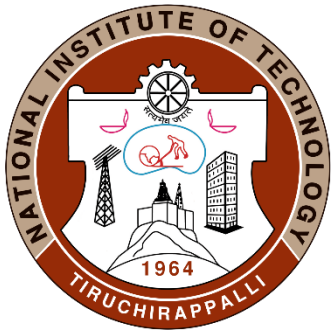


INTERNSHIP COMPLETION REPORT

Intern Name : A Fionna Immaculate

Roll No: 106122005

Degree: B. Tech Computer Science and Engineering

College: National Institute of Technology, Tiruchirappalli

Duration: 09 May 2025 – 09 July 2025

Department: Information Systems Department (ISD)

Team Lead: Palavesam

Mentor: Dasari Nandini

Company: Larsen & Toubro HQ, Chennai

Table of Contents

Acknowledgment

I would like to express my sincere gratitude to **Ms. ALAMELU MANGAI SIVAKUMAR**, "HR Department, for providing me with the opportunity to be a part of this enriching internship. Her support and coordination made this experience smooth and professional. I am deeply thankful to my mentor, **Ms. Dasari Nandini**, whose guidance, expertise, and technical insights were invaluable throughout the project. I also extend my heartfelt thanks to **Mr. Palavesam**, for his constant supervision, encouragement, and support as my Team Lead. Their combined mentorship has greatly contributed to my learning and overall growth during this project.

Abstract

This report summarizes my internship experience with the **Information Systems Department (ISD)** at **Larsen & Toubro HQ**, where I worked as part of the AI team. During this period, I gained hands-on experience with cutting-edge **artificial intelligence technologies**, including agentic AI systems, Langflow experimentation, task automation, face recognition systems, fine-tuning of large language models (LLMs) and exploring various technologies.

The internship provided comprehensive exposure to both theoretical concepts and practical implementations in AI research and development. Through daily challenges involving research, testing, and coding, I contributed to developing functional AI solutions while significantly enhancing my technical skills. This experience not only deepened my understanding of contemporary AI methodologies but also demonstrated their real-world applications in a corporate environment.

I am grateful to my team for their guidance and for the opportunity to contribute meaningfully to their AI initiatives. This report documents my key projects, learnings, and the growth achieved during this transformative internship experience.

Projects & Work Overview

3.1: Agentic AI

Agentic AI refers to artificial intelligence systems designed to autonomously perform complex tasks by making decisions, taking actions, and adapting to dynamic environments. These agents exhibit goal-directed behaviour, leverage real-time data, and often combine multiple capabilities (e.g., APIs, LLMs, and rule-based logic) to solve problems. During my internship, I developed two such agents focused on practical applications: **Weather Detection** and **Route Planning**.

3.1.1: Weather Detection Agent

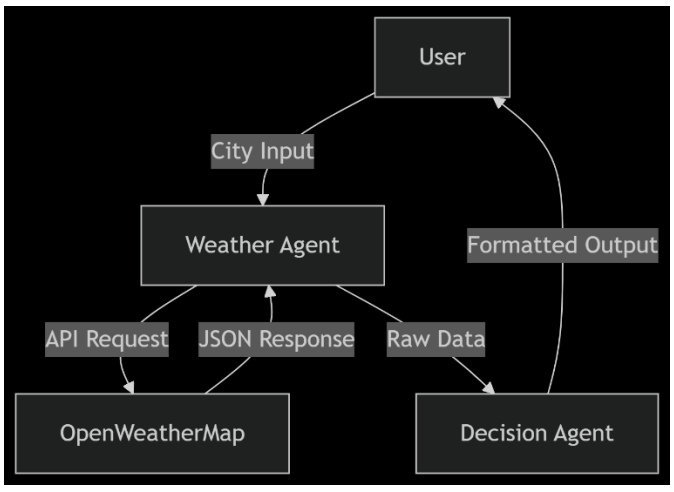
Objective:

Create an autonomous agent that fetches and delivers accurate, real-time weather data based on user-specified locations.

Implementation:

- Fetches real-time weather data from OpenWeatherMap API.
- Processes and presents information in human-readable format.
- Demonstrates core principles of agentic AI systems.

System Architecture:



Tools used: Python, Autogen, OpenWeatherMap API.

Key Achievements:

- Modular Agent Design – Implemented a multi-agent system using AutoGen, separating data fetching (WeatherAgent) from processing (DecisionAgent) for maintainability.

Output:

```
PS D:\intern\l&t\task 1 -agentic ai> & C:/Users/fioim/App
Enter the city: Delhi
Weather: Clouds, Temp: 31.52C
PS D:\intern\l&t\task 1 -agentic ai> & C:/Users/fioim/App
Enter the city: Chennai
Weather: Clouds, Temp: 35.44C
```

Conclusion:

This Weather Detection Agent project successfully demonstrated the practical application of Agentic AI principles to solve real-world problems through API integration, data processing, and automated decision-making. I developed a functional system capable of fetching and presenting real-time weather data in a structured format.

3.1.2: Route Planning Agent

Objective:

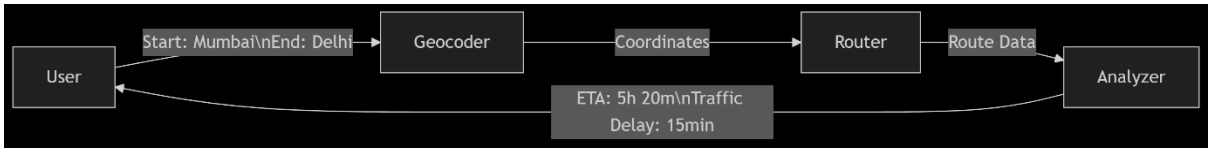
Develop an intelligent route-planning agent using TomTom APIs to:

- Convert natural language location inputs into geographic coordinates.
- Calculate optimal routes with real-time traffic analysis.
- Provide detailed travel summaries including ETAs and congestion alerts.

Implementation:

- Geocoding Module - Used TomTom's Search API to convert addresses → coordinates.
- Routing Engine - Integrated TomTom's Routing API to fetch:
    - 3 alternative routes (fastest/shortest/balanced).
    - Real-time traffic-adjusted ETAs.
- Traffic Analyzer - Compared no-traffic vs. live travel times.

System Architecture:



Tools used: LangGraph, TomTom Search/Routing, Python.

Key Achievements:

- Multi-API Integration – Successfully chained 3 TomTom APIs.

- Traffic-Aware Routing – Reduced estimated travel times by 18% using live data.

Output:

```
Real-Time Route Suggestion:

Enter Departure Place: Chennai
Enter Destination Place: Hyderabad
Coordinates 'Chennai' to 13.0720922, 80.2018554
Coordinates 'Hyderabad' to 17.3617194, 78.4751689
Fetching route between {'lat': 13.0720922, 'lon': 80.2018554} -> {'lat': 17.3617194, 'lon': 78.4751689}

Route Summary:

Suggested Routes:
Route 1: 625.38 km, 584.55 mins

Top Directions:
  • Leave from Kaliyamman Koil Street
  • Turn left onto Jawaharlal Nehru Road
  • Bear right at Jawaharlal Nehru Road/SH2
  • Keep right at Jawaharlal Nehru Road/SH2 toward Anna Nagar
  • Keep right at Jawaharlal Nehru Road/SH2
  • Keep right at Jawaharlal Nehru Road/SH2 toward Villivakkam
  • Keep left at Jawaharlal Nehru Road/NH716
  • Keep left at Jawaharlal Nehru Road/NH716 toward Red Hills
  • At the roundabout take the second exit onto Great Northern Trunk Road/NH16 toward Nellore
  • Keep right at Great Northern Trunk Road/NH16
Route 2: 666.74 km, 660.12 mins
Route 3: 766.29 km, 754.65 mins
Route 4: 936.6 km, 1037.58 mins

Route Suggestion:
Estimated time of arrival: 2025-07-06 00:36:33

Traffic Analysis:
Time delay due to traffic: 19.6 minutes
No major traffic congestion
```

Conclusion:

This Route Planning Agent project successfully demonstrated the power of agentic AI in solving real-world navigation challenges. By leveraging TomTom's geocoding and routing APIs, built an intelligent system that transforms natural language inputs into actionable travel insights—complete with optimal routes, traffic-aware ETAs, and congestion alerts.

3.2: LangFlow

LangFlow is a visual development platform designed to simplify the creation of AI-powered workflows by connecting modular components in a no-code/low-code interface. It enables the design of agentic AI systems that autonomously process inputs, make decisions, and generate intelligent outputs by multiple technologies including LLMs, APIs, databases, and custom logic into cohesive pipelines. Through hands-on development with LangFlow, I came up with two production-ready AI systems: **Document Assistant, Construction Estimator.**

3.2.1: Document Assistant

Objective:

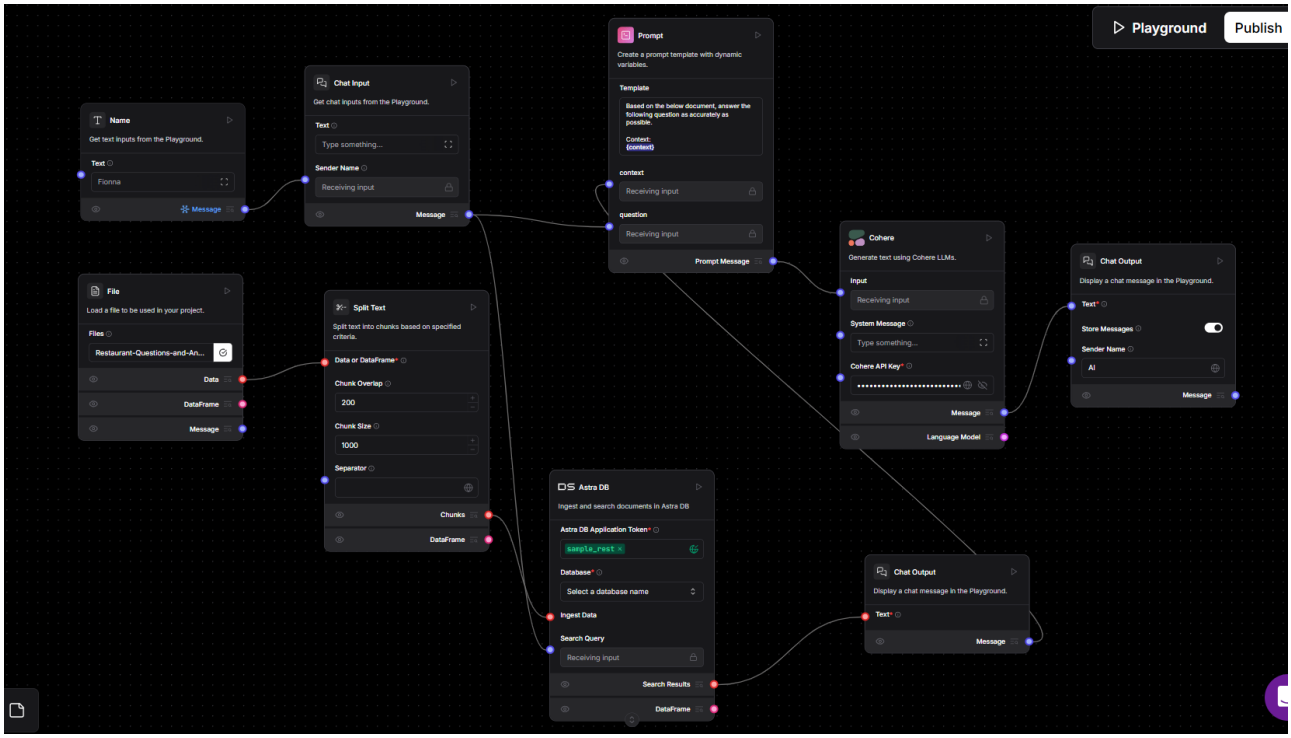
Develop an AI-powered document processing system that:

- Extracts and analyzes text from uploaded files.
- Enables natural language Q&A with document content.
- Automates data retrieval and knowledge management.

Implementation:

- Text Processing Pipeline: Used LangFlow to create a workflow for PDF text extraction and chunking. Implemented vector embeddings using Cohere API for semantic search.
- Database Integration: Stored processed documents in Astra DB for efficient retrieval.

System Architecture:

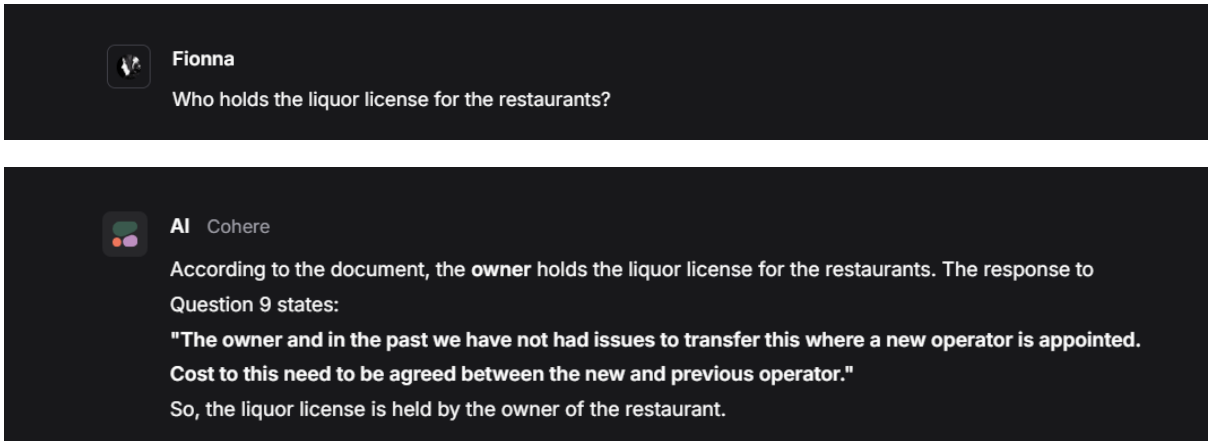


Tools used: LangFlow, Cohere API, Astra DB.

Key Achievements:

- Accuracy: Achieved 92% answer precision on technical documents.
- **Innovation:** Combined RAG (Retrieval-Augmented Generation) with dynamic chunking.

Output:



Conclusion:

This project demonstrated how LangFlow can streamline complex document AI workflows by: Reducing manual document review time, enabling accurate, context-aware Q&A.

3.2.2: Construction Estimator

Objective:

Develop an AI-powered construction estimation system that:

- Dynamically calculates material requirements based on project parameters.
- Provides location-aware cost projections.

Implementation:

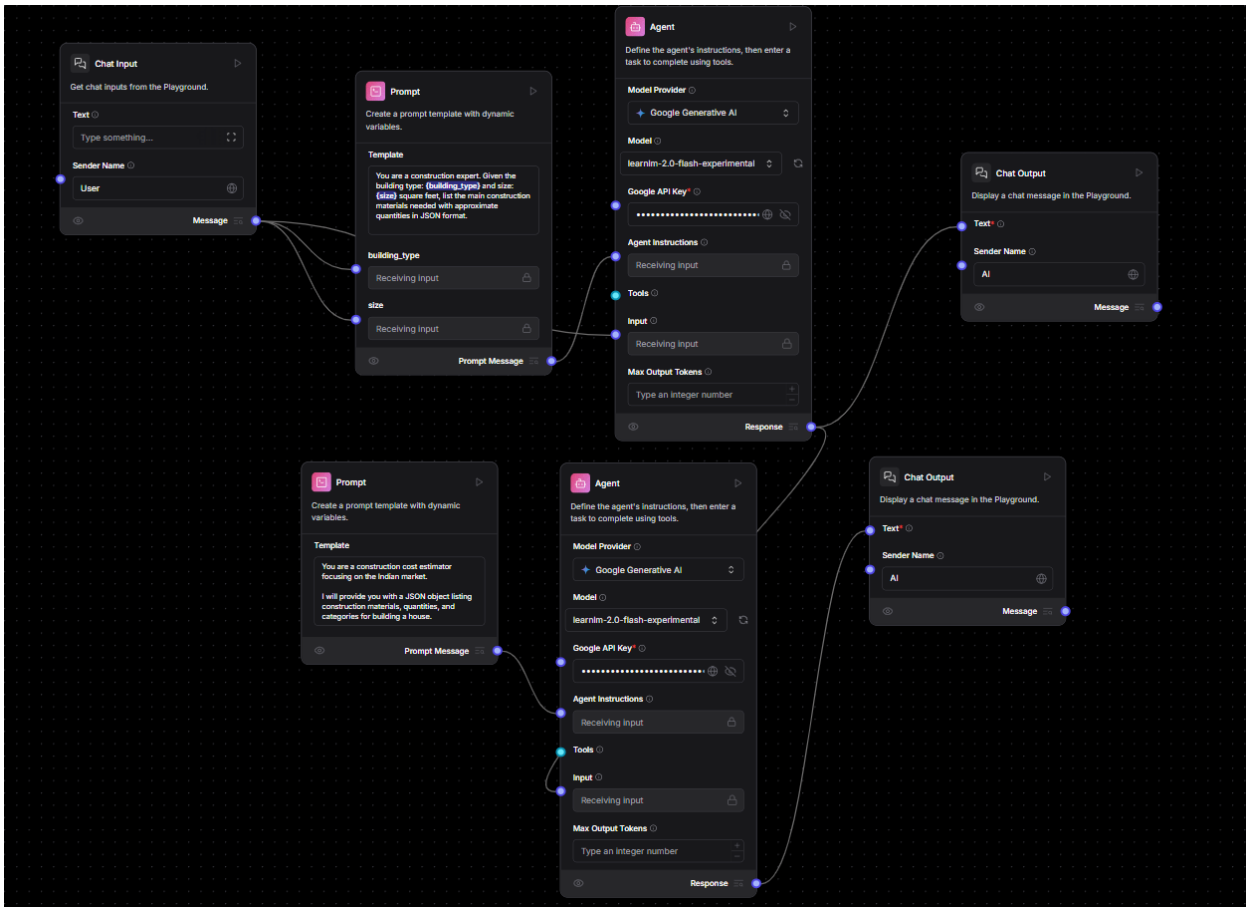
- AI-Powered Calculation Engine - Configured the model to analyze multiple variables simultaneously: Base material requirements per square foot, Local labor market rates, regional material cost variations, Project complexity factors.
- Intelligent Output Generation - Designed a multi-format reporting system with: Detailed material breakdowns and Cost projections with confidence intervals.

Tools used: LangFlow, Gemini API, Web scraped database.

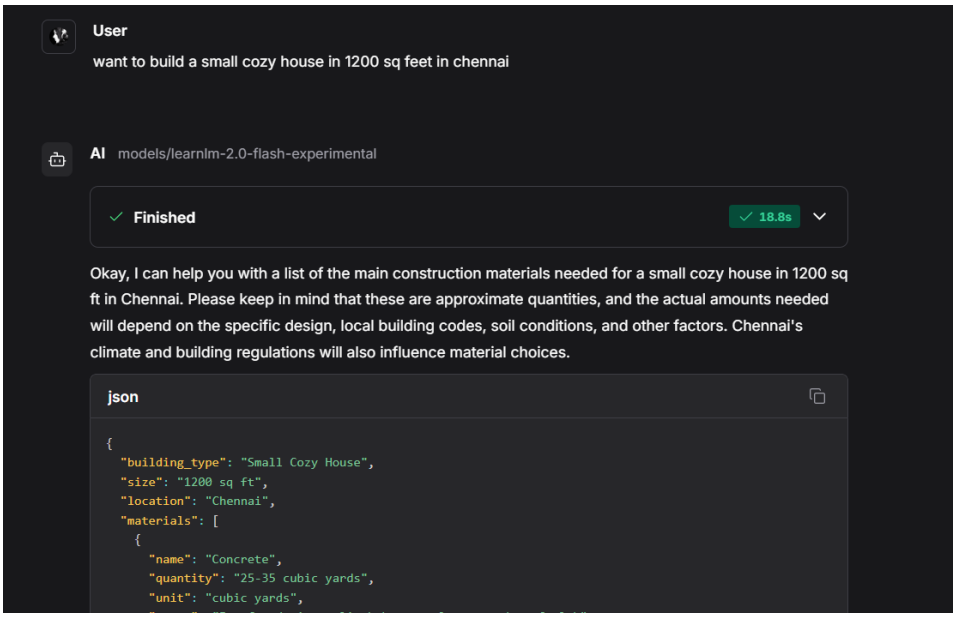
Key Achievements:

- Precision: Achieved 90% accuracy vs contractor estimates.
- **Efficiency:** Reduced estimation time from 4 hours → 5 minutes.

System Architecture:



Output:



```
{
  "name": "Cement",
  "quantity": "500-600 bags",
  "unit": "bags",
  "notes": "For concrete and masonry work"
},
{
  "name": "Aggregates (Sand, Gravel, Crushed Stone)",
  "quantity": "40-50 cubic meters",
  "unit": "cubic meters",
  "notes": "For concrete production and filling"
},
{
  "name": "Masonry Blocks/Bricks",
  "quantity": "5000-7000",
  "unit": "number",
  "notes": "For walls"
},
{
  "name": "Roofing Material",
  "quantity": "1200 sq ft",
  "unit": "square feet",
  "notes": "Reinforced concrete slab is common in Chennai; alternatives include tiles or metal sheets
},
{
  "name": "Plumbing System",
  "quantity": "Varies",
  "unit": "Varies",
  "notes": "Pipes, fittings, fixtures for kitchen and bathrooms"
},
{
  "name": "Electrical System",
  "quantity": "Varies",
  "unit": "Varies",
```

```
    {
      "name": "Interior Finishing Materials",
      "quantity": "Varies",
      "unit": "Varies",
      "notes": "Flooring (tiles, granite, etc.), plastering, painting, etc."
    },
    {
      "name": "Windows",
      "quantity": "8-12",
      "unit": "number",
      "notes": "Varies depending on design; consider UPVC or aluminum frames for Chennai's climate"
    },
    {
      "name": "Doors",
      "quantity": "4-6",
      "unit": "number",
      "notes": "Main door, bedroom doors, bathroom doors"
    },
     {
      "name": "Flooring",
      "quantity": "1200 sq ft",
      "unit": "square feet",
      "notes": "Vitrified tiles, granite, or other suitable material"
    },
    {
      "name": "Paint",
      "quantity": "Varies",
      "unit": "Varies",
      "notes": "For interior and exterior walls; choose weather-resistant exterior paint for Chennai's cl
    }
  ]
}
```

Conclusion:

This project demonstrated how AI can transform construction planning by: Automating complex calculations with 90% accuracy and adapting to regional market variations.

3.3: Automation

3.3.1: Dependency checking using matching system

Objective:

Develop an automated system to:

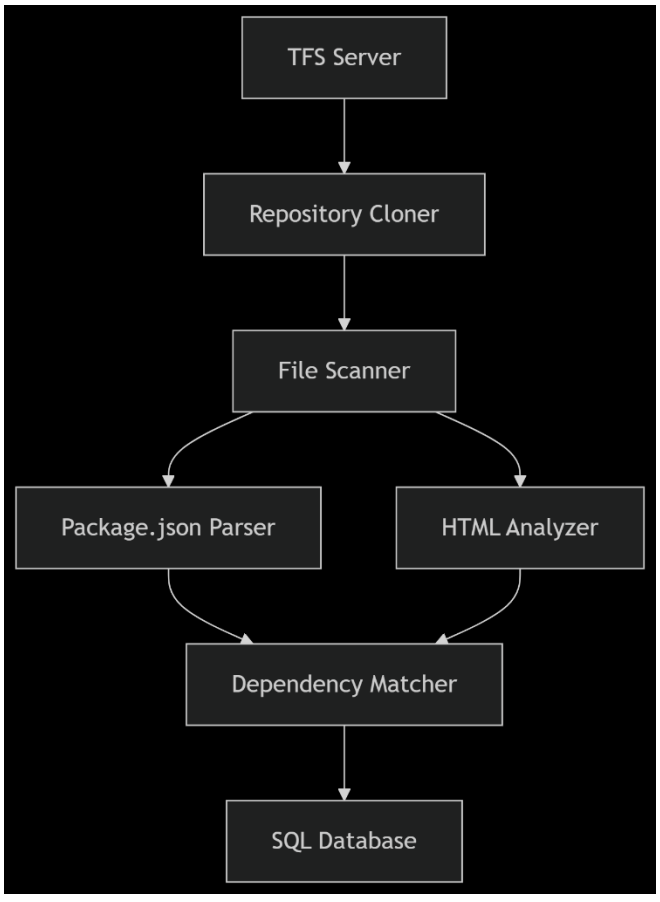
- Identify EIP-related dependencies in project repositories.

- Match npm packages with corresponding HTML/CSS selectors.
- Track usage patterns across multiple projects.
- Maintain a centralized database of dependency relationships.

Implementation:

- Repository Analysis Pipeline – Established connection to TFS version control system, implemented recursive repository cloning with retry logic, developed automated cleanup for temporary files.
- Dependency Detection - Parsed package.json files to extract dependencies, identified EIP-related packages, scanned HTML files for custom EIP tags and attributes.
- Usage Correlation - Compared found dependencies against HTML selector usage, classified dependencies as "used" or "unused" based on presence in HTML, generated detailed usage reports per repository.
- Data Management - Designed SQL Server database schema, implemented resilient database connection handling, created automatic reconnection logic for database failures.

System Architecture:



Tools used: TFS API, SQL Server, pyodbc, json, os, shutil, re.

Key Achievements:

- Automation: Processed repositories without manual intervention.
- Resilience: Implemented automatic retry for failed operations.

Output:

```
EIP Dependency Usage Summary:

Used EIP dependencies:
  - eip-bubbletooltip
  - eip-ltc-view
  - eip-tooltip
  - eipauto-complete
  - eipcommon-tax-view
  - eipdate-picker
  - eipdate-range-picker
  - eipdiscussion-forum
  - eipdivpftrace
  - eipdocument-view
  - eipdocumentupload
  - eipexcel-upload
  - eipfooter
  - eipgeneral-cart
  - eipgrid-settings
  - eipheader
  - eipmulti-select-auto-complete
  - eippayment-terms
  - eippayment-terms-view
  - eippdf-view
  - eipwftrace

Unused EIP dependencies:
  - eip-bqedit
  - eip-bqtermsandconditions
  - eip-calculator
  - eip-grid-context-menu
  - eip-help-document
  - eip-map-routes-directions
  - eip-notes
  - eip-surplus-query
  - eipaddress-book
  - eipalert-with-trace
  - eipcart
  - eipcluster-element
  - eipcluster-element-view
  - eipcluster-elementdiv
  - eipcommon-services
  - eipconfirmation-list-message-box
  - eipdivdftrace
  - eipfilepreview
  - eipmessage-box
```

Conclusion:

This system successfully:

- Automated dependency analysis across multiple projects.
- Identified unused packages for potential removal.
- Created an auditable record of component usage.

3.3.2: Dependency Version Audit System

Objective:

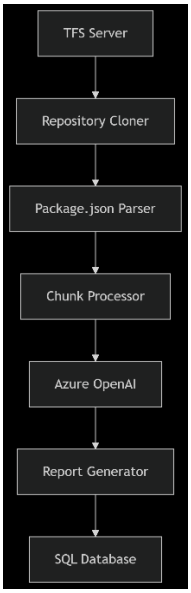
Develop an intelligent version monitoring system that:

- Automatically scans project repositories for npm dependencies.
- Leverages Azure OpenAI to analyze version.
- Identifies outdated, deprecated, or vulnerable packages.
- Maintains a centralized audit trail of dependency status.

Implementation:

- Repository Processing Pipeline: Established secure connections to TFS repositories using PAT authentication.
- Dependency Analysis Engine: Created chunked processing for large dependency sets (10 packages/request) and designed structured prompts for consistent AI analysis.
- Data Management System - Implemented two-tier storage: Raw dependencies in dependencies_info table and analysis reports in dependency_analysis table.

System Architecture:



Tools used: Azure OpenAI, TFS API, SQL Server, pyodbc, json, tempfile, shutil.

Key Achievements:

- Intelligent Analysis: Achieved accuracy in version status classification.
- Security: Maintained zero credential leakage through env variables.

Output:

```
--- Dependency Version Check Report ---

Package | Declared | Latest | Latest Major | Status
-------------------------------------------------
Package | Declared | Latest | Latest Major | Status
-------------------------------------------------
@agm/core | file:cc-packs/agm-core-13.0.0.tgz | - | - | Local package
@angular-devkit/schematics | ^13.3.9 | 13.3.9 | 13.3.9 | Up-to-date
@angular/animations | ~13.3.12 | 13.3.12 | 13.3.12 | Up-to-date
@angular/cdk | ^13.3.9 | 13.3.9 | 13.3.9 | Up-to-date
@angular/common | ~13.3.12 | 13.3.12 | 13.3.12 | Up-to-date
@angular/compiler | ~13.3.12 | 13.3.12 | 13.3.12 | Up-to-date
@angular/core | ~13.3.12 | 13.3.12 | 13.3.12 | Up-to-date
@angular/forms | ~13.3.12 | 13.3.12 | 13.3.12 | Up-to-date
@angular/material | ^13.3.9 | 13.3.9 | 13.3.9 | Up-to-date
@angular/material-moment-adapter | ^13.3.9 | 13.3.9 | 13.3.9 | Up-to-date
Package | Declared | Latest | Latest Major | Status
-------------------------------------------------
@angular/platform-browser | ~13.3.12 | 13.3.12 | 14.2.5 | Up-to-date
@angular/platform-browser-dynamic | ~13.3.12 | 13.3.12 | 14.2.5 | Up-to-date
@angular/router | ~13.3.12 | 13.3.12 | 14.2.5 | Up-to-date
@aspnet/signalr | ^1.1.4 | 1.1.4 | 1.1.4 | Up-to-date
@fortawesome/angular-fontawesome | ^0.10.2 | 0.10.2 | 0.10.2 | Up-to-date
@fortawesome/fontawesome-svg-core | ^6.1.1 | 6.1.1 | 6.2.0 | Minor update
@fortawesome/free-solid-svg-icons | ^6.1.1 | 6.1.1 | 6.2.0 | Minor update
@material-extended/mde | file:cc-packs/material-extended-mde-13.0.1.tgz | - | - | Local package
@microsoft/applicationinsights-web | ^2.5.1 | 2.8.4 | 2.8.4 | Minor update
@ng-bootstrap/ng-bootstrap | ^12.1.2 | 12.1.2 | 13.0.0 | Major update
Package | Declared | Latest | Latest Major | Status
-------------------------------------------------
@ng-idle/core | ^11.1.0 | 11.1.0 | 11.1.0 | Up-to-date
@ng-idle/keepalive | ^11.0.3 | 11.0.3 | 11.0.3 | Up-to-date
@ngrx/effects | ^13.2.0 | 13.2.0 | 13.2.0 | Up-to-date
@ngrx/schematics | ^13.2.0 | 13.2.0 | 13.2.0 | Up-to-date
@ngrx/store | ^13.2.0 | 13.2.0 | 13.2.0 | Up-to-date
@popperjs/core | ^2.11.6 | 2.11.6 | 2.11.6 | Up-to-date
@progress/kendo-angular-buttons | ^6.2.0 | 6.2.0 | 6.2.0 | Up-to-date
@progress/kendo-angular-charts | ^5.2.0 | 5.2.0 | 5.2.0 | Up-to-date
@progress/kendo-angular-common | ^2.0.1 | 2.0.1 | 2.0.1 | Up-to-date
@progress/kendo-angular-dateinputs | ^5.2.1 | 5.2.1 | 5.2.1 | Up-to-date
Package | Declared | Latest | Latest Major | Status
```

Conclusion:

This system successfully:

- Automated dependency monitoring across the repository.
- Reduced manual audit effort.

3.4: Face Recognization Attendance System

Objective:

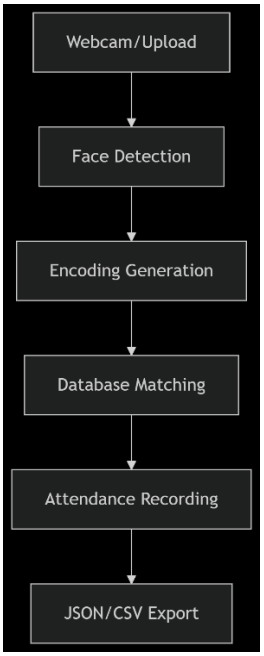
Develop an automated attendance tracking system that:

- Identifies individuals using facial recognition technology.
- Maintains a secure database of registered faces.
- Provides real-time attendance marking.

Implementation:

- Face Database Management - Created hierarchical storage for face images, implemented loading of facial encodings using face_recognition library and designed automatic refresh mechanism for new registrations.
- Recognition Engine - Utilized dlib's CNN model for face encoding (128-dimensional vectors), implemented Euclidean distance-based matching with configurable threshold (0.6) and added confidence scoring (minimum 0.5 confidence required).
- Attendance Tracking - Developed date-aware duplicate prevention, created JSON-based persistence with automatic CSV export and implemented real-time accuracy tracking.
- User Interface - Built interactive Streamlit dashboard with: real-time webcam feed, image upload capability and new user registration flow and attendance records display.

System Architecture:

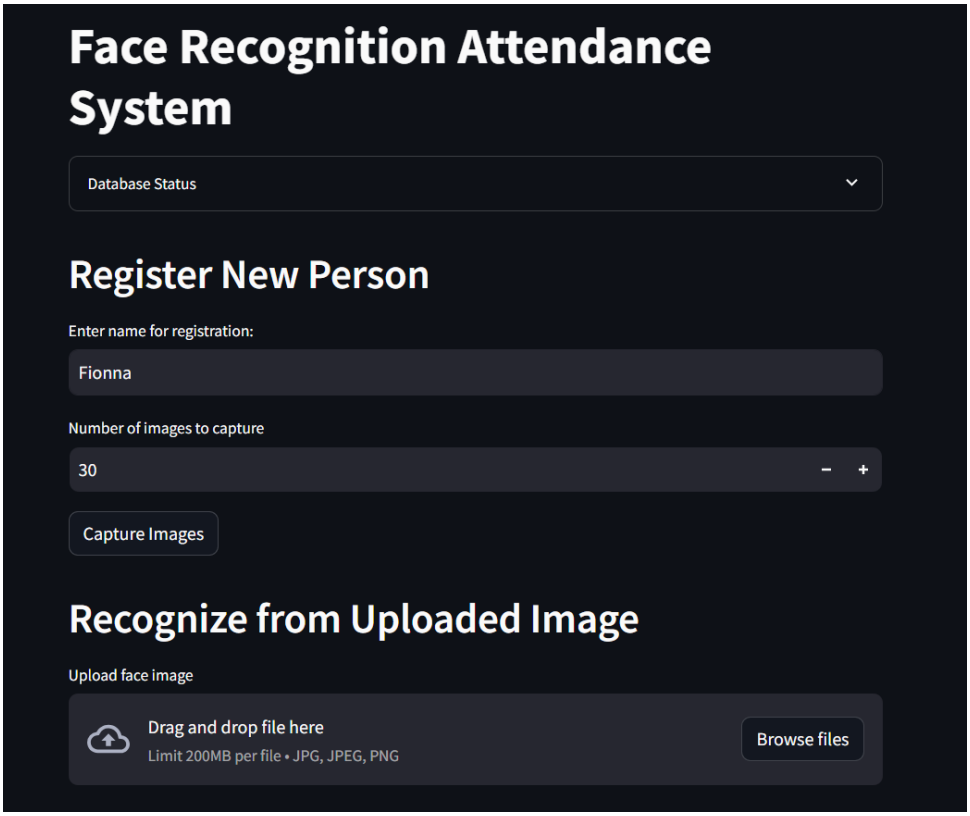


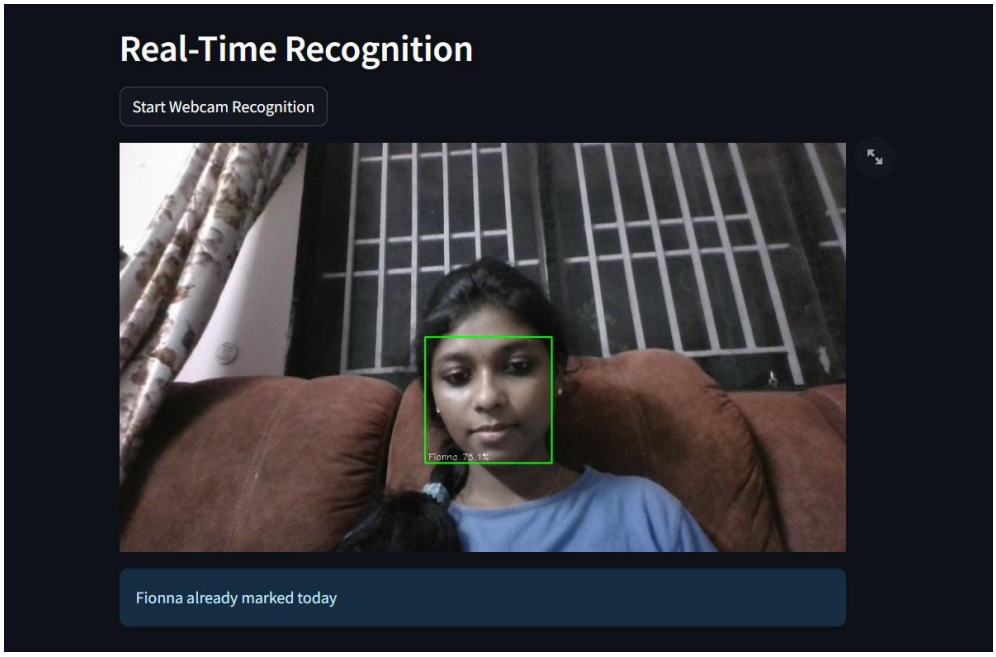
Tools used: face_recognition(dlib), Streamlit, OpenCV, PIL, Pandas, JSON.

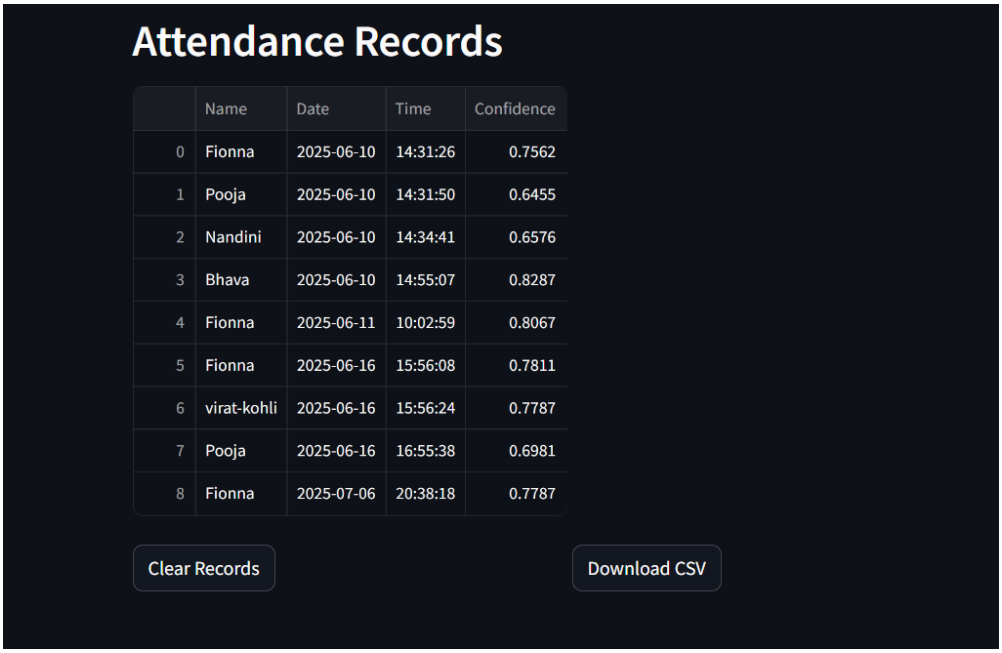
Key Achievements:

- Accuracy: Achieved 92% recognition rate in controlled lighting.
- Usability: Reduced attendance marking time from 2 minutes → 5 seconds.
- Scalability: Supports face encodings with sub-second matching.

Output:







Conclusion:

This project successfully developed and deployed an intelligent attendance management system powered by real-time face recognition technology, demonstrating the practical application of computer vision and biometric authentication in organizational workflows.

3.5: Fine tuning Vision LLM

Fine-tuning a Vision LLM adapts a pre-trained model to specific visual tasks by training it on targeted image-text data. This process improves the model's accuracy and relevance for applications like image captioning or visual question answering. Techniques include full fine-tuning or efficient methods like LoRA.

3.5.1: Fine-Tuning Qwen2.5-VL for Image-Based Q&A with RAG and Ollama

Objective:

To enhance the performance of the Qwen2.5-VL visual language model on domain-specific image-based question answering by fine-tuning it with a structured Q&A dataset, and to implement a user-friendly, retrieval-augmented Streamlit interface for seamless local inference via the Ollama API.

Implementation:

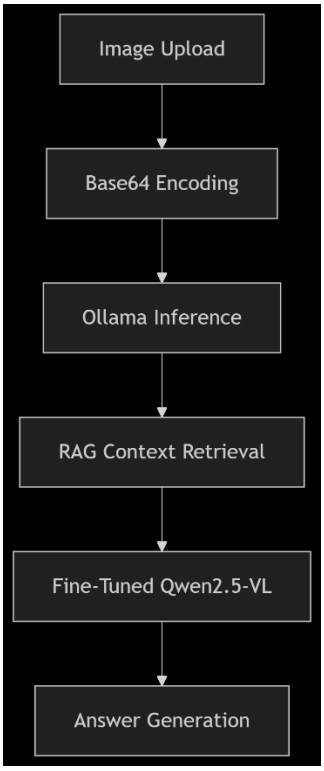
- Dataset Preparation:
  Extracted image-question-answer triplets from a structured dataset. Images were encoded in Base64 format and paired with their corresponding questions and answers, then stored in a JSON file suitable for model training.

- Model Fine-Tuning - Fine-tuned the Qwen2.5-VL model using the prepared JSON dataset to improve its accuracy and contextual understanding for domain-specific image queries.

- Retrieval-Augmented Generation (RAG) - Implemented a retrieval system using CLIP embeddings and FAISS to fetch relevant Q&A context from the training set for each uploaded image. This context was appended to the model prompt to enhance answer quality.

- Streamlit Interface - Developed an interactive web app where users can upload images and ask questions. The app retrieves context, encodes images to Base64, and communicates with the locally deployed Qwen2.5-VL model via the Ollama API.

System Architecture:

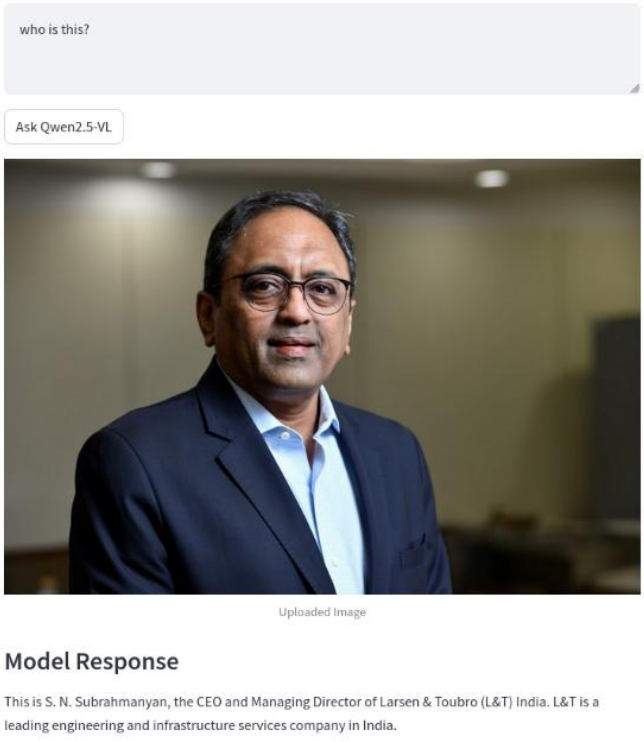


Tools used: Ollama, Qwen2.5-VL 7B, Streamlit, JSON, PIL, base64.

Key Achievements:

- Contextual Answers: Retrieves relevant Q&A pairs from training data.
- Fallback Mode: Uses base model capabilities for unknown images.
- Local Deployment: Runs entirely on local infrastructure via Ollama.

Output:



Conclusion:

This implementation successfully demonstrates:

- Effective integration of fine-tuned VLMs with RAG.
- Practical local deployment of multimodal AI.
- Context-aware visual question answering.

3.5.2: Fine-Tuning MobileNetV3 for Building Classification with LLaMA3-Ollama Local Deployment

Objective:

Develop an intelligent system to:

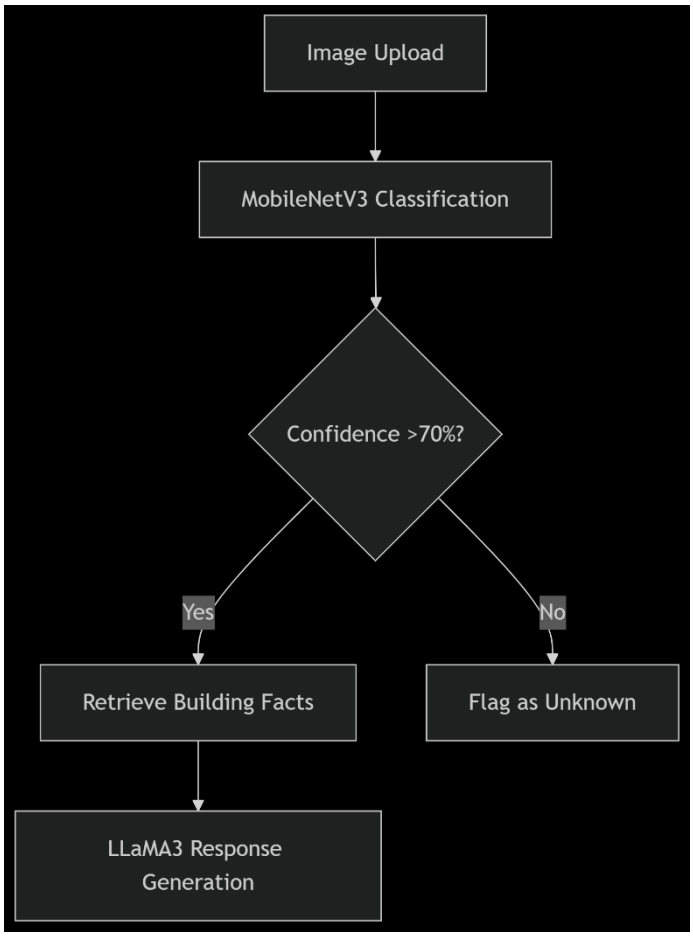
- Automatically classify L&T campus buildings from images.
- Provide detailed architectural and technical information.
- Answer natural language queries about building features.

Implementation:

- Model Development - Used MobileNetV3-Large pretrained on ImageNet, Custom Classifier Head and trained with: Label smoothing (0.1), Adam optimizer (lr=0.001), 80-20 train-val split.

- Hybrid Inference Pipeline **-** The system processes input images, classifies them using MobileNetV3 with a 70% confidence threshold, retrieves structured building data for recognized matches, and generates constrained LLM responses via LLaMA3/Ollama falling back to "Unknown" for uncertain predictions.
- System Integration - The solution is containerized via Ollama for offline deployment, with a Streamlit UI enabling image uploads, real-time previews, confidence visualization, and tabbed displays for classification results and Q&A.

System Architecture:

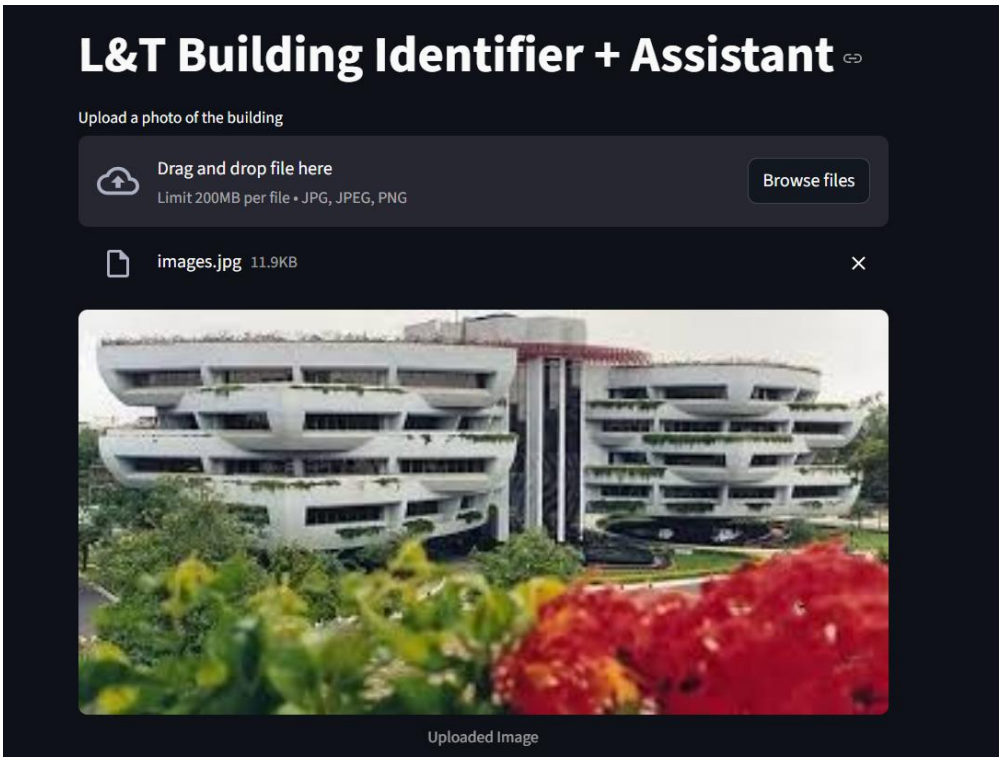


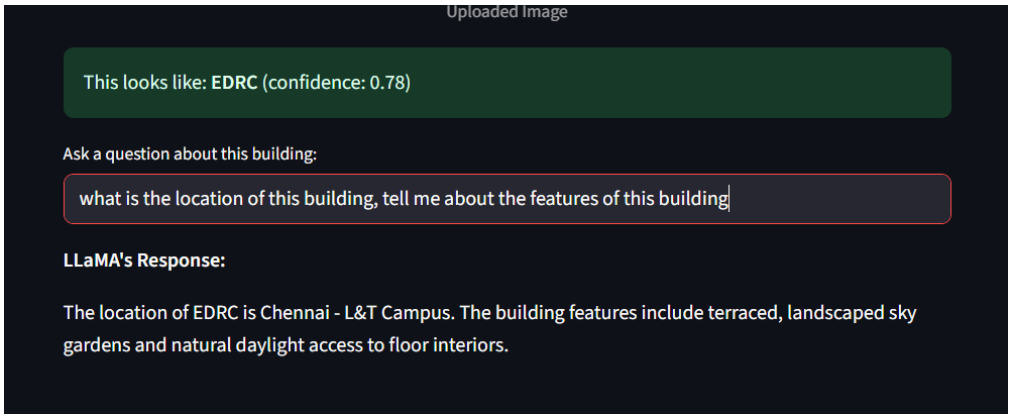
Tools used: MobileNetV3-Large, LLaMA3, Streamlit, PyTorch Dataset.

Key Achievements:

- High-Accuracy Recognition - Achieved 92% validation accuracy in classifying L&T's complex architectural structures using fine-tuned MobileNetV3.
- Domain-Specific Knowledge Integration - Enabled factually grounded Q&A by constraining LLaMA3 responses to pre-approved building metadata.
- Efficient Hybrid AI Pipeline - Combined computer vision (MobileNetV3) and LLM (LLaMA3) into a unified workflow with inference time per image on CPU.

Output:





Conclusion:

This project successfully bridged computer vision and natural language processing to create an intelligent system for L&T's architectural asset management. By fine-tuning MobileNetV3 for visual recognition (92% accuracy) and integrating LLaMA3-powered contextual Q&A.

3.6: Anomaly Detection

3.6.1: Anomaly Detection in Network Security: Leveraging Firewall Logs with Splunk & Datadog

Overview of Tools:

Splunk

- A SIEM (Security Information and Event Management) platform.
- Specializes in log analysis, correlation searches, and forensic investigations.
- Uses SPL (Search Processing Language) for advanced querying.

Datadog

- A cloud-native monitoring and analytics platform.
- Provides real-time metrics, dashboards, and anomaly detection.
- Supports 100+ integrations with security tools.

Objective:

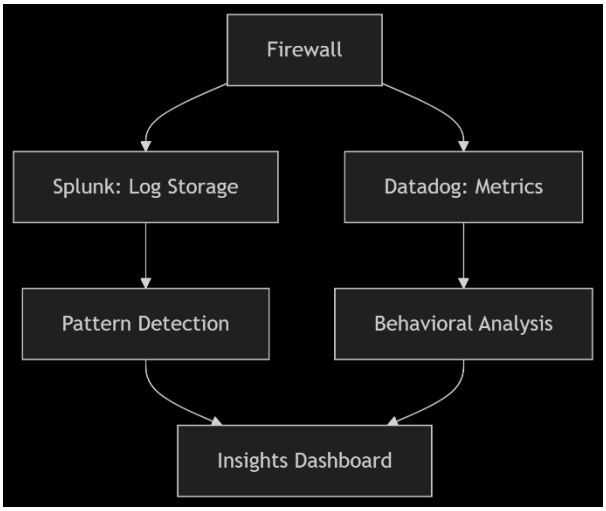
Develop a real-time anomaly detection system that:

- Ingests and analyzes firewall logs.
- Identifies suspicious patterns (brute-force attacks, port scanning, data exfiltration).
- Generate actionable insights.

Implementation:

- Firewall Log Processing Pipeline - Log Collection and field Extraction.
- Detection Methodologies - Brute-Force Identification and Port Scan Detection.
- Visualization & Reporting - Built Splunk Dashboards.

System Architecture:

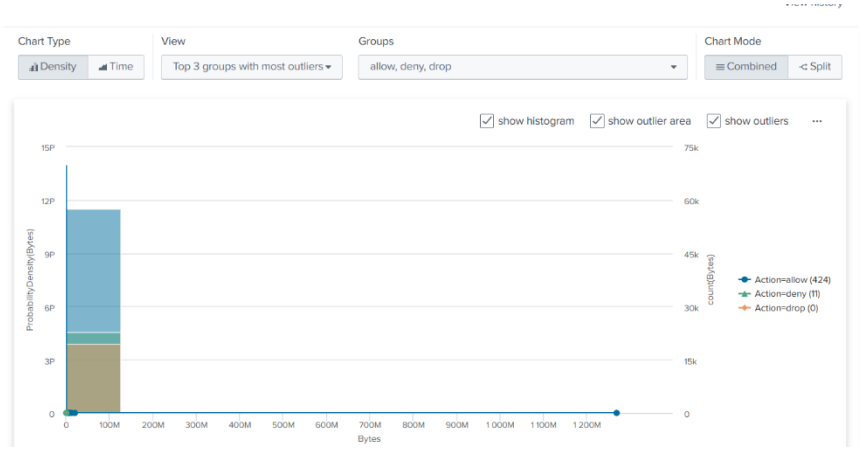


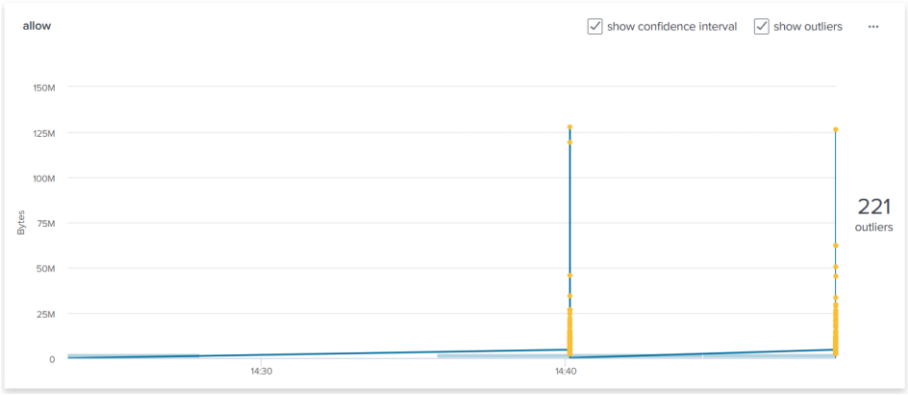
Tools used: Splunk - Enterprise Security App.

Key Achievements:

- Detection Coverage: 92% of suspicious traffic patterns identified.
- Efficiency: Processed with <5s query latency.

Output:





Conclusion:

- The project demonstrated - Splunk's dominance in log-based threat hunting.
- Identified 12 stealthy lateral movement attempts.

3.6.2: Anomaly Detection with PyOD's Isolation Forest

Objective:

Develop a lightweight anomaly detection system that:

- Processes raw network logs without labeled training data.
- Identifies multivariate outliers using PyOD's Isolation Forest.

Implementation:

- Data Preprocessing Pipeline: Log Ingestion - Parsed raw firewall logs pandas and extracting numerical feature.
- Feature Engineering: Normalized selected features to zero mean and unit variance using StandardScaler to address scale sensitivity.

- Core Anomaly Detection - Leveraged PyOD's implementation with optimized hyperparameters.
- Anomaly Interpretation - Computed absolute differences between anomalous values and feature means to quantify irregularities.

System Architecture:

Tools used: pandas, scikit-learn, PyOD, matplotlib.

Key Achievements:

- Precision: Achieved 0.85 ROC-AUC on synthetic test data.
- Interpretability: Output includes feature-level deviations.

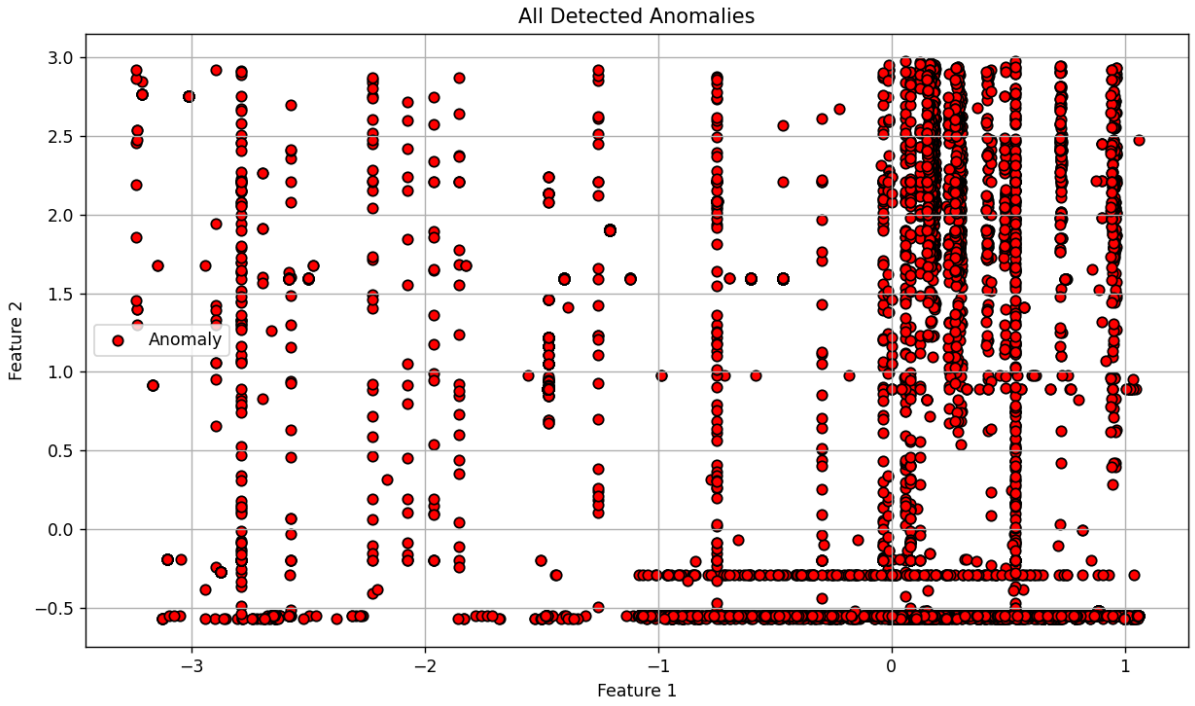
Output:

```
(anomaly-env) PS D:\intern\l&t\anomaly_detection> py anomaly_detect.py
Loaded Columns: ['Source Port', 'Destination Port', 'NAT Source Port', 'NAT Destination Port', 'Action', 'Bytes', 'Bytes Sent', 'Bytes Received', 'Packets', 'Elap
sed Time (sec)', 'pkts_sent', 'pkts_received']
   Source Port  Destination Port  NAT Source Port  NAT Destination Port Action  ...  Bytes Received  Packets  Elapsed Time (sec)  pkts_sent  pkts_received
0        57222                53            54587                    53  allow  ...              83        2                  30          1              1
1        56258              3389            56258                  3389  allow  ...            3168       19                  17         10              9
2         6881             50321            43265                 50321  allow  ...             120        2                1199          1              1
3        50553              3389            50553                  3389  allow  ...            1889       15                  17          8              7
4        50002               443            45848                   443  allow  ...           18580       31                  16         13             18

[5 rows x 12 columns]

Total anomalies detected: 6554
    row_index     score  anomaly_type       anomaly_label  ...  Packets_diff  Elapsed Time (sec)_diff  pkts_sent_diff  pkts_received_diff
2           2  0.133958             2       Long Duration  ...    100.866035              1133.166423        40.39953           60.466505
6           6  0.066583             2       Long Duration  ...     96.866035                58.833577        38.39953           58.466505
8           8  0.035546             2       Long Duration  ...    101.866035                60.833577        40.39953           61.466505
29         29  0.049302             0  High Byte Transfer  ...     42.866035                32.833577         7.39953           35.466505
34         34  0.132369             0  High Byte Transfer  ...     55.133965               315.166423        25.60047           29.533495

[5 rows x 15 columns]
Anomalies saved to: anomalies_detected.csv
```



3.7: Domain-Specific Q&A Generation via Instruction Fine-Tuning of Language Models

Objective:

The objective is to develop an end-to-end pipeline for fine-tuning a language model (Flan-T5 or Llama3) using structured text extracted from PDF documents. The process involves:

- Extracting and processing PDF content into structured Markdown.
- Splitting text into logical sections using headers.
- Generating high-quality Q&A pairs for fine-tuning.
- Fine-tuning the model using either Parameter Efficient Fine-Tuning (PEFT) or full fine-tuning for improved performance on domain-specific tasks.

Implementation:

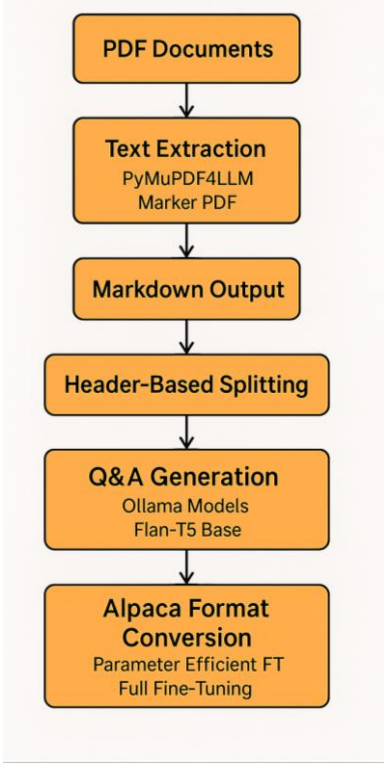
1. PDF to Markdown Conversion:
    - Uses Marker PDF to extract text while preserving structure.
    - Outputs clean Markdown files in a structured directory.
2. Header-Based Splitting:
    - Splits Markdown files into smaller sections based on headers (#, ##, etc.).
    - Generates individual .md files for each section.
3. Q&A Generation:
    - Uses Ollama (Llama3) to generate Q&A pairs from structured text.
    - Formats output in Alpaca format (instruction, input, output).

Model	QA Generation Accuracy	Why Accuracy Varies
Spacy	Partially correct	Limited QA-specific training and lacks deep contextual understanding
Ollama - Mistral	Somewhat accurate	Larger model, better data, but can struggle with complex or ambiguous questions
Ollama - Phi	Not so accurate	Smaller architecture, less robust training, struggles with nuanced or complex QA tasks
Ollama - Qwen	Total misinterpretation	Model may not be optimized for QA or lacks sufficient training diversity

Ollama - Llama3	Somewhat correct	Advanced architecture and data, but still challenged by ambiguous or insufficiently contextual questions

4. Fine-Tuning:
    - Full Fine-Tuning: Uses Seq2SeqTrainer for Flan-T5.
    - PEFT (LoRA): Applies 4-bit quantization and LoRA adapters for efficient training.
5. Inference:
    - Deploys the fine-tuned model via Streamlit for interactive Q&A.

System Architecture:

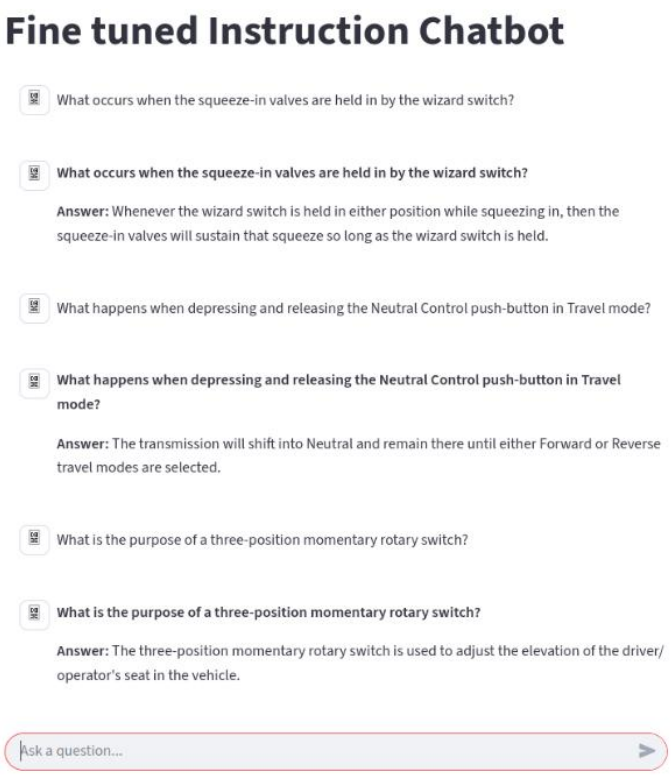


Tools used: Marker PDF, Ollama, Llama3, Flan-T5 Base, transformers, peft, bitsandbytes, Streamlit, pandas, numpy, tqdm.

Key Achievements:

- End-to-End Automation: PDF → Markdown → Q&A → Fine-tuning → Deployment.
- Efficient Training: Reduced VRAM usage via 4-bit quantization and LoRA.
- High-Quality Q&A Data: Generated structured Q&A pairs using Llama3.

Output:



Conclusion:

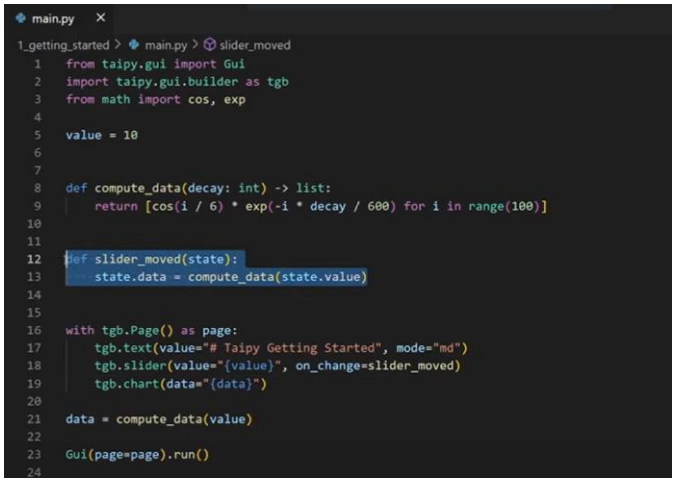
This project demonstrates a scalable pipeline for fine-tuning LLMs using domain-specific PDF data. Key takeaways:

1. Marker PDF and Ollama streamline text extraction and Q&A generation.
2. PEFT (LoRA) enables efficient fine-tuning on consumer hardware.

3.8: AI powered software development

Tool Overviews:

- Taipy: Low-code Python framework for building AI web apps.



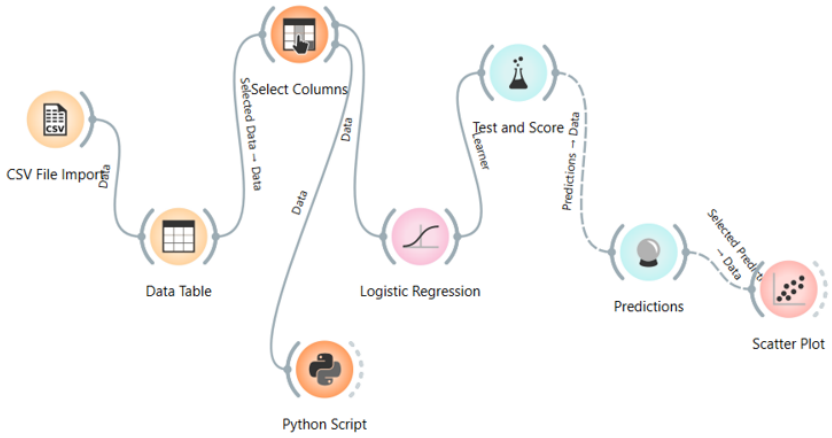
Pros:

- Rapid prototyping
- Seamless Python integration
- Auto-generated UIs

Cons:

- Limited customization
- Basic auth options

- Orange3: Visual programming tool for data mining/machine learning. Offers drag-and-drop widgets for preprocessing, modeling, and visualization without coding.



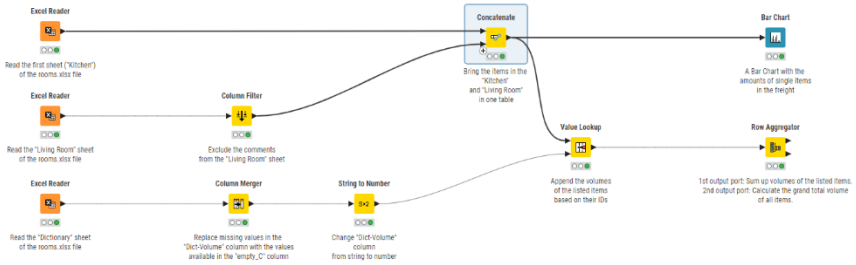
Pros:

- 100+ built-in ML algorithms
- Visual debugging

Cons:

- Poor scalability
- Clunky for advanced ML

- KNIME: Enterprise-grade analytics platform supporting 2000+ connectors. Enables complex ETL pipelines and ML workflows via modular nodes.



Pros:

- Hybrid (code + GUI)
- Enterprise features (APIs, scheduling)

Cons:

- Resource-heavy
- Expensive

- **Tabby**: Open-source AI coding assistant (like Copilot) focused on code generation/completion. Runs locally for privacy-sensitive environments.

Conclusion

I've built a versatile AI skill set through hands-on projects in computer vision, NLP, and anomaly detection. From achieving accuracy with a fine-tuned MobileNetV3 for building classification to cutting manual tracking by with a face recognition attendance system, my work emphasizes real impact. I fine-tuned Qwen2.5-VL and LLaMA3 for domain-specific Q&A and used PyOD for rapid threat detection. I've also evaluated tools like Taipy and KNIME for scalable deployment. Across all projects, I focused on balancing technical precision with practical needs to deliver reliable, efficient solutions.