# Splendom Suites Group

Real Estate Assistant Chatbot for Splendom Suites

Gonzalo Torras Serrano

## Table of Contents

## List of Figures

## List of Tables

# 1. Project Description

Splendom Suites provides specialized corporate housing and extended-stay accommodation for executives in major European cities such as Vienna, Madrid, and Barcelona. The main engineering challenge was to build an automated assistant capable of handling complex B2B queries involving specific amenities, location proximity, and apartment types. This system needed to operate without human intervention while strictly adhering to data privacy protocols and running on resource-constrained hardware, specifically a standard VPS with 8GB of RAM.

The main objective was to develop a functional chatbot for internal use, the initial prototype, designated as V1, relied on a monolithic architecture. It consolidated data from over 400 properties into a static JSON structure and utilized a non-private Llama3-70B model for feature extraction and response generation. While this approach was functional, it faced significant bottlenecks including high latency and frequent hallucinations where the model would invent amenities and user intentions. Additionally, the data ingestion pipeline depended on fragile regular expressions to parse weekly PDF updates, making maintenance very intensive.

To address these limitations, the project evolved into V2, a modular architecture centered on specialized Small Language Models. The core logic was decoupled into a custom NLU for intent classification and a NER system for entity extraction, both based on RoBERTa-base. This pipeline was trained using a mixture of an already existing dataset, made in V1 and synthetic data made by Llama3-70B and subsequently manually reviewed for quality and consistency.

The transition made a production-grade system that achieves an F1-score of 0.9776 on intent classification and approximately 0.94 on entity extraction. By shifting from a generalist LLM to optimized components, the system now runs inference in milliseconds on a CPU. This architecture eliminates hallucinations by grounding answers in deterministic logic and ensures full on-premise privacy without reliance on external cloud APIs.

# 2. Introduction & Business Context

## 2.1 Splendom Suites

Splendom Suites is a specialized provider of corporate housing and extended-stay accommodation across Europe, operating in major capitals like Madrid and Vienna as well as secondary business hubs. Their business model focuses on B2B clients, relocation agencies, and executives who require premium amenities and workspaces for stays ranging from one month to several years.

## 2.2 Problem Statement

The company's data management was highly fragmented. To answer a single client query about apartment specifics (such as amenities or square footage), agents had to manually search through over 30 different Word documents separated by country. Simultaneously, search for critical operational data like internal reference codes, pricing, and contact details were isolated in separate Excel spreadsheets. This manual workflow made cross-referencing information slow and easy for human error, as employees had to constantly switch between static files to provide complete answers.

## 2.3 Proposed Solution

To resolve this inefficiency, the initial proposal was to build a conversational assistant that would unify these disparate data sources into a single interface. At the start, the priority was rapid functionality, leading to a prototype reliant on external cloud-based LLMs. However, as the project moved toward deployment, the company acquired a shared Virtual Private Server (VPS) to host the chatbot alongside other internal tools. This introduced a strict new requirement, the solution had to run entirely locally on this limited hardware to eliminate recurring API costs and ensure data ownership. This request, combined with some lack of reliability in the cloud-based prototype, required a complete re-engineering of the system towards specialized, lightweight components like fine-tuned NER, NLU and NLG models capable of running efficiently on a CPU, while having great precision.

# 3. Phase 1: The Monolithic Approach

## 3.1 Data Engineering

The foundation of the first iteration was the creation of a unified Knowledge Base from the fragmented source files. This process involved a manual ingestion pipeline where information was segmented into several files. Firstly, all information regarding an apartment (apartment description which contained amenities, bedroom types, address and apartment name) was held in more than 20 different word files, with many apartments which no longer had a contract with Splendom Suites and many others with out-dated information. Moreover, they had various excels spreadsheets, which held information on contact information, price rates, internal codes, address and the apartment names which do currently hold an official contract with them.

Therefore, the first objective was to consolidate the word files into a single file and then clean the data to ensure that there no misleading information with the excel spreadsheet nor duplicate information. This meant to exclude outdated information, contact information and specific prices information, (the trustworthy data for this was already in the excel spreadsheet).

Afterwards, using Regular Expressions (Regex), positioning encoding and using an amenity dictionary with several synonyms, a massive JSON file was created. Then, after a long manual human revision, the JSON file was ready to segment information and give it to Llama3-70B. This approach was not scalable, because any change in the document format or any human error in formatting, led to human revision and with weekly apartments updates this led to a high labour-intensive work and unscalable. This was the final structure of the JSON file.

## 3.2 Architecture and Implementation

The first version of the project had employed a hybrid approach combining LLM-based feature extraction with deterministic Python filtering. Standard RAG (Retrieval-Augmented Generation) was deliberately avoided due to the high semantic overlap between property descriptions, since multiple units in the same building share nearly identical text regarding location and building amenities, vector retrieval lacked the granularity to reliably distinguish between specific apartment types within each apartment. Instead, the workflow relied on Llama3-70B acting as a reasoning engine and intent classification, where it routed queries in these three intentions:

- **Discover:** This involved the user wanting information for nearby apartments with specific bedroom types and amenities, such as *What apartments can I find near Calle Lagasca in Madrid with one bedroom, a gym and a pool?* This meant that the address had to be geocoded, to later compare latitude and longitude coordinates using Haversine distance calculation, to show apartments within a 2 km radius the nearest apartments (in case there were no apartments found a 5km radius was used). Then, the program filtered for the specific amenities and bedroom types, obtained by the feature extraction of Llama3-70B, with a long dictionary of synonyms.

- **Details:** This meant that the user had a specific question regarding a specific apartment, with queries such as *How many square feet does Splendom Atocha have?* The processing behind it, was to filter a specific apartment obtained with the feature extraction and then give the LLM the description of the desired apartment, to reason it and later answer the user query, this approach was vague and very costly.
- **Other:** This was a safety classifier to disregard any other sort of interaction the user wanted to ask, with queries such as *Can you help me with my math homework?* Preventing the bot from engaging in irrelevant conversations.

## 3.3 Evaluation

To validate the V1 prototype, a pilot test was conducted with 5 employees generating exactly 130 diverse queries. The system achieved a global success rate of 74% (96 queries), while the remaining 26% (34 queries) failed to produce a valid response. Moreover, strategies to improve accuracy, such as decomposing queries into multiple smaller LLM calls (Chain-of-Thought), were discarded because they exponentially increased latency and API costs, which were already bottlenecks with a single robust prompt.

| Failure Category | Percentage of Total | Count | Detailed Breakdown |
|---|---|---|---|
| Feature Extraction | 20.8% | 27 | Hallucinations Only (11): Tagging landmarks and other irrelevant as amenities (such as amenities). |
| | | | Location Only (6): Failed address parsing with missing cities or taking prepositions as part of the address. |
| | | | Mixed Errors (10): Simultaneous failure in both location parsing and amenity hallucination. |
| Intent Classification | 4.6% | 6 | Failed to distinguish *Discovery*, *Details* or *Other*, especially with long prompts. |
| Bedroom Extraction | 0.8% | 1 | A single isolated failure in detecting room count (confusing it as a real amenity instead of bedroom type). |
| Formatting Issues | 3.8% | 5 | JSON format inconsistencies within the 222 successful queries, not crucial error, but still an inconvenience. |

*Table 1: Evaluation of V1 Llama3-70B*

The performance analysis of the V1 prototype based on the 130-query sample reveals a global Accuracy of 73.8%, reflecting the system's direct success rate. However, a deeper dive into data extraction quality shows a Precision of 82.1% and a Recall of 85.7%. These figures indicate that while the model is effective at capturing information from prompts, its reliability is undermined by a significant volume of 'mixed errors' and hallucinations. The resulting F1 Score of 83.8% confirms that the current monolithic approach is not robust enough for a production environment, further justifying the transition to the specialized Phase 2 architecture to mitigate these cascading failures.

# 4. Phase 2: Modular Engineering Approach

## 4.1 Data Engineering

The transition to a supervised learning approach required labelled datasets, which did not exist. For the Entity Recognition (NER) task, a semi-automated pipeline leveraged the existing knowledge base. The process began with the 400 real apartment descriptions from the phase 1. Since the amenities were known before from the consolidated JSON dataset, a Python script used the amenities from the JSON and find them in the text description, to measure the begin and end token for each amenity, so later the custom NER could learn certain patterns.

To ensure robustness, a massive Data Augmentation strategy was implemented using a synonym dictionary to programmatically replace canonical terms with diverse variations (swapping *gym* for fitness *centre*). This expanded the dataset to approximately 10,000 descriptions instead of 400, allowing the model to learn semantic context and tolerate misspellings, a significant upgrade over the brittle dictionary from the first version of the project.

This part ensured the correct classification of the apartment descriptions this data was real and not synthetic. However, regarding the user side for the NER, a dataset which classifies information into the different classes, doesn't exist (only the 130 used for the LLM but this won't make a strong foundation) and a synthetic generation, while possible, might alter and confuse the NER and even lower the metrics seen. Therefore, for this specific scenario is best to have real world data to avoid overfitting and bias, however, another option is to rely in Small Language Models alongside RAG to answer user queries, but that is yet to be determined in V3 of the project.

Parallel to this, the NLU intent classifier required a different dataset. Lacking historical logs, a Synthetic Data Generation pipeline was built using Llama3-70B. The model was prompted to generate diverse user queries for the three target intents (Discovery, Details, Other), injecting domain-specific terminology to ensure realism. This made over 1,200 examples, which were manually reviewed for quality. This volume proved sufficient, as the pre-trained RoBERTa architecture could rapidly converge on these distinct classification boundaries without requiring massive datasets.

## 4.2 Architecture and Implementation

The transition from V1 to V2 marked a fundamental shift from a monolithic to a modular architecture. While the legacy system relied on a single, resource-heavy "black box" model for all tasks, the new design fine-grains the conversational logic into specialized components, orchestrated via a lightweight Python Flask API for maximum efficiency and control.

The architecture integrates the spaCy framework for NLP tasks due to its efficiency for CPU inference, which is critical for the deployment environment. The pipeline consists of two primary fine-tuned Transformer models based on the RoBERTa-base architecture:

*NLU Router (Intent Classification)*

This component acts as the semantic entry point. It classifies user input into one of three strategic intents: *DISCOVERY* (search queries), *DETAILS* (specific property questions), or *OTHER* (out-of-scope). This routing layer ensures that only relevant queries trigger the database search logic, saving computational resources.

*NER Extractor (Named Entity Recognition)*

The NER is used to analyse all the word documents mentioned before, first, the relevant intent is identified, this model extracts structured entities from the natural language query. It is trained to recognize domain-specific entities such as the address (near Chamberí in Madrid), the specific amenity (gym, pool, parking...) and the bedroom type (studio, one bedroom, two bedroom...).

Crucially, the retrieval process was shifted from probabilistic generation to deterministic filtering. The Python backend receives the entities extracted by the NER model and queries the consolidated JSON inventory directly. This hybrid approach of probabilistic understanding (NLU/NER) combined with deterministic retrieval, eliminates the risk of hallucinations, ensuring that the chatbot never invents a property or amenity that does not exist.

## 4.3 Evaluation

*NLU (Intent Classification)*

The RoBERTa-based NLU router was trained on synthetic data and afterwards tested on the 130 generated test set made in V1 for the cloud based infrastructure. The model presented a flawless convergence on the synthetic test set, recording an F1-score of 1.00 across all three intent classes (DISCOVERY, DETAILS, OTHER).

As illustrated in Figure 1, the model exhibits perfect separability between user intentions. This result is attributed to the distinct semantic boundaries defined in the ontology (search queries vs. specific property questions vs. out-of-scope chats) and the consistency of the synthetic training data. In a production environment, this reliability ensures that users are effectively routed to the correct logic flow, preventing "search" logic from triggering on irrelevant inputs.

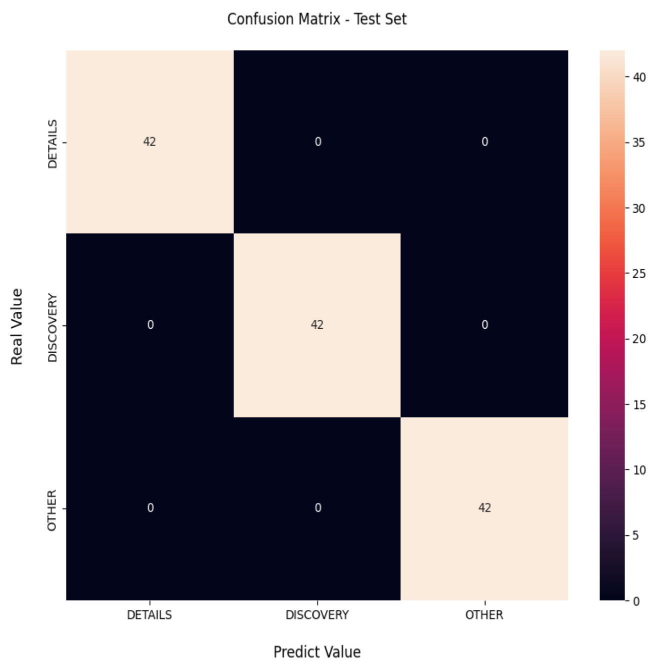

*Figure 1: Confusion Matrix for NLU*

However, it is important to acknowledge that training exclusively on synthetic data carries an important risk of overfitting to the generation patterns of the teacher model (Llama-3). Given the project's strict data ownership constraints and the zero-budget requirement for initial data labelling, synthetic generation was the only viable option using bootstrapping strategy. Even though it was tested on 130 user queries with flawless metrics, it is important to closely observe its behaviour over time. To mitigate this in the long term, the system implements a logging mechanism, real user queries are anonymized and stored to progressively replace the synthetic dataset with real-world distribution data, allowing for future re-training and validation on organic inputs.

### *NER Performance (Entity Extraction)*

The Named Entity Recognition model was trained and tested on real world data, giving it a huge advantage over the other models. The NER achieved a global F1-score of 0.956, demonstrating robust capabilities for structuring unstructured user queries. Table 2 highlights the performance for critical operational entities. (See

Appendix A for the complete metrics of all 56 entity classes).

A granular analysis reveals a strong correlation between entity importance and model accuracy. Operational entities required for core database retrieval (with each class standardized) achieving near-perfect scores. Lower performance in minority classes like BABY_COT (0.47) is attributed to semantic ambiguity and lower support in the training data.

| Entity Type | Category | Precision | Recall | F1-Score |
|---|---|---|---|---|
| ADDRESS | Location | 0.9907 | 1.0000 | 0.9953 |
| APARTMENT_NAME | Identifier | 1.0000 | 1.0000 | 1.0000 |
| TWO_BEDROOM | Unit Type | 0.9672 | 0.9833 | 0.9752 |
| INTERNET | Amenity | 0.9518 | 0.9875 | 0.9693 |
| PET_POLICY | Policy | 0.9231 | 0.9231 | 0.9231 |
| BABY_COT | Minority Class | 0.4444 | 0.5000 | 0.4706 |

*Table 2: Highlighted Evaluation of V2 NER*

# 5. Phase 3: Future Work & Roadmap

## 5.1 NLG Implementation

To complete the transition to a fully on-premise privacy-preserving architecture, the next phase focuses on the Natural Language Generation (NLG) layer. While the NLU and NER components now successfully structure the user's intent, the final response generation currently relies on rigid formats. To achieve true conversational fluidity and scalability without reverting to external cloud APIs, the system will implement a local Retrieval-Augmented Generation (RAG) pipeline, this RAG will answer the user’s specific apartment questions and also be of use when a user has a *Discover* intention, finding key apartments, filtering beforehand by distance to reduce error.

This approach avoids the brittleness of hard-coded dynamic templates, which fail to adapt to varied user query styles. Instead, the system will deploy a specialized Small Language Model (SLM), such as Phi-3 or a heavily quantized version of Llama-3-8B, directly on the VPS. Given the strict hardware constraint of 8GB RAM, these models will be utilized in 4-bit quantized formats (GGUF) to ensure the memory footprint remains under 4GB, allowing for efficient CPU inference.

In this workflow, the deterministic data retrieved by the NER (such as specific amenities or pricing from the JSON inventory) serves as the "context window" for the SLM. The model then generates a natural, human-like response based only on that retrieved context. This guarantees that the chatbot maintains the flexibility of a generative model while ensuring zero hallucinations and keeping all proprietary data within the company's secure infrastructure.

## 5.2 Advanced NLU and NER

While the current V2 architecture achieves near-perfect metrics on single-intent queries, real-world conversations often involve complex, compound requests that the current "router" cannot simultaneously process. For example, a user might ask, "I am looking for a flat in Vienna and I need to know if you accept credit cards." Currently, the system forces a binary choice (classifying this as either Discovery and Details), potentially ignoring half of the user's request.

To solve this, the next iteration of the NLU will migrate from a standard multi-class classification to a Multi-Label Classification architecture or a Sequence-to-Sequence approach. This will allow the model to predict multiple active intents (Discovery and Details) for a single input.

Regarding the NER, a quantized SLM will be integrated to handle conversational context by acting as a "query rewriter." Instead of complex state tracking, this model will merge the conversation history with the current input into a single, complete sentence (expanding "Which ones have a pool?" to "Show me apartments in Madrid with a pool"). This ensures the deterministic NER always receives fully contextualized data, preserving high extraction accuracy across multi-turn interactions.

## 6. Conclusion

The progression from Phase 1's monolithic LLM architecture to Phase 2's modular NLU/NER pipeline marks a fundamental engineering breakthrough for conversational AI in resource-constrained environments. The V1 prototype, while functional, suffered from high latency (>5s per query), frequent hallucinations, and unscalable data ingestion reliant on fragile regex patterns. In contrast, V2 delivers production-grade performance with F1-scores of 0.9776 for intent classification and 0.94 for entity extraction, running inference in milliseconds on CPU-only hardware.

This transformation validates a key principle: specialized, fine-tuned components systematically outperform generalist LLMs for well-defined business domains. By decoupling understanding (NLU/NER) from generation and grounding responses in deterministic database queries, the system eliminates the black-box unreliability of V1 while achieving full data privacy on Splendom Suites' 8GB VPS—without external APIs or cloud dependencies. The synthetic data pipeline and noise augmentation strategies further prove that high-quality labeled datasets can be engineered efficiently even without historical logs.

Phase 3 will complete this vision with local RAG-based NLG using quantized SLMs, offline optimization via "Night Shift" processing, and contextual query rewriting for true multi-turn conversations. This roadmap positions the chatbot as a scalable foundation capable of handling thousands of properties, serving as both a practical tool for Splendom Suites and a technical showcase of efficient, privacy-first AI engineering.

# Appendix A

| Entity Type | Precision | Recall | F1-Score |
|---|---|---|---|
| ADDRESS | 0.9907 | 1.0000 | 0.9953 |
| AIR_CONDITIONING | 1.0000 | 1.0000 | 1.0000 |
| AIRPORT_SHUTTLE | 1.0000 | 1.0000 | 1.0000 |
| APARTMENT_NAME | 1.0000 | 1.0000 | 1.0000 |
| BABY_COT | 0.4444 | 0.5000 | 0.4706 |
| BALCONY | 0.8913 | 0.9762 | 0.9318 |
| BAR | 0.8571 | 1.0000 | 0.9231 |
| BATHROOM_TOILETRIES | 1.0000 | 0.8611 | 0.9254 |
| BBQ | 1.0000 | 1.0000 | 1.0000 |
| BIKE_STORAGE | 1.0000 | 0.8571 | 0.9231 |
| BREAKFAST | 0.8000 | 0.6667 | 0.7273 |
| CCTV | 0.9000 | 1.0000 | 0.9474 |
| COFFEE_MACHINE | 0.9811 | 0.9811 | 0.9811 |
| CONCIERGE | 0.9333 | 1.0000 | 0.9655 |
| COWORKING_AREA | 1.0000 | 1.0000 | 1.0000 |
| DISHWASHER | 0.9697 | 1.0000 | 0.9846 |
| DRY_CLEANING | 0.8000 | 1.0000 | 0.8889 |
| ELEVATOR | 1.0000 | 1.0000 | 1.0000 |
| FIRST_AID_KIT | 1.0000 | 1.0000 | 1.0000 |
| FITNESS_CENTER | 1.0000 | 0.8750 | 0.9333 |
| FOUR_BEDROOM | 1.0000 | 1.0000 | 1.0000 |
| HAIR_DRYER | 1.0000 | 0.9655 | 0.9825 |
| HEATING | 0.9250 | 1.0000 | 0.9610 |
| HOUSEKEEPING | 1.0000 | 0.8750 | 0.9333 |
| INTERNET | 0.9518 | 0.9875 | 0.9693 |
| IRONING | 0.9394 | 1.0000 | 0.9688 |
| KETTLE | 1.0000 | 1.0000 | 1.0000 |
| KING_BED | *(No data)* | *(No data)* | *(No data)* |
| KITCHEN | 0.9881 | 0.9022 | 0.9432 |
| LAUNDRY_SERVICE | 0.8462 | 0.8800 | 0.8627 |
| LINEN_AND_TOWELS | 0.7692 | 0.7143 | 0.7407 |
| LOUNGE | 0.8000 | 0.8889 | 0.8421 |
| LUGGAGE_STORAGE | 1.0000 | 0.8750 | 0.9333 |
| MEETING_ROOM | 1.0000 | 1.0000 | 1.0000 |
| MICROWAVE | 0.9661 | 0.9661 | 0.9661 |
| NON_SMOKING | 1.0000 | 0.7500 | 0.8571 |
| ONE_BEDROOM | 0.9747 | 0.9625 | 0.9686 |
| OVEN | 0.9245 | 0.9608 | 0.9423 |
| PARKING_OPTION | 0.9149 | 0.9348 | 0.9247 |
| PET_POLICY | 0.9231 | 0.9231 | 0.9231 |
| PHOTOCOPYING_MACHINE | 1.0000 | 1.0000 | 1.0000 |

| | | | |
|---|---|---|---|
| PLAY_ROOM | 1.0000 | 0.6667 | 0.8000 |
| POOL | 1.0000 | 1.0000 | 1.0000 |
| RECEPTION_OPTION | 0.9524 | 0.9091 | 0.9302 |
| RESTAURANT | 1.0000 | 0.7500 | 0.8571 |
| ROOFTOP_TERRACE | 1.0000 | 0.8750 | 0.9333 |
| ROOM_SERVICE | 1.0000 | 1.0000 | 1.0000 |
| SAFE_BOX | 0.8889 | 0.8000 | 0.8421 |
| SMART_TV | 0.9080 | 0.9518 | 0.9294 |
| SOFA_BED | 1.0000 | 0.8636 | 0.9268 |
| STREAMING_SERVICE | 1.0000 | 0.6667 | 0.8000 |
| STUDIO | 1.0000 | 1.0000 | 1.0000 |
| TERRACE | 0.9444 | 0.8095 | 0.8718 |
| THREE_BEDROOM | 1.0000 | 0.9000 | 0.9474 |
| TOASTER | 1.0000 | 1.0000 | 1.0000 |
| TWO_BEDROOM | 0.9672 | 0.9833 | 0.9752 |
| VACUUM_CLEANER | 1.0000 | 1.0000 | 1.0000 |
| VITROCERAMIC_HOB | 0.8500 | 1.0000 | 0.9189 |
| WASHER_DRYER_SETUP | 0.9222 | 0.9326 | 0.9274 |
| WHEELCHAIR_ACCESS | 1.0000 | 1.0000 | 1.0000 |
| WORK_DESK | 1.0000 | 0.9412 | 0.9697 |

*Table 3: Total Evaluation of V2 NER*