

Project Report – LandingLens Object Detection

Project Title:

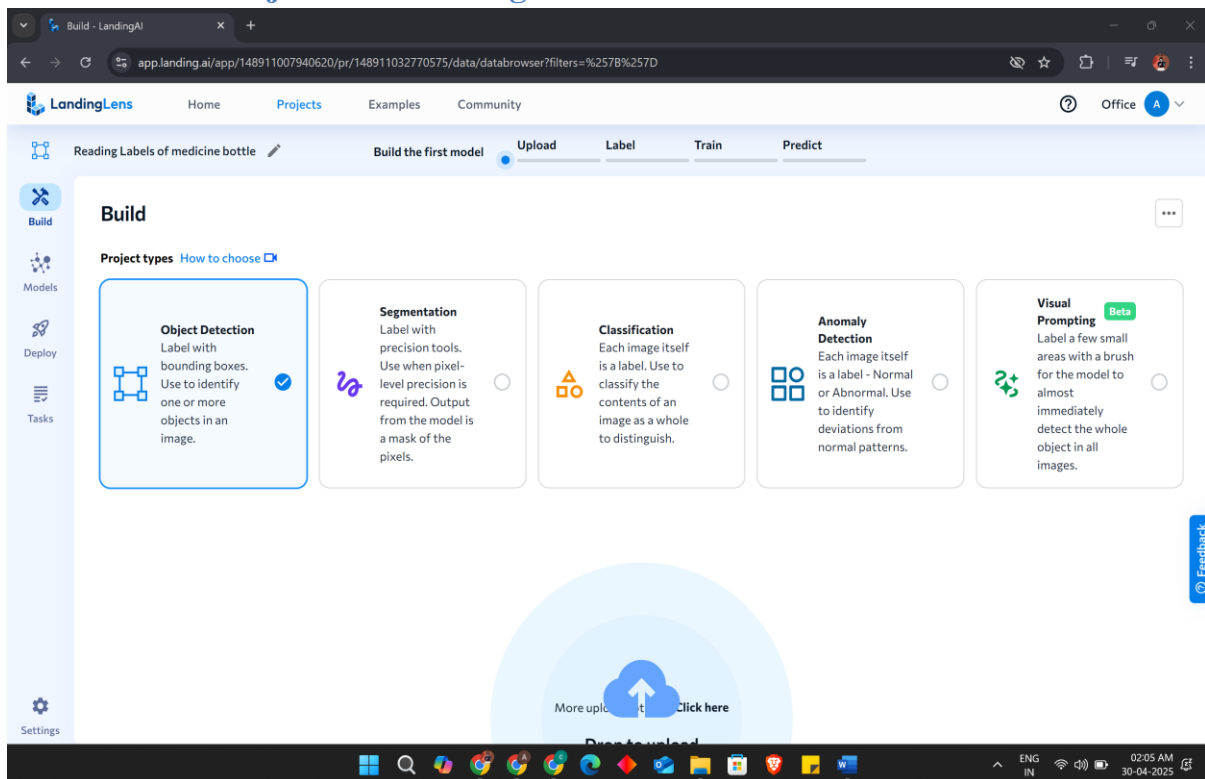
Cognitive Assistant for Reading Labels of Medicine Bottles

Project Idea:

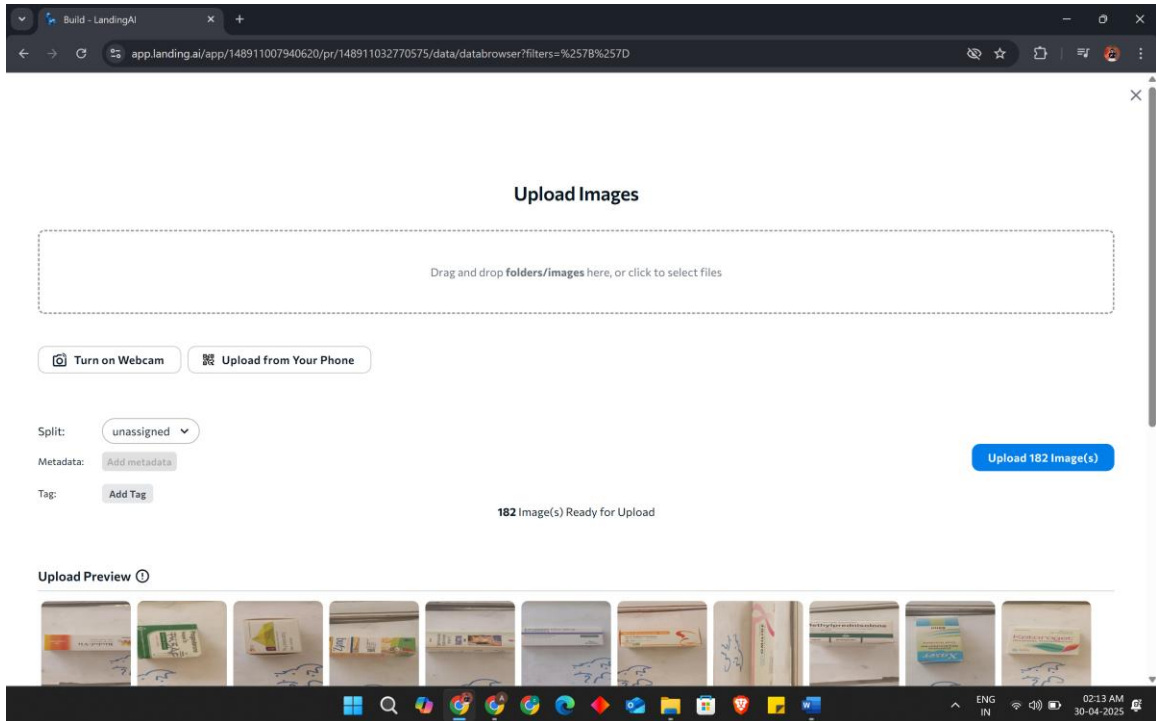
This project involves developing a cognitive assistant capable of reading and interpreting labels on medicine bottles using computer vision. By uploading various images of labeled bottles to LandingLens, a model is trained to detect and extract important information such as medicine name, dosage, and expiry date. This assistant helps patients, especially the elderly or visually impaired, in accurately identifying medicines.

Screenshots

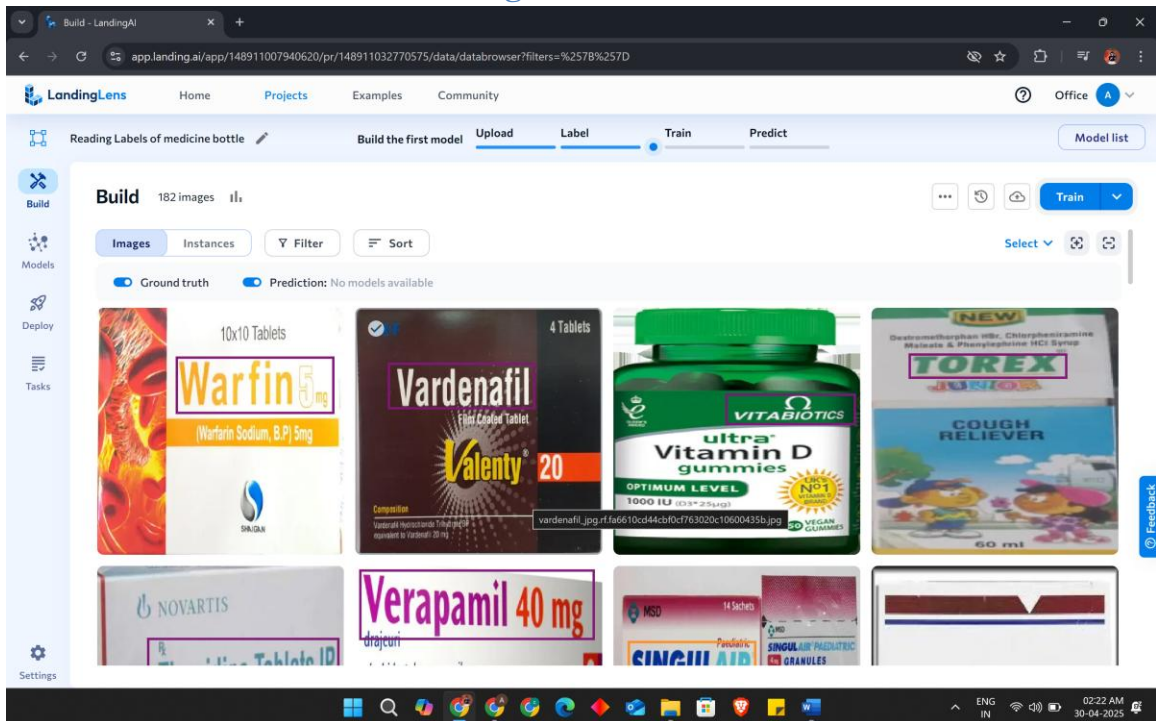
Screenshot 1: Project Creation Page



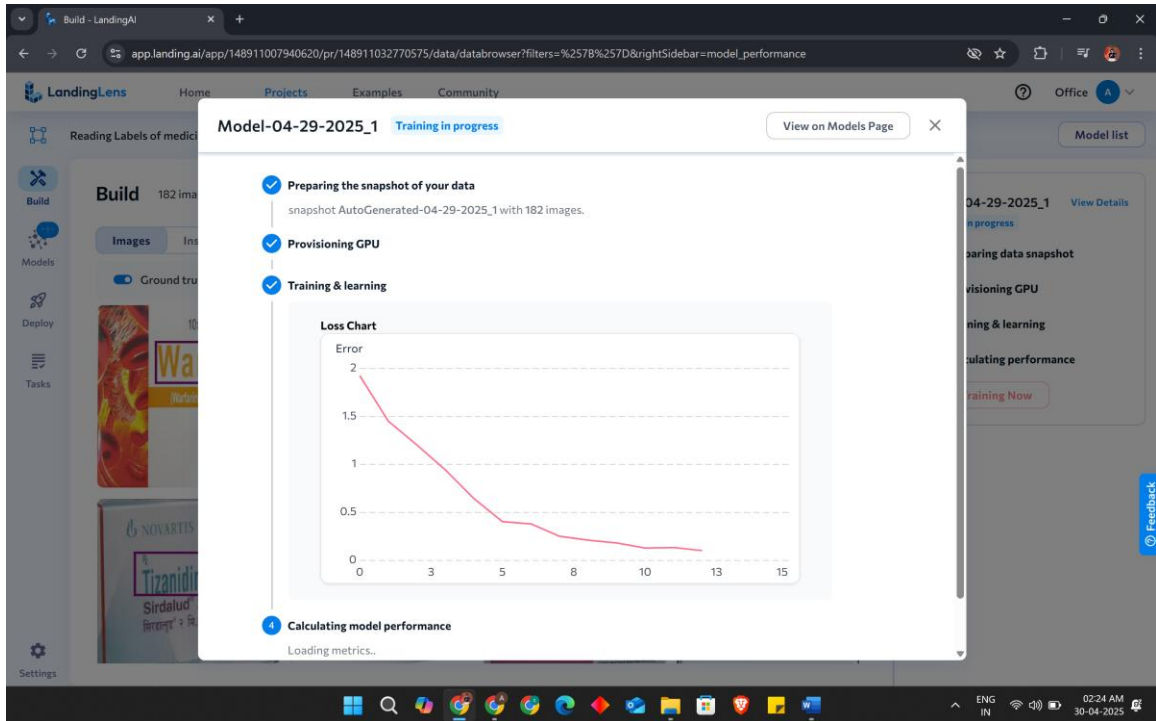
Screenshot 2: Image Upload Step



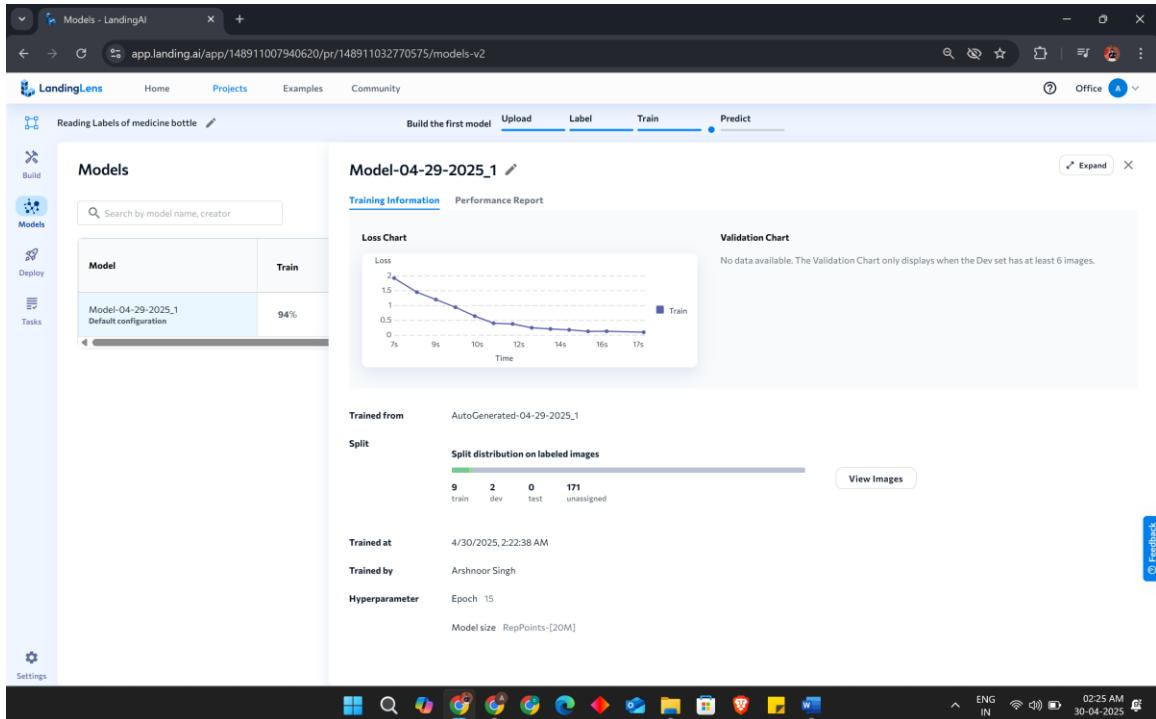
Screenshot 3: Annotation/Labeling Tool

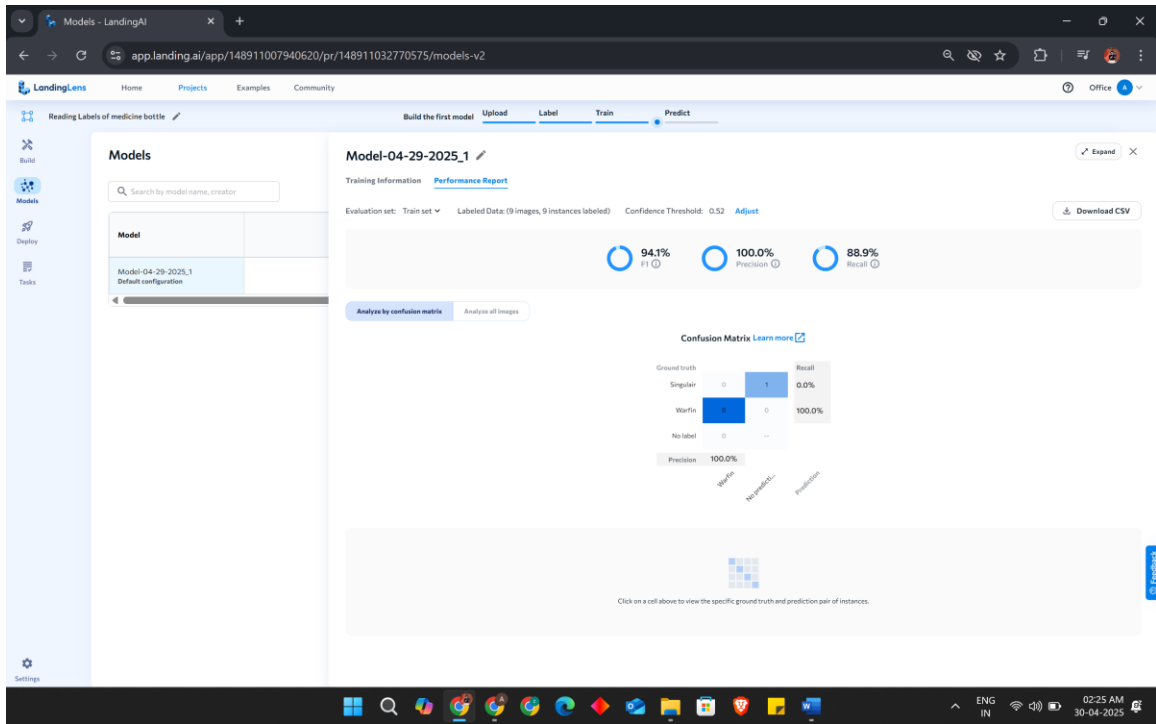


Screenshot 4: Training Progress

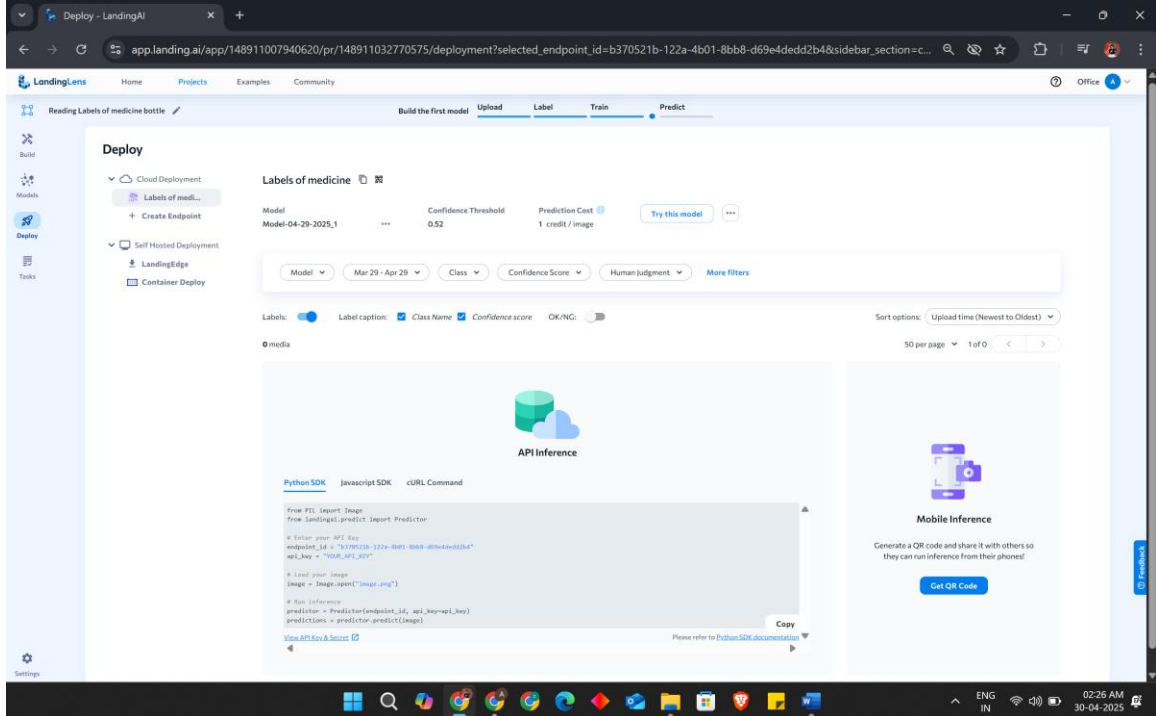


Screenshot 5: Testing Results





Screenshot 6: Deployment or API Interface



Project Report – LandingLens Classification

Project Title:

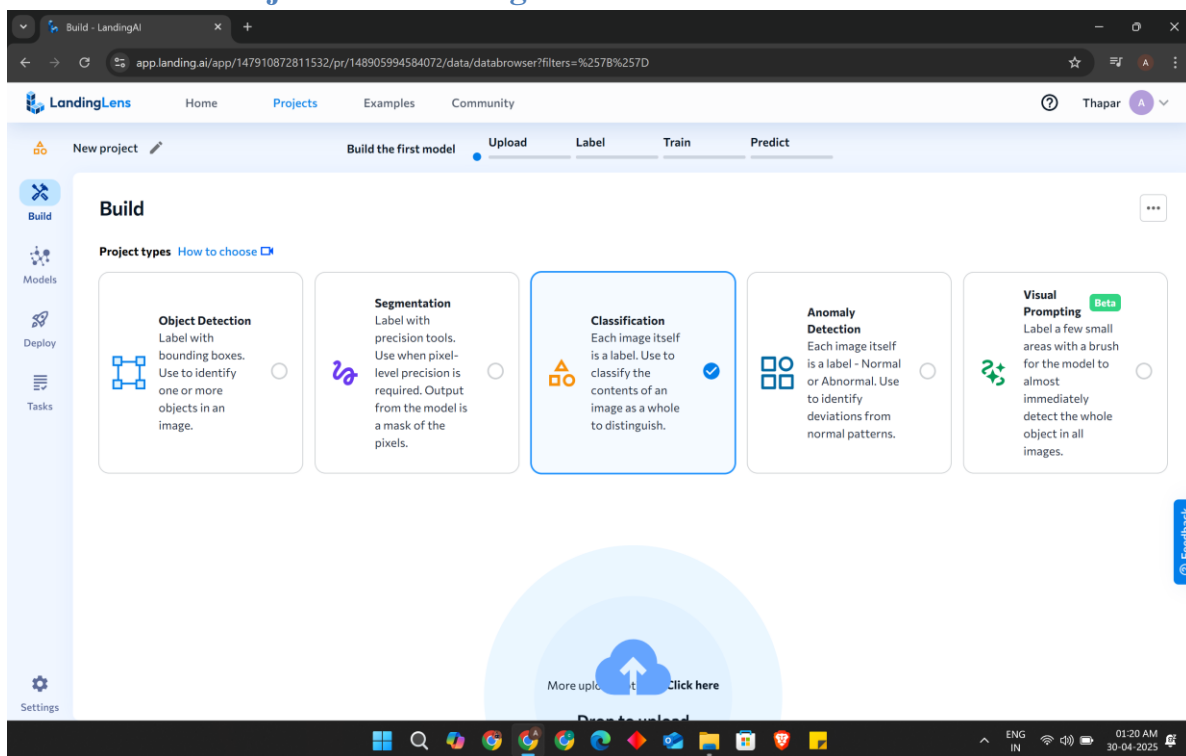
Emotion-aware Cognitive Assistant

Project Idea:

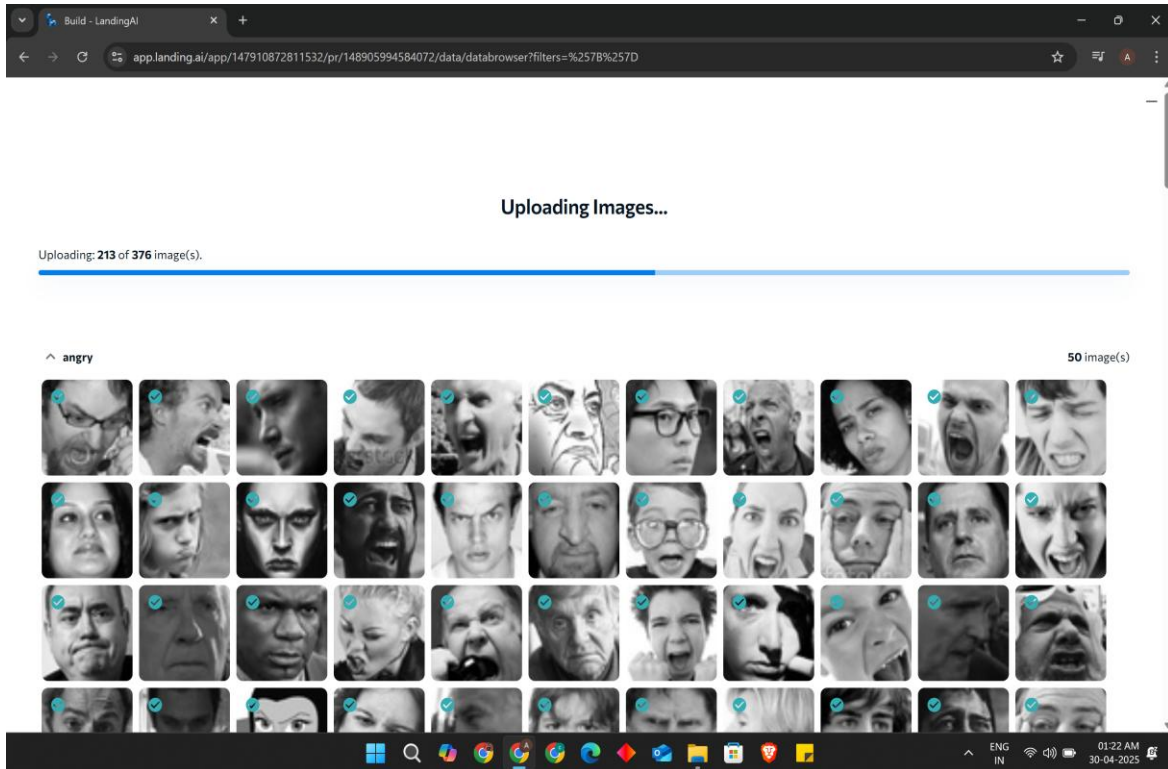
This project aims to develop an emotion-aware cognitive assistant that detects human emotions through facial expression analysis. Using LandingLens, images of different emotional expressions are uploaded and labeled. The trained model classifies emotions like happiness, anger, sadness, etc., in real time. This assistant can be integrated into interactive systems for enhanced user experience and mental health monitoring.

Screenshots

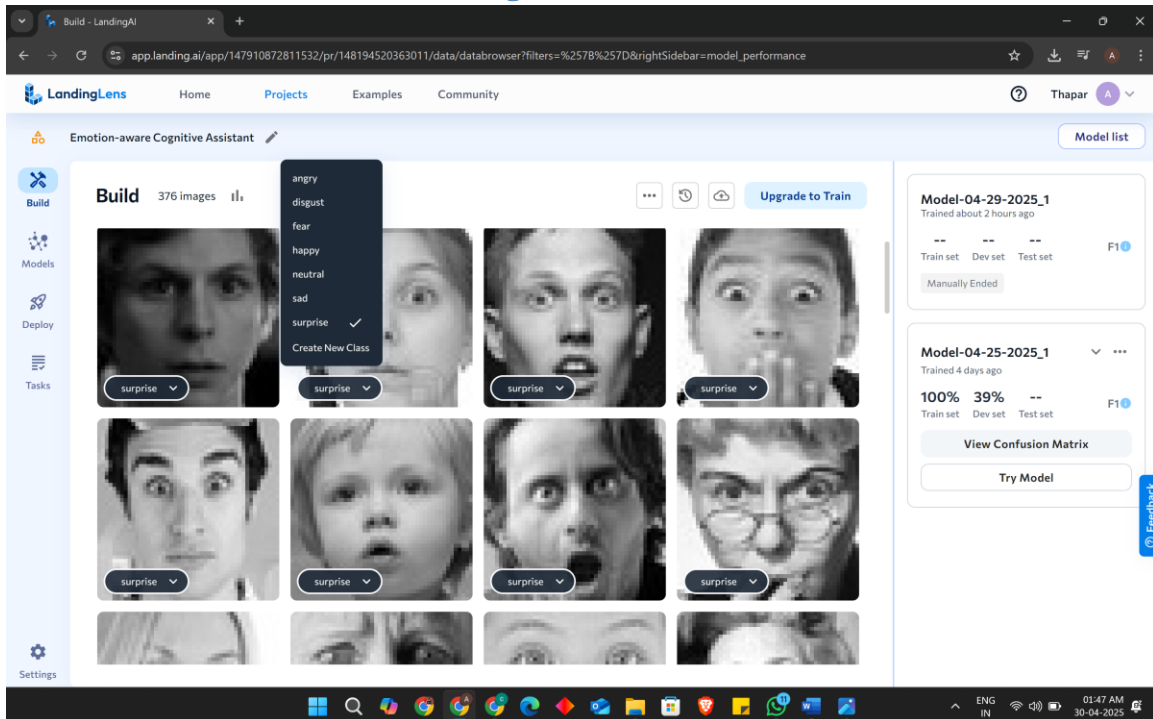
Screenshot 1: Project Creation Page



Screenshot 2: Image Upload Step



Screenshot 3: Annotation/Labeling Tool

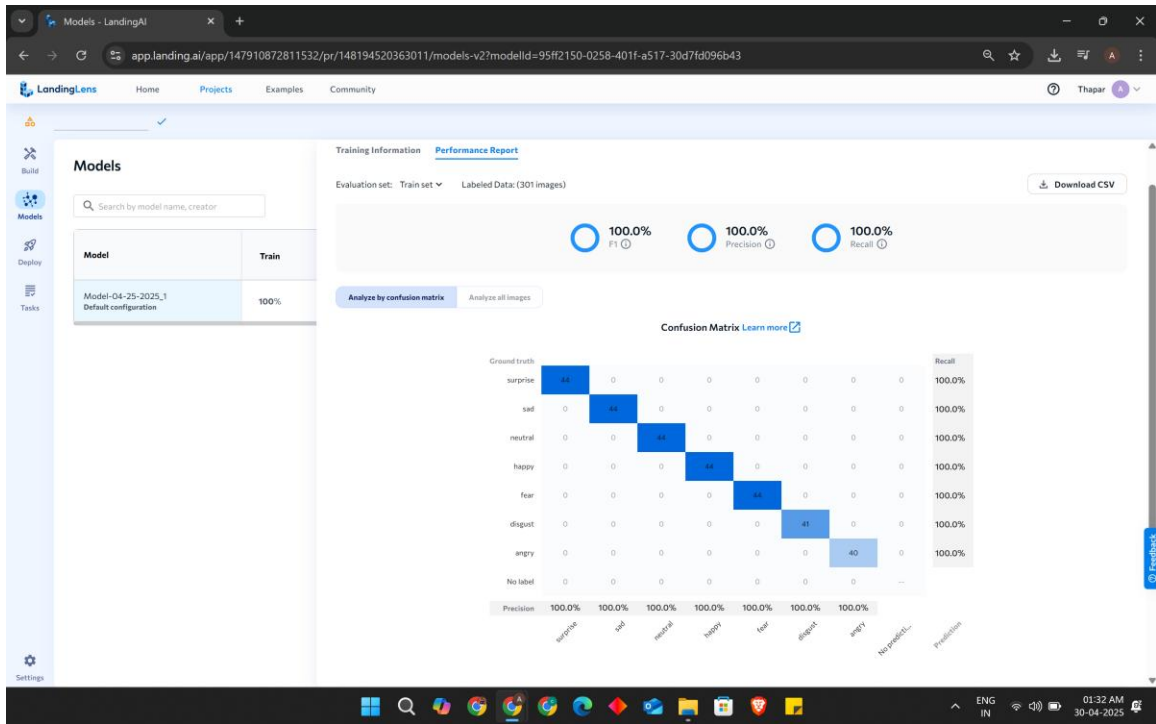


Screenshot 4: Training Progress

The screenshot shows the 'Build' tab in the LandingLens interface. The top navigation bar includes 'Home', 'Projects', 'Examples', and 'Community'. The main header has tabs for 'New project', 'Build the first model', 'Upload', 'Label', 'Train', and 'Predict'. The 'Train' tab is active, showing a progress bar with four steps: 'Preparing data snapshot', 'Provisioning GPU', 'Training & learning', and 'Calculating performance'. The 'Training & learning' step is currently in progress. Below the progress bar, there is a grid of image thumbnails showing faces with 'surprise' labels. The right sidebar shows the 'Model-04-29-2025_1' details, including a 'View Details' link and a 'Feedback' button.

Screenshot 5: Testing Results

The screenshot shows the 'Models' tab in the LandingLens interface. The top navigation bar includes 'Home', 'Projects', 'Examples', and 'Community'. The main header has tabs for 'New project', 'Build the first model', 'Upload', 'Label', 'Train', and 'Predict'. The 'Models' tab is active, showing a list of models. The 'Model-04-25-2025_1' model is selected, and its details are displayed. The 'Training Information' section shows the 'Loss Chart' and 'Validation Chart'. The 'Loss Chart' shows the training loss decreasing over time, and the 'Validation Chart' shows the validation F1 score increasing over time. The 'Split' section shows the distribution of images: 301 train, 75 dev, 0 test, and 0 unassigned. The 'Trained at' section shows the training time as 4/26/2025, 12:55:04 AM, and the 'Trained by' section shows the user as ARSHNOOR SINGH. The 'Hyperparameter' section shows the model size as ConvNext-(29M).



Screenshot 6: Deployment or API Interface

The screenshot shows the 'Deploy' page for the 'Emotion-aware Cognitive Assistant' model. The model is 'Model-04-25-2025,1' with a prediction cost of 1 credit/image. The deployment options are 'Cloud Deployment' (Emotion-aware...) and 'Self Hosted Deployment' (LandingEdge, Container Deploy). The 'API Inference' section shows the 'Python SDK' with a code snippet for inference. The 'Mobile Inference' section shows a QR code for mobile inference.

API Inference

```
from PIL import Image
from landingai.predict import Predictor

# Enter your API key
endpoint_id = "ba8777e-119-88fc-83da-87b07fab0a37"
api_key = "705d_a9f1_4917"

# Load your image
image = Image.open("image.png")

# Run inference
predictor = Predictor(endpoint_id, api_key=api_key)
predictions = predictor.predict(image)
```

Mobile Inference

Generate a QR code and share it with others so they can run inference from their phones!

[Get QR Code](#)