Overview

Build a small-scale but production-minded Real Estate Search Engine focused on residential properties. The system must:

- Train a computer-vision model to parse floorplan images and extract structured attributes: number of rooms, halls, kitchens, bathrooms (and optionally room sizes and labels).
- Ingest an Excel sheet containing property metadata (floorplan image links, textual descriptions, location, price, listing date, and links to certifications / inspection reports).
- Store data in two databases: a structured relational database (for fast queries & transactions) and an unstructured/searchable store (for text, embeddings, and full-text search).
- Build a multi-agent chatbot UI (Streamlit) backed by FastAPI that can answer user queries about properties and perform tasks using specialized agents: query routing, task decomposition & planning, structured-data agent, RAG agent, web research agent, report generation agent, renovation estimation agent.
- Implement memory to retain key user conversation context across sessions.
- Provide deployment via Docker, configuration via `.env`, and a Streamlit UI.
- Deliverables: source code, a PPT (System Architecture/Agent Architecture, Key Decisions, Challenges, Future Improvements), and a video walkthrough.
- Implementation of guardrails and observability is a bonus.

Dataset & Inputs (Provided by Instructor)

Click here for the dataset link. In this link you will find:

1. **Excel file** (CSV/XLSX) with rows for *residential properties* containing various details.
2. **Floorplan images** (URLs or archived images).
3. **Optional**: additional PDFs (inspection reports), web pages (certificates).

Phase 1 — Floorplan Model (Mandatory)

Goal

Train a model that takes a floorplan image and outputs a structured JSON with at least:

```
{
"rooms": 3,
"halls": 1,
"kitchens": 1,
"bathrooms": 2,
"rooms_detail": [{"label":"Bedroom","count":2, "approx_area":null}, ...]
}
```

Deliverables for this phase

- Training code & notebook, model weights, inference script `parse_floorplan(image) -> JSON`.
- Short README describing dataset split and metrics (accuracy for counts, IoU for segmentation).

Phase 2 — Data Ingestion & Storage

Storage design

- **Structured DB (Relational)**: PostgreSQL (or MySQL)
- **Unstructured / Vector store**: Elasticsearch, Qdrant or any Vector Store.
    - Index textual fields: `title`, `long_description`, `inspection reports`.
    - Store embeddings for RAG.

Ingestion steps

1. Read Excel file.
2. Download floorplan images and run `parse_floorplan()` to extract structured attributes.
3. Save canonical row data + parsed JSON into PostgreSQL.
4. Index textual content + metadata and embeddings in vector DB.
5. For each external link to reports/certs, fetch (or store link) and extract text (PDF parsing) to include in unstructured index.

Deliverables

- ETL script or FastAPI endpoint that ingests the Excel file and performs steps above.

Phase 3 — Agents & Chatbot Architecture

High-level agent list (minimum)

1. **Query Router**
    - Purpose: Decide which agent to invoke based on user input (intent detection + slot extraction).
2. **Planner / Task Decomposer**
    - Purpose: Break complex queries into ordered tasks (e.g., "Find 2BHK in X, estimate renovation cost, generate summary PDF")
3. **Structured Data Agent**
    - Purpose: Run SQL queries against PostgreSQL to fetch property records, filter, aggregate.
4. **RAG Agent (Unstructured Search Agent)**
    - Purpose: Retrieve and synthesize info from indexed docs (Vector DB) for long-form answers, citations.
    - **Bonus:** Generate valid citations for responses
5. **Web Research Agent**
    - Purpose: Fetch external live data (market rates, neighborhood info).
    - Can leverage services like tavily.
6. **Report Generation Agent**
    - Purpose: Generate detailed reports and graphs if requested by the user.
    - **Bonus:** Reports downloadable in pdf format.
7. **Renovation Estimation Agent**
    - Purpose: Given property size / rooms, estimate renovation costs.
8. **Memory Component**
    - Purpose: Persist user preferences across sessions (e.g., preferred location, budget, saved properties).
    - **Bonus:** Demonstrate multiple types of memory across chat sessions

---

Phase 4 — UI & Backend

Backend (FastAPI)

- Endpoints:
  - `/ingest` — upload Excel, trigger ETL.
  - `/parse-floorplan` — debug endpoint for single image.
  - `/chat` — websocket or REST for chatbot messages.

Frontend (Streamlit)

- Pages/components:
  - Simple interface to ingest Excel file for properties.
  - Page to upload a single floorplan image and retrieve results.
  - Chat page to interact with the chatbot.

Deployment & Config

- Dockerfile for backend and frontend (can be a single image or two images in docker-compose).
- `.env` to store secrets (DB connection, API keys, embedding model keys). Include `.env.example` for submission.

Student Deliverables Checklist (What to submit)

1. Private Git repo with source code + instructions & detailed README.
2. Trained model artifacts & inference script for floorplan parsing.
3. ETL scripts and DB schema.
4. FastAPI backend and Streamlit frontend code.
5. Docker Compose for deploying backend and db(s) and `.env.example`.
6. PPT containing: System Architecture/Agent Architecture, Key Decisions, Challenges, Future Improvements, metrics and experiments for floorplan model..
7. 5–7 minute video walkthrough explaining architecture and demoing features.

Example GitHub Repository Structure:

- `README.md` (Instructions, approach, and explanation)
- `src/` (Source code) and `data/` (Data files, if applicable)
- `requirements.txt` (List of dependencies)

Additional Information:

- Candidates may use **Jupyter Notebook** or **Google Colab** for model development
- The model must be trained using PyTorch or Tensorflow
- Version control with Git for the entire repository should be followed

Submission Deadline: 12:00 PM ON Nov. 09, 2025

Resources: IITGN AI/ML 2 Resources