

THE NATIONAL INSTITUTE OF POSTS AND TELECOMMUNICATIONS - RABAT

Project Report: Modern Web Development

FAANG LAY-OFFS



Carried out by :

ELGHABI TAHA
LABRIJI SAAD
ISBAINE MOHAMMED

Supervised by :

Pr. HAQIQ ABDELHAY

Table of contents

Table of contents	3
Résumé	5
Abstract	6
General Introduction	7
1 General context of the project	8
1.1 Subject presentation	8
1.2 Functional requirement	9
1.3 Besoins non fonctionnels	9
2 Analysis and Conception	10
2.1 Actors :	10
2.2 Use case diagram	10
2.3 Design and analysis of general architecture of the Plat- form	11
2.3.1 Different Elements of Architecture	11
2.4 General architecture and workflow	14
2.5 Design patterns:	15
2.5.1 Inversion of control and dependencies injection:	15
2.5.2 MVC model:	16
2.5.3 ORM	17
2.6 Conclusion	18
3 Machine Learning	19
3.1 Introduction	19

3.2	Data Scraping	20
3.3	Exploratory Data Analysis	21
3.4	Data Cleaning	22
3.5	Feature Engineering	23
3.6	Preprocessing	24
3.7	Model Building	25
3.8	Model to production	26
3.9	Conclusion	26
4	Implementation	27
4.1	Introduction	27
4.2	Tools Used:	28
4.2.1	HTML CSS :	28
4.2.2	TypeScript:	29
4.2.3	Angular:	30
4.2.4	Spring Boot:	31
4.2.5	Python :	31
4.2.6	Flask :	32
4.3	Development environment	32
4.3.1	Visual Studio Code :	32
4.3.2	Jupyter Notebook :	33
4.3.3	XAMPP :	34
4.3.4	Version and collaboration management	34
4.4	Web Platform Interfaces	36
4.4.1	Homepage	36
4.4.2	Registration page	36
4.4.3	Login Page	37
4.4.4	Main Interface	38
4.4.5	Waiting Interface	38
4.4.6	Page du sourd et/ou muet	39
	General Conclusion	41
	Bibliography	42

List of Figures

2.1	Use Case Diagram	11
2.2	Architecture	12
2.3	MVC Model	16
2.4	ORM Entity of Company	17
3.1	AI Machine Learning	19
3.2	Data Scraping	20
3.3	EDA	21
3.4	Handling Exceptions	22
3.5	Correlation matrix	23
3.6	Preprocessing	24
3.7	Model Building	25
3.8	Exporting Model	26
4.1	Logo - Flask	32
4.2	Logo - VS Code	33
4.3	Logo - Jupyter Notebook	33
4.4	Logo - XAMPP	34
4.5	Logo - Git	34
4.6	Logo - Github	35
4.7	Logo - Notion	35
4.8	Web Platfrom - Landing Page	36
4.9	Web Platfrom - Register Page	37
4.10	Web Platfrom - Login Page	37
4.11	Web Platfrom - Main Interface	38
4.12	Web Platfrom - Waiting Animation	39
4.13	Web Platfrom - Scraping Result	40
4.14	Web Platfrom - Prediction Result	40

Résumé

Ce rapport résume les travaux réalisés dans le cadre de notre projet pour le cours du Developpement Et Sécurité Des Applications Web Modernes à l'Institut National des Postes et Télécommunications qui est notre plateforme FAANG lay-offs.

L'objectif principal de ce travail est de créer une plate-forme qui **"Prédit le licenciement potentiel des employés à l'aide de l'apprentissage automatique"**.

Ce travail consiste en plusieurs phases: Choisir une source de données, Scraper et préparer l'ensemble de données, construire un modèle d'apprentissage automatique capable de prédire la probabilité d'être licencié. Crée un serveur Web Spring pour l'authentification et la gestion des utilisateurs abonnés à l'aide du modèle MVC. et un Frontend pour la plateforme avec Angular. le projet impliquera de nombreux concepts dont : Web Scraping, Supervised Learning, MVC pattern, JWT, Spring

Keywords: Web Scraping, Supervised Learning, MVC, Spring, Angular.

Abstract

This report summarizes the work carried out within the framework of our project for the Modern Web Application Development and Security Programming course in National Institute of Posts and Telecommunications which is FAANG lay-offs.

The main objective of this work is to create a Platform that will **"Predict the potential layoff of tech employees using Machine Learning"**.

This work consists of several phases of choosing a data source, Scraping and Preparing the Dataset, Building a Machine Learning model capable of prediction the probability of getting laid off, creating a Spring web server for authentication and managing subscribed users using the MVC pattern. and a Frontend for the platform with Angular. the project will involve many concepts including: Web Scraping, Supervised Learning, MVC pattern, JWT, Spring Security.

Keywords: Web Scraping, Supervised Learning, MVC, Spring, Angular.

General Introduction

We have chosen the field of employment in the Tech industry, since the recession have subverted the tech world and we as Computer Science student are concerned too with the future jobs opportunities in tech companies after our graduation.

Our project consists in proposing a solution to this problem by predicting the probability of getting laid off, the solution will be a web application and a Machine Learning model which works on the data from Linkedin platform.

In this report we will go through all the steps to showcase the utility of the project in the general context, and we will tackle the technical aspect in the analysis and the design then move on to the implementation by stating the tools that we used.

In the first chapter, we will talk about the general context of the project to put you in situation, then we will move on to the analysis and the design in the second chapter. The latter will contain a description of the use cases and the needs of each of them in terms of technologies and architecture. And finally in the 3rd chapter we will mention the tools used and insert screenshots of the application.

Chapter 1

General context of the project

This part allows us to present the circumstances of our project in a general way, you will discover the subject, the needs and the planning of the project.

1.1 Subject presentation

More than 91,000 workers in the U.S. tech sector have been laid off in mass job cuts so far in 2022. and Therefore many People are currently worried about their current position.

The FAANG Layoffs platform targets this problem and provides users (current tech employee) with the probability of getting laid off based on their Linkedin Profile, and a more sophisticated estimation using more parameters such as the company stock and the news about the it.

The app doesn't stop there, but it goes beyond that to provide the option to get more personalized recommendations for job offers using the data we have.

1.2 Functional requirement

Guest User:

- Enter Linkedin Profile
- Obtain probability of Layoff

Registered User:

- Authentication
- Enter Linkedin Profile Once
- Obtain more accurate estimate of Layoff
- Getting other suitable job Offer

1.3 Besoins non fonctionnels

Once we have dealt with the functional side, we must also consider the non-functional needs, for example:

- Confidentiality
- Easy-to-use
- Progressive Change
- Adaptability

Chapter 2

Analysis and Conception

This chapter will present an analysis of our project.

2.1 Actors :

The actors in our application are mainly as follows:

The developer: He can manages user accounts, he can also add services to the application (Add more supported companies, train model on more data)

The user: Obtain the probability of getting laid off and get a customized job offers recommendations.

2.2 Use case diagram

A use case diagram captures the behavior of a system, subsystem, class, or component as an outside user sees it. It splits the functionality of the system into coherent units, use cases, that make sense to the actors. The use cases make it possible to express the need of the users of a system, they are therefore a user-oriented vision of this need, unlike a computer vision.

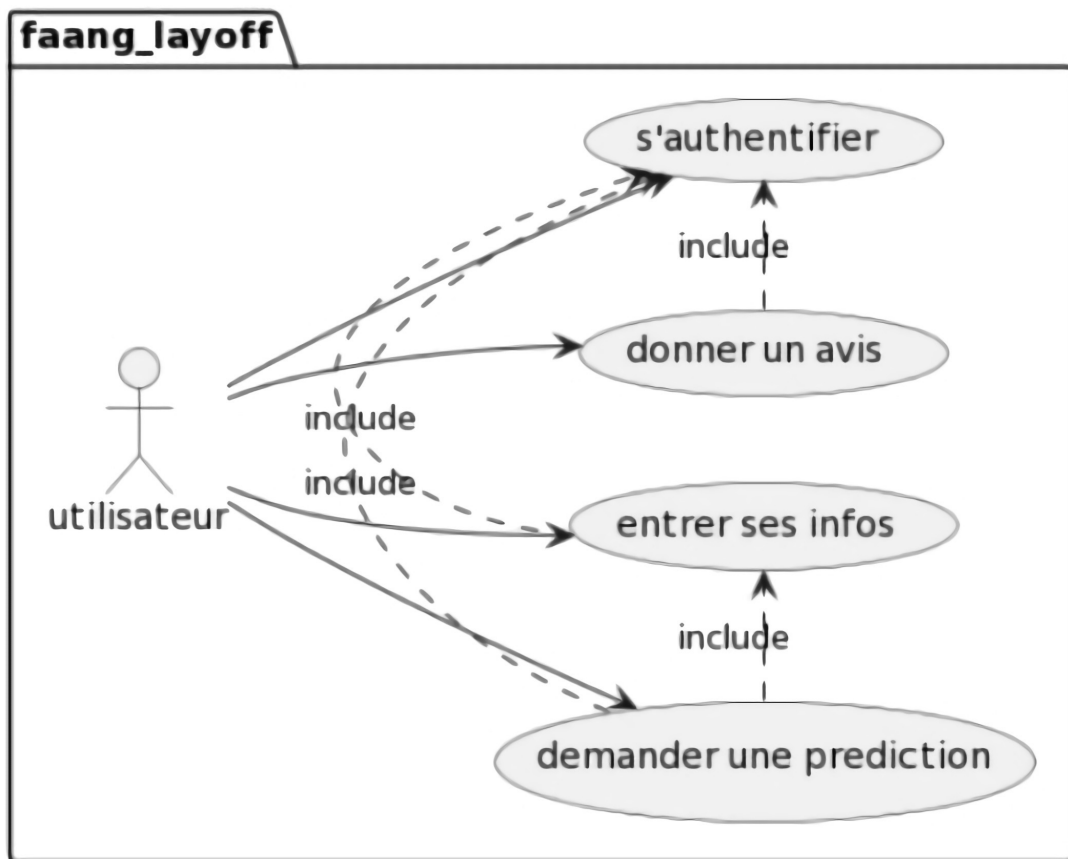


Figure 2.1: Use Case Diagram

2.3 Design and analysis of general architecture of the Platform

After presenting what the application is all about. In this part of the chapter, we will provide the general architecture of our application, including the workflow for user interaction with the application, and how the server will handle the authentication and process of user queries. And the problems of securing the information from client side to the backend and limit the possibility of getting multiple prediction to subscribed users only.

2.3.1 Different Elements of Architecture

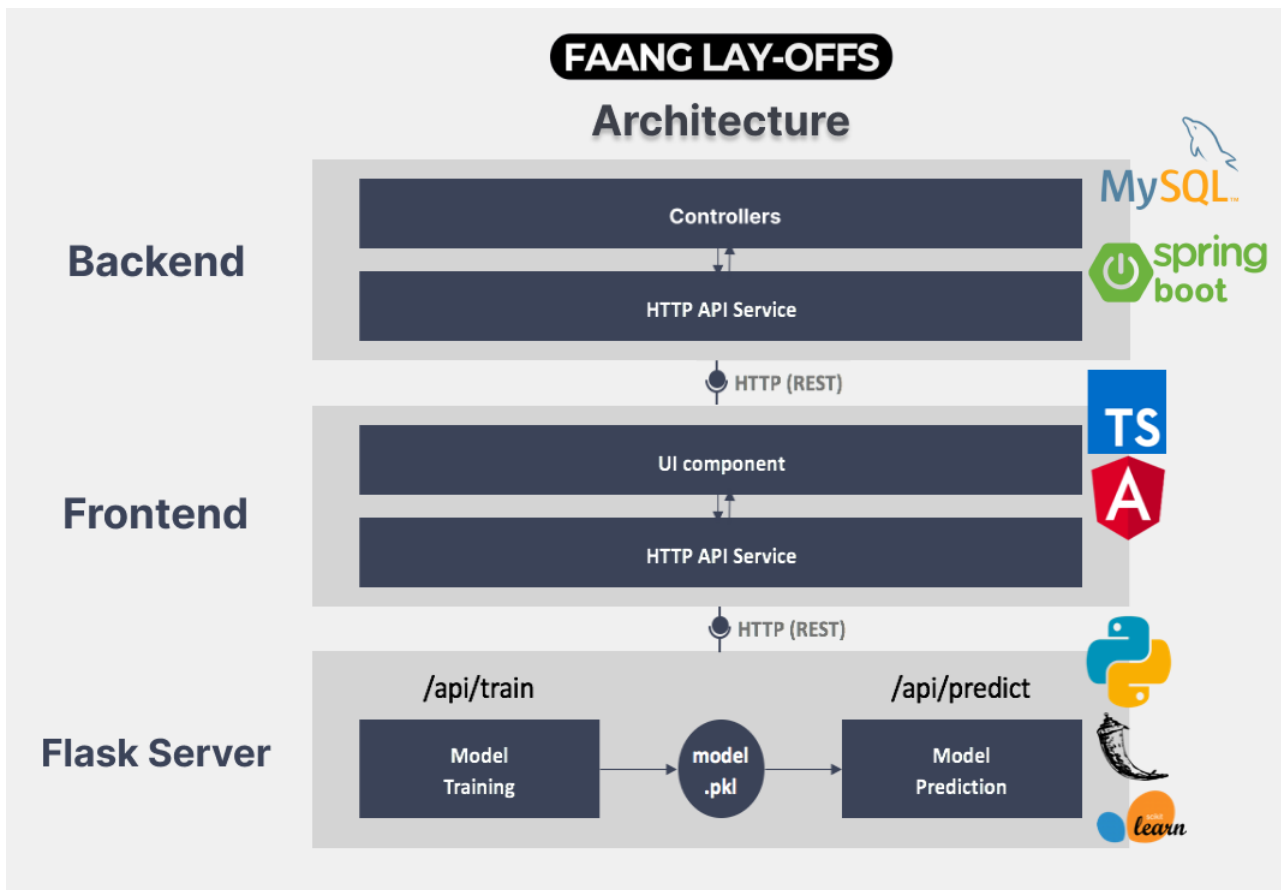


Figure 2.2: Architecture

Client Side : In web development, 'client side' refers to everything in a web application that is displayed or takes place on the client (end user device). This includes what the user sees, such as text, images, and the rest of the UI, along with any actions that an application performs within the user's browser.

Angular is the client-side in our project, It is an open-source, TypeScript-based framework designed by Google. Angular offers many advantages for which we have chosen to develop with, among these advantages:

- => Automatic synchronization with Two-way data binding.
- => Consistency and reusability of the code, thanks to the Angular CLI (line interface of command) and the documentation style guide
- => Default Ivy renderer.

=> Declarative and intuitive user interface that eliminates the need to invest a lot of time in program streams and schedule what loads first.

Server Side: Much like with client side, 'server side' means everything that happens on the server, instead of on the client. In the past, nearly all business logic ran on the server side, and this included rendering dynamic webpages, interacting with databases, identity authentication, and push notifications.

For the server side we used Spring Boot which is Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications.

Our application is based on the Restful model, so all actions are triggered by exchanging data through between controllers and the view and this entire process is satisfied through the backend which is responsible for managing:

Dataset: the database is used for storage purposes. Here we use MySql databases for production, it should be noted that the only part that uses the database is the authentication using hibernate to do the communication (ORM).

The dataset is stored in a database managed by the backend. Communication between the frontend and the backend is done through http requests by communicating encrypted data in JSON (Restful API) format. Thus, the concerned controller processes the requests and allows to extract the required data from the database, format them, and then generate the view to the user.

Users: The backend is the part responsible for making a set of routes available to the different users of the system. He will be the intermediary between the devices and the different client applications

(direct the user during his navigation within the application), this is organized by granting controllers for each category of users (Administrator, Doctor and patient).

Data security: The backend part is responsible for managing user access rights by giving rights to each authenticated user according to the authorizations he has. For example, a patient cannot access the information of other patients or doctors.

2.4 General architecture and workflow

Now that all the elements are known to the reader, we can dive into the general architecture and see how each component of our application interacts with other components. thus we will establish the knowledge necessary to see why the MVC approach is great. and can now discuss what kind of problems it solves for our use case, and also what kind of problem it generates and how we will handle these problems.

Ideal workflow:

The client sends an HTTP request including the page number and the credentials (username and password) the authentication will validate the credentials and issue a JWT token to the client containing its permissions and personal information every time it sends a request to the server it sends this JWT token, the backend server check see if the JWT token is valid (Encryption and signing RSA). Following the authentication the user can get a prediction of layoff for a linkedin profile, after that it will delegate the request to the end user flask server to extract the the information from the provided profile including: First Name, Last Name, Headline, Location, Company, Time at the comapny and total years of experience an then delegate these details to the endpoint /predict which will load the machine learning model and make a prediction on the provided information and then it will wrap the response and delegate it to the user if the user is authenticated, it will also provide subscribed users with more information about the company performance as well as alternative job offers.

2.5 Design patterns:

in this section, we will talk about the design patterns we used to develop our development, we will study only one the authentication part and the parts must follow the same design pattern.

2.5.1 Inversion of control and dependencies injection:

The spring framework is based on these two concepts which offer a production-ready design pattern.

2.5.2 MVC model:

We used the famous MVC model (with small customizations because we don't have a "VIEW" here)

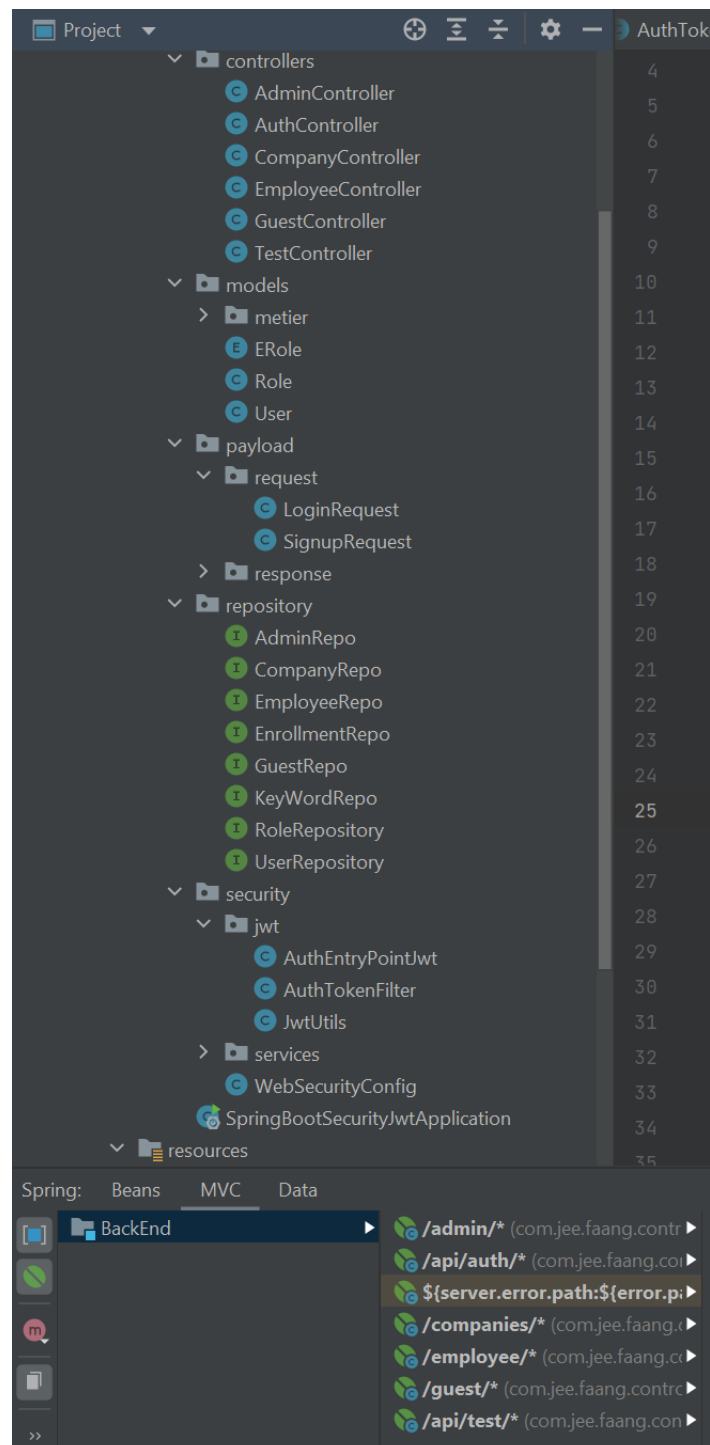
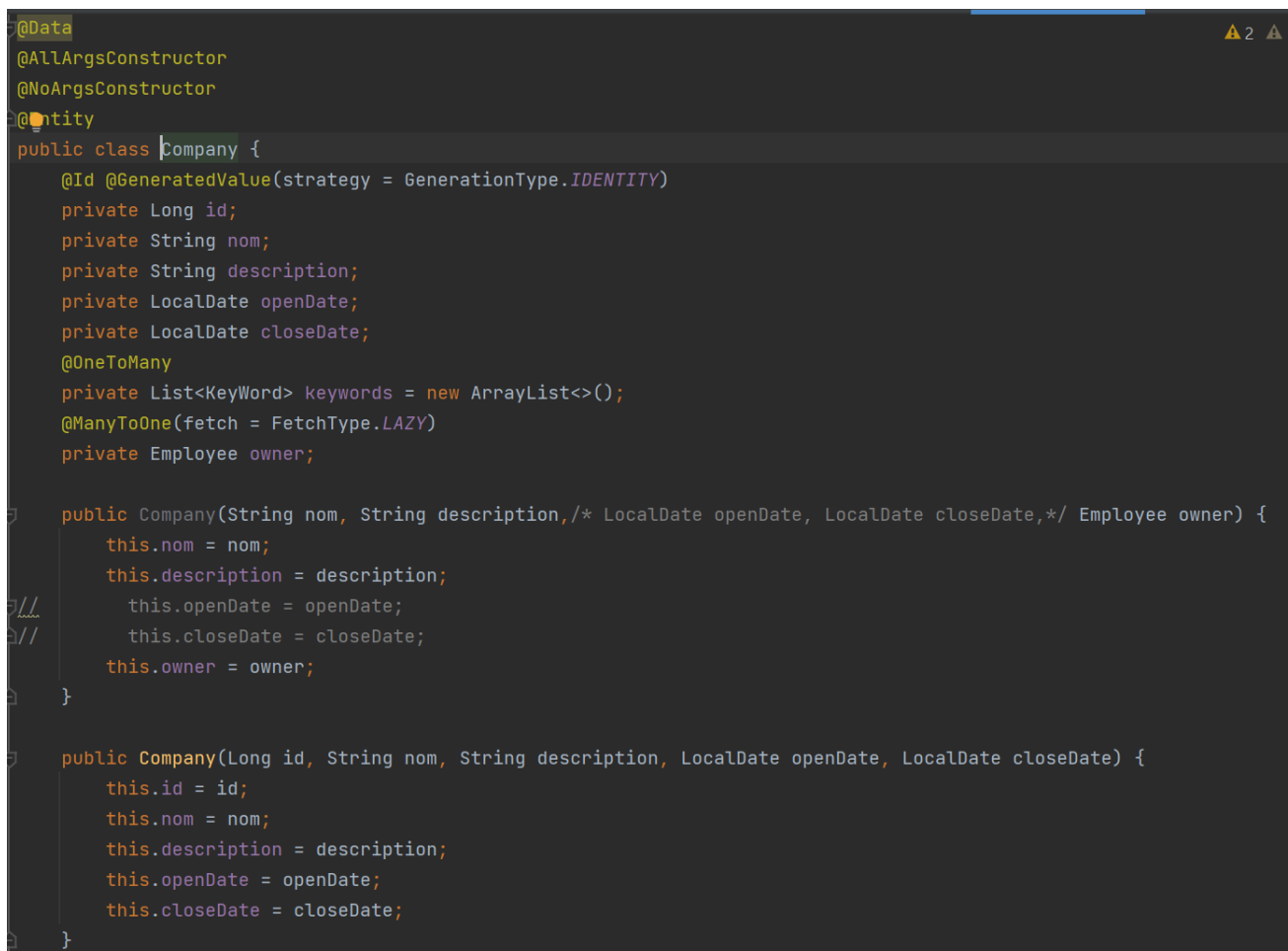


Figure 2.3: MVC Model

2.5.3 ORM

An object-relational mapping (ORM) is a type of computer program that interfaces between an application program and a relational database to simulate an object-oriented database. This program defines correspondences between the schemas of the database and the classes of the application program. We could designate it by this, “as a layer of abstraction between the object world and the relational world”. Because of its function, we find this type of program in a large number of frameworks in the form of an ORM component that has been either developed or integrated from an external solution.

A screenshot of a code editor showing the implementation of a Java class named 'Company'. The class is annotated with JPA annotations: @Data, @AllArgsConstructor, @NoArgsConstructor, and @Entity. It has private fields for id (Long), nom (String), description (String), openDate (LocalDate), closeDate (LocalDate), keywords (List<KeyWord>), and owner (Employee). The keywords field is annotated with @OneToMany, and the owner field is annotated with @ManyToOne(fetch = FetchType.LAZY). There are two constructors: one taking nom, description, openDate, closeDate, and owner, and another taking id, nom, description, openDate, and closeDate. The code is written in a dark-themed editor with syntax highlighting.

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Company {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String description;
    private LocalDate openDate;
    private LocalDate closeDate;
    @OneToMany
    private List<KeyWord> keywords = new ArrayList<>();
    @ManyToOne(fetch = FetchType.LAZY)
    private Employee owner;

    public Company(String nom, String description, /* LocalDate openDate, LocalDate closeDate,*/ Employee owner) {
        this.nom = nom;
        this.description = description;
        // this.openDate = openDate;
        // this.closeDate = closeDate;
        this.owner = owner;
    }

    public Company(Long id, String nom, String description, LocalDate openDate, LocalDate closeDate) {
        this.id = id;
        this.nom = nom;
        this.description = description;
        this.openDate = openDate;
        this.closeDate = closeDate;
    }
}
```

Figure 2.4: ORM Entity of Company

2.6 Conclusion

In this chapter we have presented in a global way the main stages of the analysis and design of our application according to the different diagrams, in order to complete the implementation phase. The next chapter will be devoted to machine learning part of the application.

Chapter 3

Machine Learning

3.1 Introduction

Our platform must be able to predict the probability of layoff from a user's linkedin profile. After getting the data, we will build a machine learning model to do this.

Machine Learning can be defined as an artificial intelligence technology that allows machines to learn without having been previously programmed specifically for this purpose. Machine Learning is explicitly tied to Big Data, as to learn and grow, computers need streams of data to analyze, to train on.

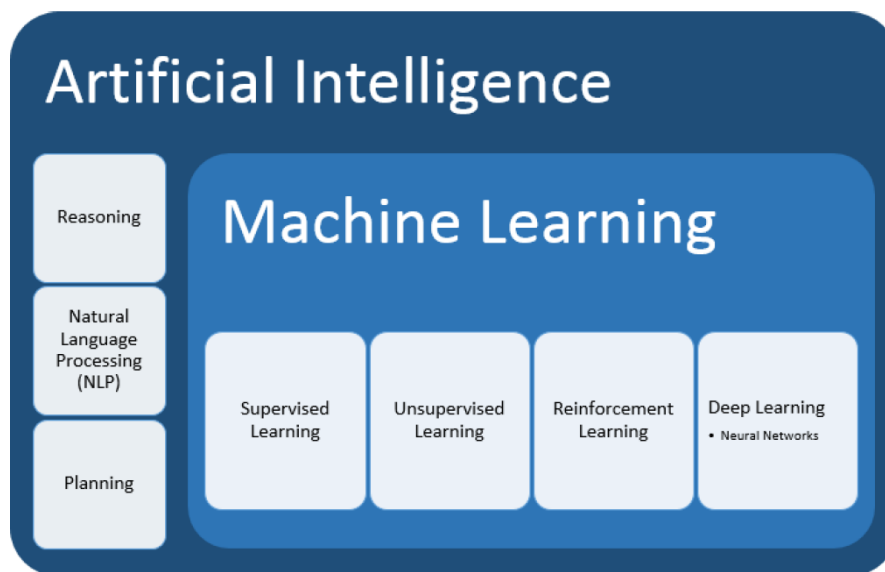


Figure 3.1: AI Machine Learning

there are different type of Machine Learning, in our case we will scrap they data and it "Labels" from linkedin so our problem will **Supervised Learning** problem. However we must go through several steps to achieve a model capable of performing this task.

3.2 Data Scraping

One of the initial considerations for any data science project is acquiring a dataset. In this particular case, no dataset was available, so we had to scrape the data ourselves using **Selenium**. LinkedIn was the preferred platform for this task. As an example, we used the company Meta. To identify individuals who had been laid off, we searched for **#metalayoffs** on LinkedIn to locate relevant profiles from posts. As for individuals who were still employed at Meta, we accessed the "People" section of Meta's LinkedIn page as a source.

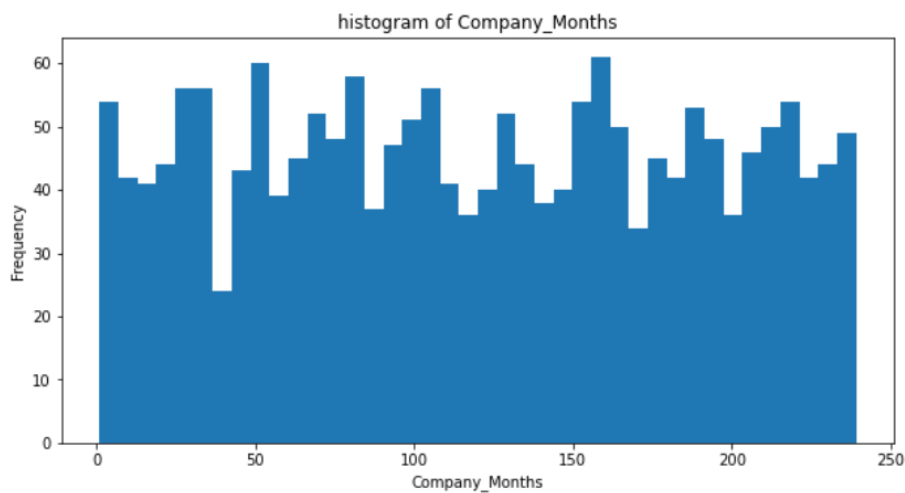
```
32 # define your company name right here:
33 Company_name = "meta"
34
35 # defining the webdriver and config btw this code will be almost the same in all of your selenium scripts
36 options = Options()
37
38 # !!! blocking browser notifications !!!
39 prefs = {"profile.default_content_setting_values.notifications": 2}
40
41 # starting in maximized window
42 options.add_argument("start-maximized")
43 options.add_argument("--disable-default-apps")
44 driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
45
46
47 driver.get("https://www.linkedin.com/login/")
48
49
50 username = os.environ.get("LINKEDIN_USERNAME")
51 password = os.environ.get("LINKEDIN_PASSWORD")
52
53 username_field = driver.find_element(By.ID,"username")
54 username_field.send_keys(username)
55
56 password_field = driver.find_element(By.ID,"password")
57 password_field.send_keys(password)
58
59 password_field.submit()
60
61 count1 = 0
62 driver.get("https://www.linkedin.com/search/results/content/?authorJobTitle=%22Meta%22&keywords=%23metalayoffs&origin=FACETED_SEARCH&sid=")
63 time.sleep(3)
64 items_urls = []
65 testing = []
66 while count1 < 10:
67     try:
68         profiles_links = driver.find_elements(By.XPATH, "//body/div[4]/div[3]/div[2]/div[1]/div[1]/main[1]/div[1]/div[1]/div[1]/div[1]/ul")
69
```

Figure 3.2: Data Scraping

3.3 Exploratory Data Analysis

Exploratory data analysis (EDA) is a process of analyzing and summarizing a dataset in order to better understand its characteristics and relationships. It involves visualizing the data and using statistical methods to identify patterns and trends. The goal of our EDA is to **gain insights**. EDA is an important step in the data science process because it helps to identify potential issues with the data that will be solved in the next step of our life cycle.

```
In [9]: histogram("Company_Months")
```



```
In [10]: histogram("Work_Months")
```

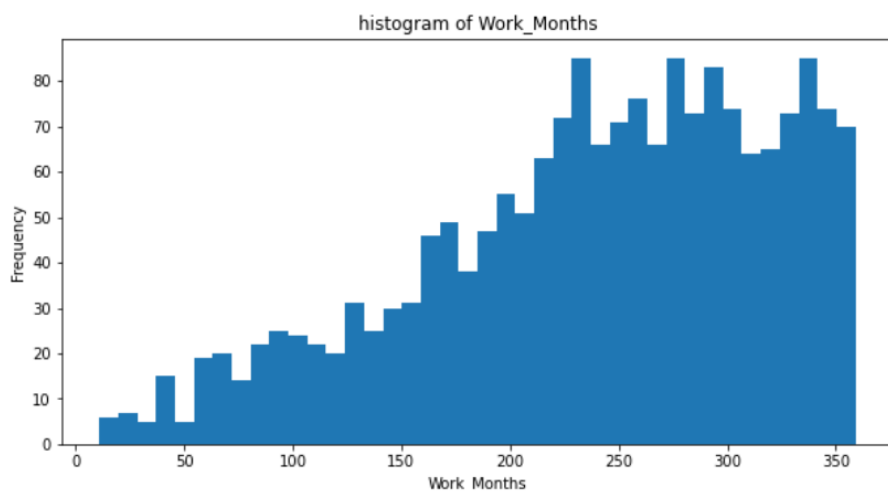
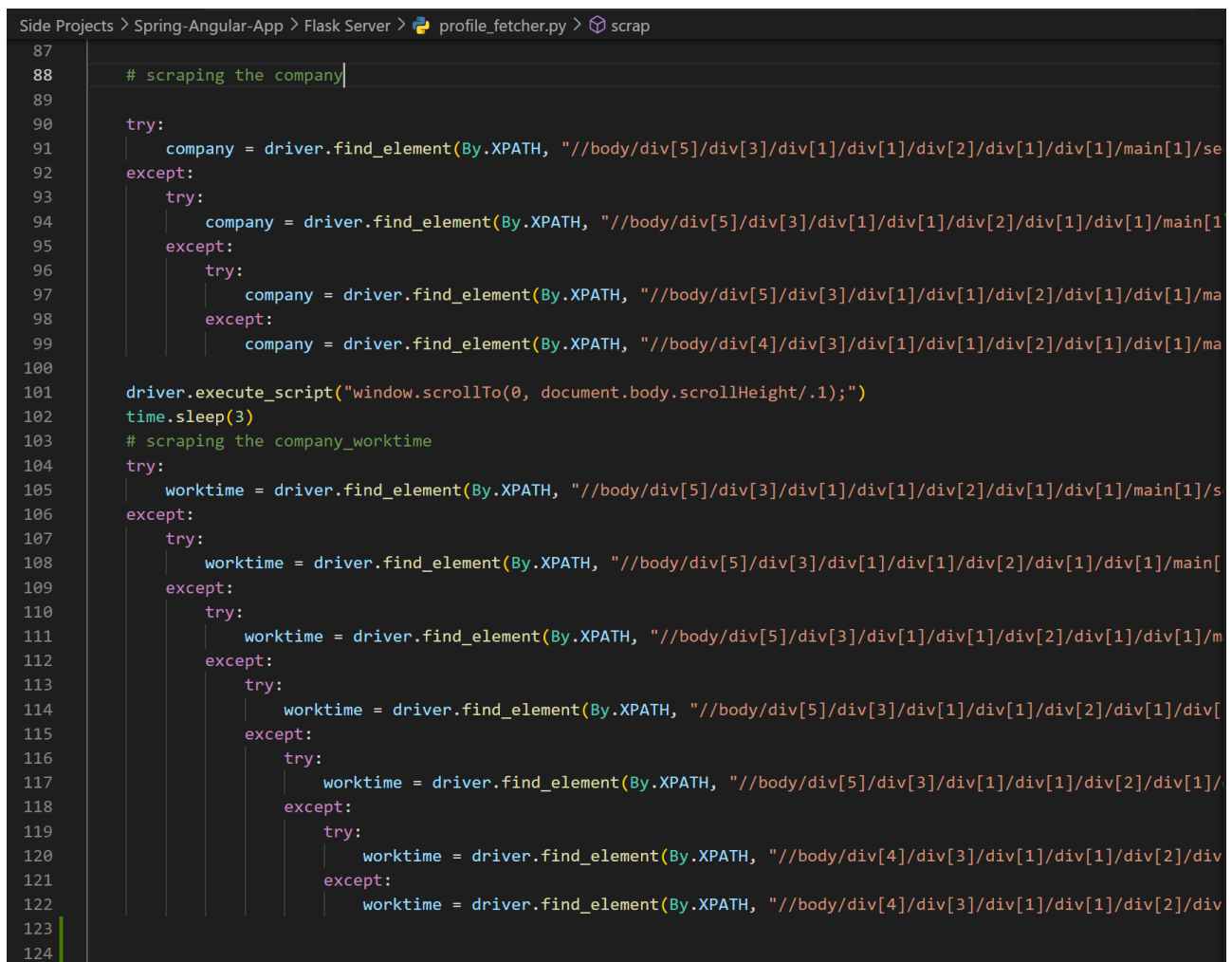


Figure 3.3: EDA

3.4 Data Cleaning

After the data understanding step, the next step in the lifecycle stages is data preparation. This step is also known as **Data Cleaning** or **Data Wrangling**. It includes steps such as selecting relevant data, integrating data by merging datasets, cleaning, managing missing values by deleting or imputing them with relevant data, dealing with erroneous data by removing them, also checking for outliers and dealing with them. In our case, we have handle multiple cases (10 try/except statements sometimes to handle the different forms in which Linkedin present data in the HTML)

So we didn't have to go through this step.



```
Side Projects > Spring-Angular-App > Flask Server > profile_fetcher.py > scrap
87
88 # scraping the company
89
90 try:
91     company = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/main[1]/se
92 except:
93     try:
94         company = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/main[1]
95     except:
96         try:
97             company = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/ma
98         except:
99             company = driver.find_element(By.XPATH, "//body/div[4]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/ma
100
101 driver.execute_script("window.scrollTo(0, document.body.scrollHeight/.1);")
102 time.sleep(3)
103 # scraping the company_worktime
104 try:
105     worktime = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/main[1]/s
106 except:
107     try:
108         worktime = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/main[
109     except:
110         try:
111             worktime = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[1]/m
112         except:
113             try:
114                 worktime = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/div[
115             except:
116                 try:
117                     worktime = driver.find_element(By.XPATH, "//body/div[5]/div[3]/div[1]/div[1]/div[2]/div[1]/
118                 except:
119                     try:
120                         worktime = driver.find_element(By.XPATH, "//body/div[4]/div[3]/div[1]/div[1]/div[2]/div
121                     except:
122                         worktime = driver.find_element(By.XPATH, "//body/div[4]/div[3]/div[1]/div[1]/div[2]/div
123
124
```

Figure 3.4: Handling Exceptions

3.5 Feature Engineering

Feature engineering is the process of transforming raw data into features that can be used to build models. It involves selecting and creating relevant features from the raw data, as well as preprocessing and to ensure that it is in a suitable format for modeling. Feature engineering is a critical step in our project because most of our features are in text format and can't be understood by our model.

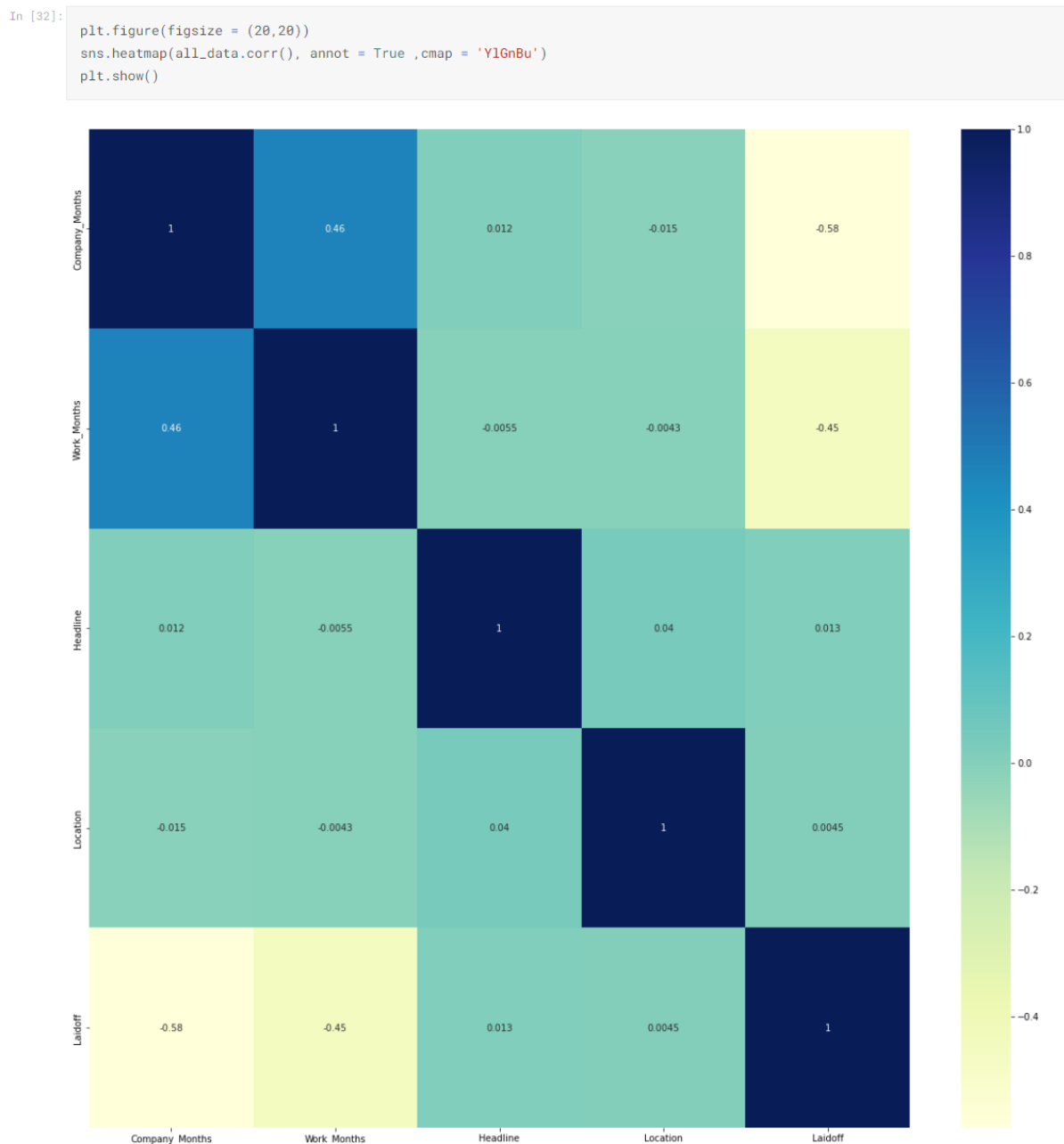


Figure 3.5: Correlation matrix

3.6 Preprocessing

Preprocessing is an important step in the data science process because it can significantly impact the performance of the analysis or model. It involves formatting the data to make it more suitable for use in a particular context.

Preprocessing

Importing Libraries

```
In [33]: from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

```
In [34]: # training_data -> attributes, target -> diabetes
        training_data = all_data.drop(['Laidoff'], axis=1)
        target = all_data[['Laidoff']]

        X_train, X_test, y_train, y_test = train_test_split(training_data, target, test_size=0.2, random_state=10)
```

```
In [35]: print('the number of training entries is: ', X_train.shape[0])
        print('the number of testing entries is: ', X_test.shape[0])
```

the number of training entries is: 1481

the number of testing entries is: 371

```
In [36]: # y_train values:
        y_train.value_counts()
```

```
Out[36]: Laidoff
        False    1291
        True     190
        dtype: int64
```

Figure 3.6: Preprocessing

3.7 Model Building

Model building is the process of creating a statistical or machine learning model using a set of training data. It involves selecting an appropriate model type and tuning its hyperparameters. In our case used the scikit learn library for modeling.

Logistic Regression

In [40]:

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(max_iter=100, solver='lbfgs', class_weight='balanced', random_state=11).fit(X_train, y_train)
y_pred = clf.predict(X_test)

print('\nAccuracy Score: ' + str(clf.score(X_test, y_test)*100) + "%")

mscore.append(['Logistic Regression', clf.score(X_test, y_test)])

print(classification_report(y_test, y_pred))
confusion(y_test, y_pred)
```

Accuracy Score: 100.0%

	precision	recall	f1-score	support
False	1.00	1.00	1.00	315
True	1.00	1.00	1.00	56
accuracy			1.00	371
macro avg	1.00	1.00	1.00	371
weighted avg	1.00	1.00	1.00	371

Figure 3.7: Model Building

3.8 Model to production

Bringing a machine learning model to production refers to the process of deploying the model in a live environment where it can be used to make predictions or decisions. This typically involves integrating the model into an existing application or system, and setting up the infrastructure and processes needed to manage and maintain the model over time. In our case, we save our model in SAV format with Joblib library.

Models to Production 🧑‍💻

```
In [47]: import joblib

# save the models to disk
joblib.dump(clf, "./Models/LR.sav") # load the models from disk
joblib.dump(rnd_clf, "./Models/RF_CLF.sav")
joblib.dump(model, "./Models/XGB.sav")

# some time later...

# load the model from disk
loaded_model = joblib.load("./Models/LR.sav")
result = loaded_model.score(X_test, y_test)
print(result)

1.0
```

Figure 3.8: Exporting Model

3.9 Conclusion

In this chapter we have gone through each single step of the data science project's life cycle. In order to know the different technologies we used in detail, the next chapter will be devoted to the Implementation phase of our application.

Chapter 4

Implementation

4.1 Introduction

After having made a design that best meets the needs of our application, we begin the implementation part of the application that we have developed, by exposing the different development tools and languages used during the production of our application as well as the results obtained.

4.2 Tools Used:

4.2.1 HTML CSS :



HTML, HyperText Markup Language, provides structure and meaning to the content by defining this content by principle of markup like, for example, titles, paragraphs or images.. etc.

CSS, or Cascading Style Sheets, is a presentation language created to style the appearance of content, using, for example, fonts or colors.

4.2.2 TypeScript:



TypeScript is a programming language that is a superset of JavaScript, meaning that it is a variant of JavaScript that includes additional features. It was developed and is maintained by Microsoft. One of the main features of TypeScript is its static typing, which allows developers to specify the data types of variables and function arguments at compile time, rather than at runtime as in JavaScript. This can help to catch errors and improve the maintainability of code. TypeScript also includes features such as interfaces, classes, and decorators, which can make it easier to write object-oriented code. It is often used in the development of large-scale applications and is supported by a number of frameworks and tools, including Angular and React.

4.2.3 Angular:



Angular is a front-end web development framework used for building single-page applications (SPAs). It is based on TypeScript, a superset of JavaScript, and follows the Model-View-Controller (MVC) architectural pattern. Angular provides a set of tools and features for building interactive and dynamic user interfaces, including templates, components, and dependency injection. It also has a built-in support for routing, form validation, and handling HTTP requests. Angular is maintained by Google and is widely used in the development of web applications.

4.2.4 Spring Boot:



Spring Boot Spring Boot is a popular Java-based framework used to build production-grade web applications and services. It is built on top of the Spring Framework and takes an opinionated view of the Spring ecosystem, meaning that it includes pre-configured versions of tools and libraries that work well together.

4.2.5 Python :



Python is particularly popular for data analysis and Machine Learning, but also for backend web development. This language is also used to scrape web data and develop productivity tools. Python comes with a built-in package called json to encode and decode JSON

data. The process of encoding JSON is usually called serialization. This term designates the transformation of data into a series of bytes (therefore in series) to be stored or transmitted on a network.

4.2.6 Flask :



Figure 4.1: Logo - Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

4.3 Development environment

4.3.1 Visual Studio Code :



Figure 4.2: Logo - VS Code

Visual Studio Code is an extensible code editor developed by Microsoft for Windows, Linux, and macOS2.

Features include debugging support, syntax highlighting, smart code completion, snippets, code refactoring, and Git built-in. Users can change the theme, hotkeys, preferences, and install extensions that add additional functionality.

4.3.2 Jupyter Notebook :



Figure 4.3: Logo - Jupyter Notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based, interactive programming environment for authoring Jupyter Notebook documents. The term "notebook" can refer to many different, context-appropriate entities, such as Jupyter web application,

Jupyter Python web server, or Jupyter document format.

4.3.3 XAMPP :



Figure 4.4: Logo - XAMPP

XAMPP is a free, open-source software package that provides a complete web server environment for local development and testing purposes. It includes Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

4.3.4 Version and collaboration management



Figure 4.5: Logo - Git

Git is a version control system that allows developers to track changes to source code and collaborate with other developers on software projects. It allows developers to work on multiple versions of a codebase simultaneously, without fear of overwriting each other's changes.



Figure 4.6: Logo - Github

GitHub is a web-based platform that allows developers to store and manage their Git repositories online. It provides a range of tools and features for collaboration, code review, and version control. It is a popular choice for hosting open-source projects and has become an essential tool for many developers.



Figure 4.7: Logo - Notion

Notion is an application for note taking, databases, Kanban boards, wikis, calendars and reminders. It is both an ideator and a wiki. This software can be used for individual use or in collaboration with others.

4.4 Web Platform Interfaces

This part lists the presentation of the application scenarios of the «**FAANG Layoffs**» web platform.

We will present in the following are screenshots of the main Pages for a User WorkFlow.

4.4.1 Homepage

This interface allows the user to access the different pages of the platform.

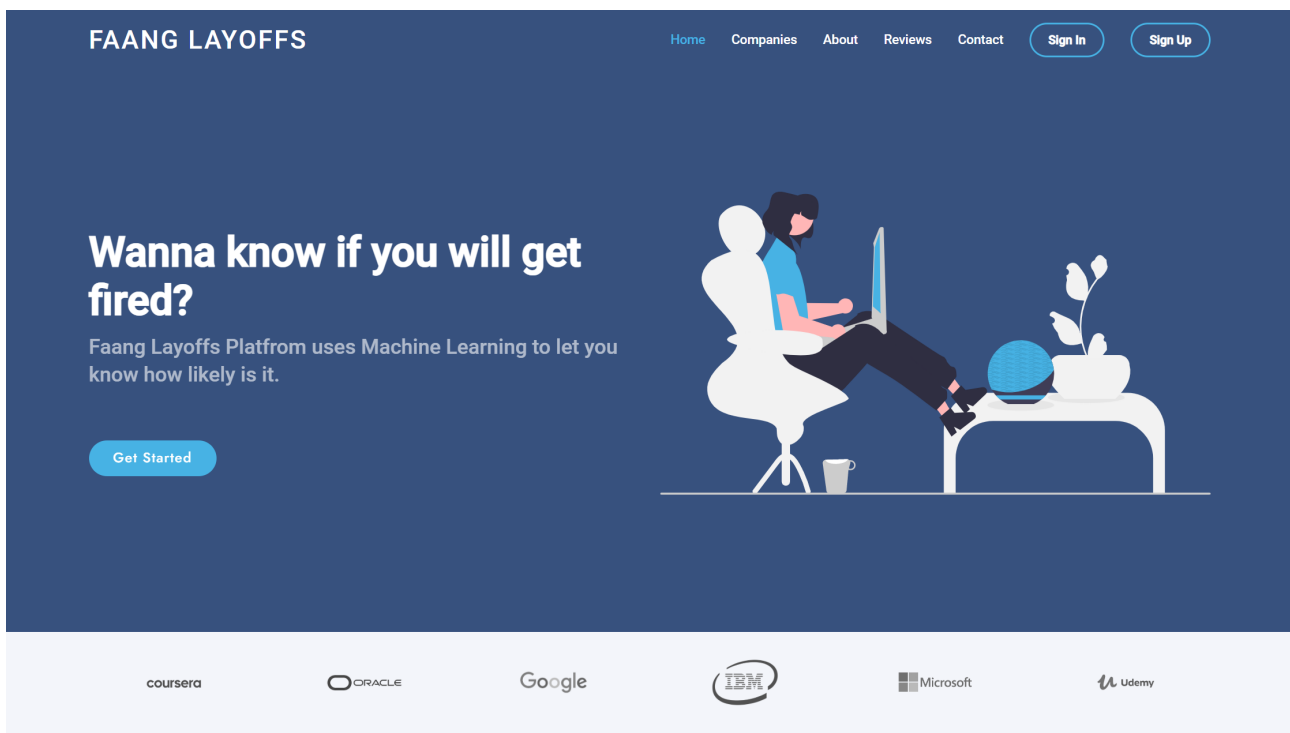


Figure 4.8: Web Platfrom - Landing Page

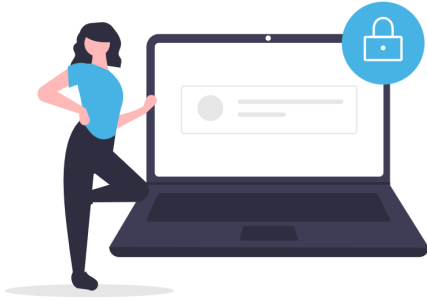
4.4.2 Registration page

The user fills in his data: name, first name, email and password to register.

FAANG LAYOFFS

Sign In

Sign Up



Sign up

First name

Last name

Profil

Employee

Email address

Password

Confirm Password

Register

Log In ?

Figure 4.9: Web Platfrom - Register Page

4.4.3 Login Page

This interface allows the user to connect with his email address and password. He can go to the registration page through the "Need account" button at the end of the interface.

FAANG LAYOFFS

Sign In

Sign Up

Welcome To Faang Layoffs

Login

Email address

Password

Login

Need account ?

Figure 4.10: Web Platfrom - Login Page

4.4.4 Main Interface

After authentication with a Employee account, the user goes to the main screen.

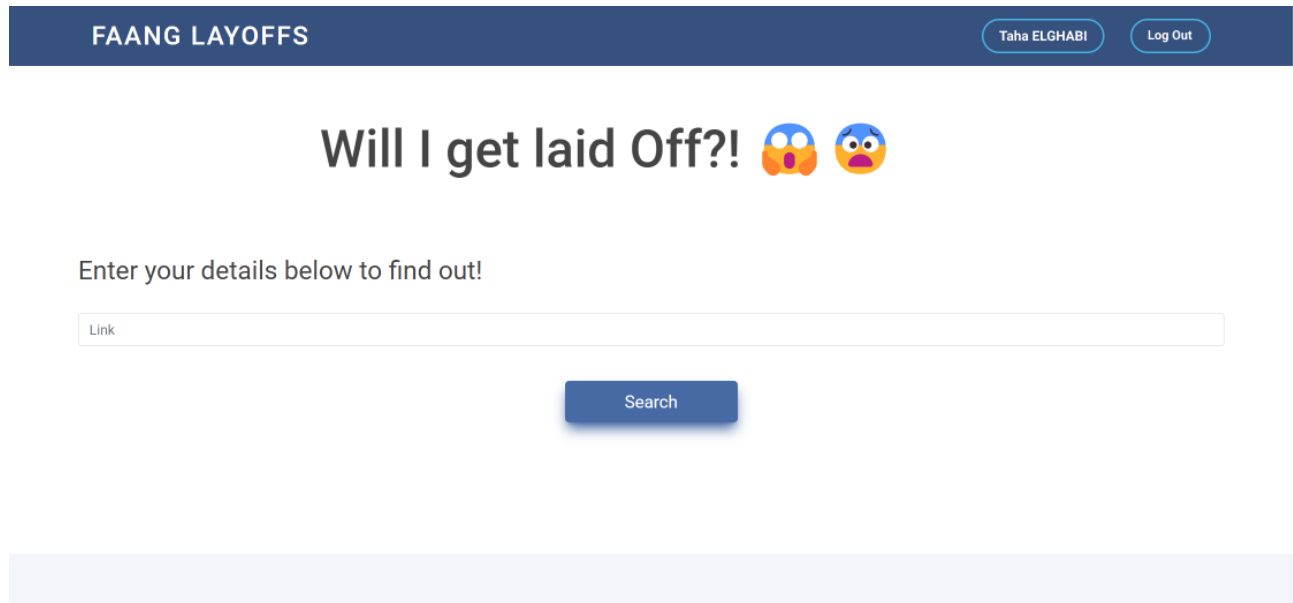
The screenshot shows the main interface of a web platform titled "FAANG LAYOFFS". At the top, there is a dark blue header bar with the title "FAANG LAYOFFS" on the left and two buttons, "Taha ELGHABI" and "Log Out", on the right. Below the header, the main content area has a white background. It features a large heading "Will I get laid Off?!" followed by two surprised face emojis. Below this heading, there is a prompt "Enter your details below to find out!". Under the prompt is a long, thin input field with the placeholder text "Link". To the right of the input field is a blue button with the text "Search". At the bottom of the main content area, there is a light blue horizontal bar.

Figure 4.11: Web Platfrom - Main Interface

This interface contains an input field where the user can paste the url of his Linkedin Profile, a button on under the seach bar which directs make the request to the server.

4.4.5 Waiting Interface

After Clicking the search button the user will be prompted with the waiting interface.

Will I get laid Off?! 🤖 🤖

Enter your details below to find out!

Search

Hold tight! we're scaping 🕒 data for you!



Figure 4.12: Web Platfrom - Waiting Animation

This interface contains a message contain the current status of the request and realtime progress of the stage of request as well as a loading animation.

4.4.6 Page du sourd et/ou muet

Once the server has processed the request, our frontend will receive the response and display the result of the scaping in following interface.

FAANG LAYOFFS

Taha ELGHABI

Log Out

Enter your details below to find out!

https://www.linkedin.com/in/ccheng00/

Search

✓

Name: **Claire Cheng**

Headline : **MIT '22, Ex-Meta, actively looking for SWE roles!**

location : **New York, New York, United States**

Company : **Massachusetts Institute of Technology**

Figure 4.13: Web Platfrom - Scraping Result

Right after our server has the information need to make an inference, the prediction will be send to the interface below:

FAANG LAYOFFS

Taha ELGHABI

Log Out

Enter your details below to find out!

https://www.linkedin.com/in/ccheng00/

Search

✓

Years at Meta : **0.1**

Total Years of Experience : **1.2**

There is 95% chance that you will be laid off

Figure 4.14: Web Platfrom - Prediction Result

General Conclusion

The work we did consisted of creating a web application that predicts the probability of a FAANG employee getting laid off. We did a little analysis of user situations. we also provided the general architecture of our application, including the workflow of user interaction with the application and the details of our Machine Learning model.

This project allowed us to acquire personal and professional experience. It was beneficial to us because we had the chance to improve our knowledge of web development, Spring Boot, Data Scaping, Machine Learning and also to develop a problem-solving capacity since we had a difficulty in choosing the most suitable framework. suitable for our application as well as in the search for solutions to errors which, of course, appear everywhere in the code.

Arriving at this point, we are still ambitious to make improvements that we did not have time to make during the module period, namely:

- User authentication by Linkedin Account
- Build a recommendation system for job offers

Bibliography

- [1] Angular Documentation. <https://angular.io/tutorial>.
- [2] Spring Boot Documentation. <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>.
- [3] Angular Documentation. <https://angular.io/tutorial>.
- [4] Selenium Documentation. <https://www.selenium.dev/documentation/>.
- [5] Article: Angular + Spring Boot JWT Based Authentication. <https://medium.com/@loizen.best/angular-10-spring-boot-jwt-token-based-authentication>.
- [6] StackOverflow. <http://www.stackoverflow.com>.