T.C. MALTEPE UNIVERSITY FACULTY OF ENGINEERING AND NATURAL SCIENCES DEPARTMENT OF SOFTWARE ENGINEERING

SE40301 Software Project Management Project

Project Report

GiggleLab: Artificial Intelligence Based Joke Generator

11 May 2025

Instructor: Prof. Dr. Ensar Gül
ensargul@maltepe.edu.tr

Prepared By:

Student Name	Student ID
Berkay Şahin	21 07 06 034
Selen Bingöl	22 07 06 17
Beyza Çelebi	20 07 06 031
Mustafa Öztürk	21 07 06 016

1. Scope of the project

1.1 Project Objective

The objective of the GiggleLab project is to develop a culturally sensitive, AI-assisted joke generation platform that delivers humorous Turkish jokes through a browser-based interface. The aim is to explore how AI can enhance entertainment in natural language generation without relying on large-scale LLM APIs.

1.2 Project Components and Goals

- **Frontend:** Developed in HTML, CSS, JavaScript – responsible for user interaction
- **Backend:** Built with Python – serves jokes and controls logic
- **Dataset:** Curated collection of Turkish jokes in fikra_dataset.json
- **Audio Module:** Randomized .mp3 laughter playback upon joke delivery
- **Objective:** Provide an accessible, engaging, and culturally relevant humor experience

2. Functionality of the project

2.1 Core Functionalities

- Display Turkish jokes from a structured JSON dataset
- Play one of multiple random laugh audio clips
- Simple and responsive interface with a click-to-generate-joke button
- Compatible with modern desktop and mobile browsers

2.2 Unique or Distinguishing Features

- Offline dataset instead of LLM or API integration
- Multi-sound audio response for engaging UX
- Clean and ethical content filtering in dataset preparation
- Fully frontend-driven design for speed and simplicity

2.3. The Functions Available at the Beginning of the Project

- Basic HTML and CSS layout
- Static joke display (no backend integration)
- Single audio file manually linked to the joke

2.4. Functions Added During the Development

- Full JSON dataset integration via Python backend
- Random audio player for laugh tracks
- UI styling with custom CSS animations
- Input/output sanitation and error handling
- Compatibility enhancements for multiple browsers

3. Missing Parts

- **Use of GPT-4 API**: Initially considered, but omitted to avoid dependency on third-party APIs
- **Prompt Input from the User**: Fixed joke delivery; no open-ended prompt functionality added
- **Joke Categories or Themes**: All jokes presented randomly; no categorization or filtering by type
- **User Feedback or Joke Storage via Database**: No backend DB integration for storing or rating content

4. Design Documents

4.1 Use Case Diagram

- **Actor:** User
- **Use Cases:** Generate joke, Hear laugh, Interact with button

4.2 Sequence Diagram

- User → Button → JS → Python script → Fetch joke/audio → Return joke/audio → Display

4.3 Activity Dıagram (Backend Logic)

- Button click → Random joke retrieval → Audio selection → Response delivery

4.4 Frontend Interface (UI Design)

- HTML/CSS structure with button element, styled joke area, and embedded audio functionality

5. Deployment

5.1 System Requirements

Any browser, Python 3.8+, Any OS, No GPU needed

5.2 Project Structure Overview

5.3 Backend Setup

5.3.1 Clone the Repository

5.3.2 Install Required Packages

5.3.3 Login to Hugging Face

5.3.4 Run the FastAPI Server

5.4 Frontend Setup

- Open `index.html` in any browser
- No build tools or compilation required

5.5 Model Deployment Logic

No external model used; joke selection and logic handled internally via Python scripts

5.6 Additional Notes

5.7 requirements.txt

6. Responsibilities for each iteration

Write the tasks and responsibilities for each iteration in the following table. Also provide graphics which show estimated times and actual implementation times.

6.1 Iteration Table

iter no/ developer	Berkay Şahin	Selin Bingöl	Beyza Çelebi	Mustafa Öztürk
Iter 1				
Iter 2				
Iter 3				
Iter 4				

6.2 Estimated vs Actual Effort

[Trello]

7. Risk management

7.1 Risk Table:

D	Risk Description	Likelihood	Impact	Risk Level	Mitigation Strategy
R1	Dataset contains biased or low-quality jokes	Medium	High	High	Enhance data preprocessing and gather jokes from diverse, verified sources
R2	Model generates inappropriate/offensive jokes	Medium	High	High	Apply filtering algorithms and moderation flags
R3	Generated jokes are not funny or meaningful	High	Medium	High	Improve dataset structure and use feedback-based review
R4	Frontend crashes on certain browsers	Low	Medium	Medium	Cross-browser testing and responsive CSS design
R5	Backend responds slowly or halts	Medium	Medium	Medium	Optimize script logic and monitor system resource usage
R6	Legal issues from copyrighted joke content	Low	High	Medium	Use only public domain or ethically cleared data
R7	Malicious user input affects display	Medium	High	High	Sanitize inputs and validate frontend/backend requests
R8	Audio playback fails on mobile	Medium	Medium	Medium	Convert to MP3 and test on common mobile browsers
R9	Team miscommunication	Medium	Medium	Medium	Weekly check-ins and Trello usage

8. Tests

8.1 Test Case Table

Test Case ID	Description	Expected Result	Status
TC-01	Button click generates a joke	New joke displayed from dataset	Pass
TC-02	Audio playback after joke	One random laugh sound plays	Pass
TC-03	Page loads in Chrome and Firefox	UI renders properly and functions as expected	Pass
TC-04	Long joke formatting	Text wraps or scrolls appropriately	Pass
TC-05	Backend handles request	No crash, returns expected JSON structure	Pass

TC-06	Empty or spam click handling	UI stays stable; joke does not repeat constantly	Pass
TC-07	HTML/CSS responsive design	Layout adjusts on mobile screens	Pass

9. Experience Gained

9.1 Berkay Şahin

9.2 Selin Bingöl

9.3 Beyza Çelebi:

I was responsible for the frontend development of the project. I began by determining the essential requirements and conducted research accordingly. Using Figma, I designed the layout and user interface of the application, learning how to translate conceptual designs into practical web components. This process helped me understand how a project layout should be constructed and allowed me to visualize and plan the interface effectively. I then converted these Figma designs into working code using HTML, CSS, and JavaScript. This project significantly enhanced my ability to convert UI prototypes into real-world interfaces and gave me insight into integrating frontend elements with Python-based backends.

9.4 Mustafa Öztürk

11. Project Repository

Github

12. References