



Design and Implementation of Artificial Intelligence-Driven Platform for Programming and Development Education

Diana Efimova

BACHELOR'S THESIS
May 2025

Bachelor's Degree Programme in Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Bachelor's Degree Programme in Software Engineering

Diana Efimova:
Design and Implementation of Artificial Intelligence-Driven Platform for Programming and Development Education

Bachelor's thesis 64 pages, appendices 13 pages
May 2025

The aim of this thesis was to explore artificial intelligence (AI) capabilities in education and demonstrate, through a practical example, how OpenAI's technology can effectively support developers in their learning process. By showcasing real-world applications, this work highlights the potential of AI-driven tools in enhancing coding education and problem-solving efficiency.

The theoretical section examines web platforms, which utilize artificial intelligence for educational purposes, along with the technologies used to develop the practical components, such as React, Python, and REST APIs. The theory is further supported by examples demonstrating the execution of practical work.

The final work includes a deployed educational platform, accessible to the public for study purposes. This platform leverages AI technologies to enhance learning experiences for developers, offering a practical tool for gaining insights based on code input. Through this implementation, the project demonstrates the positive impact of AI-driven solutions on education, fostering more efficient and interactive learning processes. The links to the deployed product and complete code are provided.

Key words: Artificial Intelligence (AI), interactive learning, practical implementation

CONTENTS

1	INTRODUCTION.....	7
2	BACKGROUND AND PURPOSE OF THE THESIS	9
2.1	OpenAI overview	9
2.1.1	History and mission.....	9
2.1.2	Key technologies.....	10
2.1.3	OpenAI API.....	10
2.2	AI in education.....	13
2.3	Expected output.....	14
2.3.1	Practical execution.....	14
2.3.2	System distinction	15
3	THEORY AND TECHNOLOGIES	17
3.1	Frontend technologies	18
3.1.1	Overview of React.....	18
3.1.2	Overview of REST API.....	19
3.1.3	Frontend technical environment overview	20
3.2	Backend technologies	21
3.2.1	Overview of Python.....	21
3.2.2	Overview of FastAPI	22
3.2.3	Backend technical environment overview.....	22
3.3	Overview of Artificial Neural Network	23
3.3.1	Fundamentals of Artificial Neural Network.....	23
3.3.2	ANN technical environment overview	25
4	IMPLEMENTATION	26
4.1	Frontend configuration.....	26
4.1.1	Project setup and dependencies	27
4.1.2	Component structure	27
4.1.3	Key functionalities	28
4.1.4	UI design and responsiveness.....	29
4.2	ANN configuration	34
4.2.1	Development environment.....	34
4.2.2	Dataset.....	34
4.2.3	Model training	36
4.2.4	Model evaluation.....	37
4.3	Backend configuration	38
4.3.1	Data processing	38
4.3.2	Model deployment.....	39

4.4 Application evaluation.....	40
5 DISCUSSION	43
REFERENCES	45
APPENDICES	49
Appendix 1. Source code for MainPage.js.	49
Appendix 2. Source code Background.js	55
Appendix 3. Source code openaiService.js.....	56
Appendix 4. Source code for ANN training.....	57
Appendix 5. Source code for application's backend.....	59
Appendix 6. Source code CSS styling for smaller screens.	61
Appendix 7. ChatGPT's response to a solely code input (ChatGPT [THL], 2025).....	63
Appendix 8. Thesis' practical work's response to a solely code input.	64

GLOSSARY

TAMK	Tampere University of Applied Sciences
cr	credit
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AST	Abstract Syntax Tree
CSS	Cascading Style Sheets
CRA	Create React App
CRUD	Create, Read, Update, Delete
DOM	Document Object Model
ELF	English as a Lingua Franca
GPT	Generative Pre-trained Transformer
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JS	JavaScript
ML	Machine Learning
MLP	Multilayer Perceptron
MVC	Model View Controller
NLP	Natural Language Processing
NumPy	Numerical Python
PyCharm	Python Charm (IDE for Python)
ReLU	Rectified Linear Unit
REST	Representational State Transfer
SDK	Software Development Kit
UML	Unified Modelling Language
VS Code	Visual Studio Code
UI	User Interface
MVP	Minimum Viable Product

1 INTRODUCTION

The purpose of this work is to assess the role of artificial intelligence (AI) in education by exploring its application in the field of learning and presenting a practical example of how artificial intelligence technologies can be used to optimize the learning experience for developers.

During recent years, a rapid increase in the popularity of AI platforms has been driven by advancements in machine learning and deep learning technologies. As a result, numerous AI-powered products have emerged across various industries, offering capabilities ranging from natural language processing to computer vision. This widespread adoption has been further accelerated by the increasing accessibility of AI tools, which no longer require extensive computational resources. Instead, businesses and individuals can leverage APIs from pre-trained models, enabling seamless integration of AI functionalities without the need for specialized hardware or complex training processes. This accessibility enables anyone with internet access to explore powerful AI capabilities in various ways, including for educational purposes (Korolov, 2019).

Due to the wide range of capabilities offered by AI models, there is significant potential for misuse, especially in sensitive fields like education. AI-powered platforms, if not properly regulated or guided, could be abused for unethical purposes, such as spreading misinformation or facilitating cheating (Akhtar, 2024).

In an era where the limits of artificial intelligence remain largely unknown and its use is extremely unregulated, it is important to spread awareness of responsible ways to utilize the technology.

This thesis aims to evaluate the use of AI technologies in education field and provide examples of appropriate use of artificial intelligence. The practical part of the work focuses on the implementation of the educational web platform, which utilizes artificial intelligence solution to support learning processes and development of programming skills. It eliminates the need for a model training and restricts code generation to keep users engaged with the studying.

The application utilizes OpenAI API to provide users with insights into their code and assist them in developing new skills. Offering four pre-trained roles for AI assistance, the application ensures that users can concentrate on specific tasks and prevents information overload. Another distinguished feature provides users with predicted code difficulty, which helps learners to estimate the complexity of their work.

The primary goal of the application is to apply the capabilities of artificial intelligence to assist and improve the learning process by providing personalized support and instant feedback, all while ensuring that users remain actively engaged and continue to develop their critical thinking skills in order to eventually be independent of AI assistance.

2 BACKGROUND AND PURPOSE OF THE THESIS

2.1 OpenAI overview

An essential part of this thesis revolves around OpenAI, an artificial intelligence research organization, as it is a key player in the world of AI implementation, and its technologies are used in the practical part of this work.

This chapter explores the theoretical background of OpenAI and company's most used technologies.

2.1.1 History and mission

The company began its journey in 2015 as a non-profit artificial intelligence research organization. Its goal, which remains unchanged, is to 'advance digital intelligence in a way that most benefits humanity as a whole' (Brockman, Sutskever and OpenAI [THL], 2015).

In 2019, OpenAI adopted a 'capped-profit' model by creating OpenAI LP, a for-profit subsidiary. This restructuring was designed to attract external investments. During these changes, the company reaffirmed its commitment to accessibility for all and emphasized long-term safety by continuing research without engaging in a competitive race that overlooks essential safety precautions (Brockman, Sutskever and OpenAI [THL], 2019).

That same year, OpenAI partnered with Microsoft, which made a significant contribution by providing cloud computing resources through its Azure platform. This collaboration greatly accelerated OpenAI's progress. With these strategic partnerships and advancements, OpenAI has continued to evolve, driving significant developments in the AI field (Brockman, 2019).

This widespread impact is particularly evident in the education sector, where OpenAI's tools have become valuable resources for students. In fact, OpenAI reports that more than one-third of their users are students who uses their products

for writing papers, exploring topic and brainstorming ideas (OpenAI [THL], 2025). This leads to the conclusion that AI has become a fundamental component of modern education, shaping the way people learn and teach.

2.1.2 Key technologies

The foundation of this thesis's practical implementation leverages the capabilities of OpenAI, the platform widely recognized for developing ChatGPT – a chatbot powered by artificial intelligence. However, OpenAI is also distinguished in the IT world for its revolutionary model architectures that extend the limit of artificial intelligence's potential. The systems are developed to manage extensive tasks including natural language comprehension and text generation.

The primary programming language used by OpenAI is Python because of its flexibility and libraries for machine learning and data analysis. Specifically, the company employs libraries like NumPy for numerical computations, Pandas for handling and processing data, and Matplotlib for generating visual representations of data, all of which contribute to OpenAI's development of ChatGPT by enabling efficient data processing, model training, and performance analysis (Design Gurus Team [THL], 2025).

Additionally, C++ is utilized in order to optimize algorithms and manage system resources effectively, contributing to the high-performance execution of AI models (Design Gurus Team [THL], 2025).

Besides advanced frameworks, OpenAI requires powerful cloud infrastructure to manage the vast amounts of data necessary for training models like ChatGPT. Microsoft Azure serves as a key partner, providing the cloud computing resources needed to support these operations (Microsoft Corporate Blogs [THL], 2025).

2.1.3 OpenAI API

This project is specifically interested in APIs provided by OpenAI. The service allows developers to utilize powerful AI models such as text generation, which is used in the project to provide information to users based on their code.

OpenAI provides API for general purpose “text in, text out” interface, which allows users to try it with tasks in English language. The API supports diverse programming languages and platforms, allowing seamless integration into existing workflows and applications (OpenAI [THL], 2020)

In order to use the API, users must create an OpenAI account and request a key. Each key is unique, tied to the user's account, and cannot be shared or transferred. Besides the key, users are required to download OpenAI's software development kit (SDK) in order to use the OpenAI API in server-side (OpenAI [THL], 2020).

```
npm install openai
```

FIGURE 1. Demonstration installation command for OpenAI's SDK.

OpenAI package allows to bypass standard API connection methods as Axios, which simplifies the process by directly handling API calls to OpenAI's servers, eliminating the need for manual REST API requests and streamlining the integration process (demonstrated in the Figure 2).

```
import OpenAI from "openai";
const client = new OpenAI();

const completion = await client.chat.completions.create({
  model: "gpt-4.1",
  messages: [
    {
      role: "user",
      content: "Write a one-sentence explanation of user's code input",
    },
  ],
});

console.log(completion.choices[0].message.content);
```

FIGURE 2. Demonstration of connection to OpenAI model with OpenAI's SDK (OpenAI [THL], 2020).

The piece of code taken from OpenAI documentation is the example of initialization of an OpenAI client, which sends a message to the GPT-4.1 model. The “message” array defines the conversation — in this case, asking for one-sentence explanation. The “completion” object stores the AI's response, and “completion.choices[0].message.content” retrieves and prints the actual reply text from the model.

The OpenAI package provides a streamlined way to interact with OpenAI's models by handling authentication, request formatting, and response parsing, reducing the need for manual REST API calls and simplifying integration into applications.

To integrate OpenAI technologies into external applications, developers must first select a pricing plan. Costs depend on many factors such as model, tokens used, build-in-tools, etc. The diversity of offered models can be examined in Table 2. This practical work uses pricing plan of gpt-4o-mini model with billing per token, which is a unit of text processing that the model uses to understand and generate responses. Tokens are parts of text that can represent a word, part of a word, or even a punctuation mark. The table below showcases the pricing used in the project.

TABLE 1. gpt-4o-mini model pricing pre token (OpenAI [THL], 2020).

Input	Cached Input	Output
\$0.15	\$0.075	\$0.6

OpenAI provides various models to be used with company's API, each serving a specific purpose. Below a table with some models and their categories is presented

TABLE 2. OpenAI models available through API (OpenAI [THL], 2020).

Reasoning models	Image generation models	Text-to-speech models	Transcription models	Embeddings model
o4-mini	GPT Image 1	CPT-4o mini TTS	Whisper	Text-embedding-3-small

o3-mini	DALLE 2	TTS-1	GPT-4o Transcribe	Text-embedding-3-large
o1-pro	DALLE 3	TTS-1 HD	GPT-4o mini Transcribe	Text-embedding-ada-002

While the thesis aims to give an understanding of the variety of available products, it is acknowledged that there are many more models available with new developments coming out regularly. Thereby, it is recommended to consult with the first source about the updated list of OpenAI products. Source is provided in references.

2.2 AI in education

Since the release of ChatGPT and the AI breakthrough in 2022, significant changes have occurred in the education sector. According to the Digital Education Council, 84% of educators now incorporate AI into their teaching practices, while 86% of students worldwide regularly use AI for studying (Kelly, 2024).

Students and educators utilize AI for various purposes, including writing, text-to-speech assistance, presentation generation, coding, and more. Many students report that AI helps streamline repetitive tasks, which allows them to focus more on creativity and engaging in meaningful discussions (Lu, 2025).

Numerous prominent educational platforms have integrated AI capabilities into their systems. A prime example is Duolingo, a pioneer in language learning. The language-learning application employs AI to personalize lessons, adapt to user proficiency levels, and ensure ongoing engagement (Patel, 2024). Another noteworthy platform is Khan Academy, which provides free online courses and has introduced Khanmigo, an AI powered by GPT. This tool offers valuable support for teachers, parents, and students across various subjects and grade levels (Pillay, 2024).

As AI technology advances, many new platforms are emerging to challenge established leaders, driven by the cutting-edge innovations they bring. If older companies fail to adopt AI capabilities, they risk losing their competitive advantage. For instance, platforms like Elsa Speak, an AI-powered language learning platform focused on teaching English pronunciation (ELSA [THL], 2025). or CodeSignal, which offers AI-driven assessments and training for programmers (CodeSignal [THL], 2025). When considering AI in programming education, this field has numerous options to assist in learning as coding co-pilots, advisors, GitHub inspectors, etc.

However, currently, there is no clear evidence that AI applications will significantly improve learning outcomes. While their potential is substantial, several risks remain, such as privacy and security concerns, with sensitive information potentially being exposed or the development of critical thinking skills could be undermined (UNESCO [THL], 2024). Therefore, it is essential to emphasize the use of platforms that leverage AI capabilities responsibly to enhance education while preserving problem-solving skills among students.

2.3 Expected output

2.3.1 Practical execution

The development phase aims to demonstrate the effective application of AI capabilities for educational purposes, specifically in programming education, by creating an AI-driven web application. The web platform features a responsive React frontend and a Python backend, ensuring smooth performance and efficient functionality.

Using OpenAI API the platform enables users to select from four distinct AI models, each designed for a specific function. These models generate outputs based on the user's code, tailored to their predefined roles.

This approach intends to efficiently assist developers in their programming process by providing precise information promptly. Users are required to complete only two actions to get the support – choose a role and insert their code. No additional information is required, the model is pre-trained and ready to assist based on the chosen role. The available roles include:

- Teacher – explains the code step by step as if teaching a beginner and suggests improvements for readability and maintainability.
- Debugger – Identifies errors and provides explanations for why they occur and highlights potential runtime issues.
- Optimizer – suggests performance improvements and identifies redundant code or unnecessary complexity.
- Commenter - automatically adds comments to the code, explains complex logic in simple terms and helping developers document their work effectively.

The feature allows users to concentrate on a specific task and get a well-defined answer based on their needs.

Another key feature of the application is the code complexity analysis. Utilizing a custom-trained Artificial Neural Network (ANN) model, the application automatically evaluates the complexity of the user's code and classifies it as Easy, Medium, or Hard. This feature is beneficial as it provides instant feedback to the user without requiring any additional steps other than submitting the code. By offering a simple and clear classification, users can quickly gauge the difficulty of their code. In collaboration with the role feature, it enables users to better understand their progress and identify areas for improvement.

2.3.2 System distinction

The web application aims to stand out by utilizing AI capabilities in the most efficient way for educational processes.

Unlike AI chatbots such as ChatGPT-4 or DeepSeek, the platform eliminates the need of writing a lengthy request to achieve the desired output.

Besides, it addresses the issue of information overload and AI generated code, which comes from using AI chats.

Compared to popular co-pilots like Codeium and Supermaven, this platform does not aim to simplify the entire development process. Instead, it is designed specifically for educational purposes, helping developers learn and ultimately become self-sufficient, reducing their reliance on tools like co-pilots.

This approach promotes a deeper understanding of the code-writing process, preparing developers to solve problems and develop solutions without the dependency on AI assistants, ultimately equipping them with the skills to excel in the field.

3 THEORY AND TECHNOLOGIES

The practical implementation of this thesis is developed using a selection of modern technologies, which are chosen with the consideration of the required functionalities, such as a seamless user interface and advanced artificial intelligence capabilities.

The backend of the project is developed using Python, where an Artificial Neural Network is trained and implemented to analyze code complexity. FastAPI is used to deploy the backend, enabling efficient handling of requests and responses.

The frontend is built with React, chosen for its extensive library support and flexibility in creating a responsive user interface. It seamlessly interacts with the Python backend through REST APIs, ensuring efficient data exchange.

Building on this foundation, the application's primary capability is inherited from OpenAI through its API, enabling the platform to leverage the power of ChatGPT-mini-40 interface.

Below the component diagram of the project is presented.

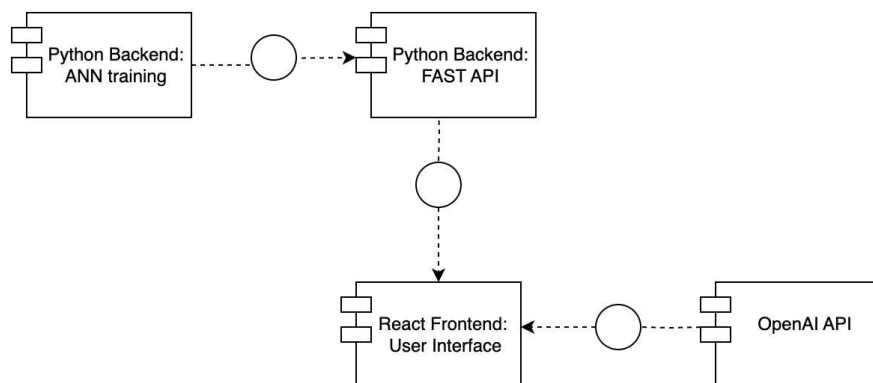


FIGURE 3. Component diagram of practical implementation.

In the following chapters, the scope of these technologies, along with the reasoning behind their selection, will be further explored.

3.1 Frontend technologies

3.1.1 Overview of React

The user interface of the practical part is built using React, a popular front-end JavaScript library developed by Facebook. React is designed for building reusable user interface (UI) components, enabling efficient and dynamic user interfaces. Its component-based architecture and virtual The Document Object Model (DOM) make it a powerful tool for developing interactive and responsive web applications (Kumar, 2025).

Few alternatives to React can be considered for a similar web development project. For instance, Inferno, a high-performance framework, which prioritizes application speed (Inferno [THL], 2025). Similarly, Vue, a framework with reduced bundle size, can be considered if application's fast performance is a priority (Vue [THL], 2025).

Nonetheless, React was chosen for this work due to its flexibility and compatibility with diverse frameworks for different purposes. React offers a high degree of flexibility for developers, allowing seamless integration with various other tools and technologies. For instance, CSS frameworks like Bootstrap, which provides tailored responsive components and grid systems to advance UI development or a range of animation libraries such as Framer Motion, which provides limitless possibilities to animate objects with minimal effort. React takes care of the integration process, allowing developers to seamlessly incorporate these libraries with minimal configuration and effort as developers can simply use the import statement to quickly integrate third-party libraries or components (Cook, 2023).

Another benefit of the framework is its seamless integration through REST APIs to fetch or send data to a backend server, which allows React to retrieve or update data as needed. APIs can be called using methods like `fetch` or libraries such as `Axios` to manage HTTP requests and responses (Halliday and Gorton, 2021).

3.1.2 Overview of REST API

In this project REST API is used to connect frontend and backend parts of the application.

A REST API provides a simple and flexible method for accessing web services without the need for complex processing. Most applications need a way to communicate with internal and third-party systems to perform various tasks, such as retrieving weather information, integrating payment gateways, accessing user authentication services, or like in this project's work obtain the power of OpenAI. RESTful APIs enable smooth information exchange by complying with well-established standards that ensure security, reliability, and efficiency in software communication (Olawanle, 2022).

All communication through a REST API is carried out using exclusively Hypertext Transfer Protocol (HTTP) requests. HTTP provides five methods that are utilized by REST architecture to perform CRUD (create, read, update, and delete) operations on data. These methods include:

- GET: Retrieves data from the server without making any changes.
- POST: Sends data to the server to create a new resource.
- PUT: Updates an existing resource with new data.
- PATCH: Partially updates an existing resource.
- DELETE: Removes a resource from the server.

Each of these methods serves a specific purpose in facilitating communication between the client and server, ensuring a standardized way to manage and manipulate data within a RESTful system (AWS [THL], 2025).

The benefits of utilizing REST API include scalability, flexibility and independence. REST APIs are highly scalable because they allow for easy distribution of requests across multiple servers, enabling applications to efficiently handle increasing traffic or data (Sanchhaya Education Private Limited [THL], 2025).

3.1.3 Frontend technical environment overview

Frontend of the practical part was developed in Visual Studio Code (VS code). VS Code is a cross-platform code editor with built-in support for various programming languages. Developed by Microsoft, it offers several beneficial for this project features:

- Support for debugging
- Highlighting of syntax
- Code refactoring
- Keyboard shortcust
- Customizable appereance

Moreover, the environment can be enhanced with various libraries and extensions in order to add languages support, additional utilities and many more features (Lardinois, 2015). VS code can be used with the range of programming languages and is free for use, which makes it a popular choice between developers

There are many alternatives to VS Code such as JetBrains IDEs and Visual Studio, however, VS Code was chosen for its easy to use interface and accessibility.

The final version of frontend was deployed with firebase making it available for public use. It is a hosting platform owned by Google, which offers a full support of an application lifecycle Firebase promises to provide fast and secure hosting, real-time database services, authentication, and easy integration with modern web technologies (Firebase [THL], n.d.). Although there are many alternatives such as AWS Amplify, Vercel, and Netlify, Firebase was chosen for its reliable infrastructure and accessibility.

3.2 Backend technologies

3.2.1 Overview of Python

Python was used to develop backend of the practical part. The programming language was chosen for its emphasis on code readability and simplicity, which allows developers to express concepts in fewer lines of code.

Python was developed by Dutch programmer Guido van Rossum in order to expand limitation in coding. It was made to be a high-level, general-purpose programming language with support of multiple programming paradigms, including object-oriented, imperative, and functional programming. The open-source and non-proprietary nature of Python significantly contributed to its rapid development and growing popularity. A large community of volunteer developers has collaboratively created thousands of modules, which have greatly enhanced Python's functionality (Munro, 2025).

There are many alternatives when it comes to developing backend for a web application. For a similar platform with utilization of machine learning (ML), it is recommended to consider object-oriented programming languages such as C++ and Java, which are highly used in enterprise development (Adebayo, 2023).

However, Python is an optimal language for developing foundational deep learning methods as it provides built-in tools necessary for implementation. One of the most widely used frameworks that leverage Python's capabilities for deep learning is TensorFlow, which can be used for a range of tasks, however its main purpose is neural network computation. Being an open-source software library, TensorFlow aims to provide simple way to deploy a computation across multiple platforms (PyPi [THL], 2025).

In this project, Python is used to develop two crucial components of the platform: ANN training and the FastAPI server. Both components are described in more detail in the following chapters.

3.2.2 Overview of FastAPI

In this project, the backend API is built using FastAPI, a modern framework utilized with Python. The framework was chosen due to its high performance and simplicity in building RESTful APIs. FastAPI is designed for efficient development of HTTP-based service APIs and provides automatic validation, data conversion and documentation (FastAPI [THL], n.d.).

The framework offers many features, among them are high performance, enhance feature development speed by 200% to 300% (FastAPI [THL], n.d.) and straightforward development.

FastAPI utilizes Pydantic, a widely used data validation library, for authentication and parsing, ensuring all inputs are presented in defined schemas. Therefore, the library defines the structure of bodies' request, cookies, and responses (Pydantic [THL], n.d.).

A simple example of FastAPI operation would be a client sending data to an API using POST method, therefore FastAPI uses Pydantic models to:

- Automatically parse the incoming JSON into Python objects.
- Validate the data types and constraints.
- Raise detailed errors if the input is invalid.

(Pydantic [THL], n.d.).

3.2.3 Backend technical environment overview

The Python code was developed in VS Code, the platform was described in Chapter 2.1.3

The backend was deployed using Render, a cloud platform that offers a simple and efficient way to host web applications. Render provides support for deploying

containerized applications and databases, allowing for easy scaling and management. It also integrates well with GitHub for automated deployments and offers seamless support for various backend technologies, including Python (Render [THL], n.d.).

Another distinguished benefit of Render is its continuous deployment capability. The platform allows automatic deployment of updates directly from GitHub. Additionally, it offers the capability to recover any previous version, enabling a quick restoration of prior states without the need to manually revert commits or modify data. Those features make the platform highly effective during the product's development phase (Krupa, 2025).

Although there are several alternatives, such as Heroku and DigitalOcean, Render was chosen for its user-friendly interface and reliable infrastructure that ensured smooth deployment and scalability for the backend services.

3.3 Overview of Artificial Neural Network

3.3.1 Fundamentals of Artificial Neural Network

A focused area of the practical part of this project is the implementation of artificial neural networks (ANN). Neural networks are a branch of machine learning that forms the basis for deep learning methods. ANNs are inspired by the way a human brain functions and how real neurons transmit signals to one another. More precisely, neural networks consist of a set of algorithms that identify data patterns and correlations (Qamar and Zardari, 2023).

A fundamental neural network consists of three elements:

- Input layer, which is submitted data, given to the system to learn and reach conclusions based on it. The input nodes forward the data to the next part, which is the hidden layer.

- Hidden layer, which is a part where all the necessary computations are performed. One ANN can have numerous hidden layers, for instance this project uses two hidden layers demonstrated in figure 3.
- Output layer, which is designed to showcase conclusion from the computations in the hidden layer.

```
model.add(Dense(50, input_dim=X.shape[1], kernel_initializer='normal', activation='relu'))  
model.add(Dense(25, activation='relu'))
```

FIGURE 4. Demonstrates two hidden layers of neural network: the first with 50 units and rectified linear unit (ReLU) activation, and the second with 25 units and ReLU activation, designed to learn complex patterns in the input data.

Next figure demonstrates the correlation between all three layers in a simple neural network.

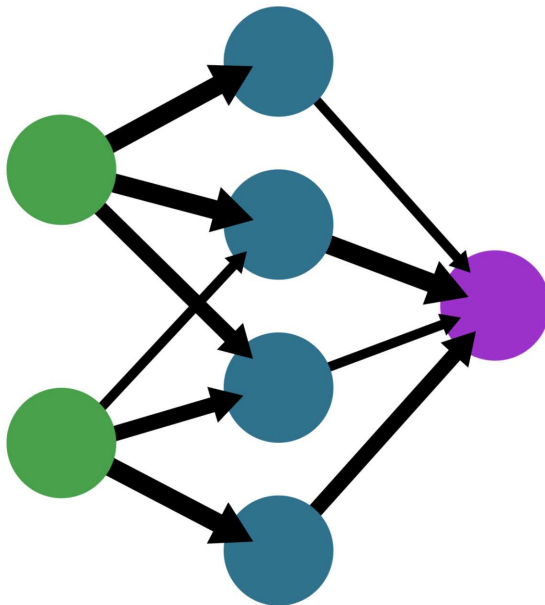


FIGURE 5. Shows ANN layers with colors, where green is the input layer, blue and the hidden layer and purple is the output layer (Deepanshi, 2021).

In summary, the interaction between the input, hidden, and output components enables the neural network to effectively process and transform data. This structure is fundamental to the network's ability to perform tasks such as classification used in this project to identify code's complexity.

3.3.2 ANN technical environment overview

This part of the project was developed in a controlled environment using Anaconda, which provides an isolated workspace with all the required dependencies for machine learning tasks. It includes over 1,500 open-source packages and tools, such as NumPy, SciPy, and TensorFlow, which are essential for building, training, and deploying machine learning models. Anaconda's environment management system helps in maintaining multiple environments, allowing users to install and manage libraries specific to different projects (Anaconda [THL], 2025).

Within this environment, the Python program is developed in Spyder (Scientific Python Development Environment), which is an open-source integrated development environment designed for scientific programming in Python. Spyder offers an advanced code editor with syntax highlighting, real-time code analysis, and debugging tools. Additionally, it provides an interactive console, variable explorer, and built-in help system, all of which enhance the development and testing of scientific applications, making it well-suited for this project's Artificial Neural Network development (Spyder [THL], 2025).

4 IMPLEMENTATION

The practical work consists of custom trained ANN model integrated in functional backend using Python and connected through FastAPI to interactive frontend. The class diagram of the web platform can be examined below.

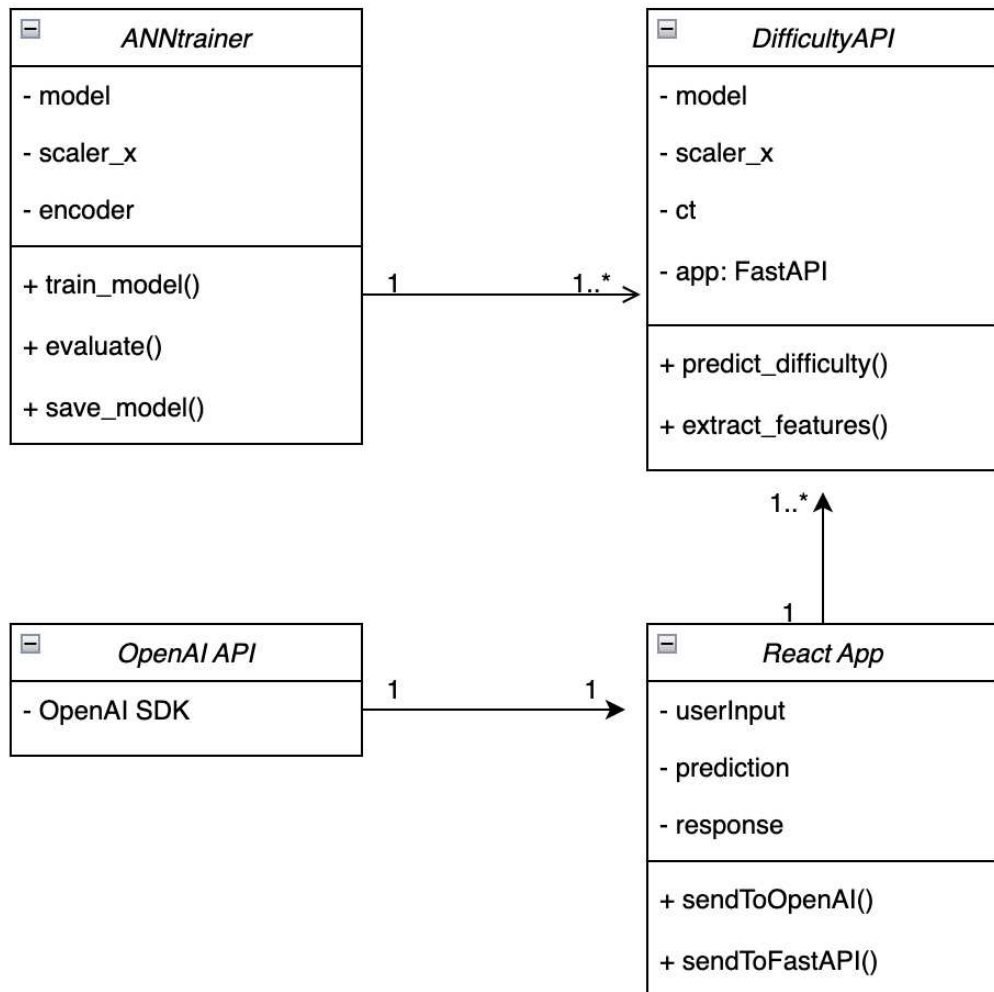


FIGURE 6. Class Diagram of the practical part.

This chapter aims to describe implementation process and components involved in the application's development.

4.1 Frontend configuration

The front-end of the practical work was developed using React.js, with its background further examined in Chapter 3.1, this section describes the configuration process, including project setup, essential dependencies, and component structure.

4.1.1 Project setup and dependencies

The development environment was configured using Create React App (CRA), which streamlines the setup process by providing a pre-configured React environment.

```
(base) dianaefimova@MacBook-Pro-Diana ~ % npx create-react-app devboost
```

FIGURE 7. Demonstrates CRA command.

The project also integrates several essential libraries to enhance functionality.

- Bootstrap – for responsive UI components and styling.
- Axios – to handle API requests, enabling communication with the backend.
- Framer Motion – to implement smooth animations and enhance user experience.
- React Icons – for adding scalable vector icons within the UI.

The dependencies were installed using the command demonstrated below.

```
(base) dianaefimova@MacBook-Pro-Diana ~ % npm install bootstrap axios framer-motion react-icons
```

FIGURE 8. Demonstrates the installment command.

4.1.2 Component structure

The application is structured into multiple reusable components, each handling a distinct part of the interface and functionality:

- MainPage.js – The main component that manages user interactions, API requests, and responses. Full code is available in Appendix 1.

- Background.js – A separate component responsible for rendering the animated background. Full code is available in Appendix 2.
- OpenAIService.js – A service module that communicates with OpenAI's API to retrieve responses. Full code is available in Appendix 3.

Additionally, several essential React hooks are utilized to enhance functionality and manage different aspects of the application.

- The useState hook is used to manage component state, enabling dynamic updates to user input, AI responses, and interface elements.
- The useEffect hook is utilized to handle side effects, such as fetching data or updating the UI in response to state changes, ensuring smooth interactions.
- The useRef hook is applied to reference specific UI elements dynamically, allowing direct manipulation without causing unnecessary re-renders.

The utilized hooks form an essential part of the application's architecture, supporting various features as state management, timers, fetching data from an API, and storing values within the component-based framework of React.

4.1.3 Key functionalities

The React frontend includes several essential features presented below.

- AI Role Selection: Users can choose an AI role before interacting with the chatbot.
- Code Input and Analysis: Users input code, which is processed to provide AI-generated feedback while also offering difficulty assessment, which comes from projects backend and described further in Chapter 4.2.
- Tutorial Overlay: A step-by-step guide highlights UI elements to help users navigate the application.
- Dynamic UI Elements: Buttons and menus toggle visibility based on user actions to ensure a clean and interactive interface.

By leveraging React.js, along with external libraries and APIs, the platform delivers an intuitive and responsive learning environment tailored for programming education.

4.1.4 UI design and responsiveness

The front-end of this application follows a minimalistic yet functional design approach, ensuring clarity and ease of navigation. The primary goal of the design is to support users' learning experience by minimizing unnecessary distractions. The color scheme primarily consists of dark tones with vibrant highlights, providing a visually appealing contrast that enhances readability. Additionally, elements like buttons and text areas are styled for consistency, making interactions seamless and engaging.

When the page is opened for the first time the highlighted tutorial is shown for new users, going over all features the tutorial explains the basic functionality of the application. The tutorial is displayed only once for new users, by utilizing localStorage, and does not interrupt the experience of returning users. The code is available in Appendix 1.

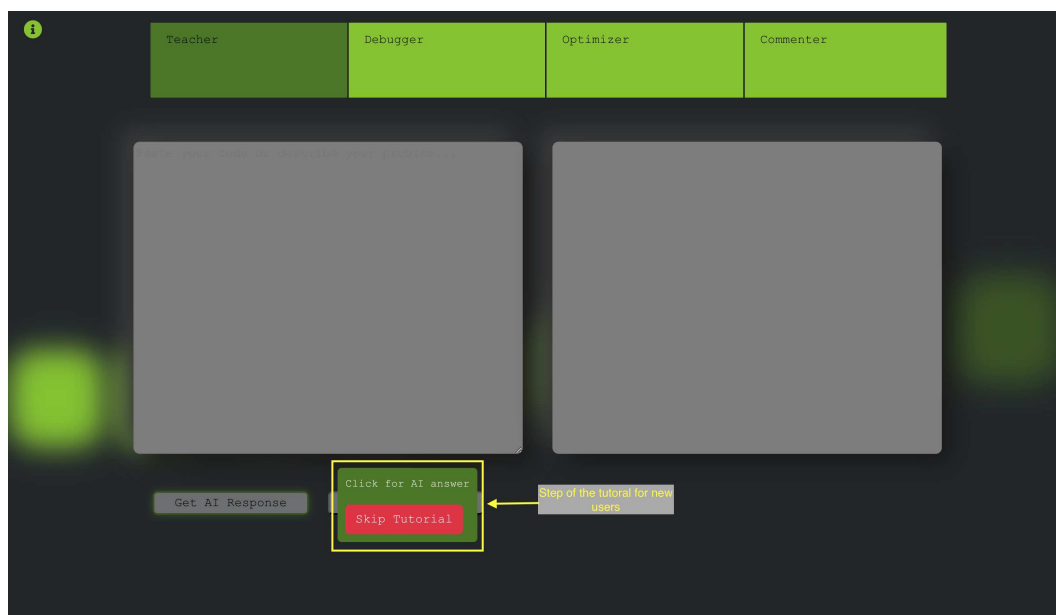


FIGURE 9. Shows the platform's tutorial.

The application provides clearly defined options of pre-trained AI roles, which users can select from the top of the page.

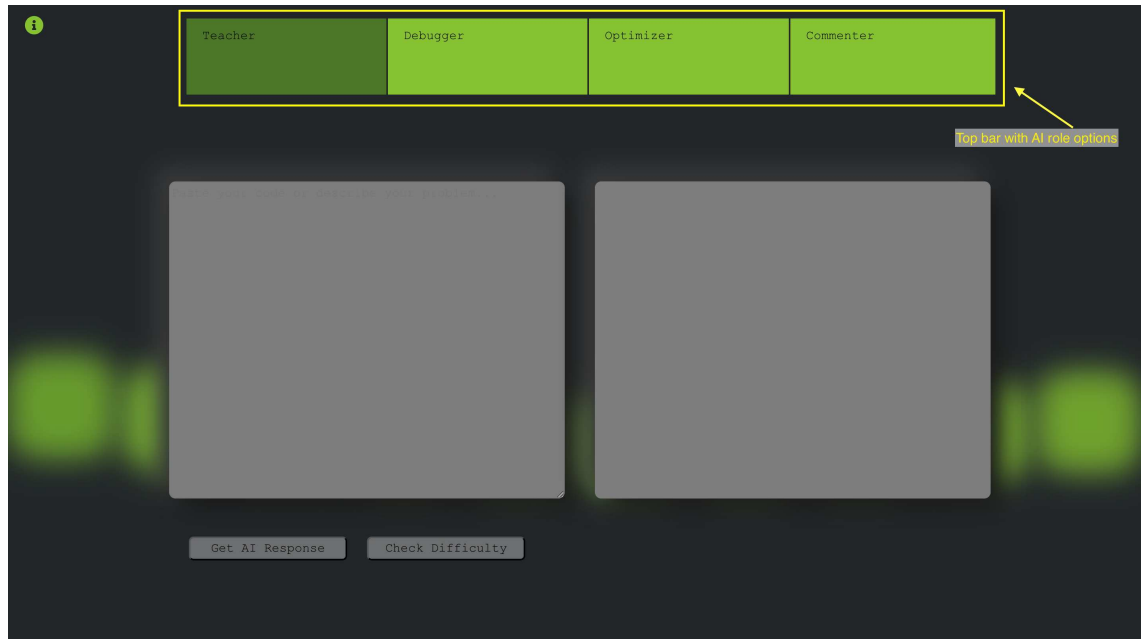


FIGURE 10. Showcases UI view with top bar highlighted.

In the center of the user interface, two windows are available: one for user code input and another for the generated response, which appears after clicking the “Get AI Response” button, based on the selected role. Both windows offer scroll through option for easier text visibility.

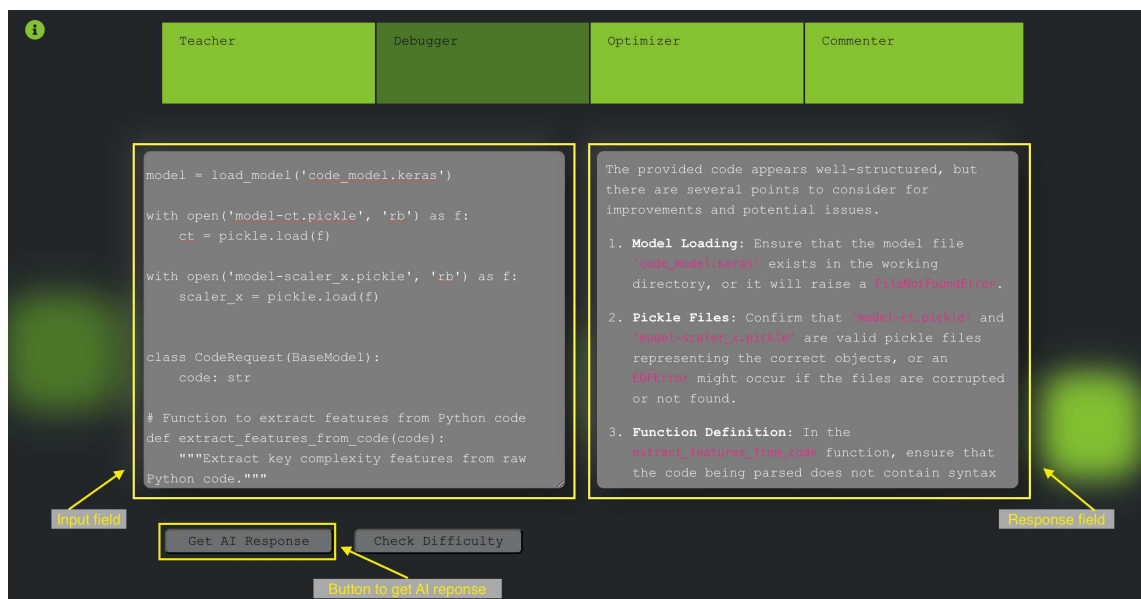


FIGURE 11. Shows UI view with user’s input and AI response.

An additional feature, the classification of the user's code difficulty, can be accessed by clicking the “Check Difficulty” button, which fetches response from backend. The source code for backend can be accessed in Appendix 5.

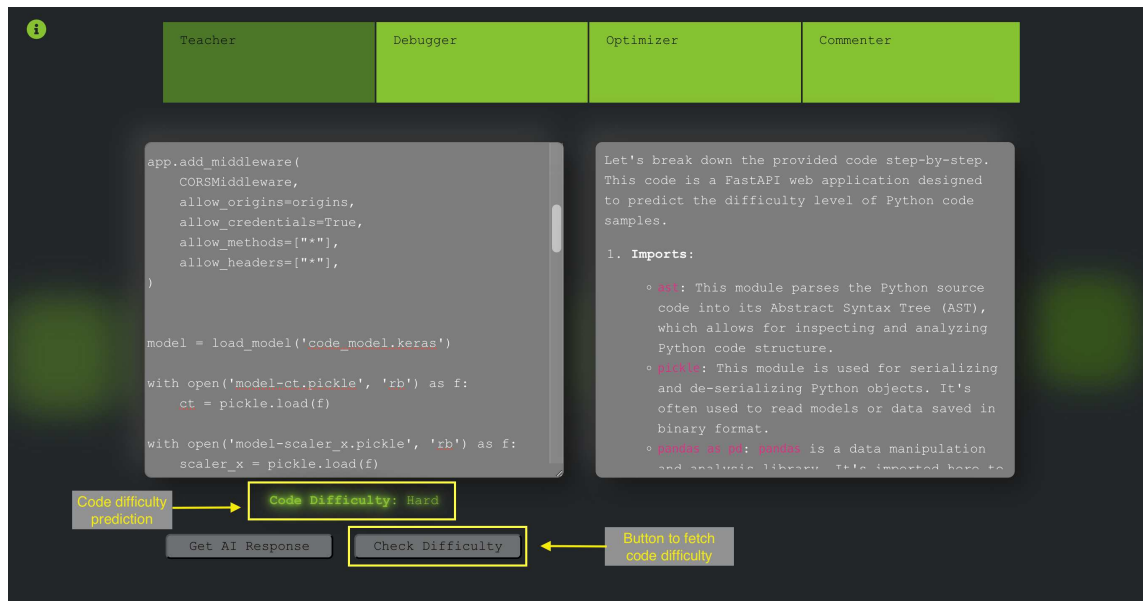


FIGURE 12. Demonstrates UI view with classified code difficulty.

If users are unsure how to operate the application, they can access the additional tutorial at any time by clicking the info icon.

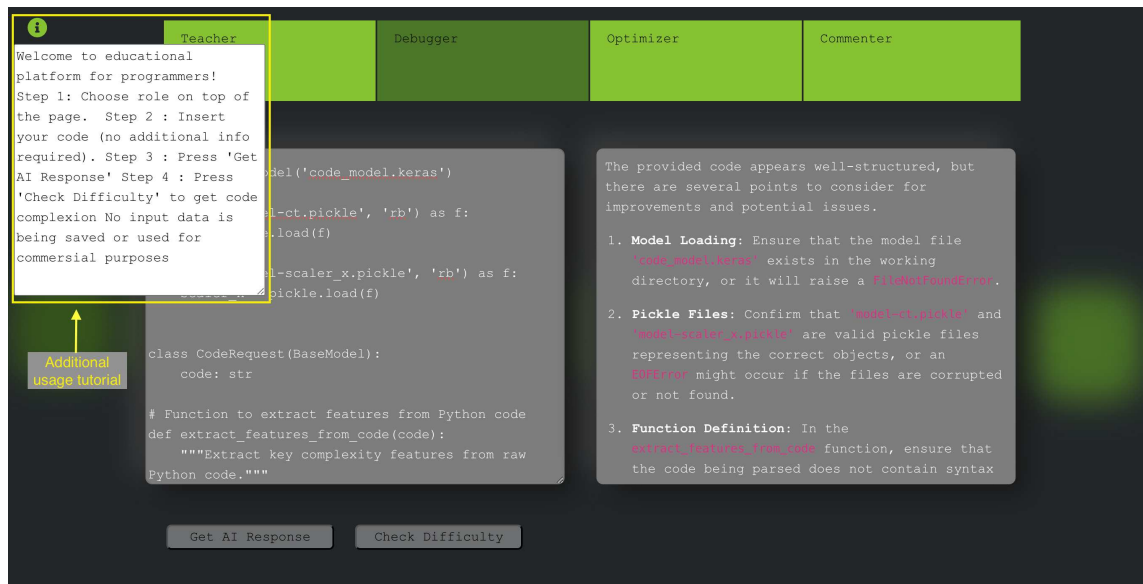


FIGURE 12. Shows UI view with additional usage application usage tutorial.

All buttons and icons are intuitively labeled, ensuring that even first-time users can easily understand how to navigate the application.

To improve accessibility and usability across various devices, the platform is designed with responsiveness in mind. The layout dynamically adapts to different screen sizes using CSS media queries, ensuring proper alignment and spacing whether accessed on a desktop, tablet, or mobile device. Interactive elements, such as buttons and text fields, adjust their sizes to accommodate touch interactions on smaller screens while maintaining a structured design on larger displays. The source code for styling smaller screen can be viewed in Appendix 6.

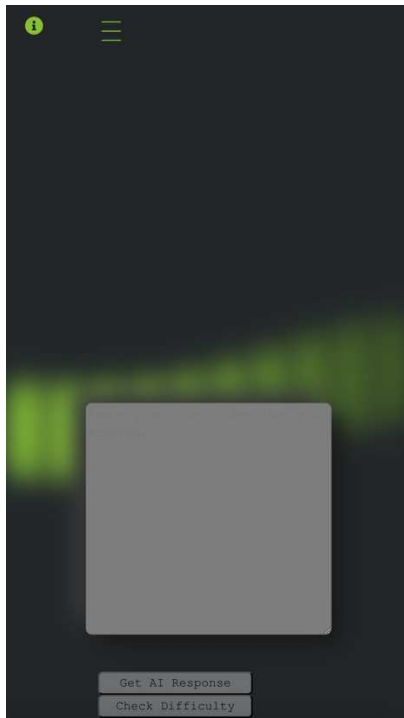


FIGURE 13. Demonstrates front page for mobile view.

The main difference between the full-size UI viewnder and the small-screen view is the merging of AI role options into a menu button to accommodate limited screen space.

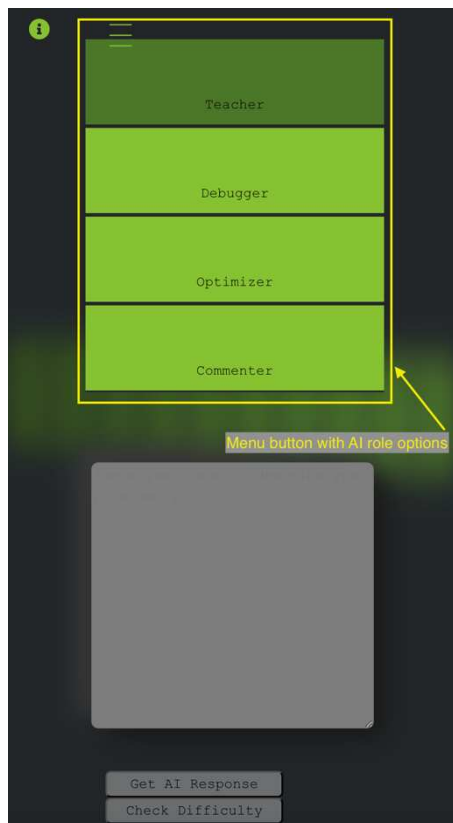


FIGURE 14. Shows front page for mobile view with menu options.

Another distinction between views is the absence of response window on smaller screens to maintain clarity and prevent visual clutter.

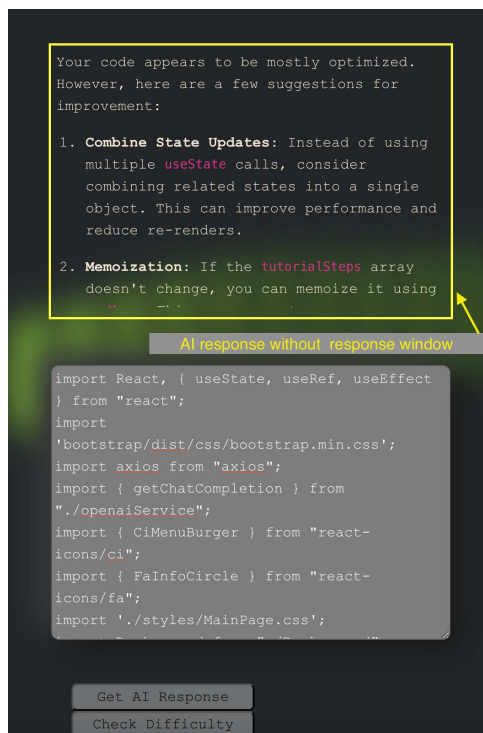


FIGURE 14. Showcases UI view for smaller screens with AI response.

The UI decisions were made with the aim of providing a pleasant user experience, designed to support users' learning objectives and minimize distractions from educational processes.

4.2 ANN configuration

4.2.1 Development environment

ANN model was trained using Anaconda, described in Chapter 2.3.2. The development included an integration of various technologies, as showcased in the list of key development tools below.

- Python: Version 3.8, used for writing scripts to preprocess data, train the machine learning model, and deploy the model via FastAPI.
- Jupyter Notebooks: Used for data exploration and experimentation, allowing quick feedback on data transformations and machine learning models.
- FastAPI: A modern web framework for building APIs, used to expose the trained model as a service that the frontend can interact with.
- Uvicorn: ASGI server used to run the FastAPI application locally and in production.
- TensorFlow and Keras: Libraries used for building and training the neural network model.
- Scikit-learn: Used for data preprocessing, such as scaling and encoding, and model evaluation.
- Pickle: A Python library used to save and load models and pre-processing objects.

The combination of these tools provided a powerful and efficient environment for developing and training the artificial neural network model, ensuring smooth integration and optimal performance throughout the project.

4.2.2 Dataset

The machine learning model aims to predict the difficulty of users' Python code based on certain features that characterize the complexity of the code. These features include the number of lines of code, the number of functions, loops, conditionals, and whether recursion is present in the code.

The dataset used for training consists of 6 columns csv (Comma-separated values) file with the following information:

- Num_Lines: The number of lines in a code.
- Num_Functions: The number of functions defined in the code.
- Has_Recursion: A binary feature that indicates whether the code contains recursion.
- Num_Loops: The number of loops (For and While loops) present in the code.
- Num_Conditionals: The number of conditionals (If statements) in the code.
- Difficulty: The number identifying a code difficulty, where 0 is easy and 2 is hard

The sample of data used for training the model is presented below.

Num_Lines	Num_Functions	Has_Recursion	Num_Loops	Num_Conditionals	Difficulty
62	1	0	6	10	2
74	0	0	3	2	2
6	3	0	7	8	2
95	0	1	7	3	2
21	6	1	4	9	2
13	4	1	1	1	1
84	6	1	10	7	2
46	3	0	8	7	2
45	8	1	0	7	2
76	7	0	0	3	2
82	8	1	9	7	2
87	1	0	1	8	2
23	6	1	5	7	2

FIGURE 12. Demonstrates the sample of data used for training ANN model.

To ensure the features were suitable for machine learning, numerical features were standardized using StandardScaler from scikit-learn. Additionally, the target variable, representing the difficulty level of the code, was one-hot encoded using Keras' to_categorical function to facilitate multi-class classification.

4.2.3 Model training

A feedforward neural network (MLP) was chosen as the model for this task due to its ability to learn complex, non-linear relationships between input features. The architecture of the neural network was as follows:

1. **Input Layer:** The input layer receives the preprocessed feature vector, which includes the encoded programming language and numerical features (lines, functions, loops, conditionals, recursion).
2. **Hidden Layers:** The model includes two hidden layers. The first layer has 50 neurons, and the second layer has 25 neurons, both using the ReLU activation function to introduce non-linearity.
3. **Output Layer:** The output layer consists of three neurons corresponding to the three difficulty classes (Easy, Medium, and Hard). A softmax activation function was applied to provide probabilities for each class.

The model was compiled using categorical cross-entropy as the loss function, suitable for multi-class classification, and Adam as the optimizer, chosen for its efficiency in training deep learning models. Accuracy was used as the evaluation metric to track the model's performance.

The model was trained over 100 epochs, using a batch size of 32, with validation on a separate test set to monitor performance and prevent overfitting.

```
# Creating and teaching a neural network
model = Sequential()
model.add(Dense(50, input_dim=X.shape[1], kernel_initializer='normal', activation='relu'))
model.add(Dense(25, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=1, validation_data=(X_test,y_test))
```

FIGURE 13. Demonstrates teaching of a neural network.

Full source code of ANN training is available in Appendix 4.

saved using Pickle for future use. This allows the model to be loaded and used in production environments.

```
# Save model
model.save('code_model.keras')

# Save encoder and scaler to disk
import pickle
with open('model-ct.pickle', 'wb') as f:
    pickle.dump(ct, f)
with open('model-scaler_x.pickle', 'wb') as f:
    pickle.dump(scaler_x, f)
```

FIGURE 15. Demonstrates saving of a neural network model.

4.3 Backend configuration

4.3.1 Data processing

In order to classify user input using the ANN model, it is loaded in the backend and the features described in Chapter 3.2.2 are extracted. This is done through the “extract_features_from_code” function, which analyses the structure of the user's code using Python's Abstract Syntax Tree (AST). It calculates key metrics such as the number of lines, functions, loops, and conditionals, and detects whether the code contains recursion. These features provide a simplified representation of the code's complexity, which is then used by the model for classification.

```

# Function to extract features from Python code
def extract_features_from_code(code):
    """Extract key complexity features from raw Python code."""
    tree = ast.parse(code)

    num_lines = len(code.splitlines())
    num_functions = sum(isinstance(node, ast.FunctionDef) for node in ast.walk(tree))
    num_loops = sum(isinstance(node, (ast.For, ast.While)) for node in ast.walk(tree))
    num_conditionals = sum(isinstance(node, ast.If) for node in ast.walk(tree))

    has_recursion = 0
    for node in ast.walk(tree):
        if isinstance(node, ast.FunctionDef):
            for sub_node in ast.walk(node):
                if isinstance(sub_node, ast.Call) and isinstance(sub_node.func, ast.Name):
                    if sub_node.func.id == node.name:
                        has_recursion = 1
                        break

    return {
        "Num_Lines": num_lines,
        "Num_Functions": num_functions,
        "Num_Loops": num_loops,
        "Num_Conditionals": num_conditionals,
        "Has_Recursion": has_recursion
    }

```

FIGURE 16. Demonstrates ast module.

Once the preprocessing steps are complete, the data is classified using the trained model and sent to frontend using POST method. The program includes exception handling to ensure the application does not crash but instead returns an HTTP 500 error. The source code can be examined in Appendix 5.

The backend is a minimalistic implementation developed specifically for this thesis, designed solely to handle requests and deliver responses from the ANN model.

4.3.2 Model deployment

To utilize the trained ANN model, a backend API was developed using FastAPI, a modern web framework built on top of Starlette and Pydantic.

FastAPI was used to define and expose the API endpoints. It handles HTTP requests and responses, dependency injection, and async support. Meanwhile, input data validation is handled by Pydantic, which provides the “BaseModel” class. A subclass “CodeRequest” is defined to ensure the incoming JSON contains a valid code string. Pydantic automatically parses and validates the request data before it reaches the endpoint logic.

To run FastAPI application, Uvicorn was used. It allows for high-performance, asynchronous execution, making the API suitable for production or local development.

CORS Middleware was added to handle Cross-Origin Resource Sharing, which is necessary when a frontend runs on a different port. It ensures that the browser doesn't block the API requests due to origin mismatch. It was integrated in order to insure smooth testing process. The code example can be observed in Appendix 5.

In the end, the backend was deployed on Render, a cloud platform, to ensure accessibility from anywhere. The platform's free tier was utilized for this project, providing sufficient resources at this stage as the application is in its MVP phase. The free tier offers 750 hours of compute time per month, which is sufficient for full-time operation. However, it has some limitations, such as shutting down after 15 minutes of inactivity, which results in longer response times for the following users (Liski, 2025). For the next version of the application, it is recommended to consider higher-tier plans or alternative deployment platforms such as AWS.

The communication between the backend and the frontend is done through HTTP requests. The frontend, which is built using React, sends user-submitted Python code to the backend API, and the API responds with the predicted difficulty level. This architecture allows the system to offer real-time predictions for Python code complexity.

4.4 Application evaluation

AI-powered web platform was developed as part of this thesis to demonstrate example of proper utilization of artificial intelligence for educational purposes. This application is currently in its Minimum Viable Product stage, focusing on core functionality to showcase the potential of AI in educational contexts. It consists of three parts:

- User Interface: leveraging OpenAI API it successfully provides user with four options to evaluate their code and learn new ways to improve, avoiding code generation and emphasizing on development of critical thinking skills.
- Custom trained ANN model: the model, trained on extensive data, evaluates users code difficulty with the 98% accuracy.
- FastAPI connection: API is designed to connect ANN model with user interface and extract features from users inputted code in order to predict their difficulty.

All parts of the application work continuously together aiming to deliver fast learning progress for developers. The application intends to be more effective than other AI platforms for educational purposes by eliminating the need to pre-train AI model, therefore saving time and supporting users to stay focused on one task at a time.

To evaluate the application, it was compared to ChatGPT's operations. Using the same input – solely code with no context, the platforms responses were examined.

ChatGPT produced moderately long response consisting of light coverage on many topics with generated parts of code and proposal to execute the work for users. ChatGPT's answer can be examined in Appendix 7.

Meanwhile, this thesis's practical work demonstrated a straightforward response concentrating on one goal, teaching, which is the default option for users. The platform evaluated the code and gave step by step break down of the inputted code. The application's response can be observed in Appendix 8.

The evaluation demonstrated that, for accomplishing specific tasks with minimal user effort, the practical work presented in this thesis is efficient in terms of time and exertion. However, assessing its long-term impact on learning outcomes would require a dedicated study group, which falls outside the scope of this thesis.

Despite the promising results, the system is not without limitations. The ANN model, while highly accurate, was trained on synthetic and pre-labeled datasets, which may not always reflect real-world learning environments. Additionally, the code prediction is currently limited to Python code recognition, which is planned to be improved by adding the ability to evaluate multiple programming languages. Future initiatives also include implementation of more AI-powered options for users and collecting user feedback could help further refine the system's performance and educational impact.

Overall, the platform demonstrates the possibility of integrating AI into coding education in a structured and user-friendly manner. It highlights the system's potential as a supportive learning tool that not only aligns with best practices but also empowers users to enhance their programming skills effectively and responsibly.

5 DISCUSSION

This thesis' work explores the application of artificial intelligence technologies in the field of education by examining established web platforms that have recently integrated neural network models into their interfaces, as well as newly emerged platforms whose products are based entirely on AI technologies. The study highlights the benefits and drawbacks of AI for academic objectives and addresses concerns related to its potential overuse.

The practical part of the thesis showcases the ethical use of artificial intelligence to maximize learning outcomes while addressing concerns related to content overgeneration of content and the decline of critical thinking. This is achieved by providing pre-trained models tailored to specific code-related tasks, with a strong focus on education. The ultimate goal is to foster users' independence, enabling them to become self-sufficient programmers rather than relying on AI-generated solutions. However, the long-lasting benefits of the work are yet to be studied over longer term with multiple users.

The main contribution of this work is the exploration of artificial intelligence, a fast-growing and widely used technology that is still lacking proper regulations and deeper academic analysis due to how new it is. This study helps address that gap by focusing on its role and risks in the context of education. Moreover, this study is valuable as it presents an open-source, AI-driven platform designed for studies benefits, making it accessible for learning and practical use.

The main highlights of the practical work in this thesis include:

- Four pretrained models, each tailored to a specific role (teacher, debugger, optimizer, commentor), designed to focus the user's attention on particular tasks and avoid information overload, thereby maximizing learning outcomes.
- Prevention of code generation to reduce users' reliance on AI assistance, encouraging the development of critical thinking skills.
- Code difficulty analysis, providing users with a clearer understanding of the complexity of their own work.

The future development of the presented platform will focus on enhancing the ANN model for predicting code complexity by incorporating a larger dataset. Additionally, more pretrained models will be integrated to expand the range of tasks available to users. The platform has been designed without a database to ensure that users' work is not stored or susceptible to exploitation, and therefore, the addition of a database is not planned for future development.

The source code for the practical part is provided in Appendices 1–5 and is also accessible on GitHub: the backend can be found at <https://github.com/dianaefimova/ANN-training>, while the frontend is available at <https://github.com/dianaefimova/ScriptaAI>. Furthermore, the application is freely accessible online at <https://scriptaai.web.app>.

This thesis acknowledges making limited use of AI tools such as OpenAI's ChatGPT, and GitHub Copilot. While some content and code were assisted by AI, academic integrity has been preserved throughout the work.

In conclusion, the thesis presents an analysis of how artificial intelligence is being applied for learning purposes and its ethical concerns. Furthermore, the practical part developed as part of this thesis can serve as an example of the conscientious use of AI technologies and can be utilized by students to enhance their programming skills.

REFERENCES

Cook, J. 2022. Building a responsive design in React – Best practices for creating a responsive design that looks great on any device. Medium. [online]. Read on 24.01.2025. <https://medium.com/%40jadencook.us/building-a-responsive-design-in-react-best-practices-for-creating-a-responsive-design-that-looks-72c11d0854fd>

DigitalOcean. n.d. How to make requests to an API using Axios in React. [online]. Read on 24.01.2025. <https://www.digitalocean.com/community/tutorials/react-axios-react>

Motion One. n.d. Documentation. [online]. Read on 24.01.2025. <https://motion.dev/docs>

Krishnan, A. 2022. How to Consume REST APIs in React. freeCodeCamp. [online]. Read on 25.01.2025. <https://www.freecodecamp.org/news/how-to-consume-rest-apis-in-react>

Amazon Web Services. n.d. What is a RESTful API?. AWS. [online]. Read on 25.01.2025. <https://aws.amazon.com/what-is/restful-api>

GeeksforGeeks. n.d. REST API Introduction. [online]. Read on 25.01.2025. <https://www.geeksforgeeks.org/rest-api-introduction/>

Design Gurus. n.d. What Technology is Used by OpenAI. DesignGurus.io. [online]. Read on 30.01.2025. <https://www.designgurus.io/answers/detail/what-technology-is-used-by-openai>

Encyclopædia Britannica. n.d. Python (computer language). [online]. Read on 05.02.2025. <https://www.britannica.com/technology/Python-computer-language>

Python Software Foundation. 2025. The Python Language Reference — Python 3.13.2 documentation. [online]. Read on 05.02.2025. <https://docs.python.org/3/reference/index.html>

Adebayo, A. 2023. Machine Learning/Deep Learning with C++ vs Python. Medium. [online]. Read on 06.02.2025. <https://medium.com/@reachadeon/machine-learning-deep-learning-with-c-vs-python-5982f6139080>

Spyder IDE. n.d. Spyder: The Scientific Python Development Environment. [online]. Read on 06.02.2025. <https://www.spyder-ide.org>

Molchanov, A. 2022. Building Neural Networks in Python. Real Python. [online]. Read on 06.02.2025. <https://realpython.com/python-ai-neural-network>

TensorFlow. 2025. *Introduction to TensorFlow*. [online]. Read on 14.02.2025. <https://www.tensorflow.org/learn>

Elshamy, H. 2023. Artificial Neural Networks: An Overview. ResearchGate. [online]. Read on 17.02.2025. https://www.researchgate.net/publication/373700317_Artificial_Neural_Networks_An_Overview

Analytics Vidhya. 2021. Beginner's Guide to Artificial Neural Network. [online]. Read on 18.02.2025. <https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-artificial-neural-network>

Study.com. 2024. Teachers Change Minds About AI. Teaching License. [online]. Read on 25.02.2025. <https://teachinglicense.study.com/featured-insights/teachers-change-minds-about-AI.html>

Digital Education Council. 2024. What Students Want: Key Results from DEC Global AI Student Survey. [online]. Read on 25.02.2025. <https://www.digitaleducationcouncil.com/post/what-students-want-key-results-from-dec-global-ai-student-survey-2024>

Heater, B. 2024. Duolingo CEO on Gamification and Generative AI. The Verge. [online]. Read on 26.02.2025. <https://www.theverge.com/24267841/luis-von-ahn-duolingo-owl-language-learning-gamification-generative-ai-android-decoder>

Khan Academy. n.d. [homepage]. [online]. Read on 01.03.2025.

<https://www.khanacademy.org/>

Dicerbo, K. 2024. Kristen DiCerbo on the Role of AI in Education. TIME Magazine. [online]. Read on 02.03.2025. <https://time.com/7012801/kristen-dicerbo>

StratX Simulations. 2023. The Negative Effects of Artificial Intelligence in Education. [online]. Read on 07.03.2025. <https://web.stratxsimulations.com/recent-posts/the-negative-effects-of-artificial-intelligence-in-education>

Walden University. 2024. Five Pros and Cons of AI in the Education Sector. [online]. Read on 07.03.2025. <https://www.waldenu.edu/programs/education/resource/five-pros-and-cons-of-ai-in-the-education-sector>

CodeSignal. n.d. [homepage]. [online]. Read on 07.03.2025. <https://codesignal.com/>

ELSA Speak. n.d. [homepage]. [online]. Read on 07.03.2025. <https://www.elsaspeak.com/>

Tiangolo, S. n.d. FastAPI Documentation. [online]. Read on 02.04.2025. <https://fastapi.tiangolo.com/>

IIEP-UNESCO. 2025. Swift justice: Universities revamp academic integrity policies amid AI misuse surge. [online]. Read on 17.05.2025. https://etico.iiep.unesco.org/en/swift-justice-universities-revamp-academic-integrity-policies-amid-ai-misuse-surge?utm_source=chatgpt.com

UNESCO. 2023. The Use of AI in Education: Deciding the Future We Want. [online]. Read on 17.04.2025. <https://www.unesco.org/en/articles/use-ai-education-deciding-future-we-want>

TechTarget. 2025. AI as a service democratizes benefits of new tech tools. [online]. Read on 17.05.2025. https://www.techtarget.com/searchenterpriseai/feature/AI-as-a-service-democratizes-benefits-of-new-tech-tools?utm_source=chatgpt.com

Pydantic. n.d. Pydantic models documentation. [online]. Read on 20.05.2025.
<https://docs.pydantic.dev/latest/concepts/models/>

OpenAI. n.d. OpenAI models documentation. [online]. Read on 21.05.2025.
<https://platform.openai.com/docs/models>

Firebase. n.d. Firebase: Build apps for any platform. [online]. Read on 27.04.2025. <https://firebase.google.com/>

TechCrunch. 2015. Microsoft shocks the world with Visual Studio Code, a free code editor for OS X, Linux, and Windows. [online]. Read on 28.04.2025.
<https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows/>

Visual Studio Code. n.d. License for Visual Studio Code. [online]. Read on 28.04.2025. <https://code.visualstudio.com/license>

Render. n.d. Render: Cloud platform for developers. [online]. Read on 28.04.2025. <https://render.com/>

Inferno.js. 2025. About Inferno. [online]. Read on 22.04.2025. <https://www.infernojs.org/about>

Vue.js. 2025. The Progressive JavaScript Framework. [online]. Read on 22.04.2025. <https://vuejs.org/>

Bejamas. 2025. Hosting Render: Next-Gen Web Hosting Solutions. [online]. Read on 07.05.2025. <https://bejamas.com/hub/hosting/render>

Liski, S. 2025. How to run a full-time app on Render's free tier without it sleeping. [online]. Read on 07.05.2025. <https://sergeiliski.medium.com/how-to-run-a-full-time-app-on-renders-free-tier-without-it-sleeping-bec26776d0b9>

APPENDICES

Appendix 1. Source code for MainPage.js.

1 (6)

```

import React, { useState, useRef, useEffect } from "react";
import 'bootstrap/dist/css/bootstrap.min.css';
import axios from "axios";
import { getChatCompletion } from "./openaiService";
import { CiMenuBurger } from "react-icons/ci";
import { FaInfoCircle } from "react-icons/fa";
import './styles/MainPage.css';
import Background from "./Background";
import { motion } from "framer-motion";
import ReactMarkdown from 'react-markdown';

const tutorialSteps = [
  { id: 1, text: "Choose a role for AI assistant", target: "roleButtons" },
  { id: 2, text: "Paste your code, no need for any other information", target: "inputField" },
  { id: 3, text: "Click for AI answer", target: "aiResponseButton" },
  { id: 4, text: "Click to get your code complexity", target: "checkDifficultyButton" },
];

const MainPage = () => {
  const [userInput, setUserInput] = useState("");
  const [response, setResponse] = useState("");
  const [loading, setLoading] = useState(false);
  const [selectedAI, setSelectedAI] = useState("teacher");
  const [prediction, setPrediction] = useState("");
  const [loading2, setLoading2] = useState(false);
  const [showButtons, setShowButtons] = useState(false);
  const [step, setStep] = useState(0);
  const [tooltipStyle, setTooltipStyle] = useState({});
  const [tooltipText, setTooltipText] = useState("");
  const [showInfo, setShowInfo] = useState(false);

  // References for tutorial step highlights
  const roleButtonsRef = useRef(null);
  const inputFieldRef = useRef(null);
  const aiResponseButtonRef = useRef(null);
  const checkDifficultyButtonRef = useRef(null);

  const refs = {
    roleButtons: roleButtonsRef,
    inputField: inputFieldRef,
    aiResponseButton: aiResponseButtonRef,
    checkDifficultyButton: checkDifficultyButtonRef,
  };
};

```

2 (6)

```

useEffect(() => {
  const tutorialComplete = localStorage.getItem("Tutorial") //check in local storage if tutorial has been completed

  if(tutorialComplete === "true") {
    setStep(tutorialSteps.length);
  } else {
    setStep(0); // Ensure tutorial starts when the page loads
  }
}, []);

useEffect(() => {
  Object.values(refs).forEach(ref => ref.current?.classList.remove("highlight"));

  const currentStep = tutorialSteps[step];
  if (currentStep && refs[currentStep.target]?.current) {
    const element = refs[currentStep.target].current;
    element.classList.add("highlight");

    setTooltipText(currentStep.text);

    // Position the tooltip near the highlighted element
    const rect = element.getBoundingClientRect();
    setTooltipStyle({
      top: rect.top + window.scrollY - 60,
      left: rect.left + window.scrollX + rect.width / 2,
      transform: "translateX(-50%)",
    });

    // Move to next step automatically after 3 seconds
    const timeout = setTimeout(() => {
      if (step < tutorialSteps.length - 1) {
        setStep(prevStep => prevStep + 1);
      } else {
        localStorage.setItem("Tutorial", "true");
        setTooltipStyle({display: "none"});
        Object.values(refs).forEach(ref => ref.current?.classList.remove("highlight"));
      }
    }, 3000);

    return () => clearTimeout(timeout);
  } else {
    setTooltipStyle({display: "none"});
    Object.values(refs).forEach(ref => ref.current?.classList.remove("highlight"));
  }
}

```

```
}, [step]);

const skipButton = () => {
  localStorage.setItem("Tutorial", "true");
  setStep(tutorialSteps.length);
  setTooltipStyle({display: "none"});
  Object.values(refs).forEach(ref => ref.current?.classList.remove("highlight"));
};

// Making tutorial start only when new page is opened

const handleShowButtons = () => {
  setShowButtons(!showButtons)
};

const InfoButton = () => {
  setShowInfo(!showInfo)
};

const handleInputChange = (e) => {
  setUserInput(e.target.value);
};

// Function to send code to backend for difficulty prediction
const handleCheckDifficulty = async () => {
  if (!userInput.trim()) {
    alert("Please enter some code before checking difficulty.");
    return;
  }

  setLoading2(true);

  try {
    const response = await axios.post("https://ann-training.onrender.com/predict/", {
      code: userInput,
    });
    setPrediction(response.data.difficulty);
  } catch (error) {
    console.error("Error predicting difficulty:", error);
    setPrediction("Error in prediction. Insert Python code.");
  }

  setLoading2(false);
};
```

4 (6)

```

// Predefined AI roles
const aiRoles = {
  teacher: "Act as a programming teacher. When a user provides code, explain every detail of the code thoroughly. Do not generate or modify any code – only explain it.",
  debugger: "Act as a debugger. When a user provides code, debug the code thoroughly. Do not generate or modify any code – only debug the code that the user inputs.",
  optimizer: "Act as a senior developer. When a user provides code, propose ways to optimize the code. Do not generate or modify any code – only propose ways to optimize it.",
  commenter: "Act as a programming assistance. When a user provides code, return the same code with clear comments. Do not generate or modify any code – only provide comments."
};

const handleSubmit = async (e) => {
  e.preventDefault();

  if (!userInput.trim()) {
    alert("Please enter some code.");
    return;
  }

  setLoading(true);
  setResponse("");

  try {
    const messages = [
      { role: "system", content: aiRoles[selectedAI] },
      { role: "user", content: userInput },
    ];
    const apiResponse = await getChatCompletion(messages);
    setResponse(apiResponse);
  } catch (error) {
    setResponse("An error occurred. Please try again.");
  } finally {
    setLoading(false);
  }
};

return (
  <div>
    <Background/>
    {step < tutorialSteps.length && (
      <div className="tooltip-box" style={tooltipStyle}>
        <motion.p key={step} initial={{ opacity: 0 }} animate={{ opacity: 1 }} transition={{ duration: 0.3 }}>
          {tooltipText}
        </motion.p>
        <button onClick={skipButton} className="skipButton btn btn-danger">
          Skip Tutorial
        </button>
      </div>
    )}
  </div>
)

```

5 (6)

```

    <div className="Info">
<button className="info-icon" onClick={InfoButton}>
  <FaInfoCircle/>
</button>

{showInfo && (
  <div className="Info-field">
    <textarea defaultValue="Welcome to educational platform for programmers!
    Step 1: Choose role on top of the page.
    Step 2 : Insert your code (no additional info required).
    Step 3 : Press 'Get AI Response'
    Step 4 : Press 'Check Difficulty' to get code complexion
    No input data is being saved or used for commersial purposes" />
  </div>
)}
</div>
<div className="menu-icon" onClick={handleShowButtons}>
  <CiMenuBurger/>
</div>

<div className="row">

<div ref={roleButtonsRef} className={`buttons ${showButtons && 'active'}}>
  {Object.keys(aiRoles).map((role) => (
    <button
      key={role}
      className={`AIbutton row ${selectedAI === role ? "active" : ""}`}
      type="button"
      onClick={() => setSelectedAI(role)}
    >
      {role.charAt(0).toUpperCase() + role.slice(1)}
    </button>
  ))}
</div>

```

```

    <div className="text">
      <form onSubmit={handleSubmit}>
        <div ref={inputFieldRef} className="input-field">
          <textarea
            className="input-text"
            value={userInput}
            onChange={handleInputChange}
            placeholder="Paste your code or describe your problem..."
            required
          />
          <div className="predictionResult">
            {prediction && <h3><b>Code Difficulty:</b> {prediction}</h3>}
          </div>
        </div>
        <button ref={aiResponseButtonRef} className="submit rounded" onClick={handleSubmit} disabled={loading}>
          {loading ? "Loading..." : "Get AI Response"}
        </button>
        <button ref={checkDifficultyButtonRef} className="submit rounded" onClick={handleCheckDifficulty} disabled={loading2}>
          {loading2 ? "Checking..." : "Check Difficulty"}
        </button>
      </form>
    </div>

    <div className="answer" style={{borderColor: 'white'}}>
      {response && (
        <div className="AI-response">
          <ReactMarkdown>{response}</ReactMarkdown>
        </div>
      )}
    </div>

  </div>
</div>
);
};

export default MainPage;

```

Appendix 2. Source code Background.js

```
import './styles/Background.css';

const Background = () => {

  |   return (
//Reusable background with animation//
<div>
  <div class="center_div">
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
    <div class="wave"></div>
  </div>
</div>
  |   );
  | };

export default Background;
```

Appendix 3. Source code openaiService.js

```
import OpenAI from "openai";

const openai = new OpenAI({
  apiKey: process.env.REACT_APP_OPENAI_API_KEY,
  dangerouslyAllowBrowser: true,
});

export const getChatCompletion = async (messages) => {
  try {
    const completion = await openai.chat.completions.create({
      model: "gpt-4o-mini",
      messages: messages,
    });
    return completion.choices[0].message.content; // Return the assistant's response
  } catch (error) {
    console.error("OpenAI API Error:", error);
    throw error;
  }
};
```

Appendix 4. Source code for ANN training

1 (2)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import seaborn as sns
import pickle
from sklearn.metrics import confusion_matrix, accuracy_score
from keras.utils import to_categorical

df = pd.read_csv('code_difficulty.csv')

# Divide X and y
X = df.iloc[:, 0:5] # Features
y = df.iloc[:, [-1]] # Difficulty

# Dummy variables
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(drop='first'), [0])], remainder='passthrough')

# Apply OneHotEncoder to 'Programming_Language' and scale other features
X = ct.fit_transform(X)

# One-hot encode the target variable (difficulty)
y = to_categorical(y, num_classes=3)

# Train and test data 80 % - 20 %
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 0)

# Scale X
scaler_x = StandardScaler(with_mean=False)
X_train = scaler_x.fit_transform(X_train)
X_test = scaler_x.transform(X_test)
```

```

# Creating and teaching a neural network
model = Sequential()
model.add(Dense(50, input_dim=X.shape[1], kernel_initializer='normal', activation='relu'))
model.add(Dense(25, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=1, validation_data=(X_test,y_test))

# visualize training & evaluate results
print(history.history.keys())
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

test_results = model.evaluate(X_test, y_test, verbose=1)
print(f'\nTest results - Loss: {test_results[0]} - Accuracy: {test_results[1]*100}%')

# Predict with test data
y_pred_proba = model.predict(X_test)
y_pred = y_pred_proba.argmax(axis=1)
y_test = y_test.argmax(axis=1)

# Confusion Matrix and metrics
cm = confusion_matrix(y_test, y_pred)
print(cm)

sns.heatmap(cm, annot=True, fmt='g')
plt.show()

# Accuracy score
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_score
acc = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted')

print(cm)
print (f'accuracy_score: {acc}')
print (f'recall_score: {recall}')
print (f'precision_score: {precision}')

sns.heatmap(cm, annot=True, fmt='g')
plt.show()

# Save model
model.save('code_model.keras')

# Save encoder and scaler to disk
import pickle
with open('model-ct.pickle', 'wb') as f:
    pickle.dump(ct, f)
with open('model-scaler_x.pickle', 'wb') as f:
    pickle.dump(scaler_x, f)

```

Appendix 5. Source code for application's backend

1 (2)

```

import ast
import pickle
import pandas as pd
import uvicorn
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware # Import CORSMiddleware
from pydantic import BaseModel
from tensorflow.keras.models import load_model

app = FastAPI()

# Allow CORS for your React frontend running on localhost:3000 for testing
origins = [
    "http://localhost:3000",
    "http://127.0.0.1:3000",
    "https://scriptaai.web.app",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

model = load_model('code_model.keras')

with open('model-ct.pickle', 'rb') as f:
    ct = pickle.load(f)

with open('model-scaler_x.pickle', 'rb') as f:
    scaler_x = pickle.load(f)

class CodeRequest(BaseModel):
    code: str

# Function to extract features from Python code
def extract_features_from_code(code):
    """Extract key complexity features from raw Python code."""
    tree = ast.parse(code)

    num_lines = len(code.splitlines())
    num_functions = sum(isinstance(node, ast.FunctionDef) for node in ast.walk(tree))
    num_loops = sum(isinstance(node, (ast.For, ast.While)) for node in ast.walk(tree))
    num_conditionals = sum(isinstance(node, ast.If) for node in ast.walk(tree))

    has_recursion = 0
    for node in ast.walk(tree):
        if isinstance(node, ast.FunctionDef):
            for sub_node in ast.walk(node):
                if isinstance(sub_node, ast.Call) and isinstance(sub_node.func, ast.Name):
                    if sub_node.func.id == node.name:
                        has_recursion = 1
                        break

```

```
    return {
        "Num_Lines": num_lines,
        "Num_Functions": num_functions,
        "Num_Loops": num_loops,
        "Num_Conditionals": num_conditionals,
        "Has_Recursion": has_recursion
    }

# API endpoint to predict code difficulty
@app.post("/predict")
async def predict_difficulty(request: CodeRequest):
    try:
        features = extract_features_from_code(request.code)
        df = pd.DataFrame([features]) # Convert to DataFrame

        X_new = ct.transform(df)
        X_new = scaler_x.transform(X_new)

        y_pred_proba = model.predict(X_new)
        y_pred = y_pred_proba.argmax(axis=1)

        difficulty_labels = ['Easy', 'Medium', 'Hard']
        predicted_difficulty = difficulty_labels[y_pred[0]]

        return {"difficulty": predicted_difficulty}

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

# Run the API (only in local development) for testing locally
if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Appendix 6. Source code CSS styling for smaller screens.

1 (2)

```
@media (max-width: 1200px) {  
  .menu-icon {  
    display:inline;  
    color: #86c232;  
    cursor: pointer;  
    z-index: 9999;  
    font-size: 34px;  
    position: absolute;  
    top: 10px;  
    margin-left: 50px;  
  }  
  .buttons {  
    position: absolute;  
    left: auto;  
    top: 30px;  
  
    width: auto;  
    height: 0;  
    transition: height 0.3s ease-in-out;  
    overflow: hidden;  
    flex-direction: column;  
    padding: 0;  
    z-index: 1000;  
    margin-left: 5px;  
  }  
  
  .buttons.active {  
    height: auto;  
    padding: 10px 0;  
  }  
  
  .row {  
    flex-direction: column-reverse;  
    align-items: center;  
  }  
  
  .text {  
  
    display: flex;  
    flex-direction: column;  
    justify-content: flex-end;  
    height: 50vh;  
    width: 80%;  
    padding: 5px;  
    z-index: 1;  
  }  
}
```

```
.input-text {
  width: 100%;
  min-height: 200px;
  box-shadow: 15px 15px 30px ■ rgb(25, 25, 25),
  -15px -15px 30px ■ rgb(60, 60, 60);
  background-color: ■ #7e7d7d;
  border-radius: 8px;
  color: □ white;
  z-index: 2;
  margin-bottom: 10px;
}

.answer {
  margin-top: 40%;
  width: 80%;
  min-height: 60px;
  height: 300px;
  display: flex;
  align-items: center;
  border-radius: 8px;
  padding: 10px;
  z-index: 2;
  position: relative;
}

.AI-response {
  margin: 0%;
  color: □ antiquewhite;
  text-align: left;
  overflow-y: auto;
  max-height: 100%;
  width: 100%;
}
}
```

Appendix 7. ChatGPT's response to a solely code input (ChatGPT [THL], 2025).

Your FastAPI app is well-structured and implements a machine learning inference endpoint for predicting code difficulty based on extracted AST features. Here are some key observations and a few suggestions for improvements or clarifications:

✔ Strengths:

- **Clear modularity:** The `extract_features_from_code` function is well-isolated, making it reusable and easy to test.
- **Model and preprocessing loading:** You're using `pickle` for the `ColumnTransformer` and `scaler`, and loading a Keras model correctly.
- **Pydantic validation:** You correctly validate input using `CodeRequest`.
- **CORS middleware:** Properly allows requests from React and Firebase deployments.

🔍 Suggestions for Improvement:

1. Security Considerations:

Parsing untrusted code using `ast.parse()` is safer than `eval()` or `exec()`, but you should still consider limiting input size or performing sanitation if the API is public.

2. Detailed Error Logging:

Consider logging the error details instead of only returning them. This can help with debugging:

```
python
import logging
logging.basicConfig(level=logging.INFO)

except Exception as e:
    logging.error(e)
    raise HTTPException(status_code=500, detail=str(e))
```

Brief information on various topics

3. Model Prediction Shape Warning:

If your Keras model outputs a 1D array, `argmax(axis=1)` may throw an error. Ensure:

```
python
y_pred_proba = model.predict(X_new)
y_pred = y_pred_proba.argmax(axis=-1) # safer
```

4. Optional Feature: Confidence Scores

You might consider returning the confidence score (e.g., `max(y_pred_proba)`) along with the prediction.

```
python
return {
    "difficulty": predicted_difficulty,
    "confidence": float(y_pred_proba[0].max())
}
```

5. Validation:

Add input validation for empty or excessively large code snippets:

5. Validation:

Add input validation for empty or excessively large code snippets:

```
python
if not request.code.strip():
    raise HTTPException(status_code=400, detail="Empty code snippet.")
if len(request.code) > 10000:
    raise HTTPException(status_code=413, detail="Code snippet too large.")
```

Code generation

6. CORS Opportunity to do the work instead of user

In production, prefer restricting CORS origins rather than using `*` to avoid security risks.

Would you like help writing unit tests for this API or creating a Dockerfile for deployment?

Appendix 8. Thesis' practical work's response to a solely code input.

Let's break down your code in detail:

- Imports:**
 - `import ast`: This imports the Abstract Syntax Tree (AST) module to parse and analyze Python code.
 - `import sys`: Imports the standard Python sys module.
 - `import pandas as pd`: Imports the pandas library for data manipulation and DataFrame handling.
 - `import uvicorn`: A lightning-fast ASGI server for running your FastAPI application.
 - `from fastapi import FastAPI, HTTPException`: Imports the FastAPI core application class and the HTTP exception class for error handling.
 - `from fastapi.middleware.cors import CORSMiddleware`: Imports the Middleware for handling Cross-Origin Resource Sharing (CORS).
 - `from pydantic import BaseModel`: Imports BaseModel for data validation and serialization.
 - `from keras.models import load_model`: Imports the function to load Keras models for predictions.
- FastAPI app:**
 - `app = FastAPI()`: Initializes a new FastAPI web application instance.
- CORS Setup:**
 - You define a list of allowed origins (`origins`) which are URLs that can access your API (important for frontend development).
 - `app.add_middleware(...)`: Adds CORS middleware, allowing requests from specified origins and enabling credentials, methods, and headers.
- Model Loading:**
 - `model = load_model(load_model_path)`: Loads a pre-trained Keras model which is presumably used for predicting code difficulty.
 - It loads two additional objects using the pickle module:
 - `cv`, presumably a CountVectorizer or similar for feature extraction.
 - `scaler`, likely a scaler used to normalize features.
- CodeRequest Class:**
 - `class CodeRequest(BaseModel)`: Defines a Pydantic model that specifies the structure of requests to your API. In this case, it expects a JSON body containing a string field named `code`.
- Feature Extraction:**
 - The function `extract_features_from_code` takes a code string and returns a list of features based on the following definitions in the code:
 - `loop_count`: Counts the occurrences of loop constructs (`for` and `while`).
 - `assignment_count`: Counts `=` statements.
 - `recursion_check`: Checks if any function calls itself, indicating recursion.
- API Endpoint:**
 - `@app.post("/predict")`: Decorator that defines an endpoint for making POST requests.
 - The function `predict_difficulty(request: CodeRequest)` processes requests:
 - It extracts features from the incoming code using the previously defined function.
 - The features are passed through the transform pipeline (`cv.transform(X)` and `scaler.transform(X)`).
 - It predicts the probabilities of different difficulty classes using the loaded model and selects the class with the highest probability.
 - It maps the predicted class index to a human-readable difficulty level.
- Error Handling:**
 - A try-except block is used to catch exceptions. If an error occurs during the prediction, a 500 HTTP error is raised with the error message.
- Code evaluation**
 - The last block checks if the script is run directly (not imported) and starts the Uvicorn server to serve the FastAPI application on all network interfaces (`0.0.0.0`) and port `8000`.

This code outlines a complete FastAPI application capable of analyzing Python code and predicting its difficulty based on defined features.