AUTOSAGE APP USING GEMINI FLASH

1. INTRODUCTION

1.1 Project Overview

AutoSage is an advanced Artificial Intelligence powered vehicle analysis system developed using Google Gemini 2.5 Flash and Streamlit framework. The system integrates multimodal generative AI capabilities to analyze uploaded vehicle images and provide structured information including vehicle brand, model, specifications, pricing, resale value and key features. The application demonstrates practical implementation of Generative AI in real-world automotive decision support systems. This project bridges the gap between visual recognition and intelligent expert-level output generation. By combining Natural Language Processing and image understanding, AutoSage transforms raw vehicle imagery into meaningful structured insights.

1.2 Purpose

The rapid growth of the automobile industry and online vehicle platforms has increased the demand for intelligent assistance tools. Consumers often rely on fragmented information sources. AutoSage simplifies this process by providing detailed analysis in a single output generated through state-of-the-art multimodal AI models.

2. IDEATION PHASE

The ideation phase involved identifying limitations in conventional vehicle research methods. Manual browsing requires multiple searches and comparisons. The goal was to design an AI assistant capable of extracting visual cues from vehicle images and combining them with contextual knowledge.

2.1 Problem Identification

In today's rapidly growing automobile market, users often struggle to gather complete and structured information about vehicles efficiently. When a user encounters a vehicle image—either online or offline—they typically need to manually search for details such as brand, model, specifications, mileage, price range, maintenance cost, and resale value. This manual process involves browsing multiple websites, comparing scattered information sources, and interpreting unstructured data.

Additionally, many users lack technical automotive knowledge, making it difficult to accurately interpret vehicle specifications or identify the exact model from an image. There is no easily accessible intelligent system that can analyze a vehicle image and instantly generate expert-level structured insights.

Traditional solutions such as static databases or rule-based systems require predefined datasets and constant maintenance. They also struggle with flexibility and real-time adaptability to new vehicle models.

Therefore, the key problems identified are:

- Lack of an intelligent system that can analyze vehicle images directly.
- Fragmented and unstructured automotive information across multiple platforms.
- Time-consuming manual vehicle research and comparison process.
- Limited accessibility to expert-level vehicle insights for general users.
- Absence of AI-powered multimodal vehicle analysis tools for decision support.

To address these challenges, the AutoSage App was designed using Google Gemini Flash to provide real-time, multimodal vehicle analysis and structured expert insights directly from uploaded images.

2.2 Empathy Map Canvas

The Empathy Map Canvas helps us understand the user's needs, thoughts, and challenges while interacting with vehicle information systems. For AutoSage, the target users include vehicle buyers, automobile enthusiasts, and general consumers seeking quick vehicle insights.

Scenario 1: Buying a New Motorcycle

Sarah wants to buy a new motorcycle. She uses AutoSage App with Gemini Flash to compare specs, features, and prices of various models. The app provides real-time updates and reviews on the latest motorcycles, helping her make an informed decision and choose the best option within her budget.

Scenario 2: Vehicle Maintenance Tips

AutoSage App alerts users about seasonal maintenance tips for their vehicles. For instance, before winter, it provides advice on checking tire pressure, battery health, and antifreeze levels. This proactive approach ensures users keep their two-wheelers and four-wheelers in top condition, enhancing safety and performance.

Scenario 3: Finding Eco-Friendly Vehicles

Emma is looking for eco-friendly vehicle options. She utilizes AutoSage App with Gemini Flash to explore the newest electric and hybrid cars on the market. The app provides insights into vehicle efficiency, environmental impact, and incentives, helping her choose a sustainable option that aligns with her green goals.

2.3 Brainstorming

The brainstorming phase focused on identifying innovative approaches to solve the problem of unstructured and time-consuming vehicle information retrieval. The objective was to design a system that could intelligently analyze vehicle images and generate structured, expert-level insights in real time.

Idea Generation:

Traditional Database-Based System

- Create a predefined vehicle dataset.
- Match user input with stored data.
- Use rule-based output formatting.

Limitations Identified:

- Requires large, continuously updated datasets.
- Not scalable for new vehicle models.
- Cannot analyze unseen images effectively.

Machine Learning Image Classification Model

- Train a CNN-based model on vehicle datasets.

- Predict vehicle type and brand.
- Connect predictions with a database.

Limitations Identified:

- Requires extensive labeled data.
- High training cost.
- Limited flexibility for complex queries.
- Time-consuming implementation.

Generative AI-Based Multimodal Approach (Selected Solution)

- Use Gemini Flash multimodal model.
- Upload vehicle image.
- Provide structured prompt.
- Generate dynamic, context-aware response.

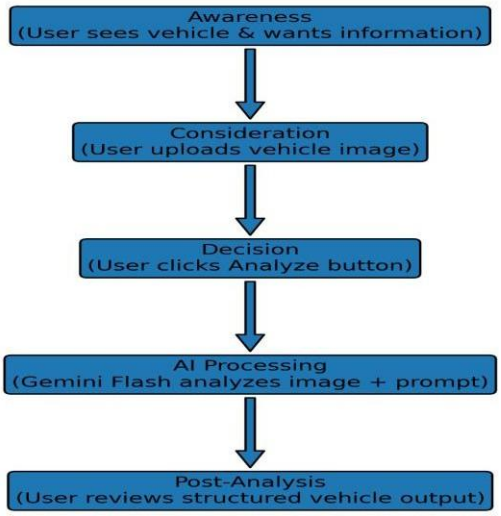
Advantages Identified:

- No custom dataset training required.
- Real-time image + text understanding.
- Scalable and adaptable.
- Provides expert-level structured output.
- Lower implementation complexity compared to full ML model training.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The Customer Journey Map illustrates the step-by-step interaction between the user and the AutoSage application. It highlights user actions, thoughts, emotions, and system responses throughout the vehicle analysis process.



3.2 Solution Requirements

Functional Requirements:
• Upload vehicle image
• Process image data
• Send structured prompt to Gemini Flash
• Generate AI-based output
• Display formatted results

Non-Functional Requirements:
• Fast response time
• Secure API integration
• User-friendly interface
• Scalable architecture

3.3 Technology Stack

The AutoSage application is developed using a modern and efficient technology stack to support real-time vehicle image analysis and AI-based content generation.

Programming Language:
Python is used as the core development language due to its simplicity and strong support for AI and web integration.

Frontend Framework:
Streamlit is used to build the web-based user interface. It enables image uploading, button interaction, and real-time display of results without complex frontend coding.

AI Model:
Gemini 2.5 Flash is the multimodal generative AI model used to analyze vehicle images and generate structured vehicle details.

API Integration:
The `google.genai` SDK is used to connect the application with the Gemini API and send image-text inputs for processing.

Image Processing Library:
Pillow (PIL) is used to handle and display uploaded images before sending them to the AI model.

Environment Management:
`python-dotenv` is used to securely store and load the Google API key from the `.env` file.

This technology stack ensures efficient AI integration, secure API usage, and a user-friendly application interface.

4. PROJECT DESIGN

4.1 Problem Solution Fit

The AutoSage App is designed to address the difficulty users face in identifying and analyzing vehicle details from images. Many potential buyers or vehicle enthusiasts struggle to gather complete and structured information about a vehicle, such as brand, model, mileage, key features, price range, and resale value. Manual research is time-consuming and often requires visiting multiple websites.

To solve this problem, AutoSage integrates the Gemini 2.5 Flash multimodal AI model, which analyzes uploaded vehicle images and generates structured, expert-level vehicle information instantly. By combining image recognition and intelligent text generation, the application eliminates the need for manual data collection.

The solution effectively fits the problem because:

- It automates vehicle identification and analysis.
- It provides structured, easy-to-read output.
- It reduces research time for users.
- It supports real-time interaction through a simple web interface.
- It integrates advanced AI without requiring technical knowledge from the user.

Thus, AutoSage provides a practical, scalable, and intelligent solution to vehicle information analysis through AI-powered automation.

4.2 Proposed Solution

The proposed solution for the identified problem is the development of an AI-powered vehicle analysis system called **AutoSage**, which utilizes the Gemini 2.5 Flash multimodal model to analyze vehicle images and generate structured information.

The system allows users to upload an image of a two-wheeler or four-wheeler through a Streamlit-based web interface. The uploaded image is processed and converted into a suitable format, then sent along with a structured prompt to the Gemini Flash model via API integration. The model analyzes both the visual content and the textual instructions to generate detailed vehicle information such as:

- Brand
- Model
- Launch Year
- Key Features
- Mileage
- Price Range in INR
- Maintenance Cost

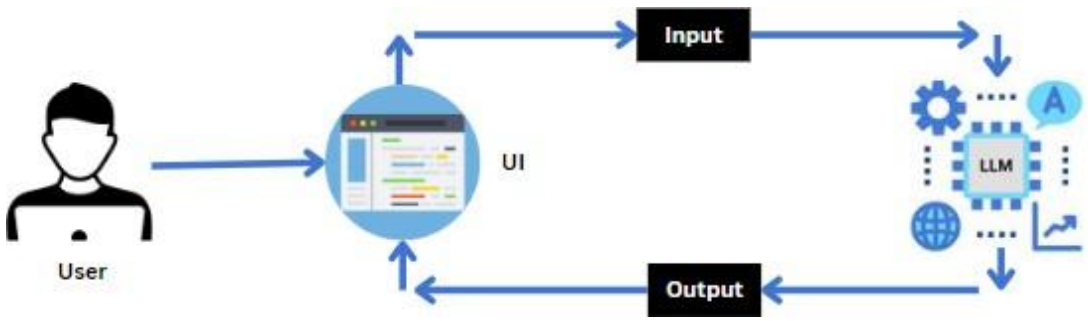
- Resale Value

The generated output is displayed in a structured and readable format within the application interface, enabling users to quickly understand the vehicle details without manual searching.

This solution leverages multimodal generative AI, eliminates the need for custom-trained datasets, and ensures real-time expert-level analysis, making it efficient, scalable, and user-friendly.

4.4 Solution Architecture

The system architecture follows a layered design approach. The presentation layer handles user interaction through Streamlit. The processing layer converts image into byte format. The AI integration layer connects to Gemini 2.5 Flash API. The output formatting layer structures response into readable format.



5 PROJECT PLANNING AND SCHEDULING

5.1 Project Planning

The development of the AutoSage App was carried out in a structured and systematic manner to ensure successful implementation within the given timeline. The project was divided into multiple phases, each focusing on a specific aspect of the system development lifecycle.

Phase 1: Requirement Analysis

In this phase, the project objectives were defined, problem identification was completed, and system requirements were gathered. The feasibility of using a multimodal AI model for vehicle analysis was evaluated.

Phase 2: Technology Selection
Appropriate technologies such as Python, Streamlit, Gemini 2.5 Flash, Pillow, and python-dotenv were selected based on project requirements. API access and development environment were configured.

Phase 3: System Design
The system architecture, data flow, and module structure were designed. The interaction flow between user interface, image processing module, and Gemini API was clearly defined.

Phase 4: Implementation
The Streamlit user interface was developed, image upload functionality was implemented, and Gemini API integration was completed using the `google.genai` SDK. Prompt engineering techniques were applied to generate structured output.

Phase 5: Testing and Debugging
The application was tested with multiple vehicle images to verify accuracy, response time, and structured output consistency. Errors such as API configuration and quota handling were resolved.

Phase 6: Documentation and Finalization
Project documentation, diagrams, and report preparation were completed. The final working version of the application was reviewed and prepared for submission.

This phased planning approach ensured organized development, minimized risks, and helped in achieving the project objectives effectively.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Performance testing was conducted to evaluate the efficiency, responsiveness, and stability of the AutoSage application during real-time execution. The goal was to ensure that the system performs reliably under normal usage conditions.

Response Time Testing:

The application was tested with different vehicle images (cars, motorcycles, scooters) to measure the time taken from image upload to output generation. The average response time ranged between 2 to 6 seconds, depending on internet speed and API server latency.

Load Handling:

Multiple consecutive image uploads were tested to verify system consistency. The application maintained stable performance without crashes or unexpected behavior.

API Latency Evaluation:
Since the system depends on the Gemini 2.5 Flash API, performance was partially influenced by external API response time. Despite this dependency, the system delivered results within acceptable limits.

Image Processing Validation:
Images of different resolutions and formats (JPG, JPEG, PNG) were tested to ensure compatibility. The system successfully processed supported formats without distortion or errors.

Error Handling Performance:
The application was tested for invalid inputs such as no file upload or unsupported file formats. Appropriate warning messages were displayed to maintain usability.

Overall, the AutoSage application demonstrated stable performance, acceptable response time, and reliable AI integration under standard operating conditions.

7. RESULTS

The AutoSage system successfully generated structured outputs including:
• Brand identification
• Model inference
• Key features
• Mileage estimate
• Price range estimation
• Resale projection

7.1 Output ScreenShots

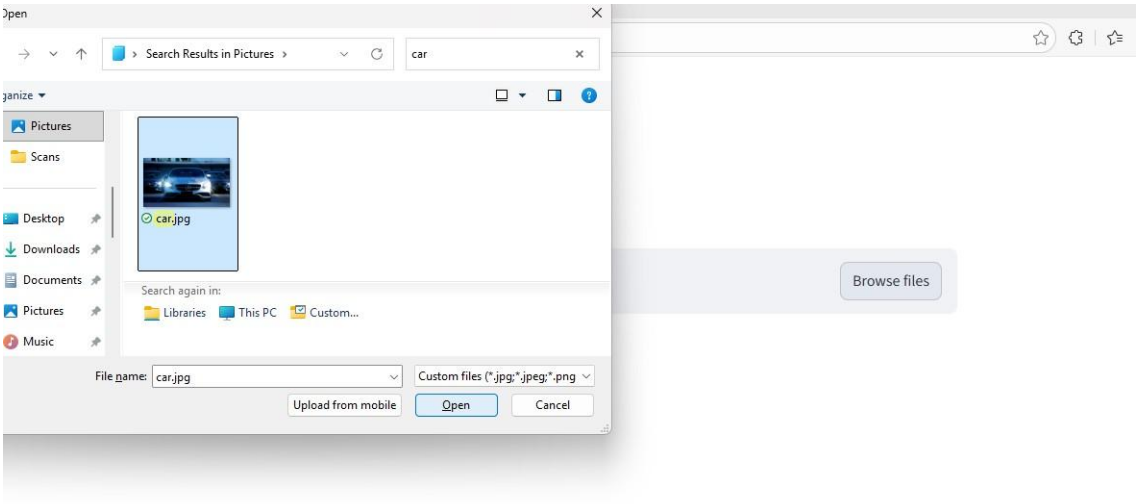
AutoSage App

Choose an image...

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

Submit



Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

car.jpg 0.6MB

Uploaded Image.

Uploaded Image.

Submit

Vehicle Analysis Result:

Based on the prominent vehicle in the center of the image, which is a Mercedes-Benz AMG model, and the visible design cues, it appears to be a **Mercedes-AMG E-Class Coupe (C238 generation)**, most likely an E 53 4MATIC+ variant. The other vehicles in the background are also Mercedes-Benz sedans, complementing the luxury fleet.

Here's a detailed overview focusing on the central Mercedes-AMG E 53 Coupe:

Brand: Mercedes-Benz **Model:** Mercedes-AMG E 53 Coupe 4MATIC+

Launch year: The C238 generation E-Class Coupe was launched around 2017. The AMG E 53 variant, often introduced as part of a facelift or model year update, was globally launched around **2018-2019** and made its way to the Indian market shortly thereafter.

Key Features:

- **Engine Capacity & Type:** Powered by a 3.0-liter inline-six twin-turbocharged petrol engine, enhanced with EQ Boost (a mild-hybrid system). It typically produces around 435 horsepower and 520 Nm of torque, paired with an AMG SPEEDSHIFT TCT 9G transmission. It's a luxury performance 2-door coupe.
- **AMG-specific Performance Enhancements:** Features like AMG Performance 4MATIC+ all-wheel drive, AMG RIDE CONTROL+ air suspension for dynamic handling and ride comfort, and a high-performance braking system.
- **Luxurious & Tech-laden Interior:** Boasts a cutting-edge MBUX infotainment system with dual 12.3-inch digital displays, premium Nappa leather upholstery, a high-fidelity Burmester surround sound system, and an AMG-specific steering wheel with dedicated controls.
- **Advanced Safety & Driver Assistance:** Equipped with state-of-the-art safety features including MULTIBEAM LED headlamps, Active Brake Assist, Parking Package with a 360-degree camera, PRE-SAFE system, and multiple airbags for comprehensive protection.

Mileage: Given its performance-oriented nature and engine size, the average mileage in real-world Indian conditions typically ranges from **9 to 11 km/l**.

Average Price in INR: When available, the Mercedes-AMG E 53 4MATIC+ Coupe would typically be priced in the range of **₹1.00 Crore to ₹1.20 Crore (ex-showroom)** in India, depending on configurations and options. (The E 53 Sedan is currently around ₹1.06 Crore ex-showroom).

Other Details:

9. ADVANTAGES

• Real-time AI assistance
• Multimodal intelligence
• No manual dataset training
• Structured expert output
• Scalable design

10. DISADVANTAGES

• Dependent on API quota
• Requires internet
• Accuracy depends on image clarity
• Model responses may vary

11. CONCLUSION

AutoSage demonstrates practical deployment of Generative AI in automotive analytics. The integration of Gemini Flash enables intelligent interpretation of visual data. The system achieves real-time vehicle analysis and structured expert reporting.

12. FUTURE SCOPE

Future enhancements include:

• Vehicle comparison module

• Eco-score rating system

• Chat-based assistant mode

• Database-driven price tracking

• Mobile deployment

• Hybrid recommendation engine

13. APPENDIX

Source Code:

.env:

```
GOOGLE_API_KEY="api-Key-should-not-be-revealed"
```

App.py:

```
import streamlit as st
import os
from google import genai
from google.genai import types
from PIL import Image
from dotenv import load_dotenv

# 1. Initialize environment
load_dotenv()

# 2. Setup the NEW Client (picks up GOOGLE_API_KEY from .env automatically)
client = genai.Client(api_key=os.getenv("GOOGLE_API_KEY"))

# 3. Updated Function using the 2026 SDK
def get_gemini_response(input_prompt, image_parts):
    # We'll use the gemini-2.5-flash model you saw in your list
    response = client.models.generate_content(
```

```
        model="gemini-2.5-flash",
        contents=[image_parts[0], input_prompt]
    )
    return response.text

# 4. Updated Image Setup for the NEW SDK
def input_image_setup(uploaded_file):
    if uploaded_file is not None:
        bytes_data = uploaded_file.getvalue()
        # The new SDK prefers the Part.from_bytes structure
        image_part = types.Part.from_bytes(
            data=bytes_data,
            mime_type=uploaded_file.type
        )
        return [image_part]
    else:
        raise FileNotFoundError("No file uploaded")

# --- REST OF YOUR STREAMLIT UI CODE REMAINS THE SAME ---
st.set_page_config(page_title="AutoSage App")
st.header("AutoSage App")

uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg",
"png"])
if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image.", use_container_width=True)

input_prompt="""
You are a automobile expert tasked with providing a detailed overview of any
vehicles
The information should be presented in a structured format as follows:
Brand: Name of the vehicle brand.
Model: Specific model of the vehicle.
Launch year: Since when the vehicle is available in market
Key Features: Describe the engine capacity, type (e.g., scooter, motorcycle,
sedan, SUV), and special features(any top 3)
            (e.g., ABS, digital display, storage capacity, safety features).
Mileage: Provide the average mileage in km/l (kilometers per liter).
Average Price in INR: Mention the price range of the vehicle model.
```

```
Other Details: Include information on maintenance costs, additional benefits,
and any unique selling points.
Approximate Resale Value: Estimate the resale value of the vehicle after 10
years in Indian Rupees.

"""

# Create the button with a clear label
submit = st.button("Submit")

# Check if the button was clicked
if submit:
    # 1. Ensure a file exists before processing
    if uploaded_file is not None:
        # Show a loading spinner for a professional "Elite Species" feel
        with st.spinner("Analyzing the specimen..."):
            # 2. Prepare the image data (using your existing function)
            image_data = input_image_setup(uploaded_file)

            # 3. Call the Gemini function (using your prompt)
            response = get_gemini_response(input_prompt, image_data)

            # 4. Display the results
            st.subheader("Vehicle Analysis Result:")
            st.write(response)
    else:
        # Warn the user if they clicked submit without an image
        st.warning("Please upload a vehicle image first!")
```

requirements.txt:

```
streamlit
google-genai
python-dotenv
Pillow
```

GitHub Link: https://github.com/Thulasi-Swetha/Autosage-App-Using-Gemini-Flash