Technical Challenge Spot2 (IC FullStack)

En Spot2, una prop-tech pionera en el sector del real estate comercial, reconocemos que la innovación constante es el pilar de nuestro éxito. Esta dedicación a la vanguardia tecnológica nos exige no solo diseñar sistemas que escalen eficientemente y cumplan con los más altos estándares de calidad de código, sino también implementar soluciones que reflejen nuestra excelencia en la ejecución y nuestras sólidas prácticas de programación. Estamos buscando individuos apasionados por la tecnología que compartan nuestra dedicación a estos principios, para que juntos podamos continuar liderando la innovación en el mercado del real estate comercial.

Challenge (Python)

Design and implement a **Conversational AI Web Application** that uses an LLM (Large Language Model) to parse user input and collect specific information fields. The system should continuously interact with the user until all required fields are filled out. A streamlit template will be provided to as the main user interface to the application you build.

Tiempo de desarrollo: 4-5 horas aprox.
Tiempo límite de entrega: 1 semana

Requirements

Backend (Python):

- **API Handling**:
    - Well define request and response schemas
    - Ability to justify on whether a stateless or stateful design for the fields is required
    - Extensible architecture to ensure high scalability
- **Prompt Design and LLM:**
    - Implement an LLM-based backend capable of parsing user messages and extracting structured information.
    - Continuously prompt the user until all required fields are collected in the response

---

Required Fields:

1. **Budget** (e.g., "I have a budget of 20,000 USD")
2. **Total Size Requirement** (e.g., "I need 500m²")
3. **Real Estate Type** (e.g., "I am looking for an office space")
4. **City** (e.g., "I want a property in Mexico City")

Additional Fields:

- The user should be able to **extend the list of fields** if they find other useful information to provide.
- These extra fields should be stored and acknowledged by the chatbot.
- Validation of fields is optional but encouraged

Testing:

- **Unit and Integration Tests:**
  - Ensure the backend correctly identifies and stores all required fields.
  - Test interactions with different types of inputs.

Security:

- Ensure input handling is robust against attacks such as prompt injection.
- The system should not accept potentially harmful or irrelevant inputs.

Performance and Scalability:

- The solution should be designed to scale with increasing user interaction.
- Justify the choice of state management and database design to store the information

Deployment:

- Provide detailed instructions for deploying the application in a production environment.
- Include any scripts or configuration files necessary for setup.
- **Bonus:** Implement CI/CD pipelines using tools like GitHub Actions or GitLab CI/CD.

Documentation:

- Include a **README file** explaining how to install, configure, and run the application locally and in the cloud.
- Provide a brief explanation of your architectural choices and how they meet the requirements.

Example User Flow:

1. **User:** "I need an office in Mexico City."
2. **Bot:** "What's your budget and total size requirement for the office?"
3. **User:** "I can spend 20,000 USD for 500m^2."
4. **Bot:** "Thank you for filling out all the forms."

Evaluation Criteria

- **Accuracy:** The chatbot accurately identifies and collects all required fields.
- **Scalability:** The architecture supports efficient scaling.
- **User Experience:** The chatbot maintains a friendly and effective interaction flow.
- **Documentation:** Clear and detailed explanation of the solution.
- **Innovation:** Creative use of LLMs and user interface design.