John Pretz SWENG861 Prof: Santosh Nalubandhu FALL Semester 2025

Introduction:

The Cat Facts API project is a comprehensive exercise in building a RESTful web service using FastAPI. The goal of the project is to demonstrate the ability to consume third-party APIs, validate and store data, design and implement CRUD operations, and enhance functionality with caching, rate-limiting, and authentication mechanisms. By integrating with the public API catfact. ninja, the project allows users to fetch, view, update, and delete cat facts in a structured and secure way.

The project highlights several essential software engineering practices, including:

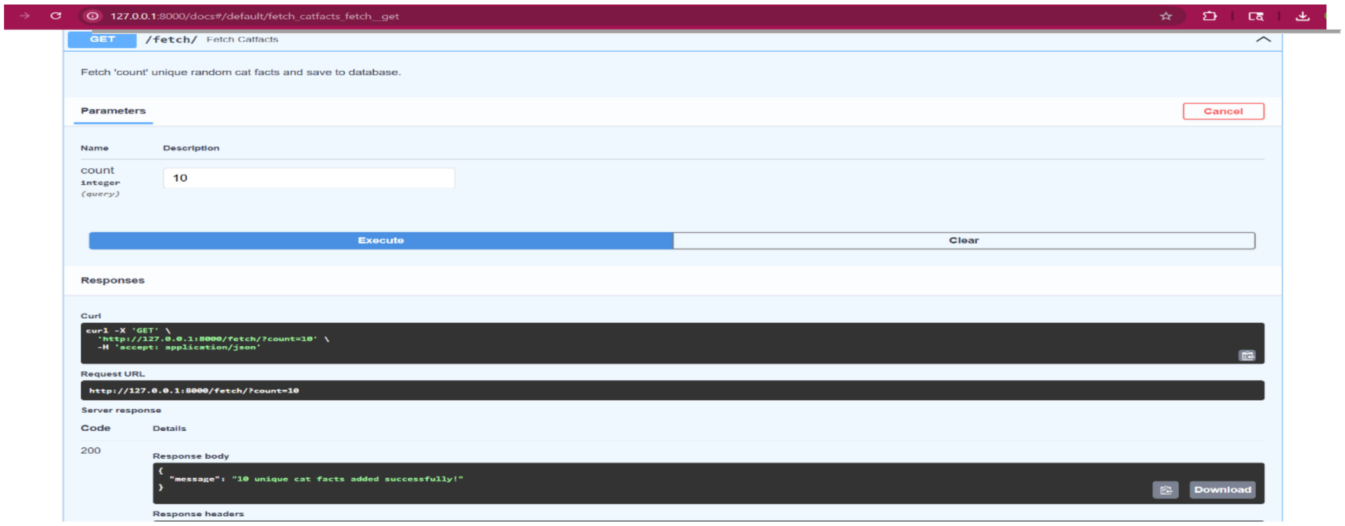
- Data validation with Pydantic to ensure database integrity.
- Database management using SQLAlchemy ORM and SQLite for persistence.
- RESTful API design, including well-structured endpoints and proper HTTP methods.
- Performance optimization through caching and rate-limiting.
- Security by implementing API key authentication for sensitive operations.

The following sections detail the implementation, features, and testing of the API:

1: Consuming 3rd Party API and Data Validation

The first part of the project focused on consuming a third-party API, catfact. ninja, to fetch cat facts. The primary goal was to retrieve structured data and validate it before storing it in the local database. Pydantic models were used to enforce strict data validation, ensuring that each cat fact contained a valid string and an automatically assigned ID. This step prevented malformed data from entering the system and ensured consistent structure throughout the database.

The following screenshot demonstrates the fetching of data from the third-party API.


127.0.0.1:8000/docs#/default/fetch_catfacts_fetch__get

GET /fetch/ Fetch Catfacts

Fetch 'count' unique random cat facts and save to database.

Parameters

Name	Description
count integer (query)	10

Responses

Curl
```
curl -X 'GET' \
  'http://127.0.0.1:8000/fetch/?count=10' \
  -H 'accept: application/json'
```
Request URL
```
http://127.0.0.1:8000/fetch/?count=10
```
Server response

Code: 200

Response body
```
{
  "message": "10 unique cat facts added successfully!"
}
```
Response headers

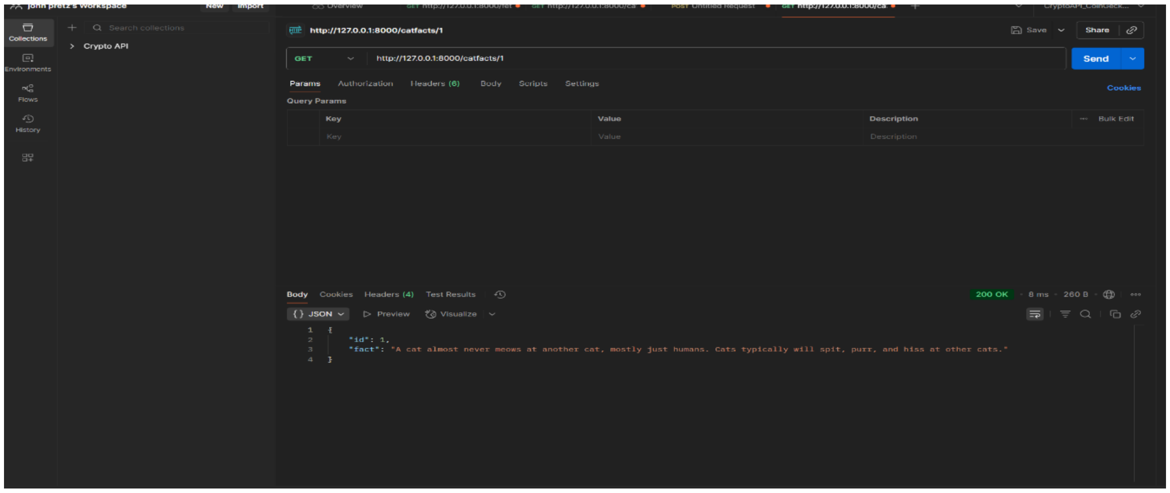
John Pretz SWENG861 Prof: Santosh Nalubandhu FALL Semester 2025

In Part 2, the project implemented a fully functional RESTful API using FastAPI. The API supports standard CRUD operations:

- Create – Users can add new cat facts to the database using the /catfacts/ POST endpoint.
- Read – The API allows fetching all cat facts or a fact by ID through /catfacts/ GET endpoints.
- Update – Existing cat facts can be modified using the /catfacts/{fact_id} PUT endpoint.
- Delete – Users can remove cat facts via the /catfacts/{fact_id} DELETE endpoint.

The endpoints were tested extensively to ensure proper behavior, including scenarios for valid and invalid requests. Database interactions were handled through SQLAlchemy ORM, ensuring that changes persisted consistently.

The following diagrams illustrate the POST catsfast operation, which creates a new cat fact, and the GET catsfacts operation, which retrieves all cat facts.



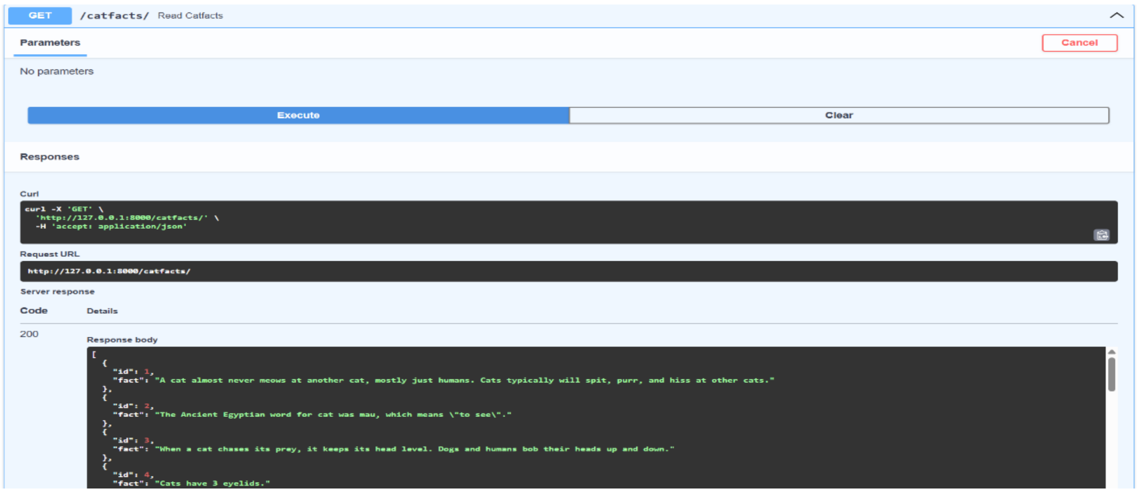
GET http://127.0.0.1:8000/catfacts/1

Query Params

Key	Value	Description
Key	Value	Description

200 OK · 8 ms · 260 B

```
{
    "id": 1,
    "fact": "A cat almost never meows at another cat, mostly just humans. Cats typically will spit, purr, and hiss at other cats."
}
```



/catfacts/ Read Catfacts

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/catfacts/' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/catfacts/
```

Server response

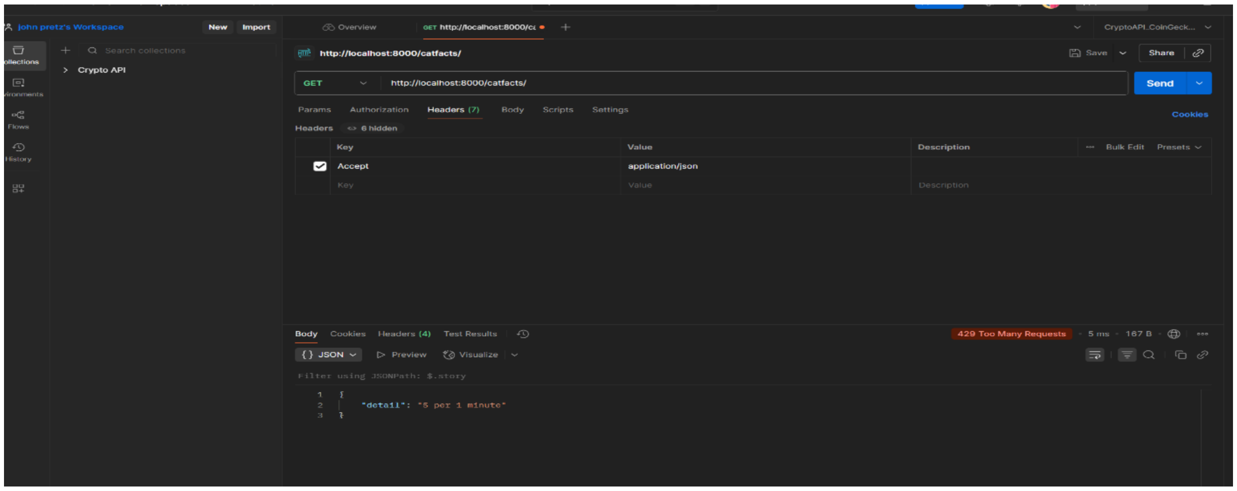
Code	Details
200	Response body

```
[
  {
    "id": 1,
    "fact": "A cat almost never meows at another cat, mostly just humans. Cats typically will spit, purr, and hiss at other cats."
  },
  {
    "id": 2,
    "fact": "The Ancient Egyptian word for cat was mau, which means \"to see\"."
  },
  {
    "id": 3,
    "fact": "When a cat chases its prey, it keeps its head level. Dogs and humans bob their heads up and down."
  },
  {
    "id": 4,
    "fact": "Cats have 3 eyelids."
```

Part 3 added performance optimizations and safeguards to the API. Caching was implemented using FastAPI Cache with an in-memory backend, improving response times for repeated requests. Rate-limiting was applied using SlowAPI, restricting users to 5 requests per minute to prevent abuse and protect the server from overload.

All endpoints were thoroughly tested to ensure the cache worked correctly, showing cache hits and misses for repeated requests. This also allowed for efficient testing of the system's behavior under constrained request limits.

The following visuals demonstrate the testing of the rate-limiting in action and cache miss on the first request.

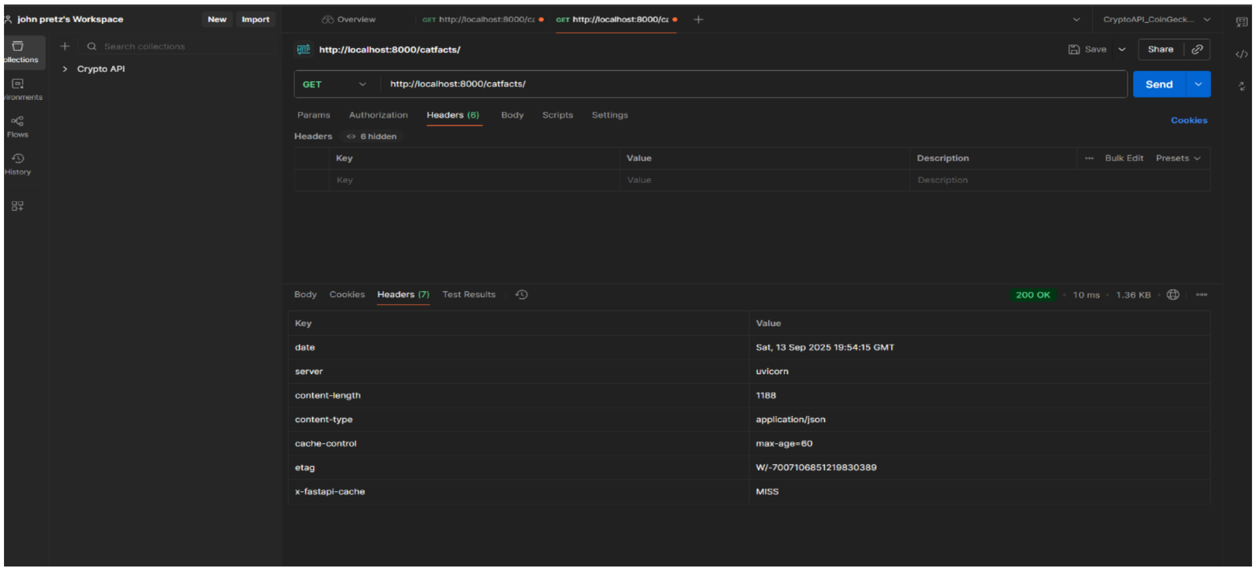


GET http://localhost:8000/catfacts/

Key	Value	Description
Accept	application/json	

429 Too Many Requests · 5 ms · 167 B

```
{
    "detail": "5 per 1 minute"
}
```



GET http://localhost:8000/catfacts/

200 OK · 10 ms · 1.36 KB

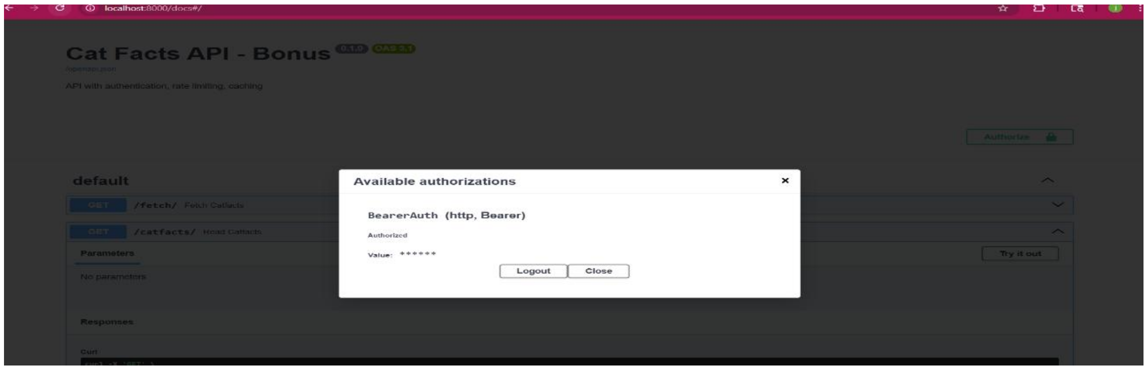
Key	Value
date	Sat, 13 Sep 2025 19:54:15 GMT
server	uvicorn
content-length	1188
content-type	application/json
cache-control	max-age=60
etag	W/-7007106851219830389
x-fastapi-cache	MISS

John Pretz   SWENG861   Prof: Santosh Nalubandhu   FALL Semester 2025

Bonus: Implementing Advanced Features

The bonus section focused on security and authentication. Sensitive endpoints such as POST, PUT, and DELETE operations were protected using API key authentication with HTTPBearer. Only users with the correct API key (mysecretkey123) can perform these operations, while GET endpoints remain publicly accessible. This ensures that only authorized clients can modify the database while still allowing read access for general users.

The following illustrations are for the Authentication test, demonstrating API key validation. The POST authentication process shows creating a fact with valid authentication.


localhost:8000/docs#/

Cat Facts API - Bonus

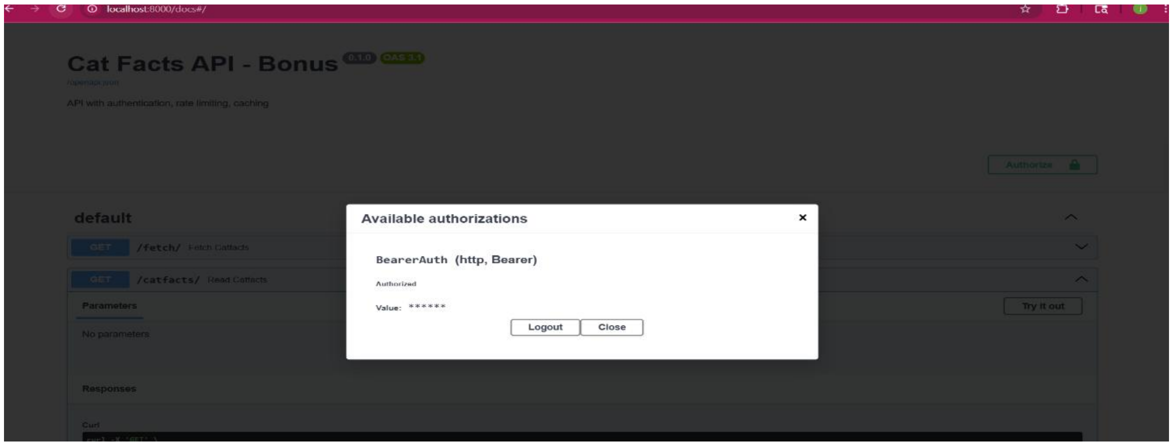
API with authentication, rate limiting, caching

Available authorizations

BearerAuth (http, Bearer)

Authorized

Value: ******


localhost:8000/docs#/

Cat Facts API - Bonus

API with authentication, rate limiting, caching

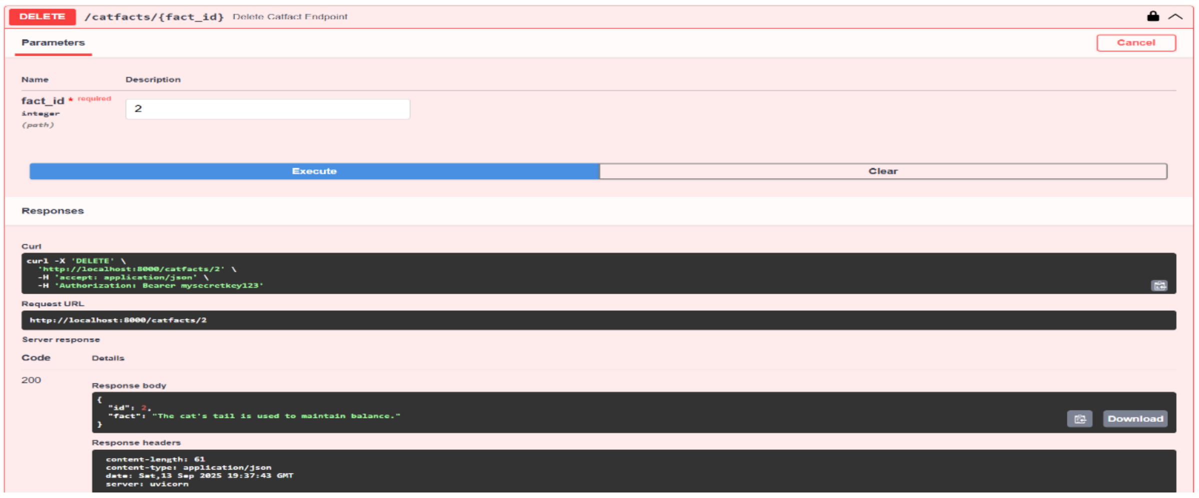
Available authorizations

BearerAuth (http, Bearer)

Authorized

Value: ******

Illustrates deletion with API key validation.



```
curl -X 'DELETE' \
  'http://localhost:8000/catfacts/2' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer mysecretkey123'
```

```
http://localhost:8000/catfacts/2
```

```
{
  "id": 2,
  "fact": "The cat's tail is used to maintain balance."
}
```

```
 content-length: 61
 content-type: application/json
 date: Sat,13 Sep 2025 19:37:43 GMT
 server: uvicorn
```

Conclusion:

The **Cat Facts API** demonstrates a complete, secure, and efficient RESTful web service. It integrates third-party data, supports full CRUD operations, and ensures performance with caching and rate-limiting. API key authentication adds a layer of security, making the system reliable and ready for real-world use.

Repository Name: https://github.com/Pretz/sweng861-2025-FALL-cats-api

References:

1. **FastAPI Documentation** – The official FastAPI documentation that helps in building modern web APIs with Python.
   - Link: https://fastapi.tiangolo.com/
2. **SQLAlchemy Documentation** – The official documentation for SQLAlchemy, used to manage SQLite database connections.
   - Link: https://www.sqlalchemy.org/
3. **Cat Facts API** – The 3rd-party API for fetching random cat facts, which was integrated into the project.
   - Link: https://catfact.ninja/
4. **Uvicorn** – A lightning-fast ASGI server for running FastAPI applications.
   - Link: https://www.uvicorn.org/
5. **FastAPI Cache** – The FastAPI extension used for caching responses to improve performance.
   - Link: https://github.com/long2ice/fastapi-cache
6. **GitHub** – Hosting platform used to manage, and version control this project.
   - Link: https://github.com/