

Source Code: AI Generated Report

Generated on: 2026-02-07 16:52:29

```
from fastapi import FastAPI, HTTPException

from pydantic import BaseModel

from sqlalchemy import create_engine, Column, Integer, String, BLOB

from sqlalchemy.ext.declarative import declarative_base

from sqlalchemy.orm import sessionmaker

from cryptography.fernet import Fernet

import pandas as pd

import uvicorn

import os


# Database setup

DATABASE_URL = "sqlite:///memory:"

engine = create_engine(DATABASE_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()


# Database model

class Password(Base):

    __tablename__ = 'passwords'

    id = Column(Integer, primary_key=True, index=True)

    service_name = Column(String, unique=True, index=True)

    encrypted_password = Column(BLOB)
```

```
Base.metadata.create_all(bind=engine)


# FastAPI app

app = FastAPI()


# Pydantic models

class PasswordCreate(BaseModel):

    service_name: str

    password: str


class StatsResponse(BaseModel):

    total_services: int

    average_password_length: float


# API endpoints

@app.post("/save")

def save_password(password_data: PasswordCreate):

    try:

        # Validate input data

        if not password_data.service_name or not password_data.password:

            raise HTTPException(status_code=400, detail="Invalid input data")


        # Generate a key and encrypt the password

        key = Fernet.generate_key()

        fernet = Fernet(key)

        encrypted_password = fernet.encrypt(password_data.password.encode())
```

```

# Save to database

db = SessionLocal()

        new_password    = Password(service_name=password_data.service_name,
encrypted_password=encrypted_password)

db.add(new_password)

db.commit()

db.refresh(new_password)

return {"status": "success"}

except Exception as e:

    raise HTTPException(status_code=500, detail=str(e))

finally:

    db.close()


@app.get("/stats", response_model=StatsResponse)

def get_stats():

    try:

        db = SessionLocal()

        passwords = db.query(Password).all()

        total_services = len(passwords)

        average_password_length = (sum(len(p.encrypted_password) for p in passwords) /
total_services) if total_services > 0 else 0

        return {"total_services": total_services, "average_password_length":
average_password_length}

    except Exception as e:

        raise HTTPException(status_code=500, detail=str(e))

```

```
finally:

    db.close()

# Test client

if __name__ == "__main__":

    os.environ['UVICORN_RELOAD'] = '1'    # Prevents port binding issues during
development

    if os.environ.get('UVICORN_RELOAD') == '1':

        print("Starting the FastAPI server...")

        uvicorn.run(app, host="127.0.0.1", port=8000, reload=True)

    else:

        print("Server is already running.")
```