NAV.AI

AI-Powered Voice Assistant for Grab Delivery Drivers

Project Documentation

UMHACKATHON 2025 : Grab Problem Statement
Theme: Economic empowerment through AI (from Grab's vision + AI)

KerisByte	
Team Member	**Role**
Wong Jia Soon	Project Lead & Backend Lead & System Integration
Shaikh Amir Husaini Bin Sh.Mohd Saifuddeen	UI/UX Designer
Ahmad Daniel Tamingsari Bin Ramlan	AI Model Integration
Ahmad Fadzril Bin Ahmad Badril	Frontend Lead
Chang Kai Yuan	Backend Voice Specialist

ABSTRACT

This project introduces an advanced AI-powered voice assistant that is specifically designed to improve operational efficiency, communication clarity, and driving safety for e-hailing and food delivery drivers. In fast-paced circumstances where multitasking is common and safety is vital, our approach offers hands-free, natural language interactions, allowing drivers to focus fully on the road while accessing critical information.

Drivers can orally request updates on delivery destinations, real-time instructions, the nearest gas station, or traffic conditions—all without physically interacting with their smartphone. This considerably reduces cognitive burden and distractions, enabling safer and more efficient operations.

To ensure reliable communication in noisy and unpredictable driving environments, the system incorporates ***DeepFilterNet***, a low-complexity **speech enhancement framework**. This component effectively suppresses background noise such as engine sounds, honking, and wind, ensuring that the driver's voice input remains clear and intelligible to the system under various conditions.

Once the audio is cleaned, it is processed through ***Whisper*** for accurate **Speech-to-Text (STT)** transcription. The transcribed text is then interpreted by ***LLaMa 3.2***, an instruction-tuned ***Large Language Model (LLM)*** optimized for deployment on resource-constrained edge devices. LLaMa 3.2 enables the assistant to understand multilingual queries, hold brief conversations, and respond intelligently to diverse user intents—ensuring global usability across different languages and accents.

After generating the response, the system uses a **Text-to-Speech (TTS)** engine to provide vocal feedback to the driver, resulting in a smooth and human-like interaction loop. The assistant can also communicate directly with compatible applications or vehicle systems to initiate specific activities, such as launching navigation or notifying customers of their arrival.

By combining noise suppression, accurate voice recognition, multilingual comprehension, and context-aware response generation, this solution represents a significant step toward safer, smarter driving support tools for gig economy drivers.

It also showcases how cutting-edge AI technologies can be deployed in edge environments to empower users with minimal distraction and maximum efficiency.

In summary, our AI voice assistant is more than a convenience; it is a safety-enhancing, productivity-boosting tool for the ever-expanding community of e-hailing and delivery drivers navigating today's complex urban landscapes.

HACKATHON TIMELINE

TABLE OF CONTENTS

INTRODUCTION

Grab's driver-partners (DAX) currently interact with an AI assistant through text-based interfaces, which can be impractical and unsafe when they are actively driving. To address these limitations, Grab aims to develop a voice-centric interface, allowing driver-partners to receive assistance without the need to physically hold and type on their phones. This transition is crucial for enhancing safety and convenience, empowering drivers to focus on the road while still benefiting from AI-powered guidance and support.

Driver-partners can interact with an AI assistant through text-based interfaces. However, this mode becomes impractical and unsafe when they are actively driving. We want to develop a voice-centric interface to actively help driver-partners without the need to physically hold and type on their phone.

PROBLEM STATEMENT

Grab aims to empower Southeast Asians to better their lives through economic enablement. We have a history of providing them technology which helps them earn a living and build their business. With the advent of GenAI, we see possibilities to accelerate this and improve the lives of people across Southeast Asia.

Our problem statement revolves around building AI-powered assistants for our DAX (drivers) and MEX (merchants) that can either give them insights and guidance or automate work.

The objective is to build a robust voice interaction system that enables reliable driver–assistant communication in challenging audio environments. You need to use some of your creative spirit to design a scenario in which audio assistance is specifically useful. The solution should:

1. **Maintain High Accuracy In Noisy Conditions**
    - Implement noise cancellation and filtering techniques to enhance voice recognition accuracy despite background interference.

2. **Adapt To Diverse Speech Patterns**
    - Utilize NLP models capable of understanding and processing regional accents, dialects, and colloquial expressions.

3. **Provide Clear, Reliable Functionality With Partial Audio Clarity**
    - Design systems that can interpret incomplete or unclear voice inputs and still deliver accurate responses.

4. **Demonstrate Resilience Across Various Environmental Challenges**
    - Ensure the solution is adaptable to different environmental conditions and can maintain reliable communication.

PERSONALIZE STRATEGIES

Core Areas for Personalization:

1. Voice Profile:

- **Accent and speech pattern Adaption:**
  NAV.Ai could learn driver accent and speech patterns over time to improve voice recognition accuracy, especially in noisy environments.

2. Route Preferences:

- **Preferred Route**:
  Allow drivers to indicate preferred routes that they are most familiar with, and NAV.AI can prioritize these suggestions.

- **Avoidance Preferences**:
  Let drivers specify the road or area they want to avoid for delivery.

3. Support Preferences:

- **Frequently Asked Questions:**
  NAV.AI could learn which support topics a driver frequently uses to provide a quicker link or voice prompt for it.

KEY CHALLENGES

The development of a voice-centric interface must overcome several real-world audio challenges:

1. Audio Conditions On The Road

Real-world driving environments present a wide range of audio challenges that can significantly impact the performance of voice-based systems. For drivers in the e-hailing and food delivery sectors, maintaining clear and uninterrupted communication with AI assistants requires overcoming several persistent and dynamic sources of noise. Below are the key audio conditions commonly encountered on the road :

(a) Traffic And Road Noise :
Urban traffic contributes a constant layer of background noise—from honking horns, braking sounds, and tire friction, to the general hum of surrounding vehicles. In congested areas or during peak hours, this acoustic clutter can mask human speech and reduce voice recognition accuracy.

(b) Vehicle Engine Sounds :
Each vehicle generates internal mechanical noise, particularly from the engine and exhaust systems. These sounds vary based on the type of vehicle, engine speed, and driving mode (idling vs. accelerating). For motorbikes and older vehicles, engine noise can be especially intrusive and may overpower a driver's voice.

(c) Weather Conditions Such As Rain And Wind :
Adverse weather greatly affects acoustic conditions. Heavy rain striking the windshield, gusting wind entering through open windows, or the roar of thunder can all disrupt speech clarity. Even the air conditioning or ventilation system can create steady noise that competes with spoken input.

(d) Urban Ambient Noise :
In densely populated city areas, ambient sounds are ever-present—construction work, pedestrian chatter, emergency sirens, street performances, or loudspeakers from roadside shops. These unpredictable and overlapping sounds create a complex acoustic environment that voice systems must navigate to isolate the user's speech.

(e) Audio System Feedback :
Drivers often use in-car entertainment or navigation systems that output audio simultaneously. Background music, navigation instructions, or incoming phone call alerts can interfere with voice commands, especially if the microphone picks up both the driver's voice and the system's audio output. This "feedback loop" can reduce transcription precision and confuse the AI's intent recognition.

2. Speech Pattern Complexities

In designing an AI-powered voice assistant for real-world users, especially those in the e-hailing and food delivery industry, it is crucial to account for the complex diversity of human speech. Unlike controlled lab environments, speech in practical use varies dramatically in tone, pace, vocabulary, and pronunciation. The following are key complexities the system must handle to ensure accurate understanding and response:

(a) Regional accents and dialects :
Drivers may come from diverse cultural and linguistic backgrounds, each bringing unique pronunciation styles, intonation patterns, and phonetic structures. For example:

- A Malaysian English speaker might pronounce vowels and consonants differently than an American or British speaker.
- Indian, Filipino, or African English accents each come with unique stress patterns and rhythmic structures.
- Dialects within the same language (e.g., Mandarin vs. Cantonese, or American Southern vs. Midwestern English) can differ so significantly that they affect word recognition.

These regional differences pose a challenge for traditional speech recognition systems, which often assume a “standard” accent. Our solution uses models like Whisper, trained on a globally diverse dataset, to enhance robustness against these variations.

(b) Variations in speech speed :

Speech tempo can vary greatly among users depending on their comfort with the system, stress level, urgency of the situation, or even their native language. Common issues include:

- Fast speakers may slur words together or skip over syllables.
- Slow speakers may include more pauses and filler words ("uh", "um") that can confuse STT engines.
- Uneven rhythm (e.g., speeding up mid-sentence) may throw off transcription timing.

An effective voice assistant must handle both rapid and deliberate speech patterns without misinterpretation or latency in processing.

(c) Use of colloquial expressions :

Drivers often use informal language, slang, or idioms in their everyday speech. For example:

- Instead of saying “navigate to the nearest petrol station,” a user might say “where can I pump fuel?” or “I need gas, bro.”
- Regional slang like “lah” (in Malaysia/Singapore), “innit” (UK), or “y’all” (Southern US) might be included naturally in requests.
- Expressions may combine multiple languages (code-switching), especially in multilingual environments like Southeast Asia.

PROJECT OBJECTIVES

1. **Enhance Speech Recognition Accuracy in Noisy Environments**

   Implement DeepFilterNet, a low-complexity, real-time speech enhancement framework that utilizes multi-frame algorithms to exploit short-time correlations in speech signals by directly estimating complex filters in the frequency domain. This approach effectively suppresses background noise, ensuring clear voice input even in challenging acoustic conditions.

2. **Adapt to Diverse Speech Patterns and Accents**

   Utilize quantized LLaMA 3.2 models, which offer reduced memory footprint and power consumption while maintaining comparable accuracy to their non-quantized counterparts. These models are optimized for multilingual dialogue and can efficiently run on edge devices, facilitating real-time, on-device natural language understanding.

3. **Ensure Reliable Functionality with Partial Audio Clarity**

   Incorporate keyword detection strategies to accurately interpret driver commands, even when audio quality is compromised. This approach enhances the system's robustness, allowing it to function effectively despite minor audio imperfections.

4. **Demonstrate Resilience Across Various Environmental Challenges**

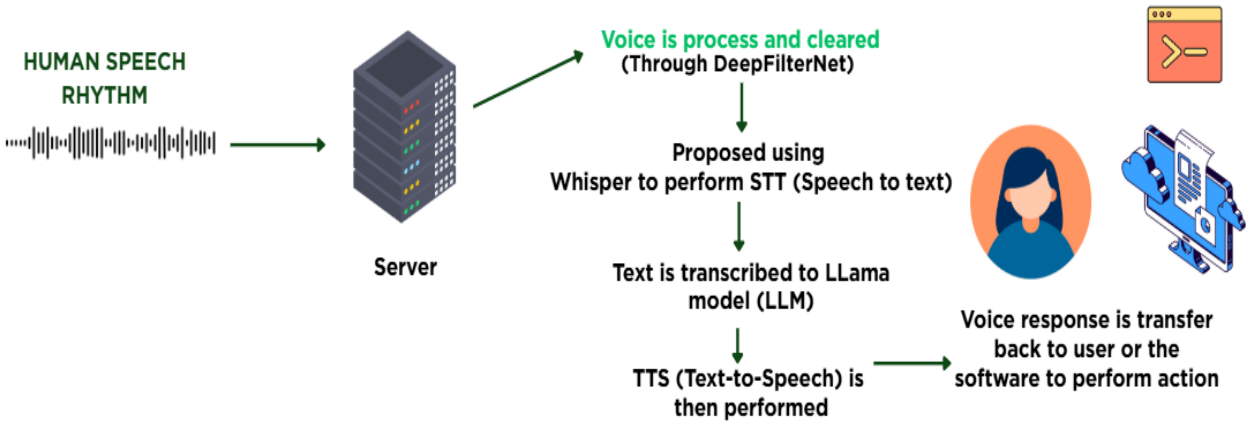
   Design the AI assistant to maintain high performance and accuracy in diverse and unpredictable acoustic environments, such as urban traffic, construction zones, and adverse weather conditions. This resilience ensures consistent functionality regardless of external noise factors.

5. **Promote Safe and Efficient Driver Interaction**

   Enable hands-free, voice-activated interactions, allowing drivers to request information without manual device handling. This design prioritizes driver safety by minimizing distractions and promoting focus on the road.

SOLUTION ARCHITECTURE

SUMMARY OF PROCESS



The system begins by capturing human speech rhythm, which is sent to a server for processing. The voice input is first filtered and enhanced using ***DeepFilterNet*** to **remove background noise** and **improve clarity**. Next, ***Whisper*** is proposed for **Speech-to-Text (STT)** conversion, transforming the cleaned audio into text. This text is then passed into a **LLaMA-based Large Language Model (LLM)** to interpret and generate an intelligent response. Finally, a **Text-to-Speech (TTS)** engine converts the LLM's output back into audio, which is either delivered to the user or used by connected software to perform actions accordingly. This pipeline ensures accurate, efficient, and natural voice interaction within the system.

DATA UTILIZATION

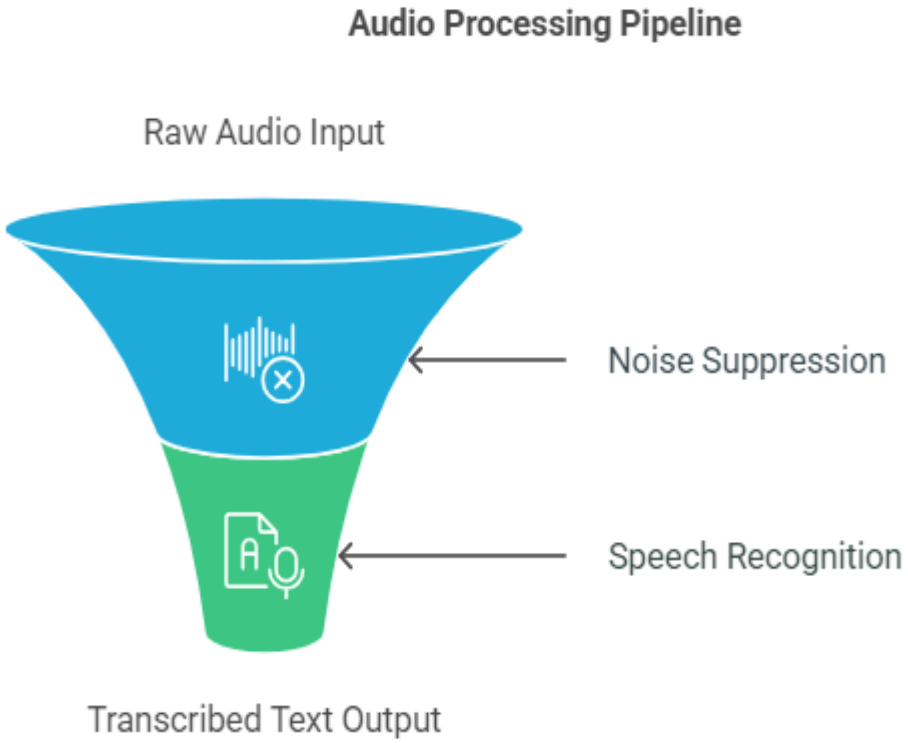
Real-Time Audio Processing

- **Noise Suppression with DeepFilterNet:**

  Incoming audio is processed using DeepFilterNet, a real-time, low-complexity speech enhancement framework. It employs multi-frame algorithms to exploit short-time correlations in speech signals by directly estimating complex filters in the frequency domain, effectively suppressing background noise and ensuring clear voice input even in challenging acoustic conditions.

- **Speech Recognition with Whisper:**

  The denoised audio is then transcribed using Whisper, an offline speech recognition toolkit that supports multiple languages and is optimized for real-time applications on mobile and embedded devices.



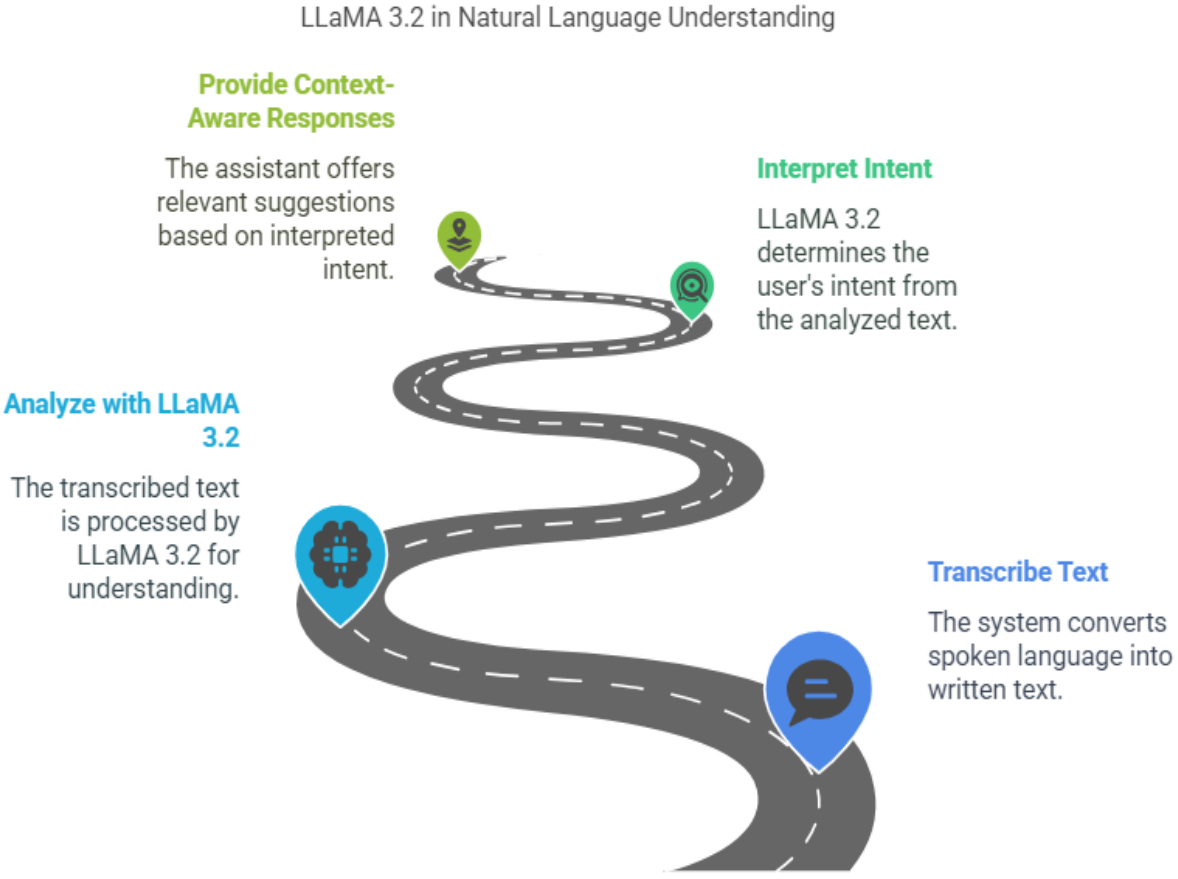
Natural Language Understanding

- **Processing with LLaMA 3.2:**

  The transcribed text is analyzed using LLaMA 3.2, Meta's multilingual large language model. The quantized versions of LLaMA 3.2 offer reduced memory footprint and power consumption while maintaining comparable accuracy to their non-quantized counterparts, making them suitable for deployment on edge devices.

- **Contextual Interpretation:**

  LLaMA 3.2 interprets the driver's intent, enabling the assistant to provide relevant and context-aware responses, such as suggesting the nearest gas station or optimal delivery routes.



Keyword Detection for Enhanced Reliability

- In scenarios where audio quality is compromised, the system employs keyword detection strategies to accurately interpret driver commands. This approach enhances the system's robustness, allowing it to function effectively despite minor audio imperfections.

UX AND UI

1. Voice-First Interaction Design

- Trigger phrase (“Hey Nava!”) to activate the assistant.
- Simple, conversational commands:
  - "How’s traffic ahead?"
  - "Where is the nearest petrol station?"
- Multilingual support (English, Malay, and Mandarin) for Southeast Asia’s diverse user base.

2. Minimal Visual UI / Glanceable Design

- Big fonts and contrast for quick glances.
- Only critical info shown briefly on screen (confirmation: “Navigating to KL Sentral).
- Use visual feedback animations (pulsing rings, waveform mic icon) to indicate:
  - Listening
  - Processing
  - Responding

3. Smart Feedback System

- Voice responses should be short, informative, and non-distracting.
  - “Where do you want to go?”
  - “Traffic is heavy ahead. Rerouting.”
- Allow quick interrupt or “Cancel” voice commands.

4. Fail-Safe Design

- Always provide audible confirmation for actions taken.
- Fallback to minimal visual interface when voice fails.
- Option to repeat or rephrase if misunderstood.

FRONTEND & BACKEND

NAV.AI

A Flutter application that integrates Google Maps, location tracking, speech-to-text input, and routing features.

Features

Library Requirement

- sdk: flutter
- cupertino_icons: ^1.0.8
- avatar_glow: ^3.0.1
- google_maps_flutter: ^2.12.1
- location: ^8.0.0
- speech_to_text: ^7.0.0
- geocoding: ^3.0.0
- http: ^1.3.0
- polyline_codec: ^0.1.6
- provider: ^6.1.4
- permission_handler: ^12.0.0+1

LLM Chat Assistant

A voice-enabled, multilingual backend assistant powered by ***LLaMA*** and ***OpenAI***, designed for **real-time conversation handling**, **navigation commands**, and seamless integration with driving or productivity apps.

Features

Library Requirement

- aiohappyeyeballs==2.6.1
- aiohttp==3.11.16
- aiosignal==1.3.2
- annotated-types==0.7.0
- anyio==4.9.0
- appdirs==1.4.4
- asttokens==3.0.0
- attrs==25.3.0
- certifi==2025.1.31
- cffi==1.17.1
- charset-normalizer==3.4.1
- click==8.1.8
- colorama==0.4.6
- coloredlogs==15.0.1
- contourpy==1.3.2
- cuda-bindings==12.8.0
- cuda-python==12.8.0
- cycler==0.12.1
- decorator==5.2.1
- DeepFilterLib==0.5.6
- DeepFilterNet==0.5.6
- diskcache==5.6.3
- distro==1.9.0
- edge-tts==7.0.0
- executing==2.2.0
- ffmpeg-python==0.2.0
- filelock==3.13.1
- flatbuffers==25.2.10
- fonttools==4.57.0
- frozenlist==1.6.0
- fsspec==2024.6.1
- future==1.0.0
- geocoder==1.38.1
- h11==0.14.0
- httpcore==1.0.8
- httpx==0.28.1
- huggingface-hub==0.30.0
- humanfriendly==10.0

- icecream==2.1.4
- idna==3.10
- ipython==9.1.0
- ipython_pygments_lexers==1.1.1
- jedi==0.19.2
- Jinja2==3.1.4
- jiter==0.9.0
- joblib==1.4.2
- kiwisolver==1.4.8
- llama_cpp_python==0.3.8
- llvmlite==0.44.0
- loguru==0.7.3
- MarkupSafe==2.1.5
- matplotlib==3.10.1
- matplotlib-inline==0.1.7
- more-itertools==10.6.0
- mpmath==1.3.0
- multidict==6.4.3
- networkx==3.3
- noisereduce==3.0.3
- numba==0.61.2
- numpy==1.26.4
- nvmath-python==0.3.0
- onnxruntime==1.21.0
- openai==1.74.0
- openai-whisper==20240930
- packaging==23.2
- parso==0.8.4
- pillow==11.0.0
- playsound==1.2.2
- prompt_toolkit==3.0.51
- propcache==0.3.1
- protobuf==6.30.2
- pure_eval==0.2.3
- PyAudio==0.2.14
- pycparser==2.22
- pydantic==2.11.3
- pydantic_core==2.33.1
- Pygments==2.19.1
- pyparsing==3.2.3

- pyreadline3==3.5.4
- python-dateutil==2.9.0.post0
- pywin32==310
- PyYAML==6.0.2
- ratelim==0.1.6
- regex==2024.11.6
- requests==2.32.3
- safetensors==0.5.3
- scipy==1.15.2
- setuptools==70.2.0
- silero-vad==5.1.2
- six==1.17.0
- sniffio==1.3.1
- sounddevice==0.5.1
- soundfile==0.13.1
- srt==3.5.3
- stack-data==0.6.3
- sympy==1.13.1
- tabulate==0.9.0
- tiktoken==0.9.0
- tokenizers==0.21.1
- torch==2.6.0+cu126
- torchaudio==2.6.0+cu126
- torchvision==0.21.0+cu126
- tqdm==4.67.1
- traitlets==5.14.3
- transformers==4.51.3
- typing-inspection==0.4.0
- typing_extensions==4.12.2
- urllib3==2.4.0
- vosk==0.3.45
- wavefile==1.6.3
- wcwidth==0.2.13
- websockets==15.0.1
- win32_setctime==1.2.0
- yarl==1.20.0

TECHNOLOGY USED

DeepFilterNet (Real-Time Noise Suppression)

DeepFilterNet is used to enhance voice clarity by filtering out environmental noise such as traffic, engine hums, and urban sounds. This ensures that voice commands are captured cleanly, even in noisy road conditions, enabling reliable speech recognition.

Whisper (Speech-to-Text Processing)

OpenAI's Whisper is integrated for automatic speech recognition (ASR), converting spoken commands into text with high accuracy across multiple languages. It plays a critical role in enabling hands-free communication between the driver and the assistant.

LLaMa 3.2 (Large Language Model for Understanding and Response)

LLaMa 3.2 is used as the core natural language processing engine. It interprets user queries, understands context, and generates intelligent responses, supporting fluid human-computer interaction in real-time.

Bolt.new (Prototyping Framework)

Bolt.New is leveraged for rapid prototyping during the hackathon, enabling the team to quickly iterate on features, UI flows, and backend logic. It accelerates testing and validation of ideas in a short timeframe.

React Native (Mobile Interface Development)

React Native is used to build the cross-platform mobile front-end interface of NAV.AI. It ensures a smooth and responsive UI/UX for Android users while reducing development time with reusable components.

Google Cloud Console (Backend Hosting & Processing)

The backend infrastructure, including the AI models and voice processing services, is hosted on Google Cloud. This ensures high availability, scalability, and performance during real-time interactions.

SUMMARY

NAV.AI is an innovative hackathon project developed to address the critical issue of driver distraction among Grab food delivery riders in Malaysia. As food delivery services continue to grow, drivers often face the challenge of multitasking while navigating, managing orders, and communicating with customers—all while contending with traffic and road safety. This project proposes a hands-free, voice-centric AI assistant designed to enhance operational efficiency and promote safer driving conditions.

The core idea of **NAV.AI** revolves around eliminating the need for manual smartphone interaction during active delivery. By introducing a seamless voice interface, drivers can access essential features such as order information, navigation, rider support, and customer updates simply by speaking. The system is activated using a designated wake phrase ("Hey Nava") or via a tactile button on the interface. For added security and personalization, **NAV.AI** incorporates voice recognition to register and verify the owner's voice, ensuring only authorized users can access it during a session.

Upon activation, a minimalist visual keyword menu is presented, enabling the driver to navigate options by saying specific terms like "Navigate", "Route", "Order", "Customer", or "Support". This interaction model reduces cognitive load, allowing drivers to keep their hands on the wheel and eyes on the road. To further enhance safety, the system is designed to detect vehicle speed and display warning messages if use is attempted at high speeds, discouraging interaction during critical driving moments.

To ensure reliable performance in challenging real-world environments, **NAV.AI** integrates several cutting-edge technologies. ***DeepFilterNet*** is employed to suppress background noise from traffic, rain, engine sounds, and urban environments, thereby enhancing the clarity of spoken commands. Speech recognition and ***Speech To Text (STT)*** is performed using ***Whisper***, which efficiently converts spoken input into text even in noisy conditions. This text is then processed by ***LLaMa 3.2***, a powerful instruction-tuned **Large Language Model (LLM)** capable of understanding natural language commands, generating responses, and performing reasoning. Finally, responses are conveyed to the user through ***Text-to-Speech (TTS)***, creating a natural and continuous voice interaction loop.

NAV.AI is more than just a technical demonstration—it represents a thoughtful reimagining of how technology can support gig economy workers, particularly in Southeast Asia where delivery services play a vital role. By reducing reliance on screens and manual controls, the project aims to decrease the risk of accidents, improve communication, and provide a smoother workflow for drivers.

Through this prototype, **NAV.AI** showcases the potential of combining AI, speech enhancement, and real-time language understanding to create a user-focused tool tailored to the needs of modern delivery professionals. It reflects the hackathon spirit of practical innovation—building a solution that is technically robust, user-friendly, and aligned with real-world challenges.

REFERENCES

Alphacep. (n.d.). *GitHub - alphacep/vosk-api: Offline speech recognition API for Android, iOS, Raspberry Pi and servers with Python, Java, C# and Node*. GitHub. https://github.com/alphacep/vosk-api?tab=readme-ov-file

Grab. (2025, April 9). *Question and answers*. https://help.grab.com/driver/en-my

Grab. (2025, April 9). *Understanding and managing fatigue*. https://help.grab.com/passenger/en-my/115006541328-I-was-involved-in-an-accident

HowToDevice. (2024, September 25). *iPhone 16: How to turn on Emergency SOS call* [Video]. YouTube. https://youtu.be/8ENsBqadrkc?si=ad-IyfOjqfQQ5EJg

Jzhang. (n.d.). *GitHub - jzhang38/TinyLlama: The TinyLlama project is an open endeavor to pretrain a 1.1B Llama model on 3 trillion tokens*. GitHub. https://github.com/jzhang38/TinyLlama

OpenAI. (n.d.-a). *GitHub - openai/whisper: Robust speech recognition via Whisper*. GitHub. https://github.com/openai/whisper

OpenAI. (n.d.-b). *Whisper README*. GitHub. https://github.com/openai/whisper/blob/main/README.md

Popa, B. (2024, September 11). *I used Google Maps with all the new features, and here's why Waze now feels useless*. autoevolution. https://www.autoevolution.com/news/i-used-google-maps-with-all-the-new-features-and-here-s-why-waze-now-feels-useless-239633.html

Schröter, H., Rosenkranz, T., Escalante-B., A. N., & Maier, A. (2023, May 14). *DeepFilterNet: Perceptually motivated real-time speech enhancement*. arXiv. https://arxiv.org/abs/2305.08227

TechRechard. (2024, May 28). *Emergency SOS on Android* [Video]. YouTube. https://youtube.com/shorts/lewWVdc-NgI?si=gqK33FVnPVW44RWd

TheServerSide. (n.d.). *Locally run Huggingface LLMs like LLaMa on your laptop or desktop with Python* [Video]. https://www.theserverside.com/video/Run-Llama-LLMs-on-your-laptop-with-Hugging-Face-and-Python

Wazeopedia, U. (2017, May 19). *Waze discuss*. https://www.waze.com/discuss/t/main-page-about-waze/378175

Whisper AI. (2022, September 22). *Whisper: OpenAI's speech recognition model* [Video]. YouTube. https://www.youtube.com/watch?v=JR36oH35Fgg