Design and development of a Web Application to Improve Tutor Search for Organizations and Individuals

Mahsa Derakhshan Alavijeh

Master's thesis
March 2025
Master in Full-Stack software development

Derakhshan Alavijeh, Mahsa

A Web-Based system for enhancing tutor discovery and scheduling efficiency

Jyväskylä: Jamk University of Applied Sciences, March 2025, 75 pages.

Degree Programme in Full Stack Software Development. Master’s thesis

Permission for open access publication: Yes

Language of publication: English

Abstract

The design and development of a tutor-finding platform which address the gaps of the online tutoring available in the market by integrating advanced features such as AI recommendations, reservation system, and secure authentication. By prioritizing a design thinking approach, the platform enhances accessibility and efficiency for both students and tutors.
A mixed-methods research methodology was used, combining quantitative surveys and qualitative interviews to gather user insights and inform an iterative design process. Implementation is based on the MERN stack, to have a scalable platform, while OpenAI-powered tools enable personalized learning experiences and real-time feedback. Compared to existing solutions, the system emphasizes usability, security, and modular architecture. Key features include advanced search filters, tutor reviews, and AI-driven learning adaptation, all designed to improve user’s satisfaction.
Building a strong foundation, future iterations will focus on refining the user interface, integrating payment systems, and expanding AI capabilities to address evolving educational needs. The findings contribute to the field by presenting a scalable, intelligent, and secure framework for developing online tutoring platforms that bridge existing gaps in digital education.

Keywords/tags (subjects)

Online tutoring, personalized learning, AI integration, MERN stack, JSON Web Tokens (JWT), reservation system

Contents

Figures

Tables

1 Introduction

In the past few years, several changes, including the COVID-19 pandemic, brought more popularity to online platforms compared to traditional face-to-face education (Sadeghi, 2022). This trend also happened in the educational field. Online tutoring helped to improved students' learning experiences. While educational institutions and individuals around the world adopt online teaching, the need for user-friendly platforms for both teachers and students to have quality education becomes demanding (Gorur & Dey, 2021). This study aims to address this need by designing and developing a website addressing the requirements of both students and teachers.

To achieve this goal, the requirements for the website were gathered. This process involved collecting data through literature review analysis, and conducting surveys and interviews with potential users. The survey results provided potential useful features of the website. Also, interviews allow to become more familiar with challenges in online tutoring. These findings inform the development of customized features and functionalities.

Based on the collected data the website was designed and developed. The website's features include advanced search - based on the tutor’s proficiency, subject, and budget-, tools to promote tutors' profiles and utilize Artificial Intelligence (AI) for giving feedback and recommendations. In addition, the development of a reservation system and a user-friendly interface ensure a smooth experience for both tutors and students.

The potential impact of the website is significant because it offers easy access to suitable tutors and enables tutors to reach a wide range of clients. In addition, AI integration enhances tutors' teaching experience, providing additional support to students while automating session management and reservation.

The development process was based on an iterative design approach. It includes potential users’ feedback at each stage. The MERN stack was chosen for implementation as it is simple and easy to learn. OpenAI was selected for AI integration because its functionalities are aligned with the website's features and requirements. The website was implemented by using the mixed-methods approach - quantitative surveys, and qualitative interviews-, conducting literature review and comparing existing platforms.

2 Litereture Review

In the next step, the effectiveness, strengths, and challenges of traditional and e-tutoring were studied. This helps to understand the concepts of conventional face-to-face tutoring methods and their role in modern e-tutoring systems. Specifically, this section explores the advantages of online tutoring over traditional settings and challenges like engagement and technical requirements of tutors and students.

Next, studies were about the process of choosing the technical stack for implementing a new tutoring website. The comparison was focusing on the MERN (MongoDB, Express.js, React.js, Node.js) and MEAN (MongoDB, Ex-press.js, Angular.js, Node.js). This section provides a rationale for the decision-making process. It includes details of how each component of the stack contributes to the overall functionality and user experience.

Then the application of AI in educational contexts, focusing on Machine Learning (ML) and Deep Learning (DL). It discusses the available and potential functionalities that provide real-time interactions in e-tutoring environments. The study also includes comparing NLP platforms like DialogFlow and ChatGPT. NPL platforms integrate conversational agents into educational settings. And by the end, a proposed prompt structure for an AI-based tutoring system was retrieved to promote active learning, instead of giving the answer directly.

2.1 Importance of online tutoring

Tutoring has been always recognized as a powerful educational tool. The reason is that it providing personalized support to students outside the standard classroom setting (Alrakhawi et al., 2023; Wang et al., 2023). In the traditional tutoring, where students receive face-to-face assistance from tutors, has its own advantages. Immediate feedback and personal human interactions makes it unique, which leads to a better adjustment to individual learning needs (Smith & Hill, 2019). This helps students for better understanding and motivation. Research suggests that when tutors and students share a common knowledge base and social background, it creates a sense of alignment that enhances the learning experience (Kwok, 2010; Kwo & Bray, 2014). The environment of traditional tutoring is supportive in a way that helps students feel more connected and engaged. This leads to improved academic performance and satisfaction.

In the recent years, e-tutoring, became more popular, especially during the Covid-19 pandemic (Alrakhawi et al., 2023; Wang et al., 2023). E-tutoring uses online platforms for flexible scheduling and accessibility with no location limitations (Raviolo et al., 2020; Suneja & Bagai, 2021). This approach has been found to maintain, and in some cases even improve, in terms of social and mental supports both, despite the lack of physical presence (Alrakhawi & Jamiat, 2023). Research shows that online interactions can provide effective learning atmosphere by providing clear explanations and maintaining supportive interactions (e.g., Yung, 2015; Zhan, Bray, Wang, Lykins, & Kwo, 2013). Research has shown that, for instance, E-tutoring can provide a variety of learning styles through multimedia resources and interactive tools, which helps to a richer and more personalized educational experience (Adams & Wilson, 2021; Zhao & Lei, 2020).

Comparing traditional and e-tutoring learning approaches, reveals that each approach has its own strengths and challenges. Although both traditional and e-tutoring approaches lead to higher educational attainment, their impacts can vary. This result variation depends on the context and usage of each approach. For example, in traditional tutoring face-to-face interactions can enhance personal connections and immediate feedback. In e-tutoring, it is less personal, more flexible and offers innovative solutions that can adapt to modern educational needs. By understanding the key benefits and advantages of each approach, students get the best result out of their studies. As education keeps evolving, mixing traditional and online tutoring methods could be the best way to support students and help them succeed.

Through direct interaction in the traditional tutoring student and tutor build a personal human connection, which helps addressing student concerns (Concorde Education, n.d.; Miteacher.ai, n.d.). However, e-tutoring is convenient in the terms of scalability, and the use of technology to support learning (Alrakhawi et al., 2023; Wang et al., 2023; Yung & Yuan, 2018). For example, online platforms contain features and tools like virtual simulations and interactive exercises. Usually these features may not be accessible in face-to-face settings (Park & Bonk, 2007; Means et al., 2013). This flexibility makes e-tutoring accessible to more students, especially those in remote or underserved areas (Okebukola, 2025, p. 87).

These advantages in e-tutoring are only effective in circumstances. Most important factors are technical proficiency and the ability to create engaging online environments (Bizami, Tasir & Kew, 2023; Gikandi, Morrow & Davis, 2011). Research shows that students in e-tutoring can be just as engaged and satisfied as those in traditional settings, or even more, as long as the online format is well-designed and supported by the right technology (Sriprakash, Proctor, & Hu, 2016). For instance, Coetze et al. (2018) found that regular online engagement and structured digital interactions can improve academic performance. However, it is important to address technical challenges and ensure that online platforms create a sense of connection and support similar to in-person learning (Bizami, Tasir & Kew, 2023; Gikandi, Morrow & Davis, 2011).

A well-designed intelligent tutoring system (ITS) should include key features that improve learning. It should adapt to each student by adjusting content based on their learning style, knowledge level, and cognitive abilities (Barrow et al., 2024). Real-time feedback and personalized assessments help fine-tune the teaching approach as needed (Zhou, 2024). AI-driven emotion recognition can also make learning more personal by considering the student's mood and engagement (Li, 2023). Plus, having a flexible curriculum helps students focus more on areas they struggle with, making the learning process smoother (Barrow et al., 2024). These features make ITS more effective than regular e-learning platforms by offering a more tailored and engaging experience (Singh, Gunjan, & Nasralla, 2022).

2.2 Technical stack

In the process of implementing the website, one of the important steps is to evaluate and justify the choice of technical stacks. This includes considering features the programming language provide. One of the important items to consider was available time and resources. Considering the prior experience with JavaScript, the options were narrowed down to two stacks: MERN (MongoDB, Express.js, React.js, Node.js) and MEAN (MongoDB, Express.js, Angular.js, Node.js). The previous knowledge of these technologies made the process more efficient in development, and helping to meet project deadlines effectively.

Both the MERN and MEAN stacks share a common backend foundation, utilizing MongoDB for database management and Express.js and Node.js for server-side handling, and executing JavaScript (Subramanian, 2019). The most significant difference between the two stacks is in the frontend framework. MERN stack uses React.js, while the MEAN stack incorporates Angular.js. React showed as one of the most popular stacks among developers and companies, while Angular.js has experienced a decline (Stack Overflow, 2023). This trend is depicted in Figure 5, which highlights the growing popularity of React.js. After a thorough evaluation of the components of both stacks, including their features in the front-end libraries, MERN stack was chosen as the stack to implement the tutor finding website.

The decision to choose MERN was based on the scalability and efficiency of development (Porter, Yang, & Xi, 2019). MERN stack is more suitable for projects that are more dynamic and responsive to user interfaces (Johnson, 2021). As a result, MERN making it a suitable choice for this implementation. React.js, offers flexibility in user interactions with the system through its component-based architecture (Subramanian, 2019). The increasing popularity of React.js, is related to its efficiency in driving complex UI components.

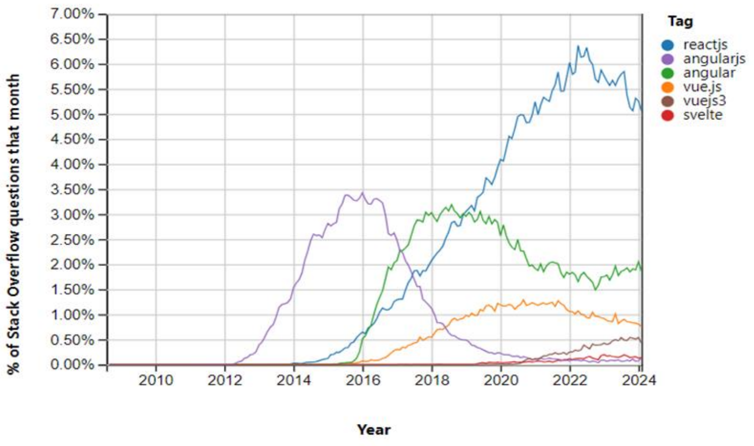


Figure 1. Trends in frontend technology

React.js

React is a JavaScript library designed for rendering user interfaces (UIs) (React, n.d.). Unlike frameworks such as Angular, React is a library. This provides developers with higher level of flexibility in architecting systems. The React's component-based structure enables the development of reusable UI elements (Libby, 2023). This can be handled without forcing a rigid pattern or structure (Subramanian, 2019). This flexibility allows developers to build more adaptable, reusable and maintainable applications.

React was initially developed by Facebook for internal use in their company to replace the conventional MVC (Model-View-Controller) architecture on the client side (Gackenheimer, 2015). The MVC model works with a model module that is directly connected to the view, can complicate implementation, especially when managing multiple models and views within a web application (Subramanian, 2019). In order to optimize the MVC model and challenges in maintaining a direct connection between model and view, Facebook adopted a declarative view structure. In this approach unlike Angular and MVC, in React separates Document Object Model (DOM) manipulation from the view layer, and then introducing a virtual DOM that sits between the HTML page and the actual DOM. This virtual DOM enables more efficient updates, as it compares the current state with the updated version, only modifying the actual DOM when changes occur (Subramanian, 2019). As noted by Subramanian (2019), "being declarative makes the views consistent, predictable, easier to maintain, and simpler to understand" (p. 6).

Developers in the React's component-based architecture are allowed to create individual UI elements as components, and each managing its state (React, n.d.). These components interact with each other through a read-only property known as "props". Props enabling the smooth exchange of data between components (Subramanian, 2019). Each component forms a node within the virtual DOM, making it reusable, fast, and easily manageable with other components. React also introduced the use of JSX (JavaScript XML). JSX simplifies the combination of HTML and JavaScript for UI representation, further enhancing the development process.

NPM and Node.js

Node Package Manager (NPM) is an important component of Node.js that provides for developers a platform to share, download, and manage code packages efficiently (NPM, n.d.). This is known as the serves as the standard package manager for Node.js, facilitating the installation and management of dependencies in JavaScript applications (Subramanian, 2017). It allows to publish and retrieve features and codes via NPM repositories (NPM, n.d.). Node.js itself enables the use of JavaScript for backend development. It also is providing a server-side environment that offers several advantages over traditional web technologies. Node.js operates on several key principles:

Single-threaded Asynchronous Execution: JavaScript is basically an asynchronous language. It operates on a single thread to execute different tasks. In order to manage multiple operations, it utilizes callback functions. The call-back functions allow processes to continue running in the background while other code is executed without stopping.

Event-driven Architecture: Node.js architecture is based on operates on an event-driven. This allows to manage tasks, which are initiated by user inputs or system-generated events (Node.js, n.d.). Callback functions are meaningful in this context, as they help the system to pass results back as soon as an event is completed.

Streamed I/O (Input/Output): Node.js facilitates streamed input and output without any stop. This feature is especially useful for real-time communication scenarios or when handling large data (Subramanian, 2019).

These features make Node.js an ideal solution to build a reliable application with a large number of connections and requests. Its architecture is designed in a way to handle performance issues during the input/output (I/O) operations. It also prevents single points of failure which means it brings more reliability for the networked applications. As Subramanian (2019) highlights, Node.js helps to build applications featured with multiple servers, minimizing performance bottlenecks. Moreover, as it is an asynchronous language, it allows efficient handling of network operations, like TCP, HTTP, UDP, and WebSocket protocols (Kurose & Ross, 2017). By this support for various network protocols, it would be convenient to implement applications such as web HTTP requests and real-time chat systems, which rely on standardized IM and chat protocols like SIP/SIMPLE and XMPP (Jennings et al., 2006).

Node.js is a runtime environment for JavaScript, which allows developers to build server-side applications using JavaScript (Node.js, n.d). Express.js also is a library to empower this library. It is renowned for its speed, minimalism, and flexibility. Unlike many frameworks that have strict rules to follow, Express.js provides developers with a significant degree of freedom. For example, it allows the implementation of various architectures, including the widely adopted Model-View-Controller (MVC) pattern (Subramanian, 2019). Express also simplifies the development process by its features such as middleware, routing, and integration with databases. All these features making it a powerful tool for building web applications.

Routing: One of Express.js's key features is its robust routing capabilities. This feature allows developers to map the URL paths to a specific controller function. This feature makes handling of incoming HTTP requests efficient. The flexibility of the routing system makes it ideal for developing RESTful APIs and full-fledged web applications (Subramanian, 2019).

Static Content Delivery: Express.js allows for the efficient serving of static files, like HTML, CSS, and client-side JavaScript. It includes minimum configurations, and as a result, developers can deploy static content directly to clients. This leads application performance to reduce server-side processing for these files (Subramanian, 2019).

Middleware: Middleware functions in Express.js allow custom operations to be performed at various stages of the request-response process. This middleware architecture allows for the modular implementation concerns. For example, logging, authentication and error handling. By utilizing middleware developers can build modular components that improve the adaptability and long-term manageability of their applications. (Brown & Smith, 2020).

Database Integration: Express.js integrates with databases ion a smooth way. Through its use of middleware and database drivers, Express facilitates the interaction between the application and the database, allowing developers to store, retrieve, and manipulate data effectively (Subramanian, 2019). This capability makes Express.js suitable for a wide range of data-driven applications.

MongoDB

The MERN stack utilizes MongoDB, a JavaScript-based NoSQL database designed for document storage and schema flexibility (Subramanian, 2017). NoSQL databases is particularly useful for their distinct features. This feature has made them the choice of many large companies. Among the most significant advantages of NoSQL databases are their availability, fault tolerance, and scalability (Shah & Kapoor, 2021).

In MongoDB, core concepts are documents, collections, and the flexible schema approach. This document-based approach minimizes join operations which are common in relational databases. This lead to better performance and decrease the expenses. MongoDB uses JSON-like structures; it integrates with JavaScript. As a result, it makes the development process highly compatible with the MERN stack, where documents interact directly to native data types. In addition, MongoDB's do not follow the schema pattern; this schema-less nature allows flexibility within collections, permitting documents with varying fields, which can be especially useful in the early stages of development or when handling dynamic data models (Shah & Kapoor, 2021).

In MongoDB, a database is a logical unit that holds collection of documents, as a grouping of related records (MongoDB, n.d.). Unlike SQL databases, which rely on foreign keys to maintain relationships between tables, MongoDB does not enforce such constraints. Instead, databases in MongoDB is similar to namespaces for collections, and most database operations, including reading and writing, occur at the collection level. However, MongoDB offers the lookup stage in the aggregation pipeline, which allows for operations similar to SQL joins, combining documents from different collections within the same database (Subramanian, 2019).

The process of creating a database in MongoDB is dynamic. A database is automatically created when the first collection or document is inserted. For instance, issuing the use <db> command selects the specified database, and subsequent operations such as inserting a document into a collection will create the database and collection if they do not already exist (MongoDB, n.d.). Administrative tasks, such as backups, performed at the database level. Each database is treated as a separate entity for connection purposes. Therefore, it is a best practice to maintain all collections related to a specific application within a single database. This leads to simplify management and connections, although MongoDB servers can host multiple databases (Subramanian, 2019).

MongoDB is a document-based database where data is stored in the form of documents (MongoDB, n.d.). These documents, which are analogous to objects, can contain nested structures. Embedded documents and arrays are two examples of theses nested structures. Through this pattern, the need for multiple tables is eliminated. While it would be required in a relational database to represent complex, nested data. For example, in a relational system, an invoice and its items would be split into separate tables, whereas MongoDB stores the entire invoice as a single document (Subramanian, 2017). Documents are structured as field-value pairs. Fields are capable of holding complex data types, including arrays and objects. MongoDB documents are similar to JSON objects, supporting various data types beyond standard primitives, such as dates, timestamps, and binary data (Subramanian, 2019).

2.3 AI in education

When discussing Artificial Intelligence (AI), a broad array of concepts emerges (e.g., machine learning, natural language processing, and computer vision) (Liu, 2016). The focus will be on exploring Machine Learning (ML) and Deep Learning (DL) systems, particularly in the context of their application through APIs integrated into web platforms. These technologies can enhance real-time interactions between tutors and students. And it enables live chat functionalities to facilitate immediate access to answers.

Machine learning is an essential branch of artificial intelligence, and it plays a crucial role in data-driven decision-making (Abu-Mostafa, 2012). It is based on three principal paradigms: supervised learning, unsupervised learning, and reinforcement learning (Brown et al., 2020). Each framework provides different strategies and techniques for model training, based on characteristics of the data and the results (Chellaraj, 2024).

In the supervised learning which involves training a model, historical dataset with is labeled of the inputs and outputs. The model learns the relationship between inputs and outputs by adjusting its predictions based on the ground-truth values (Brown et al., 2020). This approach is widely used in predictive tasks, such as forecasting student performance. For example, ChatGPT is trained on big amounts of text data. This data is with labeled responses and allowing to learn how to generate coherent and relevant replies based on user prompts (Brown et al., 2020).

In the Unsupervised learning there are no labeled outputs, and it identify patterns or clusters within data (Hastie, Tibshirani, & Friedman, 2009). For instance, a clustering algorithm could categorize students based on the characteristics similarity into inform targeted support strategies. These techniques help in broader natural language processing tasks, like organizing text data into topics or understanding hidden structures in user conversations (Jurafsky & Martin, 2020).

Reinforcement learning is based on feedback from actions in a specific environment, such as an intelligent tutoring system. The algorithm learns by rewards or penalties for actions and improving future decisions. This method has gained traction in AI gameplay systems and is being explored in education.

Deep learning (DL), is a branch of machine learning (ML). It employs deep neural networks composed of multiple layers (Qutab et al., 2024). It is used to capture complex patterns and relationships in data (Zhao et al., 2023). These neural networks are inspired by connectionist theories of cognitive science, are adaptable to various tasks through architectures such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformers, each suited to different types of data (e.g., time-series or images) (LeCun, Bengio, & Hinton, 2015; Goodfellow, Bengio, & Courville, 2016). Deep learning has become increasingly effective. It is advances in large datasets, such as the vast corpus used to train models like GPT-3 (Brown et al., 2020). It is also advanced in techniques like transfer learning, which help models adapt to tasks with limited data (Pan & Yang, 2009). Additionally, hardware innovations like GPUs and TPUs have accelerated computation but raised concerns regarding environmental impact and access disparities (Hennessy & Patterson, 2019; García-Martín et al., 2019). Although it contains many capabilities, deep learning faces challenges related to bias, misinformation, and energy efficiency.

One of the valuable tools that connect live chat interface functionalities with natural language processing (NLP) platforms are REST APIs. These platforms enable developers to design conversation scenarios, which allow users to engage with systems through natural language. Such platforms (e.g., DialogFlow and ChatGPT) help to process user queries in various language styles and respond meaningfully. By utilizing their APIs, developers can create conversational platforms that are capable for real-time interaction. A comparison of DialogFlow and ChatGPT highlights their unique features and potential applications in chatbot development and educational environments.

DialogFlow is a popular platform for building conversational agents that is developed by Google. It makes the chatbot development easier by allowing users to define conversation flows through Intent fields. Then map the correspond to specific user inputs. For each turn in the conversation, the developer enters possible user inputs and their respective responses. One of the key advantages of DialogFlow is its ability to generate a Uniform Resource Locator (URL). The URL simulates real-time conversation environments, the same as social media platforms. This instant simulation environment allows developers to test and refine chatbot interactions more easily. The straightforward and accessible interface makes DialogFlow useful for non-expert developers looking to implement chatbots quickly (Shin, Lee, & Noh, 2024).

ChatGPT, on the other hand, is a conversational AI developed by OpenAI. It is based on the GPT-4 architecture. It stands out for its ability to produce responses that resemble human communication. across a variety of contexts. ChatGPT’s flexibility makes it more useful in educational environments. For instance, it can serve as a virtual tutor, answering questions, explaining complex topics, and even assisting with skill development. For example, ChatGPT has been used in language learning experience and during this process, it provides feedback and interactive conversation practice. Additionally, educators can use ChatGPT to automate tasks such as grading, generating study guides, and responding to frequently asked questions. One of the strengths of ChatGPT is that it has capacity to adapt to a wide range of prompts. This feature helps to respond appropriately to highly specific user instructions, such as guiding students through mathematical problems step by step (OpenAI, 2023).

There are no limitations in the platform regarding subject selection for tutors and students in the education systems. As a result, using a predefined conversation approach often requires additional steps and customization. Therefore, ChatGPT was selected focusing on developing effective prompts that can directly enhance the learning experience for students. By using the flexibility of ChatGPT, the goal is to create prompts that can dynamically respond to students' needs across various subjects. And making the learning process more efficient and adaptable to individual requirements.

To design an effective AI tutor, adopting a similar approach to that recommended in OpenAI's ChatGPT documentation for educational interactions is proposed. The prompt will be automated, which means certain student-directed questions, such as asking about their grade level or subject, will be omitted. However, the prompt will maintain its focus on facilitating learning by engaging students interactively.

The final prompt will be as follows:

"You are an upbeat, encouraging tutor who helps students understand concepts of [subject] by explaining ideas and asking students questions. Only ask one question at a time. First, the student asks a question. Then ask them what they know already about the topic they have chosen. Wait for a response. Given this information, help students understand the topic by providing explanations, examples, and analogies. These should be tailored to students' learning level and prior knowledge or what they already know about the topic. Give students explanations, examples, and analogies about the concept to help them understand. You should guide students in an open-ended way. Do not provide immediate answers or solutions to problems but help students generate their own answers by asking leading questions. Ask students to explain their thinking. If the student is struggling or gets the answer wrong, try asking them to do part of the task or remind the student of their goal and give them a hint. If students improve, then praise them and show excitement. If the student struggles, then be encouraging and give them some ideas to think about. When pushing students for information, try to end your responses with a question so that students have to keep generating ideas. Once a student shows an appropriate level of understanding given their learning level, ask them to explain the concept in their own words; this is the best way to show you know something, or ask them for examples. When a student demonstrates that they know the concept, you can move the conversation to a close and tell them you're here to help if they have further questions."

This prompt structure is published by best practices of AI-based tutoring systems by OpenAI platform. The strategy behind this approach is to encourage active learning and critical thinking. This is approaching by guiding students through a series of interactive questions based on the individual, rather than providing them with direct answers. This method not only promotes independent problem-solving skills, but also allows students to build their prior knowledge. The tutor will adjust the depth and complexity of the responses based on the current student's understanding, ensuring the explanations and analogies are appropriate for their level (OpenAI, 2023).

3 Methodology

3.1 Research problem

While there is the growing demand for personalized education, there is a significant gap in the availability of comprehensive software solutions designed to connect students with suitable tutors. Students who seeking to learn a new skill must filter the potential tutors based on subject, budget, educational approaches, and location. This process could be challenging, particularly for students who are trying to find a specialized tutor. Also, academic institutions such as schools and colleges need to find suitable tutors based on their expertise and experiences. Tutors who looking for the increase of their class load and promote their profile through feedback from others, currently rely on their limited network and past students.

In the current way to find a tutor, often students depend on asking for referrals from a small network or using trial and error. This process can be time-consuming and unreliable. Tutors also face many difficulties in managing their work and reaching new students. Most platforms available today do not have the advanced tools tutors which help them to run their classes smoothly. For example, they may not offer flexible scheduling options, accurate reservations, or easy ways to manage students’ progress. Many of these platforms are also hard to use for those tutors who are not very familiar with technology.

Similarly, academic institutions also struggle to find and manage tutors. They need a system that helps them to choose tutors. These searching process usually are based on specific requirements like qualifications, teaching experience, and subject expertise. However, current platforms often lack tools to manage tutor profiles, class schedules, and feedback efficiently. This makes it even harder for schools and colleges to ensure they are providing quality education. These issues show the need for a better, more user-friendly platform to support both tutors and institutions.

3.2 Research objective

The primary objective of this research is to identify and develop an effective tutor-finding website based on this background. This goal includes several key questions that needs to be addressed thorough different dimensions of the platform’s development and functionality:

RQ1. How effective is the website in making connections between students and tutors?

This question explores the overall utility, effectiveness and impact of the platform on users by focusing on ease of use and success rates in finding appropriate matches.

RQ2. In the process of choosing a tutor, what factors influence the students' decision?

This question aims to discover the important factors, such as tutor qualifications, teaching style, and student preferences, that influence students' choices when selecting a tutor.

RQ3. What are the areas that AI can improve the functionality of the platform?

This question aims to investigate the potential roles of AI in improving user experience and efficiency.

RQ4. What are the challenges that teachers face in online sessions, and how can the platform address those?

This question focuses on identifying the common difficulties tutors encounter during tutoring sessions and how the platform could offer solutions, whether through technological features or additional resources.

RQ5. What tools or materials could add value to the online learning experience in general?

This question aims to explore tools or resources, such as interactive learning aids, virtual whiteboards, or other digital educational materials, that can enhance the quality of online tutoring sessions.

3.3 Research design

To explore the research questions, the key step to taken is to first design an appropriate research method. A mixed methods research approach combines qualitative and quantitative approaches

(Morris, 2011). This method is particularly suitable for exploring broad and complex topics that encompass multiple dimensions, which are requiring in-depth investigation (Creswell & Plano Clark, 2011). In the mixed methods research approach allows for the integration of both numerical data and descriptive data. It helps with enabling a more comprehensive understanding of the research problem (Morse, 2003). This approach strengthens the validity of the study by using the strengths of each method while compensating for their respective weaknesses (Teddlie & Tashakkori, 2009). The outcome of the study will be improved by integrating both qualitative and quantitative components, mixed-method approach which allowing for richer, more nuanced findings that contribute more effectively to the research field (Morse & Niehaus, 2009).

In the next step, a review of existing platforms was conducted. UpWork, LiveXP, CoPilot, and TutorOcean are the platforms which provide similar services. This process of comparison helped to identify key functionalities that were either missing or underdeveloped. Considering the potential features that could address common issues and challenges faced by users. Based on this analysis, an initial mockup design was created. This prototype was intentionally kept simple and focused on the core ideas from the literature review and the analysis of the existing websites, rather than detailed features.

In the process of research design, a mockup was refined and developed a more comprehensive design with the required features. In addition, the research questions were addressed through a research design. The platform was created based on two distinct user groups—students who seeking sessions or courses, and tutors who conducting the sessions. Each group of users has unique requirements and usage patterns. These requirements are complex and requiring a deeper understanding of the functionalities that address to both user types. Therefore, the research was divided into two key areas. Quantitative research to understand the effectiveness of the website and the factors influencing tutor selection and qualitative research to gather feedback on the prototype from potential tutors, and to explore their teaching strategies, session challenges, and platform preferences.

3.3.1 Existing systems

In order to identify the current solutions available in the online tutor market, a comparison was conducted among four prominent platforms The aim was to evaluate the strengths and weaknesses of each platform. Thereby discovering a comprehensive understanding of conventional methods used for tutor search. In addition, the review of the existing platforms provided insights into the unique values each of these platforms offer. And identified potential gaps that could be addressed in the new platform design to enhance their effectiveness.

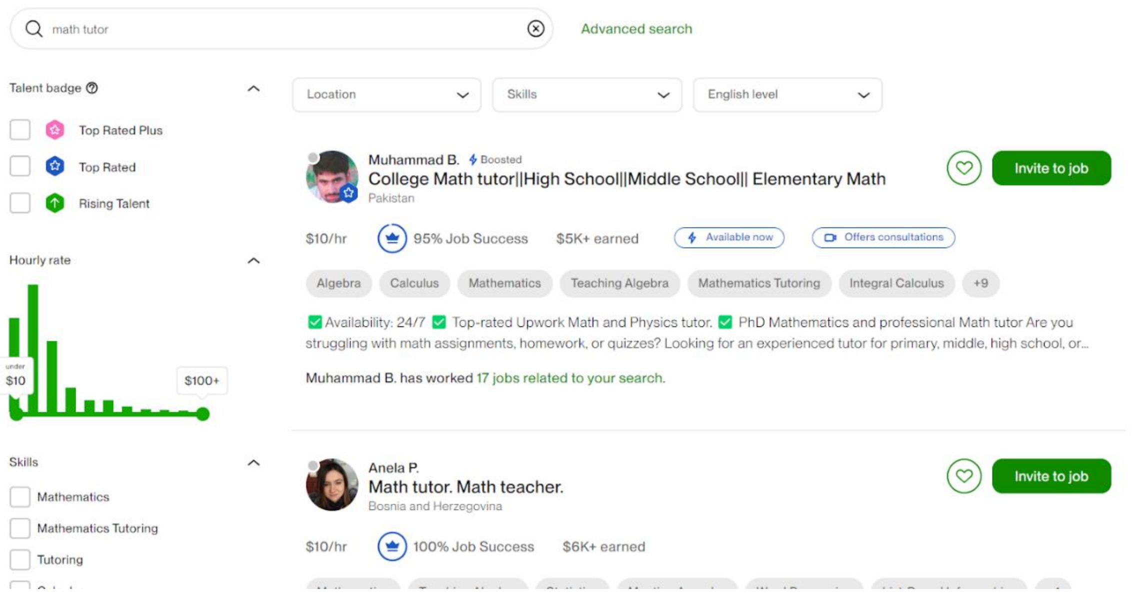
The four platforms examined were UpWork, Livexp, CoPilot, and TutorOcean. The selection of these platforms was based on their popularity, diversity in service models, and number of users within the online tutoring. UpWork and Livexp were chosen based on their wide range of freelance tutoring services. These two platforms are offering both subject based and skill based tutors. On the other hand, CoPilot and TutorOcean were included as they offer specialized focus on personalized tutoring experiences and structured platforms for seamless tutor-student interactions. This diversity in the selection of platforms provided a basis for evaluating various approaches to tutor discovery and engagement.

3.3.1.1 UpWork

Upwork is a well designed online platform that connects freelancers with clients who seeking different services. The services are including web development, graphic design, and writing and etc. The platform provides for the businesses and individuals the opportunity to outsource tasks to the qualified professionals. This has been facilitated through Upwork’s user-friendly interface and its extensive network of freelancers across different sectors. As the gig economy continues to grow, platforms like Upwork have become essential in reshaping the employment landscape by enabling flexible work arrangements for freelancers and offering businesses access to global talent (Kässi & Lehdonvirta, 2018).

One of the key features of Upwork is its robust search functionality. It allows clients to filter search results based on specific criteria, such as experience, skill set, language, location, and hourly rate (Figure 1). This features ensures that clients can efficiently find freelancers who are the best fit for their project requirements, fostering effective collaboration and high-quality deliverables

(Agrawal, Lacetera, & Lyons, 2016). Additionally, Upwork's platform includes review and rating of the clients. This feature adds level of transparency and trust between freelancers and clients. As a result, it brings contribution to improve accountability and service quality.


math tutor

Advanced search

Talent badge

Top Rated Plus

Top Rated

Rising Talent

Hourly rate

under $10

$100+

Skills

Mathematics

Mathematics Tutoring

Tutoring

Location

Skills

English level

Muhammad B. Boosted

College Math tutor||High School||Middle School|| Elementary Math

Pakistan

Invite to job

$10/hr 95% Job Success $5K+ earned Available now Offers consultations

Algebra | Calculus | Mathematics | Teaching Algebra | Mathematics Tutoring | Integral Calculus | +9

Availability: 24/7 Top-rated Upwork Math and Physics tutor. PhD Mathematics and professional Math tutor Are you struggling with math assignments, homework, or quizzes? Looking for an experienced tutor for primary, middle, high school, or...

Muhammad B. has worked 17 jobs related to your search.

Anela P.

Math tutor. Math teacher.

Bosnia and Herzegovina

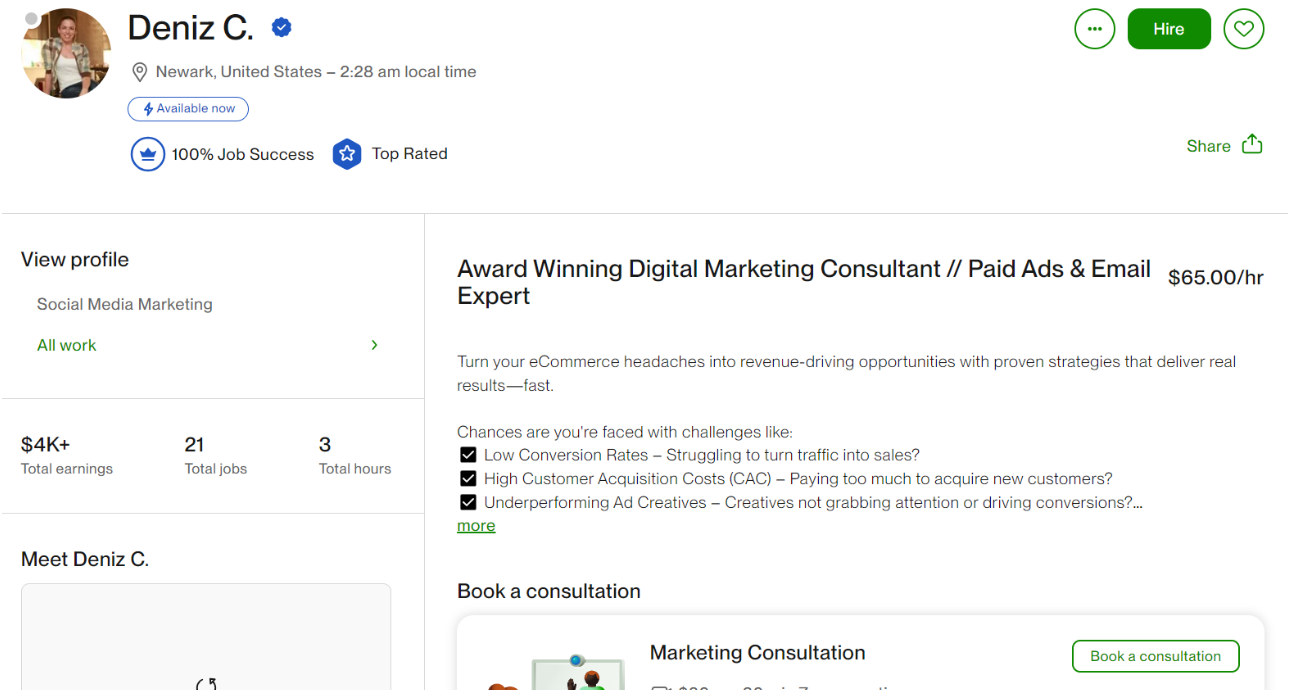
Invite to job

$10/hr 100% Job Success $6K+ earned

Figure 2. Upwork search page

In addition to its traditional freelance services, Upwork has also expanded into educational services. It is providing a platform for tutors and educators. The platform includes a specialized section for tutors to showcase their expertise and connect with students who need academic support. Through Upwork's search functionality, users can browse tutor profiles, compare rates, read reviews. They can also filter results based on the tutor's expertise, location, or teaching language. This is particularly useful for students who are seeking for personalized instruction in specific subjects, test preparation, or language learning (Chen et al., 2020).

The rating system plays a significant role in the tutoring section user experience of the platform. UpWork has considered this feature by allowing students to review and rate their tutors based on their experiences. This feature offers an insight into the tutor's teaching style, their professions state, and effectiveness (Figure 2). This feature allows prospective students to make informed decisions about which tutor best aligns with their academic needs (Huang, Fang, & Huang, 2021). Such systems promote transparency and quality assurance within the platform. Also, it helps to ensure that students receive reliable and effective tutoring services.



Deniz C.

Newark, United States – 2:28 am local time

Available now

100% Job Success Top Rated

Hire

Share

View profile

Social Media Marketing

All work

$4K+ Total earnings

21 Total jobs

3 Total hours

Meet Deniz C.

Award Winning Digital Marketing Consultant // Paid Ads & Email Expert $65.00/hr

Turn your eCommerce headaches into revenue-driving opportunities with proven strategies that deliver real results—fast.

Chances are you're faced with challenges like:
☑ Low Conversion Rates – Struggling to turn traffic into sales?
☑ High Customer Acquisition Costs (CAC) – Paying too much to acquire new customers?
☑ Underperforming Ad Creatives – Creatives not grabbing attention or driving conversions?...
more

Book a consultation

Marketing Consultation

Book a consultation

Figure 3. Upwork tutor's profile page

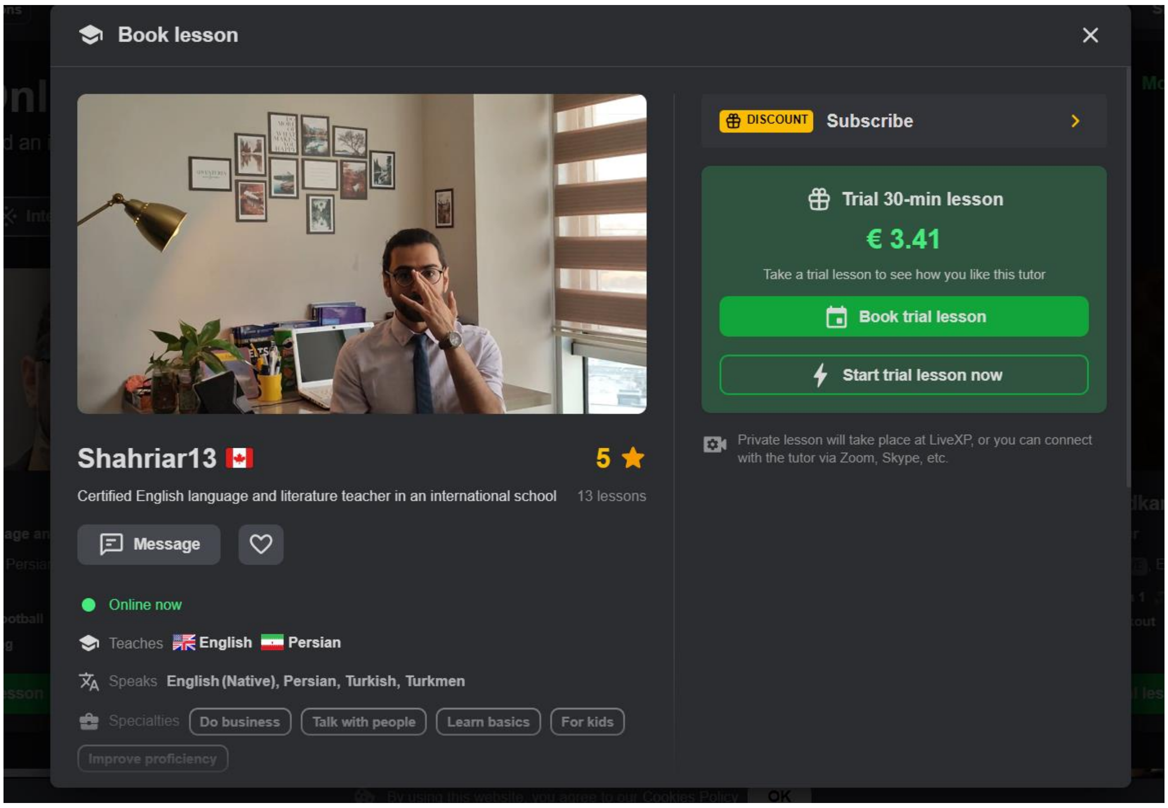
Upwork's search functionality is designed in a way that facilitate efficient tutor-student matching. This is conducted by allowing users to filter tutors based on various parameters, some of these parameters are: subskills within subjects, price range, or availability, students also can find tutors tailored with their unique learning needs (Upwork, 2023). This flexible search feature is complemented by messaging and booking system. In the booking system, which enables students to schedule sessions directly based on available time slots offered by tutors. The simplicity of this process reduces the back-and-forth communication that can often complicate the hiring process, ensuring a streamlined and user-friendly experience (Srivastava, 2019).

Upwork's approach to connecting students with tutors reveals its broader mission in the system. It empowers both freelancers and clients, fostering a global community of collaboration and skill exchange. In this platform, the company not only provides businesses with access to skilled freelancers but also offers students personalized educational support from qualified professionals around the world (Kässi & Lehdonvirta, 2018).

Despite the strong set of features that Upwork offers, there remain areas for improvement. One of the noticeable limitation in this platform is the lack of a live forum. This live communication feature where students can get instant responses, powered by artificial intelligence (AI) can be beneficial. As AI is increasingly playing an important role in education, many platforms are integrating AI-driven features such as chatbots, virtual tutors, and adaptive learning technologies, which provide personalized feedback and real-time support (Chen, Chen & Lin, 2020). These systems allow a better interactive and responsive learning environments. Where students can engage with AI-driven tools for immediate assistance with academic queries (Niu et al., 2022).

3.3.1.2 Livexp

Livexp is specialized in language learning. It is offering a range of features to enhance the tutor-student interaction. One of the platform's key strengths is its tutor profiles. It includes recorder videos of tutors that explain their teaching strategies and approaches (Figure 3). This personalized presentation allows students to gain valuable insights into the pedagogical methods of each tutor, facilitating more informed decision-making when selecting a tutor (Smith & Jones, 2021). This feature provides transparency and allows students to choose instructors whose teaching styles align with their individual learning preferences, a factor that has been shown to enhance the learning experience in language education (Brown & Lee, 2015).



Shahriar13 5

Certified English language and literature teacher in an international school 13 lessons

Online now

Teaches English Persian

Speaks English (Native), Persian, Turkish, Turkmen

Specialties Do business, Talk with people, Learn basics, For kids, Improve proficiency

Trial 30-min lesson

€ 3.41

Take a trial lesson to see how you like this tutor

Private lesson will take place at LiveXP, or you can connect with the tutor via Zoom, Skype, etc.

Figure 4. Livexp tutor’s profile page

Additionally, Livexp integrates a rating and review system. Allowing students to evaluate tutors based on their teaching effectiveness and professionalism through the introduction videos. Such systems of evaluation foster accountability and contribute to maintaining high-quality instruction on the platform (Chen & Peng, 2018). Also student feedback through these ratings plays a key role in assuring the credibility and reliability of tutors. Positive reviews reinforce a tutor’s reputation while constructive feedback helps them improve their services. The use of rating systems in educational platforms has been identified as an essential component of quality assurance in online learning environments (Koutsikouri, Lindgren, & Johanson, 2018).

The platform also streamlines the booking process. It offers a user-friendly interface that allows students to schedule sessions efficiently. This seamless booking system eliminates administrative barriers to save more time. It enables students to focus more on learning and less on logistical coordination (Garrison & Vaughan, 2008). By reducing the complexity of session scheduling, Livexp optimizes the overall efficiency of the tutoring experience, as students can easily access the instructional support they need.

While Livexp excels in providing a high-quality platform for language learning, it contains limitations as it focuses exclusively on language tutoring. Unlike broader educational platforms that offer a wide array of subjects, Livexp is limited to its services to language acquisition, which may restrict its usefulness for learners seeking support in other academic areas (Johnson, 2020). For students who are requiring tutoring in subjects such as mathematics, science, or other disciplines, alternative platforms might be more appropriate.

3.3.1.3 Copilot

Copilot is a mobile application designed to enhance the process of finding sports trainers. The process is through a unique, personalized registration system. Users are prompted to answer questions regarding their fitness goals, preferences, and specific needs during the registration process. This personalized information is then processed by advanced algorithms, which match users with trainers possessing the appropriate expertise to meet their goals (Johnson & Lee, 2020). By automating the selection process, Copilot eliminates the need for manual browsing, making it easier for users to connect with qualified trainers (Smith, 2018). Such automation improves the efficiency of the platform and enhances user satisfaction by streamlining decision-making (Chen & Huang, 2019).

One of Copilot's key strengths is its matching system. It allows users to generate personalized trainer recommendations based on user-provided data (Figure 4). This system reduces time and effort, ensuring that clients are matched with trainers whose expertise aligns with their fitness objectives (Martin & Patel, 2017). Besides this matching system, Copilot also have a smooth booking process. The booking process allows users to schedule sessions directly through the app. This integration of trainer selection and session booking into a single platform not only enhances user convenience but also increases accessibility by simplifying the overall process (Garrison & Vaughan, 2008).

However, a significant limitation of Copilot is its availability on mobile platforms only. This can be limitation for those users who prefer desktop or web-based interfaces, limiting accessibility (Brown & White, 2016). Previous studies have shown that user preferences for device types vary, and individuals who are more comfortable with desktop interfaces may find Copilot’s mobile-only approach limiting (Jones, 2019). Expanding the platform to include other platforms such as web-based access could eliminate this limitation, and improving user engagement and bringing more popularity along users.

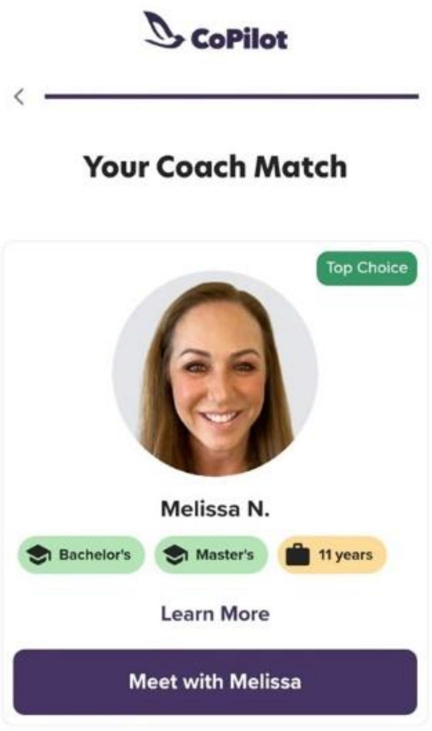


Figure 5. CoPilot coach matching result page

3.3.1.4 TutorOcean

TutorOcean is one of the well-known platforms that is connecting both educational institutions and individuals who are seeking tutoring services. It offers a comprehensive search system designed in a way that meet a wide range of educational needs. The platform includes a user-friendly interface helps students and parents to easily browse through a variety of tutors, categorized by subject to find the tailored matches for specific learning objectives (Johnson & Lee, 2021). Additionally, TutorOcean includes high security principles for user safety and trust. It implements rigorous background checks during the tutor registration process. This provides a secure learning environment for students, providing reassurance for parents and educational institutions that the tutors on the platform are both qualified and trustworthy (Smith, 2019).

Moreover, TutorOcean offers the booking process to schedule tutoring sessions. This approach enhances user experience, reducing the administrative burden of arranging lessons (Chen & Huang, 2020). However, TutorOcean's reach is primarily concentrated on traditional academic subjects such as mathematics, science, and language arts. The ones who are seeking for expertise in more specialized or niche subjects may face challenges, where its coverage may not be as comprehensive (Brown, 2020). While considering the platform's overall effectiveness, this limitation indicates for an area for improvement in expanding subject offerings beyond core academic disciplines.

Each platform offers unique functionalities based to specific niches. Upwork offers in its extensive network platform features like search filters, rating system, and streamlined booking processes. LiveXP which is specializes in language learning, offering personalized tutor profiles with videos, a rating system, and an efficient booking interface. Copilot offers in sports training, a personalized matching system based on user-provided data, seamless booking capabilities, and a mobile-centric user experience. TutorOcean serves educational institutes and individuals alike, offering categorized subjects, stringent background checks, and simplified booking processes.

3.3.2 Phase 1: Quantitative research (Survey)

The survey’s goal is to assess user perceptions regarding the effectiveness of the tutor-finding platform and find the key features that are influencing the decision-making process in selecting a tutor. First, the survey targeted Master’s students in Full-Stack Software Development. In the survey the was to gather insights from individuals with advanced technical knowledge, and asses how they evaluate the platform's functionality and usability. Second, it included individuals without higher education. This helped to understand the perspectives of users with diverse educational backgrounds who might use the platform for personal learning. Finally, the survey reached parents who may use the platform for their children. Focusing on the needs and priorities of this demographic when selecting tutors. The survey was distributed to nearly 100 individuals and received 11 responses (Appendix 1). Having a mix of user perspectives to inform platform development.

The survey consisted of multiple-choice and checkbox questions, conducted in the Google Forms platform. And it was focusing on the relations between variables, such as user satisfaction and factors influencing tutor selection (e.g., qualifications, price, reviews) to understand more details about users’ needs.

The survey results indicate that the tutor searching platforms can hold a significant value for the education sector. Nearly 55% of participants had never used a similar service, this was indicating a gap in the market. Those respondents who had previous experience with tutor-finding platforms, their priority to select a tutor included subject expertise, tutor profiles (particularly certifications and experience), availability, and reviews from previous students. These findings indicate the required features the platform should implement. Such as advanced search options based on subject, tutor availability, and ratings. Additionally, the reservation system is required for tutor availability, and detailed tutor profiles should include solid information about their experience, certifications, and student feedback.

In the survey, it was asked about features that would enhance user experience and effectiveness. Participants prioritized the features but also highlighted other important functionalities. These features included progress tracking and performance analytics, integration with educational resources, and live video chat or virtual classrooms.

Studying AI integration in the platform within the survey and one of the concerns was about the accuracy and security of AI-driven features. Approximately 70% of participants expressed moderate to high trust in AI (rating 3 or 4 out of 5). This result is suggesting that despite ongoing debates about AI in education, implementing the right AI practices could add significant value to the platform. Within the survey, it also explored potential AI features that could benefit the platform. First, participants highlighted the value of personalized education experiences and adaptive learning paths for each individual and their learning styles and pace. Second, they indicated the need for intelligent content recommendations based on their learning objectives for a more effective learning experience. Third, they highlighted the real time feedback and performance analysis to track their learning progress and improving outcomes. Fourth, they indicated their interest in incorporating with virtual tutoring or personalized assistance through chatbots or virtual assistants for the instant support. Fifth, participants suggested integrating simulations and gamified learning methods to increase motivation among the students. Sixth, they highlighted the importance of strong data security and privacy. Since it helps to build trust and protect user information. Finally, participants highlighted the ability to receive real time feedback or responses based on internet resources, with capabilities to interpret user inputs, including images or diagrams.

These insights indicated that how much potential for AI is important to enhance the platform's usability and effectiveness. And showed how the chosen features are aligning with user needs and are implemented with appropriate security measures.

Based on the acquired information indicated in the survey, the initial mock design was updated and enhanced with more detailed features using the Figma.

3.3.3 Phase 2: Qualitative research (interview)

In order to gather a deeper level of feedback a semi-structured interview with five tutors as potential users of the website was conducted.

Tutor participants are from diverse fields and expertise who are utilizing the Figma prototype of the website to gather user insights. The participants represented a range of professional backgrounds to represent a solid sample of potential platform users (Appendix 3). Interviews were held via Zoom, WhatsApp, and Google Meet, with each session lasting around 30 minutes.

The interview was designed based on two themes into two sections:

1. *Teaching experience and platform expectations*: Tutors were asked about describing their teaching experiences, challenges they faced during sessions, and their expectations for a tutor-finding platform. Additionally, they were asked to share their recommendations based on their use of similar technologies.

2. *Prototype feedback*: Then tutors were asked to interact with the Figma prototype, then provide their opinions about its usability, potential features, and overall design. To find the usefulness of the platform specific questions were asked, then collect their opinions on improvements that could be made to the platform to better meet their needs.

The selection of interviewees was intentional, to include a diverse range of subjects (Appendix 2). This diversity allowed gathering more comprehensive data, capturing the nuances required for different teaching strategies, materials, and techniques. Teaching various subjects requires different approaches, and the mode of delivery (online, offline, or hybrid) which are often depending on the subject matter itself. Also, tutors who work with different age groups were included. This age difference consideration was since sessions with younger students pose different challenges and opportunities compared to those with older or more advanced students. By considering these diverse factors, I was able to collect data that is more inclusive and provides insights into the needs of a wider variety of students and tutors.

3.4 Data analysis methods

By utilizing the mixed-methods approach in the research design broad user data was captured. It allowed to understand user preferences through quantitative data and related data as well as understanding context-specific feedback via qualitative interviews.

Quantitative research (survey)

The quantitative data were analyzed by using descriptive statistics (Gravetter & Wallnau, 2013). This summarize and organize data to provide a clear understanding of patterns, trends, and central tendencies. This helped to understand user perceptions of the tutor-finding platform and identify key factors influencing tutor selection. Descriptive statistics are used to describe, summarize, and simplify large amounts of data in a meaningful way, such as calculating measures of central tendency (mean, median, mode) and variability (standard deviation, range) (Gravetter & Wallnau, 2013). Data was collected via Google Forms which were aggregated and analyzed for correlations among variables such as user satisfaction, qualifications, and security. Descriptive statistics, including frequencies and percentages were calculated to summarize the responses.

The key findings based on survey data are as follows:

A significant portion of participants (54.5%) reported never tried online tutor-finding platforms. While nearly half of the respondents have experience with these services, a majority do not. This is showing a gap in familiarity or accessibility. While indicating an untapped market segment for online tutoring platforms.

Among users who had experience with online tutor-finding platforms, the three most valued features were: search filters for specific subjects or skills, detailed tutor profiles with qualifications and experience, and availability and scheduling options. Additionally, users valued reviews and ratings from previous students. This highlights the essential features that a tutor-finding website should include.

Participants also indicated a high valuation of verified credentials and trustworthiness in tutor profiles. In addition, they the usefulness of tracking learning progress and seamless scheduling options. Along with these three primary features, virtual classrooms and access to educational materials within the platform were noted as beneficial tools for academic success. Security was a notable feature as secondary. This is suggesting that while users are concerned about security, they prioritize educational and scheduling functionality.

Respondents also rated their trust in AI on a scale from 1 to 5. A majority (63.6%) of respondents indicated a high level of trust (4 out of 5) in AI algorithms. They indicated they are tended to ask for guidance on tutor websites from AI. In comparison, 27.3% reported a moderate level of trust, and only 9.1% expressed very high trust (5 out of 5). This distribution suggests general openness to AI-based features for tutor guidance, though some reservations remain. The absence of low-trust responses reflects an overall positive perception of AI. This can bring opportunities to enhance trust through transparency and demonstrations of AI's reliability. The platform could safely integrate AI-driven features such as personalized recommendations or live chat. However, to address potential concerns, it would be advantageous to provide clear explanations of AI’s use on the platform and to emphasize data security and privacy in AI implementations.

The platform should prioritize developing AI algorithms, in condition that accurately reflect each user’s specific needs and learning styles. 81.8% of respondents indicated that tailored AI recommendations significantly impact their trust. Additionally, data privacy and security measures, along with prior positive experiences with AI, were selected by 36.4% of respondents. This is underscoring the importance of protecting user data to prevent concerns about misuse or mismanagement. Although transparency and user feedback integration were selected by only 27.3% of participants, they are valued as secondary trust factors.

The survey also revealed that users value AI capable of aligning with their specific learning needs and goals. They indicated that AI can directly enhance learning effectiveness. Transparency in tracking them learning journey and the ability to measure improvement over time were also regarded as beneficial features. Although data security and interactive elements are secondary, they remain important to many users, particularly when sensitive learning data is involved. Ensuring data privacy is essential to establishing trust.

The relatively low interest in virtual tutoring may suggest that users still prefer human interaction for personalized guidance, even as they utilize AI-driven tutoring tools.

Based on the survey data, four key values emerged: Usability, Usefulness, Security, and Trust. By using Atlas software, features were categorized and a summary of features as presented in Table 1 achieved. This is categorizing how each feature can add value to the platform.

Table 1. Key Perceived Value of Features in Online, AI-Based Tutor Platforms

#	Value	Feature(s)	Indications
1	Usability	Usage of Online Tutor Platforms	54.5% of participants have not tried online tutor-finding platforms; which indicates a potential market gap
2	Usefulness	• Search filters • detailed tutor profiles • availability and scheduling options • reviews • Progress tracking	
3	Security	• verified tutor profiles • protection of personal info	Users highly value verified tutor profiles Data privacy/security measures (36.4%) are important
4	Trust	• AI for tutor recommendations • AI in personalized recommendations • Transparency and user feedback in AI • Human interaction in tutoring	The majority (63.6%) trust AI algorithms. 81.8% believe AI recommendations tailored to individual needs build trust. Transparency and feedback integration are less prioritized (27.3%) but still valued.

			Users value AI for aligning content to specific learning needs/goals, enhancing learning effectiveness Lower interest in virtual-only tutoring, indicating a preference for human interaction combined with AI.

Qualitative research (interview)

Interviews were transcribed verbatim from the interview recordings. Then all responses were captured in detail to ensure that nuances in participants' experiences were captured. This transcription process is essential for preserving the authenticity of responses. This also allows us to closely examine both content and context. Transcriptions were formatted into question and response pairs for an easier reference.

Transcripts as well as written notes taken during the interviews. Then they were analyzed by using Atlas software. The analysis included coding through using labeling segments of text that addressed similar topics, concepts, and experiences. By reviewing the codes, the following challenges emerged as themes from the interviews.

Student Engagement Challenges: Many tutors found that engaging students is one of the key challenges. This can be especially more important for the younger students with limited attention spans or adult learners who might have pre-existing biases or distractions. “Keeping younger students focused, especially after a long school day, is a struggle.” (Fundamental Mathematics Tutor, personal communication, September 15, 2024).

Effective Engagement Strategies: tutors use techniques that they found effective in maintaining student interest, such as gamification, using real-world examples, and incorporating multimedia elements. “Using games and real-world examples helps keep students interested in the material.” (ASP.NET Programmer Tutor, personal communication, October 3, 2024)

Feedback and Evaluation: Feedback was a recurring theme, with tutors expressing concerns about inconsistent student feedback mechanisms. Some felt evaluations did not accurately reflect their teaching due to the level of content difficulty. “Sometimes, students rate easy concepts highly, while more challenging ones lead to negative feedback.” (Civil Engineering Professor, personal communication, September 22, 2024).

AI as a Teaching Tool: AI was viewed as beneficial but with limitations. Tutors valued AI for tasks like creating examples and offering structured responses. However, there were wary of its impact on critical thinking, especially among younger students. “AI tools like ChatGPT help check explanations, but giving direct answers could harm critical thinking.” (Fundamental Mathematics Tutor, personal communication, September 15, 2024).

Preference for Customizable Learning Options: Tutors expressed a need for flexibility in how lessons are delivered. Many suggested the platform should offer both live and recorded sessions. This allows students to choose based on learning style and schedule. “Offering both live and recorded sessions would allow students more control over their learning experience.” (Architecture Professor, personal communication, September 29, 2024).

3.5 Research ethics

This research considered the integrity, reliability, and credibility of the research process. It was based on the ethical guidelines of JAMK University of Applied Sciences (JAMK University of Applied Sciences, n.d.). Considering respect for participant rights, data privacy, and academic integrity.

Informed Consent and Participant Rights

All participants who were involved in the surveys and interviews were informed about the purpose of the research, the level of voluntary, and their right to withdraw at any time. They were also Informed and gained the consent before data collection. In other words, they were informed about their roles and the use of the information provided.

Confidentiality and Data Protection

All the data that was collected during the research was anonymized. This approach helped to protect participants' privacy. Personal identifiers were removed, and data was stored securely to prevent unauthorized access.

Academic Integrity

The research process was conducted based on the academic honesty. By ensuring the originality of the content, accurate citation of all sources, and avoiding any form of plagiarism, data fabrication, or manipulation. All secondary data sources were appropriately referenced following academic standards.

Use of Artificial Intelligence (AI)

AI tools, including ChatGPT, were utilized to support various aspects. ChatGPT was used to optimize English language usage, improve language clarity, and correct grammatical errors in the written content (OpenAI, 2023). Additionally, ChatGPT was used to assist with debugging code during the development phase of the tutor-finding platform. To ensure the efficient problem-solving and code optimization. Furthermore, Atlas software which is empowered with AI was used for decoding and analyzing interview transcripts to identify key themes and insights effectively. The use of AI was conducted ethically, with human oversight to validate and interpret AI-generated outputs.

4 System design

4.1 User interface and user experience

User Interface (UI) and User Experience (UX) are fundamental components of digital product design. Each of those are serving a distinct yet complementary role. UI design emphasizes visual aesthetics and interactive elements such as buttons, icons, typography, and colors, ensuring the interface is visually appealing and intuitive for users to navigate (Mulligan, 2021). In contrast, UX design focuses on the overall user journey, prioritizing ease of use, efficiency, and satisfaction in achieving user goals. Together, UI and UX contribute to a cohesive, user-centered design that enhances both functionality and engagement (Mulligan, 2021).

Numerous UI/UX design methodologies are currently employed, including user-centered design (Hasim et al., 2019). The design thinking approach which is a widely recognized and popular method was chosen for the tutor-finding platform. Design thinking includes five stages (Figure 6):

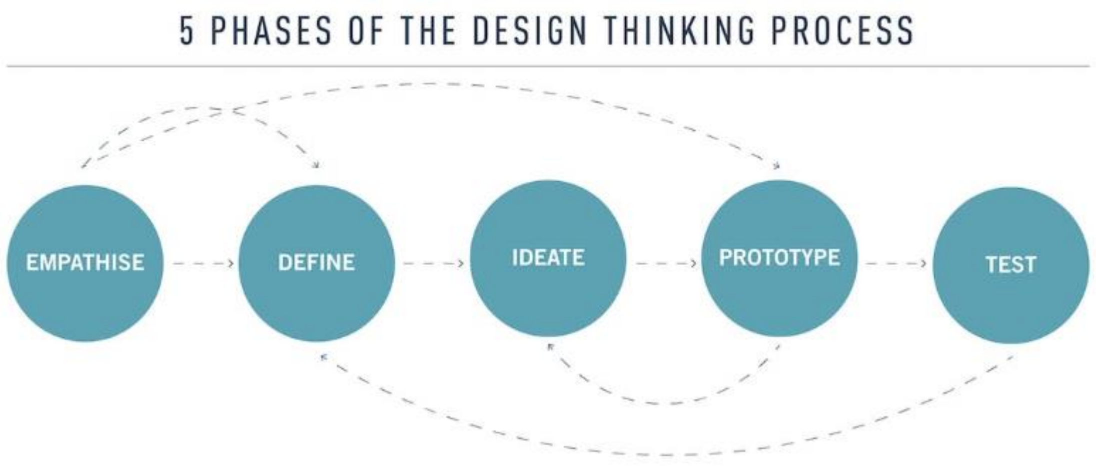


Figure 6. Steps of the design thinking process

1. Empathize: Gathering information to understand user needs through interviews and re-search.

2. Define: Analyzing collected data to clarify key problems and user requirements.

3. Ideate: Brainstorming solutions to address identified issues.

4. Prototype: Designing an interface prototype using tools such as Figma.

5. Test: Gathering user feedback to evaluate the prototype's functionality and usability, ena-bling further refinement (Wijayanti & Tanone, 2021).

The implementation of the design thinking methodology was followed in a non-linear process as shown in Figure 6. Each phase required iterative reassessment and refinement of previous steps to ensure alignment with project goals and completeness.

Empathize

At this phase focus was on understanding the challenges faced by potential users. For students, the primary concern was identifying suitable tutors. And tutors need efficient session management tools. To gain a deep insight, a comprehensive review of existing literature, market solutions, and best practices was conducted. Through that process it revealed the gaps and opportunities for improvement. Surveys and interviews provided additional depth and indicated direct insights into user problems and expectations.

Define

The data that gathered during the previous step, empathize phase, was organized and analyzed to actionable insights. By categorizing this information, a prioritized set of requirements and user needs was indicated. By assessing the findings in an iterative way such as cross-referencing survey results with interview findings and analyzing similar platforms, further refined these priorities. For instance, functionalities like AI integration were revealed with a higher priority than background checks based on user needs.

Ideate

By the gained insights from the define phase, a simple mockup was created by using Balsamiq software. This mockup helped to visualize concepts and facilitate discussions. The mockup included essential features such as user authentication (Figure 7), tutor profiles (figure 8), search functionality, and booking systems. At this level, tried to avoid excessive detail to maintain adaptability. Feedback from surveys underscored the importance of AI-assisted functionalities over secondary features like background checks. This iterative phase promoted focused discussions on practical solutions.

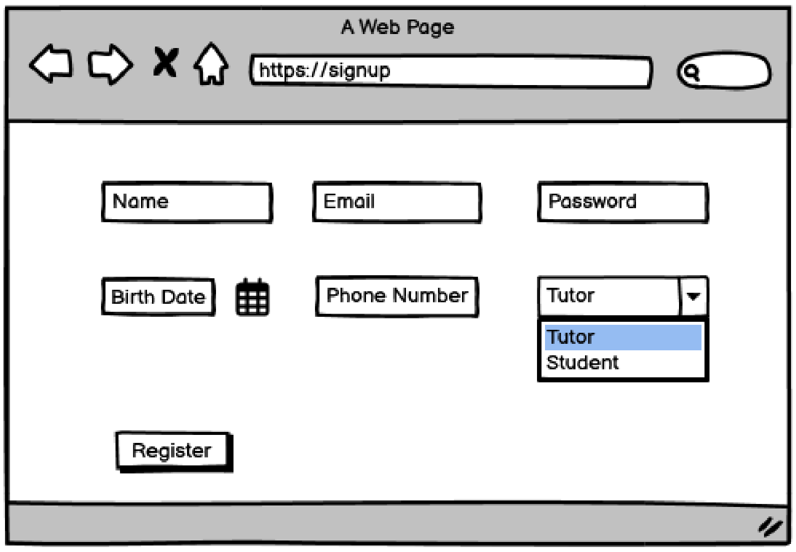


Figure 7. Signup page

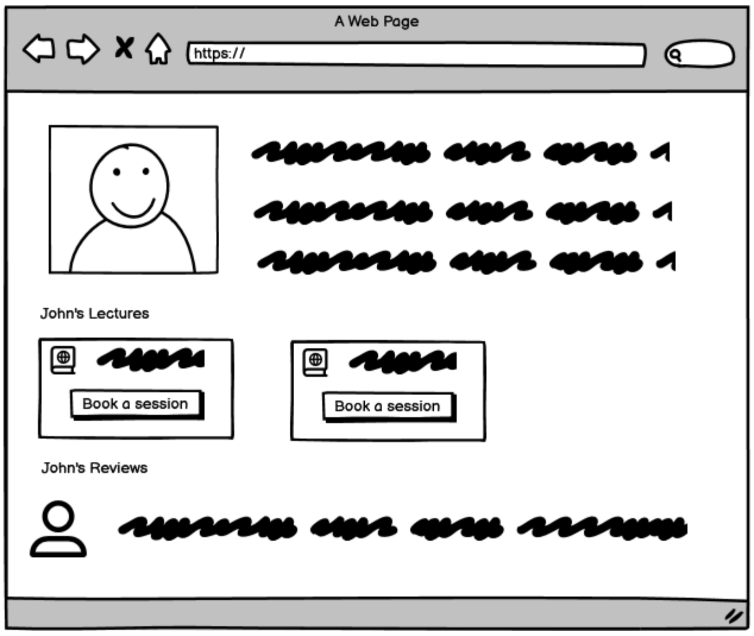


Figure 8. Profile page

Prototype

Based on the previous mockup, a higher-fidelity prototype was developed using the Figma. This design demoed key functionalities that were identified during the earlier phases. These revealed features were including advanced search filters, booking systems, and user authentication. In the design process the Pedoz-Pet-Service-Web-UI-Kit (Anita, 2021) template was used as a base design kit. This template was chosen based on its simplicity and adaptability. Complex features such as AI chatbot integration were deprioritized. This prioritization helped to deliver more high-priority elements like profile page (Figure 9 and Figure 10), Search (Figure 11 and Figure 12) and authentication, enhancing clarity and usability.

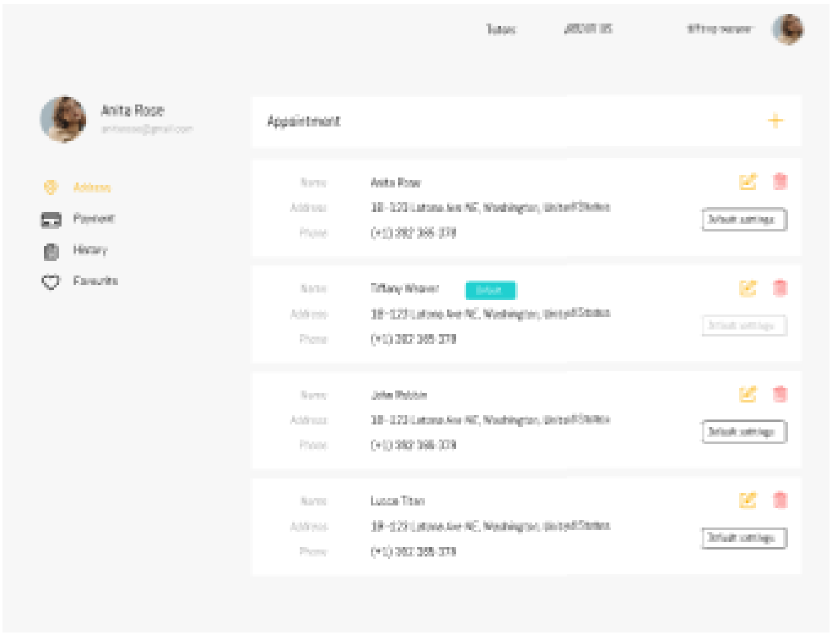


Figure 9. Profile page

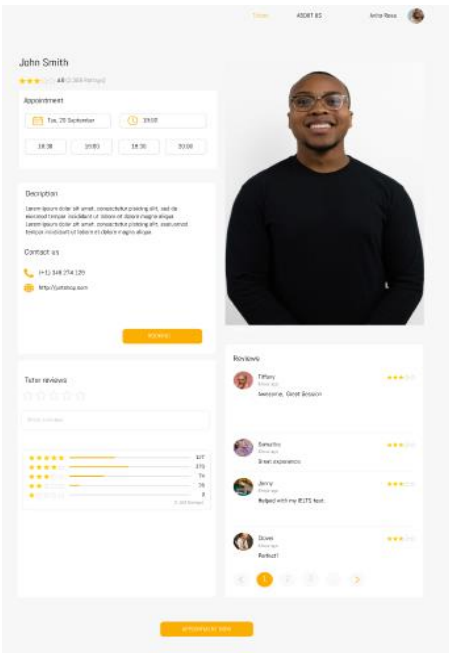
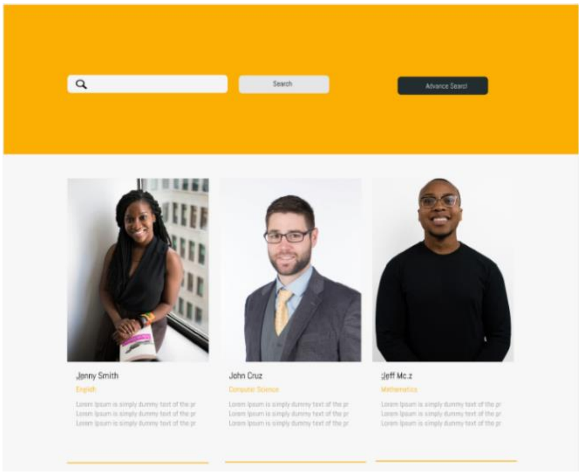
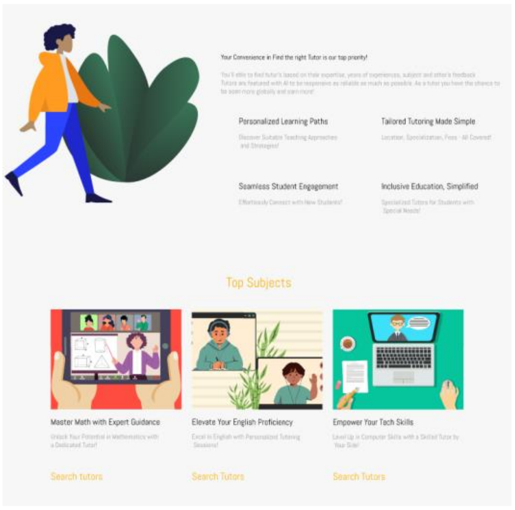


Figure 10. Tutor's profile page



Lorem Ipsum is simply dummy text of the pr
Lorem Ipsum is simply dummy text of the pr
Lorem Ipsum is simply dummy text of the pr

Figure 11. Tutor search page


Personalized Learning Paths
Tailored Tutoring Made Simple
Seamless Student Engagement
Inclusive Education, Simplified
Top Subjects
Master Math with Expert Guidance
Elevate Your English Proficiency
Empower Your Tech Skills
Search tutors
Search Tutors
Search Tutors

Your Convenience in Find the right Tutor is our top priority!

You'll able to find tutor's based on their expertise, years of experiences, subject and other's feedback Tutors are featured with AI to be responsive as reliable as much as possible. As a tutor you have the chance to be seen more globally and earn more!

Discover Suitable Teaching Approaches and Strategies!

Location, Specialization, Fees - All Covered!

Effortlessly Connect with New Students!

Specialized Tutors for Students with Special Needs!

Unlock Your Potential in Mathematics with a Dedicated Tutor!

Excel in English with Personalized Tutoring Sessions!

Level Up in Computer Skills with a Skilled Tutor by Your Side!

Figure 12. Partial of home page

Testing

The final phase was testing the usability of the design. It was done with prospective users to validate the platform's UI and UX. Feedback from this stage confirmed the usability of essential functionalities and highlighted areas for improvement. For example, user input led to deprioritizing background checks and removing this feature from the feature set. The Figma prototype was iteratively refined to align with user needs and expectations. Ultimately through this iterative approach, optimized the overall user experience.

4.2 Requirement gathering

By integrating the findings from the survey and the interviews and framing it in the design thinking approach, a range of potential functionalities for users was found, that users would find them useful and valuable. However, to develop a minimum viable product (MVP) it needed to narrow down and prioritizing the key features. Survey participants ranked learning content recommendations as one of the highest priorities. However, interviews revealed that tutors emphasized the importance of an optimized version of ChatGPT for both students and tutors.

As a result, the MVP will focus on using AI in the platform for content recommendations and integrating ChatGPT. The implementation will be tailored to deliver personalized examples and explanations. This functionality will form the platform's core feature. Additional valuable features, such as automated reservations, AI-generated feedback for assignments, and daily practice exercises, will be considered for implementation. Ensuring that the platform evolves to meet user needs over time.

Based on the findings from the literature review, survey, and interviews, several key features were discovered and established priorities. These features were considered to create an initial version of the website that addresses the needs of both students and teachers. This prioritization was based on the resources and time to allocate efficiently and to focus the most critical functionalities. For this purpose, the MoSCoW method (Must Have, Should Have, Could Have, and Won’t Have) was used. The MoSCoW method is a widely recognized technique for prioritizing software requirements. It categorizes features into four groups: Must Have, Should Have, Could Have, and Won’t Have. Must Have features are essential to the system’s functionality, while Should Have requirements are important but less urgent. Could Have features represent improvements that are not essential, and Won’t Have items are excluded from the current scope due to resource constraints. This method ensures that essential features are prioritized while allowing flexibility for future development.

MoSCoW method provides a structured framework for balancing project scope and timeline constraints. For instance, Must Have requirements typically indicates the minimum functionality needed to render a system operational. While Could-have features can be deferred without compromising the project’s success. This approach enables stakeholders to make informed decisions about prioritization, and optimizing resources while ensuring that critical needs are met. It is precious in environments characterized by dynamic requirements or limited resources (Kostev, 2023; Tudor & Walter, 2006). The results of the MoSCoW method suggest the following features for the system:

Must Have:

- Learner profiles: Profiles with knowledge levels, which can be integrated with AI prompting to provide personalized feedback.
- Booking system: A system that allows users to book sessions based on availability.
- Tutor search: A system that allows users to search for tutors.
- Data security: Strong data protection to ensure user information is safe.
- User roles: Distinct user roles for students and tutors (users can have both roles).

Should Have:

- AI assistant for real-time feedback: AI-based live chat responding to student queries, tailored to their background and profile.
- Advanced search functionality: Filters for availability, skill set, language, location, price, and ratings.
- Tutor booking slots: Students can book based on time slots offered by tutors.
- Tutor introduction videos: Tutors should upload videos about themselves, including background checks for safety.

- Review system: Feedback from students on tutors.

- Admin role: A separate administrative user category to manage system operations.

- Cancellation policies: Customized cancellation terms for each course offered.

Could Have:

- Learning styles in profiles: Include learning styles in student profiles for enhanced personalization.

- Emotion recognition: AI can adjust learning based on a student’s psychological state, e.g., integrating online video tools.

- Curriculum sequencing management: AI generates custom learning or curriculum paths based on individual student needs.

- Automated tutor-student matching: Matches students with appropriate tutors based on their goals and their profiles.

- Profile comparison: Ability to compare tutors’ rates, reviews, location, language, and certifications.

- Interactive educational materials: Include video chat, charts for achievements, and virtual classrooms.

- AI content recommendations: Suggest content based on students' learning objectives.

- AI-assisted course material creation: AI, such as ChatGPT, helps design course materials based on learning needs.

- Real-time feedback and analytics: AI-driven progress tracking and performance insights.

Based on this list of priorities, a Minimum Viable Product (MVP) was defined. That indicated the domain of design and implementation. An MVP allows a development team to validate key assumptions by releasing a basic version of the product to early adopters, facilitating rapid feedback and iteration (Lenarduzzi & Taibi, 2016).

Based on the time and resources, the items in the 'must have' and some of the items in the 'should have' were chosen for the implementation.

4.3 Front-end design

React components

React components are the fundamental building blocks of user interfaces. They allow developers to break down the UI into independent, reusable pieces. The components can be managed in isolation (React Documentation, 2021). Components in this project were categorized into two main groups: components and pages.

In addition to these two groups of components, in the project, two third-party components of Material UI and the Formik were used.

Components:

FieldComponents: These are custom fields that are designed for user input flexibility such as *CountryField*, *PriceField*, *SearchField*:

EditReservation: A component within the profile page that uses global states such as token and user for user-specific functionalities.

Layout: Acts as a reusable skeleton for pages, incorporating consistent elements like headers and footers.

RequireAuth: This component ensures users are redirected to the login page if they are unauthorized.

Pages:

- *Signup*: Uses Redux's register function to handle global state updates during user registration.
- *Login*: Implements Redux-based login forms to manage authentication globally.
- *Profile*: Utilizes global user and token states and integrates local states for managing user-specific sessions and reservations.
- *Home*: A navigational page connecting users to different functionalities such as viewing sessions and accessing the profile page.
- *SessionList*: Displays a filtered list of sessions with state variables for advanced search filters (subject, budget, country).
- *SessionDetail*: Requires authentication relying on Redux states for user and token to display session details securely.

Material UI Components:

Material-UI (MUI) is a popular React component library that implements Google's Material Design guidelines. There are components such as Avatar, Paper, Button, and Grid in this library that facilitate the rapid development. These components help with visually consistent and responsive designs. MUI components are highly customizable, ensuring adherence to project-specific themes (MUI, 2021).

Formik:

Formik is a library for building and managing forms in React applications. It simplifies form validation, state management, and error handling. Using Formik ensures cleaner code and reduces the risk of validation bugs (Formium, 2021).

State management

Effective state management is critical in designing modern web applications. It ensures smooth data flow and efficient component interactions, instead of passing all the data component by component. During the design phase, it is essential to distinguish between global states and local states to create a scalable and maintainable application. Global states typically handle data shared across multiple components, such as user authentication, while local states manage component specific logic and temporary data. This separation allows to reduce redundancy, and enhance the modularity of the application (Abramov, 2015).

Redux:

Redux is a state management library for JavaScript applications. It is designed to centralize the application's state in a single store. This structure ensures predictable state updates by using actions and reducers, which describe how the state changes in response to events. By adopting Redux, developers can achieve a unidirectional data flow and simplifying debugging and enhancing scalability, especially in larger applications (Abramov, 2015).

Redux was implemented for managing the global state of the application, particularly for user authentication. Key global states such as the user's authentication token and user data were maintained in Redux. Making token and user data accessible across all components. This approach eliminated the need to pass props manually through multiple component layers, improving maintainability and efficiency.

The token and user states were stored in Redux. These states were crucial for determining whether a user was logged in and for controlling access to certain protected routes (Figure 13). For instance, these states used in RequireAuth component to redirect unauthorized users to the login page and Profile page to display user-specific information, such as session history and reservations.

Local state:

Redux was used to manage global states while individual components relied on local states for handling specific functionalities. For the Session List Component (SessionList), local states managed advanced search filters such as subject, budget, and country. It allowed users to refine their searches. Additionally, another local state tracked the list of available sessions fetched based on these filters. Also in the Edit Reservation Component local states were utilized to manage temporary data during the process of editing a reservation. It guarantees that the form remained responsive without interfering with the global state. On the Profile Page, local states like sessionList and reservation dynamically managed the data displayed on the page, while global states, such as token and user, were used for authentication and user identification. Lastly, for the Signup and Login Pages, Redux actions facilitated user registration and login processes, with the global user and token states being updated upon successful authentication.

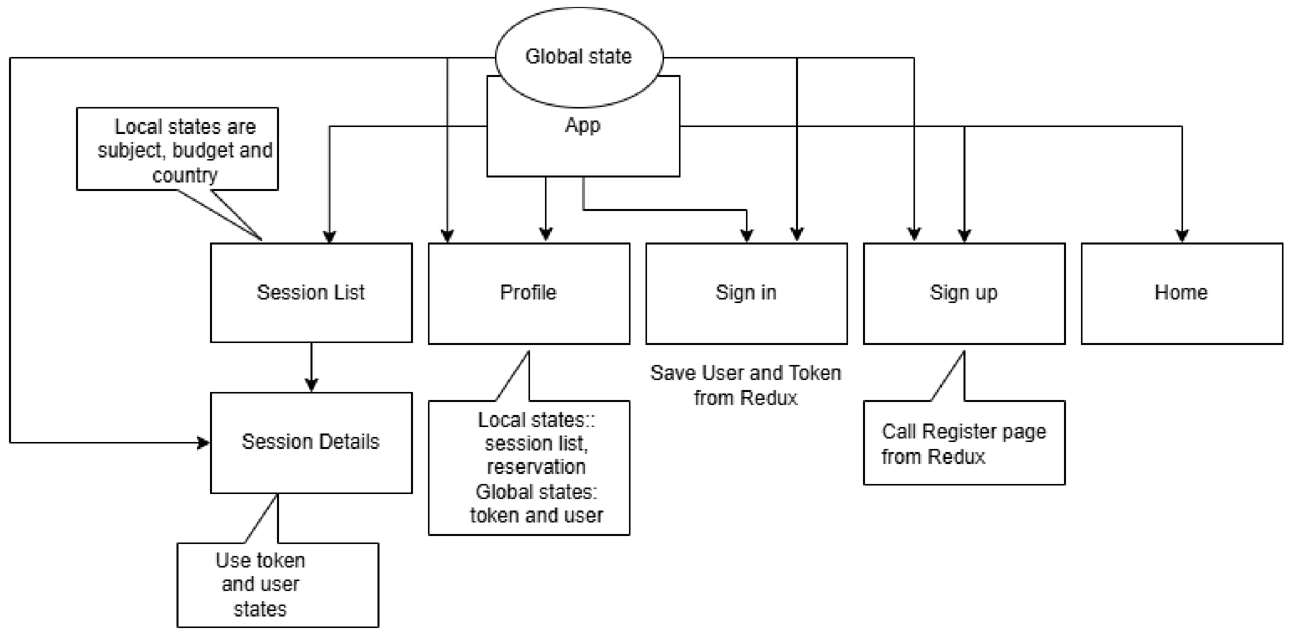

Global state
App
Local states are subject, budget and country
Session List
Profile
Sign in
Sign up
Home
Session Details
Local states:: session list, reservation Global states: token and user
Save User and Token from Redux
Call Register page from Redux
Use token and user states

Figure 13. State management

Routing

After designing the components, it is required to specify the routes of the application's navigation paths. These routes are divided in to two groups, public and protected. The public pages are accessible to any unauthorized users. And the protected ones are only accessible by registration and having a valid token.

- Public Routes: These are accessible to all users and include:
  - */login* → Login page
  - */signup* → Registration page
  - */* → Home page
- Protected Routes: Require user authentication, managed through the RequireAuth component. Unauthorized users are redirected to /login. Protected paths include:
  - */profile* → Displays authenticated user profiles.
  - */session/:id* → Shows session details for a specific session.
  - */session/new* → Allows authenticated users to create a new session.

4.4 Back-end design

The backend of this project is designed to support the core functionalities of the tutoring platform. They are including user authentication, session management, AI chatbot interaction, and reservation handling. By following a modular approach using middleware, RESTful API endpoints, and a structured database schema, the backend ensures smooth client-server communication, data consistency, and maintainability throughout the application lifecycle (Fielding, 2000; Holmes, 2018).

API endpoint

The backend follows RESTful principles to organize API endpoints. This brings clarity and consistency in how resources are created, read, updated, and deleted (CRUD operations). Each endpoint corresponds to specific functionalities, such as user management, session booking, and profile updates. This approach simplifies client-server interactions and enhancing maintainability (Fielding, 2000).

The backend structure of the project includes various API routes for the main functionalities. These functionalities are supposed to be implemented in MVP as the first phase, including user management and authentication, retrieving sessions in searching, working with AI chatbot and reservation (Table 2).

Table 2. Tutor finding API's

Route	**method**	**Description**	**Middleware**	**Access**
/uploads	Get	Serves static files.	*express.static*	Public
/api/v1/chatbot	Get/Post	Interacts with the AI chatbot.	*authenticateUser*	Authenticated
/api/v1/auth	Post/Patch	Create and Update user's details including profile picture	None	Public
/api/v1/sessions	Get	Retrieves public session listings.	None	Public

/api/v1/sessions	Get/Patch/Delete	Updates, or deletes sessions.	*authenticateUser*	Authenticated
/api/v1/reservations	Post/Delete	Manages user reservations	*authenticateUser*	Authenticated

Middleware

Express middleware is employed to support a modular and layered design. It helps with enabling a clean separation of concerns. Middleware functions in the project perform tasks such as logging request data, validating inputs, authenticating requests, and managing errors. This architecture improves maintainability by adding new features or modifying existing ones while it does not disrupt the core structure. Middleware is executed sequentially for precise control over the flow of requests and responses through the application (Express.js documentation, n.d.).

Authentication Middleware: JSON Web Token (jsonwebtoken)

Authentication is implemented using the jsonwebtoken library. This provides robust security for user authentication and authorization. JSON Web Tokens (JWTs) (jsonwebtoken Documentation, 2023) are compact, URL-safe tokens that encapsulate user identity and claims in a secure manner. During login, the server generates a JWT and sends it to the client. This create JWT includes the token in subsequent requests for access to protected resources. On the server side, the middleware decodes and verifies the token. Then validate the user's identity. This ensures that only authenticated users can access restricted endpoints, such as viewing profiles or making reservations. The stateless nature of JWTs makes them highly scalable for modern web applications (Boulton, 2019; O'Neill, 2020).

Error Handling and Validation Middleware: http-status-codes

Error handling is mainly using the http-status-codes library. Which standardizes HTTP status responses in the application. This middleware assigns meaningful status codes, such as 400 Bad Request or 401 Unauthorized to indicate the errors. The library simplifies the process of returning responses for invalid inputs or unauthorized actions. By pairing it with Express's error-handling the middleware improves debugging and user experience by providing informative error messages (Kurniawan, 2015). Validation middleware ensures that inputs are sanitized and conform to expected formats and reducing potential vulnerabilities and improving data integrity.

Logging Middleware: Morgan

Logging is a core feature for monitoring and debugging during development. The morgan middleware is integrated into the application to log HTTP requests. It is including methods, endpoints, status codes, and response times. In the development environment, morgan uses the "dev" preset, which provides concise and colored output. It will make it easier to identify issues in real-time. Logging helps in tracking application usage, detecting unusual activity, and diagnosing problems, all of which are essential for maintaining application reliability (Holmes, 2018).

Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing (CORS) middleware is used to address the client and server running on different ports. It is a common scenario in modern web development as CORS allows the server to indicate which domains are allowed to access them. In the implementation the middleware ensures that requests from the frontend React application (running on a separate port) can successfully interact with the Node.js backend. Without CORS, browsers block such requests for security reasons. By configuring CORS, the application maintains security while enabling seamless client-server communication (Krohn, 2021).

Database schema

The application employs MongoDB. It is a NoSQL database, as its primary data storage system. To interface with MongoDB, the project utilizes Mongoose, an Object Data Modeling (ODM) library for Node.js. Mongoose provides a structured approach to define schemas for the application's collections. A schema serves as a blueprint for the database. It is enforcing data integrity through validation rules. For example, schemas define the structure for entities such as users, sessions, and reservations. Then specified the required fields, default values, data types, and relationships between documents. Mongoose simplifies CRUD (Create, Read, Update, Delete) operations by offering an abstraction layer. This layer allows developers to interact with MongoDB using JavaScript objects and methods (Holmes, 2018).

The ERD for the tutoring platform (Figure 14) highlights the relationships and attributes of the core entities in the application: User, Session, Reservation, and AIAssistant. Below is a description of each entity and their relationships:

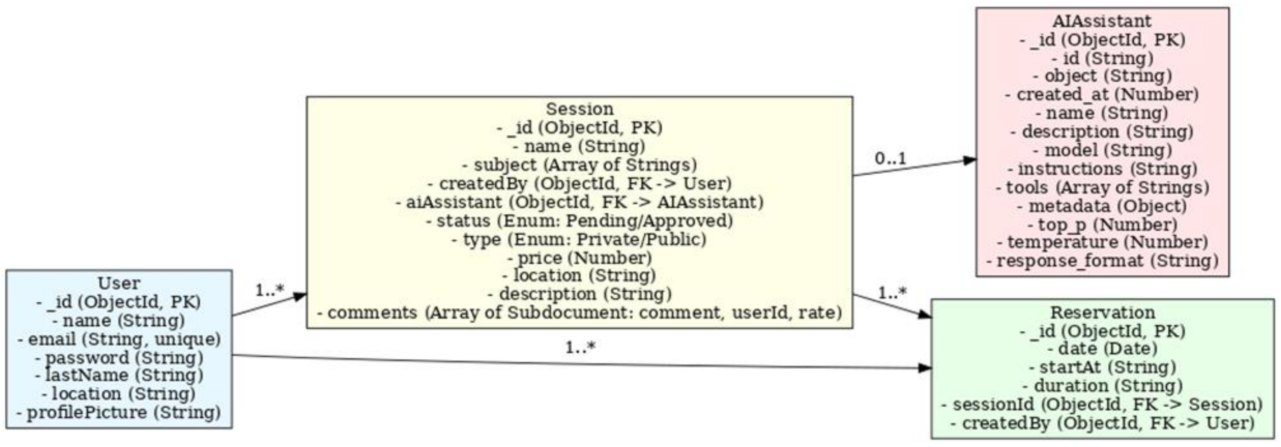


Figure 14. ERD of tutor finding

1. *User*:

   Represents users of the platform, including tutors and students.

   Key attributes: *name, email, password, lastName, location, and profilePicture*.

   Relationships:

   One-to-Many relationship with Session: A user can create multiple sessions (createdBy field in Session).

   One-to-Many relationship with Reservation: A user can create multiple reservations (createdBy field in Reservation).

2. *Session*:

   Represents tutoring sessions created by tutors.

   Key attributes: *name, subject, type, price, location, description, status, and comments*.

   Relationships:

   Many-to-One relationship with User: Each session is created by a single user (createdBy).

   One-to-Many relationship with Reservation: A session can have multiple reservations (sesionId in Reservation).

   One-to-One optional relationship with AIAssistant: Sessions may optionally use an AI assistant (aiAssistant field).

3. *Reservation*:

   Represents bookings made for sessions.

   Key attributes: *date, startAt, duration*.

   Relationships:

   Many-to-One relationship with Session: Each reservation is linked to a single session (sessionId).

   Many-to-One relationship with User: Each reservation is created by a single user (createdBy).

4. *AIAssistant*:

   Represents AI assistant configurations associated with sessions.

   Key attributes: *id, name, description, model, instructions, tools, metadata, temperature, and response_format*.

   Relationships:

   One-to-One optional relationship with Session: A session can optionally use one AI assistant.

5 Implementation

The implementation of the project was based on a structured approach. It was divided into two distinct phases to ensure incremental development and efficient testing.

In the first phase the primary focus was on designing and implementing the frontend using React. To test the user interface and interactions effectively dummy data was used as a placeholder for the backend API. Two key libraries, json-server and json-server-auth facilitated the creation and management of this mock data.

First, json-server which is a simple Node.js-based tool, allowed developers to generate a full mock REST API by providing a JSON file, enabling rapid prototyping of the front end without the need for a fully functional backend during the early stages of development (Marinho, 2018). Second, json-server-auth extended json-server by adding support for authentication and authorization to the mock API. Then it was enabling to tests of user-specific data access and simulated authentication processes within the application. Dummy data for sessions, reservations, and accounts was stored in JSON files. They were mirroring the actual structure of the MongoDB database. This approach provided a straightforward and efficient way to test the application's UI, workflows, and integration with mock API endpoints.

In the second phase, the backend was developed using Node.js and Express.js. In this phase the dummy data was replaced with a fully functional RESTful API connected to a MongoDB database. Backend implementation was included defining schemas, creating API endpoints, and using JSON Web Tokens (JWTs) for authentication. With the backend in place, the frontend was updated to interact with the live API, a seamless user experience.

5.1 Version control and repository

Throughout the development, Git was used for version control. It helped to track changes, manage features, and ensure collaboration readiness. The project's repository is hosted on the GitLab Labranet platform of JAMK University of Applied Sciences. The Gitlab repository is accessible via the link. Using Git ensured efficient branching strategies, making it easier to track progress and revert changes when necessary.

5.2 Project setup

Frontend

Create React App was used to initialize the project and streamline the development process. Several essential packages were installed to handle state management. Such as handling, validation, and API integration. They helped with efficient functionality and user experience.

Create React App (CRA) is a tool that is officially maintained by the React team. It eases the set up of the modern React applications with minimal configuration. It generates a pre-configured project structure with essential dependencies. These dependencies are including Webpack, Babel, ESLint, and testing libraries such as Jest. This boilerplate eliminates the need to manually set up complex build configurations, allowing developers to focus on coding the application logic (Facebook, 2016).

Create React App scaffolds the project by creating directories and files, such as public (for static assets) and src (for application logic). It sets up a single Webpack configuration in the background that handles file bundling, CSS preprocessing, and live development server functionality.

By using create-react-app gained advantages like rapid project initialization with zero configuration, pre-installed development tools like hot reloading and testing frameworks, optimization for production builds with minification and code splitting, and easy extensibility via react-scripts.

The package.json file in a react project is a central configuration file that manages project metadata, dependencies, and scripts. It acts as a blueprint for the application by specifying the versions of installed libraries and enabling the extension of the project with additional Node packages.

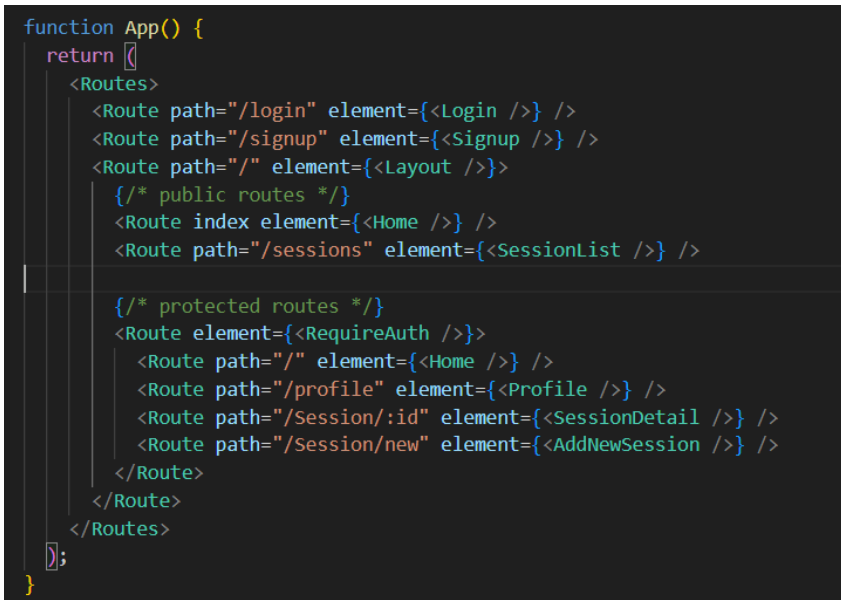
During the project setup, React Router was installed as one of the key dependencies. React Router is used to define the application's navigation and routing structure. By using the npm install, developers can integrate third-party libraries and tools to meet the application's functional requirements and improve the development process.

React-Router

One of the initial packages integrated during the development process was *React Router*. It was an instrumental in defining the application's navigation framework, helping with establishing page routing based on the predefined design system.

React Router is a robust library which is designed to manage navigation in Single-Page Applications (SPAs). Unlike traditional server-rendered applications, React Router enables dynamic rendering of components. In the rendering process there is no require full-page reloads, thereby enhancing performance and improving the user experience.

React Router offers several key features that make it a robust library for managing navigation in Single-Page Applications. First, it supports *dynamic component rendering*, which allows components to update dynamically based on the URL and it does not require full-page reloads. This functionality ensures a smooth and responsive user experience by enhancing performance. Second, it provides *support for nested routing*. This feature is enabling to create complex layouts through nested routes. This feature facilitates the reuse of components and improves modularity within the application. Third, React Router includes *parameterized routes*. It simplifies the handling of dynamic segments in URLs, such as /user/:id. This capability is beneficial for creating user-specific pages or resource-based navigation. It is also streamlining the development process and improving the platform's versatility. Finally, it *streamlined Multi-View Navigation,* by organizing application navigation into multiple views, React Router ensures seamless transitions between pages. It is aligning with the project's design system and usability goals.



```
function App() {
  return (
    <Routes>
      <Route path="/login" element={<Login />} />
      <Route path="/signup" element={<Signup />} />
      <Route path="/" element={<Layout />}>
        {/* public routes */}
        <Route index element={<Home />} />
        <Route path="/sessions" element={<SessionList />} />

        {/* protected routes */}
        <Route element={<RequireAuth />}>
          <Route path="/" element={<Home />} />
          <Route path="/profile" element={<Profile />} />
          <Route path="/Session/:id" element={<SessionDetail />} />
          <Route path="/Session/new" element={<AddNewSession />} />
        </Route>
      </Route>
    </Routes>
  );
}
```

Figure 15. Tutor finding routers

After installing React Router package, defined the application's core navigation structure. This setup included routing for primary pages, such as the home page, login, and profile (Figure 15). Each route corresponds to a React component, ensuring that the application's structure aligns with the intended design and functionality. This dynamic routing mechanism contributes significantly to the scalability and maintainability of the project.

Server setup

The server for the project was set up by using Node.js. This process began with initializing the project using the npm init command. This command generates a package.json file. A core configuration file used to manage metadata, dependencies, and scripts for the project. The package.json file enabled the installation of essential Node packages and other configurations needed to run the backend efficiently.

Server File and ES6 Modules

The main server logic is contained within a file named server.js. It serves as the entry point for the application. ES6 modules were used instead of the traditional CommonJS module system. ES6 modules, introduced with ECMAScript 2015 (ES6). It allows for cleaner syntax and better support for modern JavaScript features, including import and export keywords for managing dependencies and modularity.

This configuration allowed the use of the following syntax:

```
// ES6 Modules

import express from 'express';
```

In contrast, CommonJS uses:

```
// CommonJS Modules

const express = require('express');
```

The ES6 module system is more aligned with modern JavaScript standards and ensures compatibility with the latest libraries and tools (Axel Rauschmayer, 2015; Flanagan, 2020).

Nodemon

To enhance the development process, Nodemon was installed. Nodemon is a Node.js tool that automatically restarts the server whenever changes are detected in the codebase. This eliminates the need for manually stopping and restarting the server during development and thereby it increased the productivity.

After installation, a new script was added to package.json to run the server using Nodemon:

```
"scripts": { "start": "nodemon server" }
```

To run the server with Nodemon:

```
npm start
```

Nodemon improves the development experience by reducing downtime and streamlining the iteration process (Holmes, 2018; Gackenheimer, 2015).

dotenv

The dotenv package was installed to manage environment variables securely. Environment variables store sensitive information such as database connection strings, API keys, and other configuration data. This configuration builds outside the main codebase. And this approach enhances security and flexibility by separating sensitive data from application logic. This loads variables from a *.env* file into the process.env object, making them accessible throughout the application.

The importance of using. env file is for security and flexibility. It helps to reduces the risk of accidental exposure during version control or deployment. Also Environment variables allow for easy configuration changes between different environments (e.g., development, staging, production) without modifying the application code (Krohn, 2021).

To prevent sensitive data from being exposed in the version control system, the .env file was added to .gitignore file.

Database setup

To manage and store the application's data, MongoDB Atlas was chosen as the database solution. MongoDB Atlas is a cloud-based database service that provides a scalable and secure environment. It is suitable for deploying and managing MongoDB clusters. The steps taken to configure the database are outlined below:

1. Create an Atlas Account: Registered for a MongoDB Atlas account to gain access to the cloud-based database management service.

2. Deploy a Tier Cluster: A shared cluster with the free tier was deployed to accommodate the project's requirements during development.

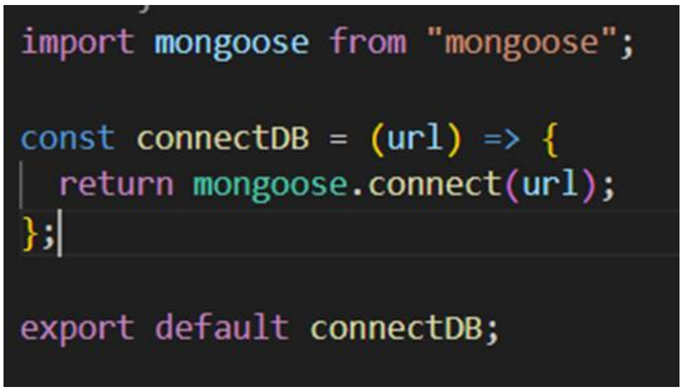
3. Add Connection IP Address to the Whitelist: The local machine's IP address was added to the access list to allow secure connections to the cluster.

4. Create a Database User: A database user was created for the cluster. This user was assigned specific credentials to securely access the database and perform operations.

5. Connect to the Cluster: The connection string provided by MongoDB Atlas was used to link the application to the database cluster. This string contains the cluster's address, database user credentials, and other connection details.

6. Insert/View Data in the Cluster: Data was inserted into the cluster directly through MongoDB Atlas's web interface during initial tests, allowing for easy inspection and verification of database functionality.

To integrate MongoDB with the project, a db repository was created. This file and a connect.js file was added to implement the database connection using Mongoose.



```
import mongoose from "mongoose";

const connectDB = (url) => {
  return mongoose.connect(url);
};

export default connectDB;
```

Figure 16. Database connection configuration

This function takes the database connection URL (obtained from MongoDB Atlas) and connects the application to the database. By using Mongoose, the project benefits from schema validation, simplified CRUD operations, and better data modeling.

In the server setup, the connectDB function was imported and called. If the database connection was successful, the server was configured to start running.

5.3 Front-end implementation

The frontend implementation was developed using TypeScript with React. TypeScript is as an additional setting for JavaScript. It introduces static typing to identify type-related errors during the development process rather than at runtime. This reduces bugs, enhances code readability, and improves overall maintainability (Bierman, Abadi, & Torgersen, 2014). Additionally, TypeScript provides better tooling support for IDEs, such as auto completion and inline documentation, which streamlines the development process.

Folder structure

The frontend project was organized in a modular and scalable folder structure. Below is an overview of the folder structure and its components:

- *package.json*: It manages the project's dependencies and scripts. It defines the required packages for the frontend, including React, Redux Toolkit, Axios, Formik, Yup, and React Router, as discussed earlier.

- *tsconfig.json*: This file contains the TypeScript configuration. It is specifying compiler options and project settings such as the output directory and strict typing rules.

The project's core src folder contains all the files required for building the application. The key subfolders and files are as follows:

- *index.tsx*: The entry point of the application. This file renders the App.tsx into the DOM using React's ReactDOM.createRoot method.

- *App.tsx*: The main component of the application, which defines the overall structure of the application.

- *Assets*: This directory Stores static assets such as images, logos, and icons.

- *Pages*: Contains files for each distinct page of the website, such as the login page, profile page, and session list page. These represent the core views of the application.

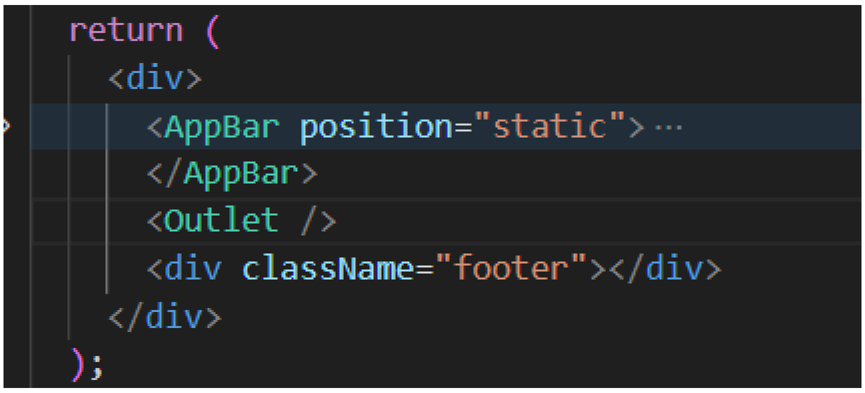
- *Components*: Includes reusable components such as headers, footers, and form inputs that can be used across different pages.

- *redux*: Contains slices created using Redux Toolkit, which manages the application's global state. Slices define reducers and actions for specific parts of the state, such as user authentication or session data (Redux Toolkit Documentation, 2020).

- *types*: Includes custom TypeScript type definitions for the project, ensuring consistent data structures across the application.

Layout

The Layout Component was implemented to standardize the application's structure. It helped to avoid code duplication as the structure of the layout component contains a reusable component. This component is representing the top navigation bar of the website as well as the Outlet component (Figure 17). The Outlet component is provided by React Router, which renders the appropriate page content based on the current route. By using <Outlet />, the layout structure (header and footer) is defined once and reused across all pages, ensuring consistency and reducing redundancy (React Training, 2016).



```
return (
  <div>
    <AppBar position="static">···
    </AppBar>
    <Outlet />
    <div className="footer"></div>
  </div>
);
```

Figure 17. Layout component

Redux and Redux toolkit

Redux Toolkit (RTK) is the official toolset for Redux. It is designed to simplify state management in React applications. It addresses common challenges such as excessive boilerplate code, configuration complexity, and repetitive reducer logic. One of its primary advantages is the *simplified syntax for defining state slices and actions.* It made possible through utilities like createSlice. Additionally, RTK enhances the *developer experience* by including built-in middleware for immutability checks and debugging tools. It helps to streamline the development process. Moreover, it provides integrated *support for handling asynchronous requests* through features like createAsynThunk. It is making it easier to manage asynchronous actions and APIs efficiently (Abramov et al., 2015).

The Redux Toolkit was employed to manage the global state related to user authentication, including user details and token management. Additionally, RTK facilitated handling API requests, enhancing the overall state management architecture. The implementation of the Redux Toolkit was structured into four key steps.

1. Defining the API Slice:
   The apiSlice served as the centralized feature for managing API interactions using Redux Toolkit's createApi. This abstraction simplified the management of API endpoints and provided reusable hooks for making API calls within React components.

2. Store Configuration:

   The store is the central repository for the application's state. Using the configureStore utility provided by Redux Toolkit. The project initialized the Redux store, configured middleware, and combined reducers.

3. Defining API Endpoints in authApiSlice:

   Specific API endpoints for user authentication were defined in the authApiSlice using the injectEndpoints method. This approach allowed the integration of new endpoints into the central apiSlice.

   The key endpoints included:

   Login Mutation: Managed user login by sending user credentials and receiving an access token from the server.

   Register Mutation: Facilitated user registration by sending user details to the backend for account creation.

   Managing Authentication State in authSlice:

   The authSlice was responsible for managing the state of user authentication, such as storing user details and access tokens. By utilizing the createSlice utility, this logic was encapsulated in a clean and modular manner.

5.4 Back-end implementation

The backend development for this project involved three core components: Controllers, Routers, and Models. These components work together to handle application logic, route requests, and interact with the database.

Routing and Controllers

The routing mechanism in Express.js provides a method for mapping specific URLs to corresponding controller functions. These functions are responsible to implement application logic. By organizing route-specific logic into separate controller functions, the codebase becomes modular and easier to maintain (Express.js Documentation, 2023).

Routes were created to handle various HTTP requests (e.g., GET, POST, PUT, DELETE). Each route file imported specific controller functions to execute logic. These routes were then imported in the server.js file using the Express middleware method app.use(routePath, router).

Database Models and Schema Validation

MongoDB stores data in collections of documents. While Mongoose provides schemas as blue-prints to define the structure of these documents. The schemas were based on the initial database design created during the planning phase.

To ensure data integrity, the Validator library was used to validate the schema field. The library offers pre-built functions. It is common for use cases such as email and URL validation. Allowing a robust and sanitized input handling before saving data to the database. Validator's integration improves the reliability of stored data by preventing invalid entries (Zakas, 2012).

Error handling

Two key libraries were used for managing errors in the backend:

1. express-async-errors:
   This package streamlines error handling in asynchronous route handlers. It is eliminating the need for repetitive try-catch blocks, and ensures that unhandled errors are automatically forwarded to the Express error-handling middleware. This mechanism is improving both developer efficiency and code readability.

2. http-status-codes:
   The http-status-codes package was implemented to standardize HTTP status codes throughout the application. By replacing hardcoded status codes with descriptive constants (e.g., StatusCodes.BAD_REQUEST). It helped the codebase to be improved the readability and consistency in API responses.

Security: Hashing Passwords with bcryptjs

To enhance security user passwords were hashed before being stored in the database by using the bcryptjs library. Hashing mechanism ensures that passwords are not stored in plaintext. As a result, it eliminates risks in the event of a database breach.

First, bcryptjs generates *secure hashed passwords* using salting techniques. In this technique adds randomness to the hash and protect against precomputed attacks. Second, the library provides *functionality to verify hashed passwords during the login process*. This verification enables giving access to authenticated users of the platform. Finally, bcryptjs guards against *dictionary and brute-force attacks* by increasing the computational cost of generating hashes. Making it significantly more challenging for attackers to compromise password security. These measures collectively strengthen the overall security of the application.

Middleware and Pre-Save Hooks in Mongoose

Mongoose provides middleware hooks to inject logic during specific stages of model operations (e.g., save, remove, update). The pre-save middleware hook (pre('save')) was employed to decode passwords before storing them in the database.

The pre-save hook was triggered during both the creation and update operations in the user controller. This ensured that sensitive information, such as passwords, was securely hashed during every save operation. Thereby It will improve the overall security of the application.

JSON Web Token (JWT)

JSON Web Token (JWT) is an open standard (RFC 7519) used for securely transmitting information. This transformed information conducted between parties as a JSON object. It is compact, self-contained, and digitally signed, ensuring both the integrity and authenticity of the transmitted data (Jones, Bradley, & Sakimura, 2015). JWTs are primarily employed in authentication and authorization.

The structure of a JWT consists of three distinct parts, separated by dots. First, the header contains metadata, including the type of token and the signing algorithm used. Second, the payload holds the claims such as user ID or roles. This body data represents the data being transmitted. Third, the signature that ensures the token's authenticity. It is conducting by verifying that it has not been altered during transmission, thereby adding an essential layer of security.

In the implementation phase, JWTs were utilized to manage user authentication. After successful login, a JWT is generated for the authenticated user. This token is sent to the client and then stored securely. For subsequent API requests the client includes the token in the HTTP request headers (e.g., Authorization: Bearer <token>). This is allowing the server to validate the user's identity and authorize access to protected resources.

Mongoose allows developers to define reusable functions. These reusable functions or instance methods are on document instances. These methods are associated with the schema and can access the document's properties, providing flexibility for handling model-specific logic (Mongoose Documentation, 2023).

An instance method named createJWT was implemented in the UserSchema to encapsulate the logic for generating JWTs. This method relies on the jsonwebtoken library to create a signed token that includes the user's unique identifier.

By using jsonwebtoken library method named sign was implemented. This was created a signed JWT with a specified payload, secret key, and optional configuration options such as expiration time.

To validate the JWT provided by the client, an authenticateUser middleware was implemented. The middleware ensures that only requests with valid tokens are granted access to protected resources (van Zandwijk & Pauwels, 2023, p. 73). Within this middleware, by calling verify method from jsonwebtoken package library, verify the authenticity of a token and decodes its payload if valid. If invalid, an error is thrown.

AI assistant

An AI assistant is a software application that is designed to execute tasks or services for users using artificial intelligence and natural language processing (NLP). These systems can interpret user input, perform computations, provide recommendations, and respond conversationally to user queries (OpenAI, n.d.-a).

OpenAI's framework for building AI assistants provides tools to integrate natural language capabilities into applications. This framework supports customization, enabling the assistant to perform domain-specific tasks, and facilitates interaction through a well-documented API (OpenAI, n.d.-a).

In the Tutor-Finding implementation the AI assistant's API was integrated. Using AI provide support for students who seeking tutoring services. The implementation utilized OpenAI's framework to facilitate intelligent conversations in real time between students and the AI assistant. However, the financial payment system requires the use of an API key but it was not implemented due to project constraints. Additionally, the frontend did not include features for enabling chat interactions between students and the AI assistant.

The implementation of the AI assistant was focusing on researching and defining requirements to simulate functionality. By understanding the use cases and user needs, this phase aimed to provide a conceptual demonstration of how the AI assistant could operate within the platform (OpenAI, n.d.-c).

An AI assistant is created by default when a session is initialized. This assistant is designed to provide educational support tailored to the needs of the user and based on tutor's profile behavior is defined by specific instructions. These instructions guide the assistant's responses and interaction style. It will ensure it aligns with the pedagogical goals of the platform (OpenAI, n.d.-a). The created AI assistant object is stored in the database. To ensure persistence integration into the session workflow.

In the API a Thread represents a sequence of conversational interactions between a user and an AI assistant. It maintains context and allowing the AI assistant to generate responses based on the flow and history of the conversation. Threads are crucial for dynamic and coherent exchanges, as they provide continuity across multiple messages (OpenAI, n.d.-b).

A Message refers to a single unit of communication within a thread. It can be sent by the user or generated by the assistant. Each message contributes to the context of the conversation, forming the basis for subsequent interactions (OpenAI, n.d.-b).

The implementation of the project includes the following steps:

Session Validation: The session and user reservation are verified to ensure the user has the appropriate permissions to access the assistant.

Assistant Creation: An AI assistant object is instantiated with predefined instructions. These instructions determine the assistant’s tone, purpose, and interaction style. While focusing on providing engaging, guided learning support.

Thread and Message Handling: A thread is created to initiate a conversation. User messages are sent to the assistant within the thread, and the assistant generates context-aware responses.

Response Processing: The system retrieves and organizes the assistant's replies, which are then displayed to the user.

6 Assessment and improvement

The tutor-finding was design and implemented to address the core requirements of the potential users of platform. The adoption of the MERN stack provided a foundation for scalability and ease of development. It includes user authentication, advanced searching (Figure 18) and reservation (Figure 19). Additionally, the inclusion of AI-assisted features demonstrates an innovative approach to enhancing user interaction and learning experiences while discovered this gap in the similar platforms. However, certain aspects of the design and implementation present opportunities for refinement and further enhancement to improve performance, usability, and scalability.

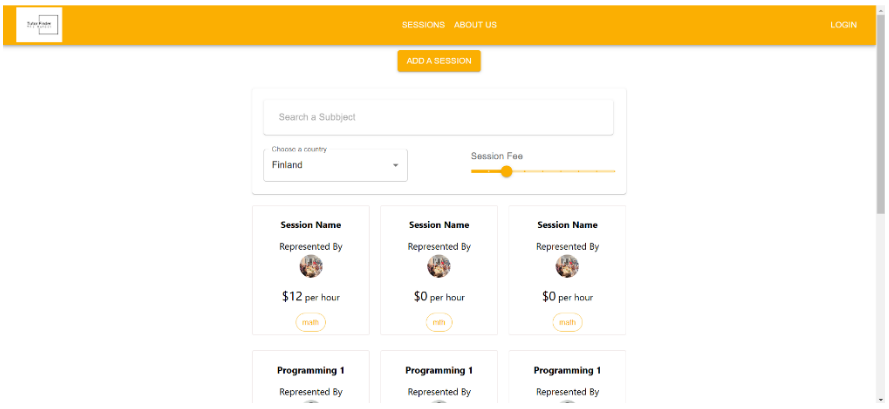
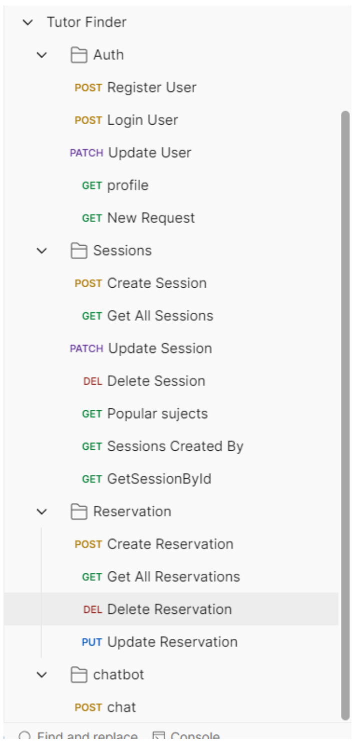


Figure 18. Advanced searching from implemented tutor finding platform



```
Tutor Finder
  Auth
    POST Register User
    POST Login User
    PATCH Update User
    GET profile
    GET New Request
  Sessions
    POST Create Session
    GET Get All Sessions
    PATCH Update Session
    DEL Delete Session
    GET Popular sujects
    GET Sessions Created By
    GET GetSessionById
  Reservation
    POST Create Reservation
    GET Get All Reservations
    DEL Delete Reservation
    PUT Update Reservation
  chatbot
    POST chat
Find and replace   Console
```

Figure 19. List of APIs implemented and retrived from Postman application

6.1 Usability and user experience

The usability and user experience of the tutor-finding platform have been implemented with a user-centered approach. The project leveraged design thinking principles to address user needs. However, certain aspects of the design require further improvements.

First, the design process included a single round of feedback using Figma prototypes. This brings limited opportunities to iteratively improve the user interface (UI) and user experience (UX). By expanding user testing to multiple phases and receiving feedback from diverse user groups could provide deeper insights into potential usability issues. In addition, the current UI could benefit from enhanced styling. Such as consistent use of borders, colors, and typography, to improve aesthetics and accessibility.

Additionally, the MOSCOW requirement prioritization approach efficiently allocated resources to essential features. It excluded lower-priority functionalities due to time and resource constraints. A notable gap in the implementation is the absence of a frontend chat interface for the AI assistant. Although the backend implementation supports this feature, its absence in the frontend limits the overall user experience. Also, the AI assistant requires payment for usage, and the system does not yet include a financial payment API. It can be critical for supporting monetized services. Addressing these gaps would enhance the platform’s functionality and usability.

6.2 Integration of features

The backend architecture supports most of the platform’s key functionalities, such as user management, session creation, and AI assistant interactions. The modular design of the backend, implementing controllers, routers, and models helped with maintainability and a clear separation of concerns. Additionally, the use of libraries like express-async-errors and http-status-codes simplifies error handling and maintains consistent response codes.

However, some areas require further improvement with feature integrations. For instance, the API endpoints could be tested by unit tests and integration tests. Tests are needed to check the edge cases, such as invalid session IDs or unexpected assistant behaviors.

On the frontend side, the app is built with a component-based structure using React. This makes it easier to reuse code and build things in pieces. This type of design helps the platform scale and be more efficient. Redux is used to manage the state. That helps keep track of stuff like auth tokens and user info without too much confusion between components. The current setup also makes it easier to add new features later without breaking things.

6.3 Scalability and performance

The tutor-finding platform demonstrates by its adoption of the MERN stack. The use of MongoDB's NoSQL database also supports horizontal scaling. It handles increased data loads and concurrent user interactions. Node.js, as a non-blocking, event-driven runtime that provides high performance and scalability for handling concurrent API requests (Holmes, 2018). It is combined with React's component-based structure. As a result, the MERN stack enables the platform to efficiently manage multiple users, sessions, and interactions with AI assistants.

Research suggests that the MERN stack is particularly well-suited for building scalable web applications based on its modularity and JavaScript-based architecture. It simplifies development and reduces overhead (Ranjan, 2020). The system currently performs well for its minimum viable product (MVP) requirements. However, further scalability measures such as implementing caching with Redis and deploying load balancers could enhance performance for the bigger scale handling.

6.4 Security and privacy

Security and privacy are critical components of the system's architecture. The platform uses HTTPS for secure data transmission. The communication between the client and server is encrypted. HTTPS employs Transport Layer Security (TLS) to protect data from interception or tampering, making it a standard for safeguarding sensitive user information in modern web applications (Rescorla, 2000).

User authentication is implemented securely through JWTs. So only authorized users allowed to access protected resources. Additionally, the modular backend design ensures that user data is accessed only through authenticated endpoints. Passwords are hashed using bcryptjs, adding an additional layer of security by ensuring that plaintext passwords are never stored in the database (bcryptjs Documentation, n.d.).

7 Future Development

Despite the full attention and effort that was put to the development of this system, there are several aspects that could be improved in the next versions. For instance, certain features such as user interface design, scalability, and testing could be refined to improve performance and user experience.

7.1 UI improvement

In the current user interface (UI) there are areas to enhance the overall user experience. First, ensuring consistency in the use of colors, typography, and borders. This consistency would improve visual appeal and align the design with modern UI principles. Second, enhancing spacing and layout. This would contribute to a better readability and make navigation across the platform more intuitive. Third, accessibility improvements, such as enabling better keyboard navigation and providing robust screen reader support, would make the platform more inclusive and cater to a wider audience (W3C, 2019).

7.2 Performance optimization

To ensure the platform that can handle increasing demands in larger scale of users and requests, performance optimization is a critical area of focus. First, database indexing should be implemented. This would be handled by adding indexes to frequently queried fields in MongoDB. This feature can significantly enhance query performance, particularly for large datasets (Mongoose Documentation, 2023). Second, caching mechanisms can be introduced using tools like Redis to reduce latency for commonly accessed data, such as session listings or user profiles. This approach would also improve response times for API calls by minimizing the need to repeatedly fetch data from the database.

7.3 Expanding requirements

The platform is employed based on the MOSCOW prioritization method. During its initial development focusing primarily on must-have and some of the should-have requirements to deliver a functional minimum viable product (MVP). To enhance the platform, future iterations should revisit the remaining categories. First, implementing remained should-have features such as advanced search filters for tutor matching and dynamic session recommendations. It would significantly enhance usability. Second, developing could-have features including detailed user analytics, personalized learning paths, and gamification elements. These features would improve user engagement and satisfaction. Expanding requirements in subsequent development cycles would allow the platform to address a broader range of user needs. As a result, it would enhance its overall competitiveness.

7.4 Automated tests

Testing is one of the most important components of ensuring software quality. However, it remained underdeveloped in the implementation. Systematic testing strategies would significantly enhance the platform's reliability and maintainability. First, implementing unit testing for individual components and functions could ensure the reliability of them. Some of the required functionalities to test are authentication, session management, and AI interactions. Tools like Jest and Mocha are well-suited for this purpose. Second, integration testing should be employed to validate the interactions between different modules. As the frontend and backend, as well as API endpoints needed to be test. These tests would verify that the system functions cohesively under various scenarios, thereby identifying and resolving potential issues early in the development process (Holmes, 2018).

7.5 Deployment and hosting

Deploying the platform on a reliable hosting environment is another crucial step. It is enabling it to serve real users effectively. The deployment process involves several key steps. First, the frontend and backend should be hosted on a scalable infrastructure, such as AWS or Heroku. This would guarantee the system can handle varying levels of user demand. Second, configuring a custom domain is essential to provide an easy accessible URL for users. Third, setting up Continuous Integration/Continuous Deployment (CI/CD) pipelines would automate testing and deployment processes that enabling faster and more reliable updates to the platform (Ranjan, 2020).

8 Conclusion

The aim was to design and develop a tutor-finding platform using the MERN stack, featured with AI integration to enhance the learning experience. To achieve this aim, user requirements were gathered. The process included conducting surveys and interviews with potential students and tutors. These steps indicated the design process and ensuring the implementation of critical features. These features included user authentication, tutor profiles, session booking, and AI-assisted support among others. The MoSCoW prioritization method ensured a focused approach to feature implementation. The use of React's component-based architecture and Redux Toolkit enabled scalability and maintainability.

In the early stages of development, a comprehensive analysis of existing platforms was conducted to identify strengths and weaknesses of existing solutions. It informed the design and functionality of the tutor-finding platform. This analysis helped the platform to not only meet users' needs but also provided a competitive edge by incorporating features that users find most valuable. Then a mock platform was developed. This mock was serving as a prototype for the core features, including user registration, tutor searching, and session management. Key achievements from this initial phase include secure authentication using JWT, a modular backend architecture, and initial AI integration. However, opportunities for improvement are remained for features such as enhancing the user interface, integrating financial payment systems, and further refining AI capabilities for real-time assistance.

The tutor-finding system is designed to offer a unique experience by offering several key features that differentiate it from existing platforms. It includes a user-friendly reservation system and enabling students to easily schedule sessions with tutors. A standout feature is the integration of AI. It tailored recommendations and learning paths to individual student needs. By providing a personalized experience that adapts to each user's progress and learning style.

By addressing gaps in the online tutoring market, the platform offers a scalable, secure, and user-centric solution for students and tutors. While at this point strongly focused on foundation, future iterations and refinements are essential for ensuring its long-term goal.

References

Abramov, D. (2015). Redux: A predictable state container for JavaScript apps. Retrieved from https://redux.js.org/

Abramov, D., Clark, A., & Redux Team. (2015). Redux Toolkit Documentation. Retrieved from https://redux-toolkit.js.org/

Abu-Mostafa, Y. S. (2012). The real origins of machine learning. Retrieved from https://phil-papers.org/rec/ABUTRO-9

Adams, C., & Wilson, A. (2021). E-tutoring and the role of technology in personalized learning. Journal of Educational Technology Research, 48(3), 235–252.

Agrawal, A., Lacetera, N., & Lyons, E. (2016). Does standardized information in online markets dis-proportionately benefit job seekers with less experience? Journal of Labor Economics, 34(2), 443–499. https://doi.org/10.1086/682338

Alrakhawi, H. A., Jamiat, N. U. R. U. L. L. I. Z. A. M., & Abu-Naser, S. S. (2023). Intelligent tutoring systems in education: A systematic review of usage, tools, effects, and evaluation. Journal of Theo-retical and Applied Information Technology, 101(4), 1205–1226.

Anita. (2021). Pedoz-Pet-Service-Web-UI-Kit. Retrieved from https://ui8.net/anita-1308ba/prod-ucts/pedoz---pet-service-web-ui-kit

Axel Rauschmayer. (2015). Exploring ES6: Upgrade to the next version of JavaScript. Leanpub.

Barrow, D., Mitrovic, A., Holland, J., Ali, M., & Kourentzes, N. (2024). Developing personalised learning support for the business forecasting curriculum: The forecasting intelligent tutoring system. Forecasting, 6(1), 204. https://doi.org/10.3390/forecast6010012

Bierman, G., Abadi, M., & Torgersen, M. (2014). Understanding TypeScript. Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 289-303.

Bizami, N. A., Tasir, Z., & Kew, S. N. (2023). Innovative pedagogical principles and technological tools capabilities for immersive blended learning: A systematic literature review. Education and Information Technologies, 28(2), 1373–1425.

Boulton, M. (2019). JWT authentication: A practical guide. O'Reilly Media.

Brown, H. (2020). Niche learning needs in digital tutoring platforms: An exploratory study. Journal of Online Education, 25(3), 56-72.

Brown, H., & White, J. (2016). User interface preferences in digital platforms: A comparative study of mobile versus desktop access. Journal of Information Technology & Software Engineering, 7(3), 128-136. https://doi.org/10.1016/j.jits.2016.12.003

Brown, H. D., & Lee, H. (2015). Teaching by principles: An interactive approach to language pedagogy (4th ed.). Pearson.

Brown, T., & Smith, J. (2020). Web development with Node.js and Express: A modular approach to building scalable applications. TechPress.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … & Amodei, D. (2020). Language models are few-shot learners. Ithaca. Retrieved from http://ezproxy.jamk.fi:2048/login?url=https://www.proquest.com/working-papers/language-models-are-few-shot-learners/docview/2407684779/se-2

Chellaraj, S. A. (2024). Collaborative community-based self-expanding (CCSE) E-learning model (Order No. 31521923). (3122641466). Retrieved from http://ezproxy.jamk.fi:2048/login?url=https://www.proquest.com/dissertations-theses/collaborative-community-based-self-expanding-ccse/docview/3122641466/se-2

Chen, C., & Peng, P. (2018). Evaluation systems in online tutoring platforms: The role of feedback in quality assurance. Journal of Educational Technology & Society, 21(2), 125-136.

Chen, L., Chen, P., & Lin, Z. (2020). Artificial intelligence in education: A review. IEEE Access, 8, 75264-75278.

Chen, R., & Huang, S. (2019). Automated decision-making in digital services: Effects of algorithms on user satisfaction. Journal of Service Research, 22(4), 388-402. https://doi.org/10.1177/1094670519868844

Chen, R., & Huang, S. (2020). Simplifying user interaction in educational platforms: A focus on booking systems. Journal of Educational Technology, 12(2), 110-121. https://doi.org/10.1016/j.jet.2020.06.005

Chen, T., Li, X., Lv, X., & Fang, X. (2020). Impact of online tutoring platforms on student performance: A cross-country analysis. Journal of Educational Technology & Society, 23(4), 45–59. https://doi.org/10.1016/j.ets.2020.02.011

Coetze, J., et al. (2018). Web conference-based tutorials: Impact on academic performance. Journal of Online Education and Teaching, 14(1), 1-15.

Concorde Education. (n.d.). Combining offline and online tutoring: The hybrid teaching model of the future. Retrieved from https://concordeeducation.com

Creswell, J. W., & Plano Clark, V. L. (2011). Designing and conducting mixed methods research (2nd ed.). SAGE Publications.

Express.js Documentation. (2023). Express - Node.js web application framework. Retrieved from https://expressjs.com

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation). University of California, Irvine. Retrieved from http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.

Formium. (2021). Formik: Build forms in React, without the tears. Retrieved from https://formik.org/

Gackenheimer, C. (2015). Introduction to Nodemon: Automate Restarts in Node.js Development. Addison-Wesley.

Gackenheimer, C. (2015). Introduction to React. Apress.

García-Martín, E., Rodrigues, C. F., Riley, G., & Grahn, H. (2019). Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing, 134, 75-88.

Garrison, D. R., & Vaughan, N. D. (2008). Blended learning in higher education: Framework, principles, and guidelines. Jossey-Bass.

Gikandi, J. W., Morrow, D., & Davis, N. E. (2011). Online formative assessment in higher education: A review of the literature. Computers & Education, 57(4), 2333–2351.

Gillani, N., Eynon, R., Chiabaut, C., & Finkel, K. (2023). Unpacking the "Black box" of AI in education. Journal of Educational Technology & Society, 26(1).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Gorur, R., & Dey, J. (2021). Making the user friendly: The ontological politics of digital data platforms. Critical Studies in Education, 62(1), 67–81.

Gravetter, F. J., & Wallnau, L. B. (2013). Statistics for the behavioral sciences (9th ed.). Belmont, CA: Wadsworth Cengage Learning.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer.

Hennessy, J., & Patterson, D. (2019). Computer architecture: A quantitative approach (6th ed.). Elsevier.

Holmes, W., Bialik, M., & Fadel, C. (2019). Artificial Intelligence in Education: Promises and Implications for Teaching and Learning. Center for Curriculum Redesign.

Holmes, A. (2018). Node.js Design Patterns (2nd ed.). Birmingham, UK: Packt Publishing.

Holmes, D. (2018). Getting Started with Node.js. Packt Publishing.

Huang, Z., Fang, X., & Huang, X. (2021). Trust and transparency in online tutoring platforms: The impact of reviews on student decision-making. Journal of Online Learning and Teaching, 17(1), 12-25. https://doi.org/10.24059/olj.v17i1.2941

Jackson, S. (2020). The evolution of personalized learning platforms in higher education. Educational Technology & Research, 40(2), 145–160.

JAMK University of Applied Sciences. (n.d.). Thesis assessment criteria. https://help.jamk.fi/opinnaytetyo/en/project-plan/assessment-criteria/

Jennings, R. B., Nahum, E. M., Olshefski, D. P., Saha, D., Shae, Z.-Y., & Waters, C. (2006). A study of Internet instant messaging and chat protocols. IEEE Network, 20(4), 16–21. https://doi.org/10.1109/MNET.2006.1668399

Johnson, A. (2020). The limitations of specialized online platforms: A critical analysis. Journal of Online Learning and Teaching, 16(1), 37-49.

Johnson, A. (2021). Modern web development frameworks: A comparative analysis. TechPress.

Johnson, M., & Lee, C. (2020). Personalization in mobile fitness applications: The role of algorithms in enhancing user experience. Journal of Health Informatics, 10(2), 200-213.

Johnson, M., & Lee, C. (2021). Digital platforms for personalized learning: The rise of online tutoring services. International Journal of Educational Technology, 18(1), 23-35.

Jones, T. (2019). Expanding platform access in service-based applications: A case for web-based interfaces. Journal of Human-Computer Interaction, 24(1), 115-124. https://doi.org/10.1080/10447318.2019.1565234

Jurafsky, D., & Martin, J. H. (2020). Speech and language processing (3rd ed.). Pearson.

Kässi, O., & Lehdonvirta, V. (2018). Online labour index: Measuring the online gig economy for policy and research. Technological Forecasting and Social Change, 137, 241–248. https://doi.org/10.1016/j.techfore.2018.07.056

Kostev, R. S. (2023). Challenges and Problems of the MoSCoW Method Application in ERP System Implementation. 2023 International Scientific Conference on Computer Science (COMSCI), Sozopol, Bulgaria, 1-4. https://doi.org/10.1109/COMSCI59259.2023.10315816.

Krohn, D. (2021). Understanding CORS: A guide to cross-origin requests. Manning Publications.

Krohn, T. (2021). dotenv: Secure environment variable management in Node.js. Retrieved from https://github.com/motdotla/dotenv

Kurniawan, B. (2015). Node.js Web Development. Packt Publishing.

Lenarduzzi, V., & Taibi, D. (2016, August). MVP explained: A systematic mapping study on the definitions of minimal viable product. In G. A. Papadopoulos, C. Gravino, & M. Höst (Eds.), 2016 42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 112–119). IEEE.

Libby, A. (2023). Developing web components with Svelte: Building a library of reusable UI components. Apress.

Liu, J. (2016). Paper on machine learning, natural language processing, and computer vision.

Mendez, A. (2020). Efficient State Management in React with Redux Toolkit. Retrieved from https://blog.logrocket.com/

Mulligan, T. (2021). UX/UI Design 2021-2022: Tutorial for beginners: The complete step-by-step guide to UX/UI design and best practices for designers with no experience.

Mongoose Documentation. (2023). Instance Methods. Retrieved from https://mongoosejs.com/docs/guide.html#methods

Mongoose Documentation. (2023). Middleware. Retrieved from https://mongoosejs.com/docs/middleware.html

Marinho, T. (2018). json-server: A full fake REST API with zero coding. Retrieved from https://github.com/typicode/json-server

Morris, R. M. (2011). An exploration of the barriers to independent study and learning in first-year university students: A mixed methods research approach. Retrieved from https://kipdf.com/an-exploration-of-the-barriers-to-independent-study-and-learning-in-first-year-u_5aafb8551723dd339c80340f.html

Node.js. (n.d.). Node.js official website. Retrieved from https://nodejs.org/en

O'Neill, M. (2020). Node.js authentication: Secure your Node.js applications. Apress.

OpenAI. (2023). ChatGPT: Optimizing language models for dialogue. https://openai.com/chatgpt

Okebukola, P. A. (Ed.). (2025). AI and quality higher education handbook: Vol. 2. AI and curriculum development for the future (p. 87). Sterling Publishers. Retrieved from http://eprints.gouni.edu.ng/4428/1/Volume%202-AI%20and%20Quality%20Higher%20Education%20Handbook-January%202025.pdf

Porter, P., Yang, S., & Xi, X. (2019). The design and implementation of a RESTful IoT service using the MERN stack. 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW), 140–145. https://doi.org/10.1109/MASSW.2019.00035

Qutab, I., Asghar, Z., Aqeel, M., Fatima, U., Naqvi, W., & Muneeb, M. Y. (2024). Optimizing senti-ment analysis: A novel hybrid model integrating PCC-HHO with BILSTM-RNN for enhanced accu-racy on diverse textual datasets. International Journal of Engineering Research & Technology, 13(10). Retrieved from https://www.ijert.org/optimizing-sentiment-analysis-a-novel-hybrid-model-integrating-pcc-hho-with-bilstm-rnn-for-enhanced-accuracy-on-diverse-textual-datasets

React Training. (2016). React Router Documentation. Retrieved from https://reactrouter.com

React. (n.d.). React – A JavaScript library for building user interfaces. Retrieved from https://leg-acy.reactjs.org/

Rescorla, E. (2000). HTTP Over TLS. RFC 2818. Retrieved from https://www.rfc-edi-tor.org/rfc/rfc2818.html

Ranjan, R. (2020). Building scalable web applications using MERN stack. International Journal of Advanced Research in Computer Science and Software Engineering, 10(5), 123-130.

Subramanian, M. (2017). Pro MERN stack: Full stack web app development with Mongo, Express, React, and Node. Apress.

Van Zandwijk, R., & Pauwels, P. (2023). Implementing secure middleware in web applications (Master's thesis, Eindhoven University of Technology). Retrieved from https://pure.tue.nl/ws/portalfiles/portal/349725887/Zandwijk_van_1354752_Pauwels_MSc_Thesis.pdf

W3C. (2019). Web Content Accessibility Guidelines (WCAG) 2.1. Retrieved from https://www.w3.org/TR/WCAG21/

Wijayanti, A., & Tanone, R. (2021). Design thinking and emotional intelligence in UI/UX design of website-based online foreign service travel expenses (BPDL) applications. 2021 2nd International Conference on Innovative and Creative Information Technology (ICITech), Salatiga, Indonesia, 130–135. https://doi.org/10.1109/ICITech50181.2021.9590156

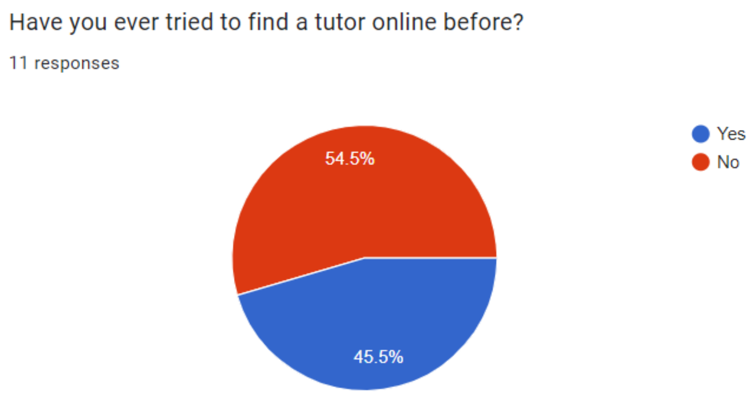
Zakas, N. C. (2012). Maintainable JavaScript: Writing Readable Code. O'Reilly Media.

Zakas, N. C. (2012). Understanding data validation in JavaScript. Retrieved from https://validatorjs.org

Zhao, Y., Guo, Q., Zhang, Y., Zheng, J., Yang, Y., Du, X., & Zhang, S. (2023). Application of deep learning for prediction of Alzheimer's disease in PET/MR imaging. Bioengineering, 10(10), 1120. https://doi.org/10.3390/bioengineering10101120

Appendices

Appendix 1. Survey questions


Have you ever tried to find a tutor online before?

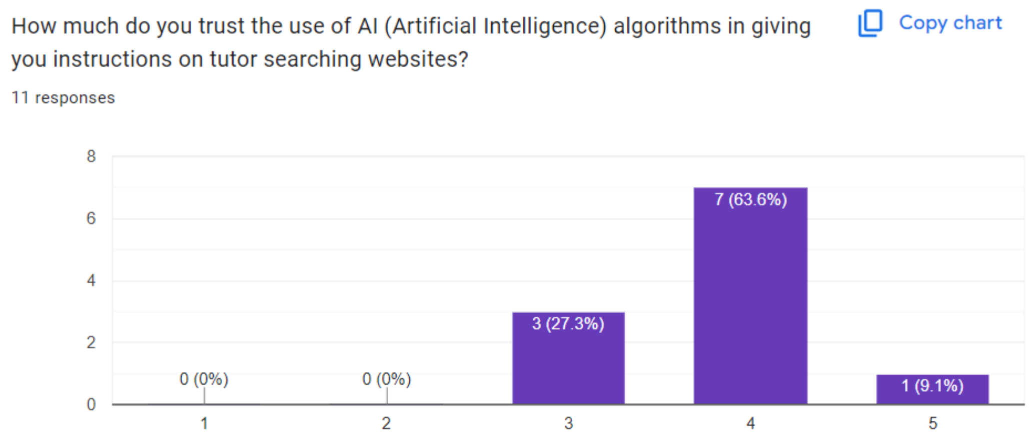
11 responses

If yes, How did you find the process of finding a tutor online?

- Search filters for specific subjects or skills (45.5%)
- Detailed tutor profiles with qualifications and experience (36.4%)
- Availability and scheduling options (27.3%)
- Secure messaging and communication tools (0%)
- Reviews and ratings from previous students (27.3%)
- Secure payment options (18.2%)
- Additional resources or materials provided by tutors (0%)

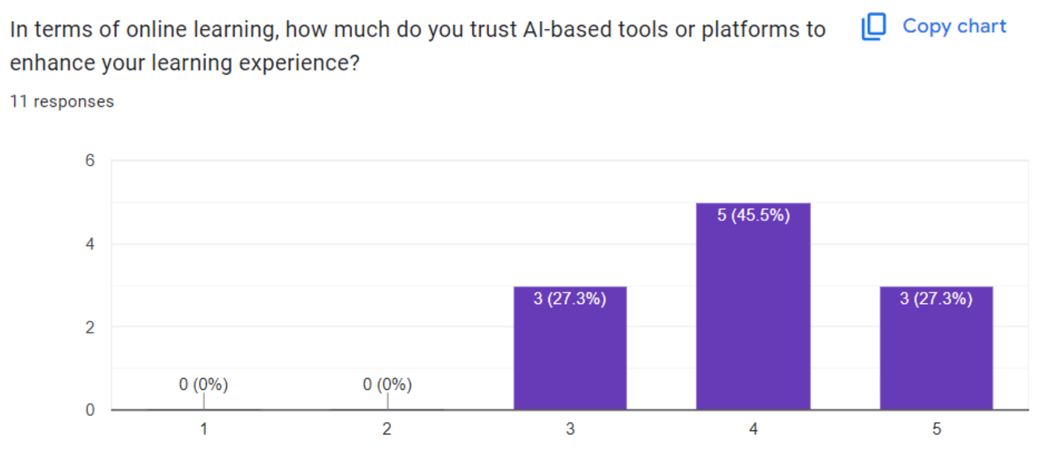
What features are needed to be implemented on tutor websites to enhance your user experience and make it more effective?

- Advanced search filters for specific criteria (e.g., location, availability, teaching style) (27.7%)
- Verified tutor profiles with background checks and certifications (63.6%)
- Interactive lesson scheduling and calendar integration (45.5%)
- Live video chat or virtual classroom functionality (36.4%)
- Student progress tracking and performance analytics (54.5%)
- Enhanced security measures for user data and transactions (27.3%)
- Integration with educational resources or learning materials (36.4%)s



What factors contribute to your level of trust in AI recommendations for tutors on online plat-forms?

- What factors contribute to your level of trust in AI recommendations for tutors on online platforms? (54.5%)
- Transparency in how AI algorithms make recommendations (27.3%)
- Personalization of recommendations tailored to individual learning needs (81.8%)
- Integration of user feedback to improve recommendations over time (27.3%)
- Confidence in the platform's data privacy and security measures (36.4%)
- Previous positive experiences with AI-driven recommendations on other platforms (36.4%)



What features or functionalities do you believe AI-based tools or platforms should prioritize to gain your trust in enhancing your online learning experience?

- Adaptive learning paths tailored to individual learning styles and pace (72.7%)
- Intelligent content recommendations based on learning objectives and preferences (90.9%)
- Real-time feedback and performance analytics to track progress (54.5%)

- Virtual tutoring or personalized assistance through chatbots or virtual assistants (27.3%)
- Interactive simulations or gamified learning experiences (45.5%)
- Robust data security and privacy measures for user data (36.4%)
- Others (9.1 %)
    - provide feedback or answers accessing real-time internet
    - capability of understanding user requirement based on the images or charts/diagrams user provide

Appendix 2. Overview of tutor skills and session details from interviews

#	Tutor Subject	Private/ Organization	Students ages/grade
1	Fundamental mathematics and Arabic literature	Private	7-12
2	Urban and architectural design	Private	20 to above/ Bachelors students
3	C# Asp.net	Private	18 to above
4	Construction management	Organization and Universities	Masters students
5	Communication skills	Organization, The agency that supports children's mental health	4 to 10

Appendix 3. Interview questions

- Have you ever conducted tutoring sessions as a tutor? what subjects or skills did you tutor?
- How would you rate your overall experience as a tutor?
- What challenges did you encounter while conducting tutoring sessions?
- What kind of tutor materials, tools did you find useful during the sessions?
- Have you ever used AI tools, to help you as a student or tutor? Tell me about your experience.
- What strategies or techniques did you find most effective in conducting successful tutoring sessions?
- Have you ever taken a class or tutoring session as a student? What subject or skill was the session for?
- How would you rate your overall experience as a student in the tutoring session?
- What aspects of the tutoring session did you find most beneficial or enjoyable as a student?
- What improvements or changes would you suggest to enhance the effectiveness of tutoring sessions from both the tutor's and student's perspectives?

Second section of interview, by sharing the Figma design

- In what way could this app be valuable for you?

- How easy do you think is this app to use?

- How could an app like this help you in your career/ school attainments?

How would you improve the design of this app?