Université des Mascareignes

Faculté des technologies de l'information et de la Communication

Bsc Hons Informatique Appliquée

KAI

Kreol Artificial Intelligence

Soumis par : BALLAH Vayuna

Norbert Ronan Adrien

Soumis à : Mr. Shiam Beeharry

Introduction

Welcome to KAI, KAI is an abbreviation of Kreol Artificial Intelligence, your ideal travelling partner while you explore the captivating island of Mauritius! Kreol Artificial Intelligence, or KAI for short, is an innovative voice assistant program created especially to improve visitors' experiences to Mauritius. With its cutting-edge features and user-friendly interface, KAI seeks to enhance, enlighten, and make your voyage through this tropical paradise genuinely remarkable.

Personalised assistance and real-time guidance are provided to travellers by KAI, utilising artificial intelligence and natural language processing to help them navigate the island with confidence and ease. KAI can assist you with all of your needs, whether you're looking for advice on the best beaches, are in the mood for some authentic Mauritian food, or are trying to find some hidden treasures off the usual route.

KAI provides thorough insights into Mauritius's rich history, diversified culture, and stunning natural beauty, encompassing everything from cultural landmarks to exciting excursions. Users can access a plethora of information with just a voice command, including historical details, regional customs, forthcoming events, and insider advice selected by experienced tourists and informed residents.

However, KAI is more than simply an online tour guide; it's a customised concierge service that learns to anticipate your needs and fits your particular way of travelling. Whether you're a passionate traveller, a lover of quiet, or a history nerd, KAI customises its suggestions to fit your preferences, making sure that every second of your vacation is spent as you see fit.

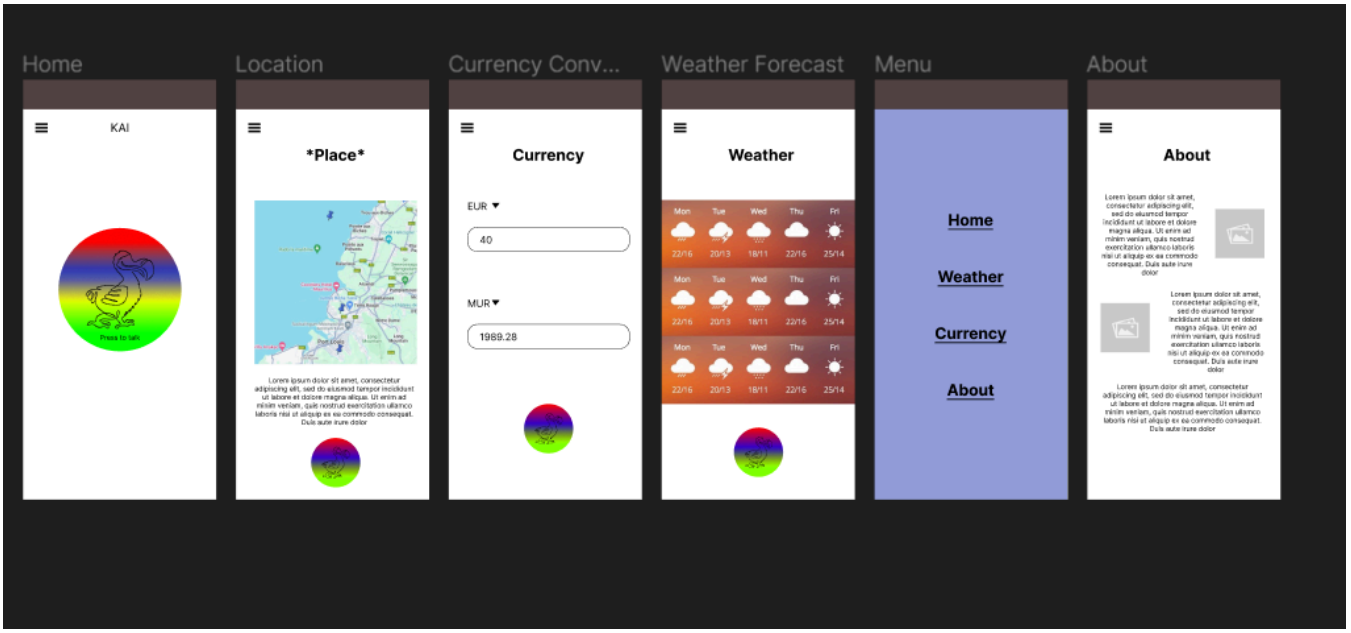
WIth KAI by your side you can confidently explore Mauritius, knowing that you have a trustworthy guide to help you every step of the way. Thus, take a seat back, unwind, and allow KAI to be your dependable guide to discovering Mauritius' beauties. This is where your journey starts.

WireFrame and Prototype

Wireframe

Prototypes and wireframes are crucial steps in the design process that have different but complementary roles to play in the development of digital goods. A web page, application screen, or interface's fundamental structure and layout are defined by wireframes, which serve as skeletal outlines. Without including specific design elements like colours or graphics, they only offer a visual depiction of where buttons, text, images, and other UI components will be positioned. Wireframes ensure clarity and efficiency in the design process by allowing designers and stakeholders to swiftly iterate on layout and flow by focusing on the basic structure. They act as a roadmap for future design decisions and act as a blueprint for the overall design direction, assisting in the alignment of stakeholders.labor.

The Wireframe of the application:



For our wireframe we created our own button with a gradient background of red , blue ,yellow and green which represents the flag of Mauritius and a Dodo bird on it as the dodo bird is our national bird . Our wireframe consists of screen like the home page which is where the main voice application is and a google maps screen so that tourist can access various places in Mauritius and visit restaurants and other attraction places , furthermore we also had a currency converter screen so as to ease the access for the tourist to be able to convert their currency into MUR. Our wireframe also consists of a weather forecast screen , menu screen and a about us screen. These screens are mainly there to provide the tourist with everyday weather forecast of Mauritius so as they can plan their schedule for the day and a menu so as the user can navigate through the options that are available and lastly a about us screen so as the user can know more about us and the application and about KAI (Kreol Artificial Intelligence) app.

Prototype

Conversely, prototypes are interactive models of a digital product that mimic its operation and user interface. Prototypes, as opposed to wireframes, allow users to interact with the interface just as they would in the finished product by going beyond static layouts. Prototypes can be anything from high-fidelity simulations that closely mimic the finished product to low-fidelity mockups with rudimentary functionality. They allow designers to get user feedback, test and confirm design assumptions, and spot usability problems early in the development process. Prototypes are essential for fine-tuning user flows, evaluating how well interactions work, and making sure the finished product fulfils user requirements and expectations.

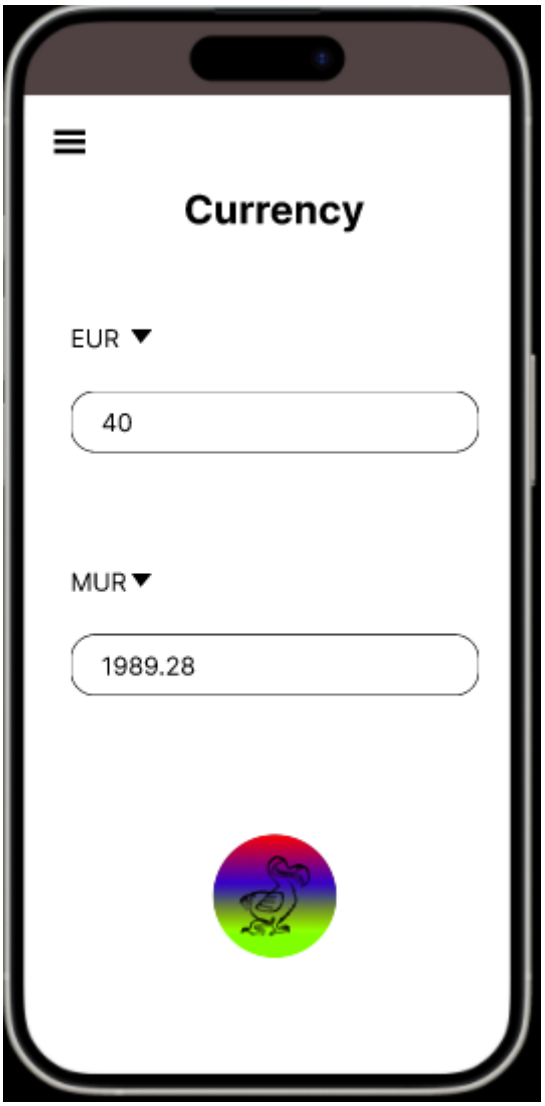
Prototypes and wireframes combined make up a potent toolkit that gives designers an organised method for creating and developing digital products. Prototypes give the design life and enable interactive exploration and validation, while wireframes create the groundwork by specifying the structure and layout. Designers may produce more unified and user-centred experiences by including wireframing and prototyping into the design process. This will eventually result in the creation of successful digital goods that satisfy the needs of their target audience.

Prototypes Below :

UNIVERSITÉ DES MASCAREIGNES

SAVOIR, C'EST POUVOIR

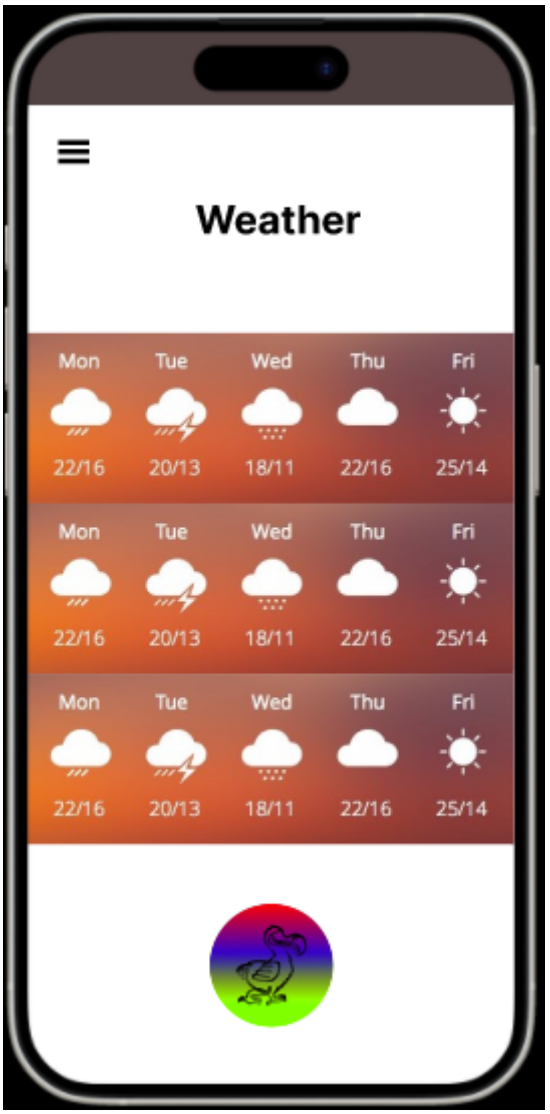
Université de Limoges


Currency
EUR
40
MUR
1989.28

UNIVERSITÉ DES MASCAREIGNES

SAVOIR, C'EST POUVOIR

Université de Limoges




Home
Weather
Currency
About

UNIVERSITÉ DES MASCAREIGNES
SAVOIR, C'EST POUVOIR

Université de Limoges

Methodology

The code is based on a simple methodology which consists of recording the voice of the user when the button is pressed, convert the record from speech to text through a library having the name “speech_to_text” and sending the text to the “openai_service” file.

```
FloatingActionButton.extended(        You, 3 days ago • homepage upda
  backgroundColor: Colors.transparent,
  onPressed: () async {
    if (await speechToText.hasPermission &&
        speechToText.isNotListening) {
      await startListening();
    } else if (speechToText.isListening) {
      final speech = await openAIService.chatGPTAPI(lastWords);
      if (speech.contains('https')) {
        generatedContent = null;
        setState(() {});
      } else {
        generatedContent = speech;
        displayText = speech;
        setState(() {});
        await systemSpeak(speech);
      }
      await stopListening();
    } else {
      initSpeechToText();
    }
  },
```

There, the text is sent to an instance of ChatGPT through an API Key.

```
FloatingActionButton.extended(        You, 3 days ago • homepage updat
  backgroundColor: Colors.transparent,
  onPressed: () async {
    if (await speechToText.hasPermission &&
        speechToText.isNotListening) {
      await startListening();
    } else if (speechToText.isListening) {
      final speech = await openAIService.chatGPTAPI(lastWords);
      if (speech.contains('https')) {
        generatedContent = null;
        setState(() {});
      } else {
        generatedContent = speech;
        displayText = speech;
        setState(() {});
        await systemSpeak(speech);
      }
      await stopListening();
    } else {
      initSpeechToText();
    }
  },
```

After answering the query, ChatGPT sends back an answer to the program in the form of a text. This text is then sent from the “openai_service” file to the “Home_Page” file where it will be displayed in a text area under the button.

```dart
Future<String> chatGPTAPI(String prompt) async {
  messages.add({
    'role': 'user',
    'content': prompt,
  });
  try {
    final res = await http.post(
      Uri.parse('https://api.openai.com/v1/chat/completions'),
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer $openAIAPIKey',
      },
      body: jsonEncode({
        "model": "gpt-3.5-turbo",      You, yesterday • openai functionality
        "messages": messages,
      }),
    );

    if (res.statusCode == 200) {
      String content =
          jsonDecode(res.body)['choices'][0]['message']['content'];
      content = content.trim();

      messages.add({
        'role': 'assistant',
        'content': content,
      });
      return content;
    }
    return 'An internal error occurred';
  } catch (e) {
    return e.toString();
  }
}
```

The text generated by Artificial Intelligence is, at the same, going the other way from the beginning and is converted to speech through the library “text_to_speech” or “flutter_tts”.

Problems

Issues encountered :

1. **Android Emulator**

   Unexpected issues occurred when trying to execute the KAI application using an Android Studio emulator; this resulted in a system crash and the freezing of all running applications, notably Visual Studio Code. The system's stability was unexpectedly disrupted as a result of the emulator's inability to execute the KAI application without hitting compatibility difficulties or resource limits. The incident highlighted the complexities and potential drawbacks of virtual environments in software development workflows, even though the primary plan was to use emulation for testing or development purposes. As a result, efforts could be focused on other approaches or optimizations to guarantee better program functionality and integration in emulation settings without jeopardising system stability overall.

2. **OpenAi chat**

   OpenAi API key code:

```
const openAIAPIKey =
'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'
;
```

   Even though using OpenAI's chat feature first seemed appealing, the project was hindered by a major drawback: the given API key expired, making it difficult to use. The platform's continuous expiration problem prevented any meaningful use from the beginning, even with attempts to create a new key. This intrinsic challenge resulted from the fact that although OpenAI's services were theoretically free, practical use was impossible due to the expiration barrier. As a result, even with the potential advantages that OpenAI's technology could provide, the failure to get past this barrier made the investigation of other options necessary, highlighting the difficulties that arise when using publicly accessible but time-limited resources for important applications.

3. Google maps

Google's Map API key code :

```
const String googleAPIKey = 'https://maps.googleapis.com/maps/api/js?sensor=false&callback=myMap';
```

Even though we had written all the programming required to include Google Maps capability into our application which was made especially for the Mauritius region and had all the features we wanted we ran across an unbreakable barrier. Getting an API key from Google was a difficult process because you had to pay for it or input a Visa card for authentication, which was required in order to use the Google Maps API. For us as students, this requirement provided a significant disadvantage because it placed a financial barrier that we were unable to surmount. Therefore, even though we were prepared to use Google Maps services to improve our application, we were unable to move forward with it due to a crucial limitation: the requirement that we pay for the API key.

Dependencies and Imports

Dependencies

For KAI Applications :

```yaml
dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.2
  speech_to_text: ^6.6.1
  http: ^1.1.0
  text_to_speech: ^0.2.3
  flutter_tts: ^3.8.5
```

- **Flutter**: This is the basic tool for making Flutter apps. It's like the foundation of the project.
- **Cupertino Icons:** These are icons used for making iOS-style apps. Think of them as small pictures you can use in your app.
- **Speech to Text:** This tool lets your app understand what someone says out loud and turn it into written words.
- **HTTP:** This allows your app to talk to other websites and get information from them.
- **Text to Speech:** This lets your app turn written words into spoken words.
- **Flutter TTS**: This is another tool for turning written words into spoken words, but it might have some extra features compared to the regular text-to-speech tool.

Imports

```
import 'package:flutter/material.dart';
import 'package:kai_app/openai_service.dart';
import 'package:speech_to_text/
speech_recognition_result.dart';
import 'package:flutter_tts/flutter_tts.dart';
import 'package:speech_to_text/speech_to_text.dart';
import 'dart:convert';
import 'package:kai_app/home_page.dart';
import 'package:kai_app/pallete.dart';
```

Flutter Material: This line imports the Flutter Material package, which gives us access to pre-built widgets and styles for making our app look nice and consistent.

OpenAI Service: This imports a file called 'openai_service.dart', which contains code for interacting with the OpenAI service, allowing our app to communicate with it
.
Speech Recognition Result: This line imports a specific part of the 'speech_to_text' package that deals with interpreting the results of speech recognition.

Flutter TTS: This imports the 'flutter_tts' package, which helps our app convert text into spoken words.

Speech to Text: This imports the 'speech_to_text' package, which helps our app understand what someone says out loud and turn it into written words.

Material and HomePage: These lines import the 'material.dart' and 'home_page.dart' files from our own project, containing code for the app's user interface and main screen.

Pallete: This imports a file called 'pallete.dart', which contains colour schemes or other visual settings used throughout the app.

JSON: This line imports code for working with JSON data, a common format for exchanging information between different parts of an app or between apps.

Annexe

GitHub: Project Files

The GitHub project file contains all the files relevant to the project .