

KILIAN PEYRON

INTERSHIP REPORT



WEB & AI
DEVELOPMENT

APRIL 13 TO JUNE
21, 2024



Robert Gordon University,
Garthdee House, Garthdee Road,
Aberdeen, AB10 7QB,
Scotland, UK

Training supervisor RGU :
Pedram Salimi

University supervisor :
Anne Daumas



Lyon 1

71 rue Peter Fink
01000 BOURG-EN-BRESSE
Phone: 04 74 45 50 65
Fax: 04 74 45 50 51
E-mail: iutbourg.info@univ-lyon1.fr

Table of Contents

Table of Contents	2
1. Acknowledgements	4
2. Introduction.....	5
3. Presentation of the University.....	6
3.1 The University	6
3.1.1 Informations	6
3.1.2 Brief History and Key Developments.....	6
3.1.3 Revenue and Market Positioning.....	6
3.1.4 Company Purpose.....	6
3.2 Presentation of the Department.....	7
3.2.1 Work Environment	7
3.2.2 Internship Organization.....	8
4. Work	9
4.1 Mission Context.....	9
4.1.1 Assigned Tasks.....	9
4.1.2 Internship Theme	10
4.1.3 Internship Challenges	10
4.1.4 Constraints	10
4.1.5 Personal Investigation Scope.....	11
4.1.6 Human Contacts	11
4.2 Preliminary project analysis	12
4.2.1 Definition of terms	12
4.2.2 Competitive Analysis.....	14
4.2.3 Technologies Used.....	16
4.2.4 Libraries Used	17
4.3 Design.....	18
4.3.1 Mockups.....	18
4.3.2 Diagrams.....	19
4.3.3 User Data	21
4.3.4 Schedule.....	23
4.3.5 Steps	24
4.4 Implementation	27
4.4.1 Front-end	27
4.4.2 Back-end.....	38
5. Conclusion.....	44
5.1 Critical Smmary.....	44
5.2 Personnal Assessment.....	44
5.3 Final Conclusion.....	45
6. References.....	46
7. Appendix.....	47
7.1 Appendix 1	47
7.2 Appendix 2	48

7.3 Appendix 3	49
7.4 Appendix 4	49
7.5 Appendix 5	50
7.6 Appendix 6	51
7.7 Appendix 7	51
7.8 Appendix 8	51
7.9 Appendix 9	52
7.10 Appendix 10	52
7.11 Appendix 11	53
7.12 Appendix 12	53
7.13 Appendix 13	53
7.14 Appendix 14	54
7.15 Appendix 15	54
7.16 Appendix 16	55
7.17 Appendix 17	55
8. Summaries.....	56
8.1 French.....	56
8.2 English.....	56
8.3 Keywords.....	56

1. Acknowledgements

I would like to express my sincere gratitude to my supervisor, **Pedram Salimi**, from Robert Gordon University. His guidance and constant support throughout my internship have been very important to me. His expertise in the field of AI has not only enhanced my technical skills but also broadened my perspective on how the AI era we know today works.

Furthermore, I would like to thank the entire faculty of IUT Lyon1 for trusting me with this experience, without whom none of this would have been possible. Finally, I would like to thank **Anne Daumas** for her follow-up and support throughout this internship.

2. Introduction

This internship has been a crucial step in my personal, academic, and professional journey. As a computer science student, this internship represents an opportunity to put into practice the theoretical knowledge acquired during my studies. It also allows me to familiarize myself with the professional world, acquire practical skills, and strengthen my professional network. Personally, this internship is an enriching adventure that allows me to discover a new cultural and professional environment in Scotland while consolidating my career plan as a developer.

I decided to do my internship abroad, specifically in Scotland, for several reasons. Firstly, I wanted to discover a country other than France. Additionally, joining an international academic institution allows me to improve and communicate in English. Robert Gordon University, with its modern facilities and team of researchers, offers an ideal working environment and valuable support for a student intern.

My main mission during this internship is to design and develop a web application. This project includes several stages: needs analysis, user interface design, and implementation of an AI model. The following report will detail each aspect of this internship, starting with the presentation of the university and the internship environment. Then, I will describe the stages of the project development.

3. Presentation of the University

3.1 The University

3.1.1 Informations

- **Name:** Robert Gordon University
- **Address:** Garthdee House, Garthdee Road, Aberdeen, AB10 7QB, Scotland, United Kingdom
- **Geographical Location Choice:** Located in Aberdeen, the third-largest city in Scotland.

3.1.2 Brief History and Key Developments

Robert Gordon University traces its origins back to 1750 with the founding of Robert Gordon's Hospital. In 1992, the institution achieved university status, becoming Robert Gordon University. It has evolved into a modern university, focusing on professional programs and strong industry integration.

3.1.3 Revenue and Market Positioning

RGU is a public university that does not generate revenue in the traditional commercial sense. It positions itself as a university of excellence, recognized for its high employability rates and close ties with industry, particularly in the energy, engineering, and health sectors.

3.1.4 Company Purpose

- **Services Offered:** Undergraduate, graduate, and postgraduate education in various fields such as engineering, health, business, computing, and creative arts.
- **Investments in Key Areas:** RGU invests heavily in research infrastructure, educational technologies, and industrial partnerships to ensure up-to-date and relevant programs for the job market.

3.2 Presentation of the Department

3.2.1 Work Environment

I carried out my mission on the 5th floor, in RGU's IT department, located in the Sir Ian Wood building. This building houses all the computer rooms, with spaces adapted for all types of computing, from robotics to general computing, including rooms equipped with Windows PCs and Macs.



Figure 1 : Photo of our workspace

The teachers offices are located on the 4th floor, also within the IT department, where they conduct their research and hold meetings. It was therefore easy for me to request information or advice from my supervisor..

3.2.2 Internship Organization

During our internship, we did not have fixed hours. The university was open from 8 am to 5 pm, and we were required to complete a total of 7 hours of work per day. This allowed us to be more flexible while respecting these working hours to maintain a regular pace.

4. Work

4.1 Mission Context

4.1.1 Assigned Tasks

1. **Development of the web application using ReactJS for the frontend:**
 - **Technology Choice:** Selection of ReactJS for its speed and efficiency in creating dynamic and responsive user interfaces.
 - **UI Components Creation:** Design and development of reusable UI components such as forms, buttons, and navigation panels.
2. **Use of FastAPI for the backend:**
 - **Server Configuration:** Setup of FastAPI to create a fast and performant backend server.
 - **API Development:** Design and implementation of RESTful APIs to enable communication between the frontend and backend.
3. **Implementation of chatbot features:**
 - **Conversation Scenarios:** Development of conversation scenarios and flows for the chatbot to handle various user interactions.
 - **Model Training:** Training the chatbot model with relevant data to improve the accuracy and relevance of responses.
4. **User data management via Firebase:**
 - **Firebase Configuration:** Setup of Firebase for database management.
 - **Data Security:** Implementation of strict security rules to protect sensitive user data, including data encryption in transit and at rest.
 - **Firebase Authentication:** Use of Firebase Authentication to manage user authentication, ensuring secure and simple user login for the application.
5. **Continuous integration with GitHub for version control and deployment:**
 - **Version Control:** Use of Git and GitHub to manage code versioning, track changes, and collaborate with other developers.

4.1.2 Internship Theme

The main theme of my internship was to develop a web application that allows users to interact with a chatbot to understand why certain decisions, such as loan applications, were denied, and to propose solutions based on factual or counterfactual explanations.

4.1.3 Internship Challenges

The challenges of the internship, which is part of a university research project, involved developing an intuitive and ergonomic user interface, ensuring the performance and security of the application, and providing clear and understandable explanations to users regarding decisions made by the AI. The primary objective was to contribute to university research by demonstrating how a web application can use advanced technologies to improve the interaction between users and AI systems while explaining its decisions.

4.1.4 Constraints

Budget:

- The project has no specific allocated budget. Netlify and Firebase platforms will be used for hosting and are available for free, reducing financial costs to zero.

Timeline and Key Dates:

- Project start: April 13, 2024
- Research and planning phase: April 13, 2024 – April 27, 2024
- Initial development: April 28, 2024 – June 6, 2024
- Testing and adjustments phase: June 7, 2024 - June 14, 2024
- Final project delivery: June 21, 2024

Complex Tasks:

- **Creating the AI Model:**
 - Preprocess the dataset (cleaning, normalization, etc.).
 - Train the model on the dataset.
 - Integrate DICE to have an xAI model.
- **Interpreting the AI Model :**
 - Analyze the results obtained by the model.
 - Integrate the model and results into the API.
 - Modify model results with OpenAI's API for more human-like results.
 - Integrate into the frontend with results display.

4.1.5 Personal Investigation Scope

My personal investigation scope involved in-depth research on the technologies used, including ViteJS, React, FastAPI, and Firebase. I also studied AI concepts, such as natural language processing (NLP) and explainable artificial intelligence (XAI), to understand how to integrate these technologies into our chatbot project.

4.1.6 Human Contacts

I primarily interacted with my internship supervisor, Pedram Salimi, who guided me throughout the project. I also collaborated with other researchers who helped me continually improve the project and discover new insights.

4.2 Preliminary project analysis

4.2.1 Definition of terms

Chatbot : A chatbot is a computer program designed to simulate a conversation with human users, particularly on the Internet. Chatbots use artificial intelligence to understand user requests and respond appropriately. They are often used to provide customer service, help users find information or perform simple tasks.

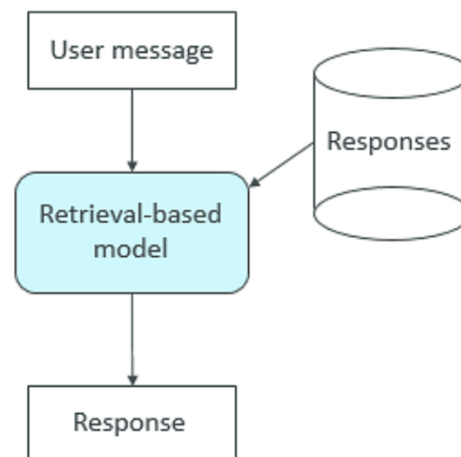


Figure 2 : Basic schematic explaining how a Chatbot works

NLP (Natural Language Processing) : NLP is a branch of AI that focuses on the interaction between computers and humans using natural language. It enables machines to understand, interpret and generate human text.

Here are the different NLP steps for a given query:

- **Tokenization**: The process of dividing text into smaller units, called tokens, which can be words, phrases or symbols.
- **Stemming**: A technique for reducing words to their basic root or form, often by removing common suffixes.
- **Lemmatization**: Like stemming, but returns the word to its basic form, considering its grammatical role in the sentence.
- **Bag of Words (BoW)**: A model that represents text as a collection of words without regard to order or grammatical structure.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: A statistical method for evaluating the importance of a word in a document in relation to a collection of documents or corpus.
- **Word Embedding**: A representation of words as vectors in a multidimensional space, capturing semantics and relationships between words.
- **N-grams**: Continuous sequences of n text elements (words or letters) that are used to predict the next element in a language sequence.

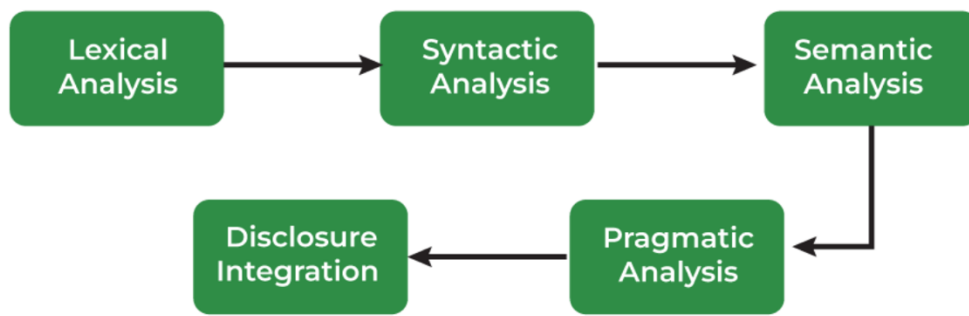


Figure 3 : Schematic explaining the different phases of NLP when a user sends a query.

NLG (Natural Language Generation) : NLG is a sub-domain of NLP that focuses on the production of natural language text from structured data. It enables machines to generate human-understandable text from raw data.

Here are the NLG steps for transforming data into understandable text:

- **Data analysis:** Understand and interpret structured data to determine the content to be generated.
- **Content structuring:** Organize information logically in preparation for text generation.
- **Application of language rules:** Use grammatical and stylistic rules to form correct, natural sentences.
- **Text generation:** Produce the final text that conveys information in a fluid and comprehensible way.
- **Optimization:** Improve the quality of the generated text by fine-tuning grammar, style and content relevance.

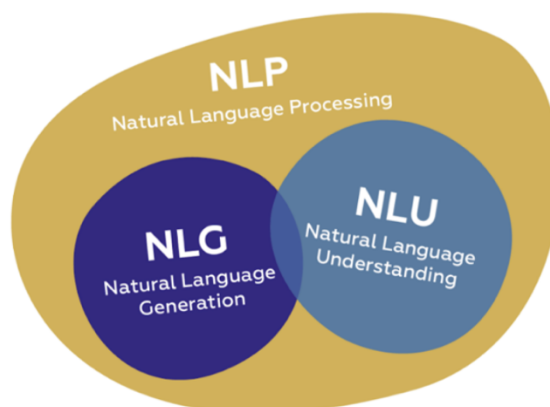


Figure 4 : Schematic showing NLP subtypes, in particular NLG (Natural Language Generation) and NLU (Natural Language Understanding).

XAI (Explainable Artificial Intelligence) : XAI is a field of AI that aims to create transparent and interpretable AI models, so that humans can understand how decisions are made by AI algorithms.

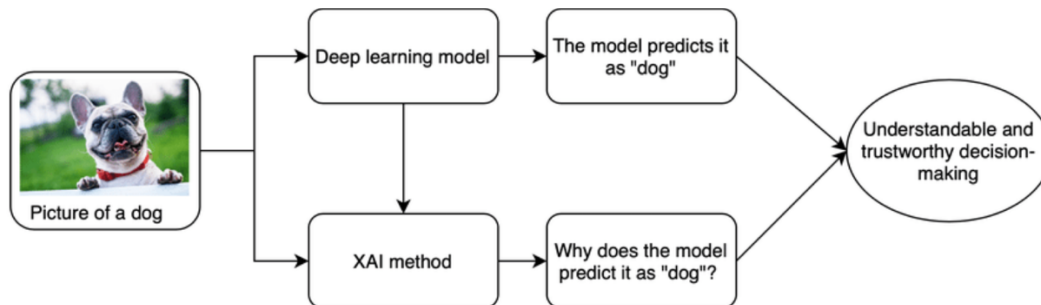


Figure 5 : Detailed schematic showing how Artificial Intelligence Explainability (XAI) informs the decisions made by a system: in this case, it provides a clear, factual explanation of why the image analyzed represents a dog.

LLM (Large Language Models) : LLMs, such as GPT (Generative Pre-trained Transformer), are language models developed from large text datasets. These models are pre-trained on a massive text corpus, enabling them to capture linguistic structures and language patterns. As a result, they can generate text that looks natural and human-like.

4.2.2 Competitive Analysis

The competitive analysis focuses primarily on a similar project, the **iSee project**, in comparison to my project.

Presentation of the iSee Project: The iSee project stands out with its modern interface and visualization-oriented approach to responses. Unlike a traditional chatbot, iSee presents responses in the form of options and images, thereby enhancing the user experience with a visual and support-oriented approach.

Advantages:

- **Modern Interface:** iSee offers an intuitive and contemporary user interface.
- **Visual Responses:** Responses are presented as images, facilitating the visual understanding of information.

Disadvantages:

- **Limited Free-Form Questions:** Users have less flexibility to ask open-ended questions, which may limit the relevance of responses in certain contexts.

Comparison with My Project: Compared to my project, iSee stands out for its visual approach and modern interface. However, my project focuses more on ergonomics and the functionality of saving requests in history. This allows users to resume a previous request at any time, offering a more personalized experience and continuous interaction.

My project aims to combine the user-friendliness of the interface with increased flexibility in user interactions, thus ensuring an enriched and adaptive user experience suitable for various usage scenarios.

4.2.3 Technologies Used

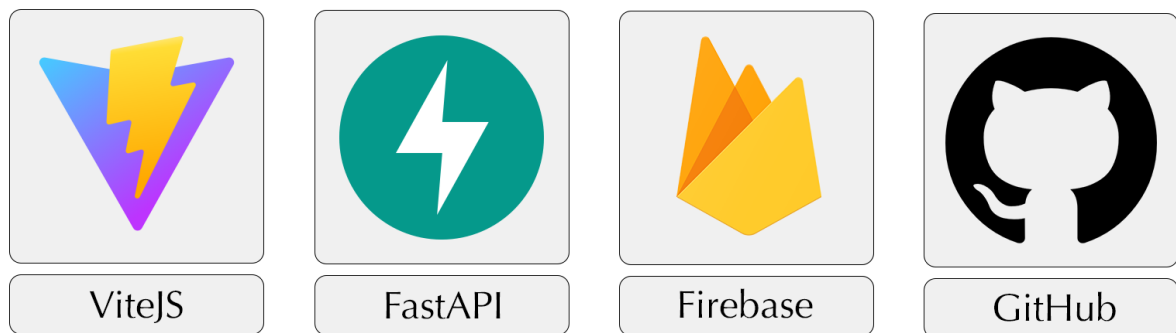


Figure 6 : Technologies Used

Technology	Detail
ViteJS	Used to create the web application in combination with React.
FastAPI	Used for the backend and managing APIs for retrieving responses from the model and processing data.
Firebase	Used for real-time database management and security features.
Github	Used for source code management and continuous integration.

I chose ViteJS coupled with ReactJS for its numerous advantages over React alone. Among these advantages, ViteJS offers an extremely fast startup time due to its bundling and development approach. Additionally, it significantly improves development performance with its ultra-fast hot module replacement. I also wanted to learn React rather than completing the project in a language I already master, like PHP, to broaden my skills and explore new technologies.

For this project, I needed an API framework and opted for Python. Most AI models are trained with Python frameworks, which facilitated integrating my AI model into a Python environment, rather than managing three different technologies for the frontend, backend, and AI model.

Regarding the database, I wanted to explore new options by turning to Firebase, a well-known NoSQL database. Unlike the SQL databases I usually use, Firebase offers major advantages such as the Firebase Authentication plugin and easy integration with Python, simplifying the development of my project.

For version control, I used GitHub. This allowed me to manage different versions of my code in an organized manner and, if desired, make the repository public so others can view or modify it.

By using these technologies, my project benefited from a comprehensive development platform, offering great scalability, performance, and security, as well as flexibility and easy integration with other services.

4.2.4 Libraries Used

Dependency	Description
animate.css	CSS library that provides ready-to-use animations for HTML elements.
axios	Promise-based HTTP client for making HTTP requests from the browser or Node.js.
firebase	Firebase SDK for integrating Firebase services (such as authentication, real-time database, storage, etc.) into a web application.
react-dom	Library for manipulating the DOM with React, necessary for rendering React components in the browser.
react-icons	Library providing a collection of popular icons as React components.
react-router-dom	Library for handling routing in React applications, allowing navigation between different views/pages.
react-router	Core of the React Router library, used with react-router-dom for routing in web applications.
react-toastify	Library for displaying toast notifications in a React application.
react	Core library for building user interfaces with React.
styled-components	Library for writing CSS in JavaScript, enabling component-level styles encapsulation in React.
vite	Fast build and development tool for front-end projects, providing a modern alternative to webpack.

4.3 Design

4.3.1 Mockups

Mockups of the application are essential for visualizing and planning the user interface before starting development.

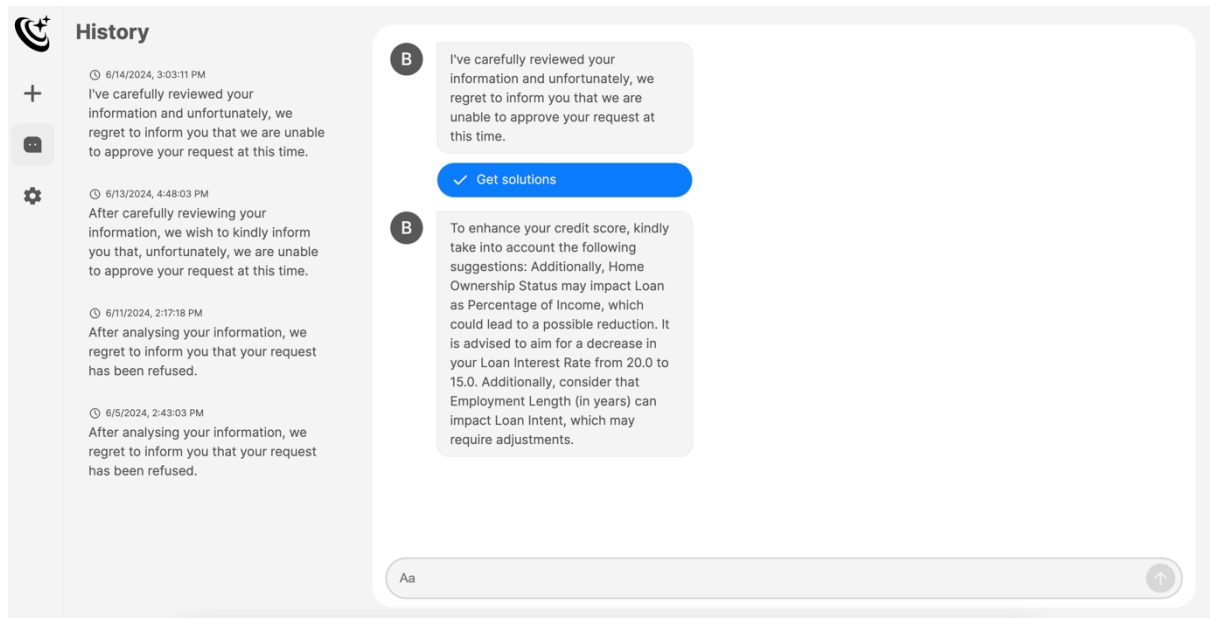


Figure 7 : Main Interface Mockup

My project includes a single main mockup integrating three major components:

- **A top sidebar** for quick access to settings, help, and other sections of the application.
- **A left sidebar** displaying the conversation history.
- **A central area** for the ongoing conversation, where the user can type messages and see the chatbot's responses

4.3.2 Diagrams

Diagrams are used to visually represent the structure and functionality of the application. Here are the main diagrams used:

- Use Case Diagram:

The use case diagram in **Appendix 1** represents the different possible interactions between a user and the project. It is structured into three main sections: User Settings, Virtual Assistant, and Authentication. Each section contains specific use cases detailing the actions available to the user.

User

The user is the main actor in this diagram, represented by a stick figure icon. All actions described in the diagram are performed by the user.

Sections & Use Cases

1. User Settings

- View user settings: This feature allows the user to view and modify personal settings.

2. Virtual Assistant

- View conversation history: Allows the user to view the history of conversations with the virtual assistant.
- Chat with the assistant: This feature allows the user to interact directly with the virtual assistant.

3. Authentication

- Log in with Google: Allows the user to log in to the site using their Google account.
- Log in with Email/Password: This feature allows the user to log in using an email address and password.
- Sign up with Google: Allows the user to create an account on the site using their Google account.
- Sign up with Email/Password: This feature allows the user to create an account using an email address and password.

- Class Diagram:

The class diagram in **Appendix 2** models a chat system with three main entities: User, Conversation, and Message.

- **User:** Contains user information and can have multiple conversations.
- **Conversation:** Associated with a user and contains multiple messages.
- **Message:** Belongs to a conversation and stores the content of the message.

- Sequence Diagrams :

The sequence diagram in **Appendix 3** shows the following project use case scenario.

1. The user submits loan criteria through the user interface.
2. The data is sent to a Python REST API.
3. The Python REST API forwards the criteria to the XAI model.
4. The XAI model checks the conditions and finds them satisfactory.
5. The XAI model returns the accepted result to the Python API.
6. The Python REST API informs the user of the loan acceptance.
7. The user receives congratulations from the system.

The sequence diagram in **Appendix 4** shows the following project use case scenario.

1. The user submits loan criteria through the user interface.
2. The data is sent to a Python REST API.
3. The Python REST API forwards the criteria to the XAI model.
4. The XAI model checks the conditions and finds inconsistencies.
5. The XAI model returns the rejected result to the Python REST API.
6. The Python REST API explains the reason for rejection to the user.
7. The user agrees with the explanation.
8. The loan application process ends.

The sequence diagram in **Appendix 5** shows the following project use case scenario.

1. The user submits loan criteria through the user interface.
2. The data is sent to a Python REST API.
3. The Python REST API forwards the criteria to the XAI model.
4. The XAI model checks the conditions and finds inconsistencies.
5. The XAI model returns the rejected result to the Python REST API.
6. The Python REST API informs the user of the loan rejection due to inconsistencies.
7. The user disagrees with the explanation and asks for a solution.
8. The XAI model proposes a resolution for the inconsistent data.
9. The Python REST API communicates the proposed resolution from the XAI model to the user.

4.3.3 User Data

4.3.3.1 Database

It was crucial to store user data on a secure platform that is easy to use both on the front end and the back end, which is why I chose the excellent Firebase.

Firestore is a NoSQL database from Firebase that stores data as documents within collections. It is designed for real-time synchronization and supports large-scale applications.

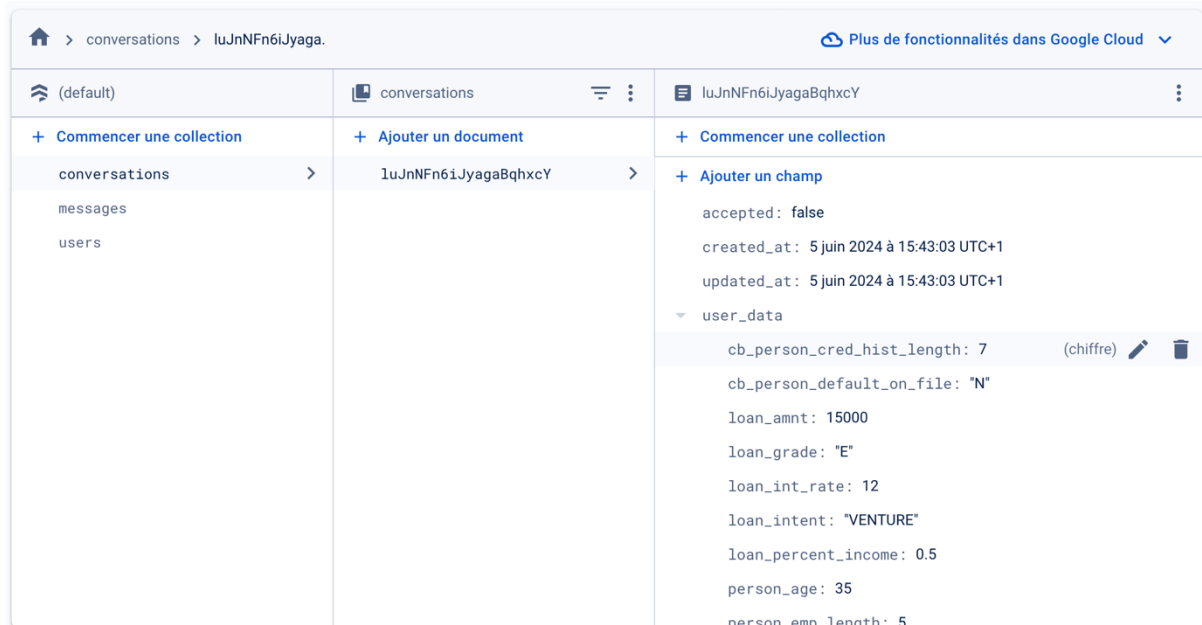


Figure 8 : Data Structure in Firebase

Each piece of data is stored in a document, which is itself stored in a collection. Each document has a unique identifier that differentiates them.

I really liked this approach because there are no predefined fields like in SQL. I decide at the time of insertion what columns I need in each document.

4.3.3.2 Firebase Authentication

As I mentioned earlier, for the authentication part, I used the Firebase Authentication plugin linked to Firebase Database. This allowed me to create a fast and secure authentication system using the React 'firebase' library. Everything was managed on the front end.

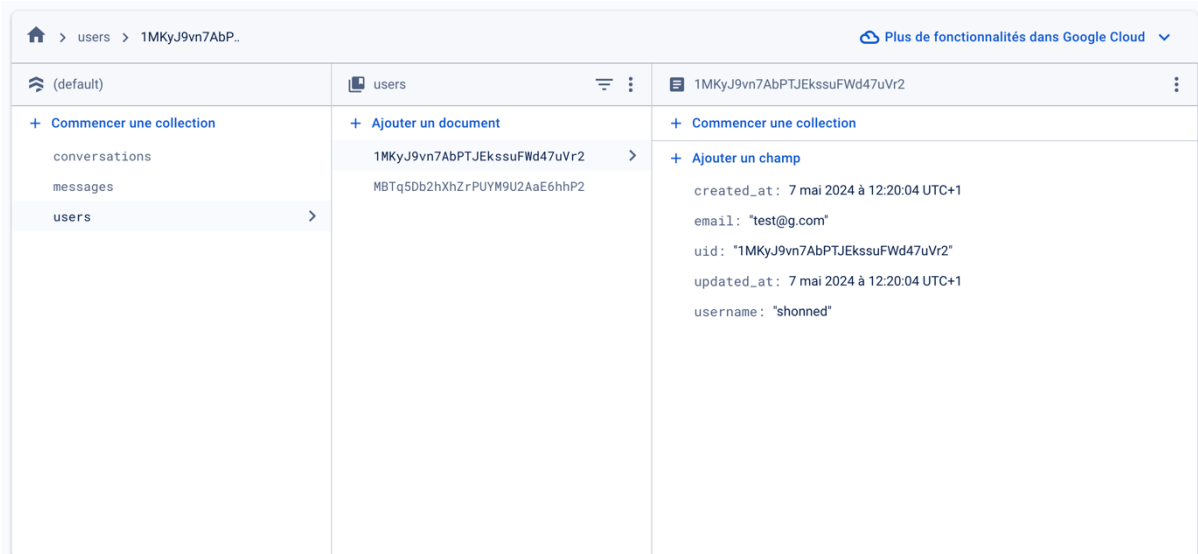


The screenshot shows the Firebase Authentication 'Users' page. At the top, there is a search bar with the placeholder text 'Recherchez par adresse e-mail, numéro de téléphone ou ID utilisateur' and a blue button labeled 'Ajouter un utilisateur'. Below the search bar is a table with the following columns: 'Identifiant', 'Fournisseurs', 'Date de création', 'Dernière connexion', and 'UID utilisateur'. The table contains two rows of user data. At the bottom right, there is a pagination control showing 'Lignes par page : 50' and '1 - 2 of 2'.

Identifiant	Fournisseurs	Date de création ↓	Dernière connexion	UID utilisateur
test@gmail.com		7 mai 2024	4 juin 2024	MBTq5Db2hXhZrPUYM9U2Aa...
test@g.com		7 mai 2024	10 mai 2024	1MKyJ9vn7AbPTJEkssuFWd4...

Figure 9 : Firebase Authentication Plugin Page

Once the user is registered in Firebase Authentication, I only needed to duplicate their information in my 'user' collection in Firebase Database to add fields necessary for the project, such as a username.



The screenshot shows the 'Users' collection in the Firebase Database. The breadcrumb navigation at the top indicates the path: 'users > 1MKyJ9vn7AbP...'. The interface is divided into three main sections. The left section shows a list of collections: 'conversations', 'messages', and 'users' (which is selected). The middle section shows a list of documents in the 'users' collection, with two documents visible: '1MKyJ9vn7AbPTJEkssuFWd47uVr2' and 'MBTq5Db2hXhZrPUYM9U2AaE6hhP2'. The right section shows the details of the selected document '1MKyJ9vn7AbPTJEkssuFWd47uVr2', displaying fields: 'created_at' (7 mai 2024 à 12:20:04 UTC+1), 'email' ('test@g.com'), 'uid' ('1MKyJ9vn7AbPTJEkssuFWd47uVr2'), 'updated_at' (7 mai 2024 à 12:20:04 UTC+1), and 'username' ('shonned').

Collection	Document	Fields
users	1MKyJ9vn7AbPTJEkssuFWd47uVr2	created_at: 7 mai 2024 à 12:20:04 UTC+1 email: "test@g.com" uid: "1MKyJ9vn7AbPTJEkssuFWd47uVr2" updated_at: 7 mai 2024 à 12:20:04 UTC+1 username: "shonned"
	MBTq5Db2hXhZrPUYM9U2AaE6hhP2	

Figure 10 : 'Users' Collection in my Database

4.3.4 Schedule

The table below presents my work schedule throughout the internship.

Week	Task
1	Research into the concepts of the subject
2	Research into the concepts of the subject
3	Preparing the analysis file
4	UI development
5	Back-end intégration with Firebase
6	Application development
7	Application development
8	Implementation of xAI model
9	Writing the internship report and preparing for the oral presentation
10	Writing the internship report and preparing for the oral presentation

4.3.5 Steps

4.3.5.1 Research & Analysis

Objective:

My primary objective was to understand user needs and meticulously analyze existing solutions to ensure that my application would effectively address the problem it aimed to solve.

Tasks:

I began by conducting interviews and surveys with users to gather detailed requirements directly from my target audience. This process provided valuable insights into their preferences, pain points, and expectations.

Analyzing competitor applications was crucial to identify best practices and successful strategies within the industry. This evaluation guided me in adopting proven approaches and potentially innovating beyond existing solutions.

I defined key functionalities to shape the core of my application. By clearly outlining what was essential based on user insights and market analysis, I ensured that my product would meet user expectations while potentially offering unique value propositions.

I developed use cases and user stories to articulate various scenarios and interactions within the application. This structured approach ensured that all potential user interactions were considered and addressed during development.

I created a specifications document covering both technical specifications and functional requirements. This document served as a detailed plan outlining precisely how my application would function and what technological capabilities it needed to effectively meet user needs.

Deliverables:

My deliverables included:

- A Competitive Analysis Report highlighting strengths, weaknesses, and strategies of competitor applications.
- Detailed Functional and Technical Specifications describing the features and technological requirements necessary for the successful development of my application.

By executing these tasks thoroughly and delivering these key documents, my goal was to ensure that my application not only met but exceeded user expectations, thereby establishing a new standard in effectively and innovatively addressing the identified problem.

4.3.5.2 Design & Development

Objective:

My goal was to design and develop the web application, ensuring a seamless user experience and robust functionality.

Tasks:

I started by designing the user interface (UI) and user experience (UX) using wireframes and prototypes to visualize the layout and interactions of the application.

Next, I set up the development environment using ReactJS for the frontend and FastAPI for the backend, establishing a solid foundation for building the application.

Developing the chatbot interface was a critical step, enabling users to interact effectively with the application through automated responses and queries.

I implemented backend logic to manage factual and counterfactual data, ensuring precise processing and retrieval of information.

Integrating the frontend with the backend via API endpoints was crucial to establish communication between the user interface and the core functionalities of the application.

Deliverables:

My deliverables included:

- Wireframes and prototypes showcasing the planned UI and UX elements.
- A fully developed frontend using ReactJS, designed to deliver a responsive and intuitive user experience.
- A fully developed backend powered by FastAPI, equipped with robust functionalities to support the application's operations.
- An integrated and functional chatbot interface, enabling seamless interaction between users and the application.

By executing these tasks rigorously and delivering these key components, my aim was to ensure that the web application not only met but exceeded user expectations, providing a reliable and engaging experience for its users.

4.3.5.3 Deployment

Objective:

My objective was to deploy the web application to a live environment on Netlify, making it accessible to users.

Tasks:

I set up the production environment on Netlify, configuring the necessary settings and infrastructure to host the application.

I deployed both the frontend and backend services to Netlify, ensuring that all components were properly integrated and functioning in the live environment.

Deliverables:

My deliverable was the deployed web application on Netlify, making it accessible and operational for users.

4.4 Implementation

4.4.1 Front-end

4.4.1.1 Authentication

- **Results**

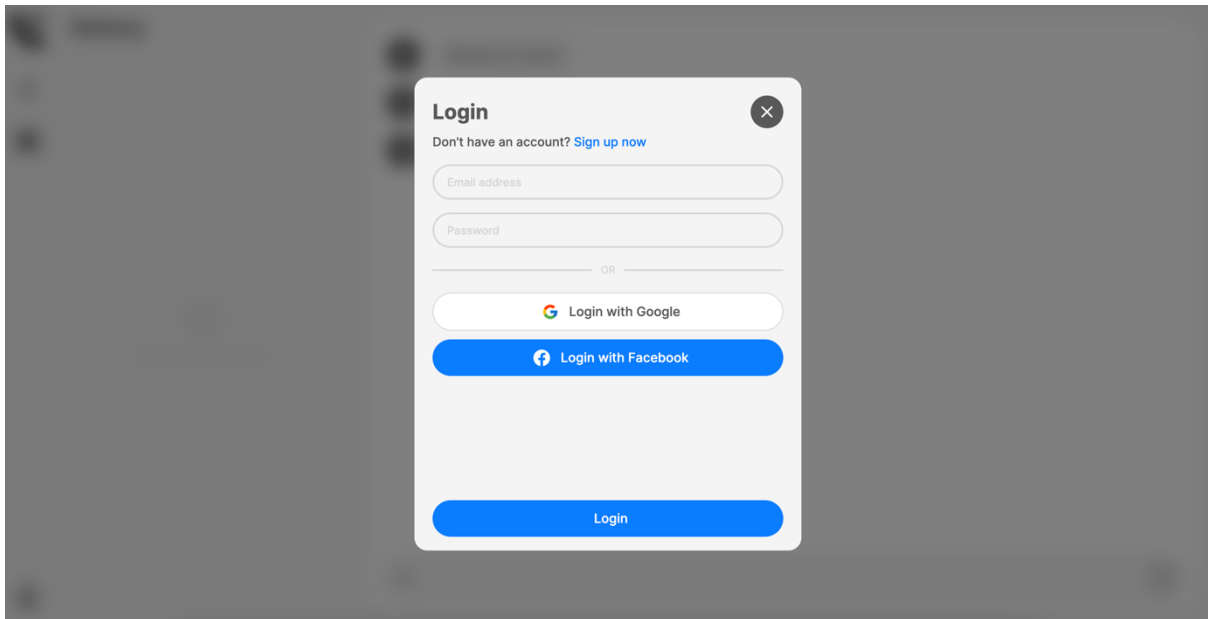


Figure 11 : User Login Interface

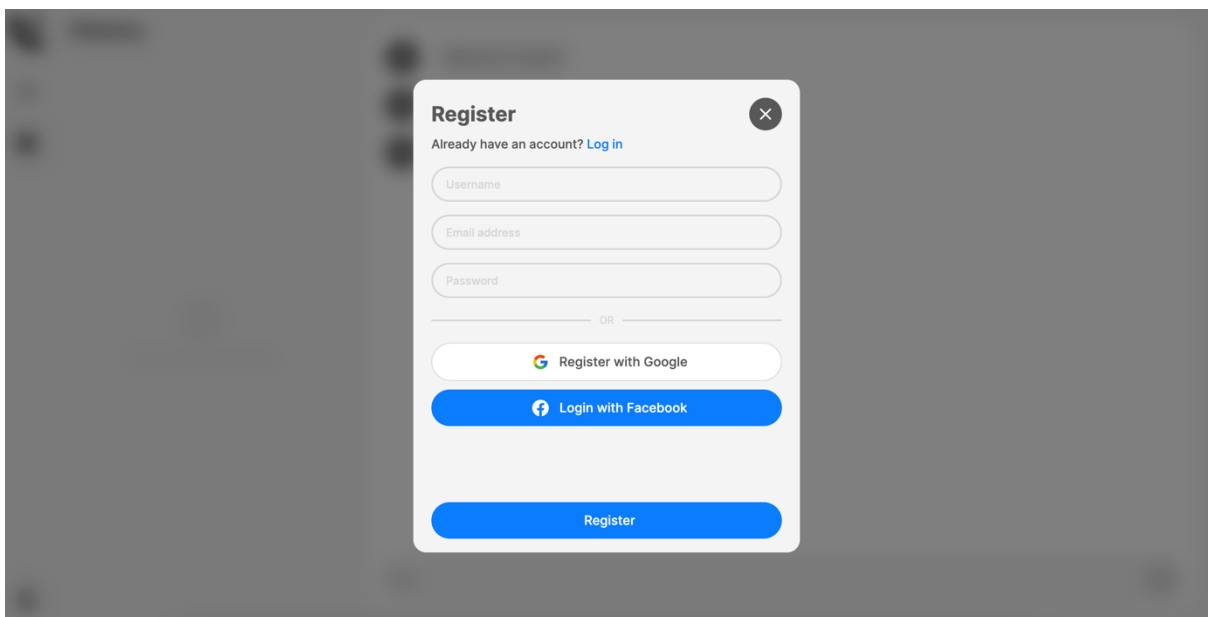


Figure 12 : User Registration Interface

- **Explanations**

Authentication in this project is managed on the front-end using the Firebase library, known for its ease of integration and use. Firebase provides a simple interface to connect external authentication services such as Google, Facebook, and many others.




Practically, this means I could easily add Google and Facebook login and registration methods to the project. These additional options offer greater flexibility to users, allowing them to choose their preferred login method. Moreover, this enhances the user experience (UX) by simplifying the login process, making access to the application smoother and more intuitive.

Using Firebase for authentication offers several advantages. First, it securely handles complex tasks related to authentication, such as storing user credentials, managing sessions, and protecting against unauthorized access. Second, by using Google and Facebook authentication services, users can log in quickly without having to create new accounts, reducing friction and enhancing user satisfaction.

In summary, integrating Firebase for authentication has simplified the development process and allowed for an improved user experience through various secure login methods.

Sélectionner un fournisseur de connexion (étape 1 sur 2)

Fournisseurs natifs

 Adresse e-mail/Mot de passe ✓
 Téléphone
 Anonyme

Autres fournisseurs










 Google ✓	 Facebook ✓	 Play Jeux
 Game Center	 Apple	 GitHub
 Microsoft	 Twitter	 Yahoo!

Figure 13 : List of authentication services available with Firebase Authentication

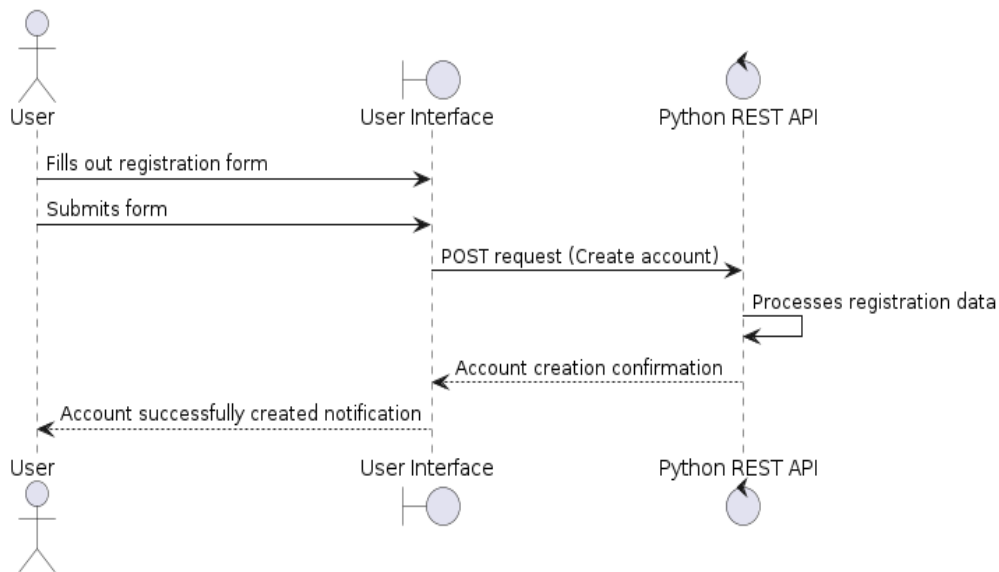


Figure 14 : Class diagram explaining the authentication system in the database

- **Code & Algorithms**

You can find the code for email/password authentication in [Appendix 6](#).

This code includes a '**createUserWithEmailAndPassword**' function, fully managed by Firebase, which registers the user in Firebase Authentication. I only needed to retrieve the user's information from the form and send it to the function.

The rest of the code, starting from setDoc, as previously mentioned, also registers the user in my database to add required information such as a username.

For authentication with external services, it was just a matter of passing the 'provider' to the Firebase '**signInWithPopup**' function, which handled opening the provider's window (see [Appendix 7](#) for Facebook and [Appendix 8](#) for Google).

- **Challenges**

The greatest difficulty I encountered during this project was sending the authentication information to the backend after the user was registered in Firebase Authentication. At that time, I was new to REST APIs, which posed several challenges, particularly regarding the security of data transmission.

One of the main obstacles was managing **CORS (Cross-Origin Resource Sharing)**. CORS is a browser security mechanism that blocks requests from different domains than the one where the application is hosted. In my case, this meant that requests sent from the front-end to the back-end were often blocked, preventing communication between the two parts of the application.

To overcome these difficulties, I had to learn how to properly configure CORS headers on the server to allow requests from my front-end application. Using FastAPI for my backend, I had to modify my API responses to include the necessary CORS headers. This involved adjusting the server settings to specify which origins were allowed to access the resources, what types of requests were permitted, and how to handle cookie permissions and other sensitive information.

4.4.1.2 History

- **Results**

History

🕒 6/14/2024, 3:03:11 PM

I've carefully reviewed your information and unfortunately, we regret to inform you that we are unable to approve your request at this time.

🕒 6/13/2024, 4:48:03 PM

After carefully reviewing your information, we wish to kindly inform you that, unfortunately, we are unable to approve your request at this time.

🕒 6/11/2024, 2:17:18 PM

After analysing your information, we regret to inform you that your request has been refused.

🕒 6/5/2024, 2:43:03 PM

After analysing your information, we regret to inform you that your request has been refused.

Figure 14 : Display of conversations in the history

- **Explanations**

For me, having a conversation history was essential. The goal was to allow a user to track their different requests and make new ones. For example, if a request is denied, the user should be able to return to the application to review the details and know what actions to take to submit a new request. These conversations are displayed in reverse chronological order, meaning that conversations with the latest messages appear at the top of the list. Each conversation shows the last message sent, allowing the user to quickly get an overview of the status of each discussion.

To implement this feature, I used a database to store messages and conversations associated with each user. Each message contains information such as the user ID, message content, and the date and time sent. Conversations are then sorted based on the timestamp of the last message to ensure the most recent ones appear first.

For example, a typical conversation in the 'conversations' collection includes fields such as '**accepted**' (indicating if the request was accepted), 'created_at' and 'updated_at' (timestamps for creation and last update), and 'user_uid' (unique user identifier). Messages associated with this conversation in the 'messages' collection contain fields like 'content' (the content of the message), 'conversation_uid' (conversation identifier), and 'created_at' (message creation timestamp).

In summary, integrating a conversation history was a priority to offer a **smooth and intuitive user experience**. Users can easily track their requests, review necessary instructions in case of a denial, and submit new requests efficiently. This feature, by organizing conversations by the last message sent and displaying a preview of the last message, helps users manage their interactions with the application proactively and organized.

- **Code & Algorithms**

You can find the code for retrieving a user's history in [**Appendix 9**](#).

This section uses the '**navigate**' function from React to ensure a smooth transition between conversations (e.g., /chat/hgRi6LKLkArLfMWppL8E). It allows the page to avoid reloading all components, minimizing user wait time and reducing excessive API requests.

- **Challenges**

The biggest challenge in this part was sorting conversations by the last message sent. The main issue I encountered was that Firebase uses a NoSQL structure, making it impossible to use '**ORDER BY**' naturally. Additionally, Firebase imposes security restrictions on sorting, which required me to create a custom index in Firebase to specify the fields I wanted to sort by. This process led to several complications in the code and especially in the API response. Here's how I resolved these challenges:

1. Creating a Custom Index:

I configured a custom index on Firebase by specifying the '**updated_at**' and '**created_at**' fields to enable sorting. This step was crucial to meet Firebase's security and query performance requirements.



The screenshot shows the 'Indexes' tab in the Firebase console. It displays two custom indexes for the 'conversations' and 'messages' collections. The 'conversations' index is sorted by 'user_uid' (ascending), 'updated_at' (descending), and '__name__' (descending). The 'messages' index is sorted by 'conversation_uid' (ascending), 'created_at' (ascending), and '__name__' (ascending). Both indexes are active and apply to the 'Groupe de collections'.

ID de collection	Champs indexés ?	Champ d'application de la requête	État
conversations	user_uid Croissant updated_at Décroissant __name__ Décroissant	Groupe de collections	Activé
messages	conversation_uid Croissant created_at Croissant __name__ Croissant	Groupe de collections	Activé

Figure 15 : Custom indexes of my database

2. Updating the Data Schema:

For each message sent, I updated the '**updated_at**' field of the corresponding conversation to always reflect the date and time of the last message.

In summary, while managing the sorting of conversations posed significant technical challenges due to Firebase's NoSQL structure and security requirements, creating custom indexes and modifying the API code allowed me to overcome these obstacles.

4.4.1. Creating a conversation

- **Results**

The image shows a chat interface with a light gray background. On the left, a bot (B) asks four questions in gray bubbles. On the right, a user (Y) provides answers in gray bubbles. For the third question, 'Do you own a home?', there are three blue buttons: 'OWN' (selected with a checkmark), 'RENT', and 'MORTGAGE'. At the bottom, there is a text input field with the number '30' and a blue send button with an upward arrow.

Figure 16 : Interface for filling out the form to send information to the model

- **Explanations**

For the most important part, I wanted to have a visually appealing interface. I aimed to adjust the size of the bubbles according to the length of the messages and add an animation when clicking on an option. My goal was to make the user experience (UX) not only functional but also aesthetically pleasing and interactive.

- **Code & Algorithms**

You can find the code for the conversation creation system in [**Appendix 10**](#).

From a technical standpoint, I implemented a system to store all user responses in a data array. This array is then sent to the API to process the data and send the response in a dedicated conversation. Once the user has finished answering the questions, the data array is sent to the API, which processes the data, executes the necessary actions, and sends the response back to the dedicated conversation, allowing the user to see the outcome of their request.

Regarding user experience (UX), I used regular expressions (regex) to filter user responses. For example, if the question is "What is your age?" and the user responds "20 years," the regex extracts "20" and adds it to the array. This system helps avoid errors in the API and therefore for the user.

- **Challenges**

Handling messages posed the primary challenge in this section. It required creating a system capable of processing messages for both user responses and assistant replies. Messages had to be added in the correct sequence while introducing a delay to prevent responses from arriving too quickly after sending a message. This necessitated meticulous management to ensure smooth and natural communication.

Another major challenge was to filter the user's responses to retain only relevant information. As mentioned earlier, it was crucial to extract only the data needed before sending it to the model via the API. To do this, information had to be sorted according to type (int, float, etc.), which required particular attention to ensure the accuracy of the data transmitted. (see **Appendix 11**)

4.4.1.4 Model results

- **Results**

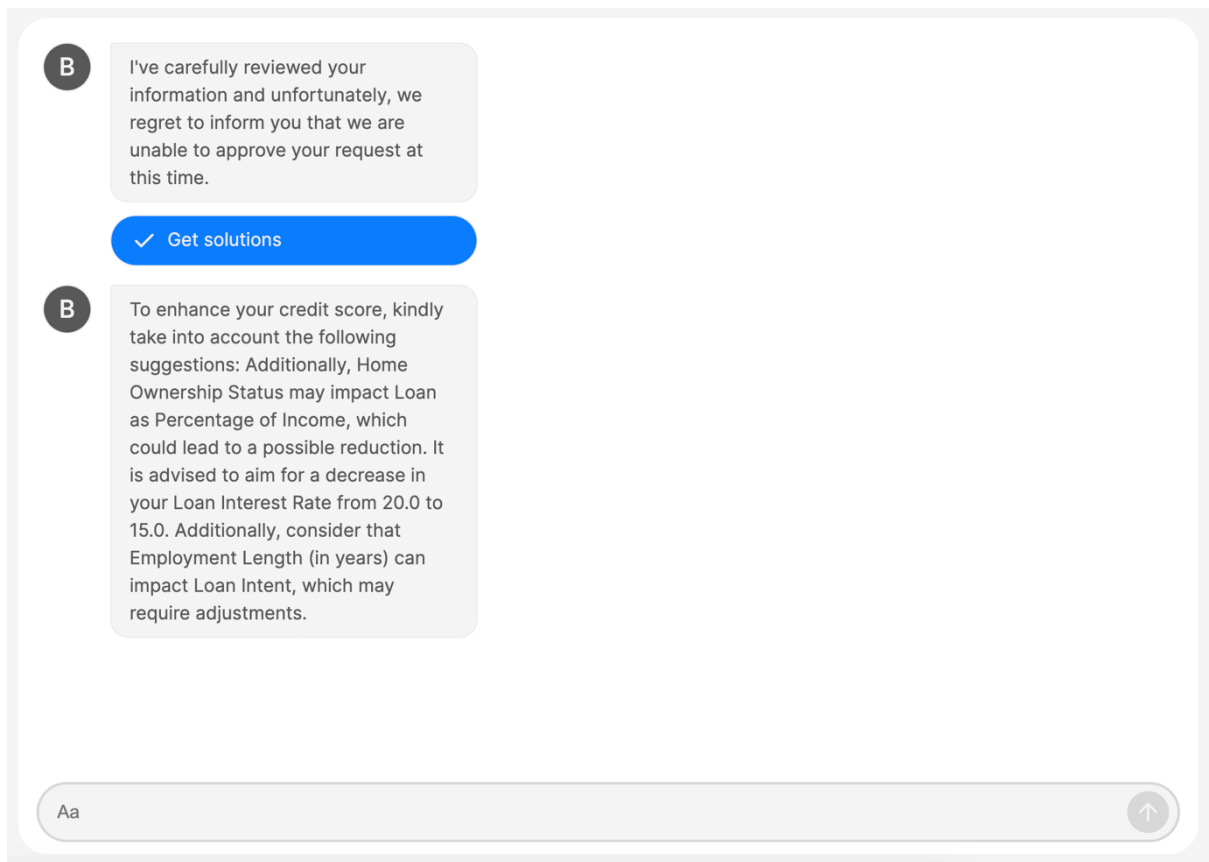


Figure 17 : Interface after a request has been refused, and display of solution(s) for the request to be accepted

- **Explanations**

This section of the application is designed to display the result of the user's request. If the request is accepted, the application notifies the user of this acceptance. On the other hand, if the request is refused, the interface offers solutions to help the user modify the request so that it can be accepted. The aim is to guide the user constructively and proactively, offering options to improve their chances of success.

- **Code & Algorithms**

You can find the code for the conversation creation system in [Appendix 12](#) and [13](#).

This section uses React's '[navigate](#)' function to ensure a smooth transition between the creation of a conversation and the redirection to a specific conversation page (for example, /chat/hgRi6LKLkArLfMWppL8E) where the answer and solutions will be displayed.

To obtain the request information, I first make a call to the API to check the existence of the conversation in the database. If the conversation exists, I then retrieve the associated messages as well as the proposed solutions if the request is refused.

For more details on this part, please refer to the back-end section.

- **Challenges**

The main difficulty lay in securing the system. We had to ensure that every conversation actually existed, and that no essential information was missing that could potentially cause errors.

4.4.2 Back-end

4.4.2.1 AI Model

- Results

```
100%|██████████| 1/1 [00:00<00:00, 10.17it/s]
```

Rejected 1:

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_percent_income	cb_
14668	24	28000		2	6.0	2	1	10000	10.37	0.36

Modifications for being accepted (loan_status = 1):

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_percent_income	cb_
0	24	28000		2	6.0	2	5	10000	19.538281	0.36
1	24	28000		2	6.0	2	4	10000	16.826571	0.36
2	24	28000		3	6.0	2	1	10000	10.370000	0.36
3	24	28000		2	6.0	2	5	1103	10.370000	0.36
4	24	28000		3	71.5	2	1	10000	10.370000	0.36

Figure 18 : Result of a rejected loan and table display of the changes needed to get approved

- Explanations

This section is undeniably the most crucial part of the project. It deals with the trained model, which plays a pivotal role in evaluating an individual's data to determine whether a loan application is accepted or not. If the application is not accepted, the model then proposes several potential solutions in a table format to facilitate the acceptance of the application.

Displaying multiple solutions is essential because not all applicants can modify certain parameters such as their interest rate or age. Therefore, it is necessary to provide various options to maximize the chances of application approval. These multiple options allow for adaptation to the different situations and constraints of the applicants, offering viable alternatives that can be more easily implemented by each applicant according to their specific circumstances.

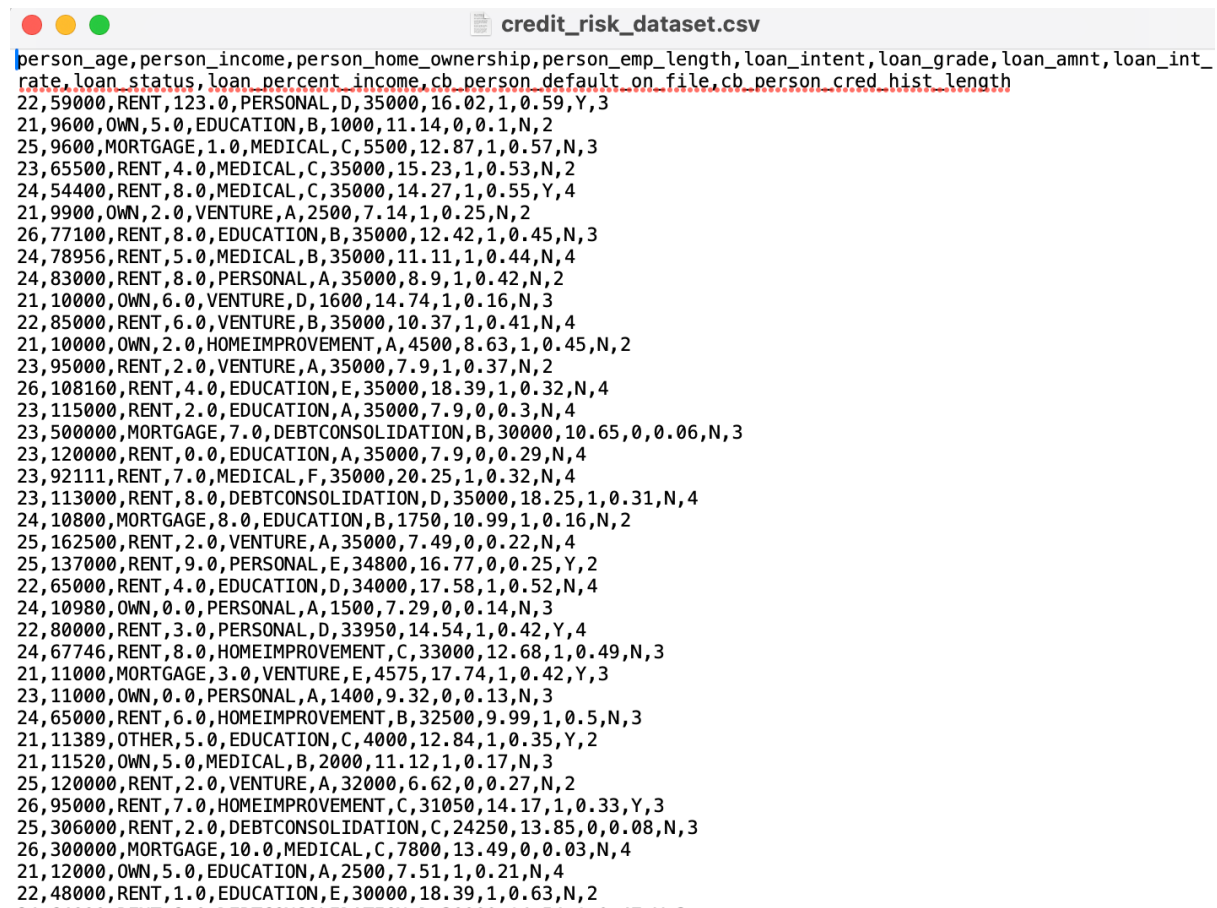
- Code & Algorithms

The code for the conversation creation system can be found in [Appendix 14](#) and [15](#).

Regarding this section, I initially set out to train my own model on a dataset. However, I quickly realized that I could not tackle the explainable AI (xAI) aspect alone. At this point, my advisor recommended the **DiCE** tool (**Diverse Counterfactual Explanations for Machine Learning Classifiers**). This tool was particularly useful for training the model and generating counterfactual solutions.

The model was trained with a dataset containing several parameters related to user information for a bank loan application. Thanks to this training, the model can respond to test information I provide and determine if the application is accepted or not. This decision is

primarily based on the 'loan_status' attribute present in the dataset fields. DiCE enabled me to generate counterfactual explanations, meaning that if an application is rejected, the model suggests possible adjustments to transform a rejected application into an accepted one. These adjustments can include various parameters such as income, credit history, or other relevant criteria, thereby allowing users to understand and improve their chances of obtaining a loan.



```

person_age,person_income,person_home_ownership,person_emp_length,loan_intent,loan_grade,loan_amnt,loan_int_
rate,loan_status,loan_percent_income,cb_person_default_on_file,cb_person_cred_hist_length
22,59000,RENT,123.0,PERSONAL,D,35000,16.02,1,0.59,Y,3
21,9600,OWN,5.0,EDUCATION,B,1000,11.14,0,0.1,N,2
25,9600,MORTGAGE,1.0,MEDICAL,C,5500,12.87,1,0.57,N,3
23,65500,RENT,4.0,MEDICAL,C,35000,15.23,1,0.53,N,2
24,54400,RENT,8.0,MEDICAL,C,35000,14.27,1,0.55,Y,4
21,9900,OWN,2.0,VENTURE,A,2500,7.14,1,0.25,N,2
26,77100,RENT,8.0,EDUCATION,B,35000,12.42,1,0.45,N,3
24,78956,RENT,5.0,MEDICAL,B,35000,11.11,1,0.44,N,4
24,83000,RENT,8.0,PERSONAL,A,35000,8.9,1,0.42,N,2
21,10000,OWN,6.0,VENTURE,D,1600,14.74,1,0.16,N,3
22,85000,RENT,6.0,VENTURE,B,35000,10.37,1,0.41,N,4
21,10000,OWN,2.0,HOMEIMPROVEMENT,A,4500,8.63,1,0.45,N,2
23,95000,RENT,2.0,VENTURE,A,35000,7.9,1,0.37,N,2
26,108160,RENT,4.0,EDUCATION,E,35000,18.39,1,0.32,N,4
23,115000,RENT,2.0,EDUCATION,A,35000,7.9,0,0.3,N,4
23,500000,MORTGAGE,7.0,DEBTCONSOLIDATION,B,30000,10.65,0,0.06,N,3
23,120000,RENT,0.0,EDUCATION,A,35000,7.9,0,0.29,N,4
23,92111,RENT,7.0,MEDICAL,F,35000,20.25,1,0.32,N,4
23,113000,RENT,8.0,DEBTCONSOLIDATION,D,35000,18.25,1,0.31,N,4
24,10800,MORTGAGE,8.0,EDUCATION,B,1750,10.99,1,0.16,N,2
25,162500,RENT,2.0,VENTURE,A,35000,7.49,0,0.22,N,4
25,137000,RENT,9.0,PERSONAL,E,34800,16.77,0,0.25,Y,2
22,65000,RENT,4.0,EDUCATION,D,34000,17.58,1,0.52,N,4
24,10980,OWN,0.0,PERSONAL,A,1500,7.29,0,0.14,N,3
22,80000,RENT,3.0,PERSONAL,D,33950,14.54,1,0.42,Y,4
24,67746,RENT,8.0,HOMEIMPROVEMENT,C,33000,12.68,1,0.49,N,3
21,11000,MORTGAGE,3.0,VENTURE,E,4575,17.74,1,0.42,Y,3
23,11000,OWN,0.0,PERSONAL,A,1400,9.32,0,0.13,N,3
24,65000,RENT,6.0,HOMEIMPROVEMENT,B,32500,9.99,1,0.5,N,3
21,11389,OTHER,5.0,EDUCATION,C,4000,12.84,1,0.35,Y,2
21,11520,OWN,5.0,MEDICAL,B,2000,11.12,1,0.17,N,3
25,120000,RENT,2.0,VENTURE,A,32000,6.62,0,0.27,N,2
26,95000,RENT,7.0,HOMEIMPROVEMENT,C,31050,14.17,1,0.33,Y,3
25,306000,RENT,2.0,DEBTCONSOLIDATION,C,24250,13.85,0,0.08,N,3
26,300000,MORTGAGE,10.0,MEDICAL,C,7800,13.49,0,0.03,N,4
21,12000,OWN,5.0,EDUCATION,A,2500,7.51,1,0.21,N,4
22,48000,RENT,1.0,EDUCATION,E,30000,18.39,1,0.63,N,2

```

Figure 19 Display of the 'credit_risk_dataset' used for model training

- **Challenges**

This section was undoubtedly the most challenging for me. I had to familiarize myself with the world of artificial intelligence and relearn the Python language. The most arduous task was constructing the model from a dataset, which involved categorizing text fields into integer values, extracting relevant data, and properly structuring the model.

The DiCE tool was immensely helpful in creating the new model, as it is a library specialized in explainable AI. With DiCE, I was able to more easily understand the necessary concepts and effectively implement the model. This library guided me through the model training process and the generation of counterfactual explanations, making the learning process more

accessible and structured. Ultimately, DiCE simplified many technical aspects and allowed me to focus on improving the model's accuracy and explainability.

4.4.2.2 Explanation of Decisions and Solutions

- **Results**

```
Explanations:
You need to change 2 features:
1. Negotiate actions to increase person_home_ownership to 3.0
2. increase loan_int_rate from 10.37 value to 19.817115

Explanations:
You need to change 2 features:
1. Take measures to raise person_home_ownership from 2.0 to 3.0
2. decrease loan_percent_income to 0.0

Explanations:
You need to change 1 features:
1. increase loan_grade to 6.0

Additional information:
Having a value of 79.0 for person_age would provide a higher chance of pass compared to a value of 24.0
```

Figure 20 : Display of solutions from a table to a natural form

- **Explanations**

This section constituted the final part of the project. It involved modifying the display of parameters to be adjusted to achieve the acceptance of an application (see [Figure 18](#)), transforming this display into a more natural and comprehensible textual format.

Once this modification was made, the final step was to send the final result of the application, as well as the proposed solutions in case of refusal, to the API. This API is responsible for displaying this information to the user, allowing them to see the suggested solutions to improve their chances of acceptance. This final step ensures that the user receives clear and actionable recommendations, facilitating the understanding and implementation of the necessary adjustments to obtain a favorable response.

- **Code & Algorithms**

This section relies on a model provided by my advisor, Pedram Salmi, which determines the type of each parameter based on the Feature Actionability Taxonomy (FAT). The types of parameters are as follows:

- **MD: Mutable Directly** - A parameter whose value can be directly modified by the recipient through concrete and unconstrained actions. For example, the loan amount in a loan dataset.
- **MI: Mutable Indirectly** - A parameter whose value changes as a result of one or more actions taken by the recipient but cannot be directly modified. For example, the current balance in a loan dataset.

- **IS: Immutable Sensitive** - A parameter whose value cannot be changed and should normally not be suggested as modifiable as it may offend the recipient. For example, race in an income dataset.
- **NSI: Immutable Non-sensitive** - A parameter whose value cannot be modified by an action of the recipient. For example, credit score in a loan dataset.

Table 2: Content and structure themes with examples and alignment to response clusters.

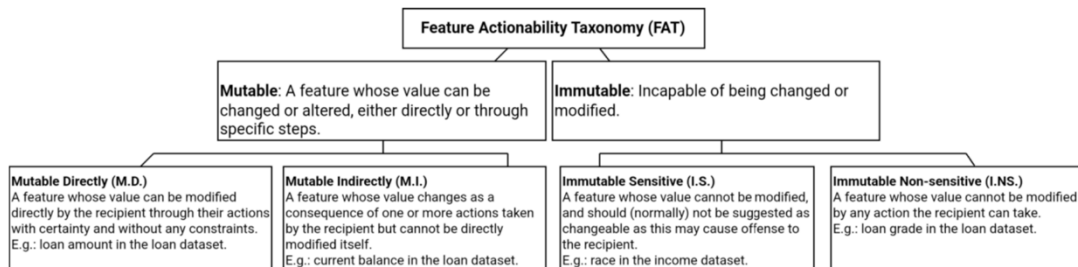


Figure 21 : Definition of the types (Feature Actionability Taxonomy, FAT)

Each parameter being classified according to this system, it is possible to use varied phrasing for the user.

```

feature_classes = {
    "person_age": "NSI",
    "person_income": "DM",
    "person_home_ownership": "IM",
    "person_emp_length": "IM",
    "loan_intent": "IM",
    "loan_grade": "DM",
    "loan_amnt": "DM",
    "loan_int_rate": "DM",
    "loan_percent_income": "DM",
    "cb_person_cred_hist_length": "IM"
}
  
```

Figure 22 : Types for each parameter of the dataset

Here is an example of the phrases used according to the type of parameter:

```

templates = {
  "DM": [
    "{ACTION} {FEATURE} from {QUERY_VALUE} value to {CF_VALUE}",
    "{ACTION} {FEATURE} to {CF_VALUE}"
  ],
  "IM": [
    "{VERB} {OBJECT} to {ACTION} {FEATURE} from {QUERY_VALUE} to {CF_VALUE}",
    "{VERB} {OBJECT} to {ACTION} {FEATURE} to {CF_VALUE}"
  ],
  "NSI": [
    "Having a value of {CF_VALUE} for {FEATURE} would provide a {COMPARATIVE} chance of {DESIRED_OUTCOME}"
  ],
  "SI": [
    "{POSSESSIVE} {FEATURE} has contributed to {OUTCOME}"
  ],
}

```

Figure 23: Phrases used according to the type of parameter

This system offers the possibility to reorganize the structure of the solution according to the taxonomy associated with each parameter. For example, take the 'person_age' parameter. This parameter is classified as **NSI (Immutable Non-sensitive)**, indicating that a person's age cannot be changed on demand. Therefore, it is crucial to adapt the structure of the solution considering this particular parameter.

To avoid repetitive use of the same verbs and words, the 'generate_counterfactual_explanation' function (available in [Appendix 16](#)) allows selecting from a list of verbs and comparatives, making the results more natural and human-like.

To make the result even more unique, my advisor and I decided to send the solution phrases to a **Large Language Model (LLM)** via the ChatGPT API. This model systematically alters the phrasing before returning it to the user, ensuring greater diversity and increased personalization of the suggestions.

The code for the ChatGPT API is available in [Appendix 17](#).

- **Challenges**

This part did not present particular difficulties, as I had the opportunity to discuss in detail with my advisor, who explained the functioning of the system. Although integrating this system into the model interpretation required me to modify my model's operation, it went smoothly thanks to his guidance.

5. Conclusion

5.1 Critical Summary

The work done during this internship led to the development of an interactive web application using modern technologies such as ReactJS for the front-end and FastAPI for the back-end. The implementation of Firebase for data management and continuous integration with GitHub were also strategic choices, providing a robust and secure platform. This project not only meets academic requirements but also aligns with Robert Gordon University's technological innovation policy, reinforcing its commitment to excellence and applied research.

5.2 Personal Assessment

Proposals for Future Work:

Adding new features to the chatbot, such as an assistant dedicated to detecting disease risks, is a proposal for future developments. The goal would be to create a scalable application that can integrate various use cases simply and efficiently.

Challenges and Solutions:

- **Technical:** The initial configuration of FastAPI and Firebase posed integration challenges, as did the creation of an xAI model. These problems were overcome through in-depth research and regular consultations with my internship supervisor.
- **Professional:** Adapting to an international work environment required effective time management and communication skills in English, strengthened by full immersion in the Scottish context.
- **Personal:** The distance from family and friends was challenging, overcome through regular video calls and social activities with local friends.

Contributions to the Company and Personal Gains:

- **Contributions to the Company:** My contribution to research projects was significant, notably through the development of this web application, improving user interaction with AI systems.
- **Personal Gains:** I developed project management skills, mastered new languages and tools (ReactJS, FastAPI, Firebase), and improved my work methods, particularly in time management and meeting deadlines.

Differences Observed Between Practice and Theory:

- **Computer Science:** Applying theoretical knowledge in software development in a practical context highlighted the importance of adaptability and real-time problem-solving.
- **Project Management:** The theory of project management proved more flexible in practice, requiring continuous adjustments and constant communication with stakeholders.
- **Decision-making and Communication Procedures:** The collaborative and iterative approach in a professional environment contrasts with the more linear and formal structure of university projects.

Connection Between the Internship and Knowledge Acquired at IUT:

This internship allowed the practical application of theoretical knowledge acquired at IUT, particularly in web development. Programming and project management courses were especially useful in structuring and successfully completing the project.

Connection Between the Internship and Future Studies or Career:

This internship confirmed my interest in a career as a web developer, with potential specialization in AI. The international experience also reinforced my desire to work abroad in the future if possible.

5.3 Final Conclusion

In retrospect, this internship was a decisive step in my professional journey, offering a valuable opportunity to combine theory and practice in an international context. The technical and interpersonal skills acquired, along with a deep understanding of the challenges and opportunities related to the development of AI solutions, prepare me to pursue a rewarding and impactful career in the field of computer science.

6. References

- **Official ReactJS Documentation**

- **Website:** <https://reactjs.org/>
- **Description:** Comprehensive documentation, guides, and tutorials for learning ReactJS, used for developing the front-end of your web application.

- **Official FastAPI Documentation**

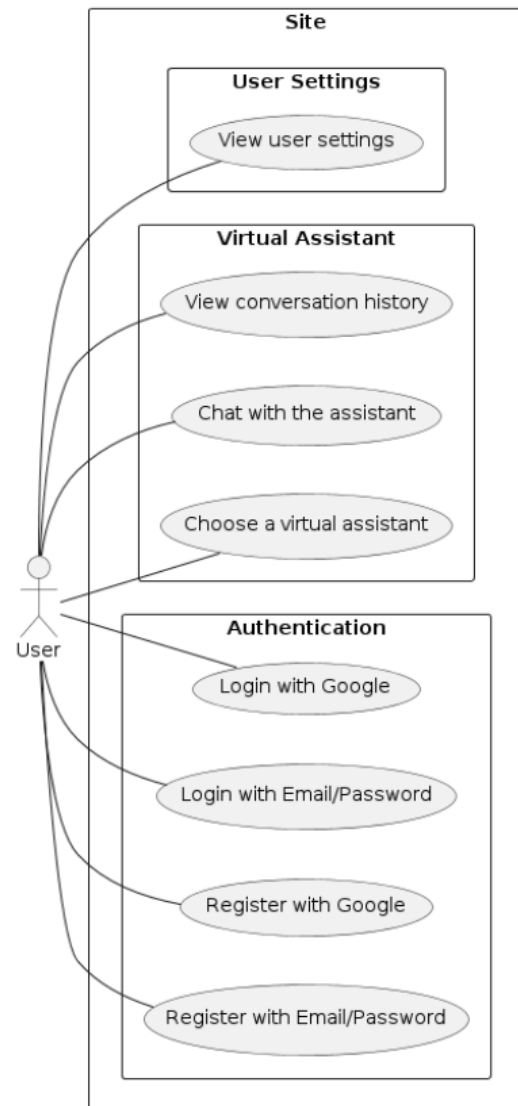
- **Website:** <https://fastapi.tiangolo.com/>
- **Description:** Detailed documentation, examples, and tips for using FastAPI, the framework used for the back-end of your application.

- **Firebase Documentation**

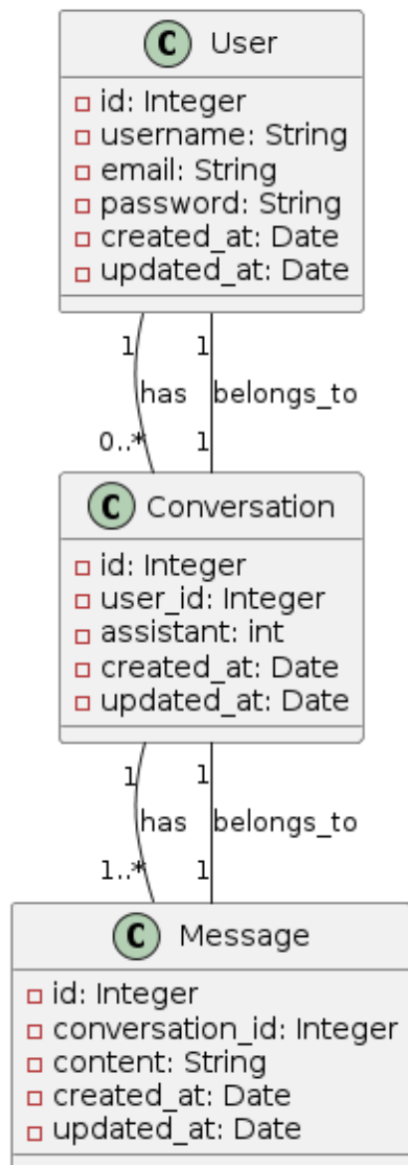
- **Website:** <https://firebase.google.com/docs>
- **Description:** Guides and documentation on Firebase, including real-time data management and data storage, used in your project for data management.

7. Appendix

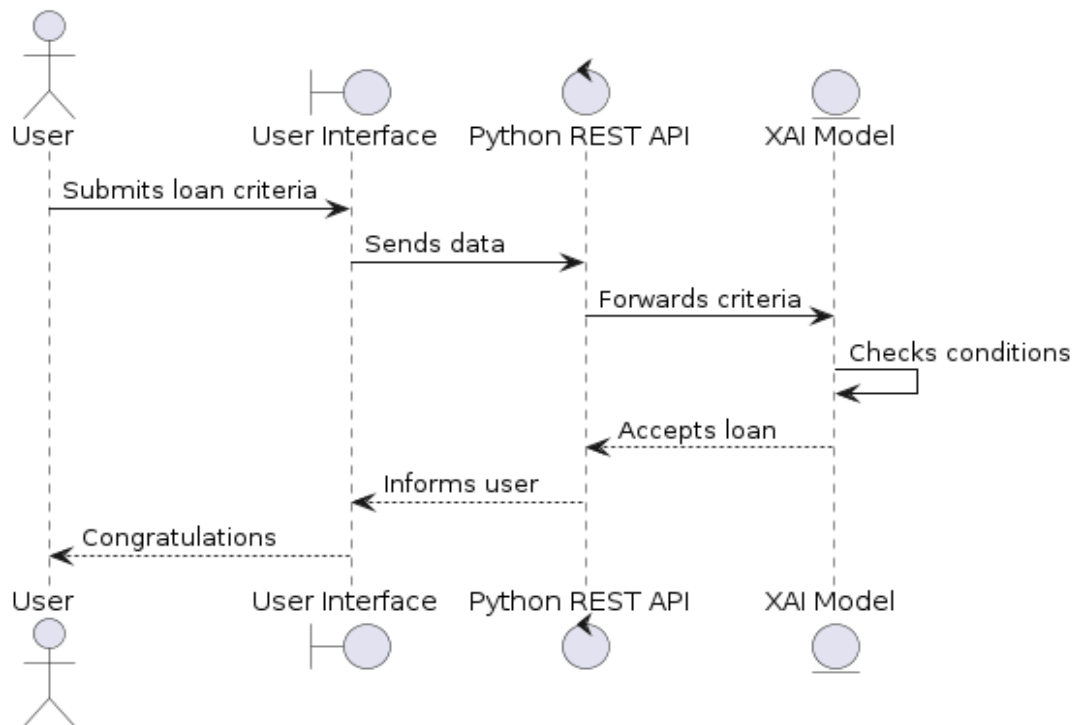
7.1 Appendix 1



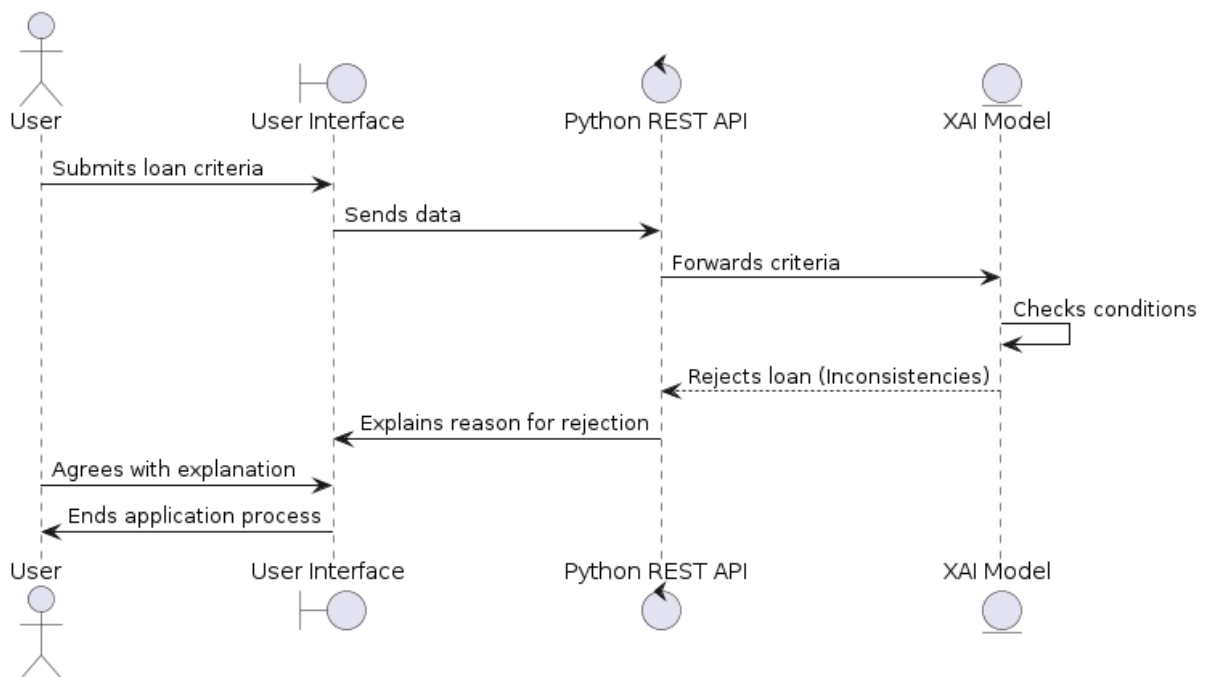
7.2 Appendix 2



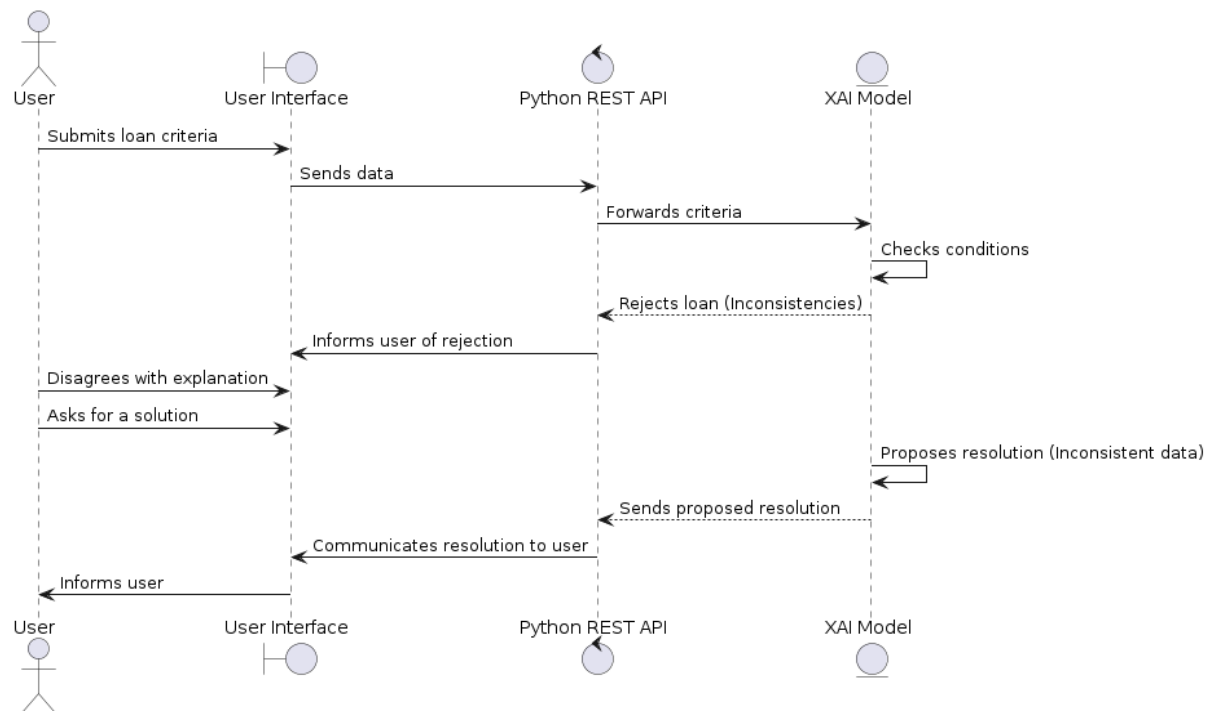
7.3 Appendix 3



7.4 Appendix 4



7.5 Appendix 5



7.6 Appendix 6

```
const handleSubmit = async (event) : Promise<void> => {
  event.preventDefault();
  setLoading( value: true);
  try {
    const userCredential = await createUserWithEmailAndPassword(auth, email, password);
    const user = userCredential.user;
    if (user) {
      const createdAt :Date = new Date();
      await setDoc(doc(db, path: "users", user.uid), data: {
        uid: user.uid,
        email: user.email,
        username: username,
        created_at: createdAt,
        updated_at: createdAt,
      });
    }
    window.location.href = "/";
    toast.success( content: "User successfully registered!", options: {
      position: "top-right"
    });
  } catch (e) {
    toast.error( content: "An error has occurred.", options: {
      position: "top-right",
    });
  }
};
```

7.7 Appendix 7

```
const handleFacebookSignIn = async () => {
  const provider = new FacebookAuthProvider();
  try {
    const result = await signInWithPopup(auth, provider);
```

7.8 Appendix 8

```
const handleGoogleSignIn = async () => {
  const provider = new GoogleAuthProvider();
  try {
    const result = await signInWithPopup(auth, provider);
```

7.9 Appendix 9

```
@router.get("/conversation/get_all/{user_uid}")
async def get_all(user_uid: str):
    user_conversations_with_messages = []
    conversations_ref = db.collection(u'conversations').where(u'user_uid', u'==', user_uid).get()
    for conversation in conversations_ref:
        conv_id = conversation.id
        conversation_with_messages = {
            "conversation_id": conv_id,
            "conversation_info": conversation.to_dict(),
            "messages": []
        }
        messages_ref = db.collection(u'messages').where(u'conversation_uid', u'==', conv_id).order_by(
            u'created_at').get()
        for message in messages_ref:
            conversation_with_messages["messages"].append(message.to_dict())
        user_conversations_with_messages.append(conversation_with_messages)
    return user_conversations_with_messages
```

7.10 Appendix 10

```
const handleCreateConversation = async () => {
    setPendingResponse( value: true);
    setTimeout( handler: async () => {
        try {
            const formData :FormData = new FormData();
            const additionalData :{user_uid: any} = {user_uid: props.user_uid...};

            const additionalDataBlob :Blob = new Blob( sources: [JSON.stringify(additionalData)], options: {type: 'application/js
            const userDataBlob :Blob = new Blob( sources: [JSON.stringify(userResponses)], options: {type: 'application/json'});

            formData.append( name: "additional_data", additionalDataBlob);
            formData.append( name: "user_data", userDataBlob);

            const response :AxiosResponse<any> = await axios.post( url: 'http://localhost:8000/api/conversation/create', formDat

            const chat_uid = response.data.id;
            navigate(`/chat/` + chat_uid);
        } catch (error) {...}
        setPendingResponse( value: false);
    }, delay);
};
```

7.11 Appendix 11

```
const questions : [{question: string, type: str... = [
  {key: 'person_age', question: 'What is your age?', type: 'number'},
  {key: 'person_income', question: 'What is your income?', type: 'number'},
  {key: 'person_home_ownership'...},
  {key: 'person_emp_length', question: 'How many years have you been employed?', type: 'number'},
  {key: 'loan_intent'...},
  {key: 'loan_grade'...},
  {key: 'loan_amnt', question: 'What is the loan amount?', type: 'number'},
  {key: 'loan_int_rate', question: 'What is the loan interest rate?', type: 'number'},
  {key: 'loan_percent_income'...},
  {key: 'cb_person_default_on_file'...},
  {key: 'cb_person_cred_hist_length', question: 'How long is your credit history?', type: 'number'},
];
```

7.12 Appendix 12

```
const fetchData = async () => {
  try {
    if (id) {
      const response : AxiosResponse<any> = await axios.get(url: `http://localhost:8000/api/conversation/get/${id}`);
      setConversation(response.data);
      setMessages( value: []);
    }
  } catch (error) {
    console.error('Error fetching data:', error);
  }
};
```

7.13 Appendix 13

```
const fetchExplanation = async () => {
  if (conversation) {
    if (!conversation.conversation_info.accepted) {
      try {
        if (id) {
          const response : AxiosResponse<any> = await axios.get(
            url: `http://localhost:8000/api/conversation/get/${id}/explanation`);
          setExplanation(response.data.explanations);
        }
      } catch (error) {
        console.error('Error fetching data:', error);
      }
    }
  }
}
```

7.14 Appendix 14

```
def predict_loan(query_dict):
    for feature in categorical_features:
        if feature in query_dict:
            le = label_encoders.get(feature)
            query_dict[feature] = le.transform([query_dict[feature]])[0]

    sample_df = pd.DataFrame([query_dict])[X.columns]
    prediction = model.predict(sample_df)[0]
    return "Accepted" if prediction == 1 else "Rejected"
```

7.15 Appendix 15

```
def explain_rejection(query_dict):
    for feature in categorical_features:
        if feature in query_dict:
            le = label_encoders.get(feature)
            query_dict[feature] = le.transform([query_dict[feature]])[0]

    sample_df = pd.DataFrame([query_dict])[X.columns]
    explanation = exp.generate_counterfactuals(sample_df, total_CFs=5, desired_class="opposite")
    counterfactuals_df = explanation.cf_examples_list[0].final_cfs_df

    explanations_list = []
    for _, cf in counterfactuals_df.iterrows():
        explanations = generate_explanations(sample_df.iloc[0], cf)
        explanations_list.append(explanations)

    query_dict = {k: (
        int(v) if isinstance(v, (np.integer, np.int64)) else float(v) if isinstance(v, (np.float64, np.float32)) else v
    ) for k, v in query_dict.items()}
    counterfactuals_list = counterfactuals_df.applymap(
        lambda x: int(x) if isinstance(x, (np.integer, np.int64)) else float(x) if isinstance(x, (
            np.float64, np.float32)) else x).to_dict(orient="records")

    return {
        "explanations": explanations_list
    }
```

7.16 Appendix 16

```
def generate_counterfactual_explanation(feature, query_value, cf_value, actionability_class):
    template = random.choice(templates[actionability_class])
    action_word = determine_action_word(query_value, cf_value)

    placeholders = {
        "{VERB}": ["Take", "Initiate", "Undertake", "Pursue", "Negotiate"],
        "{OBJECT}": ["steps", "measures", "actions"],
        "{ACTION}": [action_word],
        "{COMPARATIVE}": ["increase", "higher", "better"],
        "{DESIRED_OUTCOME}": ["accepted", "pass"],
        "{OUTCOME}": ["heart disease"],
        "{POSSESSIVE}": ["Your"],
        "{FEATURE}": [feature],
        "{QUERY_VALUE}": [query_value],
        "{CF_VALUE}": [cf_value],
    }

    for placeholder, values in placeholders.items():
        template = template.replace(placeholder, str(random.choice(values)))

    return template
```

7.17 Appendix 17

```
def improve_message(message):
    chat_completion = client.chat.completions.create(
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user",
             "content": f"Please improve the following message to be more human and polite:\n\n{message}"},
        ],
        model="gpt-3.5-turbo",
    )
    return chat_completion.choices[0].message.content
```

8. Summaries

8.1 French

Ce stage a permis de développer une application web interactive en utilisant des technologies modernes telles que ReactJS pour le front-end et FastAPI pour le back-end. L'intégration de Firebase pour la gestion des données et de GitHub pour l'intégration continue a renforcé la robustesse et la sécurité de la plateforme. Ce projet répond aux exigences académiques tout en s'alignant avec la politique d'innovation technologique de l'université Robert Gordon. En plus de renforcer mes compétences en développement web et en gestion de projet, ce stage m'a permis de découvrir le domaine de l'intelligence artificielle (IA), suscitant un intérêt profond pour cette spécialisation. Les défis rencontrés ont été surmontés grâce à des recherches approfondies et à une communication efficace avec les parties prenantes. Ce stage a confirmé mon désir de poursuivre une carrière de développeur web spécialisé en IA et a renforcé mon souhait de travailler à l'étranger.

8.2 English

This internship involved the development of an interactive web application using modern technologies such as ReactJS for the front-end and FastAPI for the back-end. The integration of Firebase for data management and GitHub for continuous integration enhanced the platform's robustness and security. This project meets academic requirements and aligns with Robert Gordon University's technological innovation policy. In addition to enhancing my skills in web development and project management, this internship introduced me to the field of artificial intelligence (AI), sparking a deep interest in this specialization. Challenges were overcome through thorough research and effective communication with stakeholders. This internship confirmed my desire to pursue a web development career specializing in AI and reinforced my wish to work abroad.

8.3 Keywords

- Robert Gordon University
- Web Development
- ReactJS
- FastAPI
- Firebase
- Artificial Intelligence
- Project Management
- International Environment