technically demanding aspect of the project. JS Canvas offers a powerful way to create custom graphics and interactive experiences, but it requires a deep understanding of JavaScript and graphical concepts. The work with JS Canvas was challenging due to the need for precise control over the rendering of graphical elements and the performance considerations inherent in real-time data visualizations.

In confronting these challenges, the project became an opportunity for professional growth and learning. Each obstacle required a tailored solution, often involving the acquisition of new skills or the innovative application of existing ones.

4.2 Web System Limitations

The development of the Graph UI application was subject to certain limitations which were either deliberately chosen or emerged as a result of scope definition and resource prioritization. Here are some of the constraints within the project.

The application was designed to handle a specified range of complexity within graph data. Massive graphs extending beyond this range were left out of scope due to exploratory learning needs considerations. The decision to limit the scope of graph complexity was made to ensure that the application could deliver a smooth and responsive experience within the defined use cases.

In focusing on core functionalities, such as graph visualization and node exploration, other potential features were not included in the current iteration of the application. This includes advanced data types representations (images, code blocks, etc.), or more extensive customization options for the UI. These were deemed beyond the scope of the initial project to maintain a tight focus on the primary objectives.

While the application supports exploratory learning through interactive graphs, it does not encompass a full suite of data analysis tools that one might find in more specialized software. The depth of data manipulation and analysis was limited to ensure clarity and usability for the target user base, which does not necessarily have a background in data science.

The UI was designed with English-speaking users in mind, and as such, it lacks multilingual support or localizations. This limitation was a result of prioritizing development resources and time constraints, and it restricts the application's accessibility to a global audience.

While keyboard navigation was implemented to enhance accessibility, other accessibility features, such as screen reader support or alternative input methods for users with disabilities, were not fully developed and tested. The scope of the project did not extend to full compliance with all web accessibility standards, which is an area identified for future development.

The application was designed as a standalone system and does not currently support integration with external systems or APIs. This was a strategic choice to ensure the robustness and stability of the core offering before considering additional integrations. So far, it works with Semantic Portal and is binded to their specific data types.

By recognizing these limitations, the project acknowledges the areas that were not addressed in the current version. These areas represent deliberate trade-offs or potential enhancements for subsequent versions. Each limitation delineates the boundaries of the project's current capabilities and sets the stage for future updates and improvements.

4.3 Project Results

1. **Created a Web-Based System**. Developed a Graph User Interface system designed for exploratory learning in IT-related topics. This system leverages the power of interactive graphs to simplify complex information, making it more accessible for learners.

2. **Built a Graph User Interface according to the plan**. Implemented a minimalistic user interface approach, focusing on reducing cognitive overload and enhancing user engagement. This design philosophy was pivotal in making complex data more understandable and engaging for users.

3. **Made Strategic Technology Stack Selection**. Chose a technology stack that enhances the application's functionality and user experience:
    a. Next.js was selected for its blend of server-side and client-side rendering, ensuring a smooth and responsive interface.
    b. TypeScript was used to ensure code reliability and maintainability.
    c. Tailwind CSS was incorporated for custom, user-friendly styling.

4. **Optimized Graph Visualization**. Integrated 'react-force-graph-2d' as the primary graph visualization library. This library was chosen for its high performance and customizability, particularly its effective use of JavaScript Canvas, which allowed for intuitive representation of complex, interconnected data.

5. **Overcome Development Challenges**. Faced and overcame multiple challenges throughout the development process, including:
    a. Selected a suitable graph library that met the project's specific requirements.
    b. Managed graph-specific data formats, requiring an in-depth understanding of knowledge modeling and graph theory.
    c. Balanced UI simplicity with functionality and integrating new features without compromising user experience.

6. **Innovated with Navigation Features**. Introduced a keyboard navigation system to enhance accessibility and efficiency. This feature is especially beneficial for users with motor disabilities and those who prefer keyboard navigation over traditional interfaces.

7. **Acknowledged Limitations and Made Recommendations for Future Development**. Recognized the limitations within the project's current scope and set the groundwork for future enhancements. These areas for improvement include handling more complex graphs, adding advanced data analysis tools, increasing accessibility features, and incorporating multilingual support.

4.4 Recommendations for Future Development

As the Graph UI application continues to be developed further and refined, several key areas emerge as ripe for future exploration and enhancement.

One of the most promising areas for development is the UI optimization, particularly for handling large and complex data sets. This could involve exploring position and space dimensions: applying space-efficient UI elements, changing the link (edge) length, etc. This would not only improve user experience but also broadens the application's capability to handle more sophisticated and data-intensive graphs.

The customizability aspect of the application also presents a significant opportunity for enhancement. Building on the current minimalist design, future iterations could focus on dynamic UI customization. This could mean allowing users to choose from a range of themes, including light and dark modes or even user-created themes. This level of personalization would not only improve accessibility, particularly for users with visual impairments, but also enhance the overall user engagement by allowing a more personalized interaction with the application.

Incorporating advanced analytical tools directly into the application could greatly enhance its utility for users. Features such as on-the-fly filtering of nodes, search functionalities within the graph, and basic statistical analysis tools would empower users to interact with the data more deeply. For instance, a user could filter nodes based on specific criteria or search for particular nodes, making the exploration process more targeted and efficient.

Another significant area for development is the integration of accessibility features. Building on the existing keyboard navigation, future versions could include voice

control capabilities, allowing users to navigate and interact with the application through voice commands. This would not only cater to users with motor impairments but also offer a hands-free experience for users who may prefer voice interaction. Additionally, implementing screen reader support and ensuring that all elements of the UI are accessible would make the application more inclusive.

Expanding the application's capabilities to include real-time collaboration tools could transform the solitary learning experience into a collaborative one. Features such as shared graph exploration, where users can simultaneously navigate and annotate a graph, or the ability to discuss findings in real-time within the application, could foster a more interactive and communal learning environment.

Lastly, the potential for community-driven content creation is promising. Allowing users, especially educators and experts, to contribute content or create custom graphs could turn the application into a dynamic and evolving educational resource. This could be facilitated through a user-friendly content creation interface within the application, where contributors can design and share their graphs, subject to quality and relevance checks. Alternatively, the integration with LLMs could be used for data creation. However, this approach is yet to be well-considered due to the concerns with the data reliability in the educational context, which could be mitigated with the manual reviewing of the materials.

Each of these potential developments not only addresses current limitations but also aligns with the overarching goal of the project: to create a dynamic, engaging, and inclusive educational tool that evolves with its users' needs and the latest technological advancements.

4.5 Final Thoughts

The project has successfully demonstrated the power of a well-thought-out user interface combined with sophisticated graph visualization tools to simplify and enrich the learning experience. The use of technologies like Next.js, TypeScript, and Tailwind CSS, along with the strategic choice of 'react-force-graph-2d', was a solid fundament for the application's functionality, ensuring a responsive and intuitive user experience. The challenges encountered, from selecting the right graph library to optimizing the UI for a diverse user base, were not just hurdles but valuable learning experiences that enriched the development process.

It is clear that the project has an opportunity for growth and further innovation. Whether it is enhancing performance for large datasets, expanding the application's accessibility features, or incorporating more advanced analytical tools, these areas for improvement signify the project's potential for evolution and adaptation.

The recommendations for future development pave a path forward, highlighting areas where the application can expand its capabilities and reach. The potential for integrating real-time collaboration tools, extending community-driven content creation, or enhancing accessibility features underscores the application's capacity to be more than just an educational tool—it can be a platform for shared knowledge and collective exploration.

In conclusion, this capstone project tries to work on the field of the transformative power of technology in education with Graph UI. It reflects a commitment to continuous learning and improvement, underscoring the importance of adapting to changing user needs and technological advancements. The journey of this project, from its conception to its realization, serves as a reminder that the process of trying to create innovative solutions is as important as the end product. The experiences, skills, and knowledge gained through this project will undoubtedly serve as a foundation for future endeavors in the field of technology and education.

Bibliography

[1] March, J.G. “Exploration and Exploitation in Organizational Learning”. In: Organization Science, Special Issue: Organizational Learning: Papers in Honor of (and by) James G. March 2.1 (1991), pp. 71–87.

[2] Rieman, J. “A Field Study of Exploratory Learning Strategies”. In: ACM Transactions on Computer-Human Interaction 3.3 (1996), p. 191. DOI: 10.1145/234526.234527.

[3] Fekete, J-D., Fisher, D., Nandi, A., & Sedlmair, M. (Eds.). “Progressive Data Analysis and Visualization”. In: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2019), pp. 2–3.

[4] Komorowski, M., Marshall, D.C., Salciccioli, J.D., & Crutain, Y. “Exploratory Data Analysis in Secondary Analysis of Electronic Health Records”. In: Secondary Analysis of Electronic Health Records (2016), pp. 185–203. DOI: 10.1007/978-3-319-43742-2_15.

[5] Herman, I., Melancon, G., & Marshall, M.S. “Graph visualization and navigation in information visualization: A survey”. In: IEEE Transactions on Visualization and Computer Graphics 6.1 (2000), pp. 24–43. DOI: 10.1109/2945.841119.

[6] Euler, L. “The seven bridges of Königsberg”. In: The World of Mathematics, vol. 1 (1956), pp. 573–580.

[7] Paoletti, T. “Leonard Euler’s solution to the Konigsberg bridge problem”. In: Convergence (2011).

[8] Barrasa, J., Hodler, A., & Webber, J. Knowledge Graphs. O'Reilly Media, Incorporated, 2021.

[9] di Battista, G., Eades, P., Tamassia, R., & Tollis, I.G. “Algorithms for Drawing Graphs: An Annotated Bibliography”. In: Computational Geometry: Theory and Applications 4.5 (1994), pp. 235–282.

[10] Novak, J.D., & Cañas, A.J. “The theory underlying concept maps and how to construct them”. In: Florida Institute for Human and Machine Cognition 1.1 (2006), pp. 1–31. Florida Institute for Human and Machine Cognition, vol. 1.1, pp. 1-31.

[11] Ausubel, D.P. “The use of advance organizers in the learning and retention of meaningful verbal material”. In: J. Educ. Psych. 51.5 (1960), pp. 267–272. DOI: 10.1037/h0046669.

[12] Tytenko, S.V. “Concept Maps, Their Application Types and Methods in Information and Learning Systems”. In: KPI Science News, no. 4 (2020), pp. 70–78. DOI: 10.20535/kpisn.2020.4.22709.

[13] Puntambekar, S., Stylianou, A., & Hübscher, R. “Improving Navigation and Learning in Hypertext Environments With Navigable Concept Maps”. In: Human–Computer Interaction 18.4 (2003), pp. 395–428. DOI: 10.1207/s15327051hci1804_3.

[14] Chen, P., Lu, Y., Zheng, V. W., Chen, X., & Yang, B. “KnowEdu: A System to Construct Knowledge Graph for Education”. In: IEEE Access, vol. 6 (2018), pp. 31553–31563. DOI: 10.1109/ACCESS.2018.2839607.

[15] Tytenko, S.V. “Interactive concept maps in ontology-oriented information and learning web-systems”. In: KPI Sci. News, no. 2 (2019), pp. 24–36. DOI: 10.20535/kpi-sn.2019.2.16751.

[16] Tytenko A., Tytenko S. “GRAPH USER INTERFACES FOR ENHANCING EXPLORATORY LEARNING: AN OVERVIEW.” In: Modern engineering and innovative technologies 29-03 (2023), pp. 124–129. URL: https://doi.org/10.30890/2567-5273.2023-29-03-047.

[17] Graph visualization UX. By Christian Miles. Cambridge Intelligence, 29th October 2018. URL: https://cambridge-intelligence.com/graph-visualization-ux-how-to-avoid-wrecking-your-graph-visualization/.

[18] Five steps to tackle big graph data visualization. By Dan Williams. Cambridge Intelligence, 21st September 2018. URL: https://cambridge-intelligence.com/big-graph-data-visualization/.

[19] Introduction. Next.js. Vercel. URL: https://nextjs.org/docs.

[20] About Node.js. The OpenJS Foundation. URL: https://nodejs.org/en/about.

[21] TypeScript Documentation. Microsoft. URL: https://www.typescriptlang.org/docs/.

[22] Documentation. Tailwind Labs Inc. URL: https://tailwindcss.com/docs/installation.

[23] Semantic portal. Ontology portals research team. APEPS department of Igor Sikorsky KPI. URL: http://semantic-portal.net/about.

[24] Vis.JS. Network. URL: https://visjs.github.io/vis-network/docs/network/.

[25] Sigma.JS. Architecture. URL: https://www.sigmajs.org/.

[26] Cytoscape.js. About. URL: https://js.cytoscape.org/#introduction/about.

[27] Eisfeld, Henriette, and Felix Kristallovich. "The rise of dark mode: A qualitative study of an emerging user interface design trend." (2020).

[28] Lopes, A., Valentim, N., Moraes, B. et al. “Applying user-centered techniques to analyze and design a mobile application”. In: J Softw Eng Res Dev 6 (2018). URL: https://doi.org/10.1186/s40411-018-0049-1.

[29] Modern-CSS. “Enhancing Accessibility: A Guide to Keyboard-Navigable Web Applications”. URL: https://modern-css.com/articles/accessibility/keyboard-navigable-web-applications

[30] WebAIM. Keyboard Accessibility. URL: https://webaim.org/techniques/keyboard/.

[31] Statistics Knowledge Portal. “Exploratory Data Analysis”. URL: https://www.jmp.com/en_us/statistics-knowledge-portal/exploratory-data-analysis.html