Humanize Web APIs, without the headache

By Team: Red
Old Dominion University
CS 410
Fall 2024

Table of Contents

Team Bios - 1/3

John Hicks

A part-time Computer Science Major at ODU, transfer student from Tidewater Community College (TCC) where he earned his Associate of Science with a specialization in Computer Science. John has been employed full-time in software development and IT roles during most of his time in school. John began his journey into software development when his parents' small business needed a website upgrade from Microsoft Frontpage to WordPress. On understanding WordPress's hook and filter mechanism, John's imagination was kindled in wondering what other ways of writing software there might be. That curiosity turned to flame and was formed into skill with the help of many friends, family, internet contributors, workplace mentors and school faculty.

Freddie Boateng

A Computer Science major with a minor in Cybersecurity. He is from Northern Virginia and currently working as a Cybersecurity Engineer with Zachary Piper Solutions. He strives to always improve and stay updated to the world of technology, enabling him to reach his goals

Team Bios - 2/3

Kobe Franssen

Full time Computer Science major at ODU while also working part time as System Administrator at the ODU Computer Science Consultant Group. Experienced in Java, Python, C++ and API handling such as with Discord Bots. Love working on cars and has 3 cats.

Diya Patel

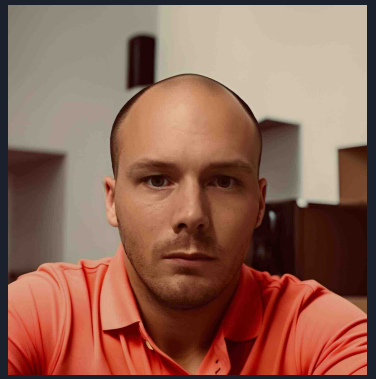
A Junior at ODU, pursuing a Bachelor's degree in Computer Science. She is interested in learning about the newest advancements in web development and artificial intelligence. She has an ongoing desire to take on new tasks and expand her skill set.

Andrew Bausas

I am a computer science major from Virginia Beach. I am to improve my skills and eventually use them to make games.

Team Bios - 3/3

Sean Baker



Sean's journey into computer science has been unconventional and spans both time and institutions. A transfer student from Piedmont Virginia Community College (PVCC), Sean earned his associate degree in computer science in 2016, but his tech journey began much earlier. At 14, he built his first WordPress site to supplement hsi allowance, which led to articles like "Ten reasons this IPhone will succeed". Since then, rather than pursuing a conventional corporate path, Sean has prioritized creativity and innovation, which has led him to work on projects that push technological boundaries, including contributing to self-driving car technologies with Edison2 and developing die cast automation software for VisiTrak Worldwide and Rockwell Automation. His self-taught, autodidactic learning approach has defined his career. Set to graduate this spring, Sean hopes to pursue a masters degree.

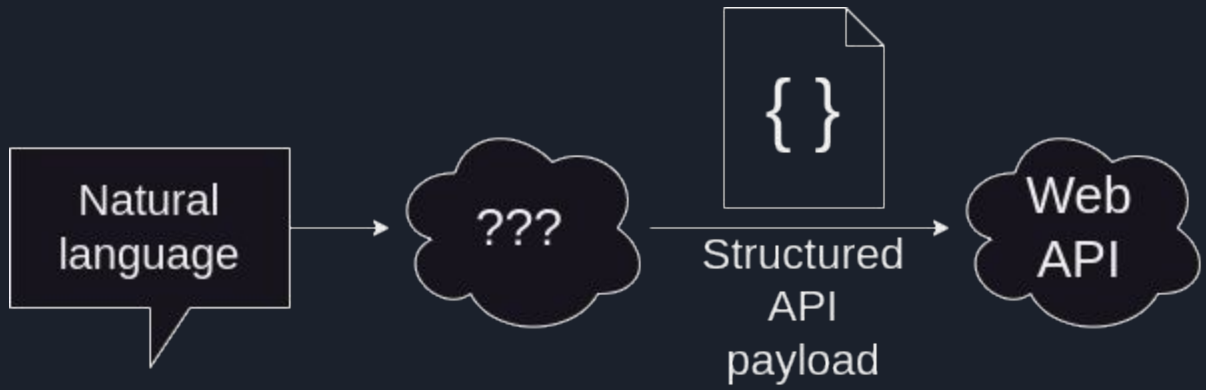
Chase Wallace

A Computer Science and Biomedical Sciences double major from Norfolk with a strong interest in neuroscience and artificial intelligence. He is always ready to learn new skills and broaden his horizons with challenging new projects.

Feasibility

Why talk with Web APIs? - The Societal Problem

- User interfaces don't speak the user's language, but users rely on apps to make things happen.
- Developers are motivated to add Natural Language Processing (NLP) features to their apps, but doing so is painstaking.
- Things happen in Web apps through Web Application Programming Interfaces (APIs).
- It is complex to turn natural language to structured data, which is what Web APIs must receive in order to work (Su et al., 2017).
- Open source contributors and researchers are attempting to use new Large Language Models (LLMs) to create Web API payloads (Zafin, 2023; Tool/Function Calling | LangChain, n.d), but there is not mature tooling in this emerging part of the market.



Why CueCode? - Problem Statement

- No LLM tools available for Web API payload generation focus on putting a human or deterministic business logic “in the loop” of payload generation.
- Enterprises and Software-as-a-Service (Saas) applications cannot afford to make every function call an LLM recommends for data entry or triggering actions; that is too high risk.
- The inability for current tools to manage the risks of LLM usage gates Enterprises and SaaS providers from developing AI applications while LLM technology matures.

=> **There’s demand for API call payload generation, but no simple, operationalized, and risk-aware tooling for it.**

CueCode will use this opportunity to commoditize the process of turning natural language into API payloads against existing existing Web APIs.

Elevator Pitch

- CueCode lets a Web application generate REST API calls from natural language with minutes of development time.
- A good OpenAPI specification and a few key questions are all CueCode needs to start generating REST API requests.
- CueCode can add AI features to your app without any backend code changes or specialized NLP or large language model (LLM) skills.
- This allows rapid development of natural language processing features, without having to risk taking humans or business rules out of the loop.

Quick example

- Example input: “I would like to book a hospital appointment on the 20th of October at 2pm for a medical checkup.”
- Output: a full REST API payload for creation of an appointment on the 20th of October, in the structured data format the REST API expects:
  - `POST https://the-appointment-app.com/api/v1/appointments/`
  - `{"client":{"last_name": "Davis", "first_name":"Patricia"}, date: “2024-10-20”, time: “1400” }`
- This output can be shown to the user, put through business logic, or sent immediately to the appointments REST API.
- The developer now has a choice about how to handle the suggested API payload.

Problem Characteristics - NLP/LLM challenges

Problems with current NLP/LLM processing for creating API calls:

- Require awareness of prompt engineering and other more complex AI techniques
  - => Time/money upskilling fullstack and Web developers.
- The NLP tools for generating API calls today are stand-alone programs and libraries that don't present a unified, opinionated solution.
  - => Developers are left building one-off solutions.
  - => Heavy boilerplate/in-house frameworks.
- Humans and application logic are kept out of the loop in other approaches; this is high-risk.

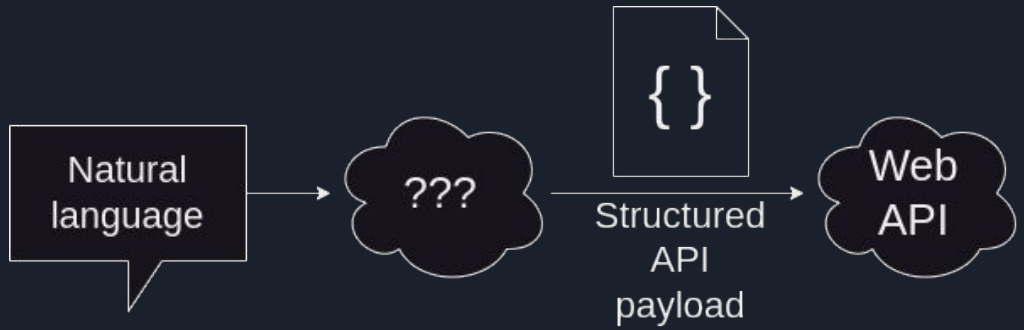
Problem Characteristics - use of API specs

- Since APIs are commonly described with specifications, why not use those?
  - (Keep a clean contract between system components.)
- OpenAPI is the leading industry standard way to describe REST (REpresentational State Transfer) APIs.
- However, there are no complete frameworks that leverage OpenAPI specifications when turning natural language to REST API calls.

Problem Characteristics - NLP/LLM challenges

Problems with current NLP/LLM processing for creating API calls:

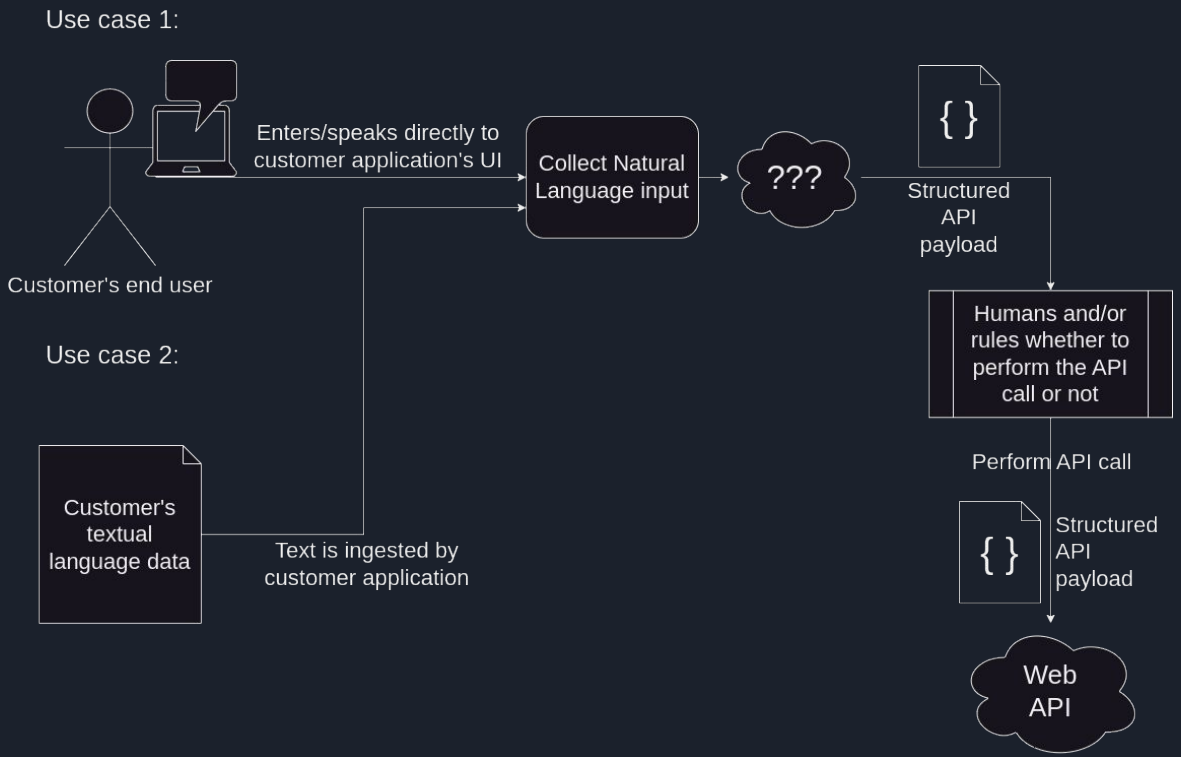
- Limiting Responses to fit an API Structure Is Difficult
- Lack of Understanding of Entity Relationships
- Absence of a Consistent Framework for Web Developers

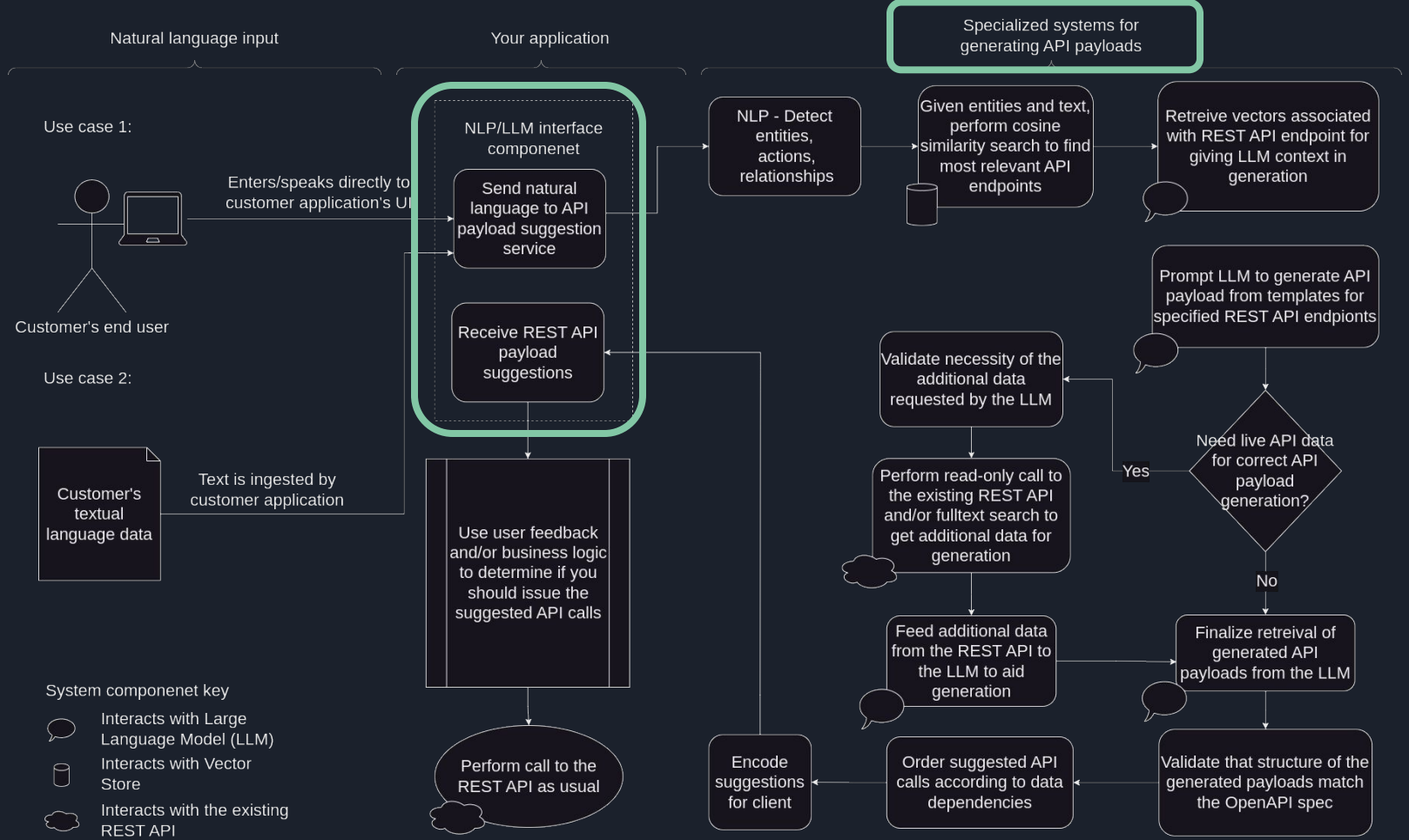


Current Process Flow

Two example use cases:

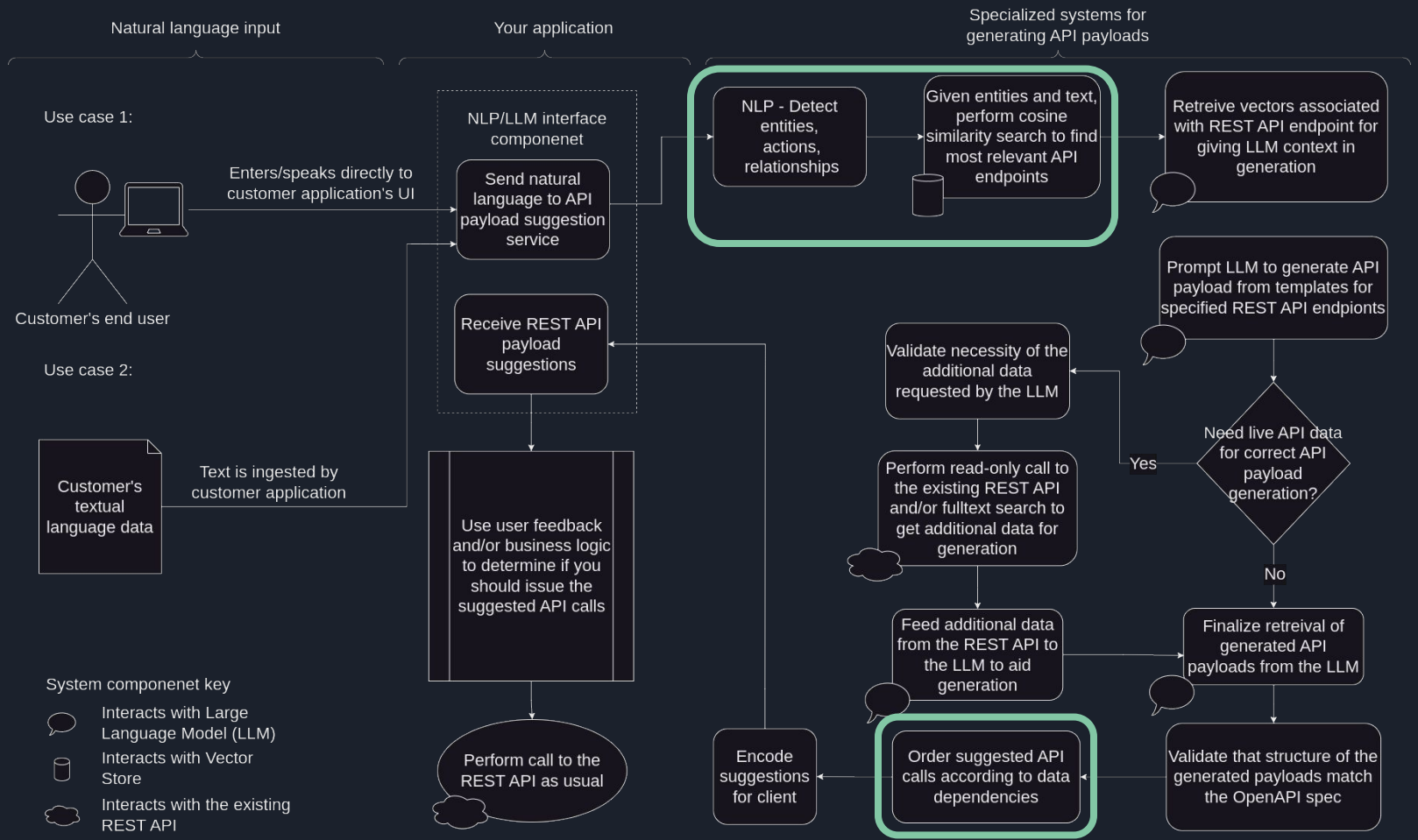
1. Interactive with your end user, validated by user after generation
2. Batch oriented, processed with business logic generation





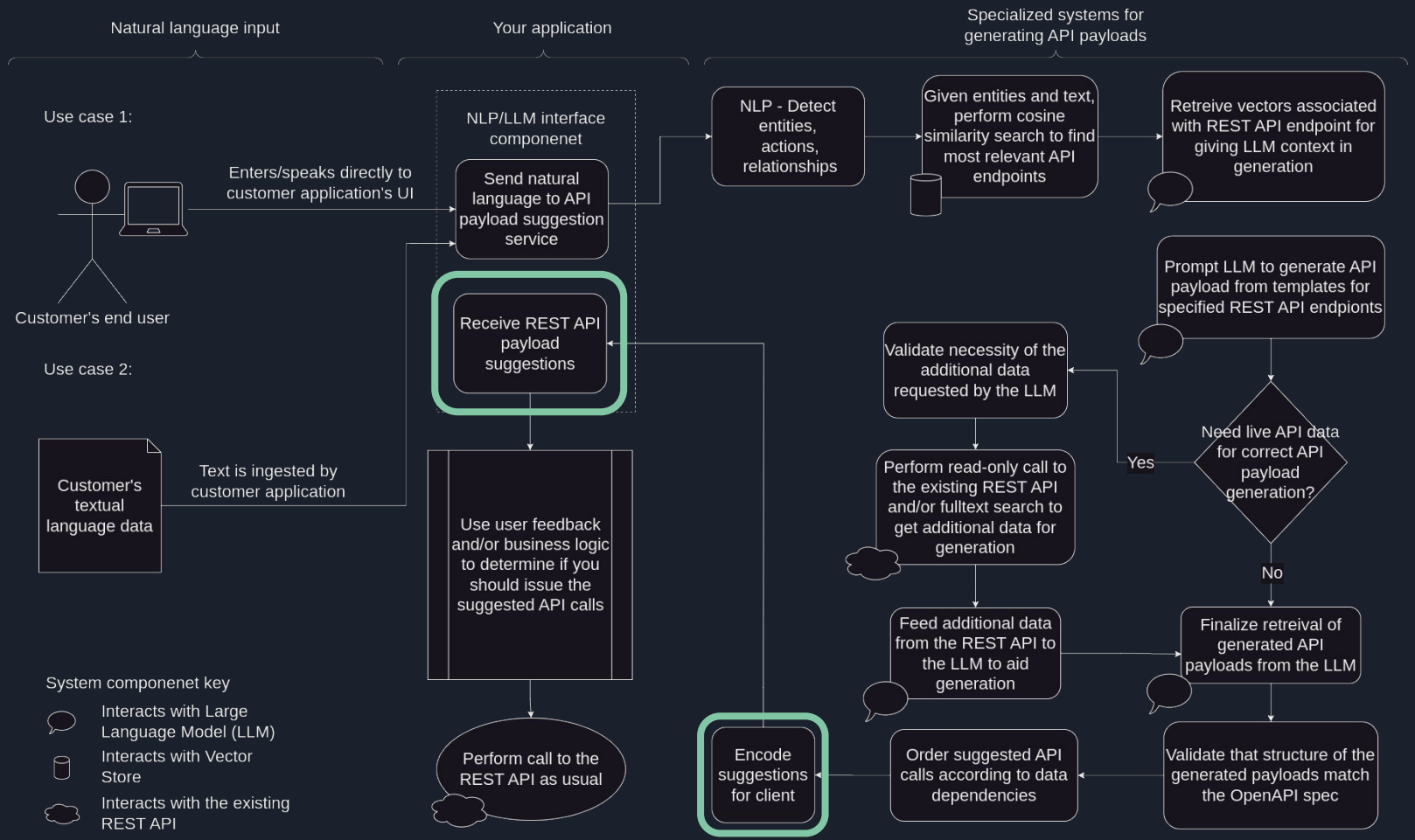
1. Design the interface between the customer's application and the API call generation code.

2. (Not shown) Encode the OpenAPI spec structure for the algorithm to use later when generating and validating payloads.



3. Tag entities and their relationships in the natural language input.

4. (Not Shown) Tell the LLM about the API structure



5. Make the existing application aware of LLM API call suggestions

2.3 Current Process Flow (Validation)

- Verify output is in JSON format (LangChain [9], Guidance AI [6])
- Once an API call is generated, confirm its structure conforms to the schema defined in the OpenAPI spec.
- Confirm that the sequence of data manipulations is consistent with the new/modified entities' relationships.
- For interactive applications confirm the generated API call with the user, and for batch applications, validate the generated API call using business logic.

6. Validate that the generated payloads conform to the OpenAPI spec.

Current Process Flow

A solution for generating API calls would ideally address the following points.

- Design the interface between the customer's application and the API call generation code.
- Encode the OpenAPI spec structure for the algorithm to use later when generating and validating payloads.. Options:
  - In Langchain, build Python classes that define the expected structure of the LLM response (Tool/Function Calling | LangChain, n.d.).
  - OpenAI, use Function Calling schema specification and hope for the best. (OpenAI Platform, n.d.; Tool/Function Calling | LangChain, n.d.)
- Tag entities and their relationships in the natural language input.
- Validate that the generated payloads conform to the OpenAPI spec.
- Tell the LLM about the API structure:
  - One-shot prompt is common in examples, but LLMs struggle to consistently generate responses that are conformant to the spec (Microsoft/Prompt-Engine, 2022/2024).
- Make the existing application aware of LLM API call suggestions:
  - For interactive apps, show the suggestions to the user.
  - For batch processing, push the generated API calls through business logic.
- Validation

No single application or framework on the market addresses all of these concerns, and implementing these solutions manually for each application that wants these features is tedious and requires expertise in using LLMs.

Solution

CueCode will provide a comprehensive service for creating Web API calls from natural language input in a risk-aware, accurate manner that puts developers - and, by extension, users - in control of when API calls are invoked.

Solution Statement

What that means:

Developers will be able to use existing API specifications, which CueCode makes understandable by LLMs, to generate the content of their API calls in conformance with their API spec.

So, our client service representative can provide input to a booking application using CueCode in natural language, “I called Patricia Davis and rescheduled her appointment from August 1st to August 16th.” The application can then use CueCode’s libraries, which have been configured using documentation about the structure of their data, to generate the following Web API request with a JSON request body:

```
POST https://the-appointment-app.com/api/v1/appointments/

{"request":{"reschedule":{"last": "Davis", "first":"Patricia", "from":{"month":8,
"day":1,"year":2024}, "to":{"month":8, "day":1,"year":2024}}}}
```

Which would then be used by the booking application to perform the API call, which will change the appointment date in their database, or prompt the user for additional information.

Solution Characteristics

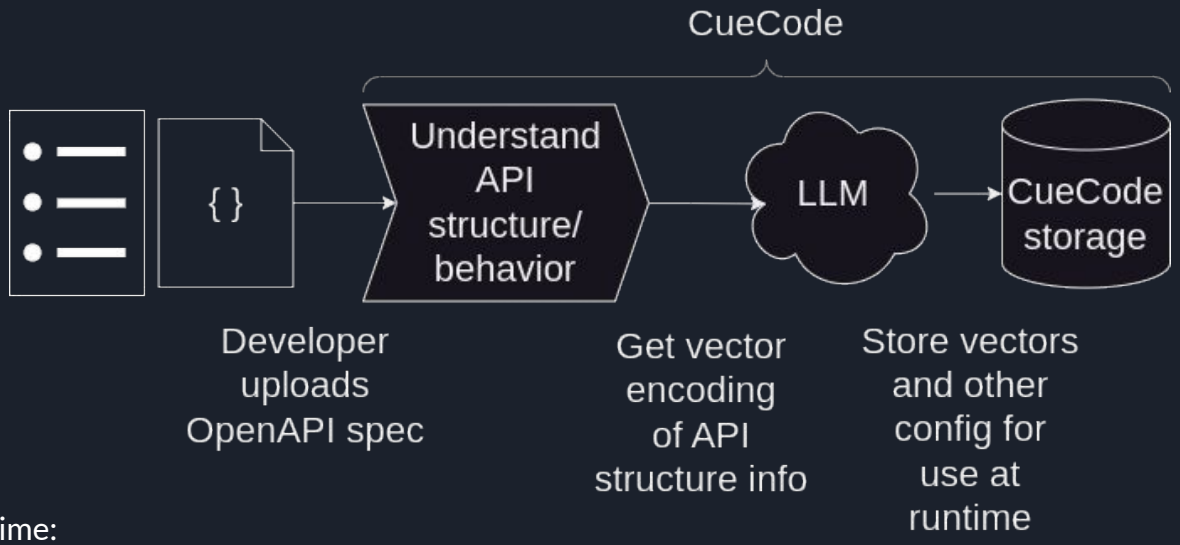
Problem Characteristics

- Forcing end users to fill out lots of forms for input is both limiting and tedious
- There is no easy way to implement using NLP to parse user input for existing applications
- It is difficult to make LLMs aware of the structure of data expected from a natural language prompt
- There is no standardized solution for translating natural language into structured data
- Translating natural language into structured data requires prompt engineering and other skill sets that do not belong to a typical front end or full stack developer
- LLM integration can cause data mutation and incorrect parsing of information

Solution Characteristics

- CueCode leverages LLM technology to parse natural language into structured data to generate API calls, simplifying the process of data entry.
- CueCode provides libraries to front end and full stack developers to easily integrate NLP into their existing applications
- Existing API specifications provide machine-readable input to guide LLMs into parsing user input from natural language, saving developers time and resources
- CueCode facilitates Human-in-the-Loop feedback to allow the end user to review the generated data in the existing user interface

Solution Process Flow (configuration)



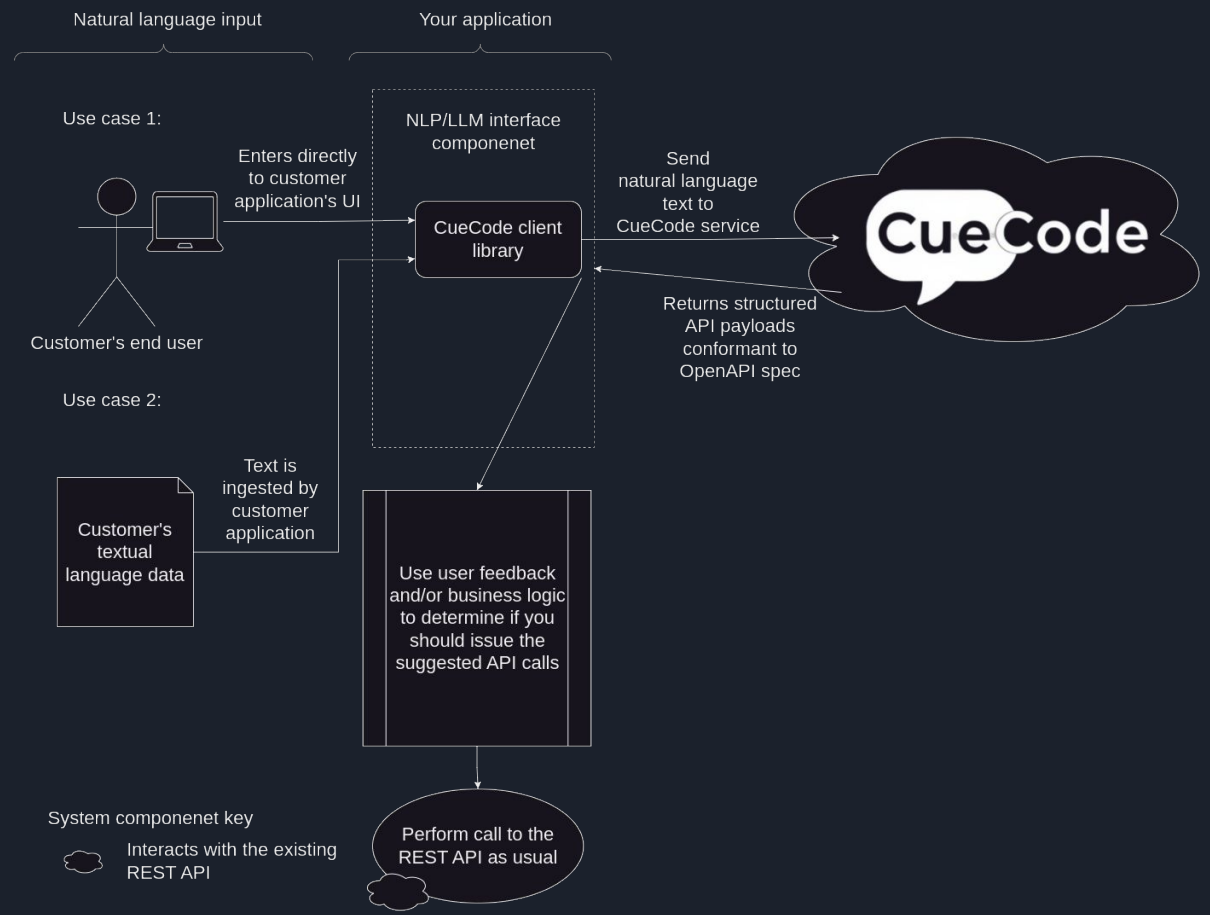
At configuration time:

- Developers ensure their OpenAPI specification is accurate.
- Developer uploads their API specification to CueCode via the Developer Portal
- Developer answer a few configuration questions.
- CueCode stores the structure and requirements for the API to aid the LLM in generating responses at runtime.
- All of this is transparent to the Developer's customers/end-users.

Solution Process Flow (runtime)

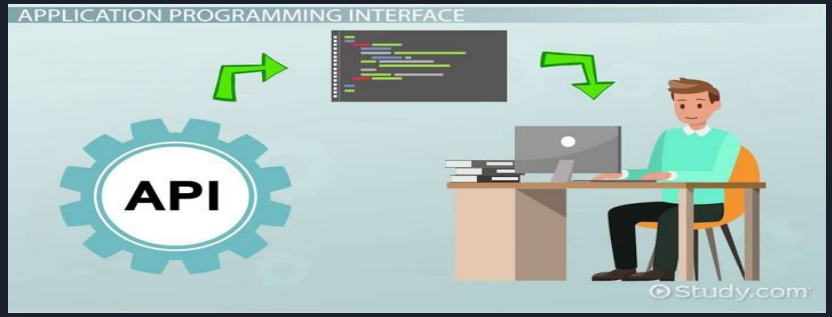
Use CueCode in your application:

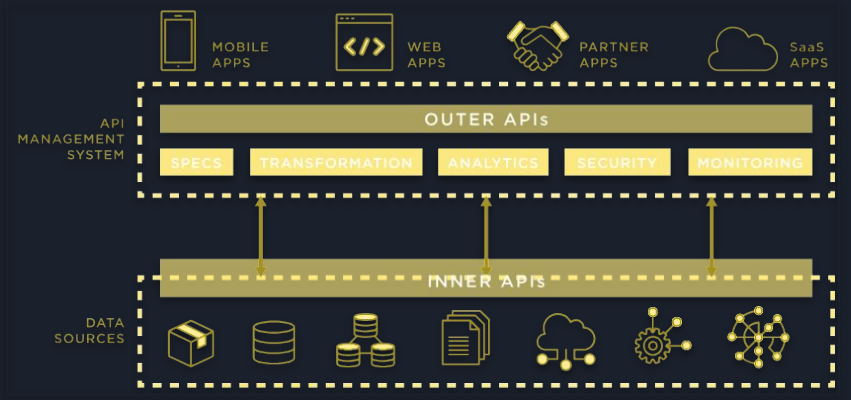
- Program your app to pass natural language text to CueCode libraries.
- Let the CueCode service figure out the structured data contained in the text.
- Use CueCode's extracted structured data within the existing application's data model. e.g.:
  - Show suggestions to the user
  - Perform API calls in a batch job
  - Validate through business rules
  - Whatever the use case requires



What it Will Do

- Will implement NLP capabilities to enable and understand natural language
- Will offer a user friendly interface (API client libraries) that developers can use
- Will provide a Developer Portal web application, where developers can upload API specifications and configure their CueCode service
- Will provide tools for customizing NLP models to fit specific domains/industries ensuring better performance for unique use cases.
- Will include documentation and support resources to help developers implement and troubleshoot various systems effectively.
- Will use REST API specifications, enabling context-aware replies that complement the distinct functionality and data structure of each application.
- Will allow for real time analysis of natural language and REST API call payload generation, enhancing user experience through immediate feedback and interactions.





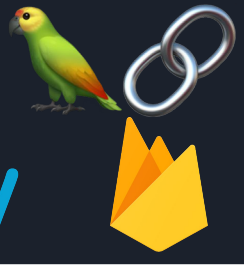
What it Will Not Do

- Will not replace human judgment when interpreting language in terms of making subjective decisions beyond its programming.
- Will not act as an AI agent
- Will not provide user-facing applications; developers will need to build their own solutions and install any necessary software/applications they need.
- Will not automatically make API calls on users' behalf; requests must first have human permission before being fulfilled.
- Will not have programming tutorials, developers will need to possess knowledge of programming to utilize CueCode effectively.
- As a student project, CueCode will not generate XML REST API payloads or GraphQL API payloads

Competition Matrix - Introduction

✔- Full Implementation
P - Partial Implementation

spaCy



Firebase

CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...

Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
Entity recognition	✔		✔	✔		P	✔
Plug and Play	✔				P	P	P
Retrieval Augmented Generation	✔	✔			✔	✔	
API call generation as a service	✔	P	P		P		P

Competition Matrix - ET

✔- Full Implementation
P - Partial Implementation

Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
Entity recognition	✔		✔	✔		P	✔

Example Prompt:
“I would like to book a hospital appointment on the 20th of October for a medical checkup.”

With entity recognition:
Book, 20th october, medical checkup

Competition Matrix - PaP

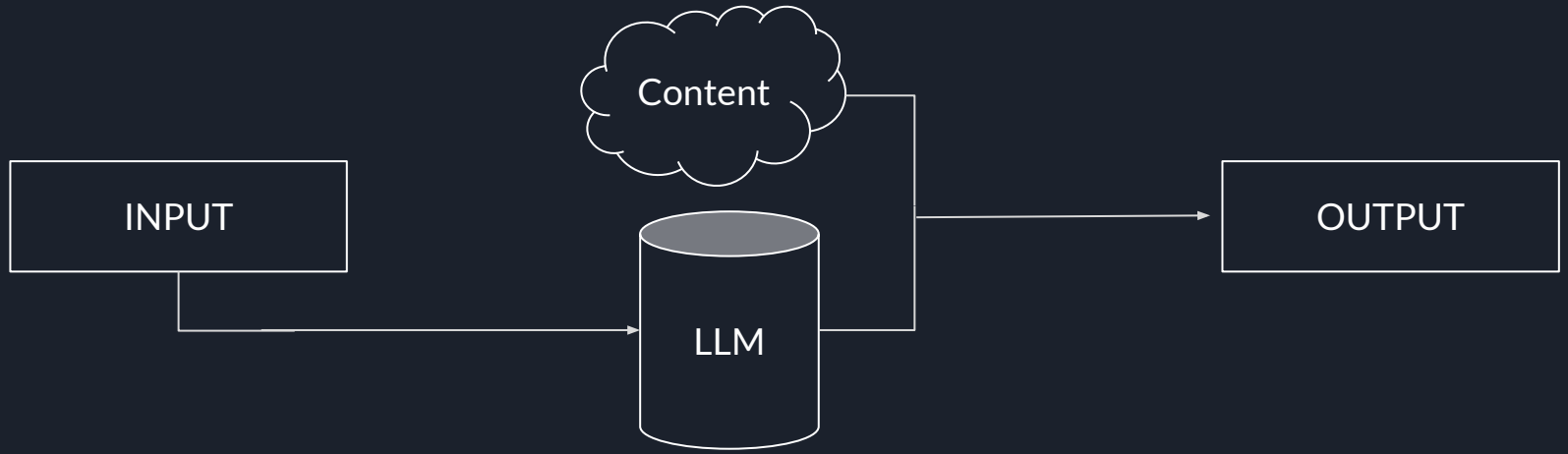
✔- Full Implementation
P - Partial Implementation

Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
Plug and Play	✔				P	P	P

Competition Matrix - RAG

✔- Full Implementation
P - Partial Implementation

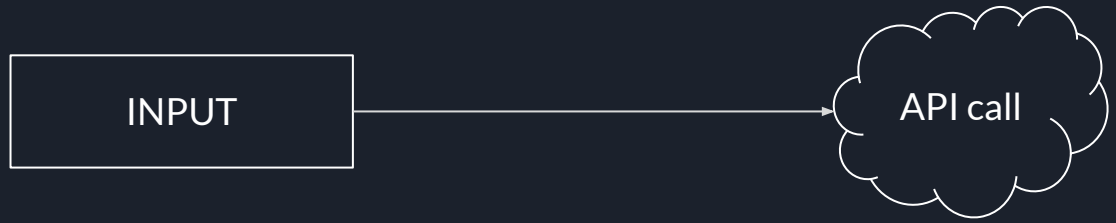
Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
Retrieval Augmented Generation	✔	✔			✔	✔	



Competition Matrix - service

✔- Full Implementation
P - Partial Implementation

Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
API call generation as a service	✔	P	P		P		P



Competition Matrix - review

Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
Entity recognition	✔		✔	✔		P	✔
Plug and Play	✔				P	P	P
Retrieval Augmented Generation	✔	✔			✔	✔	
API call generation as a service	✔	P	P		P		P

Design: How will we do it?

Notes on our engineering and current concept for CueCode's implementation

Feature table

Category	Feature	Runtime	Config-time	End user	Developer
Developer Portal	Login / Authentication		✔		✔
	Account creation / deletion		✔		✔
CueCode Config	REST API definition management		✔		✔
	Upload and manage OpenAPI specifications		✔		✔
	REST API configuration wizard		✔		✔
CueCode runtime	Process natural language and turn it into REST API payloads	✔			
	Map natural language to customer’s data entities via search or API call	✔			
Client libraries	Authentication against CueCode service	✔			

Software / Hardware Tools

- Frontend:
  - HTML
  - CSS
  - Bootstrap 5
  - Vanilla JavaScript
- API clients
  - Swagger CodeGen
- Application layer
  - Python
  - Flask Web framework
  - Jinja HTML templating
- Application libraries
  - Spacy.io
  - Thin Ollama client library
- Persistence layer
  - PostgreSQL
  - PgVector
- LLM
  - Ollama
  - Llama 3.2
- Third-party
  - identity service
  - Transactional email
- Testing
  - Jest
  - PyUnit
- CI/CD
  - GitHub Actions
  - Docker registry
- Hardware
  - GPU-equipped Kubernetes node(s) in CS Systems Group cluster

Development Tools

Version Control:

- Git with GitHub
- - The industry standard for version control is GitHub With Git. Using branching, pull requests, and issue tracking, it promotes easy collaboration and guarantees that teams function well even on challenging projects. With GitHub's built-in capabilities, we can keep an eye on changes, work together with other team members, and protect our codebase with top-notch security measures.

Integrated Development Environment (IDE):

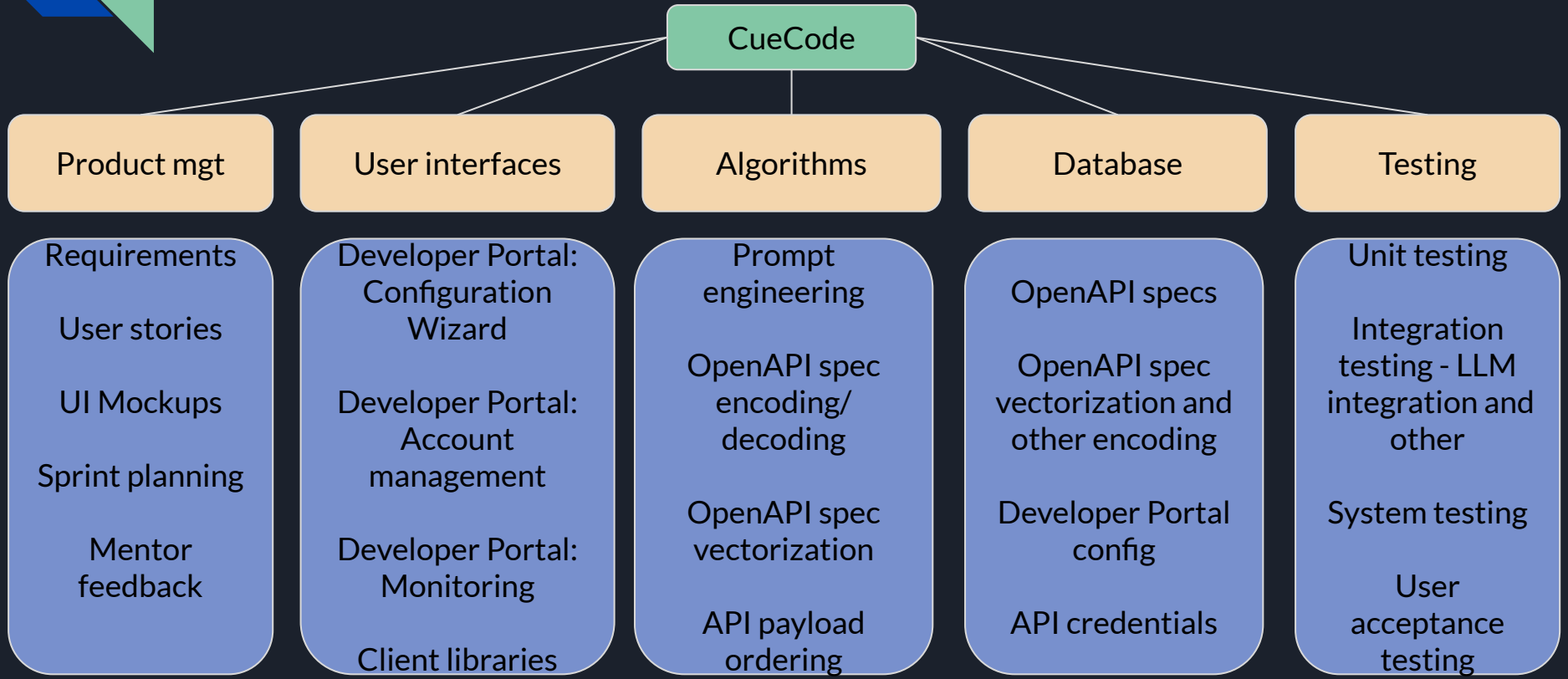
- VS Code
- - VS Code is a top option for development across many languages and frameworks because of its wide ecosystem of extensions and high esteem for flexibility. Its Git connection and real-time collaboration tool makes coding and team coordination easier and guarantee that our project stays structured and productive.
-

Continuous Integration (CI) & Continuous Deployment (CD):

- GitHub Actions and Workflows

  We manage our CI/CD pipelines with GitHub Actions, integrating deployment and testing into an easier process. Given the flexibility that GitHub Workflows offer in automating processes across the development lifecycle, we can confidently deploy, minimize manual intervention, and maintain code quality.

Work breakdown structure overview



Algorithms

- Algorithm to encode OpenAPI spec in a searchable format, likely using vectorization at config time and cosine similarity search at runtime.
- NLP - detect entities, actions, relationships in text.
- Given entities and text, find the most relevant API endpoints for which to generate payloads (likely involves cosine similarity search).
- Determine if live API data is needed to translate natural language, or if data in the natural language is sufficient to create payloads. Fetch any data needed.
- LLM Function Calls for generating consistently structured JSON output
- Prompt LLM to generate API payload using defined function calls

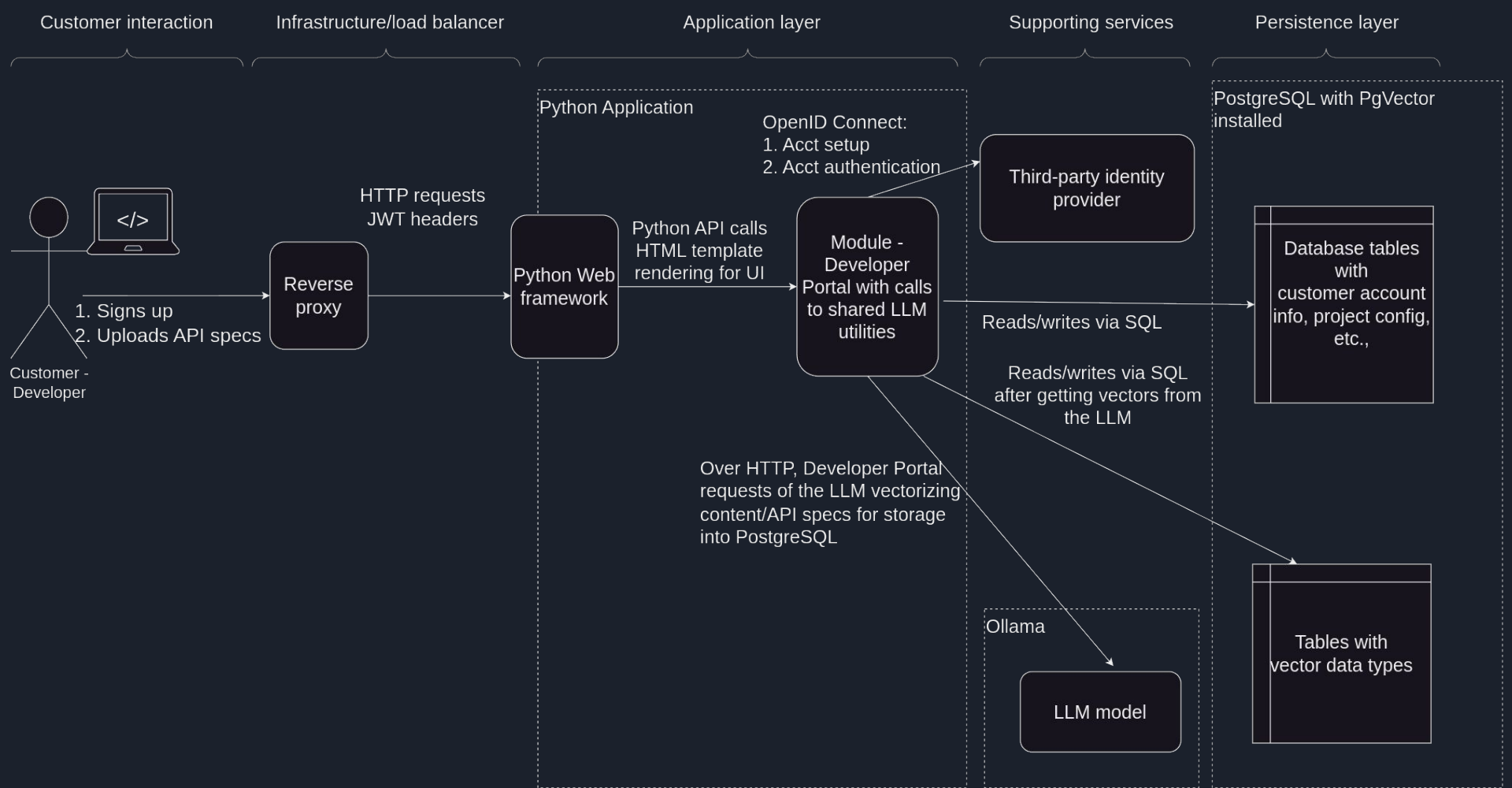
Algorithms

- Map natural language to known entities in the existing API
- Validate structure and of generated payloads
- Validate ordering of generated payloads, given dependencies
- Determine authentication needed based on OpenAPI spec and prompt Developer for it at configuration

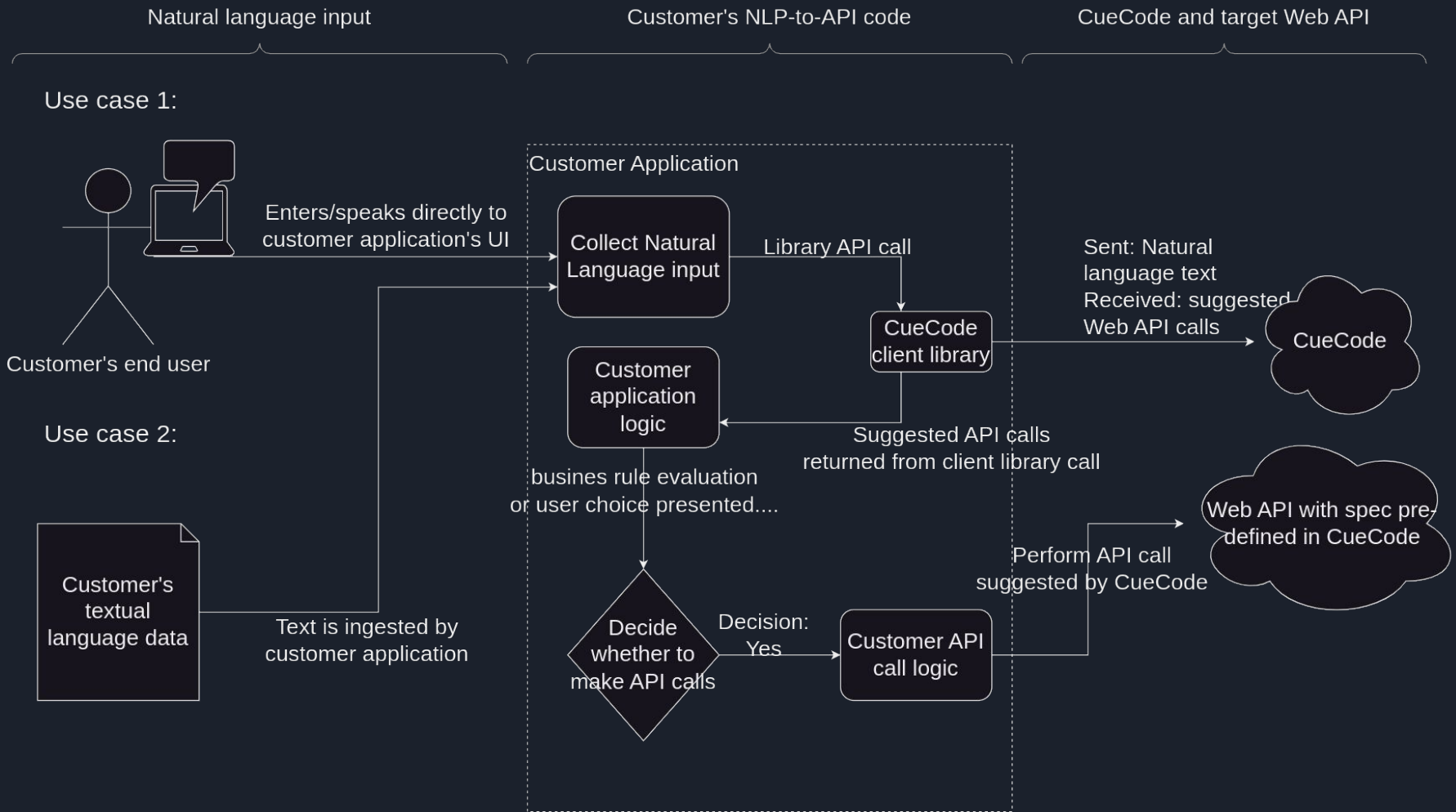
Major Functional Components

- Client libraries for customers to use for integrating with CueCode's service
  - Bindings for the CueCode runtime API
- Python modular monolith:
  - All modules exposed via Flask, a Python Web framework
  - Module: Web API payload Generation- receives natural language input and generates Web API calls from it.
  - Module: Developer Portal - account registration/management, API spec upload, configuration, generation audit and monitoring
  - Horizontally scalable via 12-factor app methodology
- PostgreSQL (Postgres) persistence:
  - PgVector extension for storing vectors generated by the LLM
  - Normal Postgre tables for customer accounts, configuration, generation monitoring and audit information
- Ollama:
  - A Web service and set of standardized LLM-call APIs that allows us to swap LLMs used while maintaining the same API contract with our Python backend.
- Third-party identity service:
  - For developer portal
  - TBD on how/whether CueCode runtime API traffic would use the same identity provider for authentication.

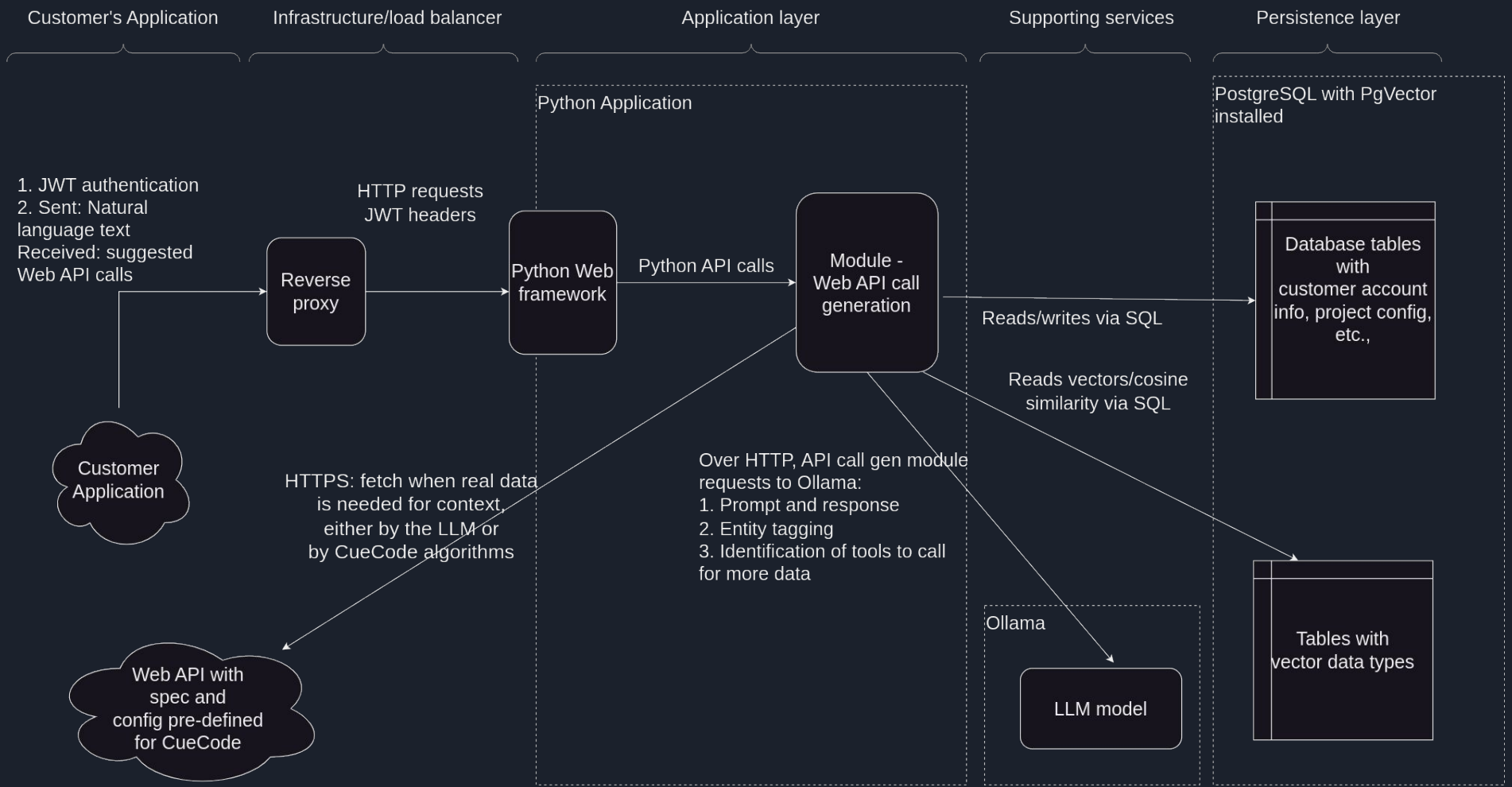
Major Functional Components Diagram - Configuration



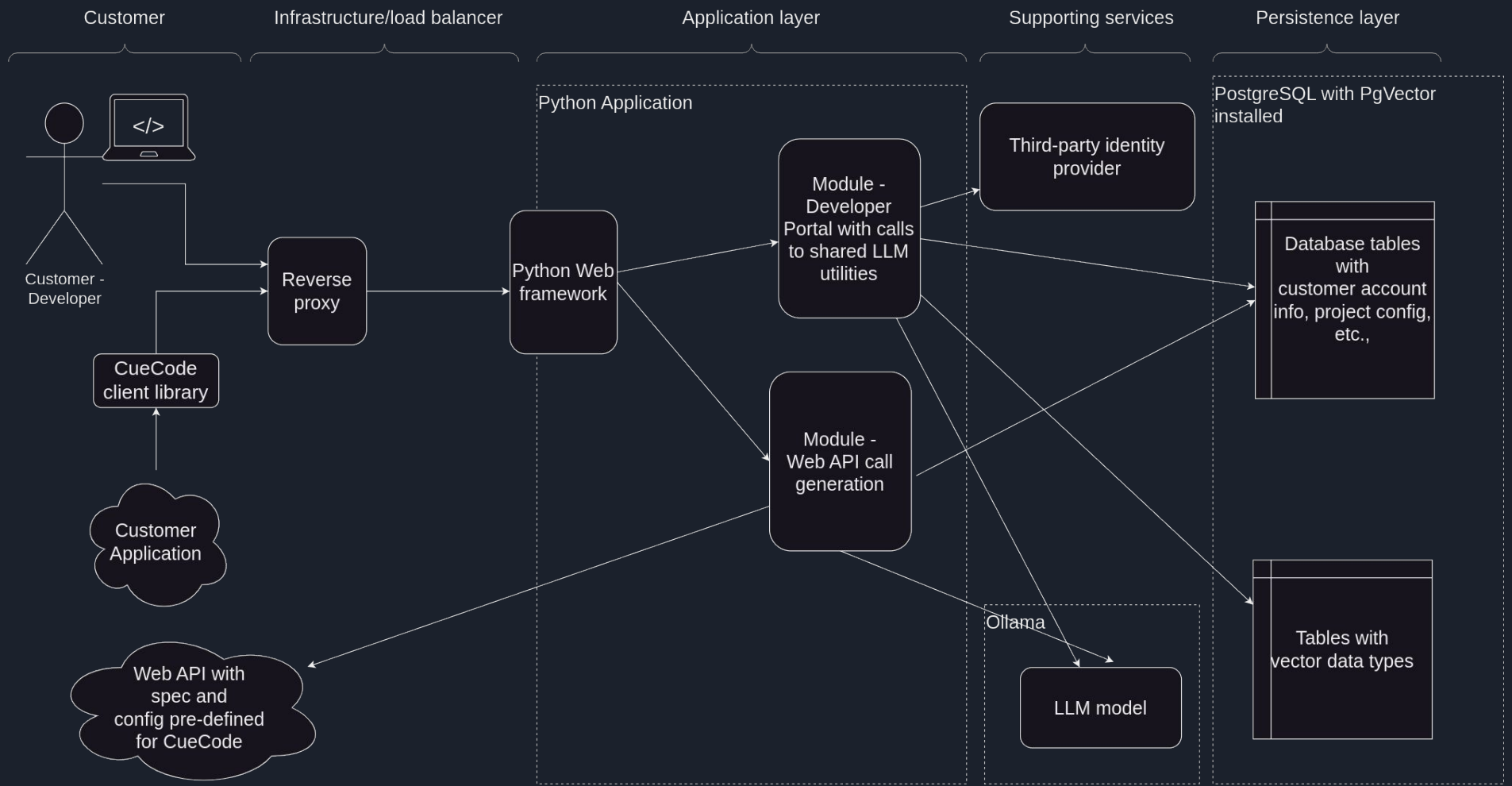
Major Functional Components Diagram - Runtime - Customer Application



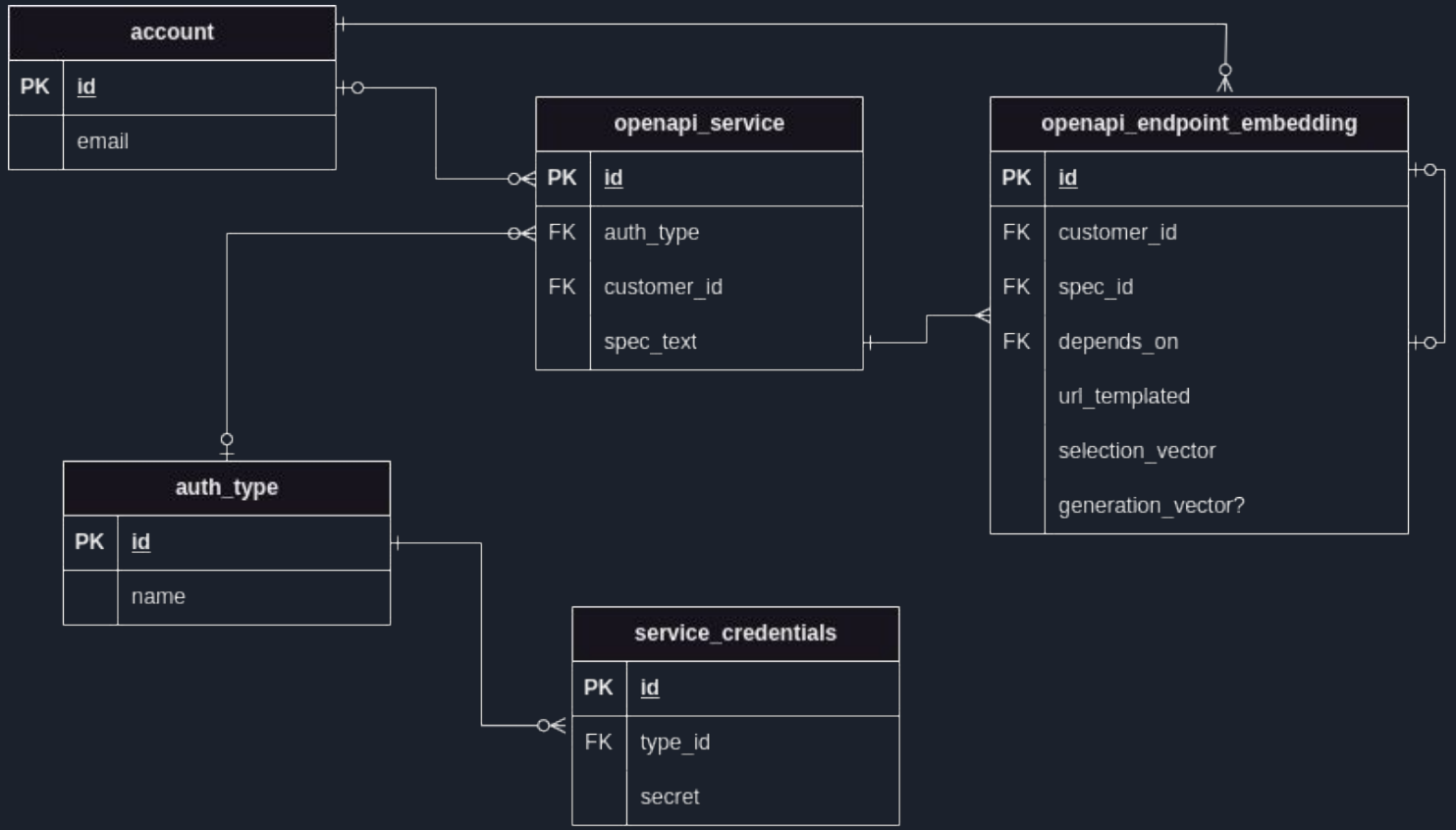
Major Functional Components Diagram - Runtime - CueCode



Major Functional Components Diagram - Overview



Entity Relation Diagram (ERD)



Risks

Risks the CueCode project faces and their mitigations

Our risk coding convention:

- “O” - Operational risks
- “R” - Regulatory risks
- “T” - Technical risks

Risks - Customer, Operational, Regulatory

O1 - Unable to procure GPU Hardware for development.

- **Mitigation approach:** Control
- **Mitigation:**
  - In Spring ‘25, execute an already approved request for GPU time with the CS Systems Group

O2 - CueCode customers may overlook critical security or operational risks when generating API calls.

- **Mitigation approach:** Continue Monitoring
- **Mitigation:** Perform thorough logging, audits to provide detailed error checking tools for developers.

Probability

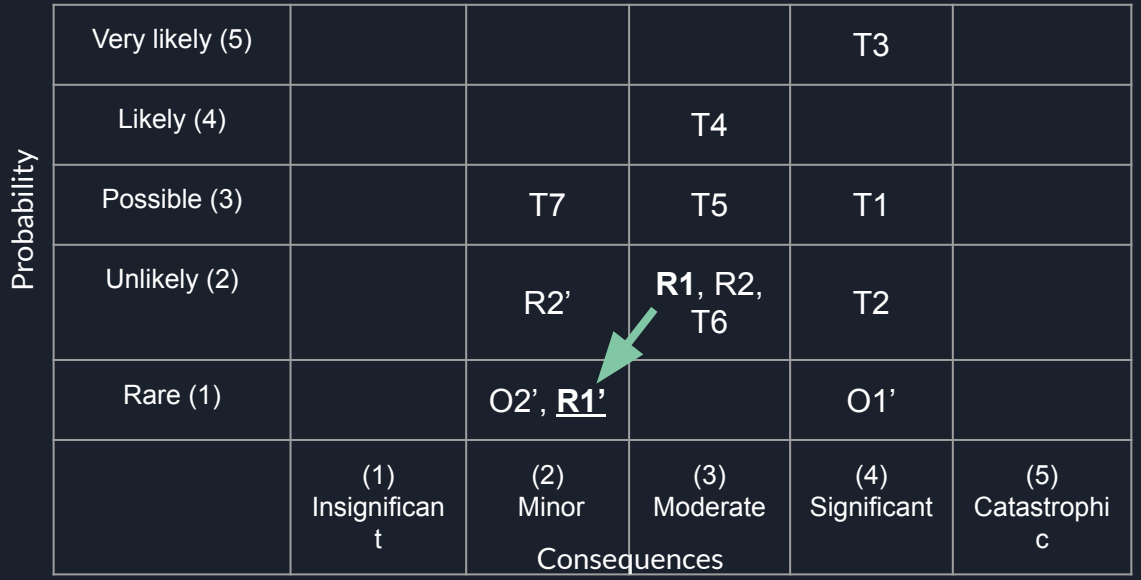
Very likely (5)				T3	
Likely (4)			T4		
Possible (3)		T7	T5	T1	**O1**
Unlikely (2)		R2’	R1, R2, T6	T2	
Rare (1)		**O2’**	**O2**	**O1’**	
	(1) Insignificant	(2) Minor	(3) Moderate	(4) Significant	(5) Catastrophic

Consequences

Risks - Customer, Operational, Regulatory

R1 - The use of API specifications might infringe on proprietary or closed API usage policies, leading to legal issues.

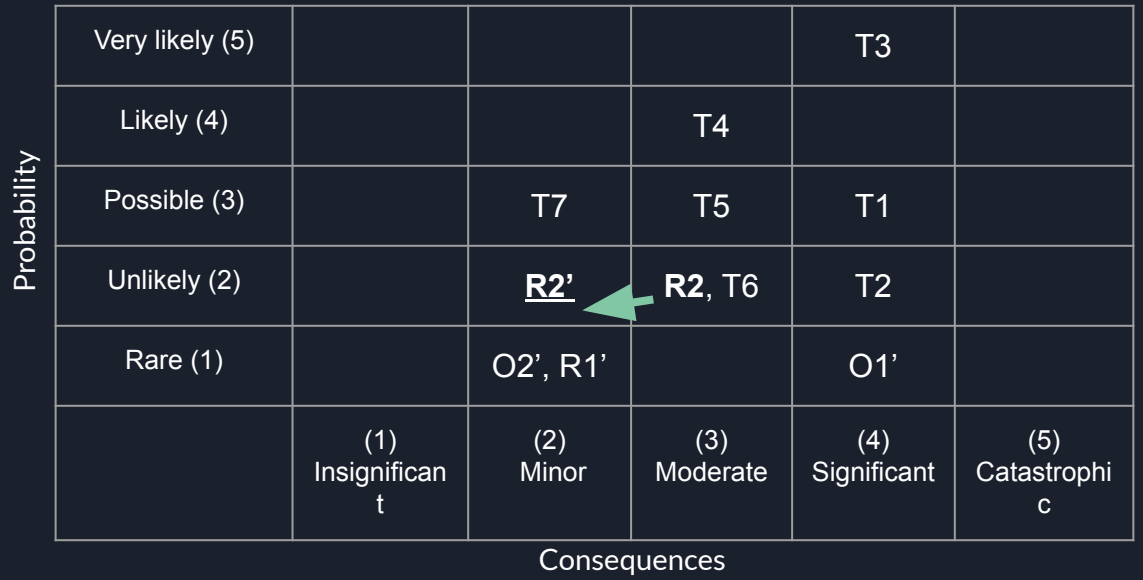
- **Mitigation approach:** Avoid
- **Mitigation:** Check downstream API usage against known limits, check with professionals about API licenses, develop and publish a platform abuse notice process for API providers to use, and stay away from violating proprietary API standards and procedures.



Risks - Customer, Operational, Regulatory

R2 - Storage of API credentials makes CueCode an enticing target for cybersecurity attacks.

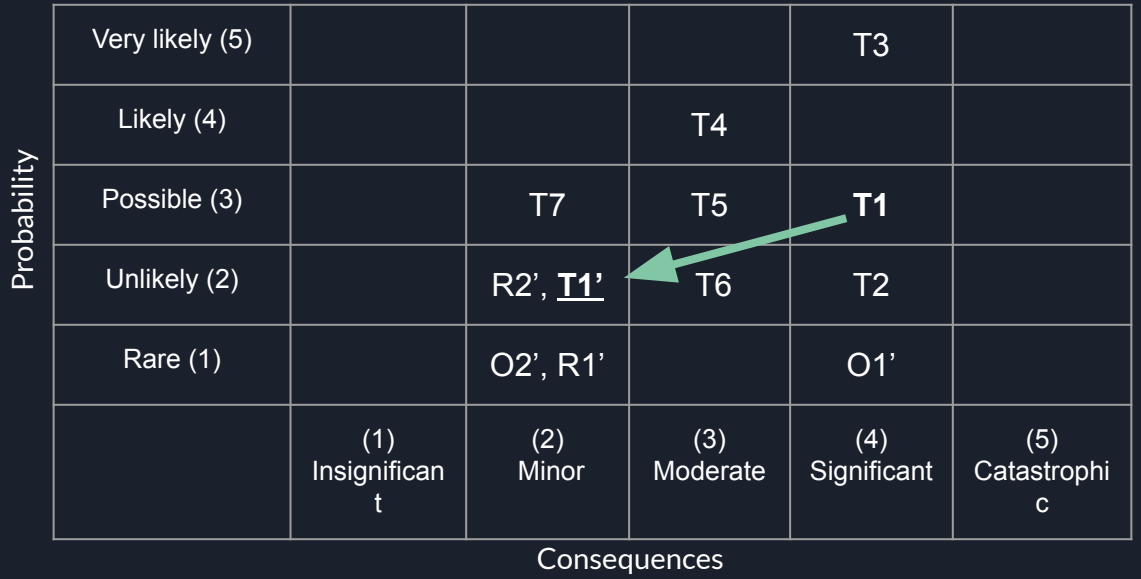
- **Mitigation approach:** Control
- **Mitigation:**
  - Legal - apply terms of use that protect CueCode in the case of data breach.
  - Technical - separate tenant credentials with care.
  - Technical - guide developers to use scoped API keys; use OAuth2 where possible for user-specific data



Risks - Technical

T1 - LLM won't generate API calls without few-shot prompt examples.

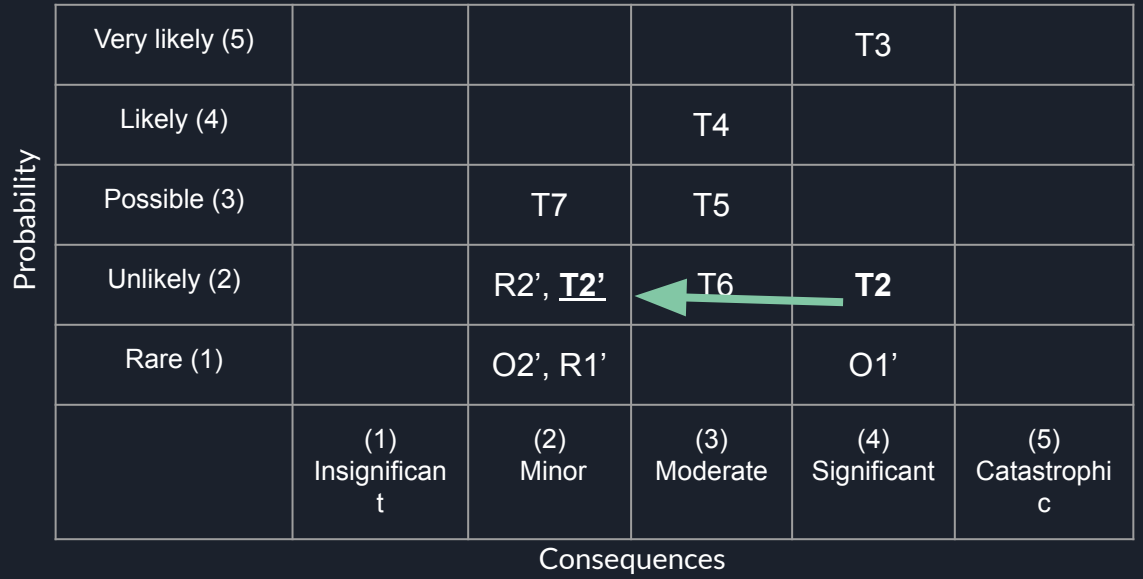
- **Mitigation approach:** Control
- **Mitigation:**
  - Validation process for prompt engineering.
  - Require that developers include a few examples in their OpenAPI specs.



Risks - Technical

T2 - LLM won't generate API calls without hundreds or thousands of examples.

- **Mitigation approach:** Continue Monitoring.
- **Mitigation:**
  - If risk is realized, then pivot to change value propositions and require backend development from the customer



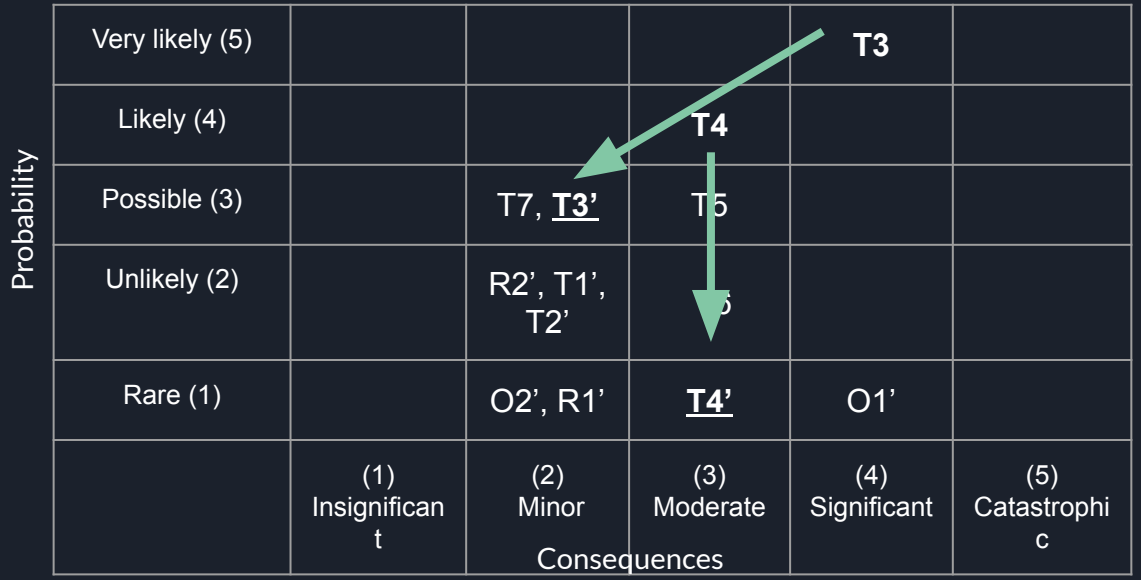
Risks - Technical

T3 - Vastness of frontend API client ecosystem precludes building CueCode client libraries for all popular languages and frameworks.

- **Mitigation approach:** Transfer
- **Mitigation:**
  - Use Swagger CodeGen for our own CueCode backend API.
  - Open-source our client library code.

T4 - Potential exposure of sensitive API information through generated API calls.

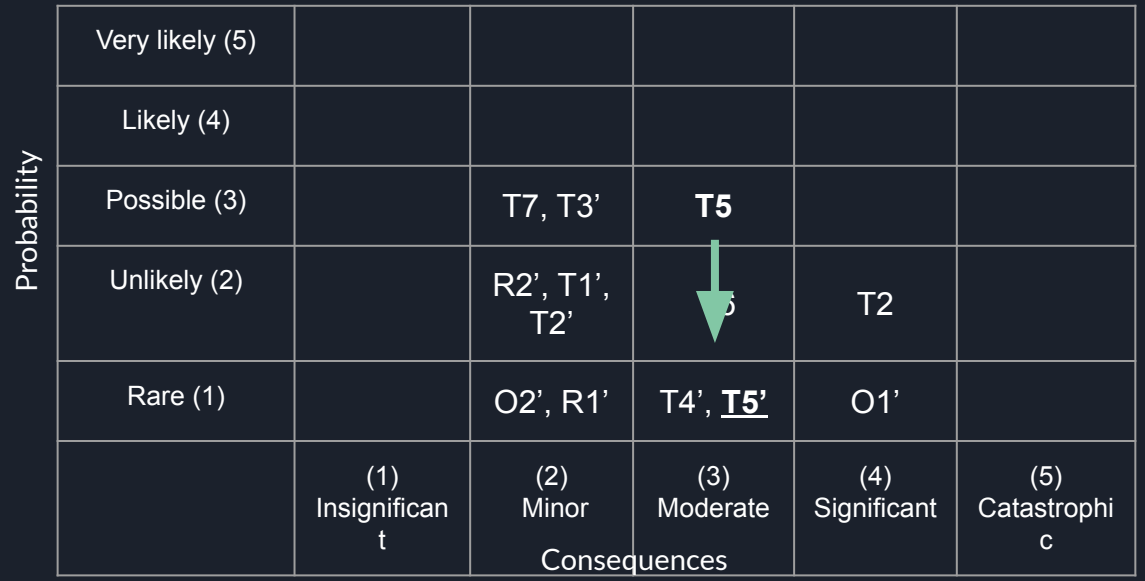
- **Mitigation approach:** Control
- **Mitigation:** separate API authentication and LLM generation concerns in the CueCode payload generation algorithm.



Risks - Technical

T5 - Obsolescence of vendor libraries and services in the greenfield AI market.

- **Mitigation approach:** Avoid
- **Mitigation:**
  - Use OLLama backend communication with the LLM, allowing swappable LLM models according to CueCode's needs.
  - Use PgVector, an extension to the FOSS PostgreSQL RDBMS, for vector storage.
  - Develop a simple Python backend without undue reliance popular AI libraries, most of which are pre-v1 and, incidentally, overfit for CueCode's purpose.



Risks - Technical

T6 - Our validation processes might find that CueCode might require a lot of time to provide accurate results, especially if generating many API payloads.

- **Mitigation approach:** Continue Monitoring
- **Mitigation:** Defer development of frontend libraries until we know whether backend processing takes so long as to require asynchronous processing, instead of request-response.

Probability

Very likely (5)					
Likely (4)					
Possible (3)		T7, T3’	**T6**		
Unlikely (2)		R2’, T1’, T2’	**T6’**	T2	
Rare (1)		O2’, R1’	T4’, T5’	O1’	
	(1) Insignificant	(2) Minor	(3) Moderate	(4) Significant	(5) Catastrophic

Consequences

Risks - Technical

T7 - Elevated demand may surpass the capacity of the system, resulting in disruptions or delays.

- **Mitigation approach:** Continue Monitoring
- **Mitigation:** As traffic increases, scalability and efficiency are ensured through:
  - Starting development with architecture that allows scaling (containerized 12-factor app)
  - Regular performance testing
  - Load balancing.

Risks - Mitigation landscape

Before

Probability						
(5)				T3		
(4)			T4			
(3)		T7	T5	T1	O1	
(2)			R1, R2, T5, T6	T2		
(1)			O2			
	(1)	**(2)**	**(3)**	**(4)**	**(5)**	

Consequences

After

Probability					
(5)					
(4)					
(3)		T3’			
(2)		R2’, T1’, T2’	T6’		
(1)		O2’, R1’, T7’	T4’, T5’	O1’	
	(1)	**(2)**	**(3)**	**(4)**	**(5)**

Consequences

Conclusion

- Leverages existing tools and techniques
- Develops a framework for developers
- Delights users
- Reduces risk for important actions/data entry

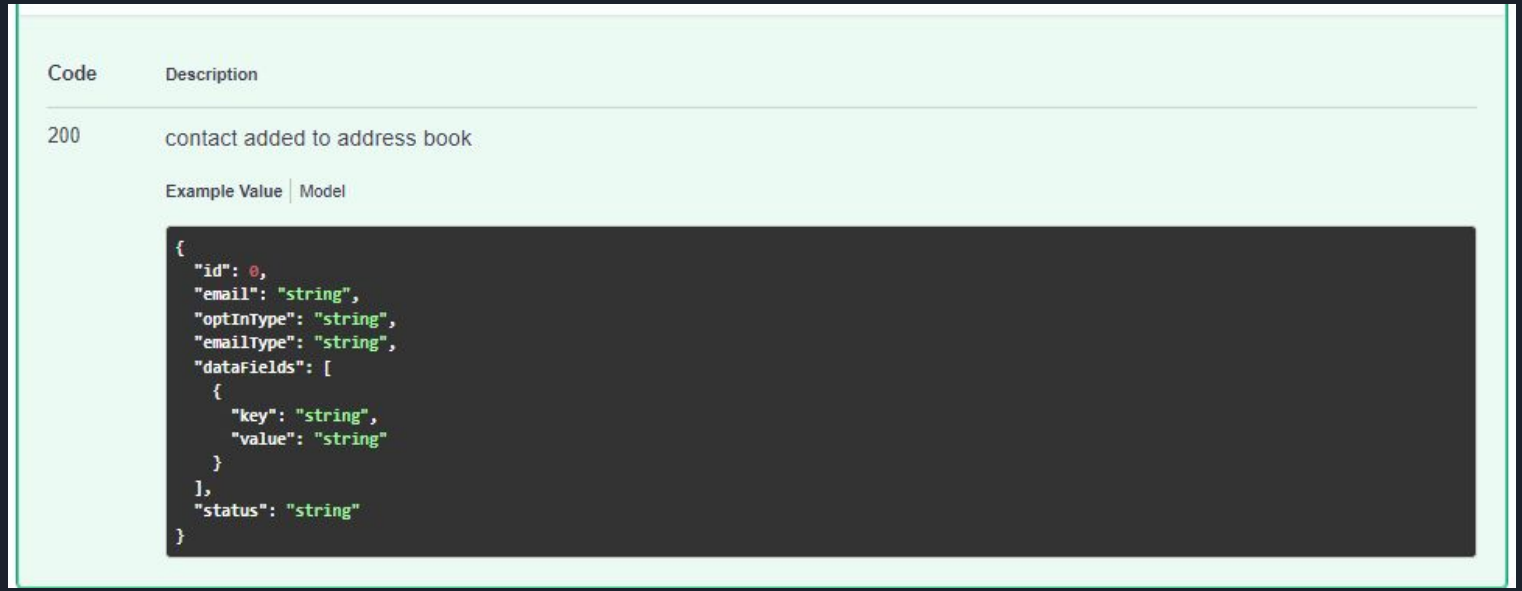
=> so that you can Humanize Web APIs, without the headache.

Appendix A

REST API tooling

How do we currently get Json Payloads

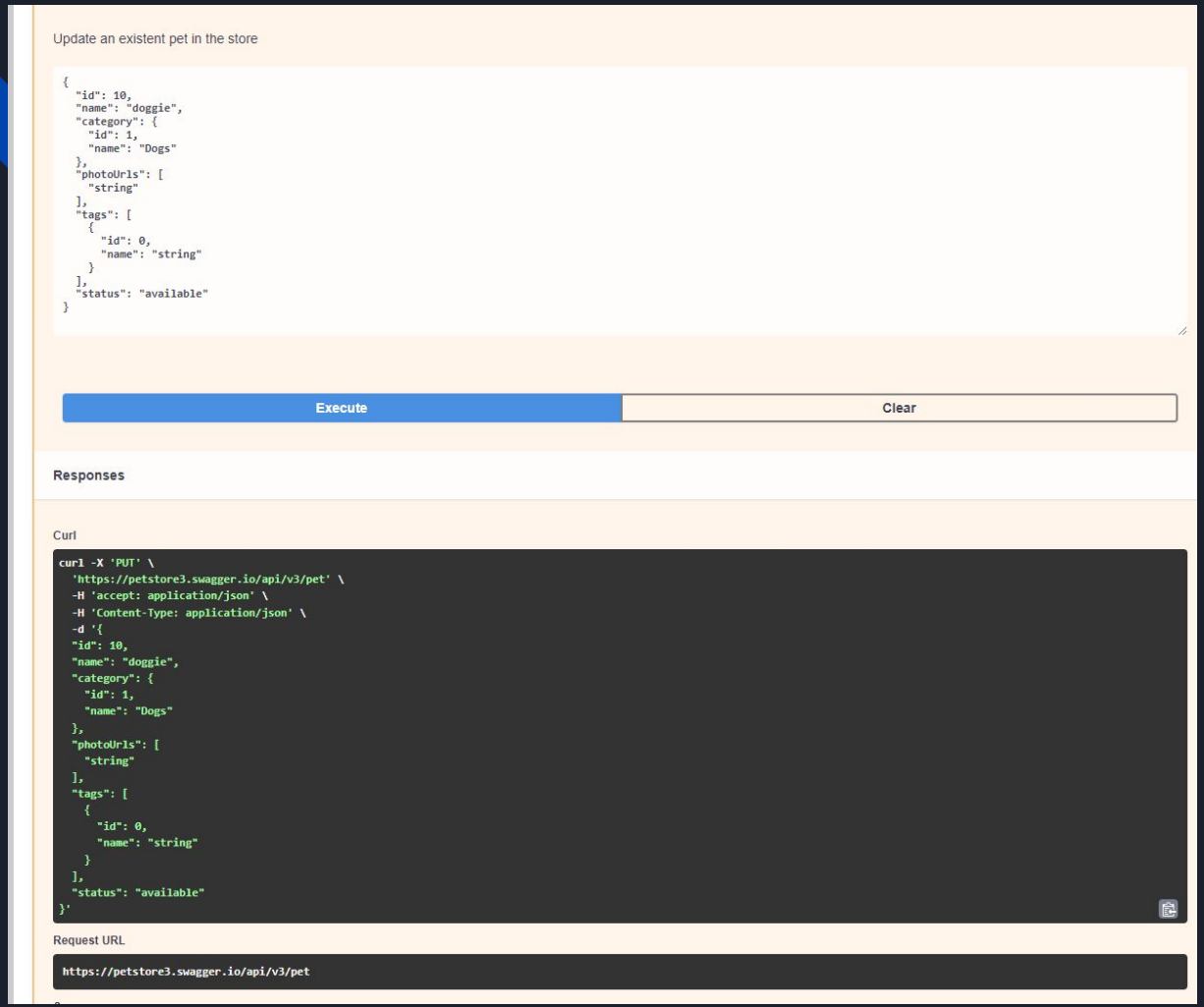
Swagger Hub, Openapi…Postman, requests libraries
And many many more



Code	Description
200	contact added to address book

Example Value | Model

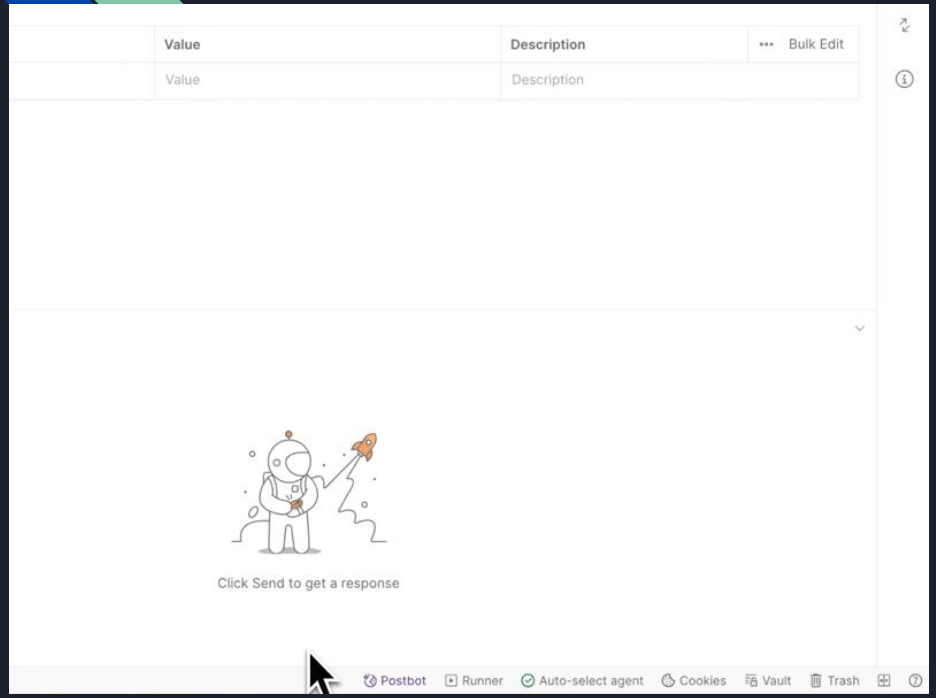
```
{
  "id": 0,
  "email": "string",
  "optInType": "string",
  "emailType": "string",
  "dataFields": [
    {
      "key": "string",
      "value": "string"
    }
  ],
  "status": "string"
}
```

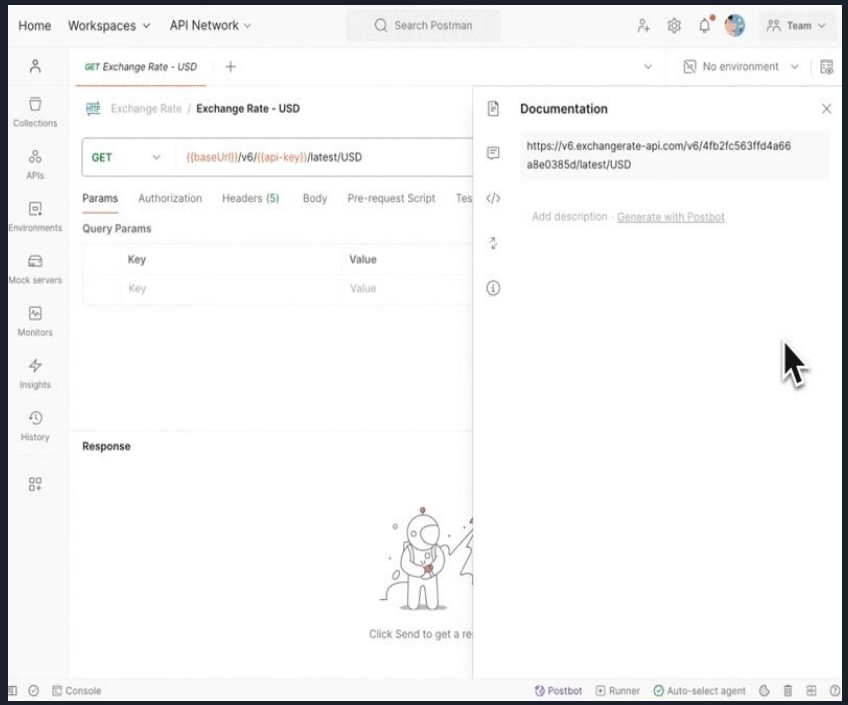


Swagger Hub Example

API TESTING
Mock Servers
API Detection

Example Postman Usage





References

References

About continuous integration with GitHub Actions. (n.d.). GitHub Docs. Retrieved October 22, 2024, from https://docs.github.com/en/actions/about-github-actions/about-continuous-integration-with-github-actions

About Git. (n.d.). GitHub Docs. Retrieved October 22, 2024, from https://docs.github.com/en/get-started/using-git/about-git

Against LLM maximalism · Explosion. (2023, May 18). https://explosion.ai/blog/explosion.ai

Baker, S. (2024). *Paragonsean/ChatBotAsync* [Python]. https://github.com/paragonsean/ChatBotAsync (Original work published 2024)

Cloud Natural Language. (n.d.). Google Cloud. Retrieved September 26, 2024, from https://cloud.google.com/natural-language

Evaluation | Genkit. (n.d.). Firebase. Retrieved September 14, 2024, from https://firebase.google.com/docs/genkit/evaluation

Firebase Genkit. (n.d.). Retrieved September 14, 2024, from https://firebase.google.com/docs/genkit

Function Calling. (n.d.). Retrieved September 14, 2024, from https://platform.openai.com/docs/guides/function-calling

References

Learn Data with Mark (Director). (2023, July 26). *Returning consistent/valid JSON with OpenAI/GPT* [Video recording]. https://www.youtube.com/watch?v=IJJkBaO15Po

Lorica, B. (2024, January 25). *Expanding AI Horizons: The Rise of Function Calling in LLMs*. Gradient Flow. https://gradientflow.com/expanding-ai-horizons-the-rise-of-function-calling-in-llms/

Merritt, R. (2023, November 15). *What Is Retrieval-Augmented Generation aka RAG?* NVIDIA Blog. https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/

Microsoft/prompt-engine. (2024). [TypeScript]. Microsoft. https://github.com/microsoft/prompt-engine (Original work published 2022)

Natural Language Processing [NLP] Market Size | Growth, 2032. (n.d.). Retrieved September 14, 2024, from https://www.fortunebusinessinsights.com/industry-reports/natural-language-processing-nlp-market-101933

OpenAI Platform. (n.d.-a). Retrieved September 10, 2024, from https://platform.openai.com

OpenAI Platform. (n.d.-b). Retrieved October 24, 2024, from https://platform.openai.com

References

OpenAPI Specification—Version 3.1.0 | Swagger. (n.d.). Retrieved September 10, 2024, from https://swagger.io/specification/

OpenAPITools/openapi-generator. (2024). [Java]. OpenAPI Tools. https://github.com/OpenAPITools/openapi-generator (Original work published 2018)

piembsystech. (2023, October 2). *Dynamic Binding in Python Language*. PiEmbSysTech. https://piembsystech.com/dynamic-binding-in-python-language/

SpaCy · Industrial-strength Natural Language Processing in Python. (n.d.). Retrieved September 26, 2024, from https://spacy.io/

Stanfordnlp/dspy. (2024). [Python]. Stanford NLP. https://github.com/stanfordnlp/dspy (Original work published 2023)

Su, Y., Awadallah, A. H., Khabsa, M., Pantel, P., Gamon, M., & Encarnacion, M. (2017). Building Natural Language Interfaces to Web APIs. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 177–186. https://doi.org/10.1145/3132847.3133009

Tool/function calling | LangChain. (n.d.). Retrieved September 14, 2024, from https://python.langchain.com/v0.1/docs/modules/model_io/chat/function_calling/

References

Tutorial: ChatGPT Over Your Data. (2023, February 6). LangChain Blog. https://blog.langchain.dev/tutorial-chatgpt-over-your-data/

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (arXiv:2201.11903). arXiv. http://arxiv.org/abs/2201.11903

What Is NLP (Natural Language Processing)? | IBM. (2021, September 23). https://www.ibm.com/topics/natural-language-processing

Why Visual Studio Code? (n.d.). Retrieved October 22, 2024, from https://code.visualstudio.com/docs/editor/whyvscode

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). *ReAct: Synergizing Reasoning and Acting in Language Models* (arXiv:2210.03629). arXiv. http://arxiv.org/abs/2210.03629

Zafin, E. at. (2023, August 15). Bridging the Gap: Exploring use of Natural Language to interact with Complex Systems. *Engineering at Zafin*. https://medium.com/engineering-zafin/bridging-the-gap-exploring-using-natural-language-to-interact-with-complex-systems-11c1b056cc19