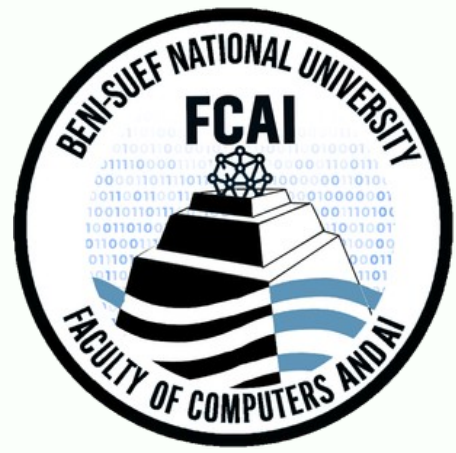
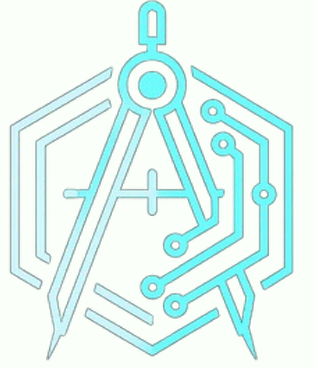




Beni-Suef National University

Faculty of Computers and Artificial Intelligence



CadArena

Integrating LLMs for an Enhanced Engineering Design Workflow

Supervisor: Dr. Amira Mohey El Din Mohamed

Presented by:

Youssef Taha • Maher Ibrahim • Omnia Ayman • Samar Nashaat

Academic Year: 2025 / 2026

OUR TEAM

Meet the Team

Youssef Taha Mohamed Badawi

Backend and System Architect

Maher Ibrahim El-Shahat

RAG Developer

Omnia Ayman Abdel-Badi

Frontend Developer

Samar Nashaat

UI / UX Developer

Dr. Amira Mohey El Din Mohamed

Academic Supervisor

Presentation Agenda

1 Introduction & Problem

What CadArena is and why it was built

2 Why Not ML?

The challenges we faced before choosing our approach

3 LLM Experiments

Zero-shot, few-shot, and why direct generation failed

4 System Architecture

How the platform is structured

5 CadStudio Core

AI-powered floor plan generation pipeline

6 DXF Engine

Converting layouts to professional CAD files

7 ArchChat & RAG

Intelligent engineering knowledge assistant

8 Frontend & UI/UX

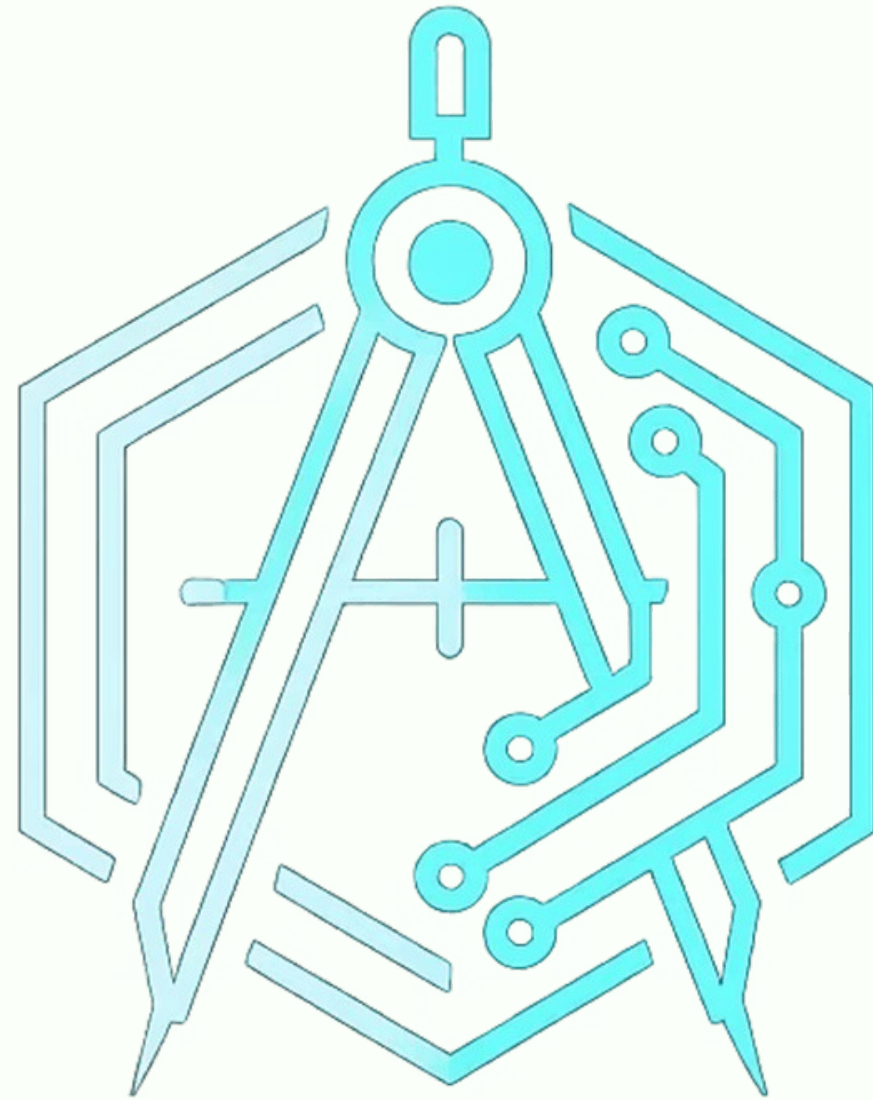
Design Studio, Community, and user experience

8 Testing & Evaluation

Accuracy metrics and performance benchmarks

10 Conclusion & Q&A

Achievements and future directions



INTRODUCTION

What is CadArena?

CadArena is an AI-powered architectural design platform that transforms natural language descriptions into professional 2D residential floor plans — validated against architectural rules and exported as DXF files compatible with AutoCAD and Revit.



NATURAL LANGUAGE → FLOOR PLAN



ARCHITECTURAL VALIDATION

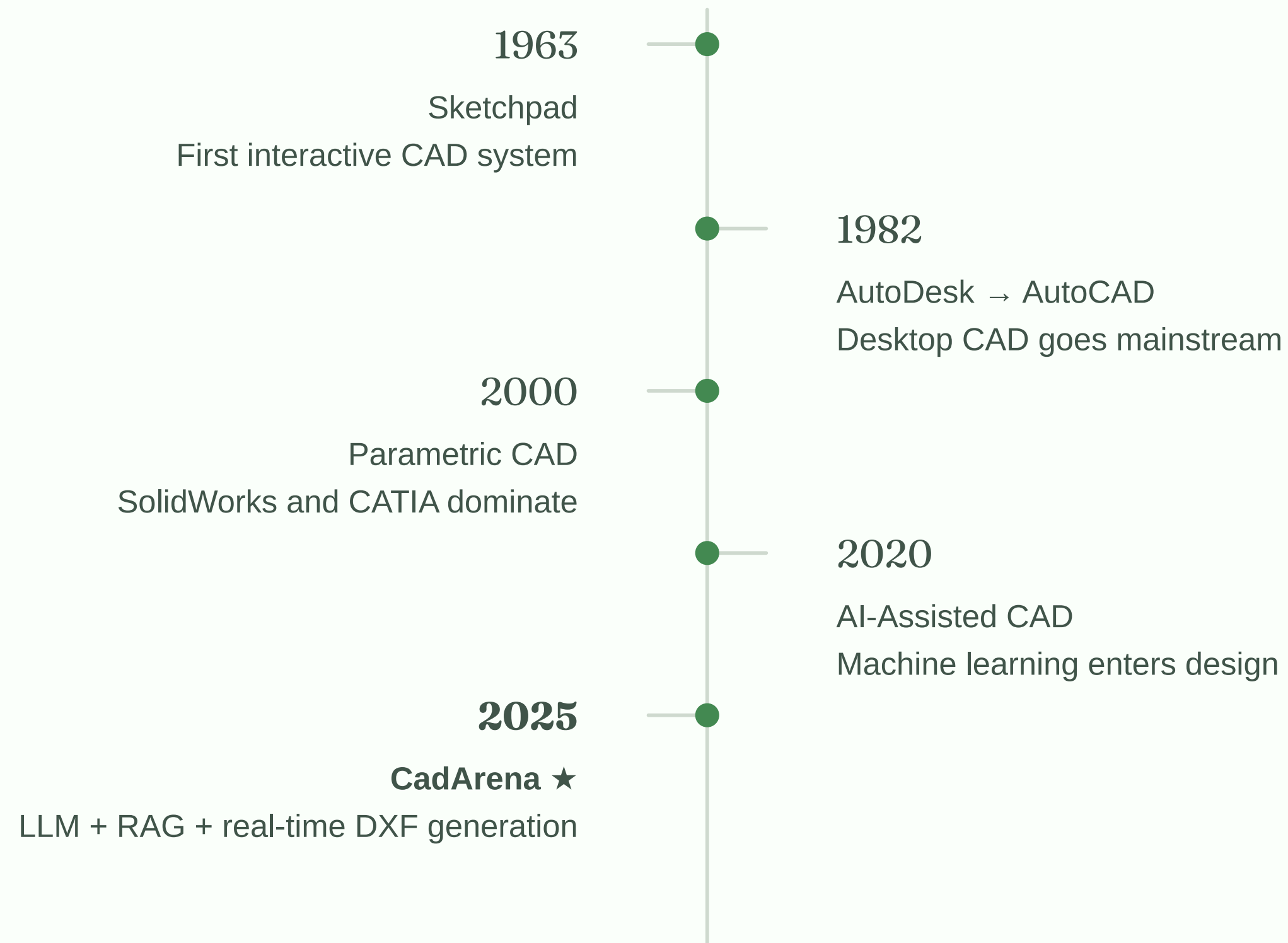


DXF EXPORT



RAG KNOWLEDGE ASSISTANT

Evolution of CAD Technology



Why Traditional CAD Workflows Fall Short?



Architects still translate ideas into drawings manually — this is slow, error-prone, and inaccessible.

1

Manual and Time-Intensive

Even simple floor plans require considerable manual time and repeated adjustments.

2

No Natural Language Interface

CAD tools require specialized commands, not plain descriptions.

3

Iterative Revisions are Costly

Each change requires manual re-drafting from scratch.

4

Validation is Separate

Compliance with design rules is checked after the design is done, not during.

5

No Integrated Knowledge Assistance

Engineers must switch between CAD tools and external references constantly.

How We Approached the Problem?

Define the Problem

Identify pain points in architectural CAD workflows

1

Face Real Challenges

Data scarcity, DXF complexity, labeling costs

2

Explore Solutions

Evaluate Machine Learning, rule-based systems, and LLMs

3

Build CadArena

LLM + RAG + rule-based DXF engine — the right combination

4

Why We Did Not Use Pure Machine Learning?

1 — No Labeled Dataset Available

No existing dataset of (prompt → validated floor plan) pairs for Arabic/Egyptian residential architecture exists. Creating one would require hundreds of expert architect hours.

2 — DXF Files Are Structurally Complex

DXF documents contain strict hierarchical entity structures, coordinate systems, and layer organization. Sequence models cannot guarantee geometrically valid outputs without massive domain-specific training data.

3 — High Labeling Cost

Annotating architectural drawings with room types, dimensions, adjacency rules, and building code compliance requires specialized expertise — not feasible at the scale needed for supervised ML.

Pure ML Requires:

- ✗ Thousands of labeled examples
- ✗ Domain-expert annotators
- ✗ Months of GPU training time
- ✗ Expensive infrastructure

→ We chose a Hybrid Architecture:

LLM for language understanding

+

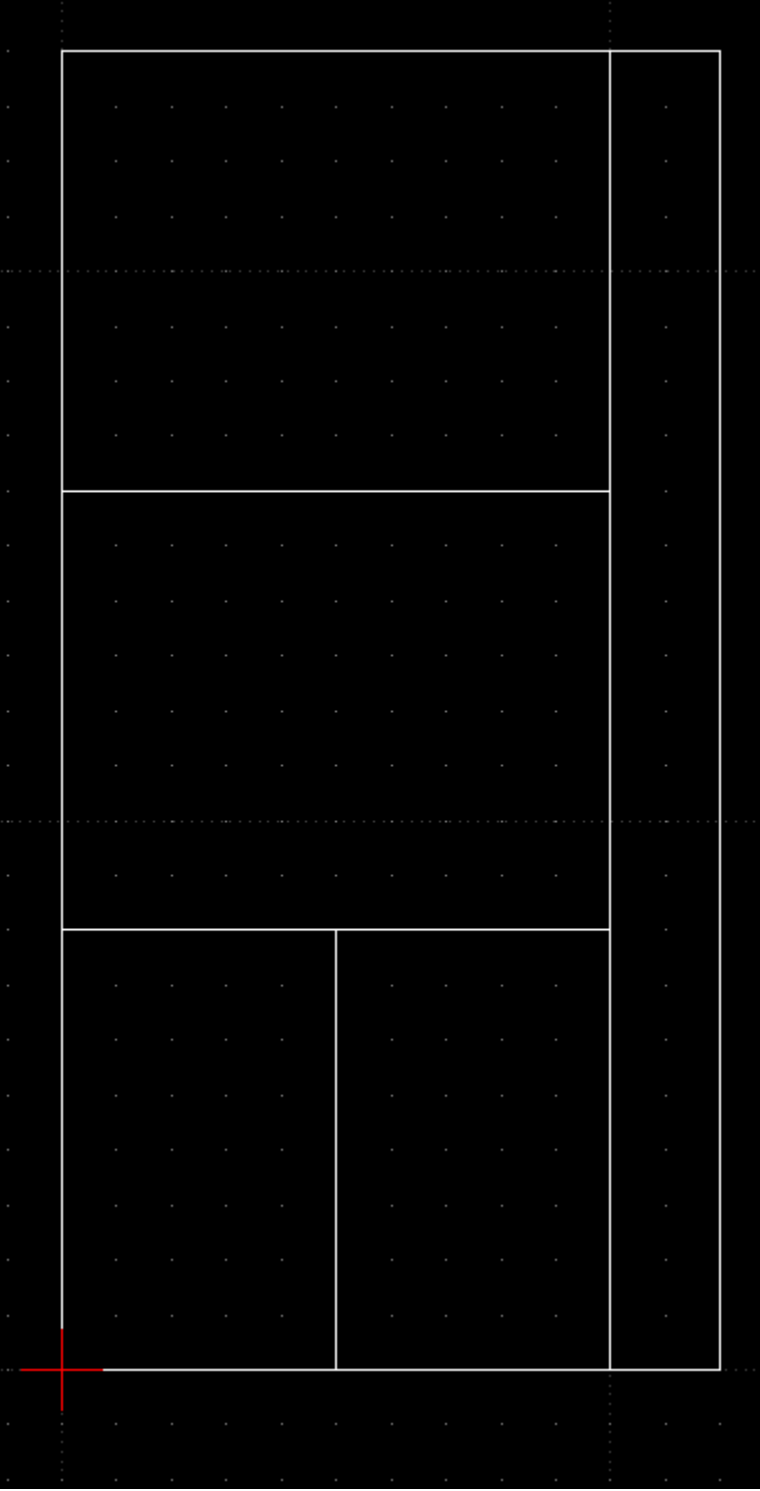
Deterministic engine for geometry

What Happened When We Tried Direct LLM Generation?

We tested both frontier and open-source models with the same architectural prompt before finalizing our architecture.

Create a DXF floor plan following the Egyptian Building Code (EBC) for a 12m × 24m residential building with two bedrooms, one kitchen, one bathroom, and a central corridor.

Model	Strategy	DXF Valid?	Geometry Correct?	EBC Compliant?	Verdict
ChatGPT (GPT-4)	Zero-shot	Partial	Poor	✗ No	✗ Rejected
ChatGPT (GPT-4)	Few-shot	Better	Inconsistent	✗ No	✗ Unreliable
Google Gemini	Zero-shot	Partial	Poor	✗ No	✗ Rejected
Google Gemini	Few-shot	Better	Inconsistent	✗ No	✗ Unreliable
LLaMA 3 (local)	Zero-shot	Rare	Very Poor	✗ No	✗ Failed
Mistral 7B (local)	Few-shot	Occasional	Poor	✗ No	✗ Failed
CadArena (Hybrid)	LLM + Rules	Always ✓	Correct ✓	✓ Yes	✓ Adopted



Specific Failures in Direct LLM-Based Design Generation

1

Hallucinated Coordinates

Models generated geometrically impossible room positions — rooms overlapping, walls misaligned, coordinates outside the building boundary.

2

Invalid DXF Structure

Output frequently broke DXF file format standards — missing section headers, incorrect entity ordering, or corrupt layer definitions that CAD software could not open.

3

No EBC Knowledge

Models had zero built-in knowledge of the Egyptian Building Code (EBC 2023) minimum room areas, corridor widths, or forbidden adjacencies.

4

Non-Deterministic Output

The same prompt produced different — sometimes completely incompatible — layouts on repeated runs due to the stochastic nature of LLMs.



Our Solution: Hybrid Architecture

LLM handles ONLY:

- Natural language understanding
- Parameter extraction

Deterministic engine handles:

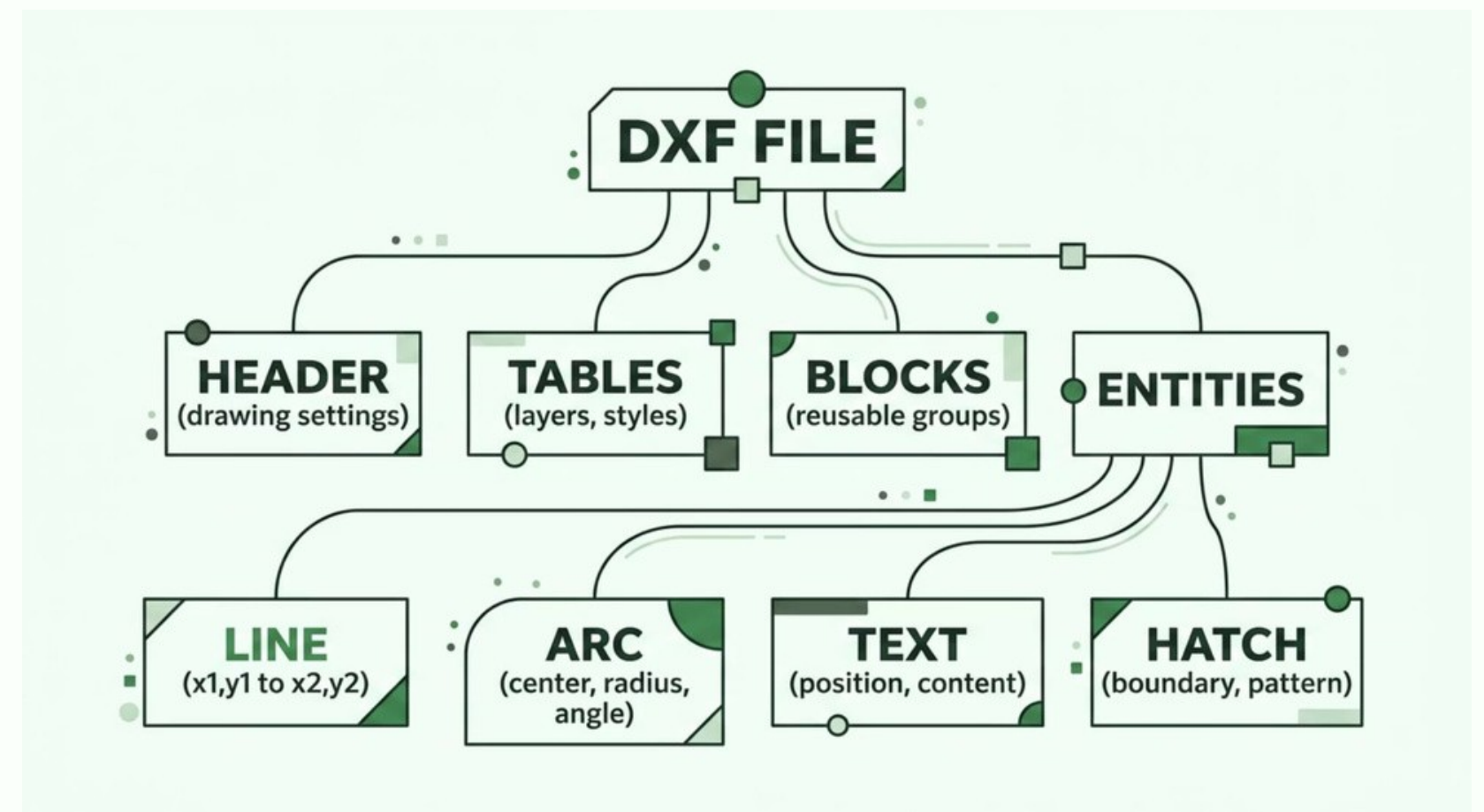
- Layout geometry generation
- Architectural rule enforcement
- EBC 2023 validation
- DXF rendering

Result: 100% structurally valid DXF output

What is a DXF File?

DXF (Drawing Exchange Format) is the universal interchange format for CAD data, created by Autodesk in 1982. It stores technical drawings as structured text — listing every geometric entity (lines, arcs, circles, text) with its exact coordinates, layer, and properties.

DXF files are readable by AutoCAD, Revit, LibreCAD, FreeCAD, and virtually every professional CAD application.



OPEN STANDARD

AUTOCAD COMPATIBLE

USED IN CADARENA

Why DXF Generation is Challenging for AI?

1

Strict Syntax Rules

A single malformed entity invalidates the entire file. No partial credit — either fully valid or completely broken.

2

Geometric Precision

Coordinates must be numerically exact. A wall that is 0.5mm off will create overlapping rooms or gaps in the floor plan.

3

Complex Relationships

Rooms must connect correctly.
Doors must align to walls.
Dimensions must reference real geometry. LLMs cannot enforce these constraints alone.

✓ CadArena separates concerns: LLM understands language → rules engine generates geometry → validator checks DXF integrity

CadArena vs. Existing Systems

System	LLM Integration	DXF Output	Real-Time	EBC Validation	Local Deploy	Limitation
AutoCAD AI	Partial (plugin)	Partial	No	No	No	Requires license + cloud
ChatCAD	Cloud LLM	Limited	Partial	No	No	Limited DXF, no validation
DeepCAD	Partial	No	No	No	Partial	Geometry only, no NL interface
Text2CAD	Cloud LLM	Basic	No	No	No	Batch only, no refinement
SketchGen	Cloud LLM	SVG only	Yes	No	No	Not engineering-grade
Rayon	Cloud LLM	DXF supported	Yes	No	No	No EBC, cloud-only
CadArena	Integrated + RAG	Comprehensive ✓	Yes (WebSocket) ✓	Yes (EBC 2023) ✓	Yes (Docker) ✓	2D residential scope

The Research Gap We Fill

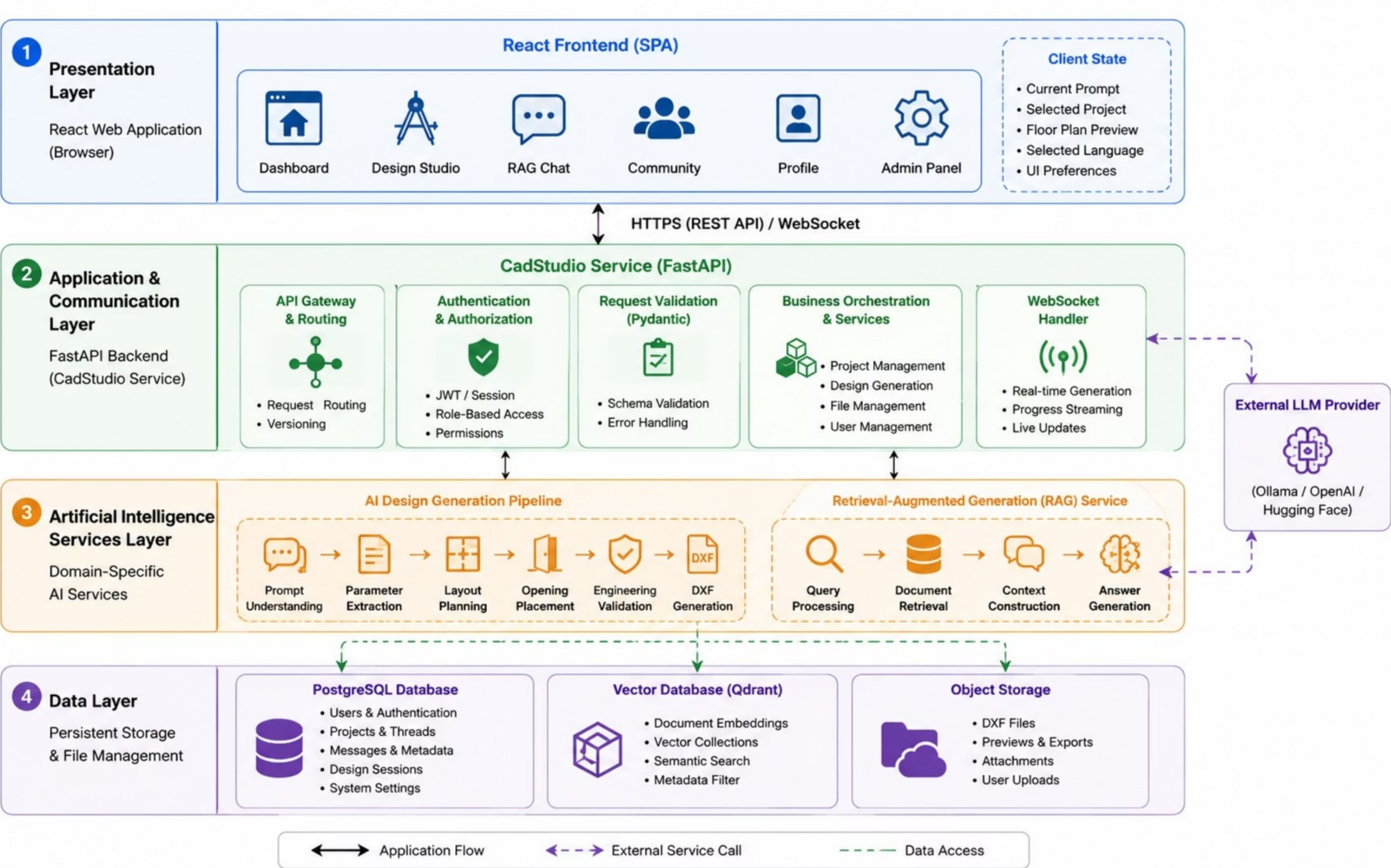
What Was Missing

- 1** — No End-to-End NL → DXF in real time
No system converts natural language to DXF in real-time.
- 2** — No iterative prompt-based refinement
Existing tools don't allow prompt-based design iteration.
- 3** — No automatic EBC 2023 validation
Building code checks are entirely manual, error-prone, and slow.
- 4** — No Arabic language CAD platform
There is no CAD solution supporting the Arabic language.
- 5** — No open-source local LLM CAD tool

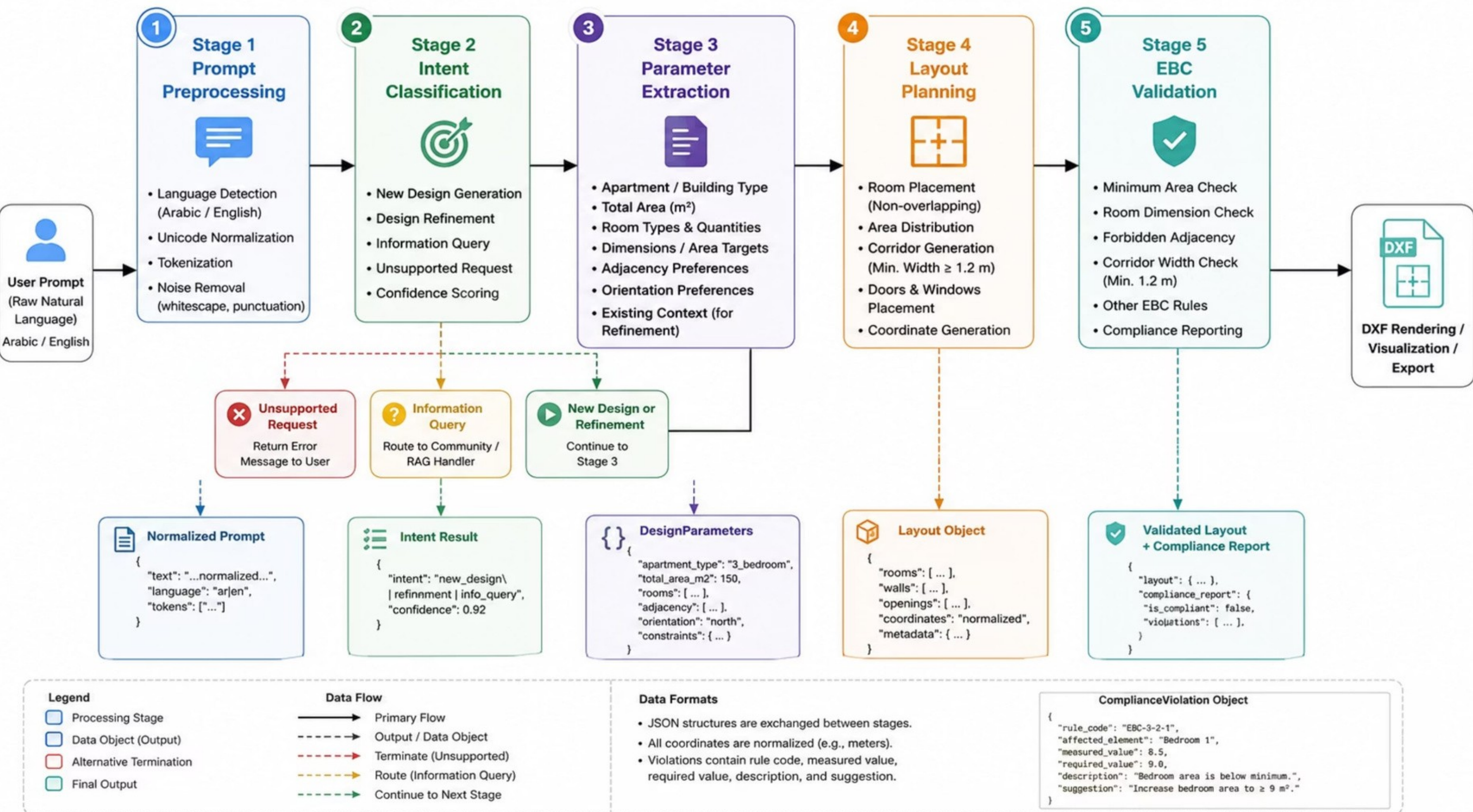
What CadArena Delivers

- 1** — Prompt → DXF in Under 5 Seconds
Real-time natural language to DXF conversion.
- 2** — Full iterative chat-based refinement
Design modifications seamlessly via chat interface.
- 3** — Automatic EBC 2023 Validation
Built-in, instant compliance checks against local codes.
- 4** — Bilingual Support
Comprehensive support for both Arabic and English input.
- 5** — Local deployment — no cloud dependency

CadArena – High-Level System Architecture



5-Stage NLP Pipeline: From Text to Floor Plan



Stage 1: Preprocessing + Stage 2: Intent Classification

Prompt Preprocessing

Input: Raw user text (Arabic or English)

Operations performed:

- Language detection (Arabic / English)
- Unicode normalization for Arabic characters
- Text cleaning and whitespace normalization
- Tokenization for downstream analysis

Output: Normalized prompt + detected language

Intent Classification — Two-Stage Strategy

First: Lightweight rule-based classifier using keyword patterns

If low confidence → LLM-based semantic classifier

Four intent classes:

- New Design Generation
- Design Refinement
- Information Query
- Unsupported Request

Output: Intent label + confidence score

 If **Unsupported** → pipeline terminates gracefully | If **Query** → routed to RAG | If **Design** → continues to Stage 3

Stage 3: Parameter Extraction + Stage 4: Layout Planning

Parameter Extraction (via LLM)

LLM extracts structured DesignParameters JSON:

- Apartment type (studio / 1BR / 2BR / 3BR / villa)
- Total target area (m²)
- Room types and quantities
- Room dimension constraints
- Adjacency preferences
- Orientation requirements

For refinements: current session context is loaded first

Output: Structured DesignParameters object

Layout Planning (Deterministic Algorithm)

Rule-based spatial planning — NOT the LLM:

- Primary spaces placed first (living room, master bedroom)
- Secondary spaces allocated in remaining area
- Service spaces (bathrooms, corridors) positioned last
- Door/window positions calculated from adjacency rules
- Non-overlapping placement enforced throughout

Output: Normalized LayoutObject with room geometries, wall segments, openings

Stage 5: Egyptian Building Code Compliance Validation

Validation Rules & Minimums

Rule Checked	Minimum Value
Bedroom area	9.0 m²
Bathroom area	2.5 m²
Kitchen area	4.0 m²
Living room area	12.0 m²
Corridor width	1.2 m (hard limit)
Room overlap	Zero tolerance
Boundary coverage	100% inside perimeter

Forbidden Adjacencies Detected:

- ✗ Bathroom ↔ Kitchen
- ✗ Bedroom ↔ Kitchen
- ✗ Bathroom ↔ Dining Room

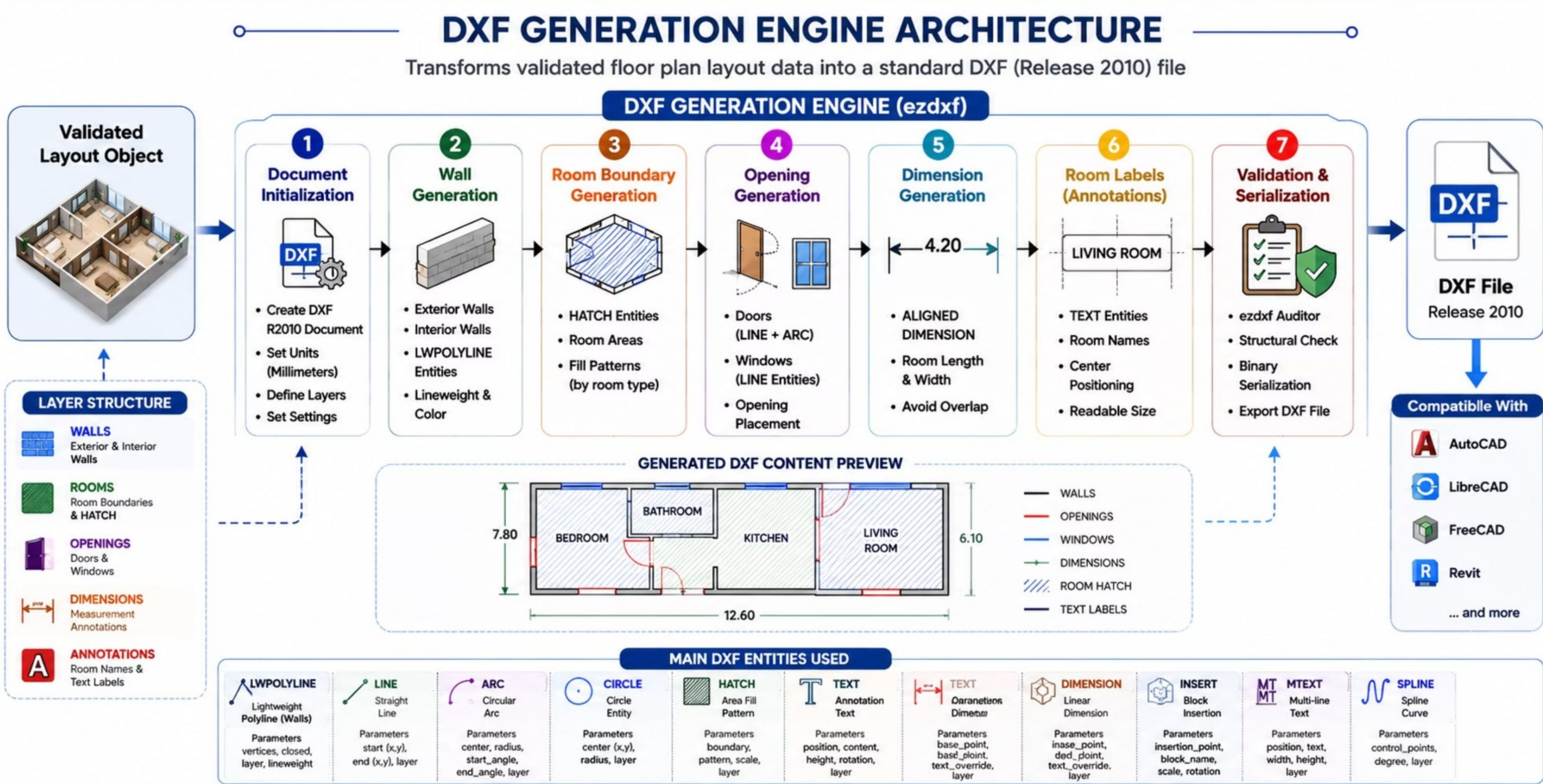
ComplianceViolation Object

Returned for each detected violation:

- rule_identifier
- affected_element
- measured_value
- required_value
- violation_description
- recommended_corrective_action

✓ Every layout is validated **BEFORE DXF generation** — violations are reported to the user immediately
EBC-01 through EBC-10 — all ✓ **Pass**

DXF Generation Engine — From Layout to CAD File



- 1

Document Initialization

Sets up DXF header, layers, and basic drawing properties.
- 2

Wall Generation

Draws wall polylines based on calculated room geometries.
- 3

Room Representation

Generates room boundaries and applies appropriate hatching patterns.
- 4

Opening Generation

Inserts doors (arcs) and windows (lines) into wall segments.
- 5

Dimension Annotations

Adds linear dimensions for room sizes and wall lengths.
- 6

Room Labels

Places text entities for room names and functional areas.
- 7

Validation & Export

Performs final DXF structure check and saves the file.

DXF Layer Organization

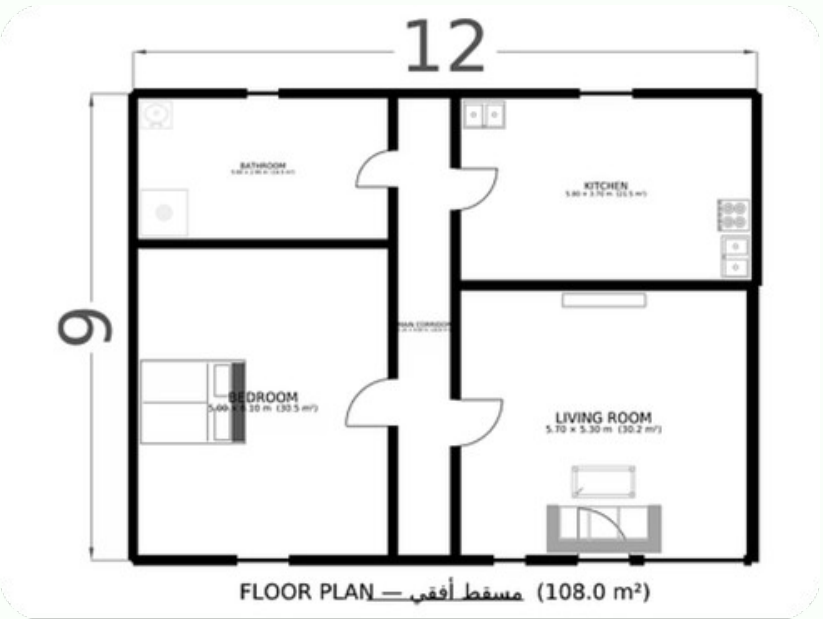
Layer	Color	Contains
WALLS	White/ Black	Structural wall polylines
ROOMS	Green	Hatch room boundaries
OPENINGS	Blue	Door arcs and window lines
DIMENSIONS	Yellow	Room measurement annotations
ANNOTATIONS	Cyan	Room name text labels

DXF Entities Supported by the Rendering Engine

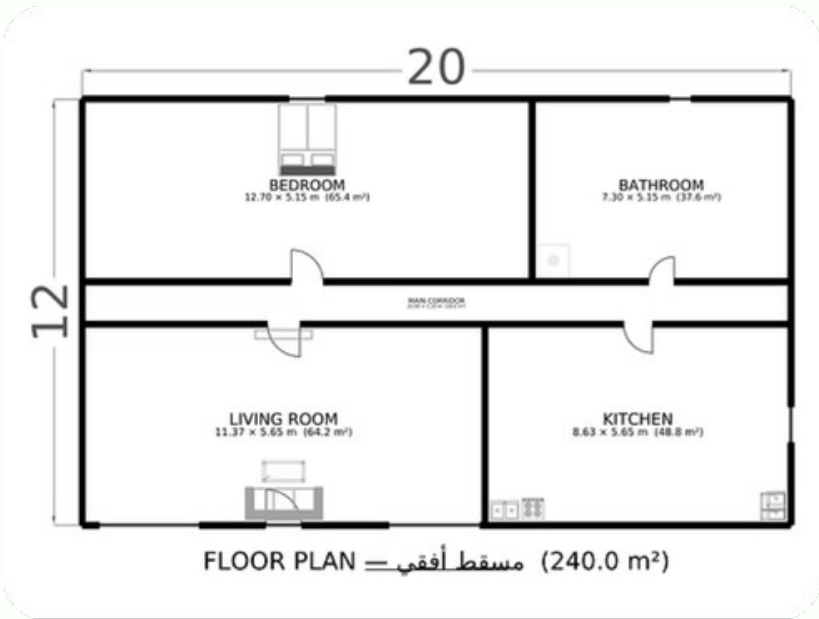
DXF Entity	Description	Key Parameters
LINE	Straight wall segment	start/end (x,y), layer, color
CIRCLE	Full circular feature	center (x,y), radius, layer
ARC	Door swing path	center, radius, start/end angle, layer
RECTANGLE	Closed room boundary	origin (x,y), width, height, layer
LWPOLYLINE	Multi-vertex wall or polygon	vertices list, closed, layer
TEXT	Room name annotation	position (x,y), content, height, layer
MTEXT	Multi-line formatted text	insertion point, content, width, layer
DIMENSION	Linear measurement mark	base point, def point, layer
HATCH	Area fill pattern	boundary, pattern name, scale, layer

20+ entity types supported in total — only relevant entities instantiated per design

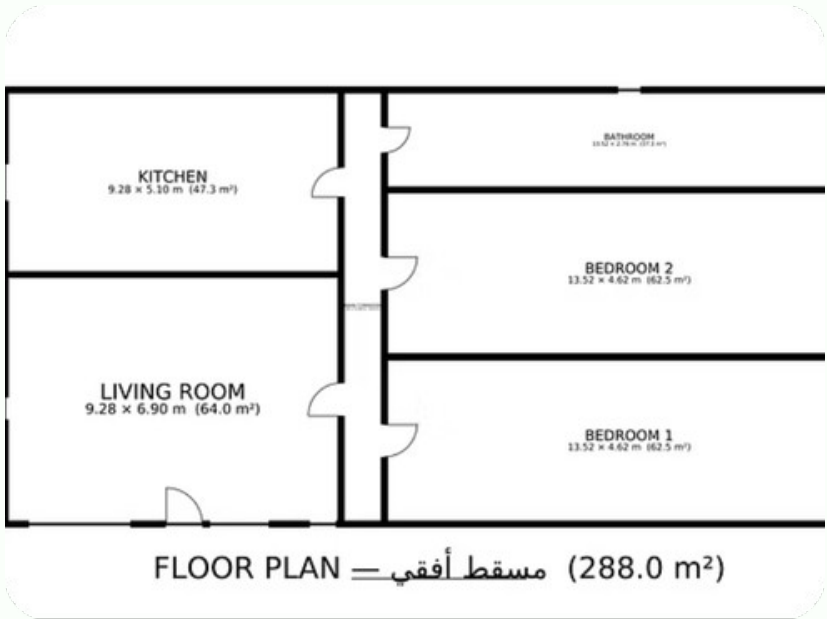
Sample Generated Floor Plans



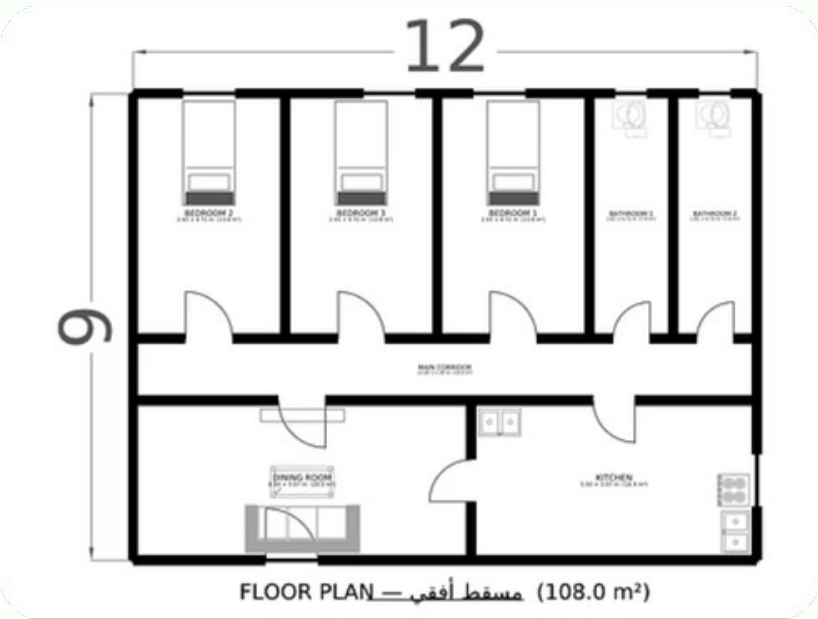
Studio Apartment
Generated Area: 108 m²



One-Bedroom Apartment
Generated Area: 240 m²



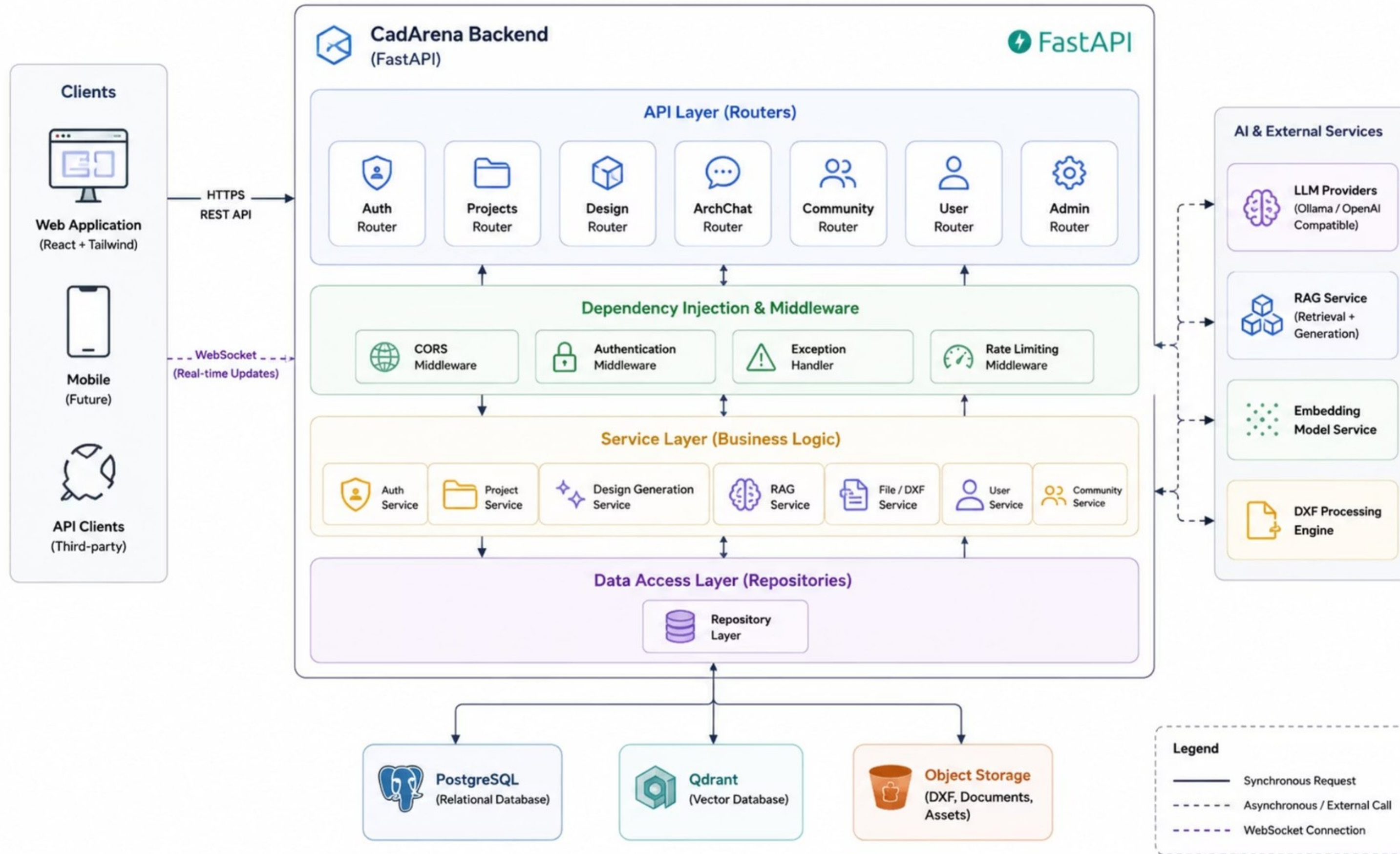
Two-Bedroom Apartment
Generated Area: 288 m²



Three-Bedroom Apartment
Generated Area: 108 m²

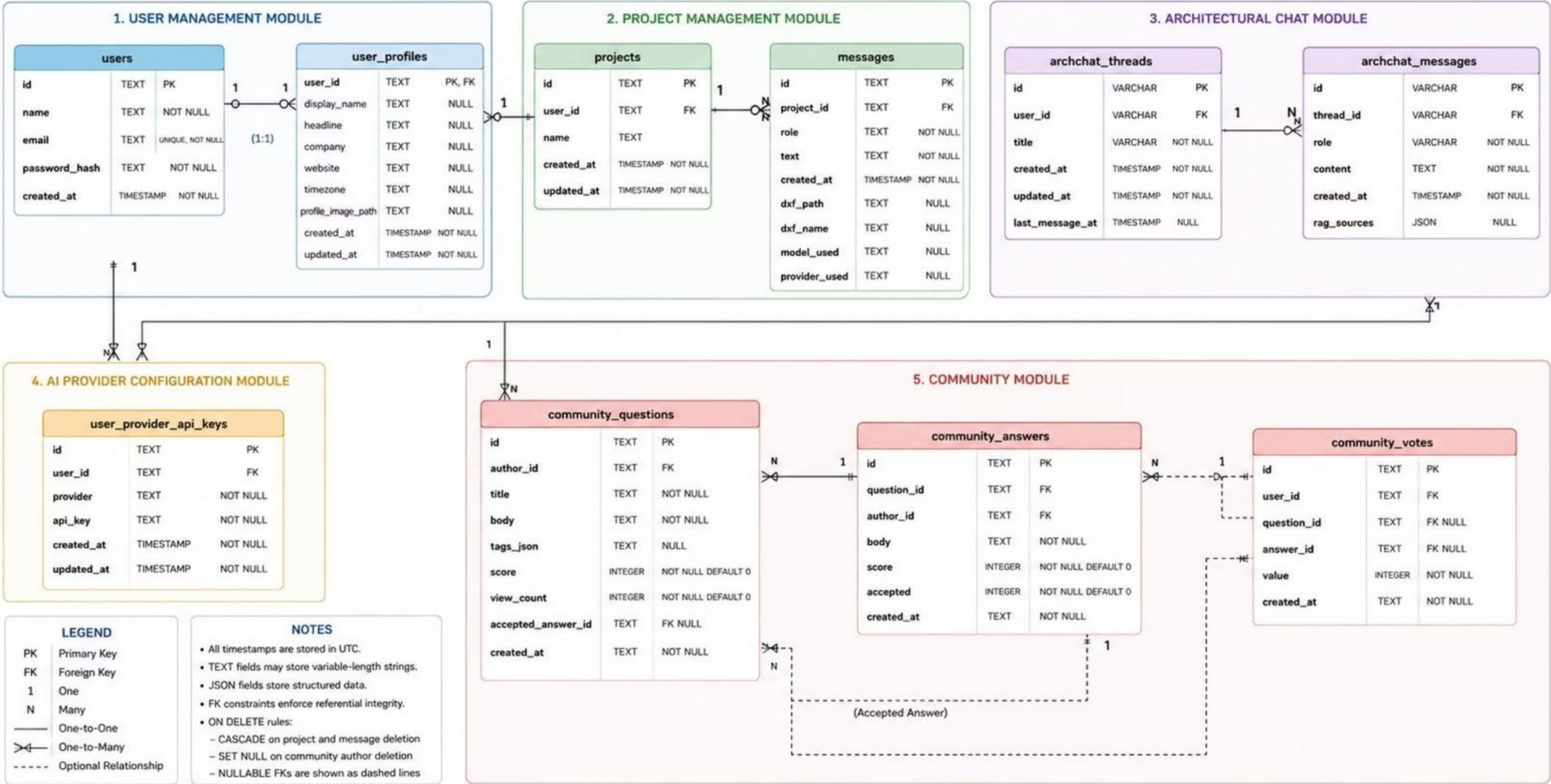
ALL OUTPUTS: EZDXF-VALIDATED · AUTOCAD-COMPATIBLE · DXF RELEASE 2010

FastAPI Backend Implementation



Database Schema

CadArena Database Schema



ArchChat — AI-Powered Engineering Knowledge Assistant

Next

Maher will present ArchChat — our Retrieval-Augmented Generation service for engineering knowledge assistance.



What is RAG and why we use it?



How ArchChat is architected (layer by layer)?



Live knowledge retrieval from engineering documents

From design generation to knowledge intelligence →

Why Retrieval-Augmented Generation?

Without RAG (Direct LLM)

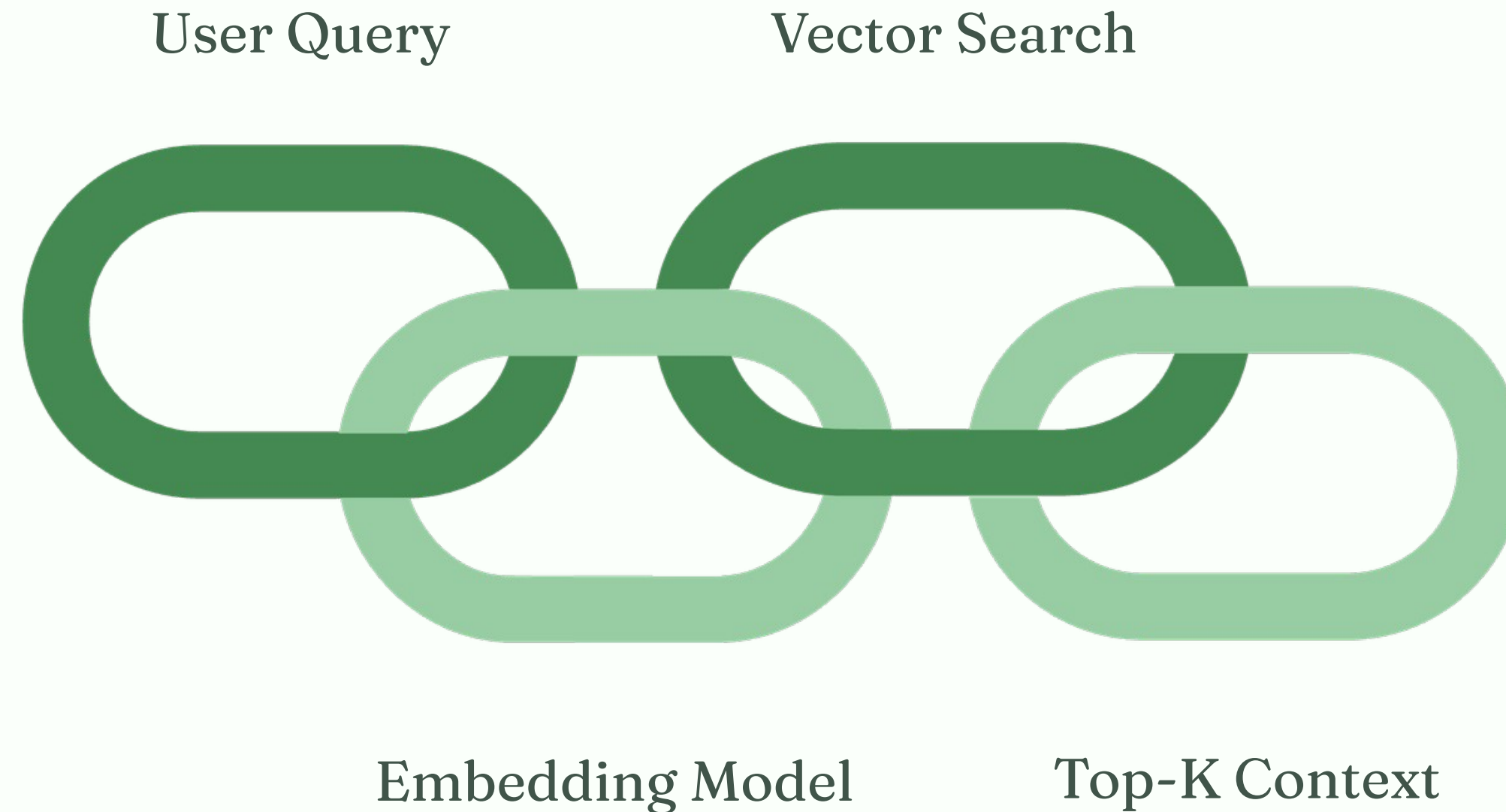
- Answers based solely on training data — may be outdated or incorrect
- No access to project-specific documents
- Cannot reference Egyptian Building Code specifics
- Hallucinations in technical domain answers
- No citations — user cannot verify the source

With RAG (ArchChat)

- Answers grounded in retrieved engineering documents
- Supports user-uploaded project documentation
- Can cite specific EBC 2023 rules with references
- Reduces hallucinations through retrieval grounding
- Every response includes source citations

RAG = RETRIEVAL-AUGMENTED GENERATION: THE LLM ANSWERS USING RETRIEVED DOCUMENTS, NOT JUST FROM MEMORY

ArchChat — Retrieval-Augmented Generation Architecture



Qdrant Vector Database

- Stores dense vector embeddings of document chunks
- Semantic similarity search at query time
- Supports project-specific and global knowledge bases

Knowledge Sources

- User-uploaded engineering documents
- Egyptian Building Code (EBC 2023) references
- Architectural design standards and guidelines
- Project conversation history for context

⚠ ArchChat is a completely separate service from the CadStudio design pipeline — independent FastAPI service, independent Qdrant instance

ArchChat — Five Processing Layers

1

1. Document Ingestion Layer

User uploads engineering PDFs, CSV, standards, or project documents. These are chunked into segments, processed by the embedding model, and then stored in Qdrant for efficient retrieval.

2

2. Embedding Layer (embed-multilingual-v3.0)

A Sentence Transformer model converts raw text chunks into dense semantic vectors. This captures the true meaning and context of the content, moving beyond simple keyword matching.

3

3. Vector Store Layer

Qdrant, our high-performance vector database, efficiently stores and indexes all generated embeddings. This enables millisecond-level semantic similarity searches across thousands of document chunks.

4

4. Retrieval Layer

The user's query is also embedded, and the top-K most semantically relevant document chunks are retrieved from Qdrant. These chunks are then combined to form a rich context window for the LLM.

5

5. Generation Layer

The retrieved context, along with the original user question, is fed into a structured LLM prompt. Our local Ollama model then generates a coherent response, complete with source citations, ensuring transparency and verifiability.

ArchChat Database Structure

archchat_threads

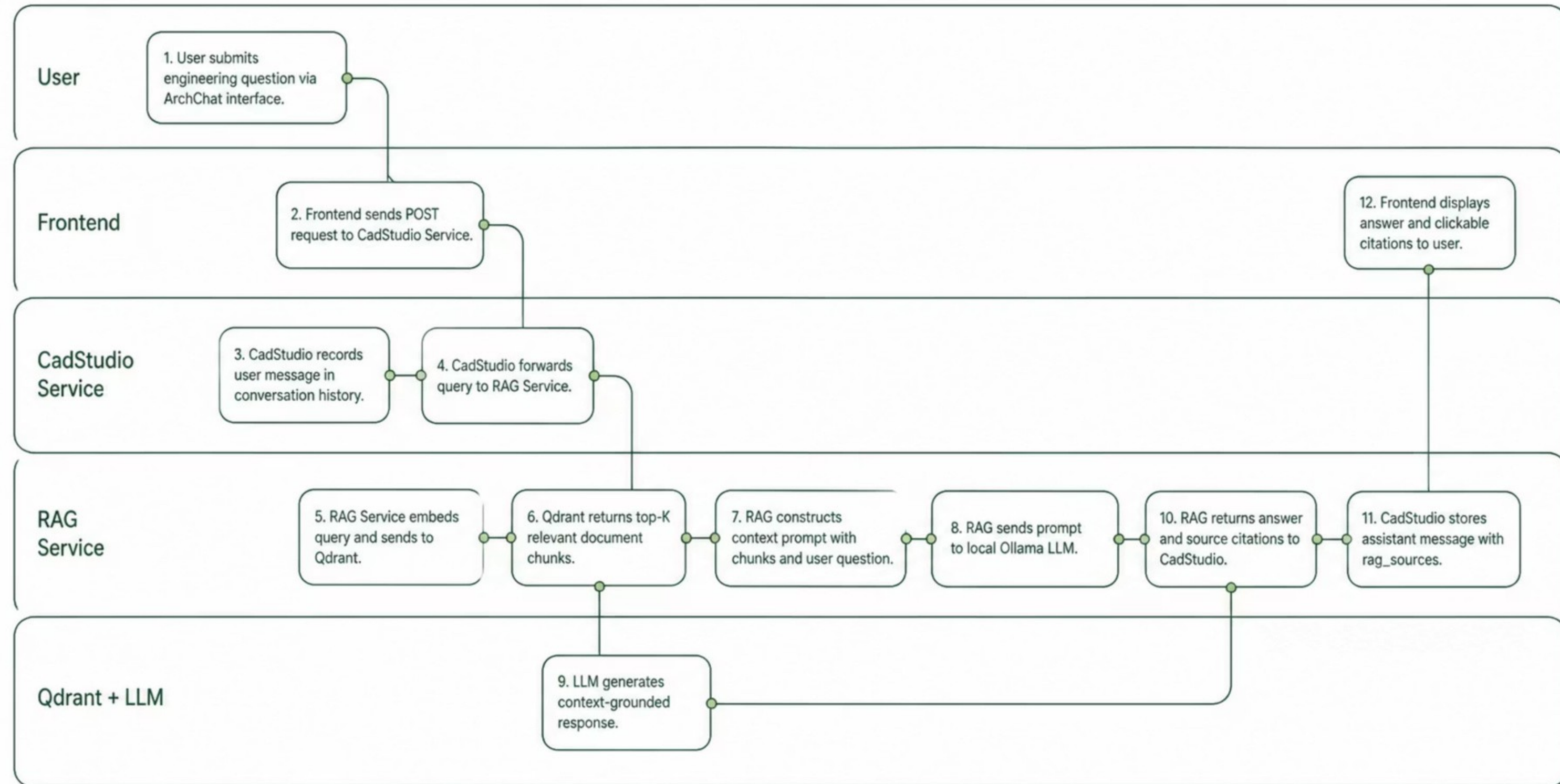
Column	Type	Description
id	VARCHAR	Thread identifier (PK)
user_id	VARCHAR	FK → users.id
title	VARCHAR	Conversation title
created_at	TIMESTAMP	Created
updated_at	TIMESTAMP	Last updated
last_message_at	TIMESTAMP	Latest activity

archchat_messages

Column	Type	Description
id	VARCHAR	Message identifier (PK)
thread_id	VARCHAR	FK → archchat_threads
role	VARCHAR	"user" or "assistant"
content	TEXT	Message body
created_at	TIMESTAMP	Sent time
rag_sources	JSON	Retrieved source citations

Each assistant message stores the RAG sources used — full traceability from answer to document

How ArchChat Answers an Engineering Question



Frontend & User Interface — Part 1

Next

Samar will present the CadArena frontend — Landing Page, Design Studio, and AI Design Generation workflow.



The Landing Page and onboarding experience



The Design Studio workspace



The AI Design Generation workflow in the UI

From backend intelligence to user-facing experience →

CadArena Landing Page



Direct access to CadStudio and ArchChat from navigation



Interactive prompt field — start generating immediately



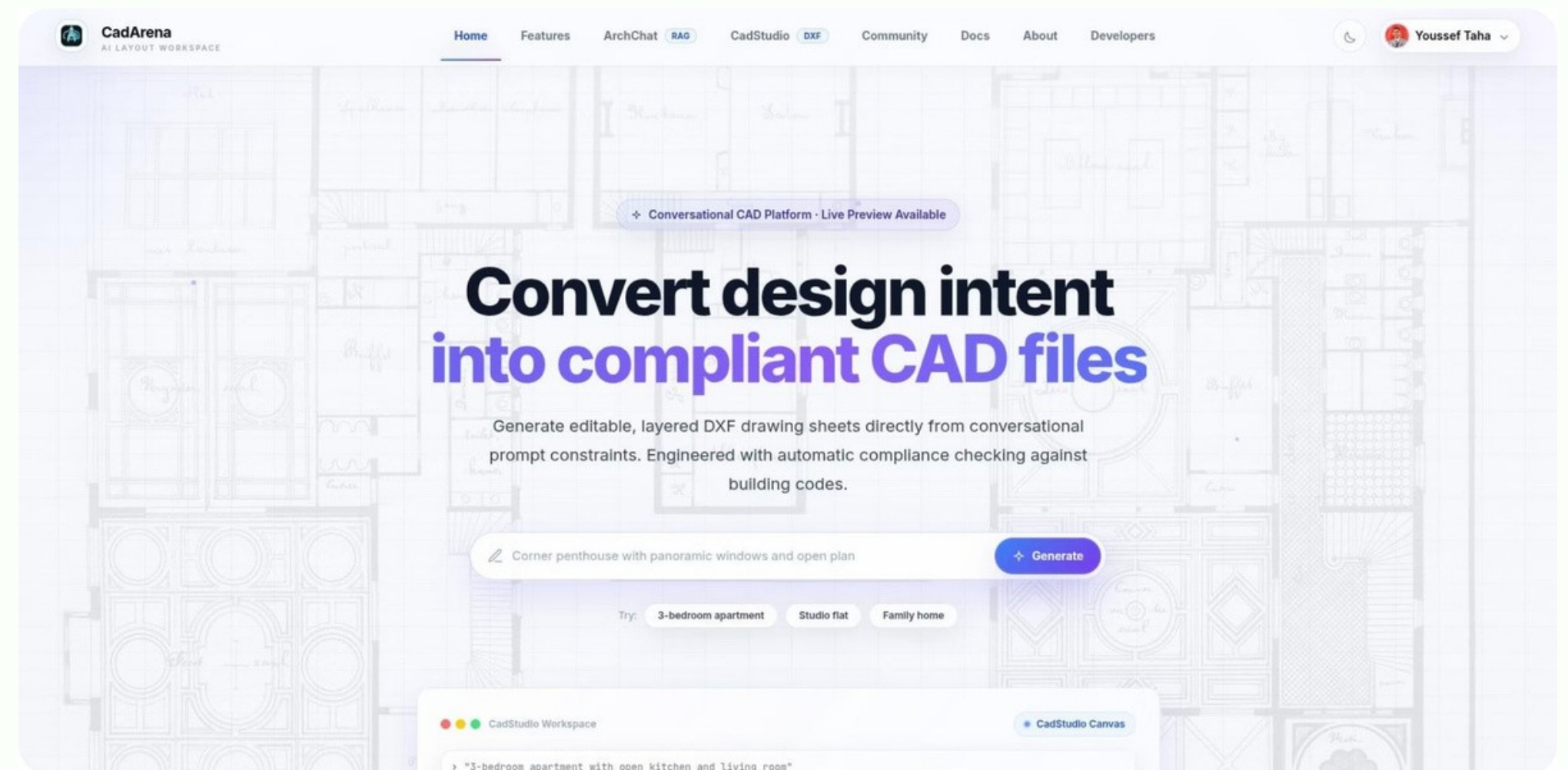
Suggested prompt examples for new users



Professional engineering-oriented branding



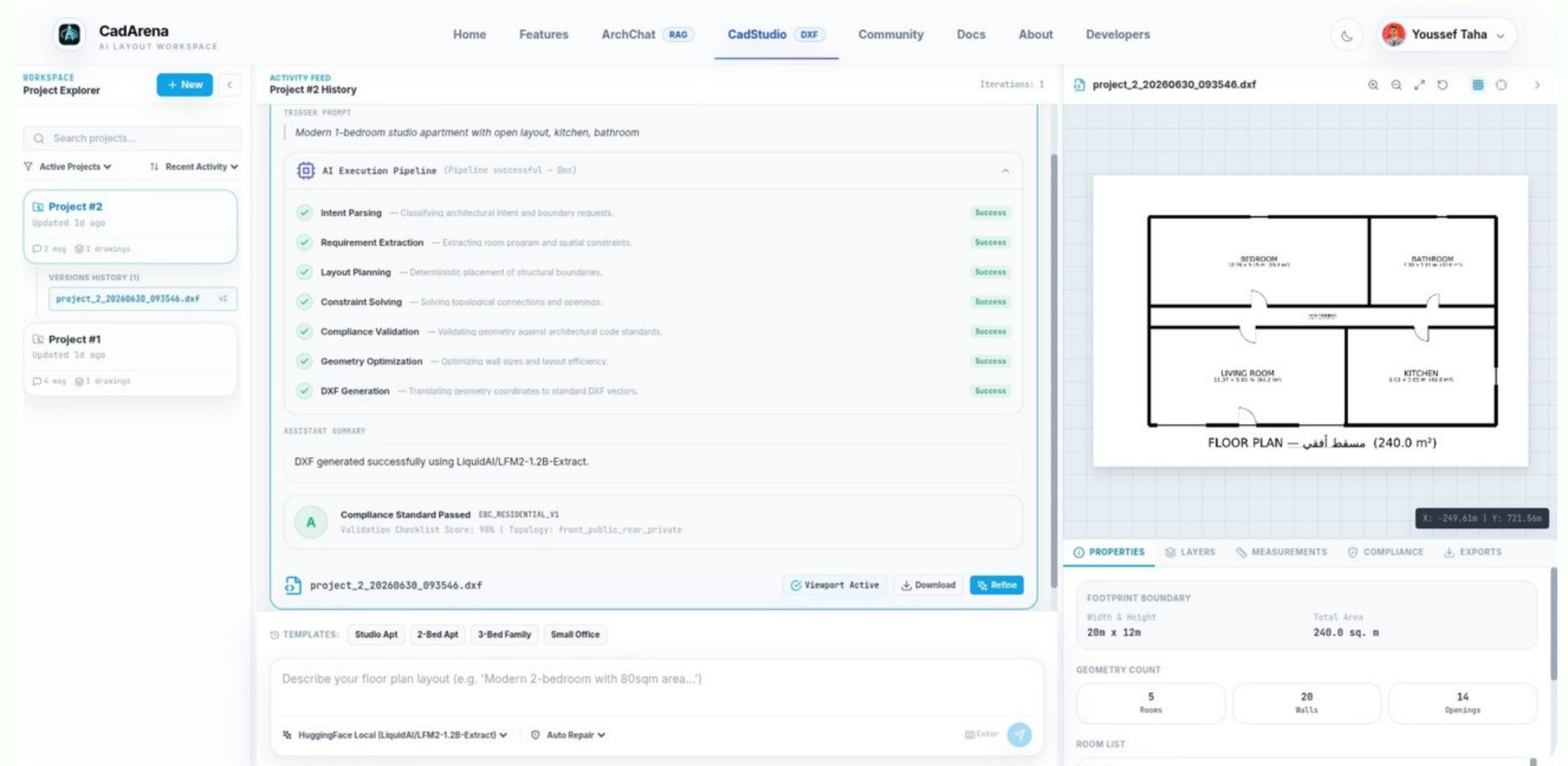
Responsive design — desktop, tablet, mobile



Built with React 18 + Tailwind CSS + Framer Motion

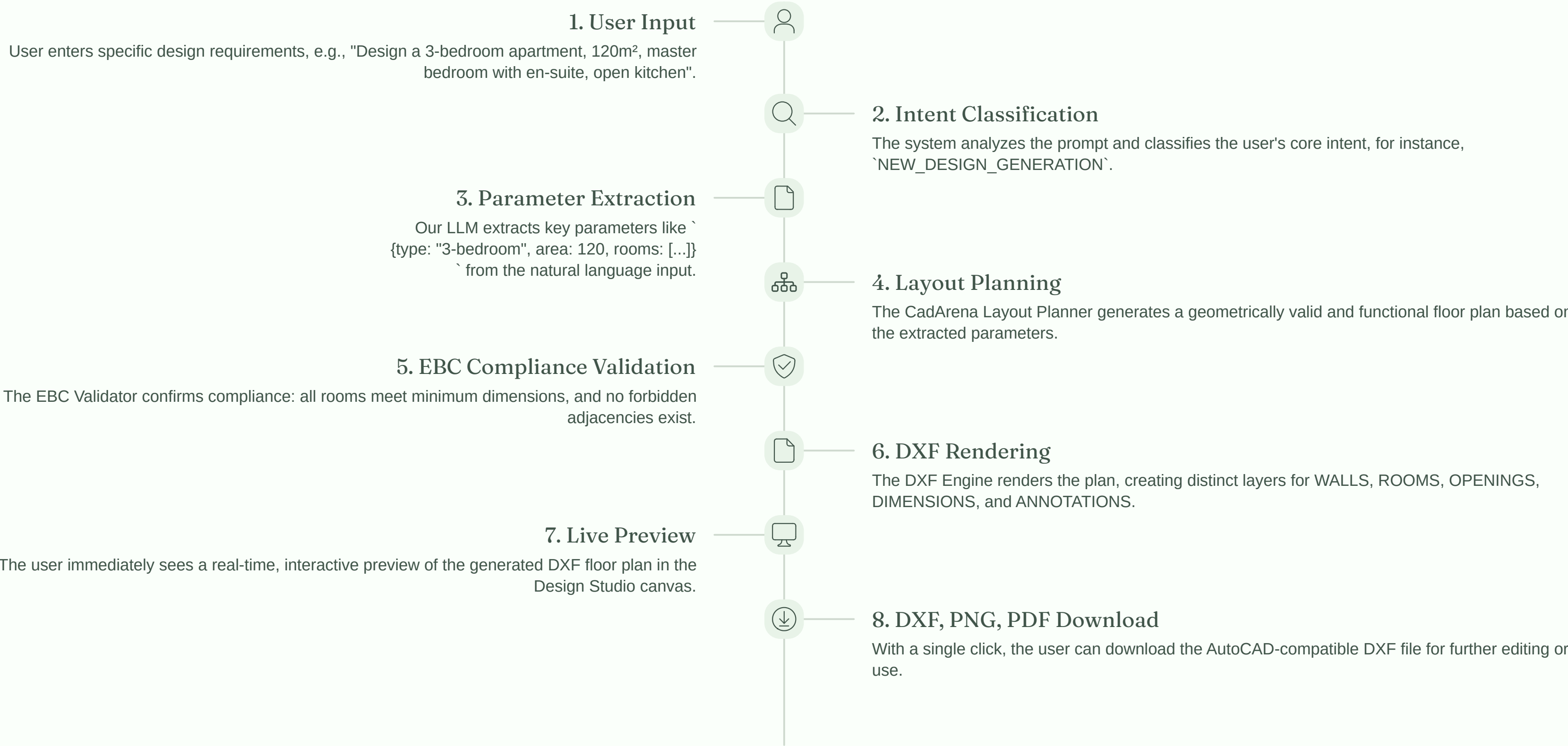
CadStudio — Design Studio Workspace

- Natural language prompt input (Arabic + English)
- Real-time DXF floor plan preview after generation
- AI model selector — choose Qwen, LLaMA, or Mistral
- Conversation history with previous design iterations
- One-click DXF download for CAD software



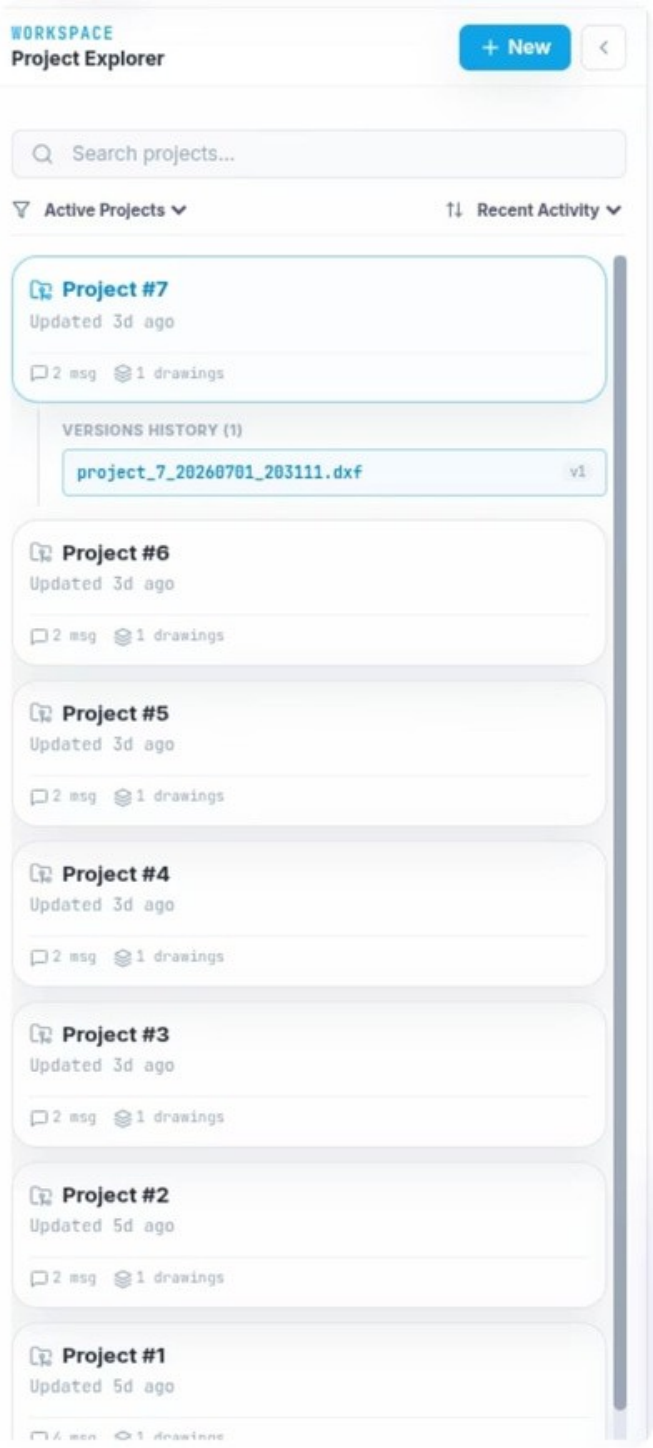
CadArena Design Studio with interactive prompt, real-time DXF preview, and essential controls.

AI-Assisted Design Generation (Multi-Agent) — End to End



Project Management — Organize Your Design Work

- ☆ **Create & Organize Projects**
Create, rename, archive, and delete architectural design projects
- ☆ **Full Conversation History**
Every prompt and generated design is saved per project — revisit any iteration
- ☆ **DXF File Linked to Prompt**
Each generated DXF is tied to its exact prompt for full traceability
- ☆ **Resume Any Session**
Open any previous project and continue from exactly where you left off
- ☆ **Secure Ownership**
Each project belongs exclusively to one authenticated user



DATA MODEL: USERS → PROJECTS → MESSAGES (PROMPT_TEXT · DXF_PATH · MODEL_USED)

Frontend & User Interface — Part 2

Next

Omnia will present the remaining frontend — ArchChat interface, Community Hub, and the overall tech stack.



ArchChat UI — engineering knowledge interface



Community Hub — Q&A and collaboration

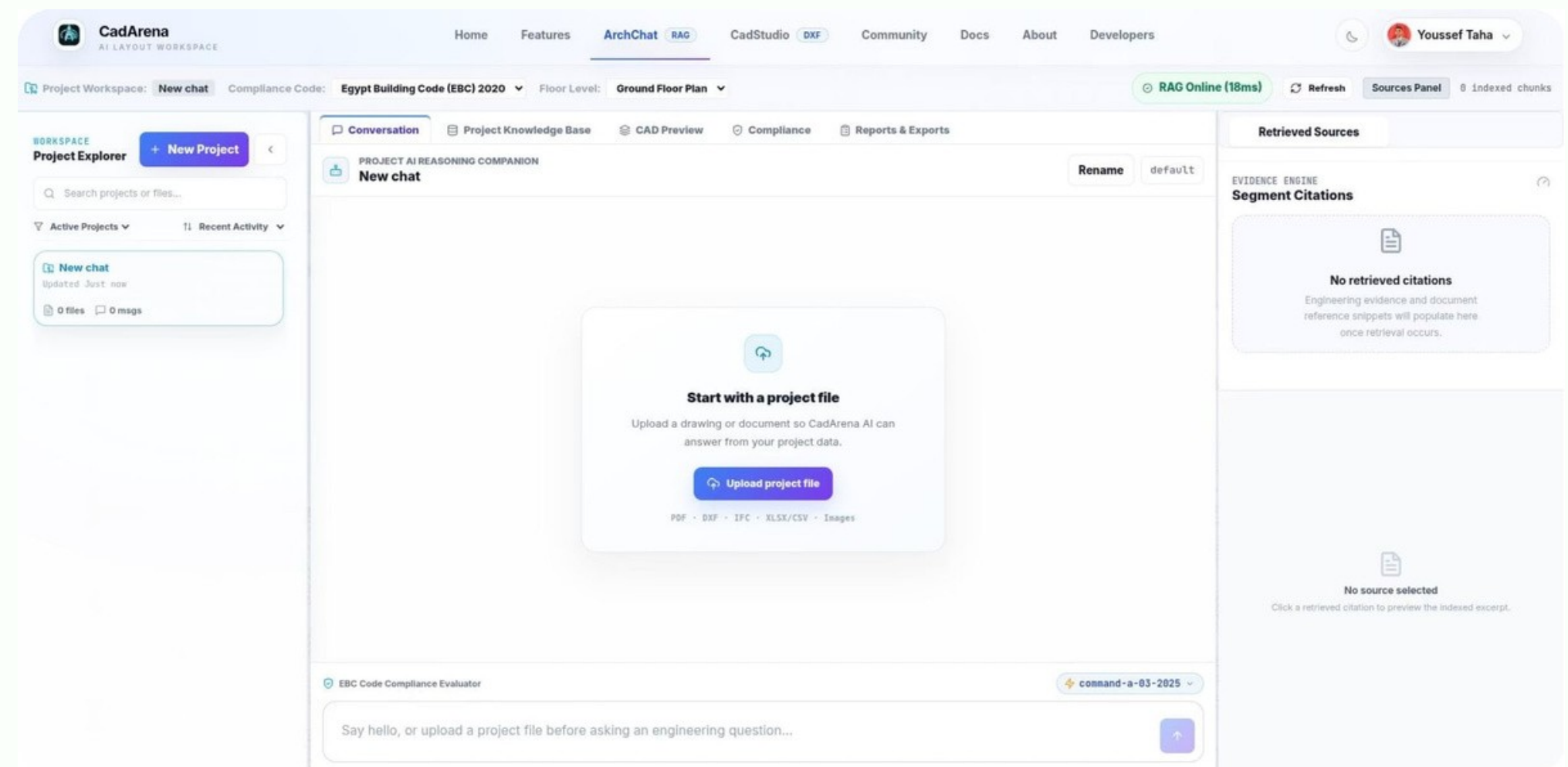


Technology stack and deployment

From design workspace to community knowledge →

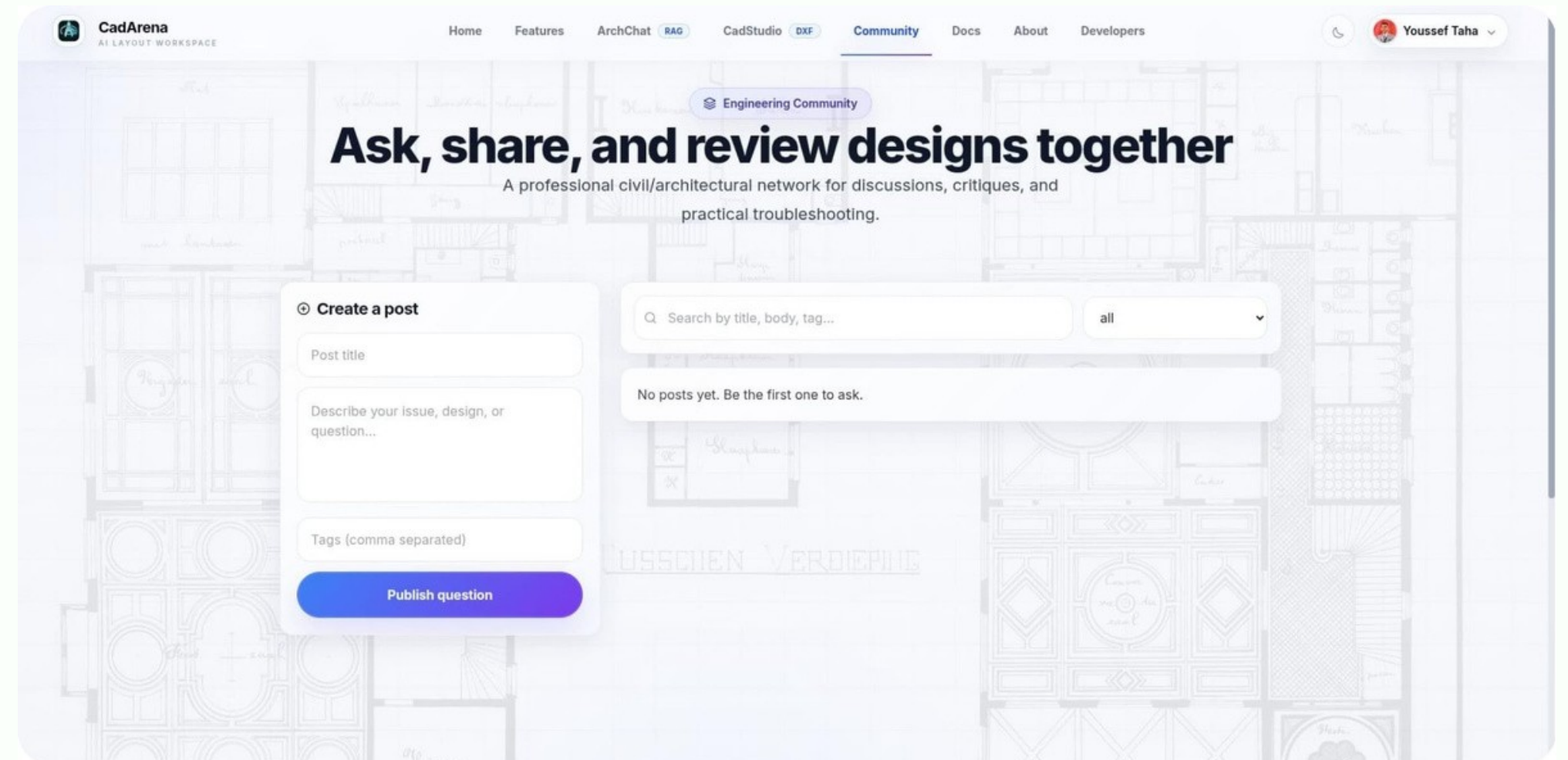
ArchChat — Engineering Knowledge Interface

- Conversational engineering assistant — ask questions in Arabic or English
- Upload project documents for project-specific knowledge retrieval
- Every answer includes clickable source citations
- Thread-based conversation history — revisit past questions
- Powered by Qdrant vector search + local Ollama LLM



Community Hub — Engineering Q&A

- Publish architectural questions — visible to all registered users
- Submit answers and replies from the community
- Upvote / downvote system — best answers rise to top
- Tag-based filtering for topic discovery
- Admin moderation for content quality



User Profile and Settings



Profile Details

Update display name, professional headline, organization, and website.



AI Provider Configuration

Configure preferred AI provider and API keys with encrypted storage.



Avatar Management

Profile image upload and avatar customization.



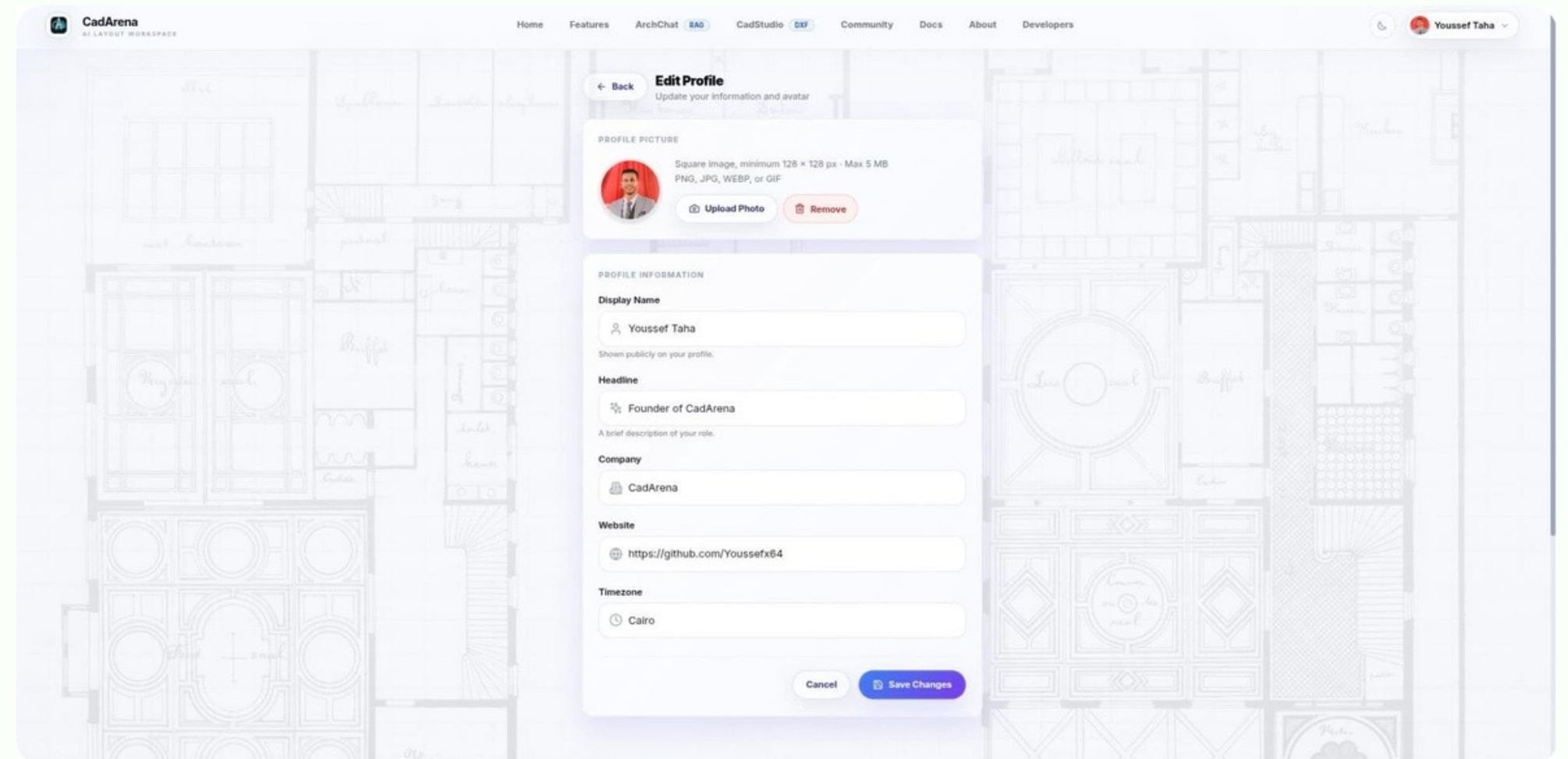
Timezone Settings

Adjust timezone for accurate timestamp display.



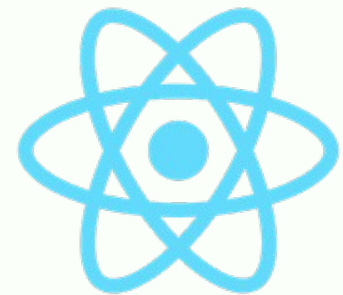
Account Security

Manage password and authentication settings for secure access.



user_provider_api_keys table — encrypted API keys for each user's chosen LLM provider.

Complete Technology Stack



React Native

React 18 + Tailwind
CSS

Frontend UI — Interactive
SPA, responsive layouts



Framer Motion +
Lucide React

UI Layer — Animations,
transitions, icons



FastAPI (Python 3.12)

Backend (MVC) — REST
APIs, business logic, AI
orchestration



Ollama

AI Runtime — Local LLM
inference (Qwen Series,
LLaMA 3, Mistral 7B)



LangChain

RAG Pipeline — Document
retrieval and prompt
construction



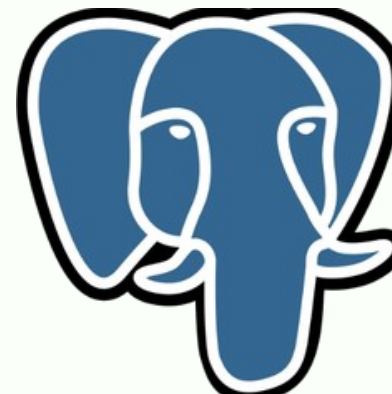
Qdrant

Vector Database — Semantic
document search



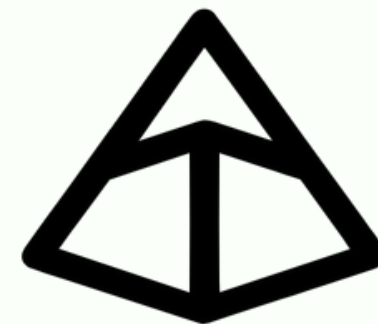
Sentence
Transformers

Embedding Model — Text-to-
vector conversion



PostgreSQL

Relational DB — Users,
projects, messages, metadata



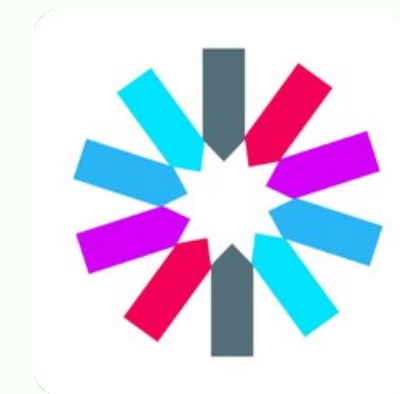
SQLAlchemy +
Pydantic

ORM + Validation — Schema
management and data
validation



ezdxf

CAD Engine — DXF file
generation (AutoCAD Release
2010)



JWT + OAuth 2.0

Authentication — Secure user
sessions and Google Sign-In



Docker + Docker
Compose

Deployment — Containerized
multi-service deployment

Testing, Evaluation & Conclusion

Returning

Youssef will present the final section —
Testing strategy, accuracy evaluation,
performance benchmarks, and conclusion.

- ☆ Testing Strategy — Unit, Integration, System, Accuracy
- ☆ Accuracy Metrics — DGSR, AFC, Validation Accuracy, DGT
- ☆ Performance Benchmarks — latency, memory, concurrent users
- ☆ Conclusion & Future Work — what we achieved and what comes next

From frontend experience to system validation →

Testing Strategy

Unit Testing

- **Tool:** pytest
- **Scope:** Individual backend services, parser modules, utility functions
- **Goal:** Verify correctness of individual software components

Integration Testing

- **Tool:** FastAPI TestClient + Integration Test Suite
- **Scope:** API routes ↔ services ↔ database ↔ RAG subsystem communication
- **Goal:** Validate interactions between system components

System Testing

- **Tool:** Manual testing through the web interface
- **Scope:** Complete end-to-end user workflows
- **Goal:** Verify major user scenarios execute successfully

Accuracy Evaluation

- **Tool:** Custom evaluation metrics
- **Scope:** AI-generated architectural layouts
- **Goal:** Measure geometric correctness, completeness, dimensional accuracy, EBC compliance

How We Measure CadArena's Accuracy?

1

Design Generation Success Rate (DGSR)

Formula: $(\text{Successful Generations} / \text{Total Prompts}) \times 100\%$

A generation is successful if the complete pipeline executes and DXF renders without errors.

2

Architectural Feature Completeness (AFC)

Formula: $(\text{Generated Features} / \text{Requested Features}) \times 100\%$

Measures whether all requested rooms and elements appear in the output.

3

Validation Accuracy (VA)

Formula: $(\text{Correctly Classified Violations} / \text{Total Violation Checks}) \times 100\%$

Measures how accurately the EBC validator detects and classifies violations.

4

Design Generation Time (DGT)

Total time from prompt submission to floor plan display in the Design Studio.
Reported as: Average / Minimum / Maximum across test cases.

5

DXF Export Success Rate (DESR)

Formula: $(\text{Successfully Exported DXF} / \text{Total Generated Layouts}) \times 100\%$
A success = file generated without errors AND opens correctly in CAD software.

Accuracy Evaluation — All Objectives Achieved

Metric	Evaluation Objective	Observed Result
Design Generation Success Rate	Generate complete layouts without runtime errors	✓ Successfully generated for all evaluated scenarios
Architectural Feature Completeness	Preserve all requested rooms and features	✓ All requested rooms correctly represented in outputs
Validation Accuracy	Correctly detect EBC constraint violations	✓ Validation module correctly identified valid and invalid designs
Design Generation Time	Provide interactive response time	✓ Generation completed within acceptable interactive time
DXF Export Success Rate	Produce downloadable CAD drawings	✓ All DXF files generated and opened correctly in CAD software

Performance Benchmark Results

2.1s

Fastest single-user generation
Studio apartment

13.8s

50 concurrent users
Stress test — marginal

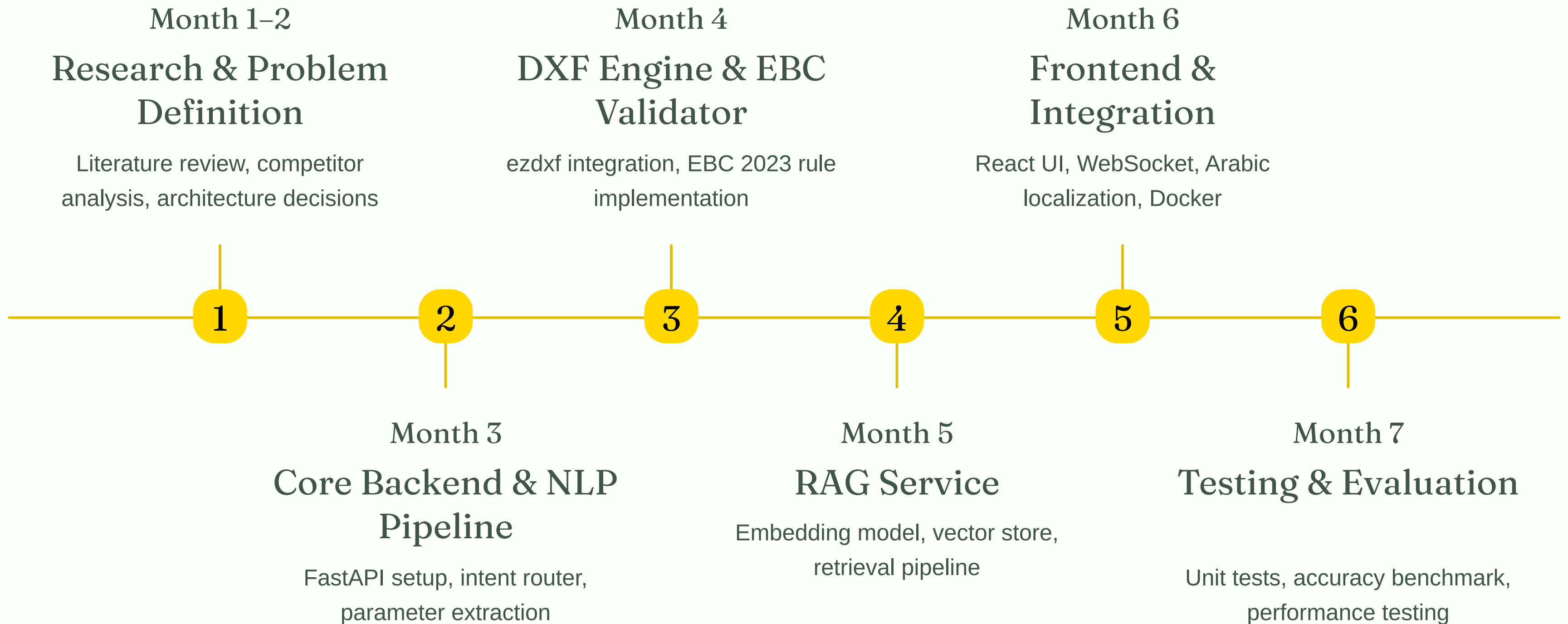
Stable

WebSocket 1-hour continuous session
Zero interruptions

Scenario	Users	Avg Latency	Peak Memory	CPU	Result
Single — Studio apartment	1	2.1s	1.8 GB	38%	✓ Pass
Single — Three-bedroom house	1	4.2s	2.3 GB	61%	✓ Pass
Single — Complex villa	1	6.8s	2.9 GB	78%	✓ Pass
Concurrent light load	10	5.1s	4.7 GB	72%	✓ Pass
Concurrent medium load	25	7.4s	9.2 GB	88%	✓ Pass
Concurrent stress test	50	13.8s	14.1 GB	96%	⚠ Marginal

50-user marginal: single Ollama GPU instance is the bottleneck — horizontal scaling resolves this.

Development Timeline



Future Directions

01	02	03
3D Architectural Modeling Extend the DXF engine to 3D solid entities. Enable STEP/IFC export for full Building Information Modeling (BIM) workflows.	Fine-tuned Domain LLM Train a specialized model on extensive Arabic/Egyptian architectural datasets. This will achieve higher accuracy in parameter extraction and intent classification.	Collaborative Multi-User Editing Implement real-time shared Design Studio sessions. Allow multiple users to work on the same project simultaneously, enhancing team productivity.
04	05	06
Mobile Application Develop native iOS and Android applications. This will enable users to capture voice prompts and view DXF previews on-the-go.	Expanded Building Codes Integrate support for additional building regulations like Saudi Building Code (SBC) and UAE Fire Code. Expand compliance validation to international standards beyond EBC 2023.	AutoCAD / Revit Plugin Create a native CAD plugin for seamless integration. Bring CadArena AI generation capabilities directly inside existing professional CAD tools.

Conclusion

CadArena successfully demonstrates that combining Large Language Models, Retrieval-Augmented Generation, and rule-based geometry engines produces a reliable, production-ready architectural design platform — without requiring any labeled training data. The system generates EBC 2023-compliant residential floor plans from natural language prompts in Arabic or English, and exports them as professional-grade DXF files, in under 5 seconds.

Acknowledgements

We extend our deepest gratitude to:

Dr. Amira Mohey El Din Mohamed

For her expert supervision, continuous guidance, and invaluable feedback throughout this project.

Beni-Suef National University

Faculty of Computers and Artificial Intelligence

Open-source community — FastAPI, ezdxf, React, LLaMA, Mistral

References

- [1] Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.
- [2] Brown et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.
- [3] Chen et al., "Evaluating Large Language Models Trained on Code," 2021.
- [4] Wu et al., "DeepCAD: A Deep Generative Network for CAD Models," ICCV, 2021.
- [5] Tikhonov, "ezdxf — Python DXF Library," 2023. ezdxf.readthedocs.io
- [6] Ramírez, "FastAPI — Modern Python Web Framework," 2023. fastapi.tiangolo.com
- [7] Egyptian Housing Ministry, "Egyptian Building Code EBC 2023," 2023.
- [8] Touvron et al., "LLaMA 2 — Open Foundation Models," 2023.
- [9] Jiang et al., "Mistral 7B," arXiv, 2023.
- [10] RFC 6455 — The WebSocket Protocol, IETF, 2011.

WORKFLOW

From natural language to structured layout coordinates

Describe your spatial boundaries, room constraints, and structural adjacencies. ArchChat compiles your description into structured vector assets.

01



Define Spatial Constraints

Specify your room quantities, desired adjacencies, and layout bounds using natural language prompts within the ArchChat interface.

02



Synthesize Layout Coordinates

The hybrid generation pipeline extracts design intents, runs them through geometric planners, and maps compliant room coordinates.

03



Export CAD Drawing Sheets

Download structured DXF assets directly compatible with industry-standard CAD software, or continue editing within the CadStudio workspace.



Entire screen



Thank You! (ألف شكر لحضراتكم)

Department of Artificial Intelligence and Data Science

Youssef Taha • Maher Ibrahim • Omnia Ayman • Samar Nashaat

cadarena.ai@gmail.com

[Github: github.com/Youssefx64/CadArena](https://github.com/Youssefx64/CadArena)

We welcome any questions from the honorable evaluation committee.

Questions & Discussion Welcome