The diagram represents a **High-Level Architecture** for **NLP (Natural Language Processing) Solutions for a Minority Language**. It contains four major components (Online Dictionary, Spell Checker, Corpus Collection, and Grammar Rule Base) as well as underlying technology choices.

---

1. SHARED SERVICE (Dictionary API)

- **Dictionary API** serves as a central service used by multiple modules like the **Spell Checker** and **Online Dictionary**. It handles the main functions of querying and managing dictionary data.
- **Pronunciation Storage** is connected to the **Dictionary API**, suggesting that this module can also handle pronunciation data for words in the dictionary.
- **Management Interface** is linked to the **Dictionary DB**, indicating that users can manage or update the dictionary content (such as words, definitions, grammar rules) through this interface.

---

2. ONLINE (Online Dictionary Module)

- **Dictionary Front-End**: This is the user interface for the dictionary service, allowing users to search and retrieve word information. It's connected to the **Dictionary DB** and **Pronunciation Storage**, meaning it retrieves both word details and their pronunciation information from the database.
- **Dictionary DB** stores all word entries, including their definitions, pronunciations, grammatical attributes, etc. It's accessible via the **Management Interface** for backend updates and content management.

---

3. SPELL (Spell Checker Module)

- **Spell Checker Front-End**: This interface allows users to input text and check for spelling errors. It communicates with the **Spell Checker Engine**.
- **Spell Checker Engine**: This component performs spell-checking by querying the **Dictionary API** to verify word correctness and suggest alternatives.

---

4. CORPUS (Corpus Collection Module)

- **Corpus Front-End**: This user-facing interface allows users to upload or search through corpora (collections of text/audio data). It interacts with the **Corpus DB**, which stores these corpora.
- **Upload Interface**: It facilitates the addition of new corpus data (texts, recordings, etc.) to the **Corpus DB**.

---

5. GRAMMAR RULE (Grammar Rule Module)

- **Grammar Rule Base Front-End**: Users interact with this interface to manage or query grammar rules.
- **Grammar Rule Engine**: This component applies the set grammar rules to validate or suggest corrections for sentence structures, agreement, etc. It interacts with the **Grammar Rules DB** to retrieve these rules.

---

6. TECH (Underlying Technologies)

- **FAST API**: A modern, high-performance web framework used for building the backend services of the system. It would likely power the **Dictionary API** and other REST services in the architecture.
- **Supabase (Postgres)**: A scalable database service using PostgreSQL, likely used for storing dictionary data, corpus data, and grammar rules.
- **Streamlit**: A front-end framework to quickly build interactive web applications, used for the **Front-End** components across modules.

Connections Between Components:

- **Spell Checker** communicates with the **Dictionary API** to verify words and offer corrections.
- **Corpus Front-End** and **Upload Interface** interact with the **Corpus DB** for storing and managing corpus data.
- **Grammar Rule Base** interacts with the **Grammar Rule Engine**, which processes grammar validation requests using data from the **Grammar Rules DB**.

Technology Stack Overview:

- **Streamlit**: Used for building lightweight web front-ends quickly and efficiently.
- **FAST API**: Manages RESTful communication between front-ends and back-end services.
- **Supabase (Postgres)**: A cloud-based relational database (PostgreSQL) providing storage for dictionary entries, grammar rules, and corpus metadata.

Summary of System Workflow:

1. **Online Dictionary** allows users to search for word meanings, which are fetched via the **Dictionary API** from the **Dictionary DB**.
2. **Spell Checker** uses the same **Dictionary API** to check and suggest word corrections.
3. **Corpus Collection** allows users to upload or search for linguistic resources, storing data in the **Corpus DB**.
4. **Grammar Rule Base** provides grammar validation via the **Grammar Rule Engine**, which applies rules from the **Grammar Rules DB**.

This architecture is modular and scalable, allowing for independent development, testing, and scaling of each component. Each module is designed to perform its tasks efficiently while interacting with shared services where needed.