ChatGPT

Buildwise AI — Multi-Agent Real Estate Lease Assistant

Buildwise AI is a conversational multi-agent assistant that guides tenants, landlords, and agents through every stage of the leasing journey. From apartment hunting to move-out, our AI co-pilot provides instant answers, personalized insights, and proactive alerts — all while logging interactions in a built-in CRM.

Overview and Vision

Renting or managing property can be **confusing and stressful**. Leases are full of legalese, city regulations are complex, and issues like repairs or compliance can sour the experience. **Buildwise AI** was created to make the real estate lease lifecycle **smart, simple, and even enjoyable**. It's like having a team of experts (legal advisor, building inspector, CRM agent, and more) **all in one friendly chat assistant**, available 24/7.

What does Buildwise AI do? It uses multiple specialized AI agents working together to **understand user needs, retrieve relevant knowledge, and provide actionable answers**. Whether you're a tenant asking *"Is this apartment a good fit?"*, a landlord checking *"Am I compliant with new laws?"*, or an agent juggling many clients, Buildwise AI has you covered. Our platform's vision is to **democratize expert knowledge** – helping users make wiser decisions and fostering trust in every rental relationship.

Features Across the Lease Lifecycle

Buildwise AI assists users at **four key stages** of renting, ensuring no question goes unanswered:

- **1. Pre-Lease (Apartment Search & Evaluation):** Help users find the best-fit apartments based on their criteria. The assistant can filter listings by neighborhood amenities, check environmental risks (e.g. flood zones, building emissions laws), analyze lease clauses, and even compute a "fit score" for each option. *Example: "Find me a pet-friendly 2BR under $3k in Brooklyn, not in a flood zone."* – Buildwise AI will search listings, cross-check flood zone data and Local Law 97 status, and summarize top choices with pros/cons.

- **2. During Lease (Onboarding & Compliance):** Once a lease is in progress (signing or just after move-in), the assistant verifies that **everything is in order and clear**. It can parse the lease document to explain terms, check compliance with local housing codes, and confirm building status (e.g. any open violations or required inspections). If ambiguities or red flags are found, it alerts the user. *Example:* reviewing a lease for illegal clauses or ensuring the building's certificates are up to date.

- **3. Post-Lease – Living (Day-to-Day Tenancy):** Throughout the tenancy, Buildwise AI becomes an **on-call housing advisor**. It can answer repair questions (*"There's mold in my apartment, what do I do?"*), provide legal guidance (*"Can my landlord enter without notice?"*), and mediate common landlord/

tenant frictions with suggestions grounded in local law. The assistant stays context-aware of the tenant's lease and history, so advice is personalized. It can even draft polite emails or documents for dispute resolution on the fly.

- **4. After Lease (Move-Out & Beyond):** When the lease term ends, the assistant helps wrap things up smoothly. It will retrieve any needed documents (lease agreements, payment history), guide tenants or landlords through dispute processes (like recovering security deposits or documenting damages), and generate a **historical summary** of the tenancy. This summary can serve as a reference for future rentals or investments. *Example:* a tenant can ask, "Can you summarize my rental history here for my next landlord?" and get a neatly formatted report.

Each stage is **seamlessly connected**. The conversation and data from earlier stages carry over, thanks to the integrated CRM. This means Buildwise AI remembers your preferences (e.g. you love quiet neighborhoods, you need pet-friendly places) and your past issues, providing continuity in assistance.

Multi-Agent Architecture: How It Works

Under the hood, Buildwise AI is powered by a team of specialized **AI agents**, each with a clear role. A central **Orchestrator** (the "AI Concierge") manages these agents, activating them as needed and combining their outputs into a coherent reply [1] [2]. This **multi-agent design** allows complex problems (like analyzing a lease or assessing a building) to be broken into smaller tasks handled in parallel, which is more efficient and traceable than a single monolithic AI [3] [4]. Here are the agents in our system:

- **Property Info Agent:** Connects to property databases and public records to fetch **building information** – ownership, year built, maintenance history, open violations, past repairs, etc. For example, it can pull NYC Department of Buildings data to see if a building has unresolved safety issues or check 311 complaints. It provides factual context about the property itself.

- **Environmental Risk Agent:** Aggregates **environmental and regulatory risk data**. It checks if a building is in a floodplain or high-fire-risk zone, looks up climate-related laws like NYC's Local Law 97 (which caps carbon emissions for large buildings [5]), and zoning designations. This agent ensures users know about any **external risks or compliance requirements** associated with the property (e.g. potential flood insurance needs, upcoming sustainability upgrades mandated by law).

- **Lease Analyzer Agent:** An AI legal assistant that **parses lease documents and local housing codes**. Using GPT-4's vision and NLP capabilities, it can ingest a PDF or image of a lease and extract key clauses. It cross-references local landlord-tenant regulations (we index documents like NYC Rent Guidelines or Tenant Rights pamphlets) to flag anything unusual. This agent explains confusing legal terms in plain language and checks that the lease aligns with applicable laws (no illegal clauses). Essentially, it's your personal lease attorney within the chat.

- **Decision Agent:** This "brain" agent **synthesizes insights** from all other agents to generate **recommendations and answers** for the user. For instance, if you ask "Is this apartment a good fit for me?", the Decision Agent will gather inputs: Property Info (building age, condition), Env Risk (flood zone, etc.), Lease analysis (clauses about pets or fees), plus your personal preferences from the CRM. It then produces a balanced answer, perhaps even a numeric *fit score*, and suggests next

steps (like questions to ask the landlord, or alternate listings). The Decision Agent ensures the final response is **holistic and user-friendly**, not just raw data.

- **CRM Agent:** Overseeing the **conversation memory and user profile**, the CRM Agent logs each interaction and tags it by stage and topic. It stores user-provided info like name, email, budget, preferred neighborhoods, and conversation history in a MongoDB database (our lightweight **CRM**). This agent retrieves relevant past chat context to keep the conversation personalized and consistent [6] [7]. For example, if a user talked about their pet a week ago, the CRM Agent will remind the system of that when discussing leases (so the AI can double-check pet policies). It also allows retrieval of past conversations or actions on demand via API.

Orchestration: All agents work in concert through a controller logic. When a user question comes in, our system decides which agents are needed. For example, the user asks about a lease clause – we mainly invoke the Lease Agent (to parse the clause and get legal context) and maybe the CRM Agent (to recall the user's lease document from memory). For a broad question like "Should I rent this place?", the orchestrator might trigger **all agents** to gather a complete picture [2]. The Decision Agent (or orchestrator itself) then **assembles a final answer**, citing specifics (like *"the building has 3 open violations for plumbing"* or *"it's in Flood Zone 2"*) along with advice. This modular approach makes the system's reasoning more transparent and extensible than a single large model [8] [9] – each agent's output can be logged and examined, which is great for debugging and trust.

Tech Stack and Innovation

Our implementation leverages **cutting-edge AI and robust backend tech** to deliver a production-ready solution:

- **LLM Backbone:** OpenAI **GPT-4** (with vision support) powers the natural language understanding and generation. GPT-4's multi-modal ability lets us feed images/PDFs (e.g. a scanned lease) directly for analysis. The LLM is used in each agent (with specialized prompts per agent role) as well as for the overall conversation flow.

- **Retrieval-Augmented Generation (RAG):** We integrate a **Haystack/LlamaIndex** pipeline for knowledge retrieval [10]. Key documents (NYC housing laws, sample lease templates, building code snippets, open data CSVs) are indexed so that when users ask something factual (*"What's the flood zone for this address?"* or *"What does clause 5 mean?"*), the relevant text is retrieved and provided to GPT-4 as context. This ensures **accurate, up-to-date answers grounded in real data** instead of the LLM guessing [11]. RAG is used heavily by the Lease and Env Risk agents to fetch specific regulations or data points.

- **Backend Framework:** Python FastAPI is used to build a **RESTful API** that serves as the interface for both the chat and CRM functionality [12]. FastAPI was chosen for its speed and ease of defining asynchronous endpoints, critical for handling concurrent agent calls and streaming LLM responses. We structured the project as a scalable service, ready to deploy to cloud (e.g. AWS or Azure) with containerization.

- **Database (CRM):** MongoDB (Atlas) is our **database for the CRM** module [13]. It stores user profiles, preferences, conversation logs, and agent outputs. We defined schemas for User, Conversation, and Message, including fields for stage tags and timestamps. This persistent store enables long-term memory: a user can return after weeks and Buildwise AI will recall their context. MongoDB's flexibility allows quick iteration and it integrates easily with Python (through Motor or PyMongo).

- **External Data & Integrations:** We tap into **open APIs and datasets** for real-time info:

- NYC OpenData for property and environmental data (e.g. a dataset of building energy grades, a feed of DOB violations).
- FEMA APIs or NYC Flood Hazard Mapper for flood zones.
- We prepared a **regulatory knowledge base** including summaries of Local Law 97, NYC tenant rights, warranty of habitability standards, etc., so the AI can reference those in answers.

- Future integration: calendar or email APIs to automate sending notices or reminders (e.g. alert landlord about a repair, or remind tenant of lease renewal dates).

- **Modular Codebase:** Our GitHub repo is organized clearly for readability and collaboration:

```
buildwise-ai/
├── backend/
│   ├── main.py            # FastAPI app setup, startup events
│   ├── api/               # API route definitions
│   │   ├── chat.py        # /chat endpoint implementation
│   │   ├── crm.py         # /crm endpoints (create_user, update_user,
etc.)
│   │   └── docs.py        # /upload_docs endpoint for RAG
│   ├── agents/            # AI agent logic modules
│   │   ├── property_info_agent.py
│   │   ├── env_risk_agent.py
│   │   ├── lease_agent.py
│   │   ├── decision_agent.py
│   │   └── crm_agent.py
│   ├── core/              # Core orchestration, utilities
│   │   ├── orchestrator.py   # logic to coordinate multiple agents
│   │   ├── rag.py            # retrieval-augmented generation utilities
│   │   └── db.py             # database connection and ORM models
│   └── data/              # Sample data, e.g. a demo lease PDF, knowledge
base files
├── README.md           # Documentation and setup instructions
└── ... (config files, requirements.txt, etc.)
```

This structure enforces **modularity**, making it easy for team members to work on different agents in parallel. It also maps to our API routes for clarity.

REST API Endpoints

Buildwise AI is accessible via a RESTful API. Below is an outline of the key endpoints (all returning JSON), aligning with the hackathon specs [14]:

- `POST /chat` – Submit a user's message and receive the assistant's response. The request can include a user/session ID to maintain context. The backend will route the query through the relevant agents and return a conversational reply (along with any relevant metadata like response time or sources). *Example:* `{"user_id": "123", "message": "Explain clause 4 of my lease"}` -> returns GPT-4 answer with context from the lease doc.

- `POST /upload_docs` – Upload documents to the system's knowledge base. This supports files like PDF, TXT, or images. Uploaded docs are indexed for RAG so the AI can pull information from them later [15]. In the prototype, this is used to upload a lease agreement or perhaps a building report. (For the hack, we also allow posting a URL to fetch data directly.)

- `POST /crm/create_user` – Create a new user profile in the CRM [16]. A user ID is generated, and provided details (name, email, preferences like budget or pet info) are stored. This ID will tie together all conversations and documents for that user.

- `PUT /crm/update_user` – Update an existing user's info by ID. E.g., after a conversation, if we extracted that the user's pet is a *cat* or their preferred move-in date is *Jan 2024*, we could save that via this endpoint (the AI can trigger it internally too).

- `GET /crm/conversations/{user_id}` – Retrieve the full conversation history and notes for a given user [17]. This is useful for a support agent or the user themselves to review what advice was given. We include stage tags (Pre-Lease, During Lease, etc.) for each message to summarize the topics discussed.

- `POST /reset` – Clear the current conversation context (memory) for a session [18]. This allows the user to start a fresh dialogue without legacy context. (The CRM still keeps the history in database, but the AI will not use it unless needed.)

These endpoints make it easy to integrate Buildwise AI into other applications as well. For instance, a property management platform could call `/chat` to get automated answers for tenants, or use the CRM endpoints to sync user data.

Personas & Use Cases

Buildwise AI creates value for multiple stakeholders in the real estate leasing ecosystem:

- **Tenants (Renters):** For renters, our assistant is like a personal rental advisor. It demystifies the process – finding apartments that truly match your needs, warning you of hidden issues (like *"this building has 5 noise complaints"* or *"flood risk zone"*), and being there throughout your lease for any questions (*rent increase legality, repair rights, how to break a lease if needed*). This empowers tenants

with **knowledge and peace of mind**. No more nervously Googling clauses or laws – just ask your AI. It's especially helpful for first-time renters or those moving to NYC who aren't familiar with local laws.

- **Real Estate Agents & Property Managers:** For professionals, Buildwise AI is a force-multiplier. It acts as an **AI assistant that can handle initial tenant queries** and screening. Agents can input a client's preferences and let the AI shortlist suitable listings with data-backed reasoning. During lease signing, the AI can auto-generate summary sheets of key terms or check if the lease adheres to rent stabilization rules. Property managers can use it to keep track of multiple tenants' issues: the CRM tagging (e.g. "Maintenance: unresolved") helps ensure nothing falls through the cracks [19]. By automating Q&As and paperwork, agents free up time to focus on high-level client service. Buildwise AI can also serve as a training tool for junior agents, providing instant answers about regulations or best practices.

- **Landlords & Investors:** Owners benefit from **proactive risk management and data insights**. The Env Risk agent will alert an owner if their building might incur fines (e.g. *"LL97 compliance required next year, projected fine $X if not addressed"*). The Property Info agent can compile maintenance and complaint history, helping landlords prioritize improvements. After a lease ends, an owner can ask the AI for a *"tenant report"* – summarizing how the tenancy went (payments, issues, etc.), which is useful for evaluating future rentals or portfolio performance. For investors doing due diligence on a property, Buildwise AI can rapidly analyze all leases and records of a building to flag anything concerning (illegal units, short lease terms, unusual clauses). Overall, it helps landlords and investors make **data-driven decisions, stay compliant, and improve tenant satisfaction**, which ultimately protects their investment.

Business Model and Sustainability

To turn Buildwise AI into a sustainable product, we propose several **revenue models**:

- **B2B SaaS for Property Firms:** Real estate agencies and property management companies can subscribe to Buildwise AI as a service. We'd offer tiered plans based on number of users or properties. This provides them an out-of-the-box AI assistant branded for their company. The value (time saved, better client conversion, reduced legal costs) justifies a recurring subscription fee.

- **Premium Tenant Services:** While a basic version could be free for tenants (perhaps sponsored by city housing initiatives or as a loss-leader), a premium tier could offer advanced features: e.g. personalized lease review by AI with a certified lawyer's oversight, or priority access to human experts for complex issues. This could be a monthly subscription at an affordable rate, giving renters an *"AI + human" concierge.

- **Lead Generation and Partnerships:** Buildwise AI can recommend services when appropriate – for example, if during a chat the user needs renters insurance or a moving company, we could integrate with providers and earn affiliate commissions (in a user-transparent way). Similarly, partnerships with insurance companies or legal firms could see our AI referring cases to them and sharing revenue.

- **Enterprise Data & Analytics:** Aggregated, anonymized data from tenant interactions can be invaluable to larger landlords and city planners. We could offer a dashboard product that shows trends (e.g. which neighborhoods have most complaints about heating, or what lease terms cause most confusion). This insight could be sold to real estate investment firms or government agencies to inform policy, **so long as privacy is protected**.

Our aim is to align the revenue model with delivering real value. Happy tenants and efficient landlords mean more word-of-mouth and trust in the platform, driving adoption.

Prototype Implementation and Demo Outline

We built a working prototype that demonstrates the core functionality. Here's how it works and how you can try it:

- **Setup:** After pulling the repository, you can run `uvicorn main:app` to start the FastAPI server (all dependencies are in `requirements.txt`). Load some initial data by calling `/upload_docs` – for the demo, we include a sample NYC residential lease PDF and an NYC tenant rights FAQ PDF in the `data/` folder. These will be indexed for RAG.

- **Creating a User:** Use the `/crm/create_user` endpoint to create a user profile for the demo. For example:

```
POST /crm/create_user
{
  "name": "Alice Renter",
  "email": "alice@example.com",
  "preferences": {"budget": 3000, "pets": true, "neighborhood": "Brooklyn"}
}
```

  You'll get back a `user_id` (e.g. `abc123`). This ID will be used in subsequent calls to tie Alice's conversations and data to her profile.

- **Chatting with the Assistant:** Now, start asking questions via the `/chat` endpoint. For example:

```
POST /chat
{
  "user_id": "abc123",
  "message": "I'm looking at 123 Maple St #5B. Is it a good apartment for
me?"
}
```

  Behind the scenes, Buildwise AI will:

  1. Parse the question and recognize it's a Pre-Lease stage query about evaluating an apartment.

2. **Property Info Agent** retrieves data on *123 Maple St* (e.g. year built, any recent violations).
3. **Env Risk Agent** checks if that address is in a known flood zone or subject to LL97 emissions rules.
4. **Lease Agent** looks for any uploaded lease or if not, may provide general lease insights (or request one).
5. **Decision Agent** synthesizes a helpful answer. The API responds with a JSON containing the assistant's reply, which might say: *"The apartment meets 8/10 of your preferences (it's within budget and pet-friendly). The building is in a Flood Zone 2 area, so consider renter's insurance. Also, it's a pre-war building – maintenance records show past plumbing issues. Lease-wise, nothing stands out as unusual. Overall, I'd rate it a good fit, but you may want to ask about recent renovations."* (Plus, the message will include references to sources if applicable.)

- **Continuing the Conversation:** You can then ask follow-ups like *"What does the lease say about pets?"* – the Lease Agent will pull the pet policy clause from the uploaded lease PDF and the assistant will answer with that specific info. All this happens in one thread identified by `user_id "abc123"`, so context is preserved (the AI knows we're still talking about that apartment/lease). The CRM agent is logging this, and if you check `/crm/conversations/abc123`, you'll see the messages and how they're tagged (e.g. the first question was tagged `pre-lease` and `evaluation`).

- **Handling During/Post Lease:** Our prototype also includes examples for later stages. For instance, after "Alice" signs a lease, she can ask *"My heat isn't working, what can I do?"* – the assistant will recognize this as a living stage (maintenance issue). It will retrieve the NYC housing code on heating requirements (via RAG) and answer that the landlord is required to provide heat, advising her to notify management in writing and possibly call 311 if not resolved in 24 hours, etc. We even implemented a small logic where if the user says "The landlord agreed to fix it by tomorrow," the CRM can mark that issue as *Resolved*. Our `/crm/update_user` can be used to update a "case status" field.

- **After Lease example:** If Alice moves out, she might say *"I left the apartment in good condition, but my landlord won't return my deposit. What now?"* The assistant will pull info on security deposit laws from the knowledge base and guide her (e.g. *"In NYC, landlords must return deposits within 14 days or provide an itemized deduction list. You can send a certified demand letter. Shall I draft one for you?"*). While we haven't fully automated letter drafting, our design allows using the LLM to generate such letters.

- **Logging and Monitoring:** Every agent's action is logged (for debugging and audit). If you run the app with `DEBUG` mode, you'll see in the console the sequence of agents invoked and any tool calls (like external API hits for data). This transparency helped us fine-tune agent prompts and ensure reliable answers.

Note: Given hackathon time constraints, not all features are 100% complete – some parts are simulated or use sample data. However, the blueprint is in place for each module, and we've prioritized an **end-to-end demo** that shows the full journey (you can actually go from searching an apartment to asking move-out questions in one continuous conversation!). The README includes instructions for setting up API keys (for OpenAI, etc.) and how to run the sample queries.

Why Buildwise AI is Unique

In summary, Buildwise AI stands out because it **combines deep domain expertise with an intuitive conversational interface**. Unlike generic chatbots, it is **purpose-built for real estate leasing**, using multi-agent AI to handle diverse tasks: document analysis, compliance checks, environmental data retrieval, and more. This specialization is what makes the assistant both intelligent and trustworthy – similar multi-agent approaches in lease management have shown significant reductions in manual work and errors [4]. We took inspiration from those successes and tailored it to everyday renters as well as professionals.

Moreover, we focused on creating a **lovable user experience**. The assistant isn't just smart; it's personable and proactive. It remembers your name, your needs, and follows up on issues. We believe this human-centric touch will delight users and build trust in the AI. By capturing the entire lease lifecycle in the CRM, Buildwise can even **anticipate needs** (for example, reminding a tenant a month before their lease ends about renewal options or providing comps if they plan to move).

Finally, Buildwise AI aligns with the hackathon's goal of pushing the envelope of AI tech. We integrated **GPT-4 (Vision + NLP)**, **RAG**, and a custom CRM in a modern API-driven application [20] [21]. The project demonstrates not only technical proficiency (with a clear architecture and working code) but also a meaningful real-world application that can be expanded into a viable product.

We're excited about the future of Buildwise AI. With more time, we envision adding a simple front-end UI (a chat web app), expanding the knowledge base (e.g. integrating actual NYC open data feeds in real-time), and enhancing each agent (perhaps using specialized models for certain tasks, like a legal BERT model for clause classification). We'd also conduct user testing with renters and property managers to refine the conversational style and ensure the AI's advice is legally sound and helpful.

In conclusion, Buildwise AI transforms the chaotic rental process into a guided experience. It empowers individuals with information and frees professionals from tedious tasks. Our multi-agent conversational CRM platform is not just a hackathon project – it's a step toward smarter, safer, and more transparent renting for everyone. Let's build a future where AI helps you **live wiser** in every building – that's the vision of Buildwise AI.

---

[1] [2] [3] [4] [8] [9] Multi-Agent AI Systems: Orchestrating AI Workflows
https://www.v7labs.com/blog/multi-agent-ai

[5] Local Law 97 - Urban Green Council
https://www.urbangreencouncil.org/what-we-do/driving-innovative-policy/ll97/

[6] [7] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] Multi-Agentic Conversational AI System.pdf
file://file-JR7n6WiuhqQW9TmPSG9Jfd