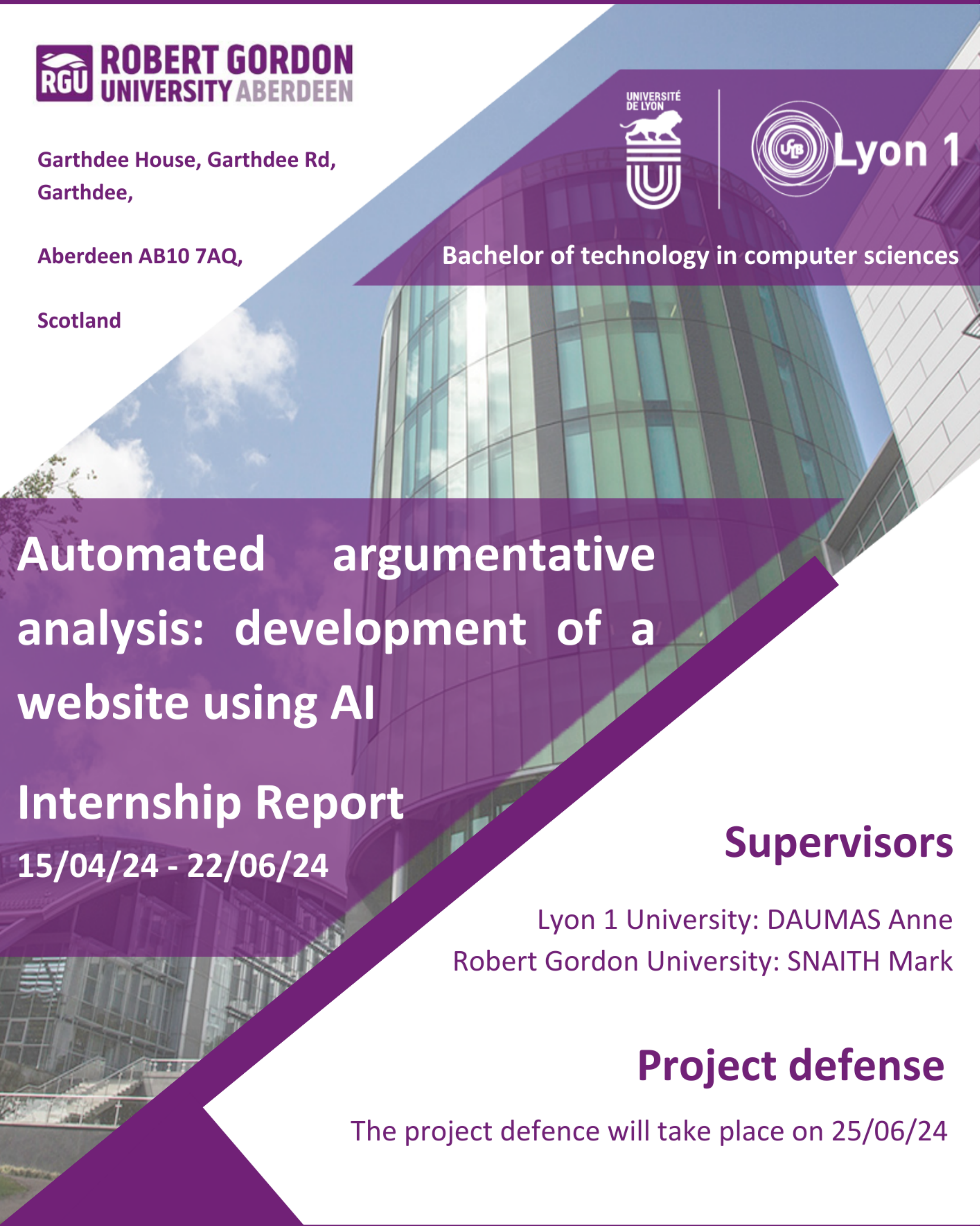
Promotion 2022 - 2025

Léo TRUX



ROBERT GORDON UNIVERSITY ABERDEEN

Garthdee House, Garthdee Rd, Garthdee,

Aberdeen AB10 7AQ,

Scotland

UNIVERSITÉ DE LYON

Lyon 1

Bachelor of technology in computer sciences

Automated argumentative analysis: development of a website using AI

Internship Report

15/04/24 - 22/06/24

Supervisors

Lyon 1 University: DAUMAS Anne
Robert Gordon University: SNAITH Mark

Project defense

The project defence will take place on 25/06/24

Acknowledgements

First of all, I'd like to thank my supervisor, Mr Mark SNAITH, who was available and attentive throughout my placement. His expertise, sound advice and patience contributed greatly to my learning and to the progress of the project. I am deeply grateful for the time he devoted to me.

I would of course like to thank my tutor at Lyon 1 University, Ms Anne DAUMAS, who helped me enormously in organising this work placement abroad, as well as monitoring me throughout. Her constant support and invaluable advice were crucial to the smooth running of this professional experience. I am deeply grateful for her availability and commitment.

I would also like to thank all the teaching staff at IUT Lyon 1 for the knowledge they passed on to me throughout the year, which was invaluable in helping me to complete this project.

Finally, I would also like to express my deep gratitude to the Auvergne-Rhône-Alpes region and the CROUS for awarding me the grant that enabled me to complete this work placement in the best possible conditions. This financial support was essential in covering the costs associated with this experience, enabling me to concentrate fully on my assignments and my learning.

Foreword

On page 64 of this report, you will find a fold-out A3 sheet for easier access to the appendices.

TABLE OF CONTENTS

I. INTRODUCTION

This work placement in Scotland, which began on 15 April and ends on 22 June, represented a very important step in my personal development. In fact, it was my first professional experience in the world of IT development, and it was also taking place in a foreign country, allowing me to get out of my comfort zone and develop essential interpersonal skills. From an academic point of view, this placement is a key stage in the validation of my BUT 2 at the University of Lyon 1. It represented an opportunity to apply the theoretical knowledge acquired during my studies in a practical context. This link between theory and practice gave me a better understanding of the concepts I had studied.

From a professional point of view, this work placement in Scotland was a decisive experience. It gave me my first significant immersion in the world of work, helping me to understand the expectations and requirements of the professional sector. This practical experience is a major asset to my CV, demonstrating my ability to work in an international environment and adapt to a variety of cultural contexts.

I was lucky enough to find this placement thanks to Lyon 1 University, which offered all BUT 2 students the opportunity to do their placement abroad in several countries, including Scotland. This offer was part of a partnership with various universities and organisations. I chose to do my placement abroad for several reasons, in particular to discover a new country but also to improve my English, a major skill that is very important in all professional fields. The placement took place at Robert Gordon University in Aberdeen, and we'll come back to that in more detail in the next section.

My assignment on this placement was to work on argumentative analysis in general. Carrying out an argumentative analysis of a text can be long and rigorous, so I had to develop a website to automate the whole process of carrying out an argumentative analysis. In this report, you will find a presentation of the company and the work placement environment. Next, you will find the context of the assignment, followed by the entire analysis, design and final implementation phase of my project. Finally, you will find a conclusion with an overview of the field.

II. THE UNIVERSITY

II.1 PRESENTATION OF THE UNIVERSITY

I was lucky enough to do my placement at Robert Gordon University (RGU), a UK university located at Garthdee House, Garthdee Road, Aberdeen (AB10 7QB), Scotland, UK. Aberdeen is a city renowned for its oil and gas industry, as well as its commitment to innovation and academic research.

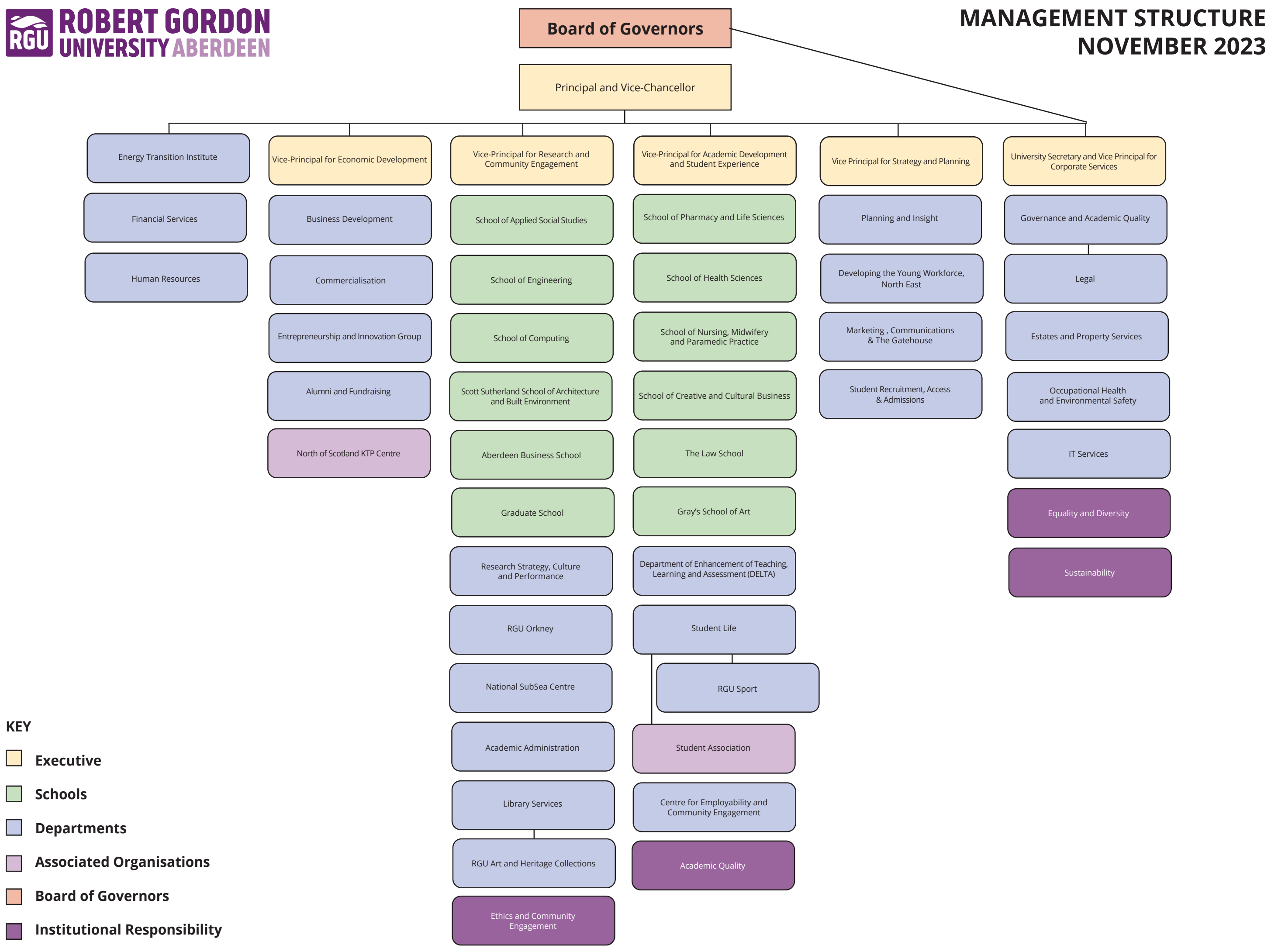
Robert Gordon University was founded in 1750 as an educational institution and became a university in 1992. Over the years, RGU has evolved into a modern institution, focused on vocational education and applied research, with strong links to industry and employment-oriented study programmes.

Robert Gordon University (RGU) has an endowment of £2.48 million in 2022 and a budget of £114.7 million for 2021-2022. It has an academic staff of 662, an administrative staff of 933 and a total of 16,787 students, positioning itself as a leading university for graduate employability, offering practical, industry-focused programmes. The university's premises are modern and recent (built 9 years ago) and well-equipped, contributing to an optimal learning environment. Ranked among the top ten universities in Scotland, RGU stands out in the market by providing a high-quality education that effectively prepares students for the professional world.

RGU offers a variety of services, including higher education, applied research, consultancy and professional development. The university produces and distributes research publications, continuing education and professional development programmes. By investing in key areas such as technology, health, engineering and renewable energy, RGU has state-of-the-art facilities to support teaching and research.

RGU has close relationships with local and international companies such as cBP, Shell, TotalEnergies, Wood, NHS Scotland and Accenture. These collaborations focus on research projects and continuing education services. The main markets for RGU's products and services include the healthcare, engineering, business and creative technology sectors. RGU distinguishes itself from its competitors through its professional orientation and close links with industry. The university relies on innovation, collaboration with industry and the expansion of its international programmes for its strategic development. Adhering to strict academic quality standards, RGU is regularly assessed by external bodies to ensure the excellence of its teaching and research. To attract national and international students, the university uses a variety of marketing strategies, including online campaigns, events and partnerships with schools and companies.

RGU is structured to optimise governance, academic quality, innovation and student services. Here is an overview of the university's organisation chart.



ROBERT GORDON UNIVERSITY ABERDEEN
MANAGEMENT STRUCTURE NOVEMBER 2023
Board of Governors
Principal and Vice-Chancellor
Energy Transition Institute
Financial Services
Human Resources
Vice-Principal for Economic Development
Business Development
Commercialisation
Entrepreneurship and Innovation Group
Alumni and Fundraising
North of Scotland KTP Centre
Vice-Principal for Research and Community Engagement
School of Applied Social Studies
School of Engineering
School of Computing
Scott Sutherland School of Architecture and Built Environment
Aberdeen Business School
Graduate School
Research Strategy, Culture and Performance
RGU Orkney
National SubSea Centre
Academic Administration
Library Services
RGU Art and Heritage Collections
Ethics and Community Engagement
Vice-Principal for Academic Development and Student Experience
School of Pharmacy and Life Sciences
School of Health Sciences
School of Nursing, Midwifery and Paramedic Practice
School of Creative and Cultural Business
The Law School
Gray's School of Art
Department of Enhancement of Teaching, Learning and Assessment (DELTA)
Student Life
RGU Sport
Student Association
Centre for Employability and Community Engagement
Academic Quality
Vice Principal for Strategy and Planning
Planning and Insight
Developing the Young Workforce, North East
Marketing , Communications & The Gatehouse
Student Recruitment, Access & Admissions
University Secretary and Vice Principal for Corporate Services
Governance and Academic Quality
Legal
Estates and Property Services
Occupational Health and Environmental Safety
IT Services
Equality and Diversity
Sustainability
KEY
Executive
Schools
Departments
Associated Organisations
Board of Governors
Institutional Responsibility

II.2 Internship environment

As you can see from the organisation chart on the previous page, the university has a large number of departments and schools. I did my placement in the School of Computing. This is an academic entity dedicated to teaching and research in the field of computing. It offers a range of undergraduate and postgraduate programmes, including bachelor's and master's degrees.

During my placement, I had the opportunity to work in a laboratory reserved for international students. There were around 8 international students sharing the room. The university provided me with a personal Outlook account, which gave me access to all the tools I needed to complete my project. I had a desktop computer with two screens at my disposal, but I preferred to use my own machine to carry out the project.

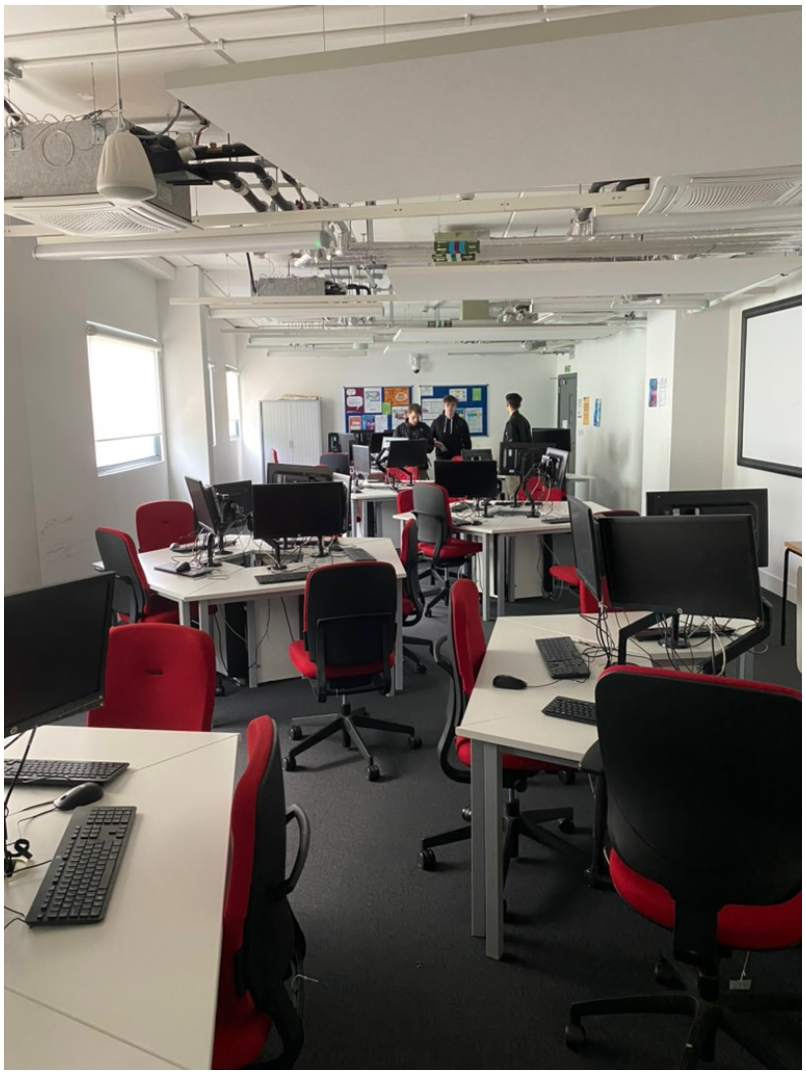


Figure 1 – Project room

III. PROJECT CONTEXT

III.1 PRESENTATION OF THE INTERNSHIP THEME

The theme of my placement at Robert Gordon University (RGU) was the development of a website using artificial intelligence for automated argumentative analysis of user-supplied texts. The aim of this assignment was to create a web application capable of segmenting, classifying and analysing the arguments present in a text and the relationships between them, by identifying premises, conclusions and logical relationships.

III.2 THE DIFFERENT TASKS TO BE CARRIED OUT

To carry out this mission successfully, I had to break down the main mission into several main tasks. Here they are:

1. **Website design and development:** Website development was one of the most important tasks. This included thinking about the organisation/style of the user interface, as well as developing the frontend.

2. **Implementing the Flask API & the database:** To enable the website to communicate with the backend, i.e. the server that processes the data and executes the application logic, it was essential to develop reliable and secure connection points. This was what the Flask API contains. Next I had to implement the database to store all the data I need.

3. **Dataset creation:** The aim of the assignment was to automate the process of argumentative text analysis using AI. In order to create high-performance AI models, I needed to create a dataset containing example data in order to train the various models.

4. **Models' creation:** To make the various processes involved in argumentative analysis automatic, it was essential to create and used AI models that will perform the redundant analysis tasks for the user.

III.3 The mission

The assignment given by the university was based on the need to develop an innovative tool capable of carrying out automated argumentative analysis of texts in the context of research. The process of argumentative text analysis is often redundant, and some parts of the process can be tedious. With the increasing complexity of information and the high volume of textual data, it is becoming essential to have advanced solutions to help users better understand and analyse the arguments presented. The main technical objective was to design a high-performance, user-friendly website. The site had to offer a modern, intuitive and easy-to-use interface, capable of effectively managing user queries. The integration of artificial intelligence models was crucial to automate text analysis, guaranteeing accurate and reliable results without human intervention. Development prospects included the ongoing evolution of the application with the addition of new functionalities, the improvement of AI algorithms and the optimisation of performance.

III.4 Constraints

Throughout the development of the project, there were a number of constraints that could jeopardize its development

- **Limited time:** The project had to be completed in 10 weeks, requiring rigorous management of time and priorities to meet the deadline. Each phase of the project had to be carefully planned and executed to ensure timely completion.

- **Human resources:** Given that the project was being developed in a university context, human resources were limited to one person, i.e. me. This constraint could limit the ability to work several tasks simultaneously and required efficient work management.

- **Data access:** Although I had access to analytical data through my tutor and the Monkey Puzzle database, acquiring and annotating the data could take a lot of time and effort, especially when creating the dataset. I had to ensure that the data was of high quality and well annotated to effectively train the artificial intelligence models.

- **Technical complexity:** Developing an artificial intelligence model for argumentative analysis was technically complex. It could require several iterations and adjustments to achieve the desired performance. Moreover, I had no experience in this field, which made the task even more complex. As a result, I had to do a lot of research and testing, which risked delaying development.

- **Component integration:** Ensuring seamless integration between the frontend, backend and AI could pose technical challenges. I had to ensure that I was continually testing and adjusting these integrations to guarantee optimal performance and a consistent, fluid user experience.

III.5 Personal field of investigation & human contact

I developed this project entirely on my own, without sharing the task with another developer. However, I had my internship tutor to support me, Mr Mark SNAITH. He was able to advise me throughout the project and provided me with valuable guidance, constructive feedback and technical advice, which helped me to overcome the challenges I encountered and ensure the quality of the final product. At the start of the project, he gave me lots of resources to help me understand the subject and argumentative analysis, including a lecture on the subject from him, answers to exercises and a tutorial to follow on MonkeyPuzzle. We had weekly meetings where I showed him the progress of the project and he gave me additional advice in return.

None of the work had been done before I started; I took charge of the project from its initial conception to its complete development. Everything was done from scratch, from planning to implementation, including feature development and final testing.

IV. PROJECT ANALYSIS

IV.1 Project aims and concept

The aim of this project was to develop an innovative website that offered automated argumentative analysis of user-provided texts. The website was designed to use artificial intelligence techniques to analyse the content of the texts and identify the main arguments, as well as their supporting and opposing claims.

The project's concept was based on the idea that argumentative analysis is a complex task requiring a deep understanding of language, context, and the logical relationships between different statements. By using AI algorithms, the website aimed to provide accurate and reliable analysis without the need for human intervention.

Overall, the project aimed to provide a valuable tool for anyone interested in understanding and analysing arguments in a more systematic and rigorous way, and to contribute to the advancement of AI research in the field of natural language processing.

IV.2 Key terminology

IV.2.1 Argumentative analysis

An augmentative analysis is a process who allows people to have a better understanding of the arguments in texts. Indeed, the text they give will be separated into premises and conclusions to help better understand the arguments being made, which parts of the statement support others, and what conflict might exist.

A **premise** is a span of text that isn't supported by anything else and is used to provide support to conclusions.

A **conclusion** is something supported by premises and, potentially, **sub-conclusions** – and a **sub-conclusion** is something supported by premises and, potentially, other sub-conclusions as a stepping-stone towards the overall conclusion.

For example, let’s take the following text:

The RGU School of Computing is a great place to study. The staff are friendly, the labs are state-of-the-art, and the subjects are engaging. So, you will have an amazing time during your degree.

To do an argumentative analysis, we have to follow 4 steps: segmentation, classification, references and conflicts.

IV.2.1.1 Segmentation

The segmentation process involves dividing the text into smaller, meaningful units such as single sentences or parts of sentences before and after conjunctions, with the general rule being to identify the shortest spans of text that still make sense. After the segmentation part, our text will look like this:

*(**The RGU School of Computing is a great place to study**). (**The staff are Friendly**), (**the labs are state-of-the-art**), and (**the subjects are engaging**). So (**you will have an amazing time during your degree**).*

Here, the texts in bold between parentheses are the results of segmentation process. Now we can move forward to the next part.

IV.2.1.2 Classification

Classification is the 2nd step of our argumentative analysis. The aim of this step is to understand what role each individual part of the text and/or each individual segment carries in the argument what role it plays in the argument. If we take our text segmented, after the classification part, it looks like this:

(The RGU School of Computing is a great place to study). (The staff are friendly), (the labs are state-of-the-art), and (the subjects are engaging). So (you will have an amazing time during your degree).

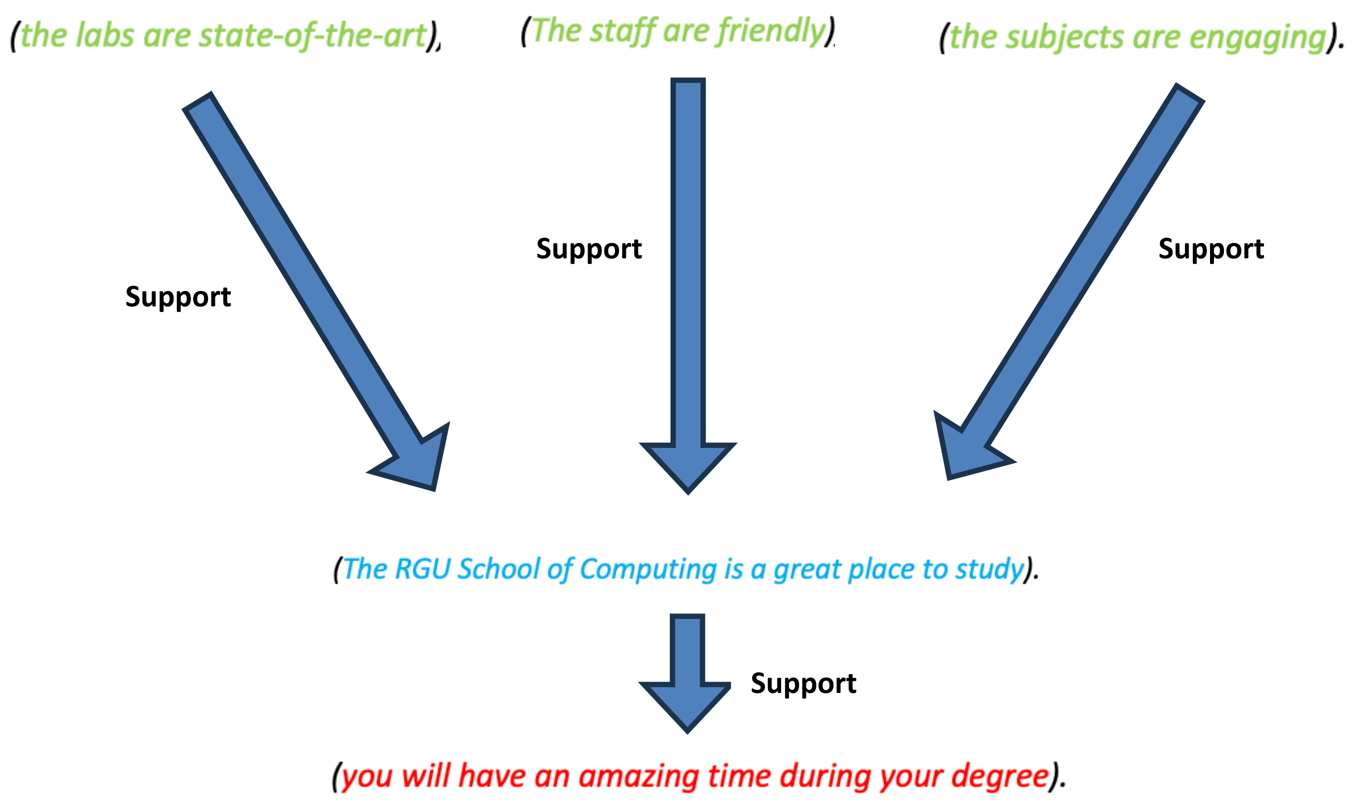
🟦 Sub-conclusion

🟩 Premise

🟥 Conclusion

IV.2.1.3 References

The inference identification step involves analysing the relationships between the different components of the text, such as premises and conclusions, to understand how they support each other. This involves identifying which premises support sub-conclusions, whether any premises directly support the overall conclusion, and whether premises work independently or together to support a (sub-)conclusion. Clues in the text, such as words like "because", "so", "therefore", and "and", can help in this process. The goal is to create a clear map of the argument and understand how the different parts work together to support the main conclusion.


(the labs are state-of-the-art),
(The staff are friendly).
(the subjects are engaging).
Support
Support
Support
(The RGU School of Computing is a great place to study).
Support
(you will have an amazing time during your degree).

IV.2.1.4 Conflicts

The last part of argumentative analysis is to identify conflicts in the text. The aim is to identify any opposing arguments or conflicting viewpoints presented in the text. Counterarguments are analysed in the same way as the main argument, by breaking them down into premises and conclusions. Once the counterarguments are identified, the type of conflict is determined, such as undermining or rebutting, based on the text. This step helps to provide a comprehensive understanding of the argument and the different perspectives presented. In our example, there is no conflicts, but if we take this text for example:

Although the RGU School of Computing has friendly staff and state-of-the-art labs, some students have found the coursework to be too challenging, leading to high levels of stress and anxiety. Therefore, it is not guaranteed that every student will have an amazing time during their degree.

In this text, there is a conflict between the first part of the sentence, which presents positive aspects of the RGU School of Computing (friendly staff and state-of-the-art labs), and the second part, which introduces a counterargument (some students have found the coursework to be too challenging, leading to high levels of stress and anxiety). This conflict can be classified as a rebutting conflict, as it directly challenges the conclusion of the first part of the sentence (that every student will have an amazing time during their degree). By identifying this conflict, we can gain a more nuanced understanding of the argument being presented and evaluate the strengths and weaknesses of both sides.

IV.2.2 API

An API (Application Programming Interface) is a programming interface that allows different software applications to communicate with each other. In our case, the Python API will be used to provide an interface between our web application, the argumentative analysis model and the database. The API will allow our web application to send requests to the argumentative analysis model and receive responses in the form of structured data. In addition, the API will also be responsible for communicating with the database to store and retrieve the data required for argumentative analysis. In short, the API is a key element of our software architecture that will allow our web application to communicate effectively with the argumentative analysis model.

IV.2.3 Dataset

A dataset is a collection of organised and structured data used to train and test machine learning models. The aim of a dataset is to provide high-quality, representative data for training an AI model. The larger and more diverse the dataset, the more the model will be able to generalise and perform on new examples of data.

IV.3 STATUS OF THE PROJECT

When I started this project, no prior work had been done. I took charge of the project from its initial conception through to its complete development. Everything was done from scratch, from planning to implementation, including feature development and final testing. To understand what an argumentative analysis is, all I had at my disposal were documents and courses provided by Mr Mark Snaith. These resources enabled me to familiarise myself with the concepts and techniques involved in argumentative analysis.

IV.4 Competitive analysis

IV.4.1 MonkeyPuzzle

https://open-argumentation.github.io/MonkeyPuzzle/index.html

MonkeyPuzzle is a free, open-source online argumentative analysis tool. It allows users to visualise and analyse complex arguments using interactive diagrams. Users can import text files or write directly into the MonkeyPuzzle interface to create argument diagrams. Arguments can be organised into premises, conclusions and sub-arguments, and users can add labels and comments to clarify their reasoning. MonkeyPuzzle also supports the import and export of Argument Interchange Format (AIF) files, allowing users to share and collaborate on argument analyses. All in all, MonkeyPuzzle is a useful tool for students, teachers, researchers and professionals who want to analyse and understand complex arguments.

Benefits	**Inconvenient**
Free	Interface not ergonomic and difficult to use
Open source	All argumentative analysis must be done by hand
Importing and exporting files	Lack of support and documentation
Personalisation of diagrams	

The project will have a more modern and easier-to-use interface than MonkeyPuzzle, enabling users to be more efficient in their tasks. Moreover, the argumentative analysis that give MonkeyPuzzle is not automized at all, that means that users have to realise all the argumentative analysis alone and after creating their diagrams. In our website, the argumentative analysis will be totally automized thanks to our artificial intelligence models, so our users will save time and be more efficient.

IV.4.2 ChatGPT

https://chat.openai.com/

ChatGPT is a language model developed by OpenAI, an artificial intelligence research company. It is a natural language processing model based on the GPT (Generative Pre-trained Transformer) architecture that has been trained on a large corpus of textual data. ChatGPT can generate coherent, relevant responses to questions posed in natural language, as well as holding fluid, natural conversations with users.

Benefits	**Inconvenient**
Free	Lack of support and documentation
Interface ergonomic and easy to use	Cannot exporting your argumentative analysis' results
Argumentative analysis is done by AI	May generated incorrect results
One of the most powerful ***NLP***[1] in the world	Cannot generate diagrams of an argumentative analysis

In comparison with our project, ChatGPT is a more general tool that can be used for a variety of natural language processing tasks, whereas our project focuses specifically on argumentative analysis. Indeed, this last one cannot generate diagrams for a better understanding of an argumentative analysis.

IV.5 Dataset structure & AI models

IV.5.1 The dataset

To carry out this project successfully, it was essential to have a dataset that could be used to train the various AI models. The first step was to choose a data structure that made sense and was consistent with the objective of our AI. For this, I chose a JSON format for several reasons: it was modular, had a clear structure, and was widely used for data interchange.

I was fortunate to have access to numerous examples of argumentative analyses extracted from the MonkeyPuzzle database. I drew inspiration from this structure, known as SADface, to better manipulate the data. For example, if we took the following text:

"Fall is the best time to visit America's great cities, beaches, and mountains. The foliage is breath-taking, the weather is cooler, and the crowds are gone. So, you can really relax and enjoy yourself."

Dataset structure of this example is available on page 65 in the appendix

IV.5.2 AI models

Creating a single AI model to perform the entire argumentative analysis was a complex task. To make it easier, the task was divided into several models, each responsible for a specific aspect of the analysis. This approach simplified the problem and allowed for more manageable and efficient training and tuning of the models.

Each model focused on a particular task:

1. **Segmentation Model:** This model split the text into meaningful segments.
2. **Classification Model:** This model classified each segment as a premise, conclusion, or other.
3. **Relation Model:** This model identified and labelled the relationships between segments, such as support or conflict.

By breaking down the task into these specialized models, I improved the overall performance and accuracy of the argumentative analysis.

IV.6 TECHNOLOGY USED

During my internship, I had the chance to have no constraints in terms of technology choices.

IV.6.1 Presentation

React is an open-source JavaScript library used to create interactive user interfaces. It is used to create reusable components for building web and mobile applications. React is maintained by Facebook and an active community of developers.

Flask is a Python web micro-framework used to create RESTful APIs. It is lightweight, easy to use and offers great flexibility for building web applications. Flask is used to build small- to large-scale web applications.

GitHub is a collaborative development platform based on Git. It is used to store, share and collaborate on software development projects. GitHub offers features such as version control, project management, code review and continuous integration. GitHub is widely used by developers around the world to host their open source and private projects.

Firebase is a mobile and web application development platform developed by Google. It offers a wide range of services and tools to help developers build high-quality apps quickly and efficiently. The platform includes features such as Realtime Database, Authentication, Cloud Firestore, Hosting, Cloud Functions, Analytics, and Cloud Storage. These features enable developers to store and sync data in real-time, authenticate users easily, run backend code in response to events, host web apps securely, analyse app usage, track crashes, and store user-generated content in the cloud.

WebStorm is an integrated development environment (IDE) for JavaScript, developed by JetBrains. It offers advanced features for web development, including an intelligent code editor, debugging tools and integration with version control systems such as Git. WebStorm is particularly popular for its full support for React, its ability to manage complex projects and its user-friendly interface, making it a popular choice among web developers.

IV.6.2 Advantage & inconvenient

Technology	Advantage	Inconvenient
GitHub	- Easy, effective version control - Secure, backed-up code storage. - Easy continuous integration	- Can be difficult for beginners to learn. - Large projects can be difficult to manage
React	- Popular JavaScript library for creating responsive and dynamic user interfaces. - Offers great flexibility in user interface design.	- Steep learning curve for JavaScript beginners - Requires complex initial configuration for certain projects
Flask	- Easy to use and configure. - Lightweight and flexible - Extensions available to add functionality	- Less suitable for large-scale applications - Requires in-depth knowledge of Python for optimal use. - Few built-in features by default
Firebase	- Easy to install and use. - Simplified authentication - Extensive documentation and community support	- Limited control over server-side logic - Pricing can become expensive as the project scales - Limited flexibility in data modelling
WS	- Excellent built-in support for React - Integrates version control tools such as Git - Offers powerful tools for code refactoring and code review	- Paid software - Can be greedy in system resources

IV.6.3 Justification of technology choices

React: The need for a highly interactive and dynamic user interface made React the best choice for me. Its component-based architecture simplified the development and maintenance of the user interface. Additionally, React simplified the code structure for using JavaScript, which was particularly useful in this project for creating responsive and maintainable user interfaces. I also chose this technology because I wanted to learn something new, and increase my level in JavaScript too.

Flask: Given the project's requirement for a lightweight and flexible backend to handle API requests, Flask was ideal due to its simplicity and powerful extension capabilities. Moreover, Flask had enormous documentation, as well as a huge library, making backend development much easier.

GitHub: The robust version control and collaboration features of GitHub ensured that code management. I also mastered this technology, which was the main reason I chose it.

Firebase: Firebase's real-time database and simplified authentication mechanisms were crucial for this project, allowing for seamless user data management and quick development cycles. Moreover, this technology is owned by Google, which was perfect because it simplified the authentication with Google and had a huge amount of documentation.

WebStorm: WebStorm was the preferred IDE for this project due to its excellent built-in support for React. Although WebStorm is paid software, I benefited from a free student license, making it an accessible and invaluable tool for my development needs. Additionally, while WebStorm can be demanding in terms of system resources, my current setup was sufficient to handle its demands, allowing me to fully leverage its capabilities without performance issues.

IV.6.4 Libraries used

IV.6.4.1 Frontend

Firebase is a comprehensive app development platform provided by Google. It offers various services such as authentication, real-time database, cloud storage, and more.

D3.js (Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers using SVG, HTML, and CSS.

Axios is a promise-based HTTP client for the browser and Node.js. It makes it easy to send asynchronous HTTP requests to REST endpoints and perform CRUD operations.

Redux: Redux is a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments, and are easy to test.

IV.6.4.2 Backend

Flask is a lightweight WSGI web application framework in Python. It is designed with simplicity and flexibility in mind.

OpenAI provides powerful APIs for natural language processing (NLP) and other AI tasks.

The Natural Language Toolkit (NLTK) is a suite of libraries and programs for symbolic and statistical natural language processing for English.

SpaCy is an open-source software library for advanced natural language processing, written in Python and Cython.

Scikit-learn (Sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms.

Joblib is a set of tools to provide lightweight pipelining in Python. It is specifically designed to serialize and deserialize Python objects.

The Transformers library by Hugging Face provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio.

PyTorch is an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing.

IV.7 Specifications

This analysis enabled me to create the following use case diagram.

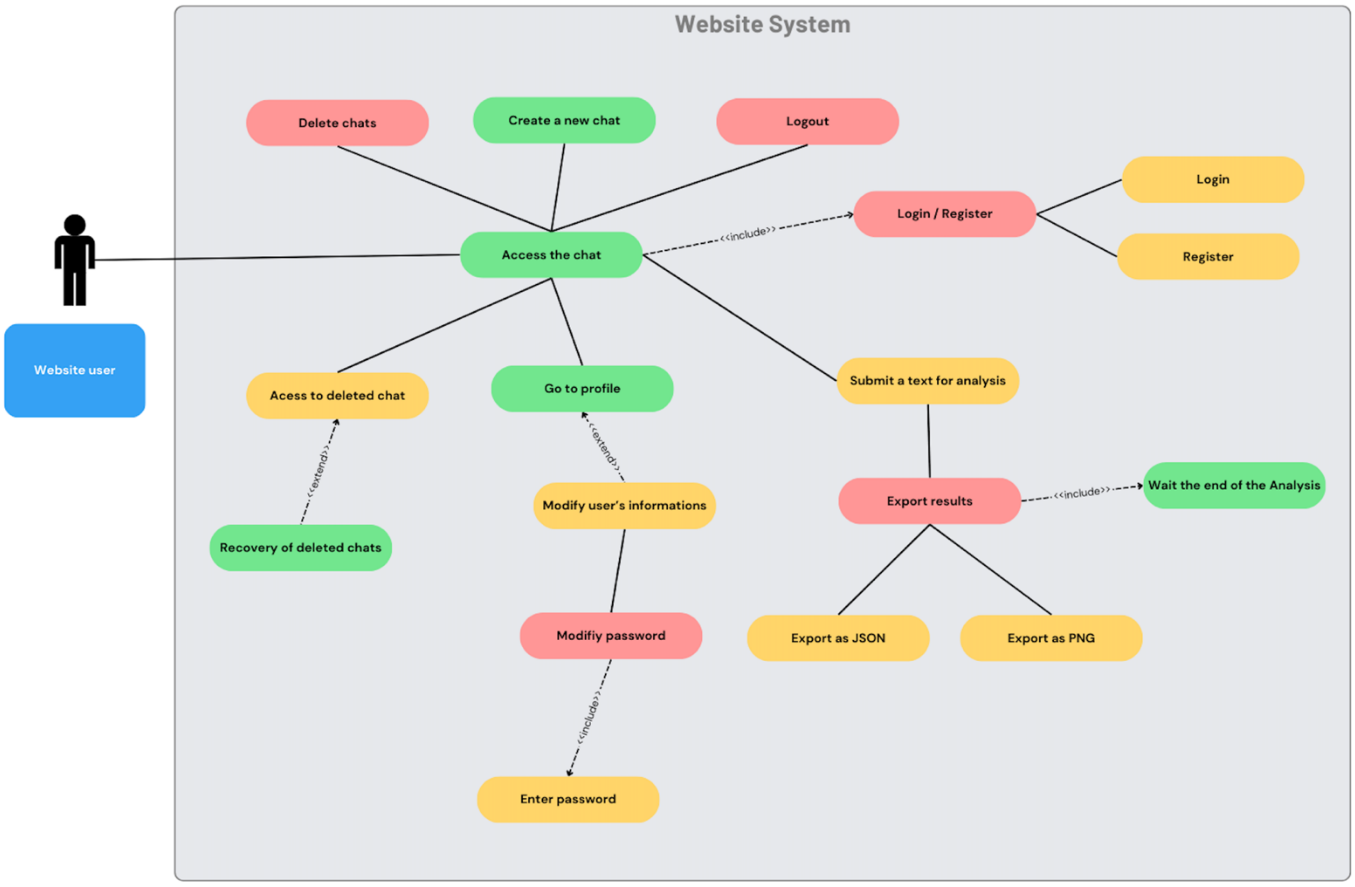


Figure 2 - Use case diagram

On the website, users can perform various actions. They can log in or register to create a new account, which gives them access to the full range of features. Once logged in, users can access the chat system where they can create new chat sessions, delete existing sessions or log out when they have finished. The site also allows users to submit texts for argumentative analysis, with a waiting period required for the analysis to be completed. Once the analysis is complete, users can export the results in JSON or PNG format. Finally, users can edit their personal details.

V. CONCEPTION

V.1 PROJECT ORGANISATION

The application was divided into two main parts: the frontend, developed with React, and the backend, developed with Flask (python). The frontend communicates with the backend via the Flask API. The backend manages the business logic, interactions with the database, and the creation and integration of AI models.

V.1.1 Frontend architecture

The front-end architecture was based on React and was organised in a modular fashion. Each component had a specific responsibility and contributed to the application as a whole by offering distinct functionalities.

1. **App:** Root component of the application. It encapsulates all the other components and manages the overall structure.
   Sub-components: Auth, ChatContainer, Modal, Home, Firebase, SideBar, Form.

2. **Auth:** Manages user authentication, including login and registration.
   Sub-components: Login, register.

3. **ChatContainer:** Manages the chat page where the results of the argumentative analysis will be presented.
   Sub-components: Chat (Main component of the chat), Graph (Chat-related graphical display), Message (Management of individual messages), Tool (Additional tools for browsing and exporting results)

4. **Modal:** Manages all site modals.

5. **Home:** Component of the home page displaying the main information and navigation options.
   Subs-components: NewChat (manage the creation of a new chat), Tool (Additional tools for browsing and exporting results)

6. **Firebase:** Manages integration with Firebase for features such as authentication, real-time storage, etc.

7. **Sidebar:** Sidebar for navigation, viewing chat history and site settings
   Subs-components: History (Viewing conversation history)

8. **Form:** Manages reusable form elements such as buttons and input fields.
   Subs-components: Button (reusable button), Input (reusable input)

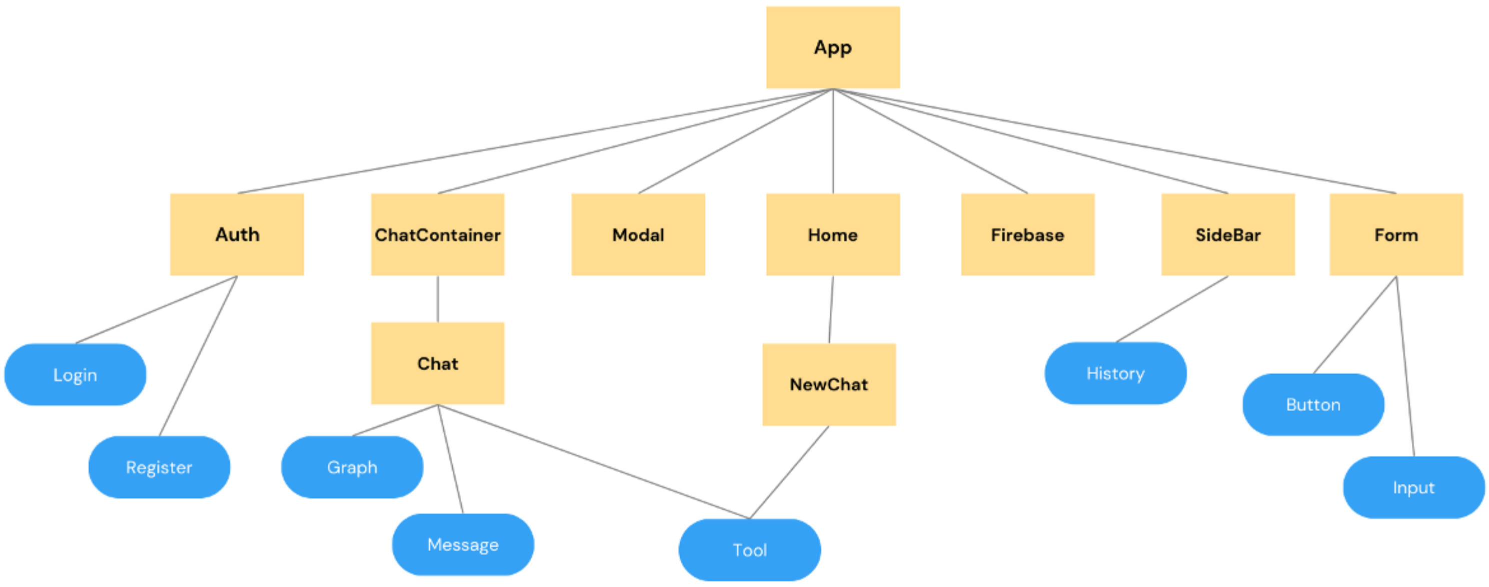


Figure 3 - React component diagram

V.1.2 Backend architecture

The backend architecture was organised into a number of modules that manage different aspects of the application, including APIs, interactions with Firebase, dataset creation, training of AI models and their use in argumentative analysis.

1. **API:** Provides the endpoints for the application's various functions, enabling communication between the frontend and backend.
   Subs-components:
   - Analyse: Contains routes and logic for analysis functions.
   - Chats: Manages routes and logic for chat functions.
   - Users: Manages routes and logic for user functionality, such as registration and authentication.

2. **Firebase:** Contains scripts for integration with Firebase, including authentication and real-time data management.

3. **Dataset:** Management and creation of datasets used for training and classifying AI models.

4. **Train:** Contains scripts and models for training and test AI algorithms.
   Subs-components:
   - Relations: Manages the training and testing of the relationship model
   - Classifications: Manages the training and testing of the classification model
   - Data: Contains the dataset used to train models

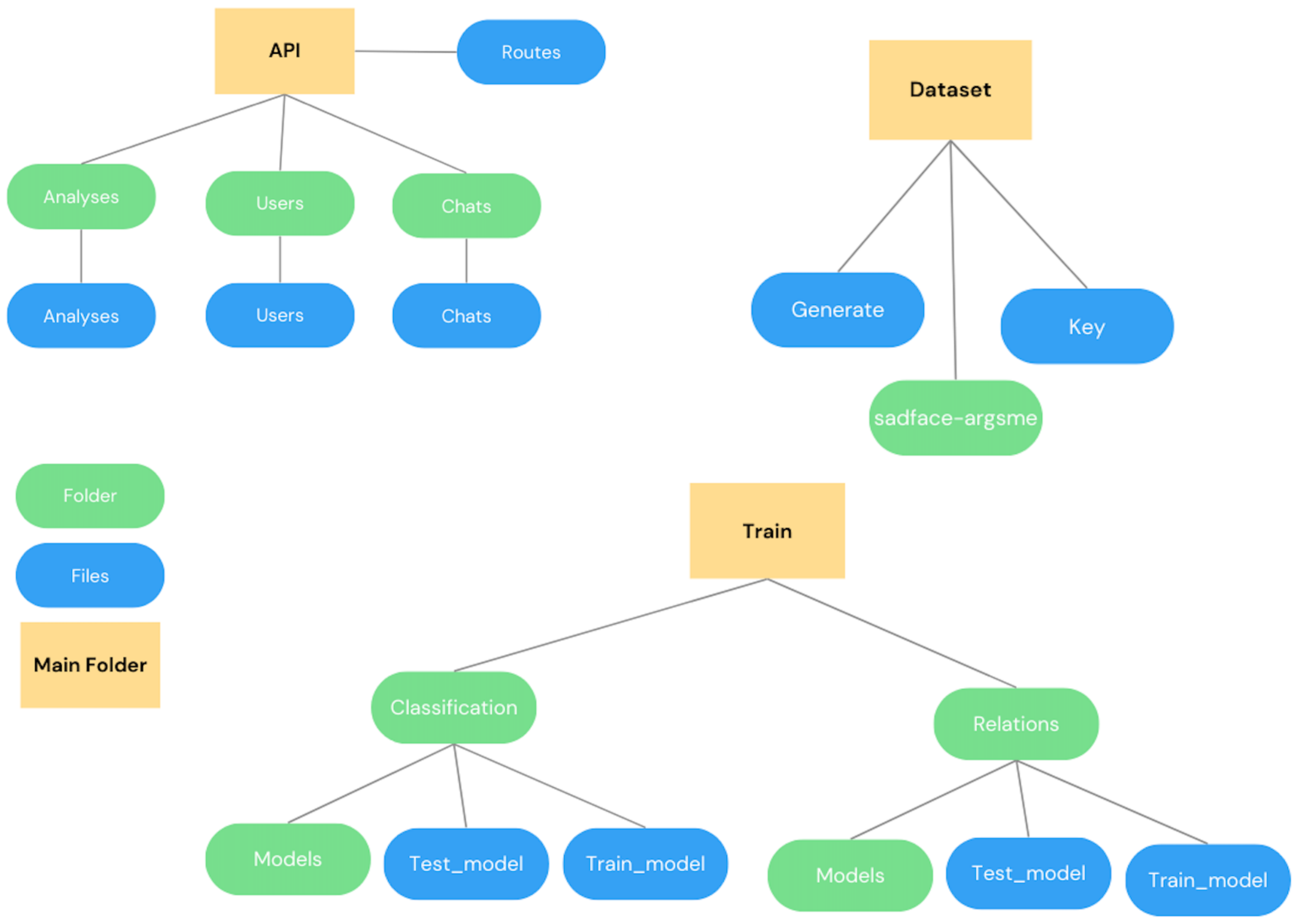


Figure 4 - Backend modules diagram

V.2 Database organisation

During my internship, I also had to design a data model to store all the information for my website. The primary data I needed to store included two main entities: the users and the analyses.

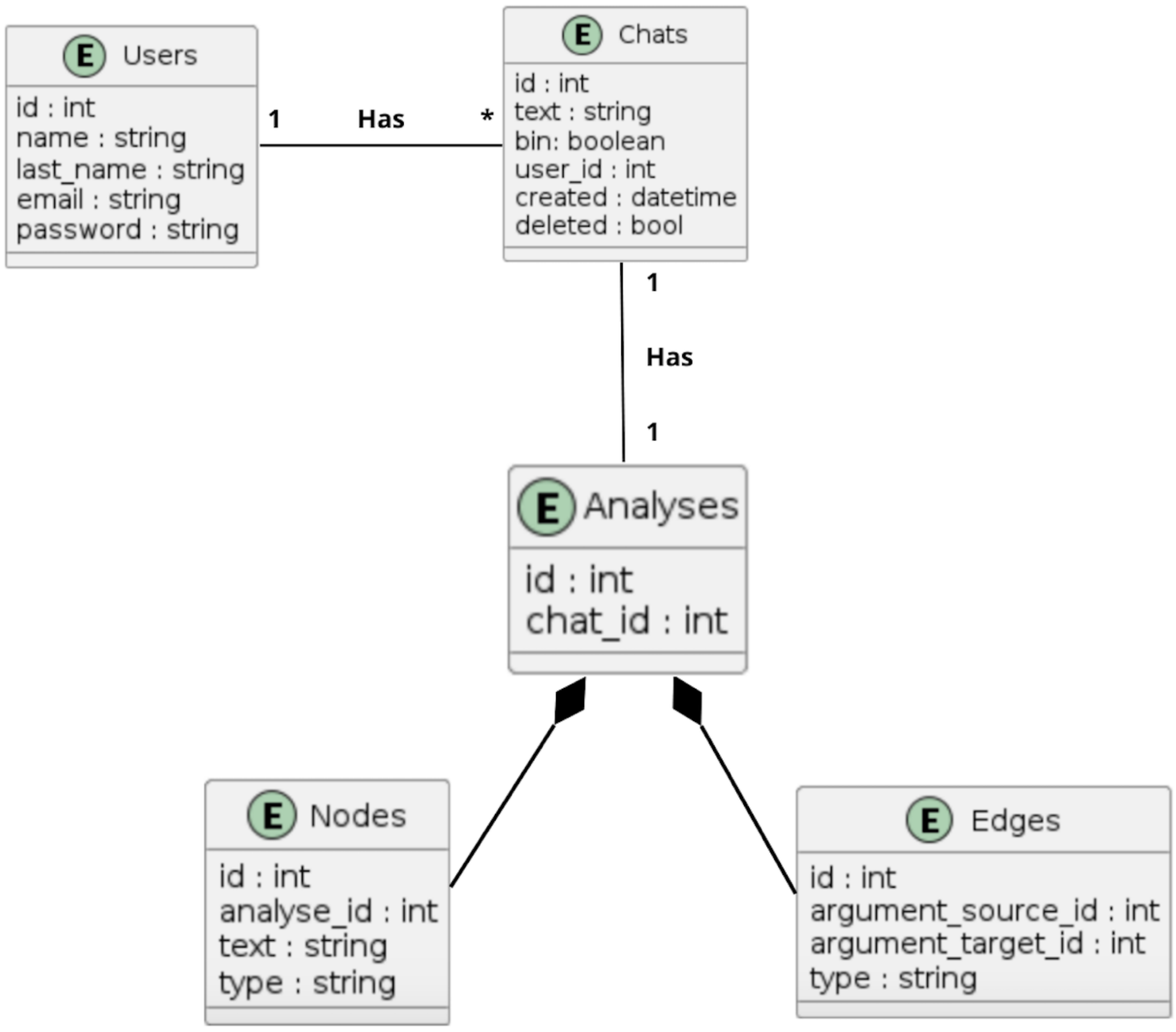


Figure 5 - Class diagram

It should be noted that this scheme has evolved throughout the development process. This is the final result. The analysis is divided into 3 tables to make it more modular and because of its JSON structure, as we saw on the dataset section.

V.3 Main interaction

For a better understanding of the interaction involved in carrying out an argumentative analysis, take a look at this simplified sequence diagram below

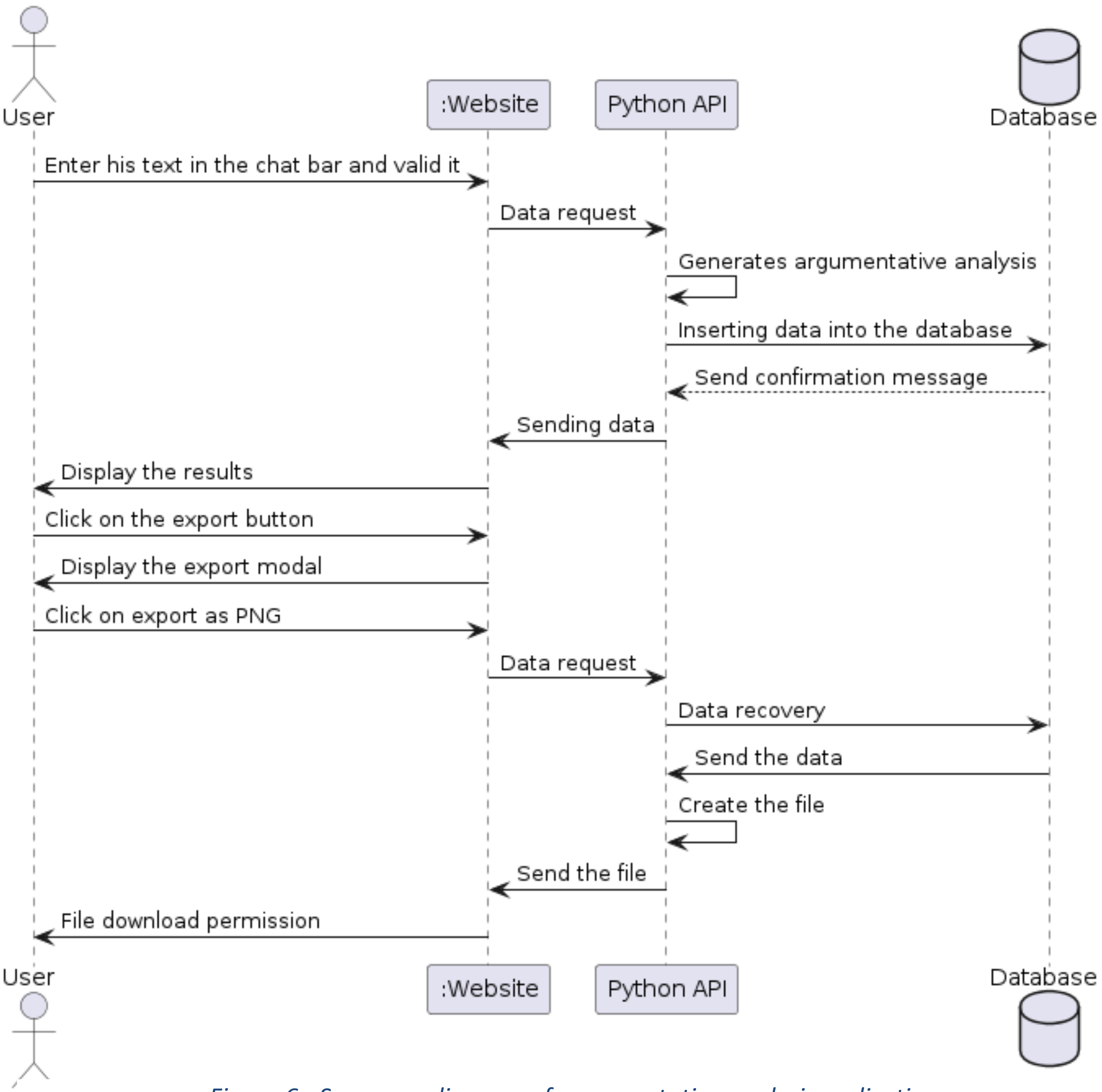
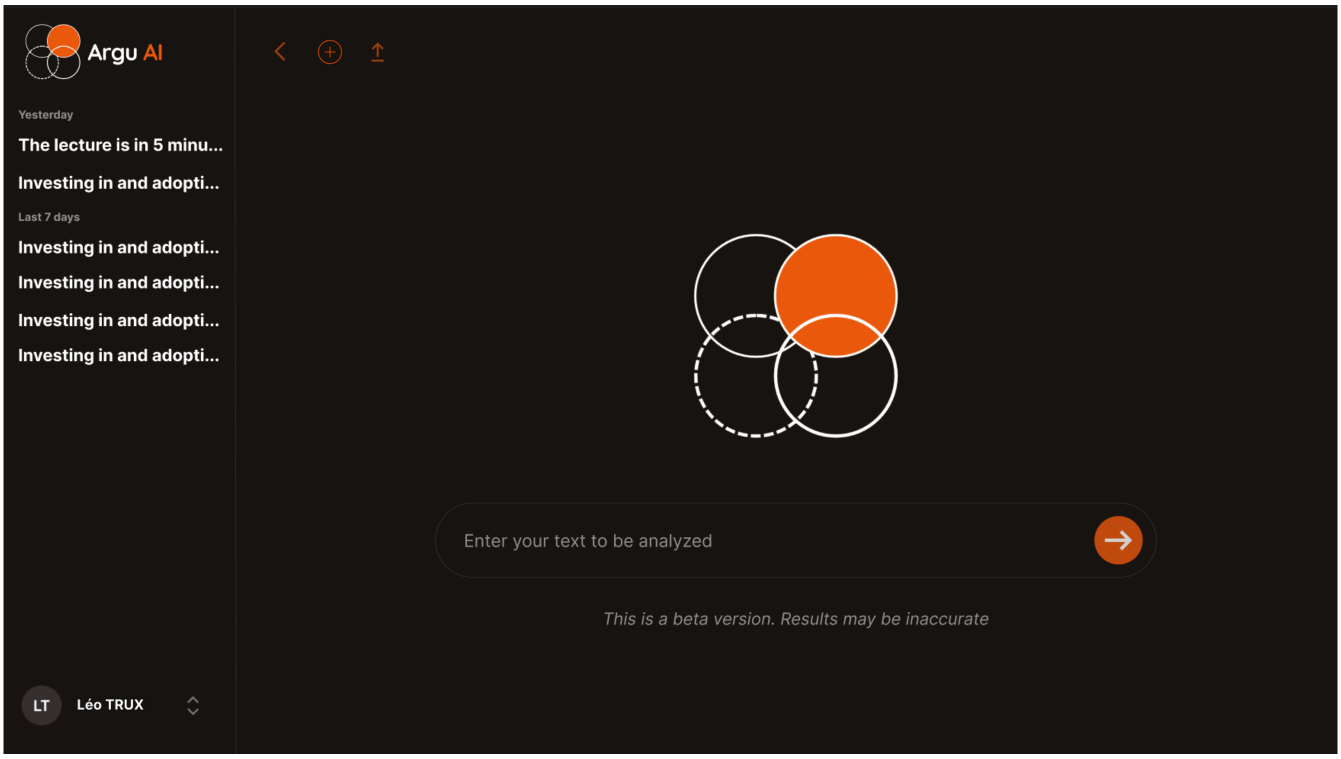


Figure 6 - Sequence diagram of argumentative analysis realisation

This sequence diagram illustrates the main interactions between the user, the website, the Python API and the database to perform an argumentative analysis. It highlights the key steps required to process, display and export the results of the analysis, providing a clear overview of how the system works.

V.4 Mock-up

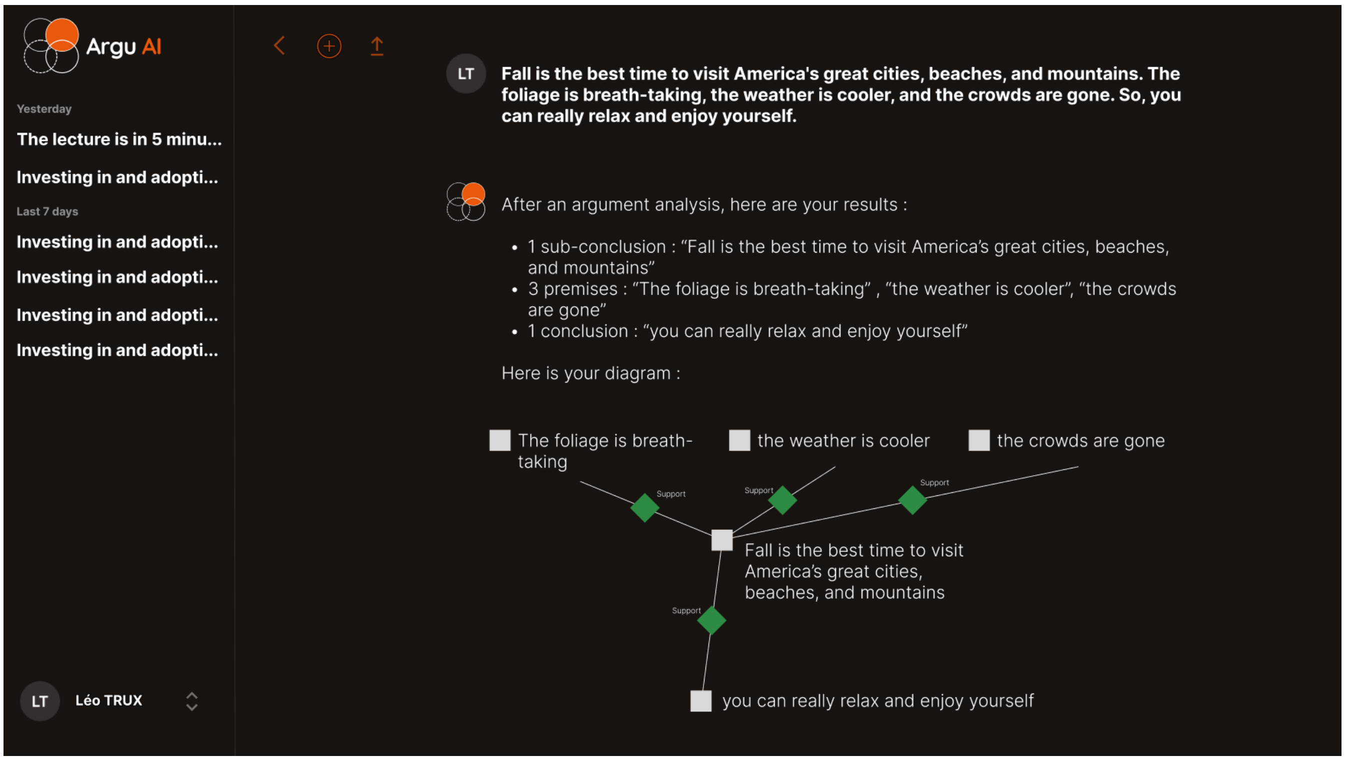


Argu AI
Yesterday
The lecture is in 5 minu...
Investing in and adopti...
Last 7 days
Investing in and adopti...
Investing in and adopti...
Investing in and adopti...
Investing in and adopti...
LT Léo TRUX

Enter your text to be analyzed

This is a beta version. Results may be inaccurate

Figure 7 - Home page design



Argu AI
Yesterday
The lecture is in 5 minu...
Investing in and adopti...
Last 7 days
Investing in and adopti...
Investing in and adopti...
Investing in and adopti...
Investing in and adopti...
LT Léo TRUX

LT **Fall is the best time to visit America's great cities, beaches, and mountains. The foliage is breath-taking, the weather is cooler, and the crowds are gone. So, you can really relax and enjoy yourself.**

After an argument analysis, here are your results :

- 1 sub-conclusion : "Fall is the best time to visit America's great cities, beaches, and mountains"
- 3 premises : "The foliage is breath-taking" , "the weather is cooler", "the crowds are gone"
- 1 conclusion : "you can really relax and enjoy yourself"

Here is your diagram :

The foliage is breath-taking
the weather is cooler
the crowds are gone
Support
Support
Support
Fall is the best time to visit America's great cities, beaches, and mountains
Support
you can really relax and enjoy yourself

Figure 8 - Home page (after argumentative analysis) design

V.5 Planning

Week 1	Research into the concepts of the subject
Week 2	Research into the concepts of the subject
Week 3	Preparing the analysis & conception files
Week 4	UI development (frontend)
Week 5	UI development (frontend)
Week 6	API Flask development (backend)
Week 7	AI development (backend)
Week 8	AI development (backend)
Week 9	AI implementation & testing
Week 10	Code review + code optimisation

Figure 9 - Project's planning

At the start of the project, the emphasis was on in-depth research into the concepts involved in argumentative analysis, the technologies to be used and an overall understanding of the project. The research continued over the first two weeks, with a focus on gathering documentary resources, studying existing examples and familiarising ourselves with the necessary tools.

The third week was devoted to preparing the analysis and design file and drawing up the development plan. Weeks 4 and 5 were then devoted to developing the user interface using React. Week 6 marked the start of the backend development using Flask.

Weeks 7 and 8 were entirely devoted to developing the artificial intelligence models. This process involved selecting the appropriate models, training these models with the prepared dataset, and optimising the algorithms to ensure accurate and reliable analyses of the arguments in the submitted texts.

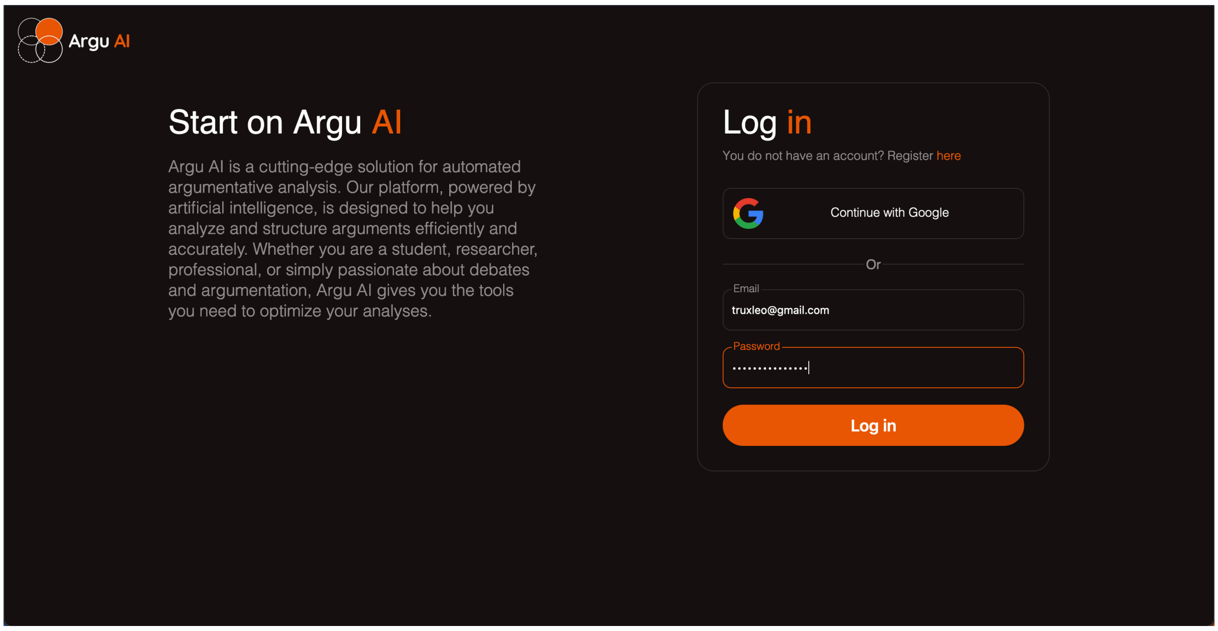
Week 9, 10 were devoted to implementing the AI models in the overall system and carrying out rigorous tests to check that they were working properly. These tests ensured smooth integration between the frontend, the backend and the AI models, guaranteeing a fluid and efficient user experience. Finally, the last week was devoted to optimising the code.

VI. REALISATION

VI.1 WEBSITE DEVELOPMENT

The first stage of development was to build the website itself so that the website could offer users a fluid, intuitive and fast experience, without them getting lost on the site. Using React, HTML, CSS and the Flask API, I was able to achieve the following results, which I'm going to present to you by main part.

VI.1.1 Authentication



Argu AI

Start on Argu AI

Argu AI is a cutting-edge solution for automated argumentative analysis. Our platform, powered by artificial intelligence, is designed to help you analyze and structure arguments efficiently and accurately. Whether you are a student, researcher, professional, or simply passionate about debates and argumentation, Argu AI gives you the tools you need to optimize your analyses.

Log in

You do not have an account? Register here

Continue with Google

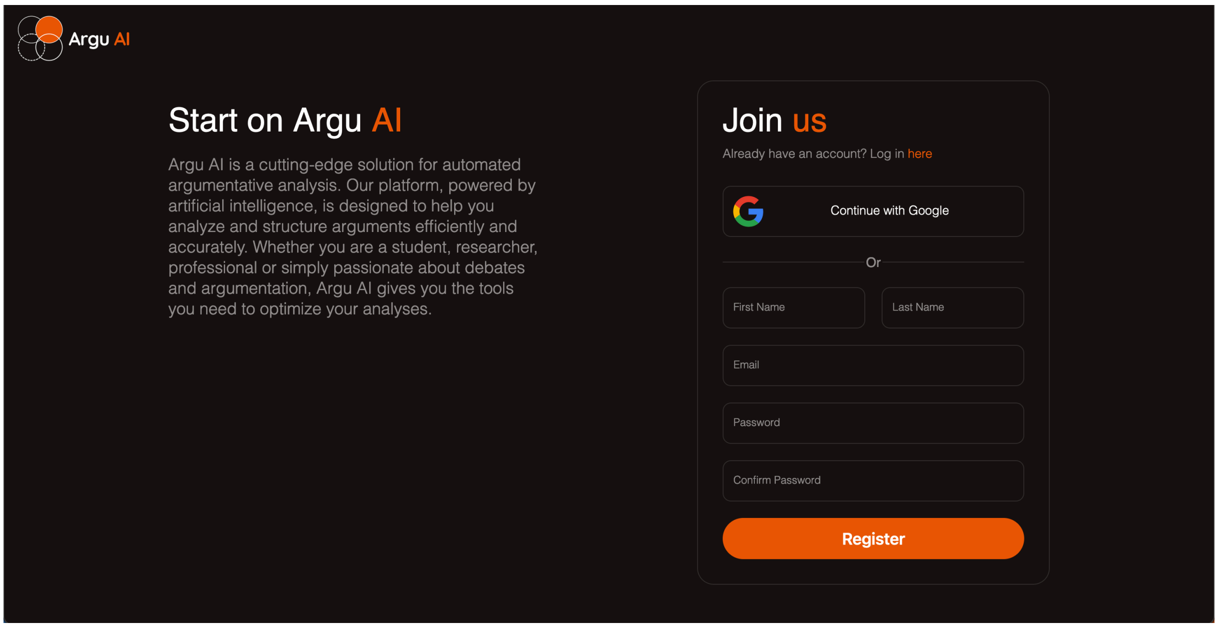
Or

Email: truxleo@gmail.com

Password: ················

Log in

Figure 10 - Login page



Start on Argu AI

Argu AI is a cutting-edge solution for automated argumentative analysis. Our platform, powered by artificial intelligence, is designed to help you analyze and structure arguments efficiently and accurately. Whether you are a student, researcher, professional or simply passionate about debates and argumentation, Argu AI gives you the tools you need to optimize your analyses.

Join us

Already have an account? Log in here

Figure 11 - Register page

Authentication brings together the login and account creation functions. The login page allows users to log in either by using their Google Account for a quick and secure connection, or by filling in a form with their email address and password. Error messages are displayed if the user enters incorrect information or leaves fields blank, providing clear instructions for correcting errors. The account creation page is similar to the login page but includes additional fields to collect the information needed to create an account, such as first and last name, email address, password and password confirmation.

To implement the authentication functions, I had to develop several **React components**. The main components include the Login and Register forms. These are themselves made up of reusable sub-components for input fields and buttons. These components have been designed to be modular and easily reusable, enabling efficient management of user interactions and form validations. The **integration of Firebase** played a crucial role in the management of authentication and the storage of user data. The database (Firestore) was configured to securely store user information such as names, email addresses and unique identifiers (UIDs). This database was then linked to the React application to ensure real-time synchronisation of user data. In addition, authentication via Firebase was configured to enable a secure connection with Google, providing a fast and convenient connection method for users. By combining these components and services, I was able to create a complete authentication system, offering both the flexibility of manual login and the convenience of logging in via Google.

On the site, there are therefore two ways of authenticating, and each method uses a different algorithm.

The first method of authentication using email and password when filling in the form is very standard. When the user enters their email address and password in the connection form, this information is immediately validated on the client side to check that it conforms to the expected formats (for example, a valid email address and a non-empty password). If the information is valid, it is then sent to Firebase Auth via the '*signInWithEmailAndPassword'* API. Firebase then checks the information provided and, if correct, authenticates the user by returning an authentication token and the user's details. In the event of an error, such as an incorrect password or an unregistered email address, an error message is returned by Firebase and displayed to the user. This process ensures that only the correct information is used for access, while providing immediate feedback in the event of a problem.

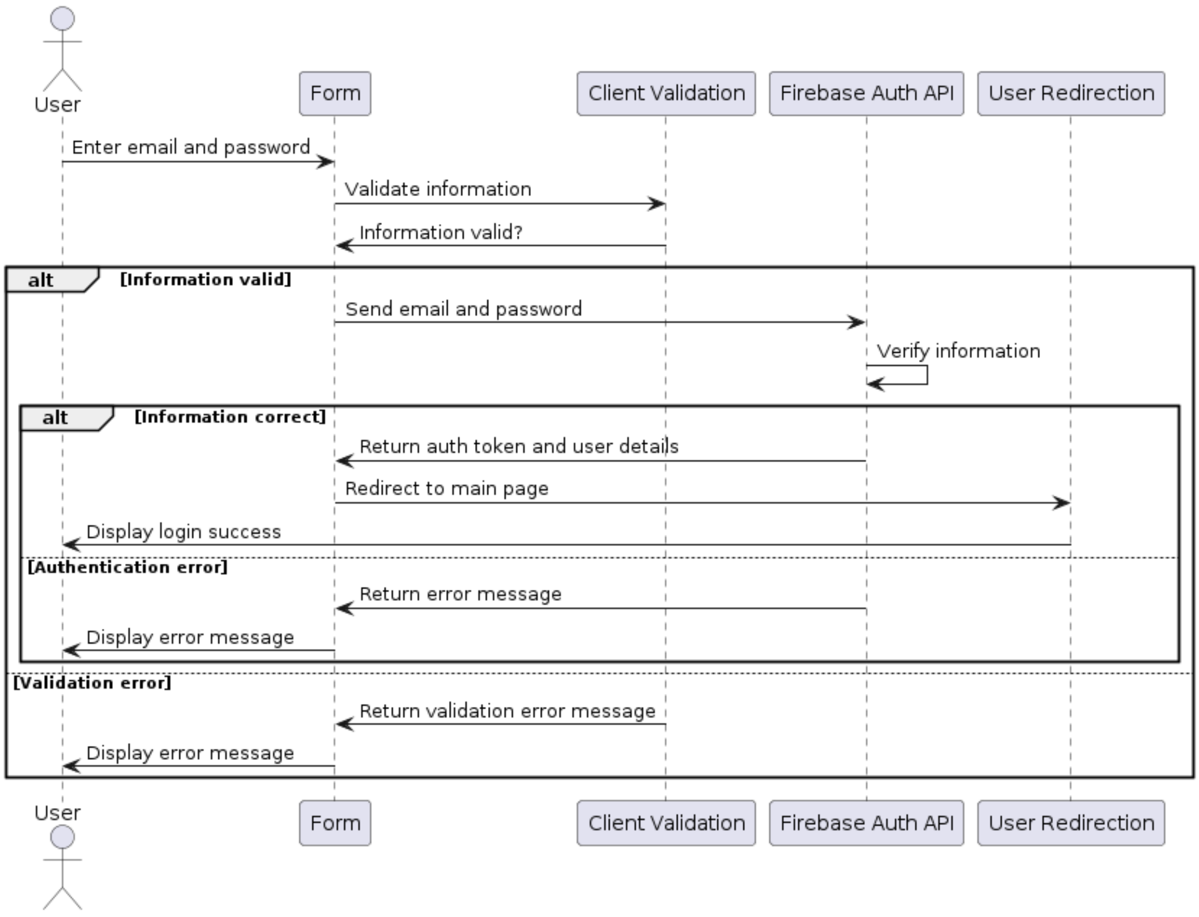


Figure 12 - Authentication sequence diagram, form method

The second method, authentication with Google, uses Firebase's GoogleAuthProvider, enabling a fast and secure connection. When the user clicks on the ‘Sign in with Google’ button, the React component triggers the ‘*signInWithPopup’* method using the GoogleAuthProvider. A Google popup window then opens, asking the user to sign in with their Google account. Once the Google login details have been provided, Firebase receives this information and proceeds to create or retrieve the corresponding user in Firebase Auth. If the user is new, a user document is created in Firestore with the necessary information. If the connection is successful, the user is redirected to the main page of the application. If there is an error during the process, an error message is displayed to the user.

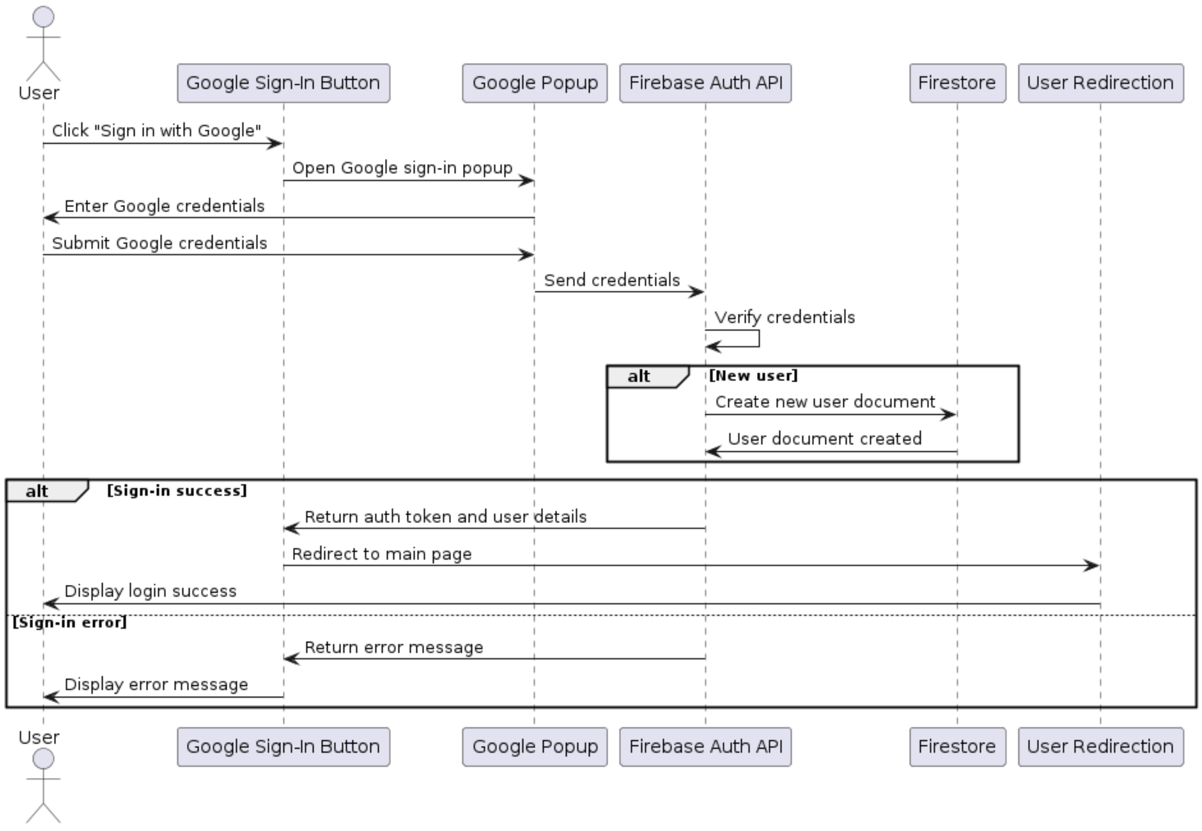


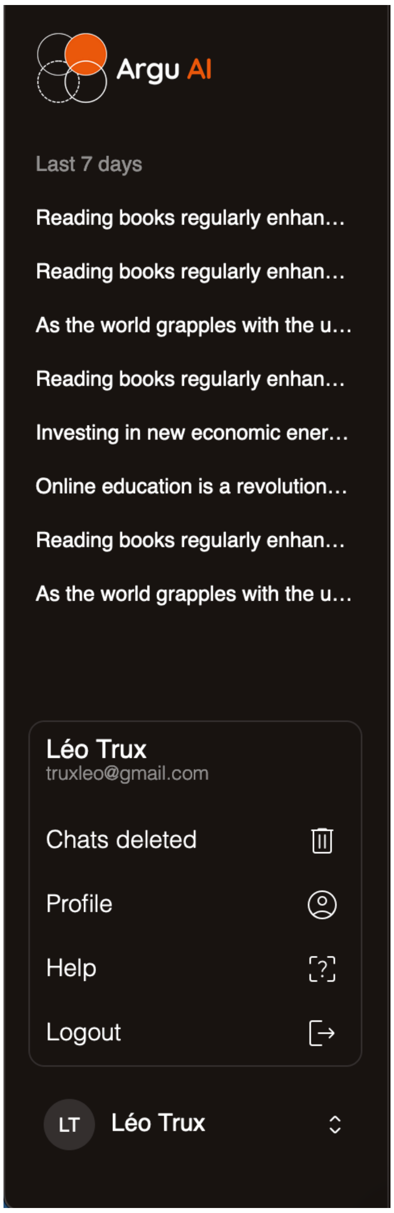
Figure 13 - Authentication sequence diagram, Google method

During the implementation of authentication, several technical difficulties were encountered and resolved. Firstly, managing users logged in via Google or via the login form caused problems due to differences in user data. The information provided by Google, such as full name and email address, is extracted directly from the user's Google profile, whereas users who sign up via the form provide this information manually. So, I ended up with a different structure of users. To harmonise the data, I had to define a common user strcture in Firestore, where basic information such as the UID, first name, last name and email are stored uniformly, regardless of the connection method used.

Finally, the management of the authentication object ('Auth') once the user had logged in and its transmission through all the application components posed a problem. Some of my components didn't have access to it. Once the user was logged in, all I had to do was retrieve the Firebase Auth object from my App component (parent component) and pass it on to all the child components to gain access to the logged-in user's data throughout the site. To do this, a global authentication context was created using the React Context. This enables the user's authentication state to be stored and made accessible to all components, ensuring consistent management of the connection state. In this way, each component can access the logged-in user's information without having to request this information again, improving the application's efficiency and responsiveness.

VI.1.2 Navigation & user Interface

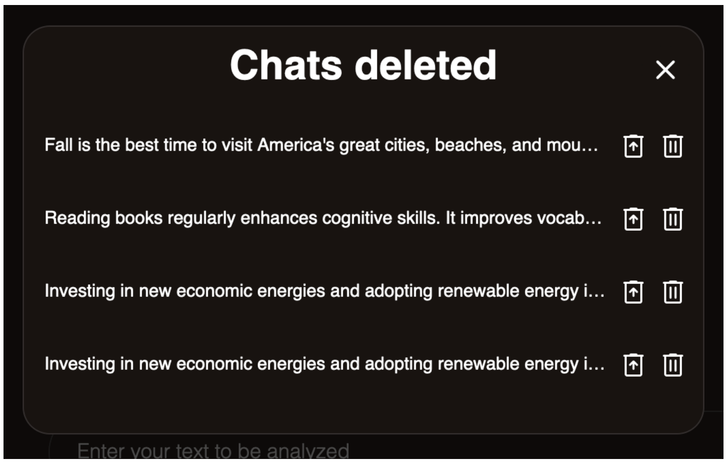
Site navigation is mainly managed by a sidebar that allows users to move easily between the different sections of the site, such as the different argumentative analyses of the user, access to the deleted chat (modal) or user profile management (modal).



Reading books regularly enhan…
Reading books regularly enhan…
As the world grapples with the u…
Reading books regularly enhan…
Investing in new economic ener…
Online education is a revolution…
Reading books regularly enhan…
As the world grapples with the u…

Léo Trux
truxleo@gmail.com

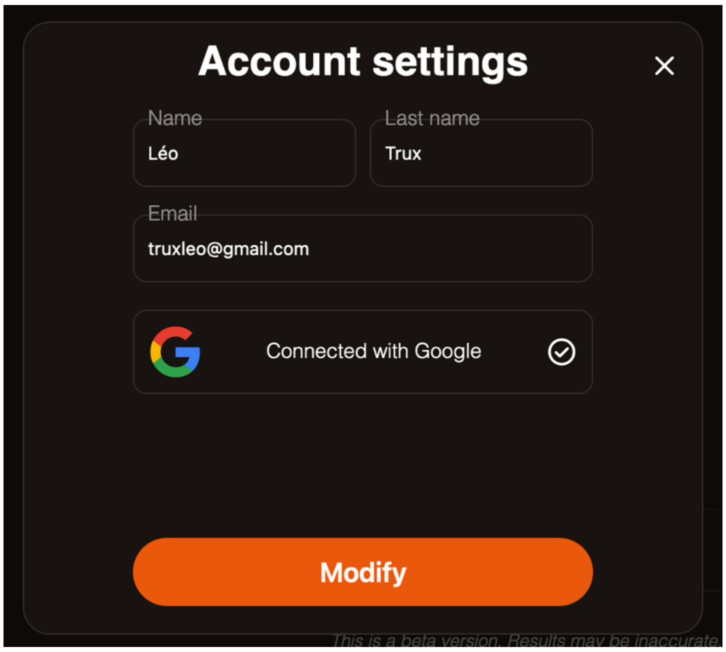
Figure 14 – Sidebar



Fall is the best time to visit America's great cities, beaches, and mou…
Reading books regularly enhances cognitive skills. It improves vocab…
Investing in new economic energies and adopting renewable energy i…
Investing in new economic energies and adopting renewable energy i…

Enter your text to be analyzed

Figure 15 - Modal chat deleted



Name: Léo
Last name: Trux
Email: truxleo@gmail.com

This is a beta version. Results may be inaccurate

Figure 16 - Modal account settings

Several React components were developed and integrated to create the navigation and user interface. **The sidebar** was designed to be dynamic and responsive, enabling fluid navigation between the different sections of the site. **Modals** have been implemented to offer additional interactions without interrupting the user experience. These components use the global context of the application to access user data and maintain consistency of state across the site.

At the same time, I set up a **Flask API** using python to interact with the database. This API makes it possible to retrieve and modify elements, such as chats or user information, in a secure and efficient way. The Flask API manages the HTTP requests sent by the React components and performs the necessary operations in Python on the database, such as retrieving user information, updating profiles and managing deleted chats. Thanks to this architecture, the front-end components can communicate seamlessly with the back end, ensuring fluid, responsive integration of the site's various functionalities.

One of the major strengths of using React to develop the user interface is React Router's ‘navigate’ function. This function considerably improves the user experience by enabling smooth navigation without page reloads. When the user clicks on a link in the sidebar or performs an action, the ‘navigate’ function modifies the browser URL and renders the appropriate component without refreshing the entire page. This not only speeds up transitions between pages, but also **preserves the state** of the application and **reduces the load** on the server.

To manage interactions between the frontend and the backend, a Flask API has been implemented. This API is essential for retrieving and modifying the data stored in the database, ensuring fluid communication between the client (React) and the server (Flask). The client-server relationship works as follows: the React components send HTTP requests (GET, POST, PUT, DELETE) to the Flask API depending on the user's actions. For example, when user information is modified, a PUT request is sent to the API with the new data. The Flask API receives these requests and performs the necessary operations on the database. This may include retrieving user information, updating profiles, or managing deleted chats. After processing the request, Flask sends a response back to the client with the results of the operation. For example, a successful request to update the user profile will return a success status and the new updated information. The React frontend receives the response and updates the user interface accordingly. The components are re-rendered with the new data, ensuring that the user sees the changes immediately.

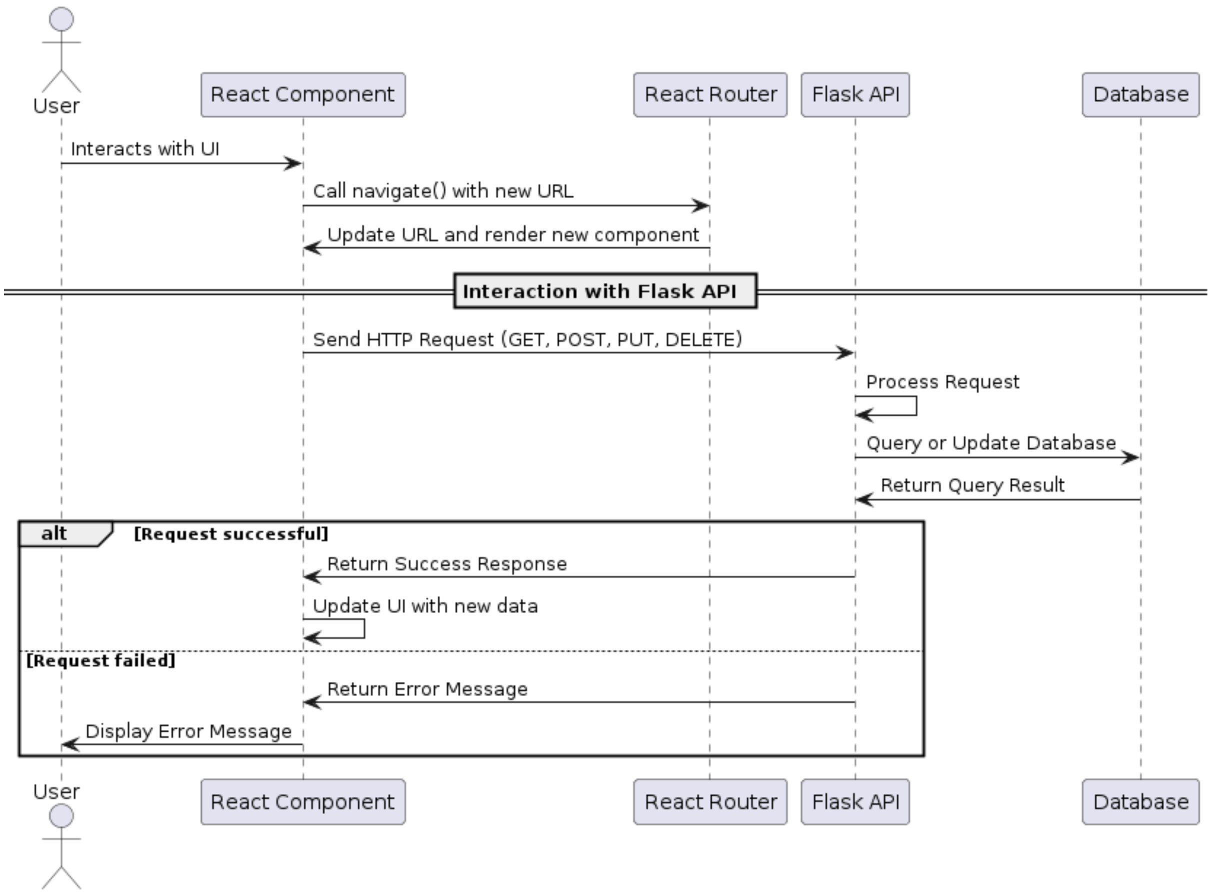
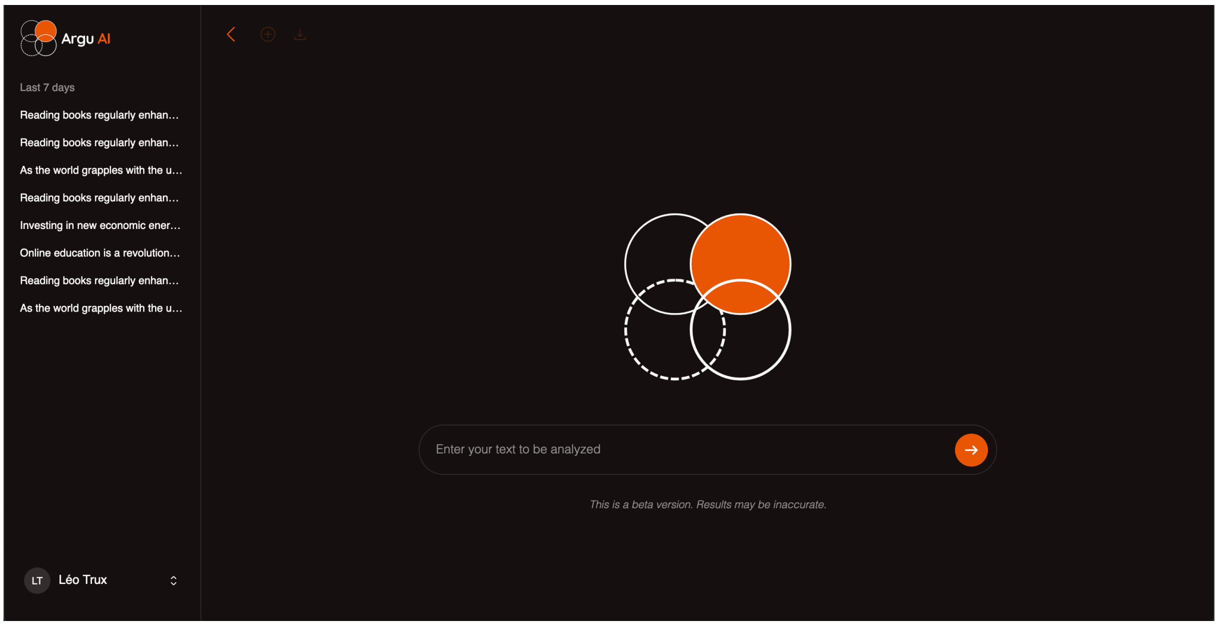


Figure 17 - Navigate & relation client-server with Flask API

When the Flask API was implemented, a major difficulty was encountered concerning communication with the various API routes. Initially, requests sent from the frontend were not allowed to access the API routes due to security restrictions linked to CORS (Cross-Origin Resource Sharing) policies. This restriction prevented the frontend from communicating effectively with the backend, making it impossible to retrieve or modify the necessary data. To solve this problem, it was necessary to do a thorough search of the Flask documentation and try out various configurations. After several attempts, I found that the addition of CORS authorization was necessary to allow requests between the frontend and the backend. By integrating the Flask-CORS extension into the Flask application, the problem was solved.

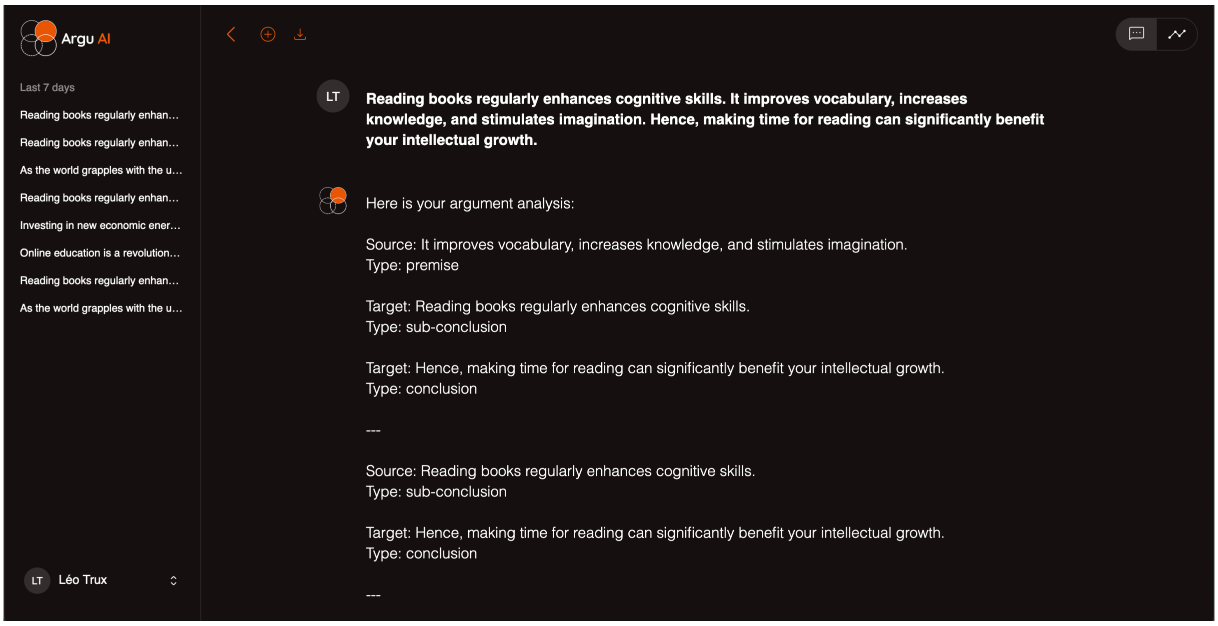
VI.1.3 Analysis management

This section is the heart of the website, as it is here that users can enter their text to perform an argumentative analysis. The analysis is carried out on the following two pages: firstly, the home page, inviting the user to enter the text to be analysed, and then the results presentation page, which appears once the argumentative analysis has been completed. On the results presentation page, two 'display modes' are available for viewing the results: text mode and graphic mode.



Argu AI
Last 7 days
Reading books regularly enhan...
Reading books regularly enhan...
As the world grapples with the u...
Reading books regularly enhan...
Investing in new economic ener...
Online education is a revolution...
Reading books regularly enhan...
As the world grapples with the u...
Enter your text to be analyzed
This is a beta version. Results may be inaccurate.
LT
Léo Trux

Figure 18 - Home page (before argumentative analysis)



Argu AI
Last 7 days
Reading books regularly enhan...
Reading books regularly enhan...
As the world grapples with the u...
Reading books regularly enhan...
Investing in new economic ener...
Online education is a revolution...
Reading books regularly enhan...
As the world grapples with the u...
LT
Léo Trux

LT **Reading books regularly enhances cognitive skills. It improves vocabulary, increases knowledge, and stimulates imagination. Hence, making time for reading can significantly benefit your intellectual growth.**

Here is your argument analysis:

Source: It improves vocabulary, increases knowledge, and stimulates imagination.
Type: premise

Target: Reading books regularly enhances cognitive skills.
Type: sub-conclusion

Target: Hence, making time for reading can significantly benefit your intellectual growth.
Type: conclusion

Source: Reading books regularly enhances cognitive skills.
Type: sub-conclusion

Target: Hence, making time for reading can significantly benefit your intellectual growth.
Type: conclusion

Figure 19 - Home page (after argumentative analysis, text mode)

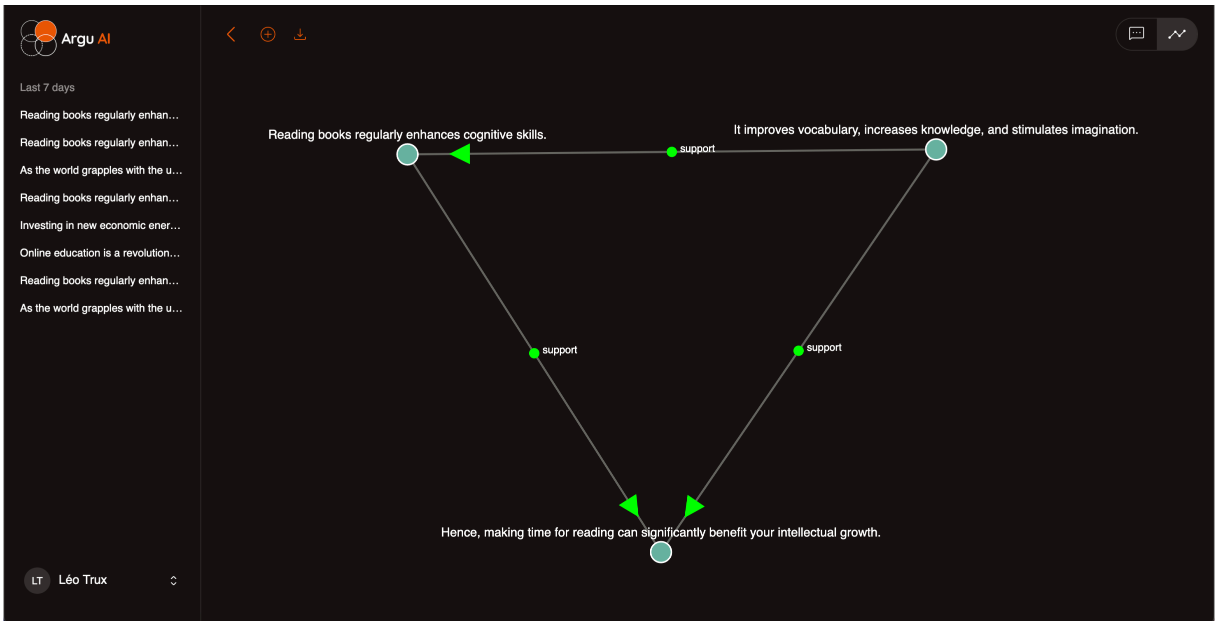
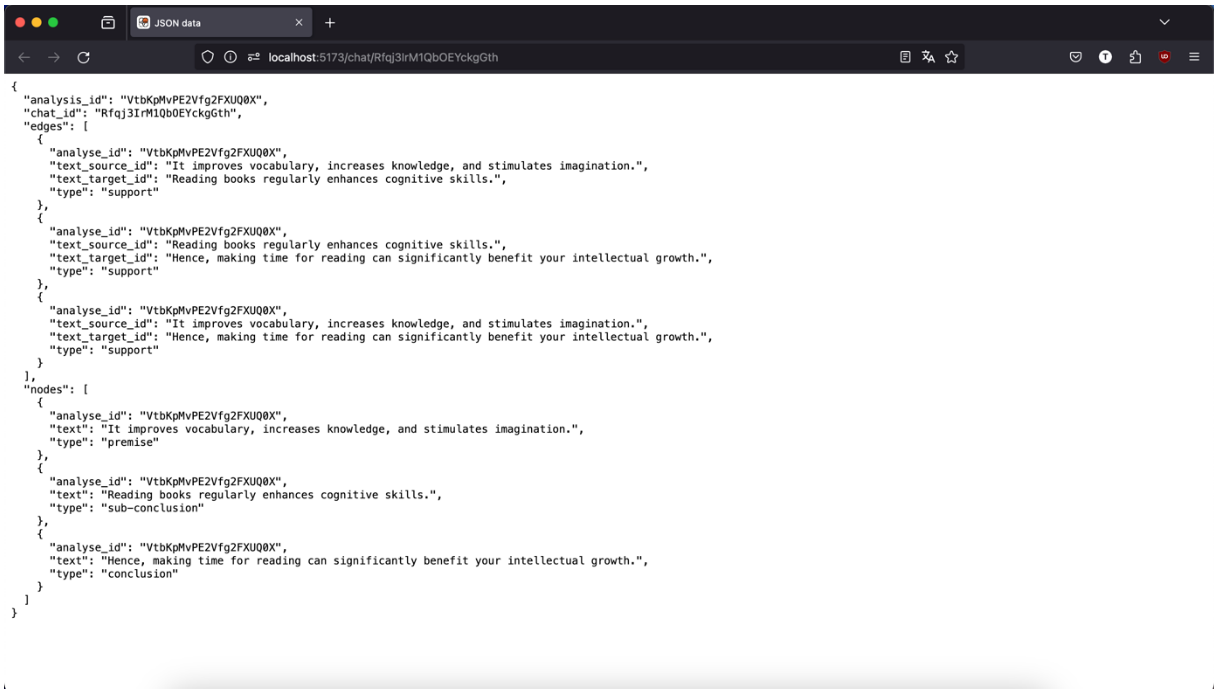

Argu AI
Last 7 days
Reading books regularly enhan...
Reading books regularly enhan...
As the world grapples with the u...
Reading books regularly enhan...
Investing in new economic ener...
Online education is a revolution...
Reading books regularly enhan...
As the world grapples with the u...
LT Léo Trux
Reading books regularly enhances cognitive skills.
It improves vocabulary, increases knowledge, and stimulates imagination.
support
support
support
Hence, making time for reading can significantly benefit your intellectual growth.

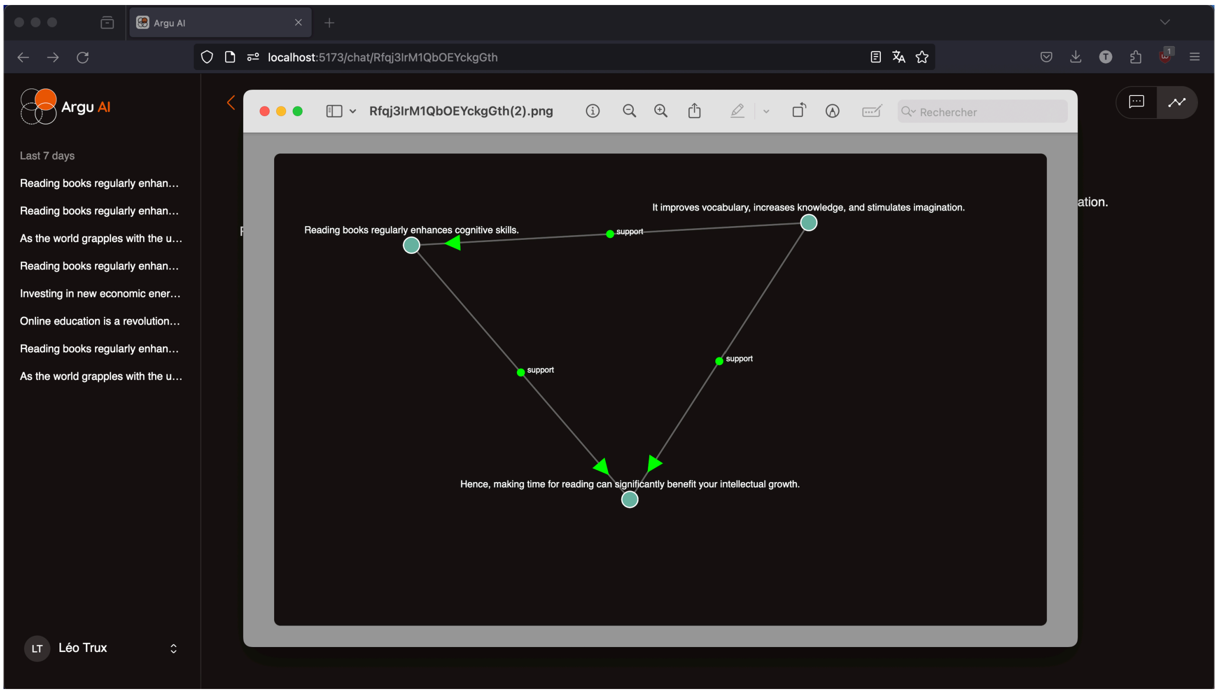
Figure 20 - Home page (after argumentative analysis, diagram mode)

Once the analysis has been completed, the user can export the results in two different ways, either in JSON format or as a PNG diagram.



```
{
  "analysis_id": "VtbKpMvPE2Vfg2FXUQ0X",
  "chat_id": "Rfqj3IrM1QbOEYckgGth",
  "edges": [
    {
      "analyse_id": "VtbKpMvPE2Vfg2FXUQ0X",
      "text_source_id": "It improves vocabulary, increases knowledge, and stimulates imagination.",
      "text_target_id": "Reading books regularly enhances cognitive skills.",
      "type": "support"
    },
    {
      "analyse_id": "VtbKpMvPE2Vfg2FXUQ0X",
      "text_source_id": "Reading books regularly enhances cognitive skills.",
      "text_target_id": "Hence, making time for reading can significantly benefit your intellectual growth.",
      "type": "support"
    },
    {
      "analyse_id": "VtbKpMvPE2Vfg2FXUQ0X",
      "text_source_id": "It improves vocabulary, increases knowledge, and stimulates imagination.",
      "text_target_id": "Hence, making time for reading can significantly benefit your intellectual growth.",
      "type": "support"
    }
  ],
  "nodes": [
    {
      "analyse_id": "VtbKpMvPE2Vfg2FXUQ0X",
      "text": "It improves vocabulary, increases knowledge, and stimulates imagination.",
      "type": "premise"
    },
    {
      "analyse_id": "VtbKpMvPE2Vfg2FXUQ0X",
      "text": "Reading books regularly enhances cognitive skills.",
      "type": "sub-conclusion"
    },
    {
      "analyse_id": "VtbKpMvPE2Vfg2FXUQ0X",
      "text": "Hence, making time for reading can significantly benefit your intellectual growth.",
      "type": "conclusion"
    }
  ]
}
```

Figure 21 - JSON export data



Argu AI
Last 7 days
Reading books regularly enhan...
Reading books regularly enhan...
As the world grapples with the u...
Reading books regularly enhan...
Investing in new economic ener...
Online education is a revolution...
Reading books regularly enhan...
As the world grapples with the u...
Léo Trux
localhost:5173/chat/Rfqj3lrM1QbOEYckgGth
Rfqj3lrM1QbOEYckgGth(2).png
Rechercher
It improves vocabulary, increases knowledge, and stimulates imagination.
Reading books regularly enhances cognitive skills.
support
support
support
Hence, making time for reading can significantly benefit your intellectual growth.

Figure 22 - PGN export data

To carry out the argumentative analysis and present the results effectively, I had to develop two main components on the frontend: ‘*NewChat*’ and ‘*ChatContainer*’. *NewChat* groups together the page that allows the user to enter their text and *ChatContainer* groups together the results page, after the argumentative analysis has been carried out. To create the graph showing the results of the analysis, I had to use *D3.JS*, a JavaScript library for creating graphs.

On the backend, the Flask API plays a crucial role in text processing and data management. Several routes have been put in place, notably the ‘chat’ and ‘analysis’ routes, to store and manage the data linked to argumentative analysis. Texts submitted by users are sent to these routes to be stored, analysed and the results returned to the frontend for display.

It is important to note that the argumentative analysis is carried out using artificial intelligence models that I had to create for this project. However, this part will not be covered here; a dedicated section in this report will provide a detailed explanation of the implementation and operation of the AI models.

The important thing here is to understand how the argumentative analysis is carried out. When the user enters their text into the '*NewChat*' component and submits it, several steps are triggered to process the text and navigate to the results page. Once the text has been submitted, it is sent to the Flask API '*Chats*' endpoint, which adds the text to the Firestore database and creates a new chat. The sidebar is then notified to refresh and display this new chat. The argumentative analysis is then carried out using AI models and recorded in the database via the '*Analyses*' endpoint of the Flask API. Once the analysis has been completed and the data stored, React Router's '*Navigate*' function is used to redirect the user to the '*ChatContainer*' page using the chat ID created. On this page, the results of the analysis are retrieved from the database based on the chat ID and presented to the user in detail, including interactive visualisations created using D3.js.

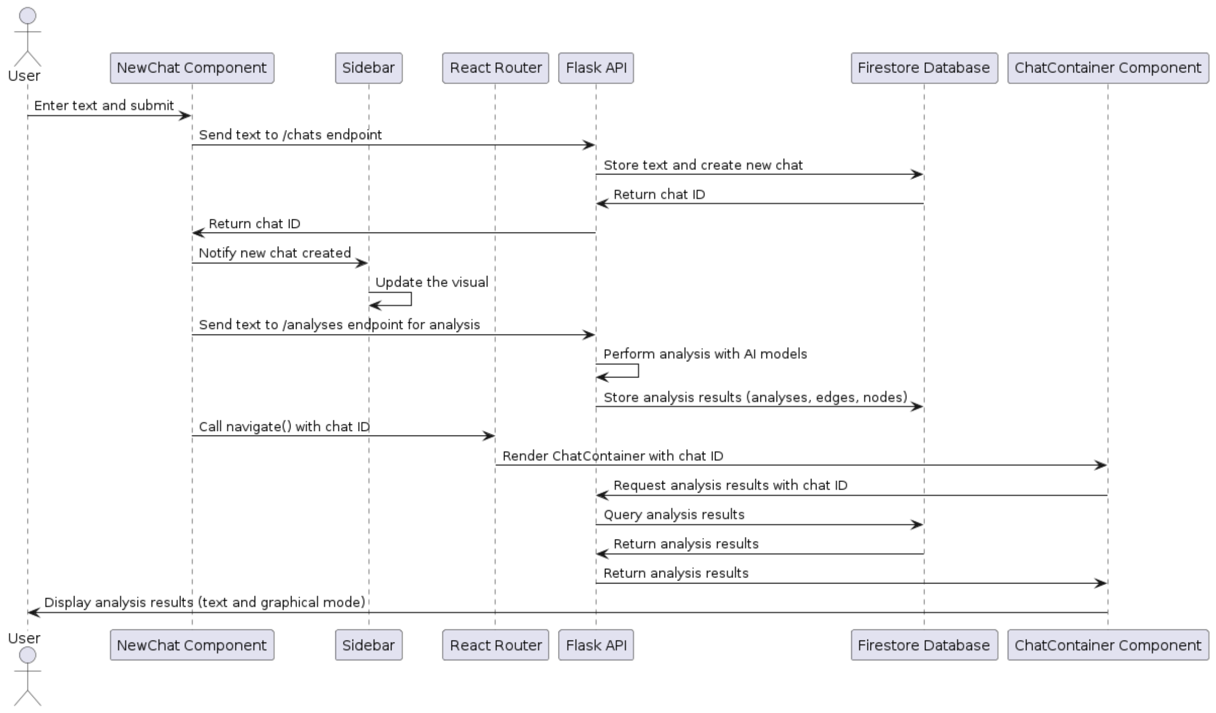

User
NewChat Component
Sidebar
React Router
Flask API
Firestore Database
ChatContainer Component
Enter text and submit
Send text to /chats endpoint
Store text and create new chat
Return chat ID
Return chat ID
Notify new chat created
Update the visual
Send text to /analyses endpoint for analysis
Perform analysis with AI models
Store analysis results (analyses, edges, nodes)
Call navigate() with chat ID
Render ChatContainer with chat ID
Request analysis results with chat ID
Query analysis results
Return analysis results
Return analysis results
Display analysis results (text and graphical mode)

Figure 23 - Sequence diagram of analyse management

In this part, the difficulty was not so much technical as conceptual. It was mainly a question of making API calls and being rigorous about inserting data. I had done this stage at the end of the web development, after overcoming the difficulties associated with the Flask API in the previous parts. The only technical difficulty here was discovering the D3.js library in order to create the graph showing the results of the analysis. Fortunately for me, the documentation for this library is very well done and enabled me to create the graph I was imagining.

The main challenge here was to design a coherent and efficient flow for the argumentative analysis process. To meet this conceptual challenge, I had to take inspiration from sites that use chatbots in order to understand how they work. By observing these sites, I was able to learn and draw inspiration from how they manage user interactions and transitions between different stages of the process. So I tried to understand and reproduce how they worked to come up with a solution that worked for my site.

VI.2 Dataset creation

The next major task was to develop the AI models to automate the argumentative analysis. However, before this could be done, it was essential to create a solid dataset to guarantee the efficiency and accuracy of these models.

To develop AI models capable of automating argumentative analysis, it was crucial to create a solid dataset. As mentioned in the section on the structure of the dataset in the project analysis, this structure was inspired by the *SADface* (JSON) schema of *MonkeyPuzzle*. Thanks to the *MonkeyPuzzle* database, provided by my tutor, and the OpenAI API, I was able to create a high-quality dataset in sufficient quantity. This dataset contains around a thousand annotated argumentative analyses covering various subjects. Let me insist that the quantity of data alone does not guarantee the quality of the dataset. The diversity and precision of the examples are essential to improve the performance of the models.

To create a solid dataset that could be used to develop AI models, it was necessary to start from the *Monkeypuzzle* database. So, the first stage had already been completed, I had a database to build on. However, this database wasn't properly annotated and contained information that wasn't useful to me. Re-annotating the entire dataset manually was impossible due to the time and resources it would have required. To overcome this challenge, I used a semi-manual method using the OpenAI API to facilitate the annotation process. This API allows users to send queries to an NLP (Natural Language Processing) model in order to meet their needs.

The aim of using this method was simple: **save much time as possible** by avoiding redundant tasks on a large amount of data. The program first retrieves a JSON element from the *MonkeyPuzzle* database and removes all the data that was useless to me. Then, it provides the new cleaned JSON to the *openAI* API so that it can identify the types of each segment (premises, conclusions, conflicts, etc). In order for this task to be carried out in the best possible way, it was essential to give a context and an instruction to the NLP model of the *openAI* API. This is called a prompt. For my part, here's the prompt I gave it:

```
You are an intelligent assistant who helps to classify elements in structured
debates. Your task is to receive a JSON object containing "nodes" and "edges",
classify each node as "premise", "sub-conclusion", "conclusion", etc., and classify
the relationships between nodes as "conflict" or "support". You must return only
the node IDs and their types, as well as the types of relationships between nodes
specified in "edges". Return only the following information in JSON format:
You can't have in the nodes section a "type": "atom" !!!
In the edge section, the type for each relation must be support or conflict,
nothing else !
{
    "nodes": [
        {"id": "node_id_1", "type": "node_type_1"},
        {"id": "node_id_2", "type": "node_type_2"},
        ...
    ],
    "edges": [
        {"source_id": "source_node_id_1", "target_id": "target_node_id_1", "type":
"relation_type_1"},
        {"source_id": "source_node_id_2", "target_id": "target_node_id_2", "type":
"relation_type_2"},
        ...
    ]
}
```

Once the API had returned the result, all I had to do was check the data structure of the result, and if it was correct, I added it to my dataset. To write a good prompt, I followed 3 steps in his writing:

1. **Giving a context:** Setting the context ensures that the model understands the background and scope of the task. This helps the model generate responses that are relevant and appropriate to the given situation.

2. **Giving a guideline:** Clearly stating the task helps the model understand exactly what is expected of it. Guidelines direct the model's focus, ensuring that it performs the desired action.

3. **Giving an example:** Examples serve as a concrete reference for the model, illustrating the type of output that is expected.

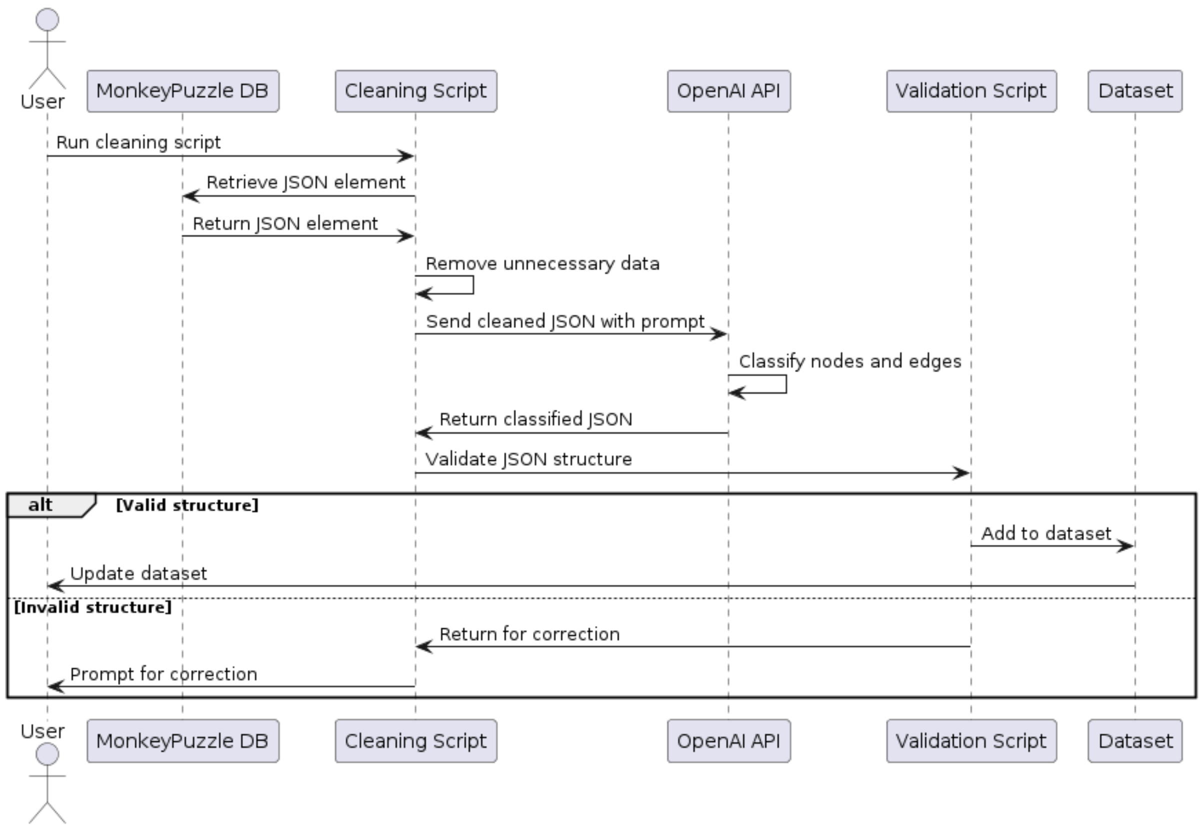


Figure 24 - Sequence diagram of dataset's creation

The main difficulty encountered when creating the dataset was using the *openAI* API. The difficulty didn't lie in implementing the API, which was easy to do thanks to the *openAI* documentation, but in optimising the prompts to obtain accurate and reliable results. It was necessary for me to test a large number of prompts to find the one that produced the best results with the fewest errors. But even when I did this, the results of the API were still unpredictable and could at any time cause errors that would crash the program. I therefore had to analyse most of the recurring errors in order to prevent the programme from crashing.

VI.3 AI MODELS

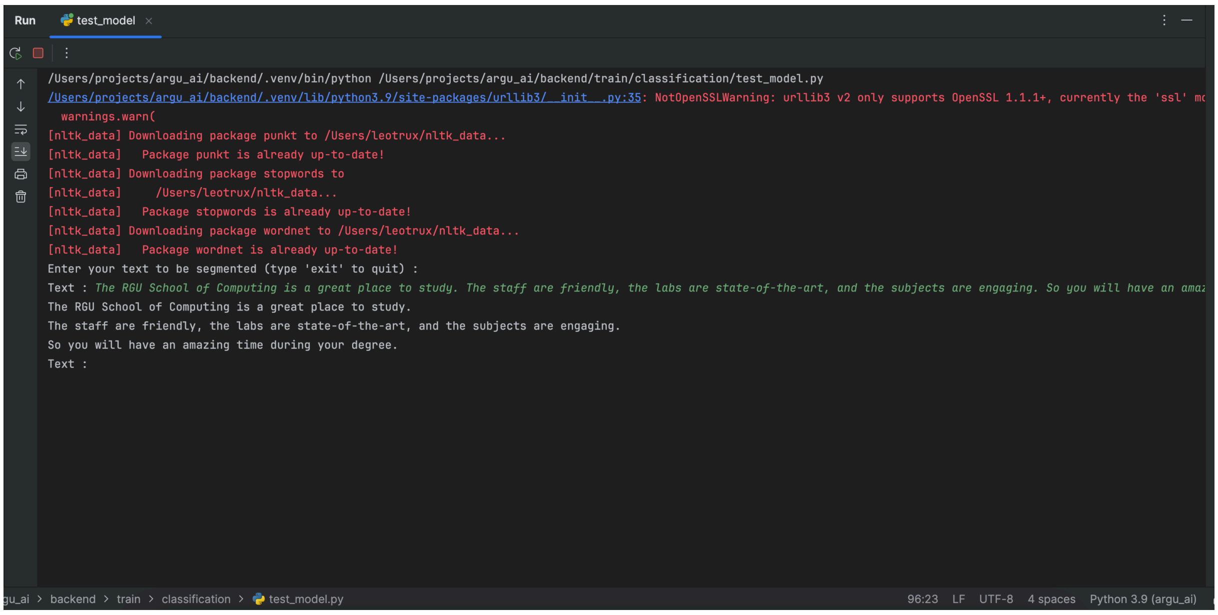
Once the dataset had been built up, I could move on to the main part of the project: creating the AI models. As mentioned above, the aim of these models is to optimise the process of argumentative analysis of a text. As a reminder, there are three main stages in this process: **segmentation**, **classification** and **reference**.

VI.3.1 Segmentation model

The aim here was to set up a text segmentation model, which will make it possible to segment the text into several bits of text that make sense. For example, if we want to segment the following sentence:

The RGU School of Computing is a great place to study. The staff are friendly, the labs are state-of-the-art, and the subjects are engaging. So, you will have an amazing time during your degree.

Entering it into the programme gives us the following result:



```
/Users/projects/argu_ai/backend/.venv/bin/python /Users/projects/argu_ai/backend/train/classification/test_model.py
/Users/projects/argu_ai/backend/.venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' mo
  warnings.warn(
[nltk_data] Downloading package punkt to /Users/leotrux/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/leotrux/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/leotrux/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Enter your text to be segmented (type 'exit' to quit) :
Text : The RGU School of Computing is a great place to study. The staff are friendly, the labs are state-of-the-art, and the subjects are engaging. So you will have an amaz
The RGU School of Computing is a great place to study.
The staff are friendly, the labs are state-of-the-art, and the subjects are engaging.
So you will have an amazing time during your degree.
Text :
```

gu_ai > backend > train > classification > test_model.py
96:23 LF UTF-8 4 spaces Python 3.9 (argu_ai)

Figure 25 - Segmentation model results

As illustrated in the figure above, each segment represents a significant textual unit, facilitating argumentative analysis and the subsequent stages of classification and reference.

To achieve these segmentation results, several steps were implemented. Firstly, I used the *'spaCy'* library and its pre-trained model `*en_core_web_sm*` for natural language processing. In fact, I didn't create a model here, I used one that was already pre-trained in order to save time. The spaCy `*en_core_web_sm*` model is a pre-trained natural language processing (NLP) model that provides tools for parsing, named entity recognition, and other NLP tasks. Here is a more detailed explanation of how it works. The model comprises several key components. For example, here's what happens for each step in the model with the following sentence: 'The RGU school is a great place to study'.

This example is available on page 66 in the appendix.

However, the model alone **did not solve my objective**. It lacked precision and segmented the text into sentences that were too long. I therefore had to develop a custom function to improve the segmentation of the template. This function first uses the spaCy model to analyse the text, then applies specific rules to determine the segmentation points based on punctuation, such as full stops, question marks, exclamation marks, semi-colons and spaces. The results of the model are then adjusted by re-segmenting certain segments using continuation keywords such as "therefore", "thus", "so", etc., in order to obtain more precise segments.

The function starts by initialising a list to store the text segments and uses variables to manage the segmentation indices and continuation segments. By traversing each token in the text document, the function segments the text when punctuation marks indicating the end of a sentence are encountered, or before continuation keywords to maintain argumentative coherence. Commas are also handled so as not to interrupt continuation segments.

This function is available on page 68 in the appendix.

The main difficulty encountered here was directing spaCy's pre-trained model towards an outcome that would be beneficial for my purpose: segmentation for argumentative analysis. The spaCy model, although powerful, did not segment the text precisely enough to meet the specific needs of argumentative analysis. To overcome this difficulty, I had to test several techniques to steer the result of the segmentation function towards what I expected. I have also found that premises are often identified together if they are part of the same sentence. For example, the sentence ***'The staff are friendly, the labs are state-of-the-art, and the subjects are engaging***' is segmented by my model as follows: *'The staff are friendly, the labs are state-of-the-art, and the subjects are engaging*' when in reality it should read: *'The staff are friendly*' / *'the labs are state-of-the-art*' / *'the subjects are engaging*'. I haven't managed to separate these segments with code, as this would require a much more competent NLP model. However, this won't be a problem for the future.

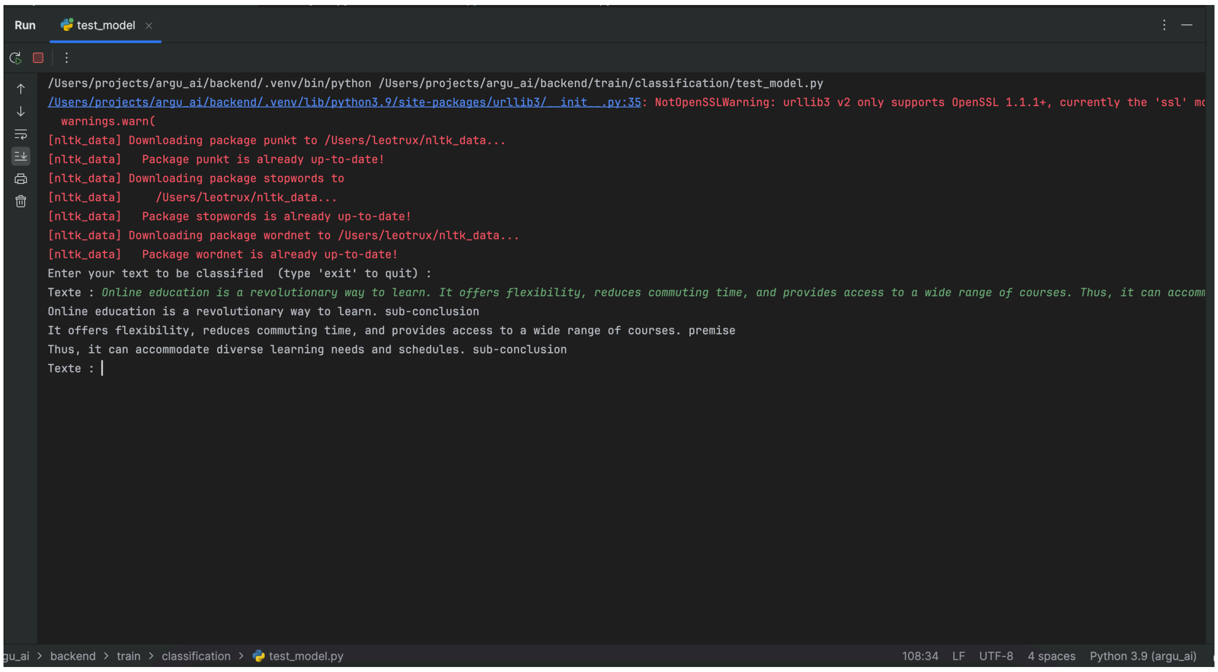
VI.3.2 Classification model

After completing the segmentation model, the next step was to **identify the type of each segment** (premise, sub-conclusion, conclusion, etc.).

To create a classification model capable of categorising the argumentative text segments, I used several data pre-processing techniques, as well as machine learning algorithms to train and evaluate the model. The model was trained to classify text segments into different categories such as "conclusion", "conflict", "premise", and "sub-conclusion". Here are the results I obtained using the classification model. This time, let's use another example sentence:

“Online education is a revolutionary way to learn. It offers flexibility, reduces commuting time, and provides access to a wide range of courses. Thus, it can accommodate diverse learning needs and schedules.”

Entering it into the programme gives us the following result:


Run
test_model
gu_ai > backend > train > classification > test_model.py
108:34 LF UTF-8 4 spaces Python 3.9 (argu_ai)

```
/Users/projects/argu_ai/backend/.venv/bin/python /Users/projects/argu_ai/backend/train/classification/test_model.py
/Users/projects/argu_ai/backend/.venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' mo
  warnings.warn(
[nltk_data] Downloading package punkt to /Users/leotrux/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/leotrux/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/leotrux/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Enter your text to be classified  (type 'exit' to quit) :
Texte : Online education is a revolutionary way to learn. It offers flexibility, reduces commuting time, and provides access to a wide range of courses. Thus, it can accomm
Online education is a revolutionary way to learn. sub-conclusion
It offers flexibility, reduces commuting time, and provides access to a wide range of courses. premise
Thus, it can accommodate diverse learning needs and schedules. sub-conclusion
Texte : |
```

Figure 26 - Classification model results

Here, the model takes each segment as input and predicts its type as output.

To achieve this result, I had to follow some rigorous steps to ensure that the model could learn and generalise effectively on the data. Here are the main steps in the process:

The first step was therefore **to retrieve all the data from the dataset to be able to manipulate it better**, so I retrieved and loaded this data containing, I remind you, the text segments annotated in the form of nodes (textual propositions) and edges (relations). However, for the classification model, I didn't need the edges (relationships) part of the dataset; this data will be used for the relationship model.

Once this stage had been completed, I could move on to the second, **pre-processing the textual data**. This stage involved cleaning up the text by removing special characters and punctuation, tokenising the text into words, removing stopwords and lemmatising to normalise the text. This pre-processing was essential to transform the raw data into a format that could be used by the model. For example, consider the following data from the dataset:

```
{
    "nodes": [
        {
            "id": "12bf09e5-7ed4-49a9-a24d-5f6431cc91d0",
            "text": "Online education is a revolutionary way to learn.",
            "type": "sub-conclusion"
        },
        {
            "id": "47fd0b2b-5a21-422b-9c69-cda9f22b9652",
            "text": "It offers flexibility, reduces commuting time, and
provides access to a wide range of courses.",
            "type": "premise"
        },
        {
            "id": "ad3e1db9-d900-4b75-9fab-cc93383e4232",
            "text": "Thus, it can accommodate diverse learning needs and
schedules.",
            "type": "conclusion"
        }
    ]
}
```

1. Conversion to lower case

Results:

- “online education is a revolutionary way to learn."
- "it offers flexibility, reduces commuting time, and provides access to a wide range of courses."
- "thus, it can accommodate diverse learning needs and schedules."

2. Deleting special characters & punctuation

Results:

- “online education is a revolutionary way to learn"
- "it offers flexibility reduces commuting time and provides access to a wide range of courses"
- "thus it can accommodate diverse learning needs and schedules"

3. Tokenisation in words

Results:

- ["online", "education", "is", "a", "revolutionary", "way", "to", "learn"]"
- ["it", "offers", "flexibility", "reduces", "commuting", "time", "and", "provides", "access", "to", "a", "wide", "range", "of", "courses"]
- ["thus", "it", "can", "accommodate", "diverse", "learning", "needs", "and", "schedules"]

Here, tokenisation involves dividing the text into basic units called tokens. Each word, punctuation mark or number can be a token. This makes it easier to manipulate and analyse the text, as each token can be processed independently. Tokenisation is a crucial step, as it prepares the text for subsequent processing stages, such as deleting empty words and lemmatisation.

4. Deleting stopwords

Results:

- ["online", "education", "revolutionary", "way", "learn"]"
- ["offers", "flexibility", "reduces", "commuting", "time", "provides", "access", "wide", "range", "courses"]
- ["thus", "accommodate", "diverse", "learning", "needs", "schedules"]

Removing stopwords improves the model's performance. Empty words such as 'the', 'and', 'is' etc. do not add significant value to the context of the text. By removing them, we reduce the noise and the number of dimensions in the vector space of words, which improves the efficiency and performance of the model.

5. Lemmatisation

Results:

- ["online", "education", "revolutionary", "way", "learn"]
- ["offer", "flexibility", "reduce", "commuting", "time", "provide", "access", "wide", "range", "course"]
- ["thus", "accommodate", "diverse", "learning", "need", "schedule"]

Lemmatisation involves reducing words to their basic form or lemma. For example, "learning" becomes "learn", "offers" becomes "offer". Lemmatisation is essential for standardising the different forms of a word so that they can be processed in a uniform way by the model. This reduces complexity and improves the accuracy of text analysis.

Pre-processing transforms the raw text into a standardised and structured form, ready for use by machine learning models.

After pre-processing the data, the next step was **to prepare the data for training the model**. This involved dividing the data into training and test sets. A training set is all the data (extracted from the dataset in my case) that are used to adjust the data in the future model. A test set is a set of data different from the training set, enabling the model to assess its ability to generalise its knowledge to new data. Each textual proposition in the dataset was pre-processed using the steps defined above, then converted into numerical representations using ***TF-IDF vectorisation***[2].

The final step was **to choose the classification model and train it**. For training the text classification model, I had to test a lot of models to find the one with the best results, which led me to choose the RandomForestClassifier model because of its robustness and ability to handle complex data. I used ***SMOTE***[3] oversampling to address class imbalance, generating synthetic examples for minority classes to balance the class distribution. In an argumentative analysis, there are inevitably a lot of premises and few sub-conclusions/conclusions, and this was reflected in my dataset. These two classes were therefore under-represented. Next, a grid search (***GridSearchCV***[4]) was carried out to optimise the model's hyperparameters, which led to an improvement in its performance. Hyperparameters are model parameters that cannot be learned directly from the data during training but must be defined before training. They include parameters such as the number of trees in a random forest, the maximum tree depth, the minimum number of samples required to split a node, and so on.

As mentioned previously, I used a Random Forest model to classify the text segments. Random Forest is a supervised learning algorithm that constructs numerous decision trees from random sub-samples of the training dataset to predict the class of a datum.

The Random Forest algorithm consists of 3 key steps:

1. Bootstrap Sampling

The algorithm creates several subsamples of the original training dataset using bootstrap sampling. This means that there will be repeated rows in our sub-samples.

2. Construction of decision trees

For each bootstrapped dataset, a decision tree is constructed using a random selection of attributes at each node. This helps to introduce diversity and reduce correlation between trees.

3. Ensemble voting

For classification tasks, each decision tree in the forest gives a class prediction. The final class assigned to an observation is the one that receives the most votes among all the trees.

Suppose we need to predict the type of a text segment in an argumentative analysis. The segment types are "conclusion", "premise", "sub-conclusion", and "conflict". The features of each segment are the TF-IDF terms extracted from the text. To illustrate, we construct two bootstrap subsamples from our training data:

Original training data

1. "Online education is a revolutionary way to learn." **- sub-conclusion**
2. "It offers flexibility, reduces commuting time, and provides access to a wide range of courses." **- premise**
3. "Thus, it can accommodate diverse learning needs and schedules." - **conclusion**

Boostrapped dataset 1

1. "Online education is a revolutionary way to learn." **- sub-conclusion**
2. "Thus, it can accommodate diverse learning needs and schedules." **- conclusion**

Boostrapped dataset 2

1. "It offers flexibility, reduces commuting time, and provides access to a wide range of courses." **- premise**
2. "Thus, it can accommodate diverse learning needs and schedules." **- conclusion**

In the next step, the model will build a decision tree for each bootstrap dataset based, for each tree, on a random selection of textual attributes, represented by the TF-IDF values of the words present in the text segments. Each node in the tree uses a random subset of these attributes to make decisions and split the data.

The decision trees are available on page 69 in the appendix

The text to be predicted in our example is as follows:

"It offers flexibility, reduces commuting time, and provides access to a wide range of courses."

This will then pass through all the decision trees created, in this case the 2 above.

1. **Decision tree 1**
   - Checks whether "revolutionary" is present (No)
   - Prediction: Conclusion.
2. **Decision tree 2:**
   - Checks whether "flexibility" is present (Yes)
   - Checks whether "reduces" is present (Yes)
   - Prediction: Premise.

The final stage (ensemble voting) consists of recording the results of each decision tree and taking the majority result, which gives a robust and accurate final prediction. For example, if 60% of the trees predict "premise", 30% predict "sub-conclusion", and 10% predict "conclusion", then the majority class is "premise", and it is this class that will be assigned to the text segment in question.

Setting up the Random Forest model presented a number of technical challenges. I had no previous experience of machine learning models, so I had to do a lot of research to understand and apply the necessary concepts.

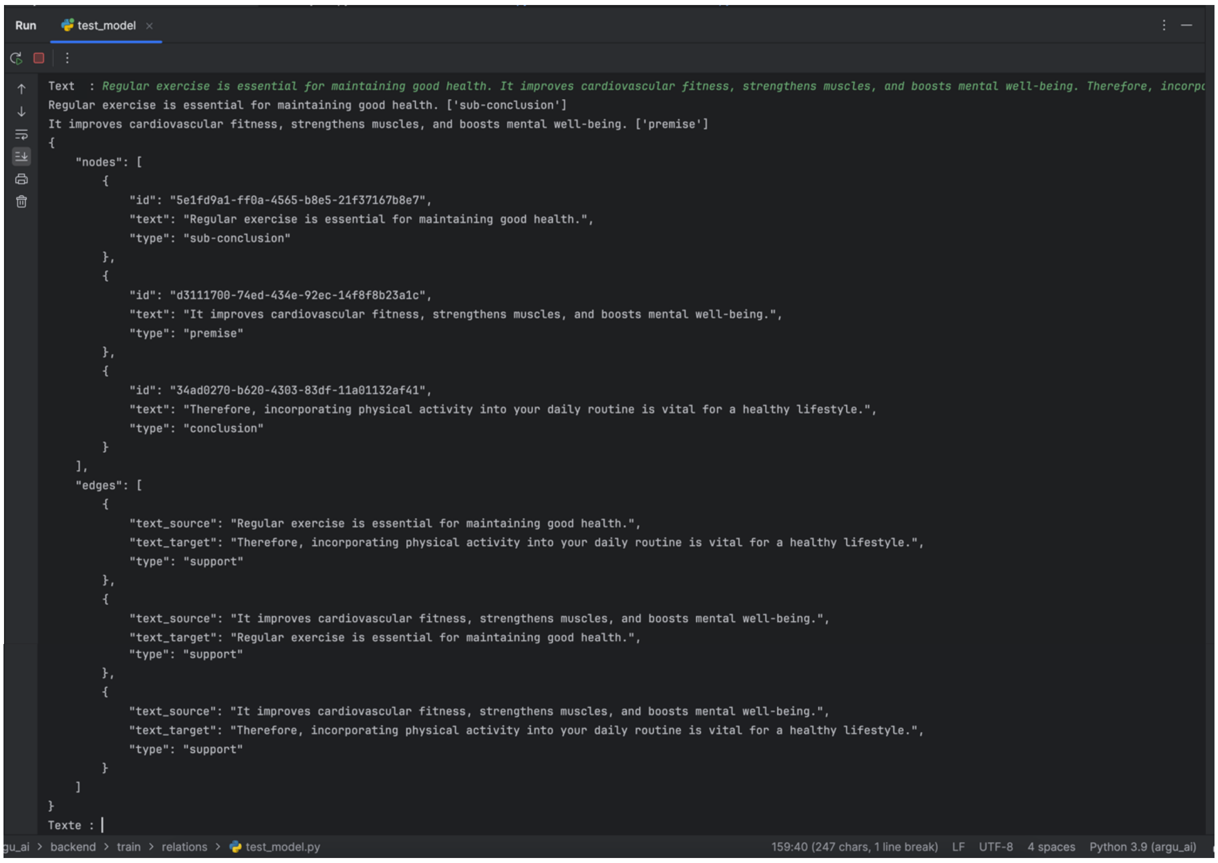
Another major challenge was optimising the model's performance. Finding the right hyperparameters to maximise model performance was not an easy task. To overcome this difficulty, I used the grid search technique (GridSearchCV) to test different combinations of hyperparameters and identify the optimal values. This approach allowed me to systematically explore and evaluate different configurations to improve the accuracy of the model.

Finally, I encountered difficulties related to the imbalance of classes in the dataset, which could bias the model's predictions. To deal with this, I used the Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic examples of minority classes, thus balancing the class distribution. By combining these techniques and carrying out numerous tests, I was able to maximise the performance of my model and obtain the results I wanted for my project.

VI.3.3 Relationship model

After developing and optimising the models for segmenting and classifying text segments, I set about creating the relations model to identify the connections between the different parts of the argumentative text. This section presents the results obtained by this model, which plays a crucial role in the complete and coherent analysis of arguments. Let's change the test sentence again and take the following one:

"Regular exercise is essential for maintaining good health. It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being. Therefore, incorporating physical activity into your daily routine is vital for a healthy lifestyle."


Run
test_model
gu_ai > backend > train > relations > test_model.py
159:40 (247 chars, 1 line break) LF UTF-8 4 spaces Python 3.9 (argu_ai)

```
Text  : Regular exercise is essential for maintaining good health. It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being. Therefore, incorpo
Regular exercise is essential for maintaining good health. ['sub-conclusion']
It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being. ['premise']
{
    "nodes": [
        {
            "id": "5e1fd9a1-ff0a-4565-b8e5-21f37167b8e7",
            "text": "Regular exercise is essential for maintaining good health.",
            "type": "sub-conclusion"
        },
        {
            "id": "d3111700-74ed-434e-92ec-14f8f8b23a1c",
            "text": "It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being.",
            "type": "premise"
        },
        {
            "id": "34ad0270-b620-4303-83df-11a01132af41",
            "text": "Therefore, incorporating physical activity into your daily routine is vital for a healthy lifestyle.",
            "type": "conclusion"
        }
    ],
    "edges": [
        {
            "text_source": "Regular exercise is essential for maintaining good health.",
            "text_target": "Therefore, incorporating physical activity into your daily routine is vital for a healthy lifestyle.",
            "type": "support"
        },
        {
            "text_source": "It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being.",
            "text_target": "Regular exercise is essential for maintaining good health.",
            "type": "support"
        },
        {
            "text_source": "It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being.",
            "text_target": "Therefore, incorporating physical activity into your daily routine is vital for a healthy lifestyle.",
            "type": "support"
        }
    ]
}
Texte : 
```

Figure 27 - Relationship model results

Here we have a combined result with the classification model (nodes part) and the classification model (edges part), creating the final JSON of the argumentative analysis of the text. In the 'Edges' part of the JSON, we can see that the relationships have been realised thanks to the model: the sub-conclusion supports the conclusion, the premise supports the sub-conclusion and the conclusion.

To obtain these results, I followed the same four steps as for the previous model: data recovery, pre-processing of the textual data, preparation of the training data and training of the model. This time, however, I used the relationship data between the segments (the 'edges') of the dataset, because the aim was to train the model to identify the types of relationship between the text segments.

So, I used the ***DistilBERT***[5] model, a ***transformer***[6] model pre-trained on BERT (Bidirectional Encoder Representations from Transformers).

This is how the relationship model works. Let's start with the previous example:

"Regular exercise is essential for maintaining good health. It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being. Therefore, incorporating physical activity into your daily routine is vital for a healthy lifestyle."

Here, the segmentation and classification models are used to obtain the identity segments, i.e.:

- *"Regular exercise is essential for maintaining good health"* **– sub-conclusion**
- *"It improves cardiovascular fitness, strengthens muscles, and boosts mental well-being"* **– premise**
- *"Therefore, incorporating physical activity into your daily routine is vital for a healthy lifestyle"* **– conclusion**

The relationship model will then create all the possible pairs, in our case we will concentrate on just one:

"sub-conclusion: regular exercise is essential for maintaining good health [SEP] premise: it improves cardiovascular fitness strengthens muscles boosts mental well-being"

1. Tokenisation

The pair is tokenised, which gives us the following:

["sub-conclusion:", "regular", "exercise", "is", ..., "[SEP]", "premise:", "it", "improves", ...]

2. Position encoding

Each token is assigned a position in the sequence to maintain word order:

[(token1, pos1), (token2, pos2), ..., (tokenN, posN)]

Once the data has been tokenised and encoded, it is submitted to the transform layers, which will identify the textual relationships. This is the main part of the model. Each transform layer contains 4 key stages.

3. Attention multi-Head

Attention allows a model to focus on different parts of an input sequence (for example, a text) to better understand the context of each word. In a text, the meaning of one word may depend on several other words in the sentence. Attention helps to capture these dependencies.

Multi-headed attention enhances this concept by allowing the model to focus simultaneously on different parts of the sequence with several attention "heads".

An attention head is a mechanism in natural language processing models that allows the model to focus on different parts of a word sequence at the same time.

Each attention head can learn different relationships and features between words. For example, one head may focus on grammatical relationships, while another may focus on semantic relationships.

Let's take the sentence "The black cat sleeps on the sofa" to illustrate how attention heads work:

- **Head 1 (syntactic relation):** Can focus on the link between "cat" and "sleep" (subject-verb).
- **Head 2 (syntactic relation):** Can focus on the link between "cat" and "black" (noun-adjective).
- **Head 3 (syntactic relation):** Can focus on the link between "sleep" and "on" (verb-preposition).

By dividing attention into several heads, the model can perform several calculations in parallel, increasing its efficiency.

The end result of this step is a set of vectors enriched by the contextual information captured by multi-head attention.

4. Add & Norm (Addition and Normalisation)

The vectors resulting from multi-head attention are added to the initial input vectors and normalised. This helps to retain the original information while stabilising model learning.

5. Feed-Forward Layer

Each vector passes through a layer that applies additional transformations to capture more complex relationships between words. This step further enriches the understanding of the model.

6. Add & Norm (Addition and Normalisation)

A second addition and normalisation step are applied to stabilise the new representations after the transformation.

The steps described above (Multi-headed Attention, Add & Norm, Feed-Forward, Add & Norm) make up a single Transformer layer. In a typical Transformer model, several such layers are stacked. Each layer takes the output of the previous layer as input, thus improving the word representations at each stage.

7. Classification

The final vectors, which are contextual representations of the segments, are passed to a classification layer. This layer determines whether the segments are related (1) or not (0). The output of the model is therefore binary.

For our example pair, the model produces a prediction indicating whether the segment pair is in a relationship. This prediction is based on the rich context representations generated by the attention and transformation stages of the Transformer model.

The difficulty wasn't as great as for the classification model because I already knew how to retrieve the data for training from previous experience. However, there were several specific difficulties to overcome for the relationship model. The first major difficulty was to find the appropriate model for identifying relationships between text segments. I therefore explored several model architectures before choosing DistilBERT for its combination of efficiency and accuracy.

To achieve this, I carried out extensive research into natural language processing (NLP) models suitable for relation classification tasks. DistilBERT proved to be an optimal choice due to its lightweight and BERT-like performance.

Once the DistilBERT model had been selected, the next challenge was to gain an in-depth understanding of how it worked and to implement it correctly for my specific purpose. I therefore had to do a lot of research, which paid off in the end, thanks in particular to Hugging Face's official documentation on DistilBERT and the associated tutorials.

VII. CONCLUSION

This project enabled me to develop AI models to automate argumentative analysis, a complex process that required several stages of development and optimisation. From a critical point of view, the work carried out demonstrated the importance of the interaction between theoretical research and technical implementation. For example, the selection and adaptation of pre-trained models such as DistilBERT showed how advances in natural language processing can be applied to meet specific needs.

The main difficulty encountered during this project concerned the creation and implementation of the AI models. The first major difficulty related to the creation of a quality dataset. Although I had access to the Monkeypuzzle database, it was not properly annotated and contained information that was unnecessary for my purposes. To overcome this problem, I used a semi-manual method by integrating the OpenAI API to facilitate the annotation process. This method involved writing precise prompts to direct the API towards correct annotations, followed by manual checking to ensure accuracy. Next, setting up the segmentation and classification models involved adapting pre-trained models such as spaCy or creating a model using RandomForestClassifier. For the segmentation model, I had to develop a custom function to improve the accuracy of the segments generated. This involved several cycles of testing and adjustment to obtain text segments that were relevant for argumentative analysis. For the classification and relations model, the main difficulty was to understand and implement the models for the task in question. I had to adapt the training and evaluation scripts and configure the optimal hyperparameters to maximise the model's performance.

On a personal level, one of the challenges has been managing the stress and high workload. The complexity of the technical tasks and my inexperience in the field of artificial intelligence have sometimes been a source of pressure. However, I didn't let this pressure overwhelm me and I kept moving forward to reach my final goal, in particular by doing a lot of research in this field.

I believe that my work has contributed to the university's research on the topic of argumentative analysis by automating the process. By developing AI models capable of segmenting, classifying and identifying relationships between text segments, I have not only improved the efficiency of argumentative analysis but also provided a solid basis for future research and applications. In addition, the tools and models developed can be integrated into various projects and initiatives at the university, offering advanced AI-based solutions for text analysis.

The experience of this project has enabled me to acquire technical skills in technologies that I wasn't familiar with before. I learnt to use React for front-end development, Python and Flask for back-end development, and I deepened my knowledge of AI by working with advanced models such as RandomForestClassifier and DistilBERT. Through this project, I've also improved my working methods, particularly in terms of organisation. Thanks to the analysis and design phases I acquired during my university studies, I was able to organise the project over the long term. This knowledge helped me enormously and reduced my stress levels.

Although the project has achieved its initial objectives, there is still room for improvement, particularly for the AI models used in argumentative analysis. Indeed, one of the main areas for improvement concerns optimising the models to make them more accurate and robust. AI models can always be refined to achieve better performance. For example, it would be beneficial to explore other model architectures or further refine hyperparameters to optimise the accuracy and robustness of predictions. This could also involve improving the dataset, by diversifying it further in terms of quality and quantity. This would enable the various models to generalise better and handle a greater variety of cases.

This project has enabled me to strengthen my choice of areas for the future. In fact, the two areas that most appeal to me in IT are web development and artificial intelligence. Web development has always been an area of interest to me because of its dynamic, creative nature and its direct impact on the user experience. AI is another area that has always fascinated me because of its potential to automate all the unpleasant tasks of everyday life, but also because of its enormous potential for evolution and its future impact on society. This project was a unique opportunity to work on these two aspects simultaneously, which I thoroughly enjoyed and which has reinforced my choices for a future professional career.

VIII. BIBLIOGRAPHY & SITOGRAPHY

University information

- https://www.rgu.ac.uk/
- https://en.wikipedia.org/wiki/Robert_Gordon_University

Documentation

- https://legacy.reactjs.org/docs/getting-started.html
- https://flask.palletsprojects.com/en/3.0.x/
- https://docs.python.org/3/index.html
- https://platform.openai.com/docs/introduction
- https://stackoverflow.com/

AI Research

- https://medium.com/analytics-vidhya/a-super-simple-explanation-to-random-forest-classifier-d73c9a3307ee
- https://huggingface.co/docs/transformers/model_doc/distilbert
- https://www.youtube.com/watch?v=SZorAJ4I-sA&t=377s
- https://www.youtube.com/watch?v=v6VJ2RO66Ag
- https://paperswithcode.com/
- https://news.microsoft.com/source/topics/ai/

Lexicon

- https://www.wikipedia.org/
- https://www.google.co.uk/

IX. LEXICON

***NLP (Natural Language Processing)*[1]**

Natural Language Processing (NLP) is a branch of artificial intelligence that focuses on the interaction between computers and human languages. The aim of NLP is to enable computers to understand, interpret and respond intelligently to natural language input. Applications of NLP include machine translation, speech recognition, sentiment analysis, and much more.

TF-IDF Vectorisation[2]

TF-IDF (Term Frequency-Inverse Document Frequency) vectorisation is a technique used in NLP to evaluate the importance of a word in a document in relation to a corpus of documents. It consists of two parts:

1. TF (Term Frequency): measures the frequency with which a term appears in a document. The more frequently a word appears, the higher its TF.
2. **IDF (Inverse Document Frequency):** measures the importance of the term in the corpus as a whole. Common words in the corpus will have a low IDF value, and rare words will have a high IDF value.

***SMOTE (Synthetic Minority Over-sampling Technique)*[3]**

SMOTE is a technique used to balance unbalanced datasets by artificially increasing the number of examples in minority classes. Rather than simply duplicating existing instances of the minority class, SMOTE creates new synthetic examples by interpolating between existing examples. This technique helps to improve the performance of machine learning algorithms that can be biased towards majority classes.

***GridSearchCV*[4]**

GridSearchCV is a cross-validation method used to optimise the hyperparameters of a machine learning model. It works by systematically exploring a grid of values for the specified hyperparameters and evaluating the performance of the model for each possible combination of these values. GridSearchCV uses cross-validation to ensure that the results obtained are generalizable and helps to find the optimal configuration of hyperparameters for a given model.

***DISTILBERT*[5]**

BERT is a natural language processing model developed by Google. It is distinguished by its ability to read text in both directions, i.e. from left to right and from right to left at the same time, enabling it to better understand the context of words. During his pre-training, BERT hides certain words in a sentence and tries to guess them based on the surrounding words. He also learns to predict whether a sentence logically follows another. This dual approach helps BERT to better understand the relationships between words and sentences. Once pre-trained, BERT can be tuned for specific tasks such as text classification or question answering, delivering impressive performance on a variety of natural language processing applications.

DistilBERT is a simplified, lighter version of BERT, designed to be faster and less resource intensive. It is around 60% smaller than BERT and twice as fast, while retaining around 97% of BERT's performance. To create DistilBERT, the researchers used a technique called knowledge distillation, where a smaller model is trained to reproduce the behaviour of a larger, more complex model, in this case BERT. DistilBERT retains the advantages of BERT, while being better suited to applications in real time and on devices with limited resources. It is particularly useful for systems requiring a rapid response, such as chatbots and voice assistants.

***Transformer Model*[6]**

The Transformer model is a neural network architecture introduced in the 2017 paper "Attention Is All You Need" by Vaswani et al. Transformers are designed to handle sequences of data, such as sentences, using an attention mechanism that allows the model to focus on different parts of the sequence for each prediction. This approach has revolutionised NLP by dramatically improving performance on a variety of tasks, including machine translation, language understanding and text generation. Transformers form the basis of many advanced NLP models, such as BERT, GPT and their derivatives.

X. APPENDIX

Dataset structure

```
{
  "nodes": [
    {
      "id": "a934fa39-6f22-4ce8-93c7-070194dcafe3",
      "text": "Fall is the best time to visit America's great cities, beaches, and
mountains",
      "type": "sub-conclusion"
    },
    {
      "id": "781744be-3b46-4736-b25f-0601a3a9f7f2",
      "text": "The foliage is breath-taking",
      "type": "premise"
    },
    {
      "id": "5eeb6a30-eb7a-459a-a16c-e597ec7eab98",
      "text": "the weather is cooler",
      "type": "premise"
    },
    {
      "id": "9a564cf1-3f37-4b00-bf25-d67dd69e773a",
      "text": "the crowds are gone",
      "type": "premise"
    },
    {
      "id": "9a664cf1-3f37-4b00-bf25-d67dd69e773a",
      "text": "you can really relax and enjoy yourself",
      "type": "conclusion"
    }
  ],
  "edges": [
    {
      "text_source": "The foliage is breath-taking",
      "text_target": "Fall is the best time to visit America's great cities,
beaches, and mountains",
      "type": "support"
    },
    {
      "text_source": "the weather is cooler",
      "text_target": "Fall is the best time to visit America's great cities,
beaches, and mountains",
      "type": "support"
    },
    {
      "text_source": "the crowds are gone",
      "text_target": "Fall is the best time to visit America's great cities,
beaches, and mountains",
      "type": "support"
    },
    {
      "text_source": "Fall is the best time to visit America's great cities,
beaches, and mountains",
      "text_target": "you can really relax and enjoy yourself",
      "type": "support"
    }
  ]
}
```

Segmentation model key steps

1. Tokenisation

Separation of text into basic units (tokens), such as words and punctuation marks

Results:

[The] [RGU] [School] [of] [Computing] [is] [a] [great] [place] [to] [study] [.]

2. Part-of-speech tagging

Assign grammatical categories (nouns, verbs, etc.) to each token

Results:

[The/DET] [RGU/PROPN] [School/NOUN] [of/ADP] [Computing/PROPN] [is/VERB] [a/DET] [great/ADJ] [place/NOUN] [to/PART] [study/VERB] [./PUNCT]

3. Dependency Parsing

Identification of syntactic relationships between tokens (for example, subject-verb-object)

Results:

[The/DET --mod--> School/NOUN] [RGU/PROPN --compound--> School/NOUN] [School/NOUN --nsubj--> is/VERB] [is/VERB --attr--> place/NOUN] [place/NOUN --prep--> to/PART] [to/PART --pcomp--> study/VERB]

4. Named Entity Recognition (NER)

Identification of syntactic relationships between tokens (for example, subject-verb-object)

Results:

[RGU/ORG] [School of Computing/ORG]

This segmentation and syntactic analysis transform the text into a rich, detailed structure, making it easier to process later. The key point here is that, by going through these stages, the model acquires an in-depth understanding of the grammatical and syntactic structure of the text. This understanding enables the text to be segmented in a logical and coherent way.

Segmentation function

```
def segment_text(doc):
    segments = []
    start = 0
    include_next = False

    continuation_keywords = ['so', 'therefore', 'thus', 'then',
'consequently', 'hence', 'accordingly', 'as a result', 'because', 'as
such', 'henceforth', 'henceforward', 'subsequently']

    for token in doc:
        if token.text in ('.', '?', '!', ';') or token.is_space:
            segments.append(doc[start:token.i + 1].text.strip())
            start = token.i + 1
            include_next = False
        elif token.text in continuation_keywords:
            segments.append(doc[start:token.i].text.strip())
            start = token.i
            include_next = True
        elif token.text == ',' and include_next:
            include_next = False

    if start < len(doc):
        segments.append(doc[start:].text.strip())

    return segments
```

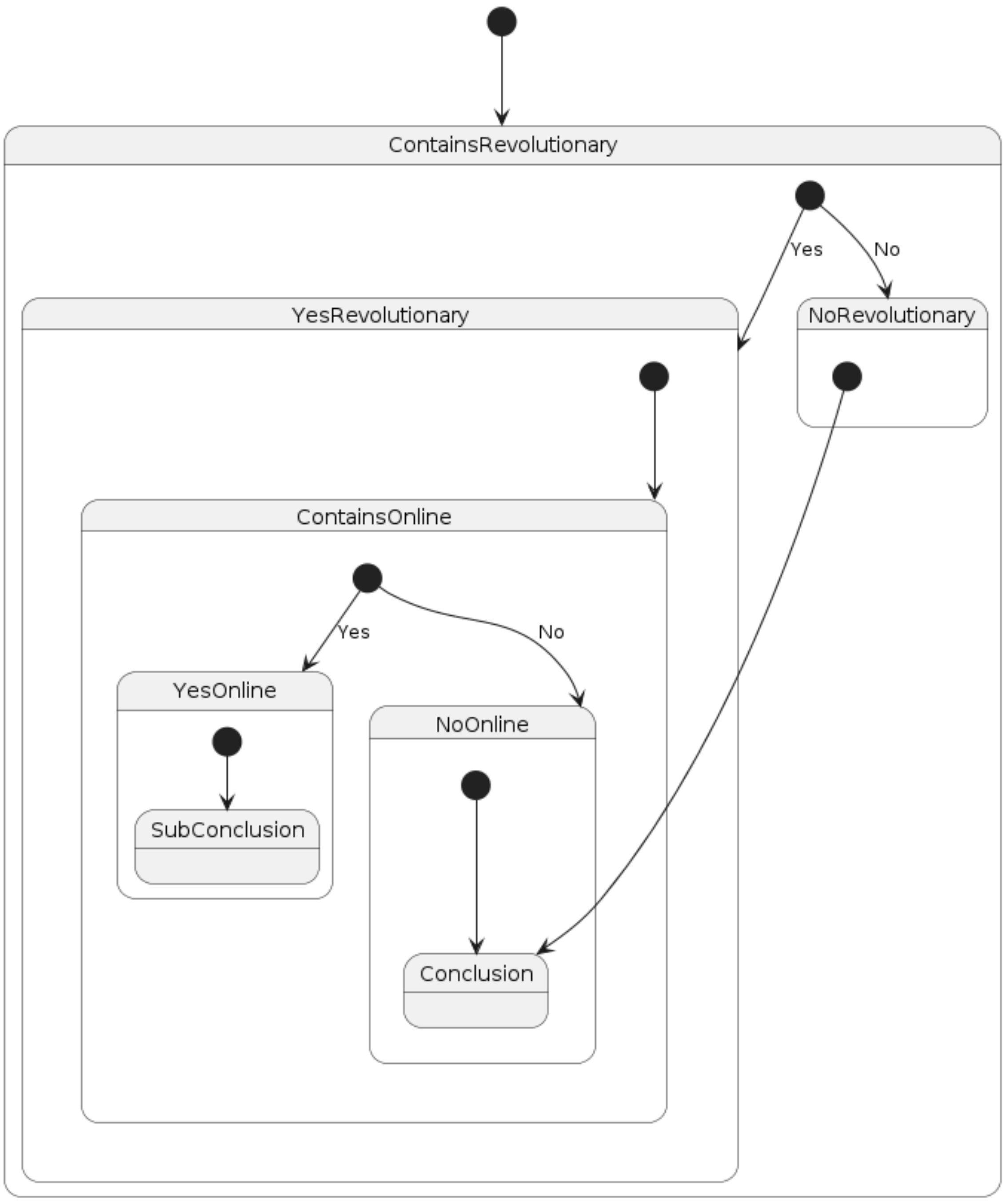


Figure 28 - Decision tree bootstrapped dataset 1

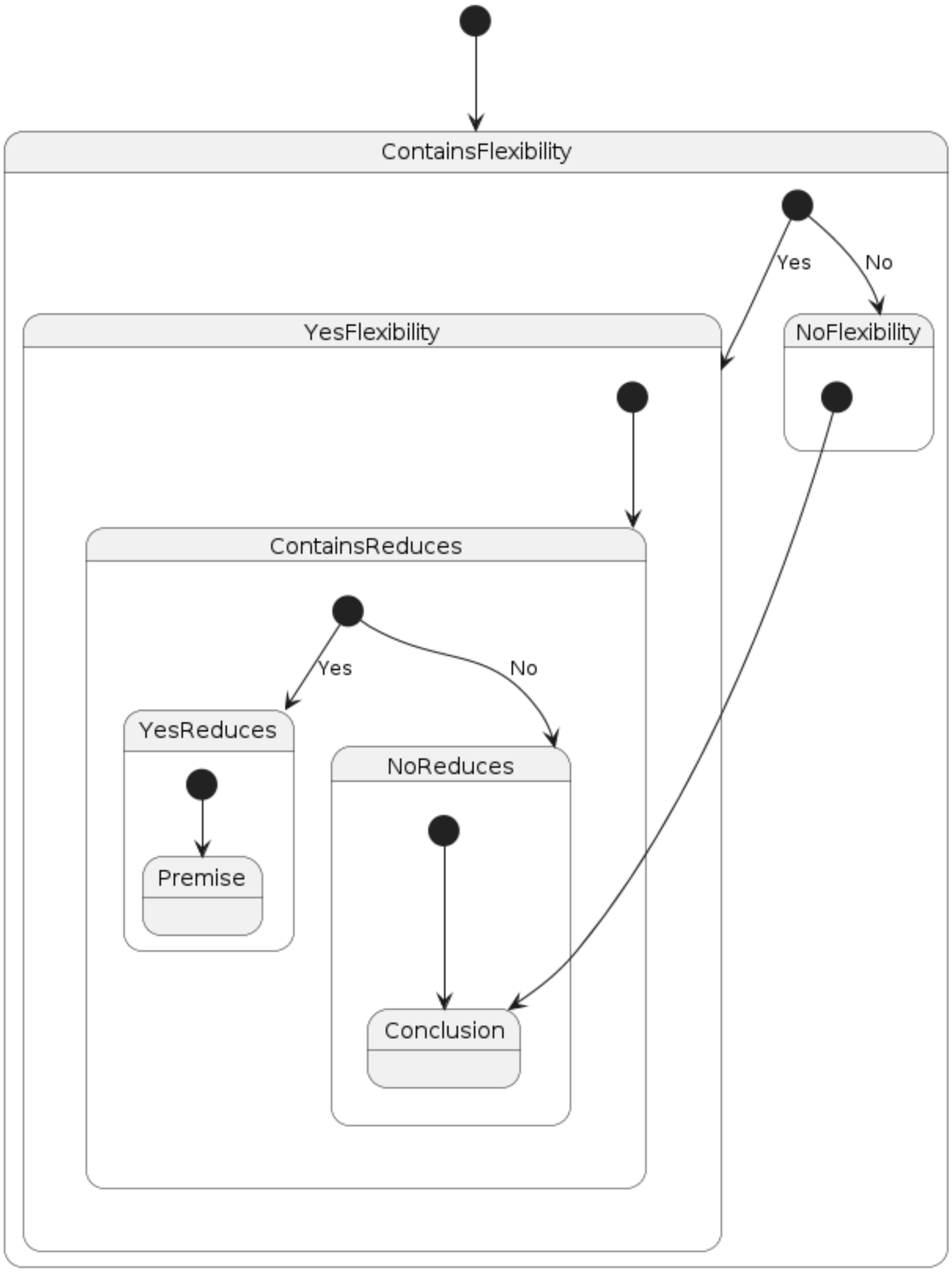


Figure 29 - Decision tree bootstrapped dataset 2

XI. SUMMARY

Résumé

Ce rapport décrit un stage effectué à l'Université Robert Gordon en Écosse, qui portait sur le développement d'un site web utilisant l'intelligence artificielle pour l'analyse argumentative automatisée des textes. Le projet visait à créer une application web capable de segmenter, classifier et analyser les arguments présents dans un texte. Le rapport détaille les différentes étapes du projet, y compris la conception du site, le développement des API et de la base de données, la création d'un ensemble de données, et le développement des modèles d'IA.

Summary

The report outlines an internship at Robert Gordon University in Scotland, focusing on developing a website powered by artificial intelligence for automated argumentative text analysis. The project aimed to build a web application capable of segmenting, classifying, and analysing arguments within texts. The document details the project's stages, including site design, API and database development, dataset creation, and AI model development. This project facilitated the acquisition of skills in web development and AI and contributed to research in argumentative analysis.

Key-words

AI – Argumentative analysis – Web development – DistilBert – RandomForestClassifier – Automation