PROJECT REPORT

Building a Conversational AI Chatbot for Alternative Transportation During Strikes in France

June - July 2023

Report elaborated by:

Juan Olano, Viktor Ivanenko, Shaifali Khulbe, Hamza El Belghiti,

Teofilo Acholla, Muhammad Hafil, Feten Ben

Dans les rues de Paris, les gens se battent,
Les grèves et les tumultes sont monnaie courante.
Mais six travailleurs ont travaillé avec ardeur,
Pour créer une application qui évite la torpeur.

Hamza, Guillaume, Juan, Shaifali, Teofilo et Viktor,
Ont uni leurs forces pour unir leurs savoirs,
Une application dédiée aux discussions,
Pour aider les gens à éviter les perturbations.

Leur chatbot intelligent répond sans relâche,
Aux questions des passants et leur donne la marche,
À suivre pour éviter les rues encombrées,
Et les trajets perturbés, facilement évités.

Ils ont offert cette application en héritage,
À la ville de Paris, pour la soulager de ses présages,
De grèves et de conflits, cause de tension et de stress,
Pour que chacun se déplace en paix, sans jamais de détresse.

Hamza, Guillaume, Juan, Shaifali, Teofilo et Viktor,
Sont les héros modernes de la ville de Paris,
Leurs efforts et leur dévouement ont fait la différence,
Leur chatbot intelligent, leur plus belle récompense.

– GPT 3.5 Turbo: Chanson de geste

EXECUTIVE SUMMARY

The current social crisis in France is resulting in repetitive general strikes. These strikes often have a significant impact on transportation in Île-de-France. For example, train drivers' strikes can lead to many train cancellations and the disturbance of transportation schedules. Strikes led by bus drivers, subway agents, and other public transport workers also lead to a significant disturbance whereas the citizens struggle to arrive at their workplaces, universities, etc

During a strike day, users may struggle to find reliable and accurate information about alternative transportation in Ile-De-France. The current transportation infrastructure lacks an efficient and effective way to provide transport users with personalized information and assistance

The goal of this project is to develop a chatbot application that helps the citizens in the Ile-De-France by providing them with reliable and accurate information about alternative transportation on strike days

INTRODUCTION & PROBLEM STATEMENT

The Omdena Ile-De-France local chapter addresses the field of traveler information in Île-de-France regarding strike management. When there are strikes in public transportation, it becomes essential to provide travelers with up-to-date and reliable information to help them plan their journeys effectively. In this regard, data collection from websites becomes an essential resource to obtain the necessary information in real-time.

In Île-de-France, strikes in public transportation can have a major impact on the daily mobility of travelers. It is crucial to have accurate information about disruptions, modified schedules, alternative services, and other relevant details. By using scraping techniques, we can extract this information directly from the websites of transport operators and other reliable sources.

Therefore, this project aims to collect and analyze data from various websites to provide travelers with the necessary information during strikes. By extracting real-time data on public transportation schedules, service modifications, and alternative routes, we will be able to generate valuable and updated information to help travelers make informed decisions and mitigate the impacts of strikes on their journeys.

In this report, we will present in detail our scraping methodology, the challenges we encountered during data collection, and the results of our analysis. We will also highlight the importance of information obtained through web scraping in improving strike management in public transportation in Île-de-France.

DATA COLLECTION

In order to provide comprehensive and up-to-date information regarding transportation and strikes, our data collection efforts focused on gathering a wide range of relevant data sources. These sources were carefully selected to ensure that all aspects of transportation, including strike-related information, were captured. The collected data encompasses various sources, including APIs, websites, and geographic databases, to gather details on public transportation, Vélib stations, upcoming strikes, and street networks. By incorporating these diverse data sets, we aim to provide travelers with comprehensive and accurate information to assist them in planning their journeys amidst transportation disruptions and strike events. Here are the data sources we have used:

- RATP Stations informations: https://prim.iledefrance-mobilites.fr/fr/donnees-statiques/emplacement-des-gares-idf (Contains informations like Geo point of the station, name, mode (RER, Transilien, Metro...), line (D, B, C...), exploitant (SNCF, RATP...)
- BUS Stations information: https://prim.iledefrance-mobilites.fr/fr/donnees-statiques/amenagements-bus-en-ile-de-france (Contains information like type line, department id...)
- VELIB Stations: https://velib-metropole-opendata.smoove.pro/opendata/Velib_Metropole/station_status.json and https://velib-metropole-opendata.smoove.pro/opendata/Velib_Metropole/station_information.json
- Traffic Status: Scraped from https://www.ratp.fr/ (Contains informations about each line and their corresponding traffic status, ex "RER A, Traffic normal")
- Strikes: Scraped from https://www.evous.fr/Les-Manifestations-a-Paris-la-semaine-1176044.html (strikes by week)

However, we encountered several constraints during the process:

- Lack of Traffic Status Information: The system did not provide real-time traffic status updates for bus lines and users interested in using bikes (VELIB). Consequently, it was unable to suggest the most suitable routes for them.
- Insufficient Information about Bus Stations and Lines: The available data on bus stations and lines, particularly their destinations, was inadequate. This limited the system's ability to provide comprehensive information to users.
- Unreliable Web Scraping for RATP Station Traffic Status: The web scraping technique employed to retrieve traffic status information for RATP stations proved to be unreliable, leading to inconsistent and inaccurate data.
- Absence of Departure and Arrival Schedules: The system lacked the functionality to display departure and arrival schedules for various transportation modes such as metro, bus, and RER.

To address these limitations, a suitable solution was sought, leading to the discovery of the Navitia API (https://navitia.io/). Navitia is an open-source web API initially designed to provide traveler information on urban transportation networks. It offers several advantages over the previous approach:

- Comprehensive Disruption Information: Navitia provides access to disruptions across all types of public transportation, making it a more reliable alternative to web scraping methods.
- Departure and Arrival Times: The API offers accurate departure and arrival times for different modes of transportation.
- Detailed Journey Information: Navitia stands out by offering complete journey details, considering the user's preferred mode of transportation (metro, bus, RER), current mode of transportation (slow walker, bike, luggage), step-by-step directions, departure and arrival times, and a comprehensive map of the route with expected duration and distances.

To retrieve the desired information, a call is made to the Navitia API, specifying parameters such as departure and destination geographic points, region (Île-de-France), traveler type, commercial mode, and the maximum number of suggested journeys. The API responds with a JSON-based response, which can be processed and utilized within the system. The collected data does not contain any personal information. Therefore, we are not bound by privacy compliance requirements.

DATA PROCESSING

Before incorporating Navitia, we developed an initial version (**Version 1** in our project) of the application using the data sources mentioned in the data collection section. In this version, we have implemented a data processing workflow for RATP stations. We converted a JSON file into a pandas dataframe, adjusting the transport type to match the data obtained from scraping the traffic information on the RATP website. For web scraping, we utilized BeautifulSoup and set up an AWS Lambda function that runs every 5 minutes, storing the output as a CSV file in an AWS S3 bucket. To find the nearest RATP stations, we load the traffic status data from the S3 bucket using the aws boto3 library, merge it with the existing station information based on the shared "name" field (e.g., "RER A"), and apply a proximity algorithm to determine the nearest viable station to the user's location. This approach ensures regular and accurate updates of RATP station information.

Subsequently, we developed another version that relies on Navitia (**Version 2**). In this version, Navitia returns a well formatted json file, and the only thing we have to do is a little processing of the durations (from a format in seconds to an Hour: Min:Sec format) and of the departure times and arrivals times also (conversion to datetime).

MODELING AND ALGORITHMS

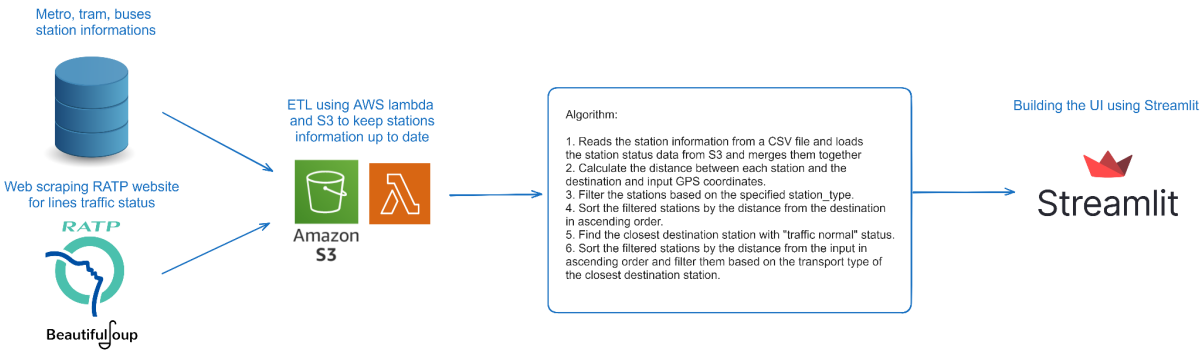
Explanation of the algorithms used to determine alternative routes

Discussion on how strike information is integrated into the model

Considerations regarding performance and accuracy

In this section, we will present the two versions of the architecture of our application.

1. **Architecture Chatbot Version 1**



The architecture of the app in **Version 1** involved a multi-step process to gather and analyze data for providing optimal route suggestions to users. Initially, station information such as location, transportation type, and associated lines was manually collected. Then, a web scraper was implemented using BeautifulSoup to extract real-time traffic updates from various websites. This allowed the app to stay updated on disruptions or constructions affecting different lanes. The scraped data was stored in CSV format using AWS Lambda and Amazon S3 buckets.

Leveraging AWS S3 and Python's S3 Boto3 library, the app retrieved the data and performed an algorithmic analysis. The algorithm calculated the distance between each station destination and the input GPS coordinates, filtering and sorting stations based on proximity and traffic status. The goal was to identify the closest station with favorable traffic conditions, ultimately providing users with an optimal route. This architecture ensured that the app could deliver reliable and efficient navigation solutions to its users.

In order to connect our system and models to the real world, we used a technique called RAG (*Retrieval Augmented Generation*).

RAG is a framework for building systems that make use of external data sources. RAG is a great way to connect via API and other sources of our application to the real world.

Our model OpenStreetMap to determine the shortest path between the user's starting point and their destination, considering not only the physical distance but also any disruptions due to strikes. Information about strikes is integrated into the model by marking any affected stations or routes as inaccessible with information obtained from the APIs, forcing the model to reroute around these problem areas.

We also added our own “Distance Calculator” as seen in the following python code:

DISTANCE_CALCULATOR

Snippet of the code:

```
# Created by Viktor Ivanenko
from geopy import distance
import numpy as np

def distance_calculator(*args):
     """ Takes all the route locations, points coordinates and
returns summary distance. Could deal with unlimited points.
Points coordinates should be inputed within a list argument."""

    list_args = []
    list_args.append(*args)
    indx=len(*args)
    dist_list = []
    for ind in range(indx):
        if ind > 0:
                    dist  =  distance.distance(list_args[0][ind-1],
list_args[0][ind])
```

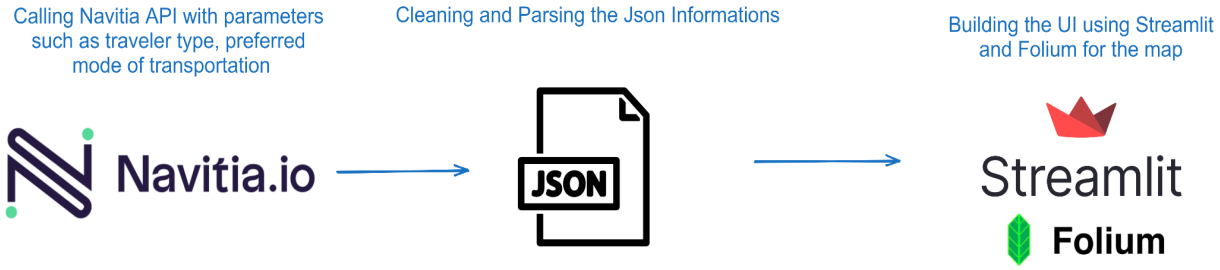
```
            dist_list.append(dist)

    return np.sum(dist_list)

# Checkers
print(distance_calculator([(41.49008,   -71.312796),   (41.499498,
-81.695391), (41.499498, -71.312796)]))
print(distance_calculator([(41.49008,   -71.312796),   (41.499498,
-81.695391), (41.499498, -71.312796), (41.495432, -81.695391)]))
print(help(distance_calculator))
```

The model's performance was evaluated by measuring the accuracy of proposed routes and the processing time for each query.

2. **Architecture Chatbot V2**



In **Version 2** of the app, a new architecture was implemented to enhance the user experience and provide more flexibility in choosing preferred travel parameters. The project incorporated Navitia, an API that allows users to specify their preferred mode of transportation, such as car, bike, or walking. By calling the Navitia API, the application retrieved a JSON file containing relevant travel information. The JSON data was then processed and cleaned to extract the necessary details.

The project utilized Streamlit, a Python library for building interactive web applications, to display the extracted information to users.

Additionally, the architecture employed Folium, a mapping library, to showcase the routes visually on interactive maps. With this architecture, users can easily customize their travel preferences and view the suggested routes in a user-friendly interface.

The integration of Navitia, Streamlit, and Folium improved the overall functionality and visual presentation of the application, making it more convenient and engaging for users.

GRAPHICAL INTERFACE DEVELOPMENT

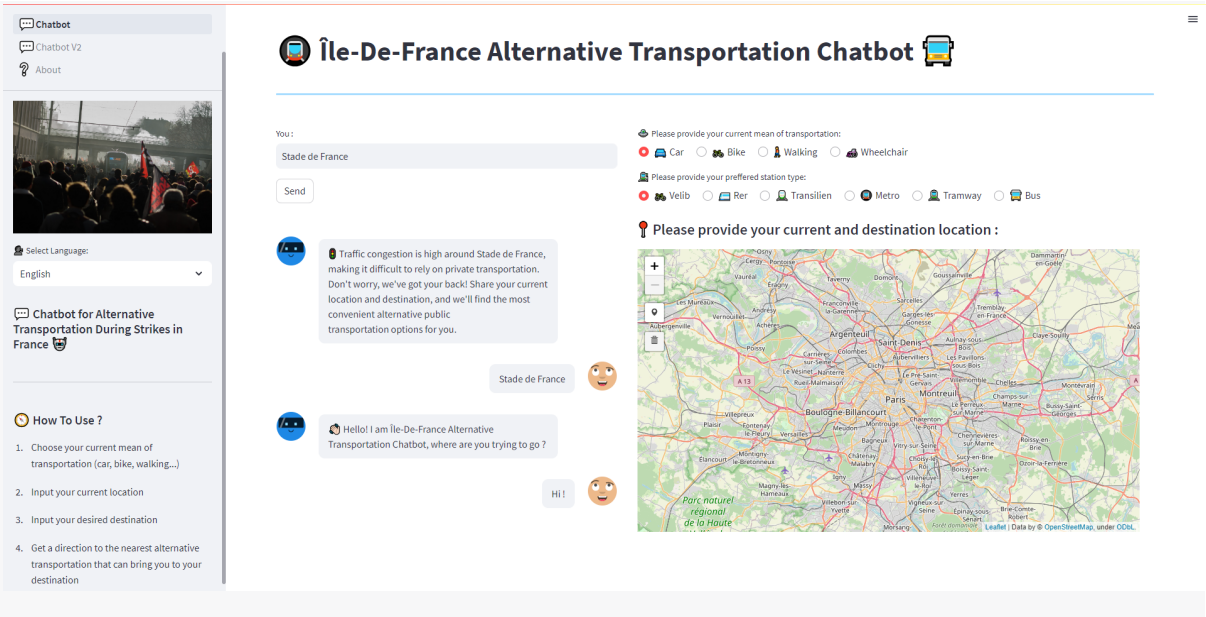
Description of the architecture and tools used to create the graphical interface

Explanation of the features provided to users (source and destination selection, route visualization, etc.)

Screenshots or demonstration of the developed graphical interface

The graphical interface was developed using Streamlit, an open-source app framework for Machine Learning and Data Science teams, and Folium for mapping purposes.

The graphical interface of our project is designed to provide a user-friendly and intuitive experience for travelers



Upon launching the application, users are welcomed by a well-structured user interface comprising three distinct sections. The left section encompasses various application features, with a primary focus on language selection.

We provide language options such as **French**, **English**, and **Spanish**, catering to the preferences of our target audience, who primarily utilize these languages.

The central section of the interface is dedicated to the chatbot functionality, allowing users to interact and seek assistance for their travel-related queries. This interactive

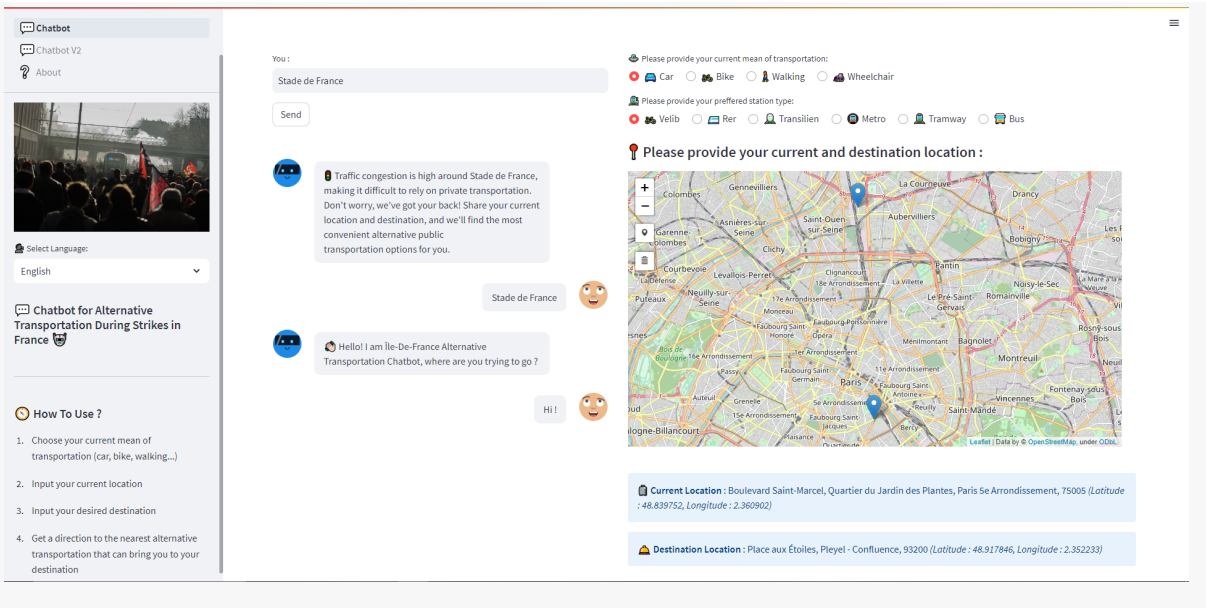
component enhances the user experience by providing real-time responses and guidance.

On the right side of the interface, users are presented with the flexibility to select their preferred mode of transportation. The options that the app provides are: Car, Bike, Walking, Wheelchair.

The app also offers different options regarding the preferred type of station. The options that the app provides are: Velib, RER, Transilen, Metro, Train, and Bus.

Additionally, a dynamic map is prominently displayed, empowering travelers to easily identify their current location and desired destination. This visual representation facilitates efficient route planning and navigation.

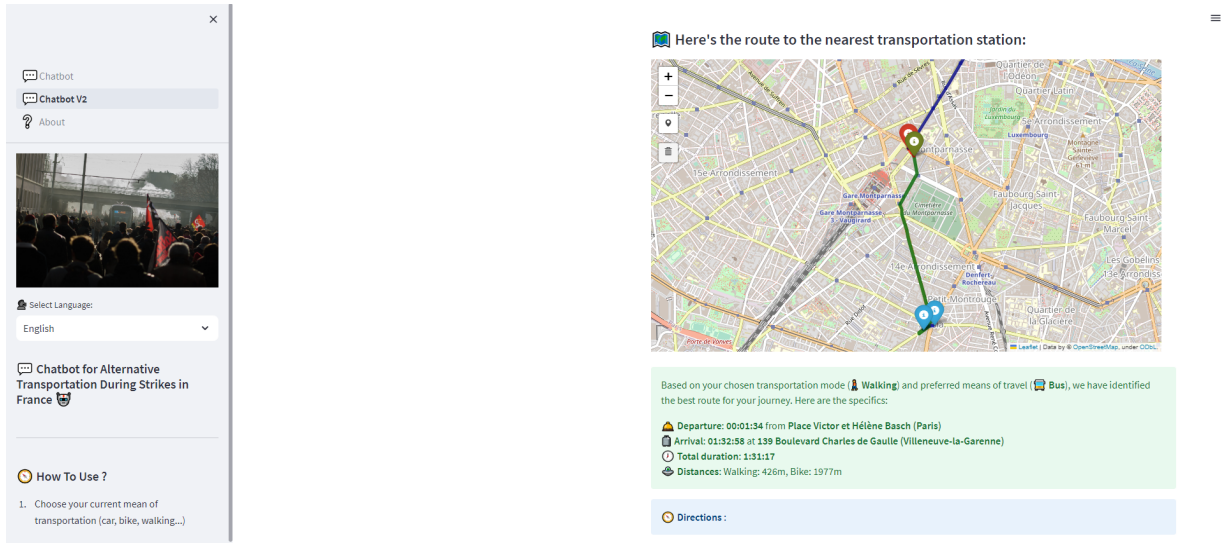
The figure below illustrates the visual layout of the interface, emphasizing the arrangement of these sections for improved usability and intuitive interaction

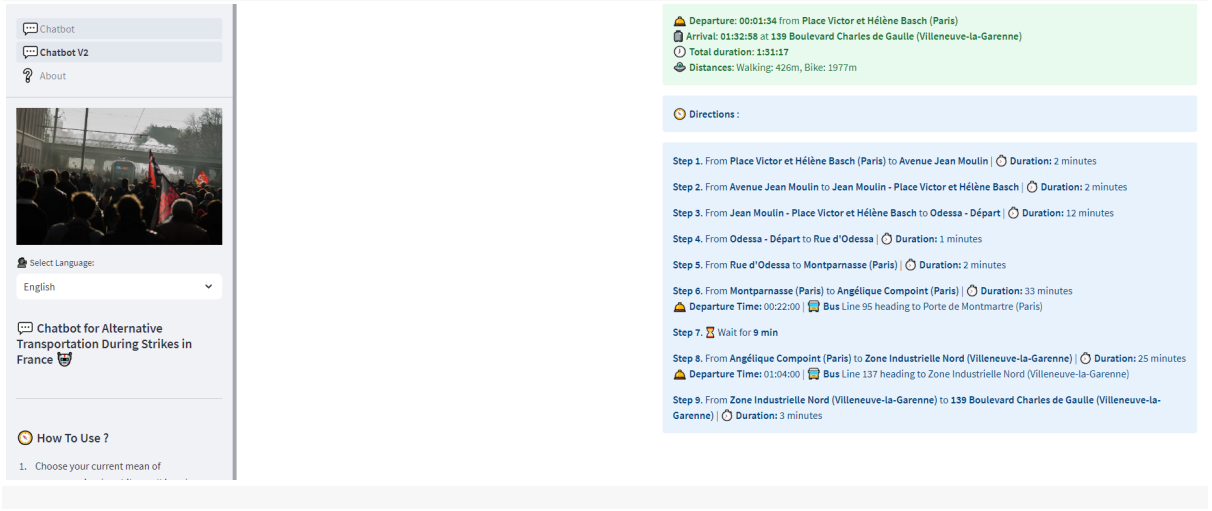


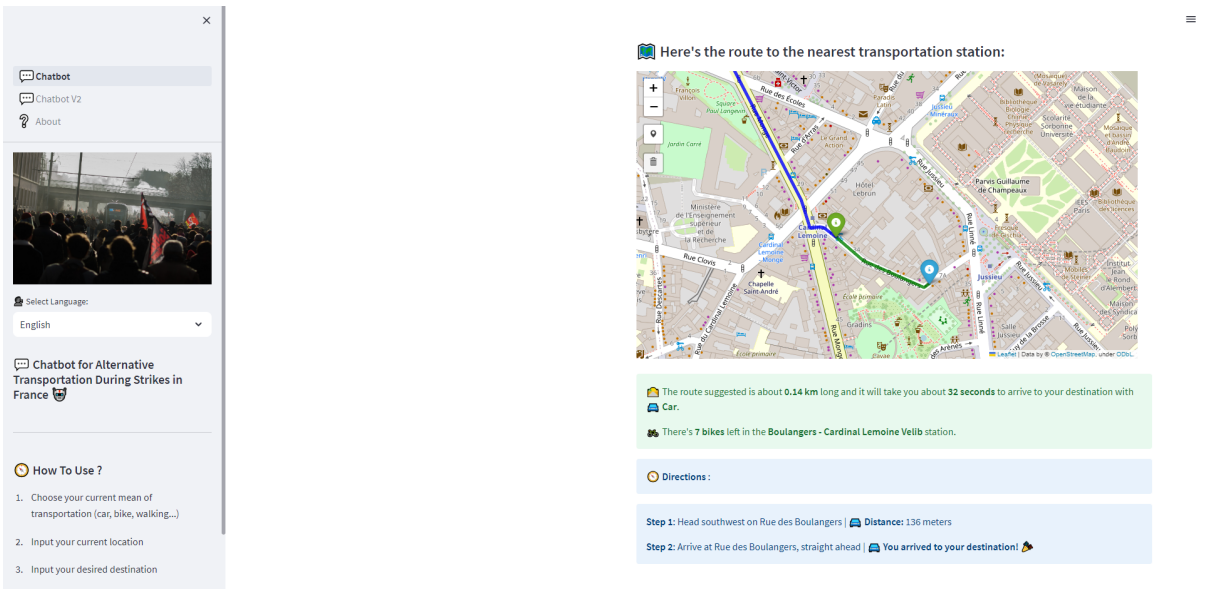
Once the traveler has selected their current location and destination, the interface provides them with the nearest station. The application displays a response indicating any system disruptions, if applicable, and offers an alternative response as well. It also shows the estimated distance to the destination and the approximate time required to reach it.

For travelers seeking detailed directions, the application offers step-by-step instructions. These directions are presented in a consistent format, facilitating navigation throughout the process without the need for constant interaction with the application.

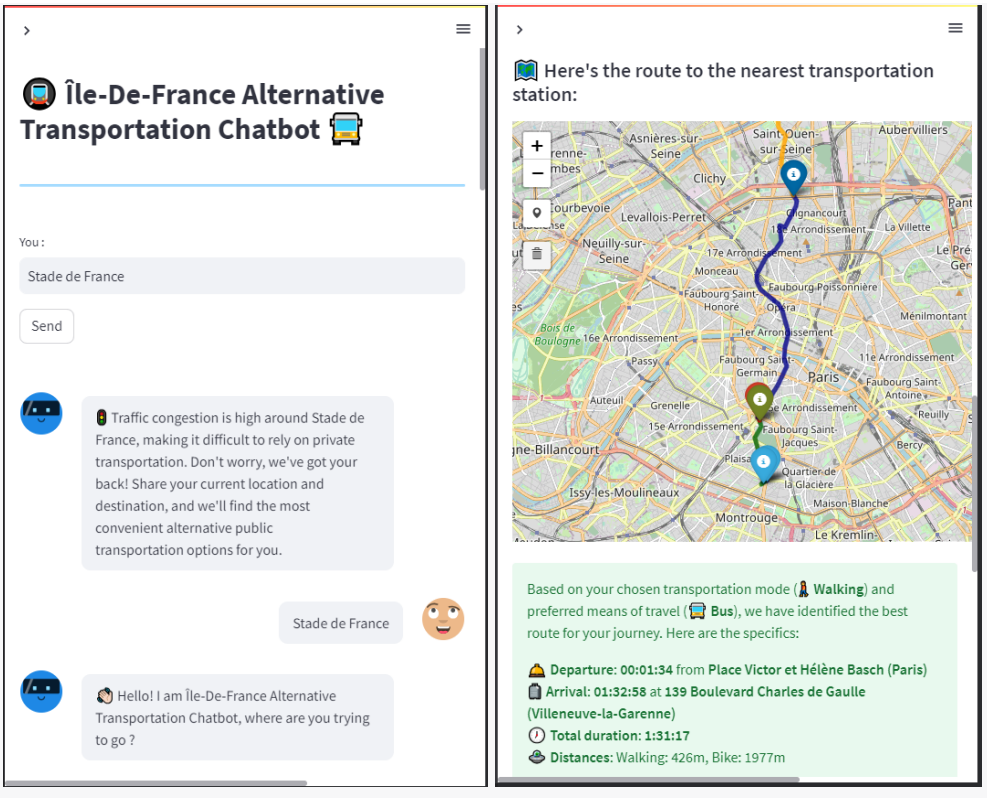
The following figures provide examples of the proposed itineraries. One demonstrates the preferred mode of transportation, cycling, while the other illustrates the bus option.





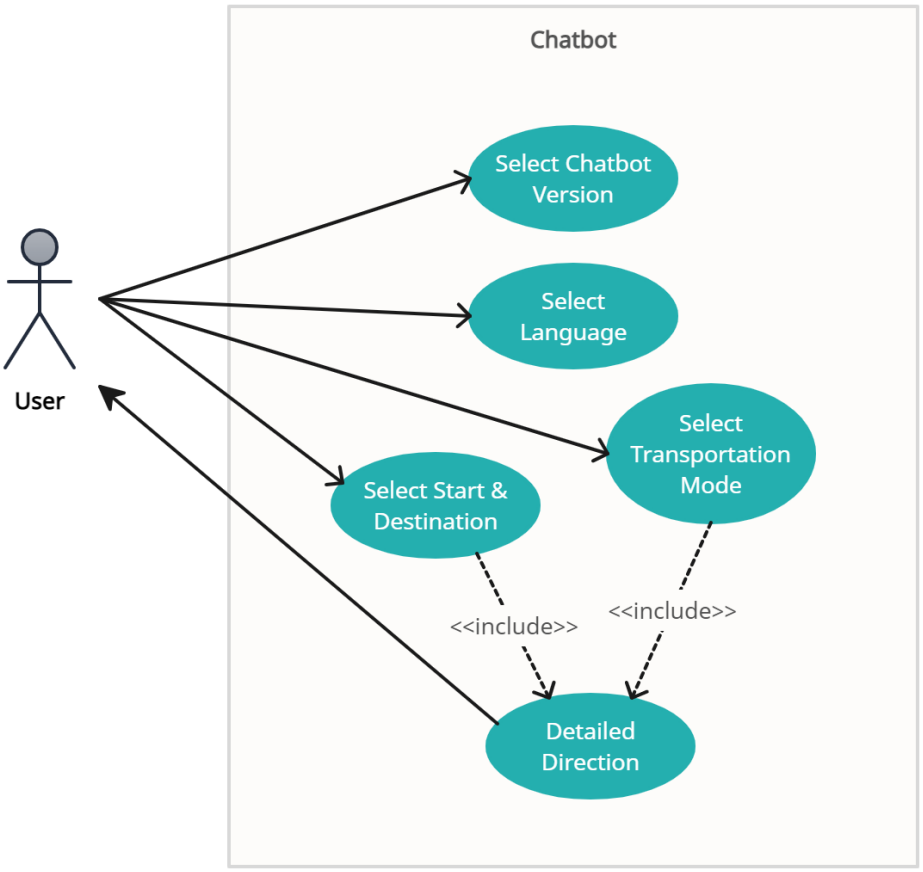


In addition, our application has been specifically tailored for mobile devices, ensuring a seamless user experience across various platforms.



In conclusion, our graphical interface provides a seamless and user-friendly experience. It offers clear information regarding destinations, routes, travel times, and transportation options. With its simple design and intuitive features, our application is specifically designed to meet the needs of users in their daily travels.

USE CASE DIAGRAM



CHATBOT IMPLEMENTATION

Presentation of the technologies and tools used to create the chatbot

Description of the chatbot's functionalities (handling user queries, managing dialogues, etc.)

Examples of simple phrases that the chatbot can understand and process

The chatbot was built using pre-defined conversation templates that are injected with variable information depending on each case. The chatbot can handle user queries, manage dialogues, and deliver responses in a conversational manner. Simple phrases that the chatbot can understand include "I need to go to [destination]" and "What's the fastest way to get to [destination]?".

The following predefined prompt templates were used:

1. The recommended path spans 3.3 kilometers and you can reach the closest station after walking for roughly 00:38:45 hours.

2. To arrive at the nearest station on foot, the suggested course of action involves covering a distance of 3.3 km and a time span of approximately 00:38:45 hours.

3. The shortest suggested route to the nearest station can be covered on foot, with a distance of 3.3 km and an estimated time of 00:38:45 hours.

4. According to our recommendation, you can reach the nearest station by covering a distance of 3.3 km on foot, which will take you around 00:38:45 hours.

5. Our suggestion involves taking the 3.3 km path on foot to reach the closest station, which can take approximately 00:38:45 hours.

6. The route recommended for traveling on foot to the nearest station spans 3.3 km, taking approximately 00:38:45 hours to travel.

7. The shortest route suggested to reach the nearest station is an estimated 3.3 km and will require about 00:38:45 hours of walking.

8. Our proposal involves walking for roughly 00:38:45 hours on a 3.3 km path to reach the closest station.

9. The fastest recommended way to get to the nearest station is by walking a distance of 3.3 km, which

EVALUATION AND RESULTS

Evaluation measures used to assess the system's performance (accuracy of proposed routes, responsiveness of the chatbot, etc.)

Results obtained during testing and evaluations

Discussions on limitations and possible improvements

The latest version of our BOT can be found here:

FINAL BOT

The system's performance was evaluated based on the accuracy of proposed routes, the chatbot's response time, and overall user satisfaction. Results from testing and evaluations done by team members with hypothetical information showed a high degree of accuracy in the proposed routes and a quick response time from the chatbot.

There are, however, limitations such as the accuracy of the information on strikes and potential delays in data update.

CONCLUSION

Recapitulation of the project accomplishments

Reflection on the effectiveness of the developed system

Suggestions for future developments

The project accomplished its goal of developing a chatbot application to assist citizens in Île-de-France during strike days by providing reliable and accurate alternative transportation information.

The system has proven effective in providing real-time updates and proposing optimal travel routes, considering possible disruptions due to strikes. It is important to note that the reliability and real-time performance of the system highly depends on the data sources that the system is using, as described in the DATA COLLECTION section above.

Future developments could include expanding the coverage area beyond Île-de-France, improving the chatbot's natural language processing capabilities, and integrating other forms of transport like car-sharing services.

REFERENCES

List of data sources, scientific articles, or resources used in the project

https://omdena.com/local-chapters/ile-de-france-france-chapter/

https://www.ratp.fr/

https://prim.iledefrance-mobilites.fr/

https://doc.navitia.io/

https://www.iledefrance-mobilites.fr/aide-et-contacts/velib

https://worldinparis.com/transport-in-france-strike-news-tips-for-traveling-to-paris

https://www.sortiraparis.com/en/news/in-paris/articles/288815-march-28-2023-strike-carpools-offered-in-paris-and-the-ile-de-france-region

https://folium.streamlit.app/

https://www.geeksforgeeks.org/working-with-geospatial-data-in-python/

https://www.learndatasci.com/tutorials/geospatial-data-python-geopandas-shapely/

https://towardsdatascience.com/automated-scraping-using-aws-lambda-with-layers-aws-s3-and-current-superbowl-odds-dc7f9c6d27b1

https://py-googletrans.readthedocs.io/en/latest/

https://docs.streamlit.io/knowledge-base/tutorials/build-conversational-apps