**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](https://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](https://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.

A screenshot of a computer

AI-generated content may be incorrect.The following screenshot demonstrates the fetching of data from the third-party API.

**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.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**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.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**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.

A screenshot of a computer

AI-generated content may be incorrect.The following illustrations are for the Authentication test, demonstrating API key validation. The POST authentication process shows creating a fact with valid authentication.

A screenshot of a computer

AI-generated content may be incorrect.

Illustrates deletion with API key validation.

A screenshot of a computer

AI-generated content may be incorrect.

**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/>