Dope Dictionary

The AI Whisperer, JSON, and the Weekend Warrior

Andrew Berg, Jason Bluhm, Quinn Simmonds

Introduction

A fruit of verdant hue, or crimson bright, a globe of sweetness, tart and crisp delight.

A delightful sphere of concentrated sweetness, a miniature cosmos of sugars and acids, perfectly poised between the branches of gravity's embrace.

Tools and Technology

Now listen here, an "apple" in Django's world, that's a kinda fruit, see? Sweet, juicy, and maybe even a little bit tart, depending on the kind.

Google Gemini LLM

User

Django Backend

Java Spring Microservice

Nested HashMap Database

Application Programming Interface

Spring Rest Controller
Communication for Django Backend and Spring Microservice (HashMap)

Google Gemini API
Communication for Django Backend and Google Gemini

```
@PostMapping("/add")
public ResponseEntity<String> add(
        @RequestParam(required = true) String style,
        @RequestParam(required = true) String word,
        @RequestParam(required = true) String definition) {
```

```
@GetMapping("/get")
public ResponseEntity<String> get(
        @RequestParam(required = true) String style,
        @RequestParam(required = true) String word) {
```

Django Backend

```python
def get_definition_service(request):
    if request.method == "GET":
        try:
            style = request.GET.get("style", "").strip()
            word = request.GET.get("word", "").strip()

            if not style or not word:
                return render(request, "home.html", {"definition": "Please provide both style and word"})

            # URL encode parameters to handle special characters
            encoded_style = quote(style)
            encoded_word = quote(word)
            encoded_base_url = "http://localhost:8080/dictionary"
```


```python
# Try to get definition from Java service
response = requests.get(
    f"{encoded_base_url}/get",
    params={"style": encoded_style, "word": encoded_word},
    timeout=5
)
response.raise_for_status()
```

```python
# Store in Java HashTable micro-service
store_response = requests.post(
    f"{encoded_base_url}/add",
    params={
        "style": encoded_style,
        "word": encoded_word,
        "definition": quote(answer_llm)
    },
```

```python
return render(request, "home.html")
```

Bot Talk


```python
def get_definition_llm(style, word):
    # Get the directory containing the current file
    current_dir = Path(__file__).resolve().parent
    env_path = current_dir / '.env'

    # Load environment variables from .env file
    load_dotenv(env_path)

    # Load Google API key from .env file 🔑
    api_key = os.getenv('GOOGLE_API_KEY')
    if not api_key:
        return "Error: Google API key not configured. Please check your .env file."

    try:
        # Configure Gemini 💡
        genai.configure(api_key=api_key)

        # Initialize model with gemini-1.5-flash for faster responses 🚀
        gemini_model = genai.GenerativeModel('gemini-1.5-flash')

        # Prompt the LLM for the definition 💬
        prompt = (
            f"Define '{word}' in {style} style. Make it:, Safe for Work, Clear, "
            f"Fun yet informative, end with a period, and 1 to 2 sentences."
        )

        # configure gemini model request parameters 📝
        response = gemini_model.generate_content(
            prompt,
            generation_config={
                'temperature': 0.5,  # Lower temperature for faster, more focused responses
                'top_p': 0.8,        # Reduce sampling space
                'top_k': 30,         # Limit token selection
                'max_output_tokens': 40  # Limit response length
            }
        )

        return response.text # Return the definition
    except Exception as e:
        return f"Error generating definition: {str(e)}"
```

Data Structures & Algorithms Java Microservice


```java
@Service
public class DictionaryService implements Serializable {
    private static final long serialVersionUID = 1L;
    private final HashMap<String, HashMap<String, String>> dictionary;
    private static final String STORAGE_FILE = "hashmap.ser";

    public DictionaryService() {
        this.dictionary = new HashMap<>();
        loadFile();
    }
```

```java
public String getDefinition(String style, String word) {
    HashMap<String, String> styleMap = this.dictionary.get(style);
    return styleMap != null ? styleMap.get(word) : null;
}
```

```java
public void putDefinition(String style, String word, String definition) {
    HashMap<String, String> styleMap = this.dictionary.computeIfAbsent(style, k -> new HashMap<>())
    styleMap.put(word, definition);
}
```

Demo

Local Server Demo

Future Improvements

- Launch Publically
- Make Prettier
- Custom styles
- Monetize
- Retire on a Beach

Conclusion

- Stylized dictionary service
- Google Gemini AI for definition generation
- Django Backend for web app logic
- Java Spring Boot microservice for nested hash-map implementation