Task 1: Python & AI
Prepared By: Sahan
Date:24/02/2025

# Pneumonia Detection Using Deep Learning

-A Machine Learning Solution for Chest X-Ray Analysis-

## Problem Statement

The objective of this project is to develop a deep learning model for classifying chest X-ray images into two categories: "Normal" and "Pneumonia." The solution involves training a Convolutional Neural Network (CNN) using TensorFlow and leveraging VGG16 as a feature extractor. The model is trained with data augmentation and class balancing techniques to improve generalization

After training, the model is evaluated using accuracy, classification reports, and confusion matrices.Finally, the trained model is saved for future use, and it can be deployed as a Flask API to enable real-time predictions on a local PC.

## Approach

*Phase1: Data Preparation*

The Chest X-Ray Images (Pneumonia) dataset from Kaggle was accessed and loaded in Google Colab to streamline data handling. Images were resized to 224x224 pixels and normalized to a 0-1 scale for compatibility with the VGG16 model. Data augmentation techniques were applied to enhance model generalization.

```
# Enhanced data augmentation for training
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    brightness_range=[0.8, 1.2],
    fill_mode='nearest',
    validation_split=0.2
)
```

*Figure 1 - Data preprocessing*

*Phase2: Model Training*

| Layer (type) | Output Shape | Param # |
|---|---|---|
| vgg16 (Functional) | (None, 7, 7, 512) | 14,714,688 |
| flatten (Flatten) | (None, 25088) | 0 |
| dense (Dense) | (None, 256) | 6,422,784 |
| dropout (Dropout) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 128) | 32,896 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 1) | 129 |

**Total params:** 21,170,497 (80.76 MB)

**Trainable params:** 6,455,809 (24.63 MB)

**Non-trainable params:** 14,714,688 (56.13 MB)

*Figure 3 - Model Summary*

```
Epoch 1/20
147/147 ━━━━━━━━━━━━━━━━━━━━ 1151s 8s/step - accuracy: 0.6219 - loss: 0.6983 - val_accuracy: 0.8961 - val_loss: 0.3283
Epoch 2/20
147/147 ━━━━━━━━━━━━━━━━━━━━ 82s 561ms/step - accuracy: 0.8263 - loss: 0.4247 - val_accuracy: 0.9148 - val_loss: 0.2606
Epoch 3/20
147/147 ━━━━━━━━━━━━━━━━━━━━ 84s 571ms/step - accuracy: 0.8520 - loss: 0.3530 - val_accuracy: 0.8961 - val_loss: 0.2926
Epoch 4/20
147/147 ━━━━━━━━━━━━━━━━━━━━ 142s 574ms/step - accuracy: 0.8795 - loss: 0.3289 - val_accuracy: 0.9131 - val_loss: 0.2686
Epoch 5/20
147/147 ━━━━━━━━━━━━━━━━━━━━ 85s 574ms/step - accuracy: 0.8868 - loss: 0.3203 - val_accuracy: 0.8416 - val_loss: 0.3996
```

*Figure 2 - Model Training*

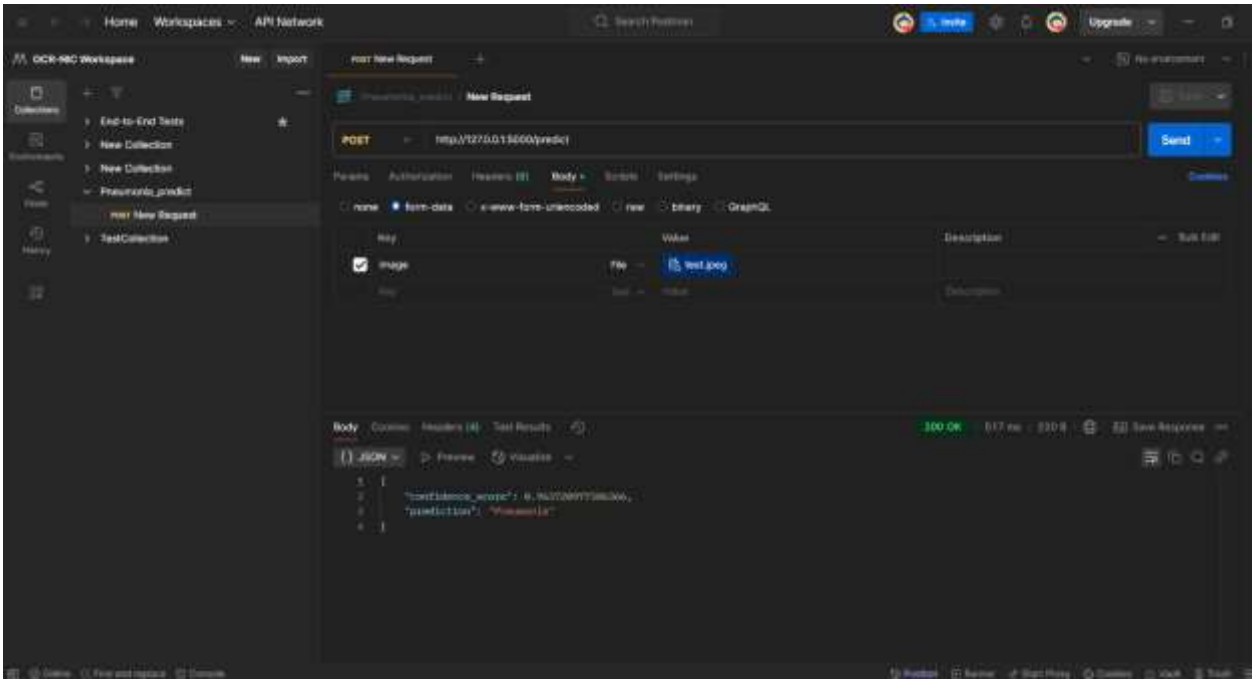



*Figure 4 - Postman Response*

## Results and Outputs

### Results:

```
39/39 ━━━━━━━━━━━━━━━━━━━━ 8s 205ms/step - accuracy: 0.8131 - loss: 0.4178
Test Loss: 0.3746
Test Accuracy: 0.8381
39/39 ━━━━━━━━━━━━━━━━━━━━ 8s 194ms/step
```

*Figure 5*

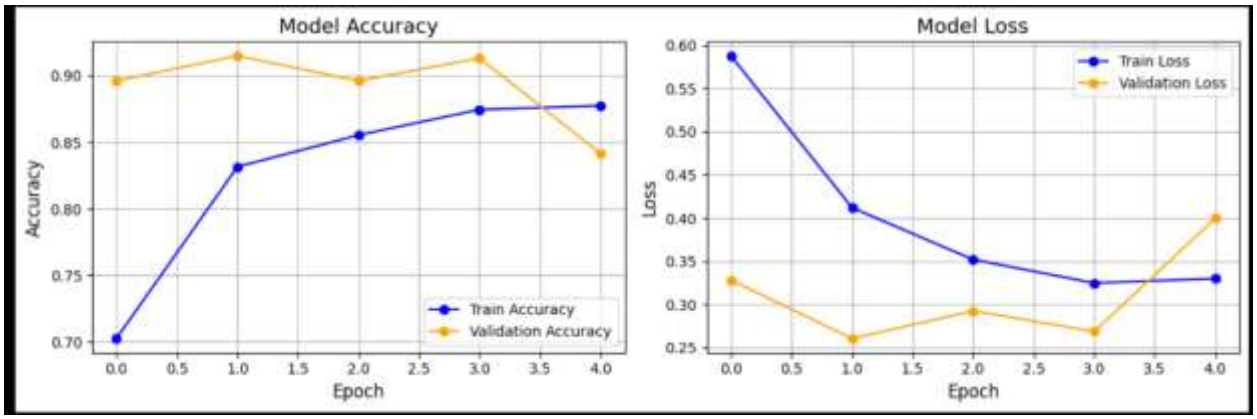



*Figure 6*

**Prediction Example**:

A sample X-ray image was tested using both the API and local script, correctly identifying it as Pneumonia with a confidence score of 0.96


```
{
    "confidence_score": 0.963720977306366,
    "prediction": "Pneumonia"
}
```


*Figure 7 - Test Output*

**Model Evaluation**:

The model performed better at detecting "Pneumonia" (recall: 0.87) than "Normal" (recall: 0.78)

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Normal | 0.79 | 0.78 | 0.78 | 234 |
| Pneumonia | 0.87 | 0.87 | 0.87 | 390 |
| accuracy | | | 0.84 | 624 |
| macro avg | 0.83 | 0.83 | 0.83 | 624 |
| weighted avg | 0.84 | 0.84 | 0.84 | 624 |

*Figure 8 - Classification report*

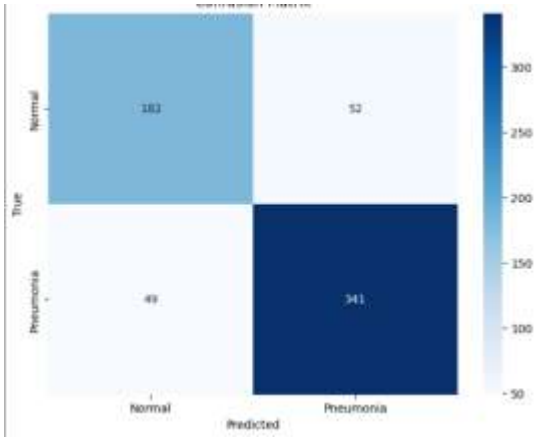



*Figure 9 - Confusion matrix*

## Challenges and Solutions

**Challenge:** The dataset might have more pneumonia cases than normal cases, leading to a biased model.
**Solution**: Applied data augmentation techniques such as flipping, rotation, and zooming to artificially balance the dataset and prevent overfitting.

**Challenge**: Postman API testing initially returned an error: "path should be path-like or io.BytesIO, not FileStorage" when uploading images directly.
**Solution**: Modified the API to accept an image_path in a JSON request instead, allowing the server to load the image directly using the provided path.

**Additional Resources:**

Postman Link: https://ocr-nic.postman.co/workspace/predict_data~22baf82b-3e5d-4e43-880a-acec25247369/request/42844679-0f5c6e7d-e7de-4b84-906e-a501089e6e1b?action=share&creator=42844679&ctx=documentation

Task 1: AI Sales Agent
Prepared By: Sahan
Date:24/02/2025

# AI-Powered Phone Sales Assistant

- An Interactive Solution for Product Queries and Visualization-

## Problem Statement

The objective of this project is to develop an AI-powered sales agent for a tech store that answers user queries about phones and displays product images on request. The solution involves integrating a product catalog, using OpenAI's GPT-4o for natural language responses, and building a Tkinter-based GUI with image display functionality using Pillow. This document summarizes the methodology, implementation, results, and challenges encountered during the process.

## Approach

*Phase 1: Data Preparation*

The project began with organizing a product catalog, structured as a Python dictionary. This catalog detailed three phones—iPhone 13, Samsung Galaxy S23, and Google Pixel 7—listing their prices, features, stock levels, and image file paths (e.g., data/iPhone_13.jpg). To prepare for image integration, I ensured all image files were stored in a local data folder and confirmed their paths worked seamlessly with the Pillow library for later processing.

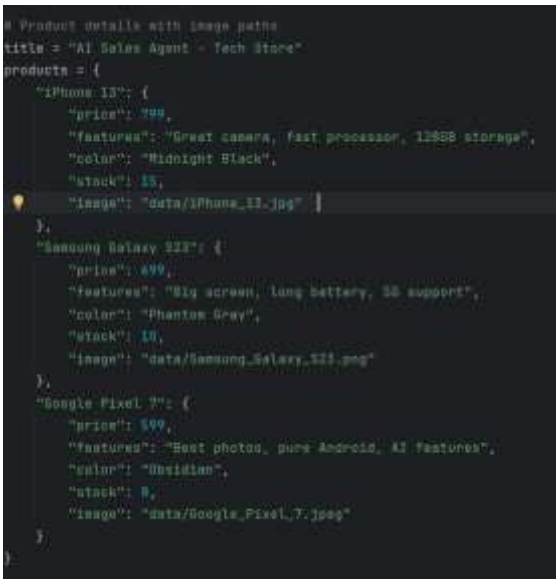



*Figure 10-product catalog*

*Phase 2: AI Integration and GUI Development*

For the core functionality, I leveraged OpenAI's GPT-4o model to produce natural, conversational responses by embedding the product catalog into a custom prompt. Alongside this, I developed a user interface using Tkinter, featuring a chat window for dialogue, an input field for user queries, and a label to display images. The AI processes text with a max_tokens limit of 200 for concise replies, while the GUI dynamically loads and resizes images to 200x200 pixels for a clean presentation.


*Figure 11- simple GUI for Agent*

*Phase 3: Image Display Integration*

To enable image visualization, I adjusted the AI prompt to insert [SHOW_IMAGE: phone_name] tags whenever a user asks to see a phone. The Tkinter GUI was enhanced with Pillow to load these images from the data folder and display them when the tag appears in the AI's response. The system works by accepting user input, forwarding it to the AI, showing the reply in the chat window, and rendering the appropriate phone image upon detecting the [SHOW_IMAGE] trigger.


*Figure 12- Image display*

## Results and Outputs

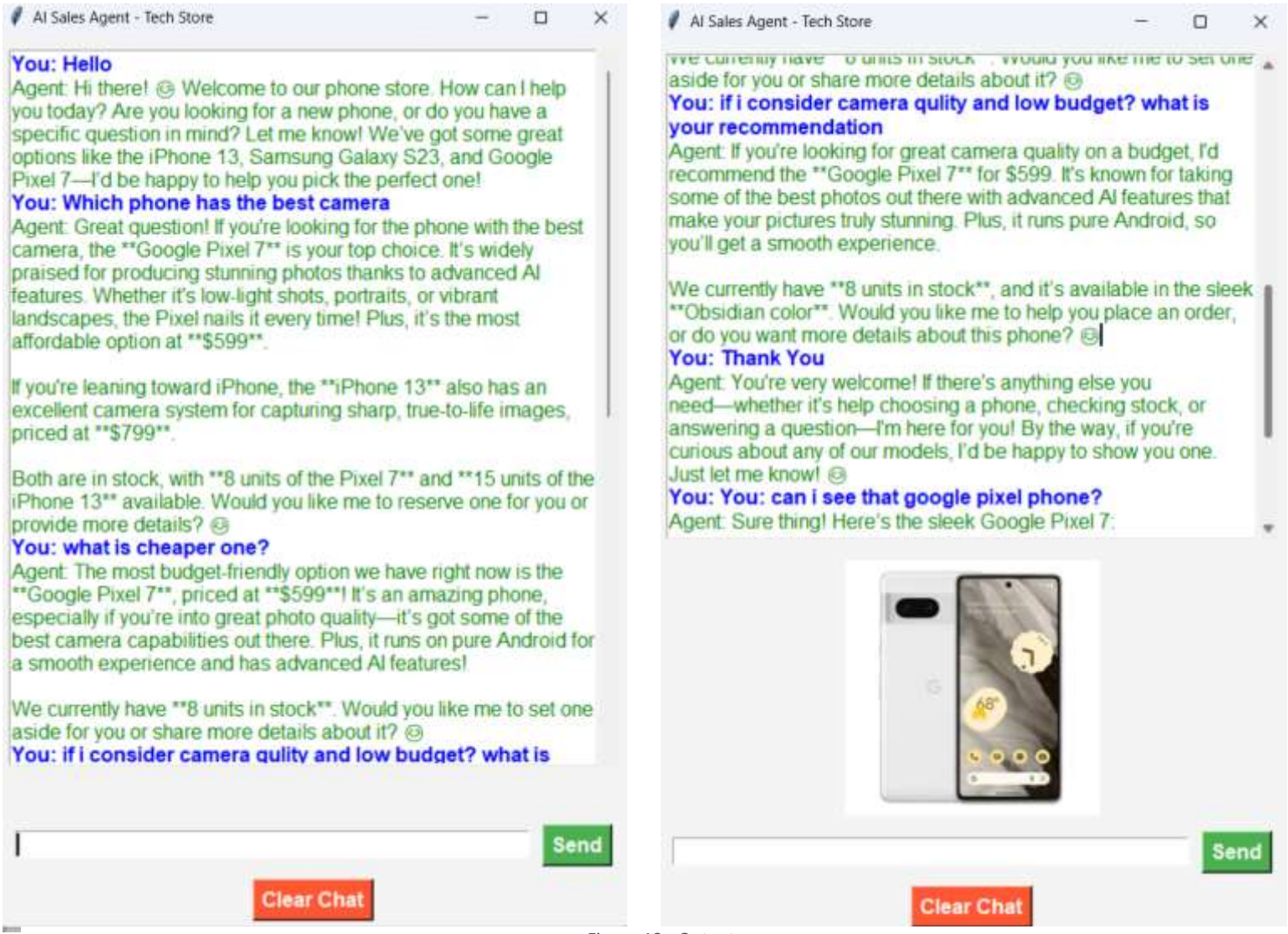



*Figure 13- Outputs*

## Challenges and Solutions

Challenge 1: Image Not Displaying Due to Name Mismatch

- **Issue:** AI response used [SHOW_IMAGE: iPhone_13] (underscore), but the catalog key was "iPhone 13" (space), causing the image display to fail.

- **Solution:** Updated the send_message function to handle both formats by replacing spaces with underscores dynamically (phone.replace(" ", "_")).

Challenge 2: Potential Image Path Errors

- **Issue:** Image files might not match the catalog paths (e.g., iPhone_13.jpg vs. iPhone 13.jpg), leading to "Image not available" errors.
- **Solution:** Standardized filenames to data/iPhone_13.jpg in the catalog and verified existence using os.path.exists() before loading.

## Additional Resources:

Demo Video: https://drive.google.com/file/d/129WBEh2815rOpyy_wy-xZjD3_ZGO22iU/view?usp=sharing

Task 1: Flutter App Development
Prepared By: Sahan
Date:24/02/2025

# Travel Destinations App

- Mobile Experience for Exploring Global Destinations-

## Problem Statement

The goal of this project was to develop a Flutter-based mobile application, "Travel Destinations App," that allows users to explore travel destinations with a futuristic, neon-themed user interface. The app needed to include a welcome screen, a home screen with listing destinations, and a details screen for each destination. Additional features included adding buttons for user interaction, applying consistent styling, and ensuring a visually appealing design with animations and transparency effects. This document summarizes the solutions implemented, demonstrates the app's functionality with screenshots, and highlights challenges faced during development.

The app uses the *provider* package, with a *ChangeNotifierProvider* in main.dart supplying a *DestinationProvide*r to manage a list of destinations, enabling HomeScreen and DetailsScreen to access and display data efficiently while ensuring scalability and a single source of truth.

## Approach & Results

*Phase 1: Setting Up the Project Structure and Data Model*

Initialized a Flutter project with provider and google_fonts, creating a Destination model and DestinationProvider for managing destinations (Paris, Tokyo, New York), establishing a foundation for dynamic data display.

*Phase 2: Designing the Welcome Screen*

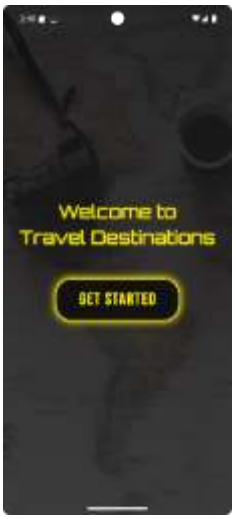



*Figure 14-welcome*

*Phase 3: Building the Home Screen*

Built the HomeScreen with a ListView.builder for DestinationCard widgets, a graphical background, and a neon yellow AppBar, delivering a scrollable destination list with a futuristic look.


*Figure 16-Home Screen*

*Phase 4: Creating the Details Screen*

Developed the DetailsScreen using a CustomScrollView with a SliverAppBar and SliverList for details, adding a Row of neon-themed buttons ("View on Google Maps," "Book a Trip"), providing a detailed view with interactivity.


*Figure 17- Details_Screen*

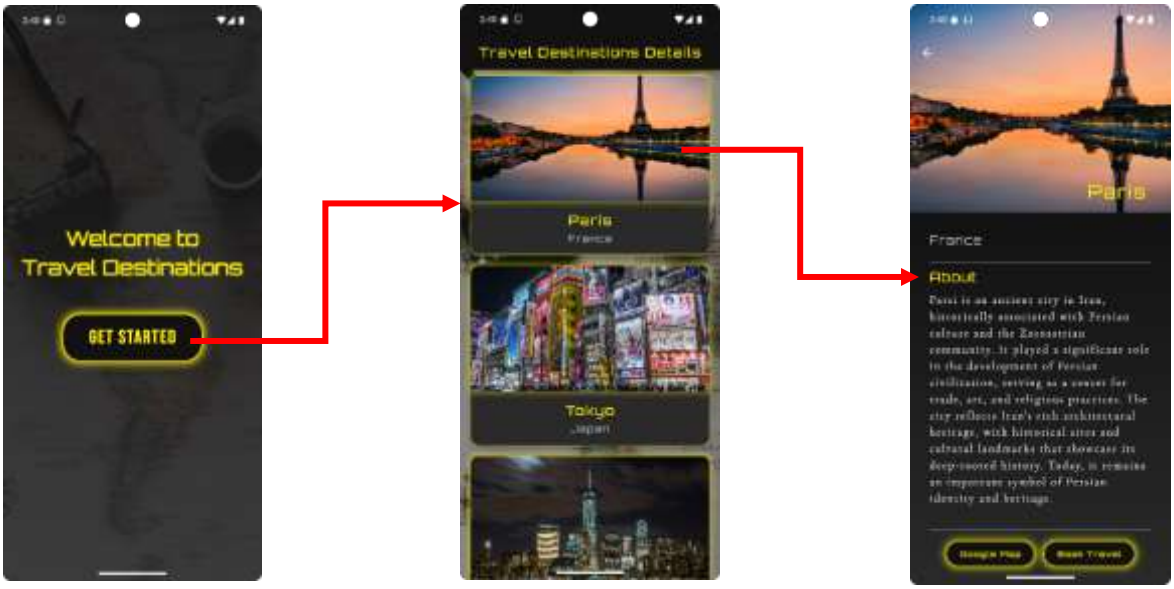



*Figure 18-App flow*

## Additional Resources:

**Demo Video**: https://drive.google.com/file/d/1j9J3ZSJGVE3tI5kjGiJSHgmm-TodP1ZP/view?usp=sharing

**GitHub link:**

Task 1: Web Development
Prepared By: Sahan
Date:24/02/2025

# Landing page for robots

- web page for sell robots-

## Problem Statement

The objective was to create a responsive landing page for a fictional product or service, showcasing proficiency in web development. The task required designing a page for "RoboAI," a company offering smart AI robots for various needs (home assistance, industrial automation, and security). The goal was to promote these innovative solutions to potential customers through an engaging, functional, and visually appealing interface.

## Approach

- **HTML**: Build a semantic structure with a fixed navigation bar, a hero section, a features grid, and a footer.
- **CSS**: Use Flexbox for the navbar, CSS Grid for the features section, and media queries for responsiveness across devices (desktop to mobile). I chose a futuristic dark theme with yellow accents to match the robotics focus, incorporating animations for visual appeal.
- **JavaScript**: Add interactivity with a scroll effect for the navbar, a hamburger menu toggle for mobile, and dropdown menu functionality.

## Result and Output

### Desktop View - Full Page


*Figure 19-Desktop view*

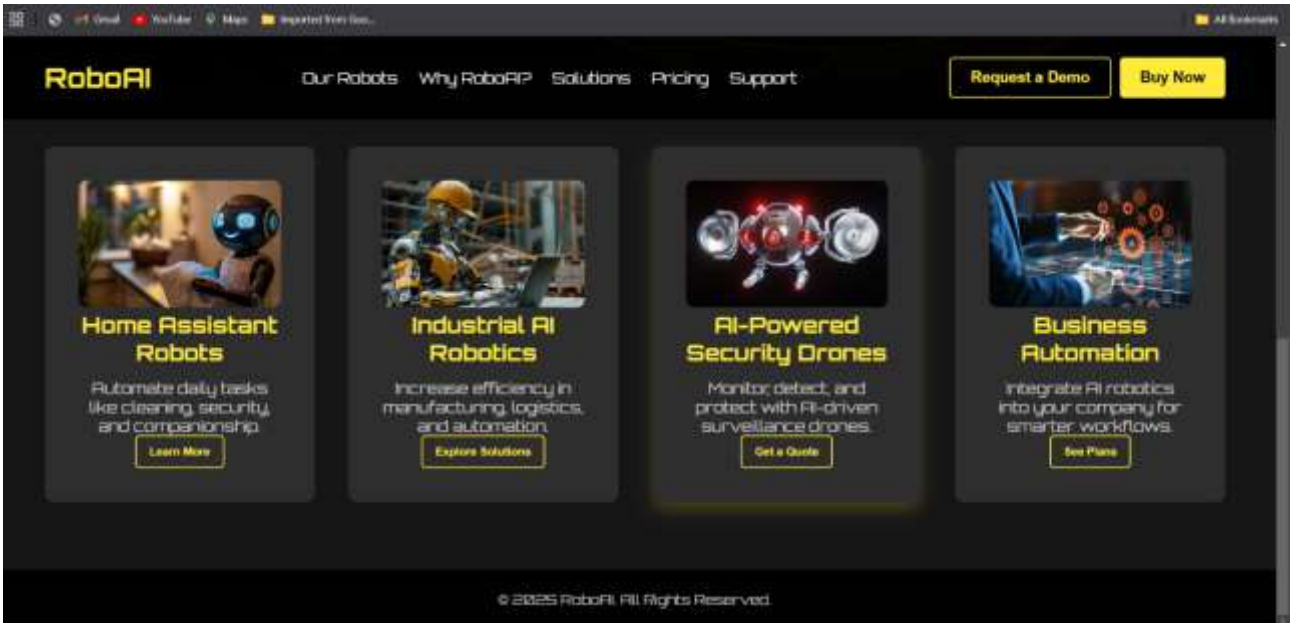
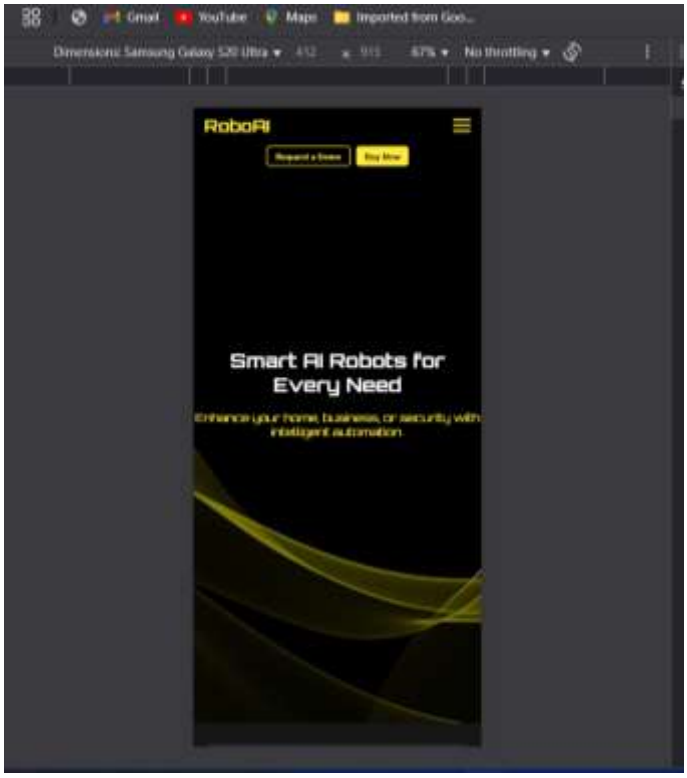



*Figure 20-Desktop View*

## Mobile View ( more on demo video)



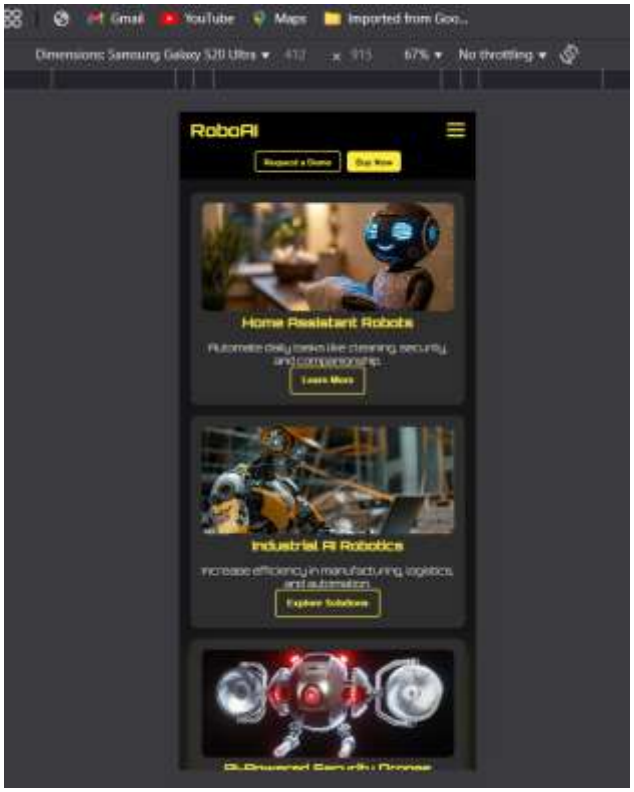




*Figure 21-Mobile view*

## Challenges and Solutions

**Challenge:** The desktop navbar with hover-based dropdowns didn't work intuitively on mobile devices.
**Solution**: I added a hamburger menu that shows up on mobile (screens 768px or smaller). JavaScript makes it open and close the navigation links when clicked. I also made the dropdown menus work with a tap instead of a hover on mobile, using event listeners. CSS changes collapse the layout to fit smaller screens and keep it easy to use.

**Challenge**: Initial animations for text and cards felt too slow or abrupt, disrupting the user experience.
**Solution**: Adjusted animation durations (e.g., fadeIn at 1s, slideUp at 0.8s) and added staggered delays (e.g., 0.2s increments for feature cards) to create a smooth, professional effect.

## Additional Resources:

**Demo Video:** https://drive.google.com/file/d/1kQ3aNbEilIzmXeNDJsMWgFL8UPd2-jsZ/view?usp=sharing