Senior Python Developer Test (K.Kirati) - Feb 2024

Test Duration: 2 weeks

Thank you for the recent discussion. I'm genuinely excited to work with you. This test represents a slice of AltoTech's mission to harness IoT in the building/hotel sector. While there's a clear framework, innovative approaches are always appreciated. Should you have any questions, please reach out. Let's get started!

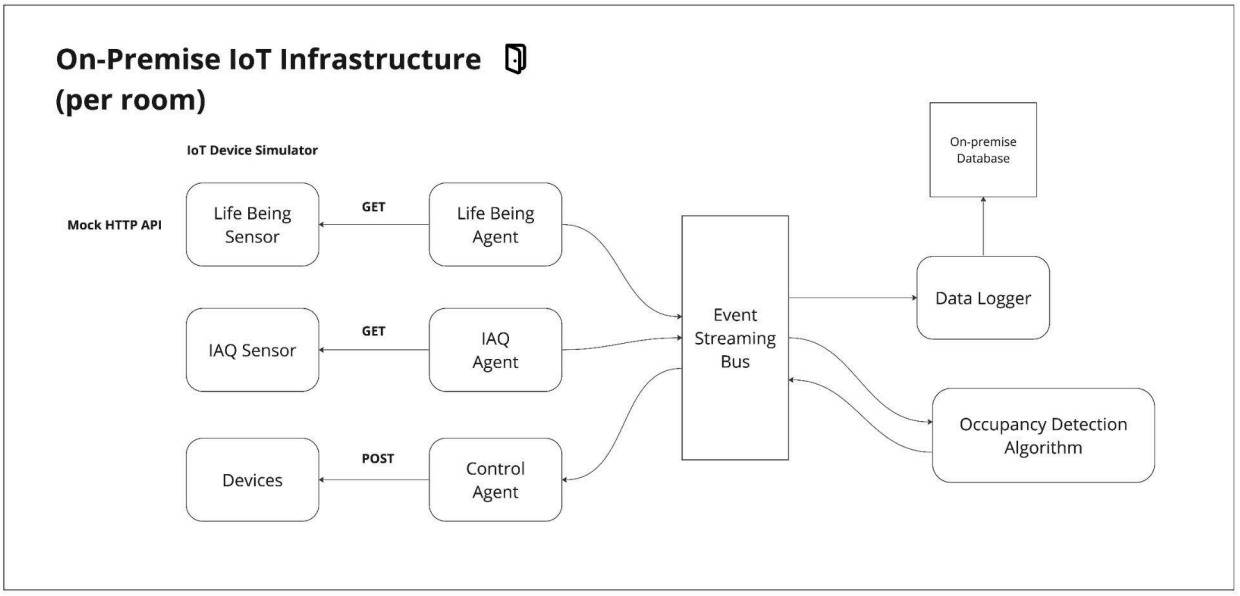
Objective: Create a simplified version of our Smart Hotel & Chain management system with the following goals:

- Reduce electricity consumption.
- Offer transparent data on hotel room usage.
- Ease hotel managers to manage multiple hotels in a single platform.
- Enhance the hotel's guest experience with the Large Language Model (LLM) interface.

How?

- **IoT Integration in Rooms**: Equip hotel rooms with:
    - Life Being Sensor: Detects motion.
    - IAQ Sensor: Measures indoor environment data.
    - Controller: Manages air conditioning and lighting.
- **Centralized hotels management**: Using data from each hotel, enables users to access comprehensive information through a unified platform. While data from each source might be in a different name and format, Ontology standards such as Project Haystack or Brick Schema can facilitate interoperability and data normalization, ensuring that information from diverse systems can be integrated and utilized efficiently.
- **Energy Conservation**: Using data from sensors, adjust settings to conserve energy, such as setting AC to 26-27°C when no guest is detected.
- **Enhance Guest Experience**: Allow guests to access room data and control devices via a web app on their mobile devices. To simplify the user interaction and enhance their experience, a chatbot-like interface is needed.

1. On-premise IoT Infrastructure (1 hotel):



- **IoT Device Simulation:** Use Python (or any language/tool of your choice) to Mock data points that simulate the IoT devices. These simulators will read data from the provided CSV file and make them available via an HTTP API.

  IoT Data Description:

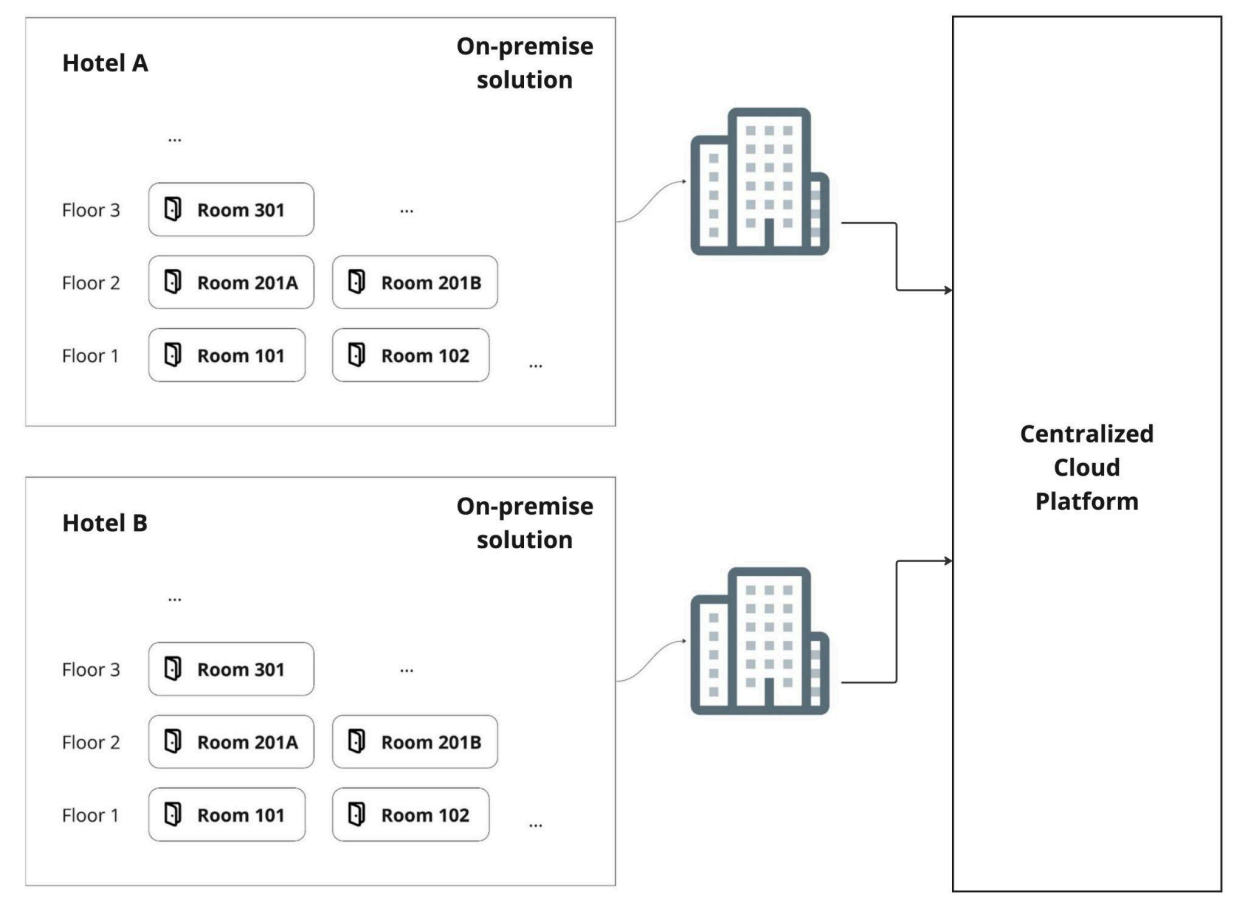
  - **Life Being Sensor:**
    - Datapoints: presence_state, sensitivity, online_status.
  - **IAQ Sensor:**
    - Datapoints: noise, co2, pm25, humidity, temperature, illuminance, online_status, device_status.
- **Agent Creation:** Design an agent to fetch data from the IoT device every 5 seconds via HTTP GET request.
- **Event Streaming:** The agent should publish messages to an event streaming bus. You can use any tool or platform of your choice for the event streaming bus, for instance, MQTT, Redis Streams, or Apache Kafka.
- **Data Logger & Occupancy Detection Agents:** These agents will listen to relevant topics from the bus. The Data Logger writes to an on-premise database, while the Occupancy Detection Agent determines device control decisions.
- **Cloud Database Strategy:** Implement a method to transfer these messages to a centralized cloud database.

Note: You're free to suggest improvements or alternative strategies.

2. IoT Infrastructure Deployment (1 hotel):

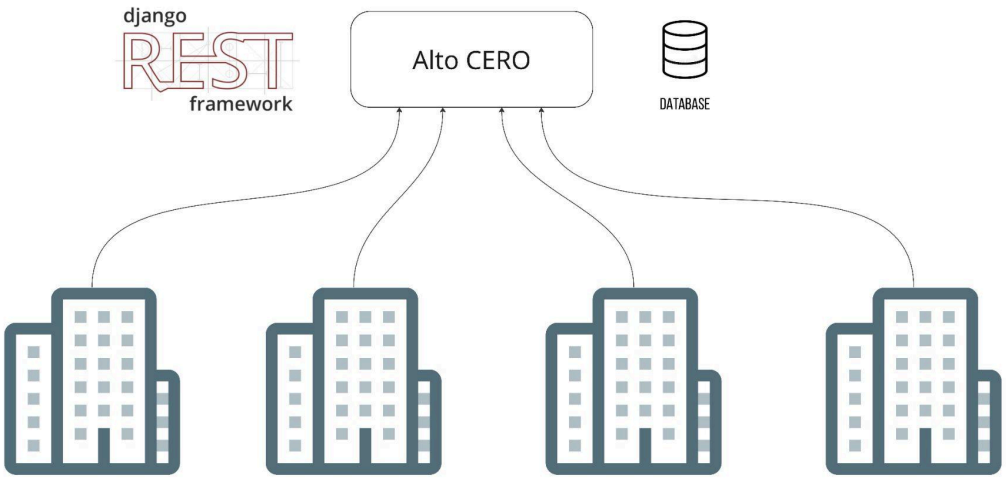
- **Ease of Deployment:** the infrastructure should be able to deploy easily with minimal effort like a simple bash script that can be run by a non-tech person. Tips: Write a bash script that uses Docker to set up and run the entire infrastructure. Docker-compose can be used to orchestrate multiple containers for different services.
- **Scaling:** The system should be scalable, allowing simulation of multiple rooms, floors, and hotels. This allows our software to scale to multiple sites with our partner system integrators to deploy on-site instead of our engineers.
- **Simulate:** Deploy the infrastructure to simulate multiple rooms, floors, and hotels in your local machine. Please contact me if there is any computational concern.

3. Cloud Infrastructure concept (support multiple hotels):



With the on-premise IoT infrastructure in place, the next step is to migrate it to a cloud platform. Users should have quick access to the IoT data. This calls for a robust

Backend service equipped with APIs to fetch this data. Given that our solution is destined for numerous hotels, each with its own set of floors and rooms, the database relations need careful crafting. The rooms and floors, with identifiers like 101, 102, 201A, and floor 3, should be easily distinguishable. Additionally, APIs to manage devices in each room are a must. Here are the critical APIs to be provided:



4. Data Ontology Standards on Multiple Property for Interoperability:

To demonstrate smart city solutions that are required to manage multiple properties at once, you need to implement Brick Schema ontology standards across the property management system to ensure seamless integration and access to information across multiple buildings. This involves structuring data according to a predefined model that simplifies the understanding of building operations and the relationships between different entities (sites, floors, rooms, devices, and datapoints).

Applying this ontology standard not only facilitates a unified platform for current operational needs but also establishes a foundation that enables future 3rd-party developers to build additional functionalities or applications on top of our existing platform, enhancing its capabilities and versatility for smart city integrations.

Tasks:

- **Brick Schema Integration**: Integrate Brick Schema ontology standards to define and relate the data structure for sites, floors, rooms, devices, and their datapoints. This ensures a standardized approach to managing and accessing IoT device data across multiple properties.
    - source: https://brickschema.org/

- **Graph Database Implementation**: Utilize a graph relational database to model the hierarchy and relationships of site > floor > room > device > datapoints. This database should effectively represent the connections between each entity, allowing for intuitive querying and data access.

- **Ontology Mapping**: For each property (hotel), apply the Brick Schema to map all IoT devices and their datapoints. This includes:
    - Defining each site, floor, room, and device within the Brick Schema framework.
    - Associating each datapoint with its corresponding device and room, ensuring that data is easily navigable and understandable across different levels of the property hierarchy.

- **Backend Service & APIs:** This service should provide APIs to fetch and control IoT data. Design the database to accommodate multiple hotels, floors, and rooms.

  Essential APIs include:
    - url/hotels/: Retrieve all hotels.

- url/hotels/<hotel_id>/floors/: Access floors in a hotel.
- url/floors/<floor_id>/rooms/: List rooms on a floor.
- url/rooms/<room_id>/data/: Get IoT data for a room.
- url/rooms/<room_id>/data/life_being/: Life Being sensor data.
- url/rooms/<room_id>/data/iaq/: IAQ sensor data.
- Device control API (Tip: POST with a payload structure)

Execute these scenarios to showcase the benefits of graph-relational and ontology implementations:

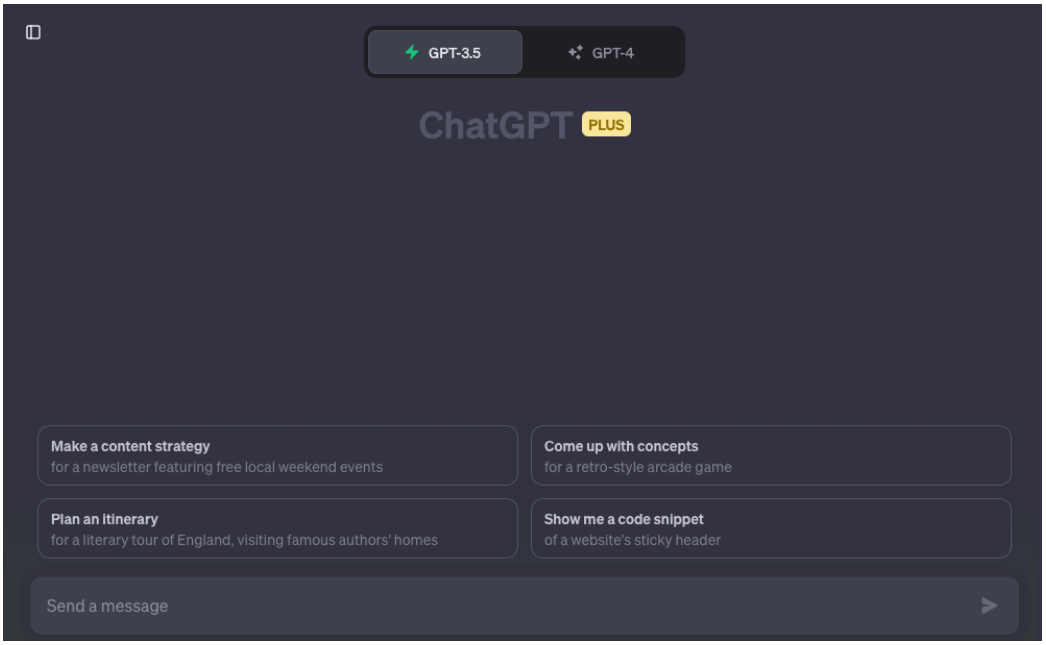
- "Retrieve average room temperature from all room in Hotel A's floor 1"
- "Retrieve all room name in Hotel B that has CO2 level more than 800 ppm"
- "Retrieve the count of rooms occupied in Hotel A over a 24hr period", assuming rooms with a CO2 level above 800 ppm indicate human presence and are therefore considered occupied."

Deliverables:

- A graph relational database schema that reflects the site > floor > room > device > datapoints hierarchy according to the Brick Schema ontology.
- Modified Backend Service & APIs to support ontology-based data access and device control.
- Testing usecases to validate the implementation of the Brick Schema ontology.
- Documentation on the ontology application and system architecture.

To summarize, this setup will utilize a graph database for managing relational data alongside time-series databases for archiving IoT data.

(optional) 5. LLM Interface for Smart Hotel:



- **Chatbot-like Interface**: The system should have an easy-to-use interface for hotel guests to interact with IoT devices within their room through Text or Image or Voice interactions. (you can use UI framework ex. Gradio)
- **Smart Functionality**: The guest interface should provide smart functionality to enhance the guest experience. Essential features include:
    1. Access real-time and historical IoT sensor data
    2. Control IoT devices (ex. Air Conditioner, Television, etc.)
- **OpenAI's Function Calling Feature**: One big challenge in the development of an LLM system is for the LLM to prepare the required parameters and call our custom Python functions correctly. OpenAI has just announced their new feature called 'Function Calling,' which allows this event to happen with ease. Please apply OpenAI's 'Function Calling' method to the system to seamlessly connect our user requests to our software, presenting Smart Hotel experiences to guests.
- **Generative AI Communication Logging**: In the development of AI systems, monitoring AI performance is a must. The system needs to store the chat history between users and AI. The system needs to provide a method for AI developers to find insights for improvement and for the product team to discover new customer needs.

Note: You're free to suggest improvements or alternative experiences for hotel guests.

6. Optional Tasks:

- **User Roles & Authorization:** You don't have to implement it but provide a strategy to introduce diverse user roles (e.g., guest, staff, admin) with distinct permissions and access levels.
- **Optimization Algorithm:** You don't have to implement it but provide an logic or main idea on optimization algorithm within the Occupancy Detection Agent. Alternatively, lay out a strategy enabling data scientists to enhance this algorithm in the future.

Guidelines:

- **Tools:** While Django, Docker, and GitHub are mandatory, you have freedom in the choice of tools for the IoT device simulation, agent creation, and event streaming.
- **Submission:**
  - **GitHub Repository:** This should house all code. The README must contain clear instructions for reproducing your setup.
  - **Github Wiki:**
    - System Architecture: Visual representation of connectivity between modules.
    - Flow Charts: Visual representation of processes.
    - Reasoning: Justify the tools and methodologies you adopted.
    - API & Technical Documentation: Detailed explanation of your APIs and technicalities.
    - ER Diagram: Showcase the database relationships.
    - Provide a demo in the form of screenshots or videos.
  - Feel free to use any presentation material. We expected a real DEMO.

Remember, at AltoTech, collaboration is key. Consider me as your teammate and don't hesitate to reach out.

We recognize the depth of this task, but it mirrors the challenges and rewards of working at AltoTech. It's not just about your current expertise but also your adaptability and learning pace. Your efforts will resonate with AltoTech's mission and enhance your professional journey. Good Luck !!

Resources:

- Brick Schema Ontology: https://brickschema.org/
- OpenAI documentation: https://platform.openai.com/docs/introduction
- OpenAI's Function Calling: https://openai.com/blog/function-calling-and-other-api-updates
- Youtube on OpenAI's Function Calling: https://www.youtube.com/watch?v=0lOSvOoF2to
- Sample script on OpenAI's Function Calling: https://github.com/Sentdex/ChatGPT-API-Basics/blob/main/function_calling.ipynb
- Gradio document: https://www.gradio.app/docs/interface