GitHub FAQ Chatbot with Fallback Mechanism

TECHNICAL DOCUMENTATION

By Shivang Shukla
12214963

Github repo:

[shivangs107/github_chatbot: A telegram chatbot to answer github questions using sentence bert and T5.]

Table of Contents

Introduction

The GitHub FAQ Chatbot is a Telegram-based AI assistant designed to answer Git and GitHub-related queries using a pre-trained FAQ dataset and a fallback mechanism that retrieves real-time information from GitHub’s public API when no local match is found.

The system is fully containerized using Docker, ensuring scalability and easy deployment. It leverages:

- NLP models (Sentence-BERT, T5) for semantic search and answer enhancement.
- FAISS for fast similarity search.
- MongoDB for logging and analytics.

Project Overview

Objectives

- Provide instant, accurate answers to GitHub-related questions.
- Log all interactions for analytics and improvement.
- Fall back to GitHub's API when no local answer exists.
- Containerize the system for easy deployment.

Key Features

- Telegram Bot Interface – Users interact via Telegram.
- Semantic Search – Finds the best FAQ match using Sentence-BERT + FAISS.
- Answer Enhancement – Improves responses using T5.
- Fallback Mechanism – Fetches live data from GitHub API if no FAQ match.
- Logging & Analytics – Stores all queries in MongoDB.
- Dockerized Deployment – Runs in containers for scalability.

System Architecture

The system consists of three main services:

1. Telegram Bot (bot.py)
    - Listens to user queries.
    - Sends them to the FastAPI backend.
    - Displays responses in Markdown format.
2. FastAPI Backend (main.py)
    - Processes queries using NLP models.
    - Retrieves answers from FAISS-indexed FAQ dataset.
    - Enhances answers with T5.
    - Logs interactions in MongoDB.
3. MongoDB (logger.py)
    - Stores:
        - Query logs
        - User analytics
        - Conversation history

Key Components

1. Telegram Bot (bot.py)
   github_chatbot/bot.py at main · shivangs107/github_chatbot
   - Handles user interactions via Telegram.
   - Sends queries to the FastAPI backend.
   - Displays responses with Markdown formatting.

   Key Functions:

   - start() – Welcomes users.
   - handle_message() – Processes user input, calls API, and formats responses.
2. FastAPI Backend (main.py)
   github_chatbot/app/main.py at main · shivangs107/github_chatbot
   - Loads NLP models (Sentence-BERT, T5) at startup.
   - Searches FAQ dataset using FAISS.
   - Enhances answers with T5 (minor improvements).
   - Logs queries via logger.py.

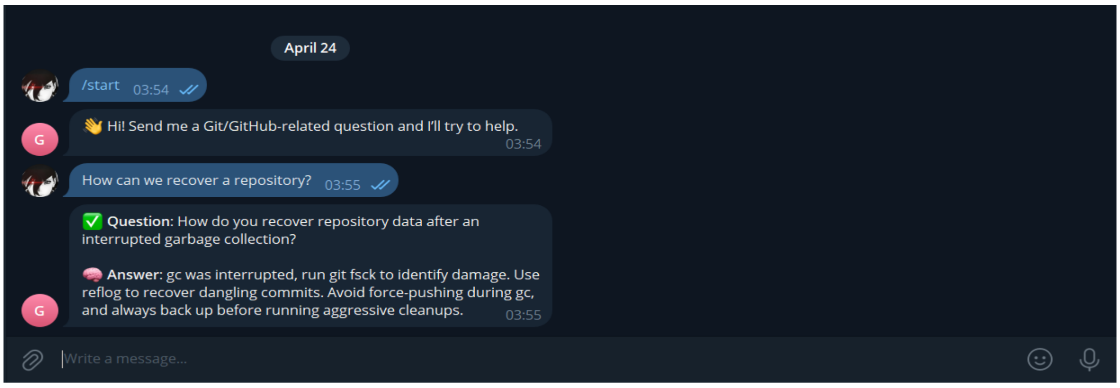
   Key Endpoints:

   - / – Health check.
   - /query – Processes user questions.

3. MongoDB (logger.py)
   github_chatbot/app/logger.py at main · shivangs107/github_chatbot
   - Logs queries in 4 collections:
     - logs – Basic query info.
     - conversations – Full chat history.
     - users – User-specific analytics.
     - analytics – Daily query counts.
4. FAQ Indexing (faq_index.py)
   github_chatbot/app/faq_index.py at main · shivangs107/github_chatbot
   - Generates embeddings using Sentence-BERT.
   - Builds a FAISS index for fast similarity search.
   - Saves lookup files (faq_lookup.json, faq_index.index).
5. Dockerization (Dockerfile, docker-compose.yaml)
   github_chatbot/docker-compose.yaml at main · shivangs107/github_chatbot
   - Multi-stage Docker build (reduces image size).
   - Three services:
     - api (FastAPI backend).
     - bot (Telegram bot).
     - mongo (MongoDB).

Chatbot Workflow

1. User sends a query via Telegram (bot.py).



April 24

/start 03:54

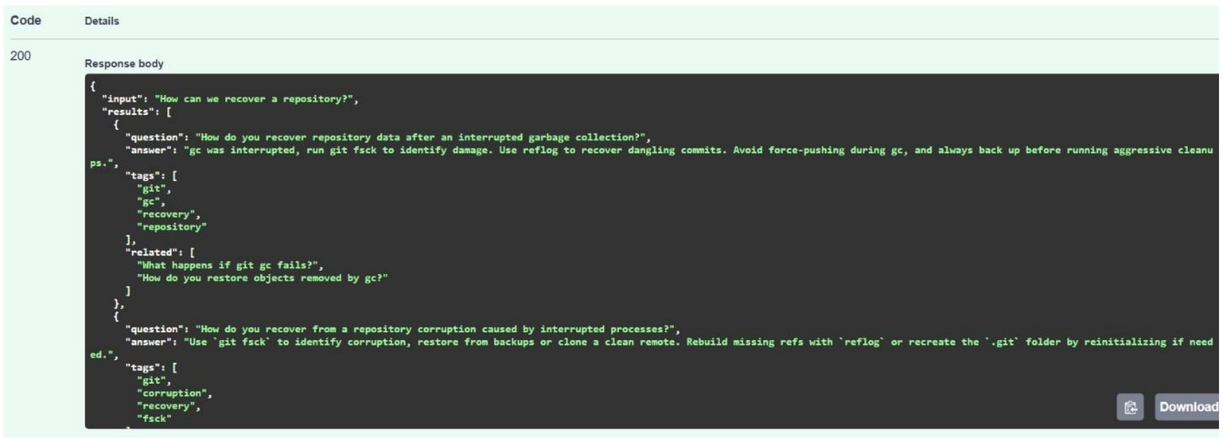
👋 Hi! Send me a Git/GitHub-related question and I'll try to help. 03:54

How can we recover a repository? 03:55

✅ **Question:** How do you recover repository data after an interrupted garbage collection?

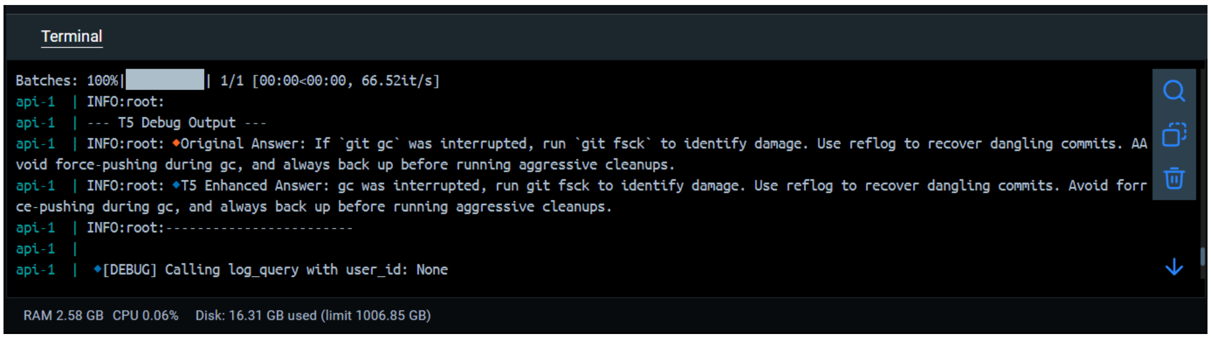
🧠 **Answer:** gc was interrupted, run git fsck to identify damage. Use reflog to recover dangling commits. Avoid force-pushing during gc, and always back up before running aggressive cleanups. 03:55

2. Bot forwards query to FastAPI (/query endpoint).



```
{
  "input": "How can we recover a repository?",
  "results": [
    {
      "question": "How do you recover repository data after an interrupted garbage collection?",
      "answer": "gc was interrupted, run git fsck to identify damage. Use reflog to recover dangling commits. Avoid force-pushing during gc, and always back up before running aggressive cleanu
ps.",
      "tags": [
        "git",
        "gc",
        "recovery",
        "repository"
      ],
      "related": [
        "What happens if git gc fails?",
        "How do you restore objects removed by gc?"
      ]
    },
    {
      "question": "How do you recover from a repository corruption caused by interrupted processes?",
      "answer": "Use `git fsck` to identify corruption, restore from backups or clone a clean remote. Rebuild missing refs with `reflog` or recreate the `.git` folder by reinitializing if need
ed.",
      "tags": [
        "git",
        "corruption",
        "recovery",
        "fsck"
```

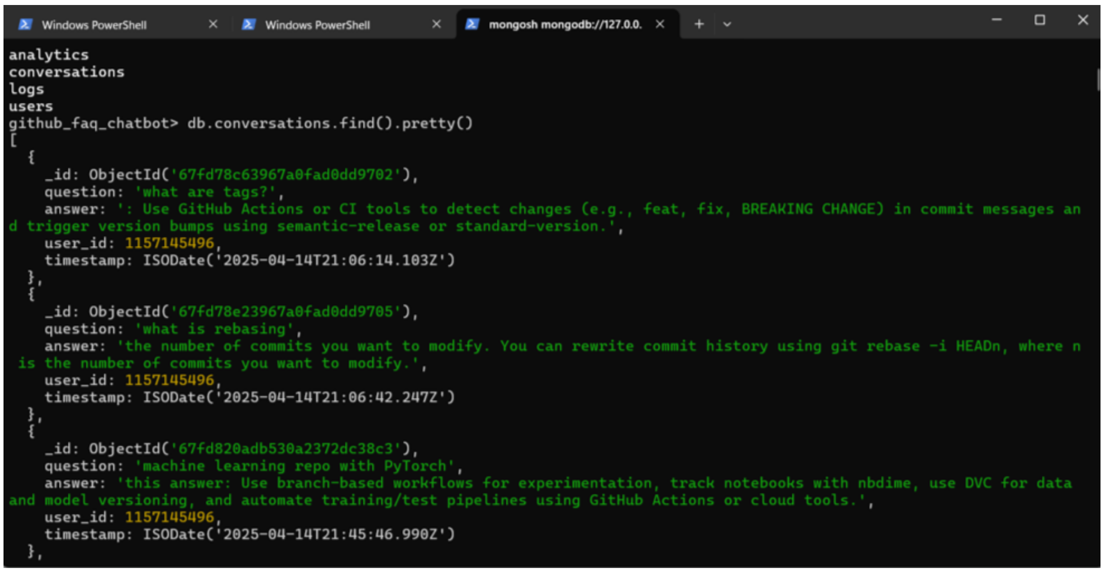
3. Backend processes query:
   - Encodes question using Sentence-BERT.
   - Searches FAISS index for closest FAQ match.
   - Enhances answer with T5.



```
Batches: 100%|██████████| 1/1 [00:00<00:00, 66.52it/s]
api-1  | INFO:root:
api-1  | --- T5 Debug Output ---
api-1  | INFO:root: ♦Original Answer: If `git gc` was interrupted, run `git fsck` to identify damage. Use reflog to recover dangling commits. AA
void force-pushing during gc, and always back up before running aggressive cleanups.
api-1  | INFO:root: ♦T5 Enhanced Answer: gc was interrupted, run git fsck to identify damage. Use reflog to recover dangling commits. Avoid forr
ce-pushing during gc, and always back up before running aggressive cleanups.
api-1  | INFO:root:-----------------------
api-1  |
api-1  |  ♦[DEBUG] Calling log_query with user_id: None
```

RAM 2.58 GB CPU 0.06% Disk: 16.31 GB used (limit 1006.85 GB)

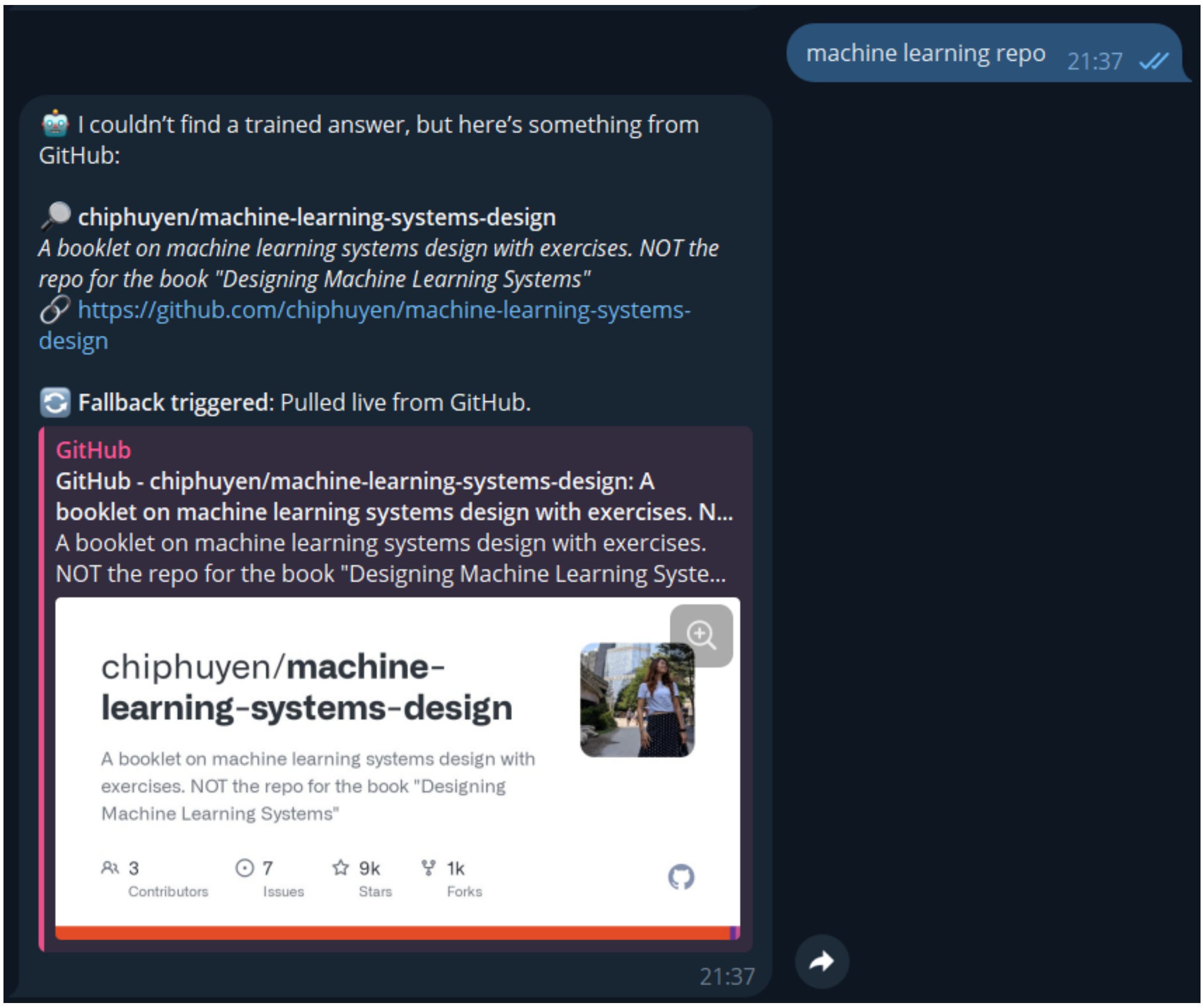
4. If no match found, triggers fallback mechanism (GitHub API).
5. Response sent back to Telegram.
6. All interactions logged in MongoDB.



```
analytics
conversations
logs
users
github_faq_chatbot> db.conversations.find().pretty()
[
  {
    _id: ObjectId('67fd78c63967a0fad0dd9702'),
    question: 'what are tags?',
    answer: ': Use GitHub Actions or CI tools to detect changes (e.g., feat, fix, BREAKING CHANGE) in commit messages an
d trigger version bumps using semantic-release or standard-version.',
    user_id: 1157145496,
    timestamp: ISODate('2025-04-14T21:06:14.103Z')
  },
  {
    _id: ObjectId('67fd78e23967a0fad0dd9705'),
    question: 'what is rebasing',
    answer: 'the number of commits you want to modify. You can rewrite commit history using git rebase -i HEADn, where n
 is the number of commits you want to modify.',
    user_id: 1157145496,
    timestamp: ISODate('2025-04-14T21:06:42.247Z')
  },
  {
    _id: ObjectId('67fd820adb530a2372dc38c3'),
    question: 'machine learning repo with PyTorch',
    answer: 'this answer: Use branch-based workflows for experimentation, track notebooks with nbdime, use DVC for data
and model versioning, and automate training/test pipelines using GitHub Actions or cloud tools.',
    user_id: 1157145496,
    timestamp: ISODate('2025-04-14T21:45:46.990Z')
  },
```

Fallback Mechanism

1. Triggered when no FAQ match is found.
2. Queries GitHub API for:
   - Repository names.
   - Descriptions.
   - URLs.
3. Returns a fallback message indicating live data.



machine learning repo 21:37

🤖 I couldn't find a trained answer, but here's something from GitHub:

🔎 **chiphuyen/machine-learning-systems-design**
A booklet on machine learning systems design with exercises. NOT the repo for the book "Designing Machine Learning Systems"
🔗 https://github.com/chiphuyen/machine-learning-systems-design

🔄 **Fallback triggered:** Pulled live from GitHub.

GitHub
GitHub - chiphuyen/machine-learning-systems-design: A booklet on machine learning systems design with exercises. N...
A booklet on machine learning systems design with exercises. NOT the repo for the book "Designing Machine Learning Syste...

chiphuyen/**machine-learning-systems-design**

A booklet on machine learning systems design with exercises. NOT the repo for the book "Designing Machine Learning Systems"

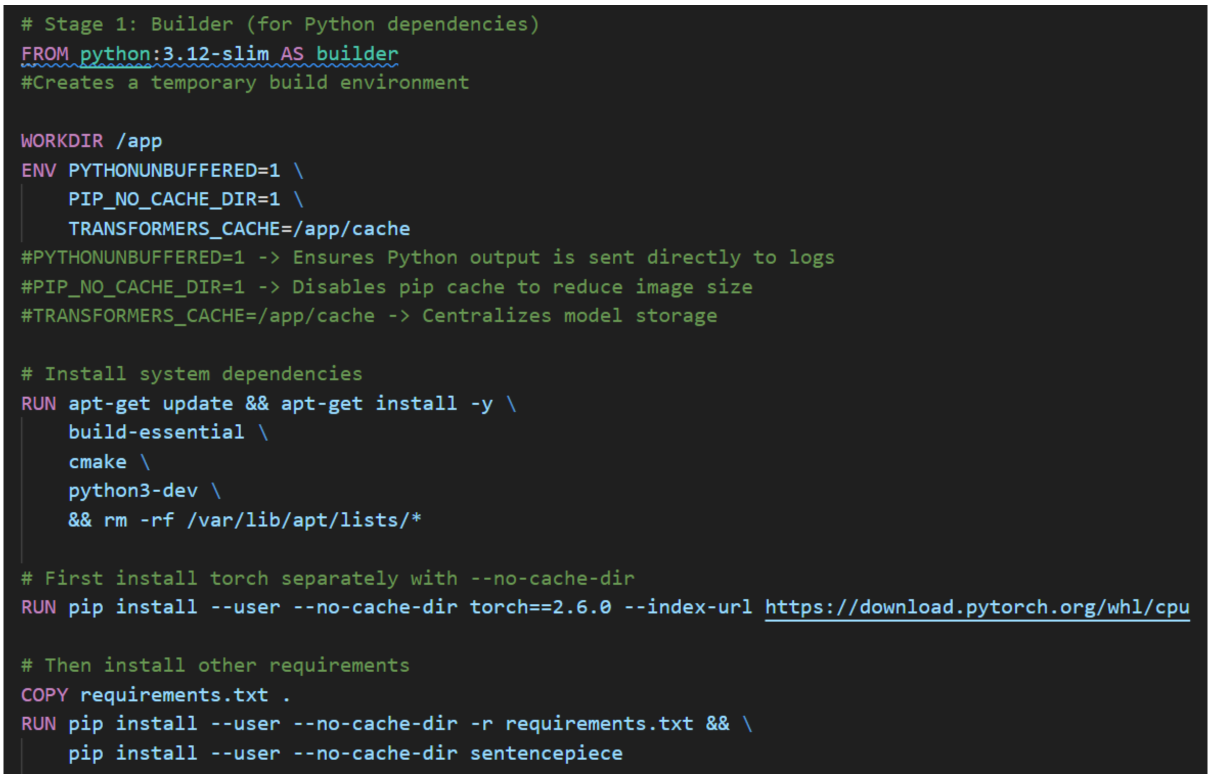
3 Contributors · 7 Issues · 9k Stars · 1k Forks

21:37

Docker Setup & Deployment

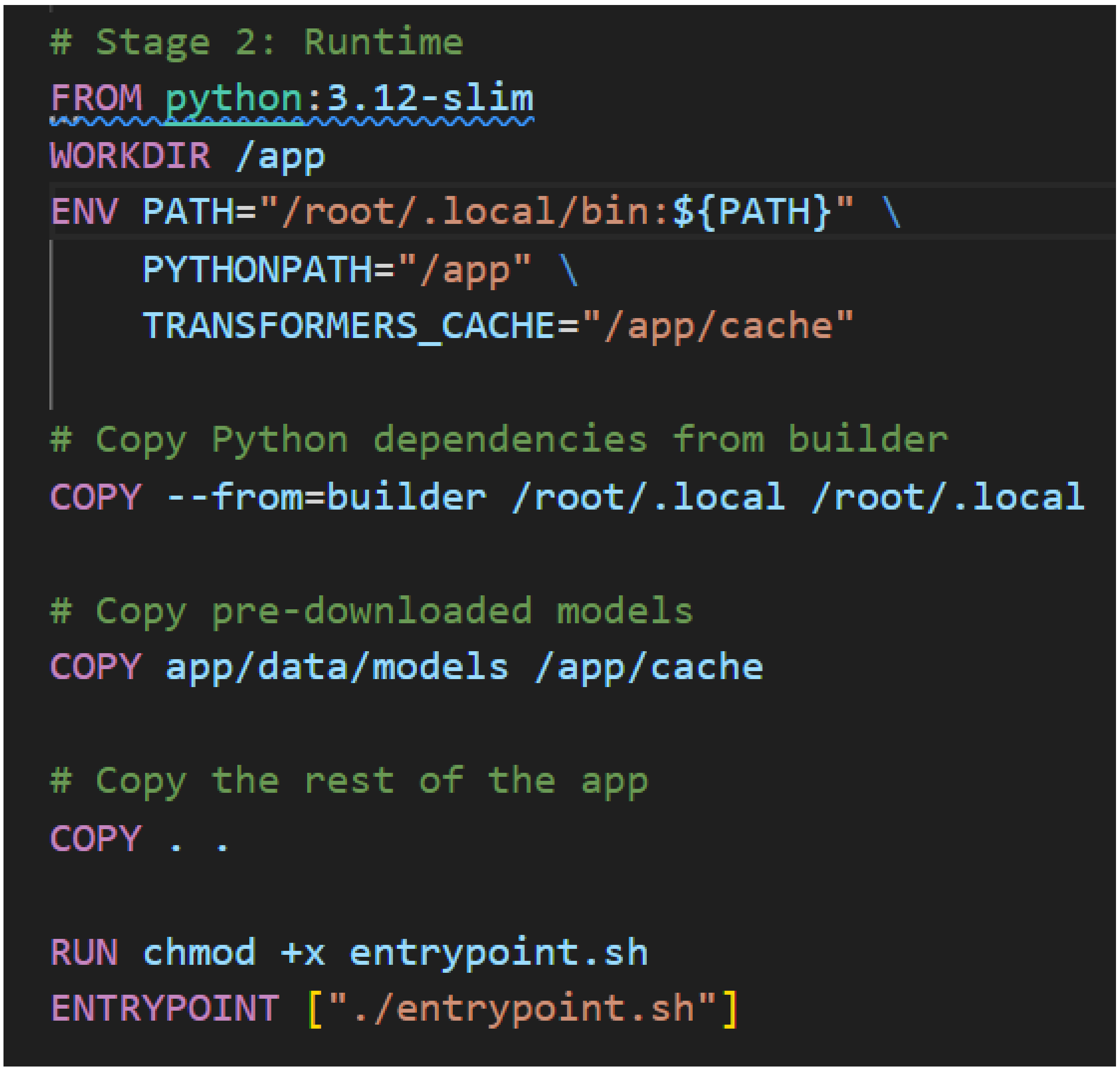
Dockerfile (Multi-Stage Build)

1. Builder Stage:
    - Installs Python dependencies (torch, sentence-transformers).
    - Downloads models (all-MiniLM-L6-v2, t5-small).



```
# Stage 1: Builder (for Python dependencies)
FROM python:3.12-slim AS builder
#Creates a temporary build environment

WORKDIR /app
ENV PYTHONUNBUFFERED=1 \
    PIP_NO_CACHE_DIR=1 \
    TRANSFORMERS_CACHE=/app/cache
#PYTHONUNBUFFERED=1 -> Ensures Python output is sent directly to logs
#PIP_NO_CACHE_DIR=1 -> Disables pip cache to reduce image size
#TRANSFORMERS_CACHE=/app/cache -> Centralizes model storage

# Install system dependencies
RUN apt-get update && apt-get install -y \
    build-essential \
    cmake \
    python3-dev \
    && rm -rf /var/lib/apt/lists/*

# First install torch separately with --no-cache-dir
RUN pip install --user --no-cache-dir torch==2.6.0 --index-url https://download.pytorch.org/whl/cpu

# Then install other requirements
COPY requirements.txt .
RUN pip install --user --no-cache-dir -r requirements.txt && \
    pip install --user --no-cache-dir sentencepiece
```

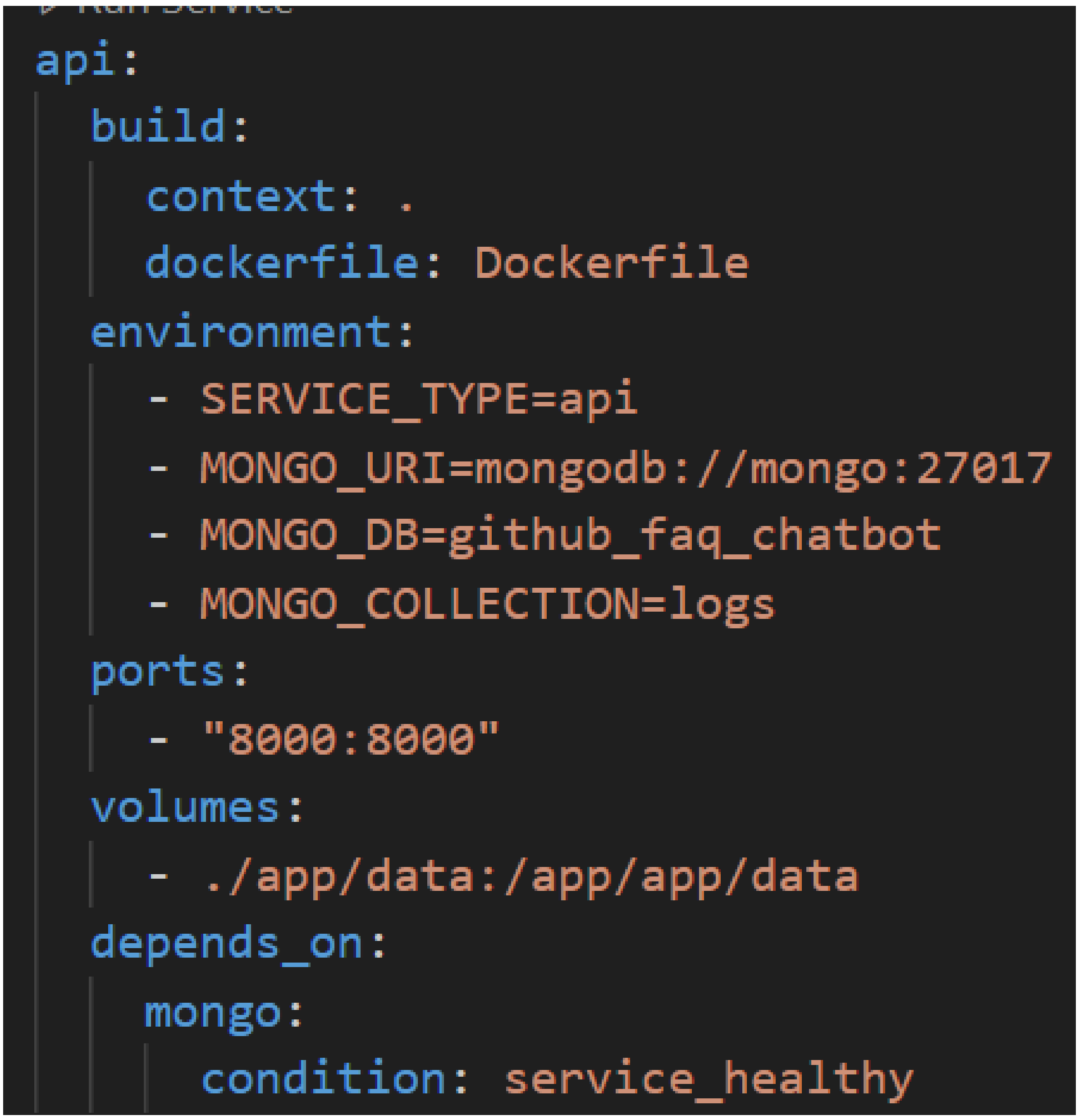
2. Runtime Stage:
   - Copies only necessary files.
   - Sets up entrypoint (entrypoint.sh).



```dockerfile
# Stage 2: Runtime
FROM python:3.12-slim
WORKDIR /app
ENV PATH="/root/.local/bin:${PATH}" \
    PYTHONPATH="/app" \
    TRANSFORMERS_CACHE="/app/cache"

# Copy Python dependencies from builder
COPY --from=builder /root/.local /root/.local

# Copy pre-downloaded models
COPY app/data/models /app/cache

# Copy the rest of the app
COPY . .

RUN chmod +x entrypoint.sh
ENTRYPOINT ["./entrypoint.sh"]
```

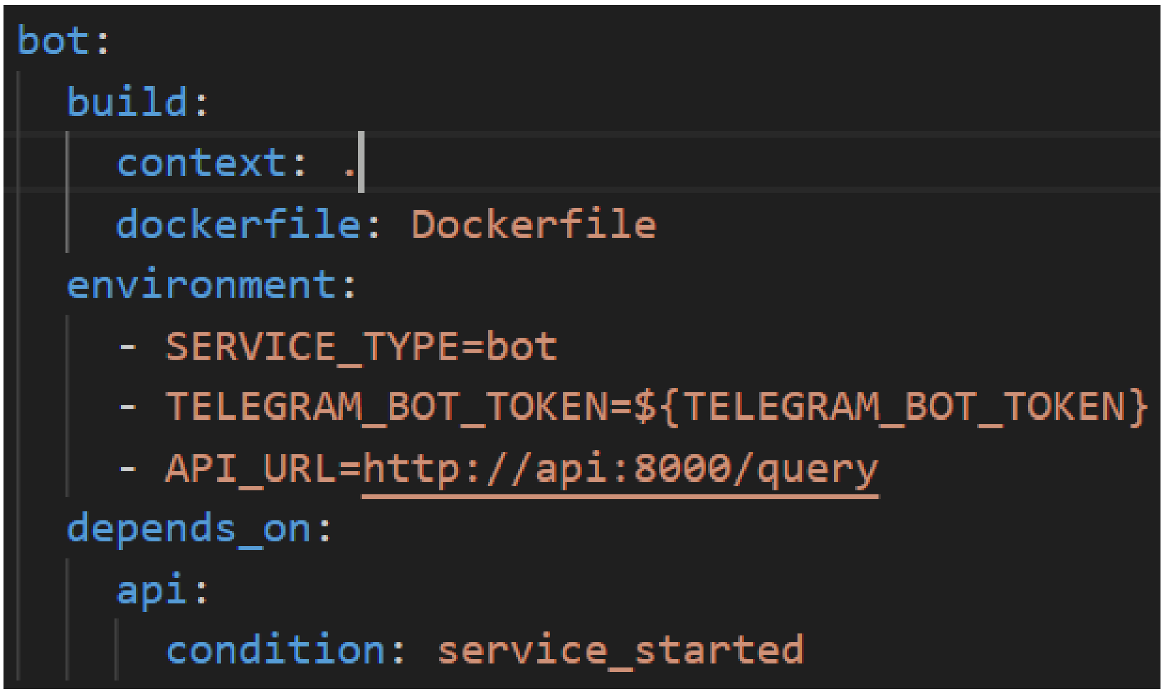
Docker-compose.yaml

1. Orchestrates 3 services:
   - api (FastAPI on port 8000).



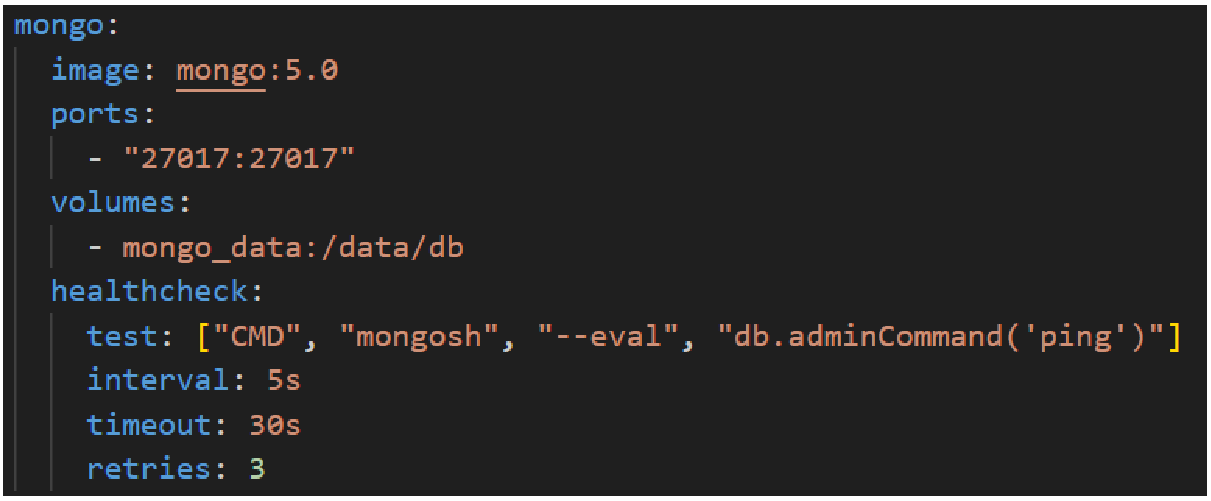
```yaml
api:
  build:
    context: .
    dockerfile: Dockerfile
  environment:
    - SERVICE_TYPE=api
    - MONGO_URI=mongodb://mongo:27017
    - MONGO_DB=github_faq_chatbot
    - MONGO_COLLECTION=logs
  ports:
    - "8000:8000"
  volumes:
    - ./app/data:/app/app/data
  depends_on:
    mongo:
      condition: service_healthy
```

- bot (Telegram bot).



```yaml
bot:
  build:
    context: .
    dockerfile: Dockerfile
  environment:
    - SERVICE_TYPE=bot
    - TELEGRAM_BOT_TOKEN=${TELEGRAM_BOT_TOKEN}
    - API_URL=http://api:8000/query
  depends_on:
    api:
      condition: service_started
```

- mongo (MongoDB on port 27017).



```yaml
mongo:
  image: mongo:5.0
  ports:
    - "27017:27017"
  volumes:
    - mongo_data:/data/db
  healthcheck:
    test: ["CMD", "mongosh", "--eval", "db.adminCommand('ping')"]
    interval: 5s
    timeout: 30s
    retries: 3
```

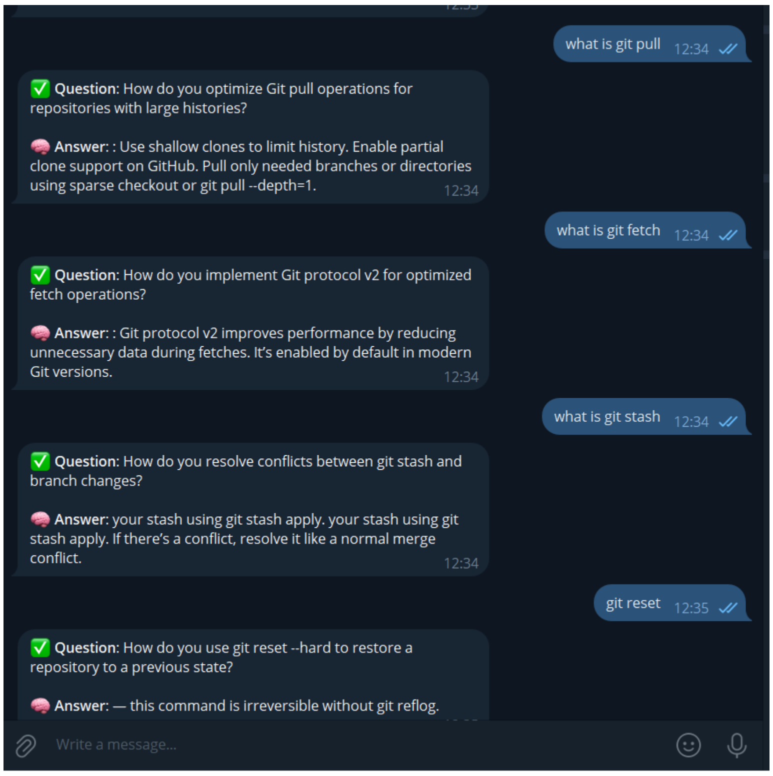
2. Deployment Steps:
   - docker-compose build
   - docker-compose up

Logging & Analytics

1. Stores in MongoDB:
    - User queries (logs).
    - Full conversations (conversations).
    - User activity (users).
    - Daily stats (analytics).

Current Limitations

1. Not always accurate (depends on FAQ dataset).
2. No follow-up question handling.
3. Mixed questions not supported (only first query processed).



what is git pull 12:34

✅ **Question**: How do you optimize Git pull operations for repositories with large histories?

🧠 **Answer**: : Use shallow clones to limit history. Enable partial clone support on GitHub. Pull only needed branches or directories using sparse checkout or git pull --depth=1. 12:34

what is git fetch 12:34

✅ **Question**: How do you implement Git protocol v2 for optimized fetch operations?

🧠 **Answer**: : Git protocol v2 improves performance by reducing unnecessary data during fetches. It's enabled by default in modern Git versions. 12:34

what is git stash 12:34

✅ **Question**: How do you resolve conflicts between git stash and branch changes?

🧠 **Answer**: your stash using git stash apply. your stash using git stash apply. If there's a conflict, resolve it like a normal merge conflict. 12:34

git reset 12:35

✅ **Question**: How do you use git reset --hard to restore a repository to a previous state?

🧠 **Answer**: — this command is irreversible without git reflog.

Future Improvements

1. Train FAQ model on more GitHub-specific data.
2. Integrate StackOverflow/GitLab APIs for broader fallback.
3. Add follow-up question handling.
4. Personalize responses based on user history.

Conclusion

1. Successfully built a Dockerized Telegram chatbot for GitHub FAQs.
2. Uses NLP + FAISS for semantic search and T5 for answer enhancement.
3. Fallback mechanism ensures live data retrieval.
4. Future improvements will enhance accuracy and usability.