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

SE40301 Software Project Management Project

Risk 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 06 017
Beyza Çelebi	20 07 06 031
Mustafa Öztürk	21 07 06 016

TABLE OF CONTENT

1. Introduction

The GiggleLab project is an AI-based joke generator that uses a custom Turkish joke dataset (`fikra_dataset.json`) and a Python-based backend to generate culturally appropriate and humorous content. The frontend was developed using HTML, CSS, and JavaScript, while the backend logic and model interactions are written in Python.

2. Risk Identification

The team has identified the following categories of risks:

- **Data and Model Risks**
  - Low-quality or imbalanced dataset
  - Generation of inappropriate or non-humorous content
  - Intellectual property issues from dataset content
- **Application-Level Risks**
  - Frontend crashes or responsiveness problems
  - Backend latency or model freezing
  - Inadequate response to user inputs
- **Security and Privacy Risks**
  - User data misuse or leakage
  - Inappropriate user-generated inputs affecting the model
- **Infrastructure and Deployment Risks**
  - Server inaccessibility or downtime
  - High memory or CPU usage during peak load
- **Project Execution Risks**
  - Team coordination breakdowns
  - Incompatible software versions
  - Logical errors during development

- **Data Consistency Risks**
    - Repetition or semantic duplication in jokes
    - Lack of diversity in humor types (regional/cultural variance)
- **Script-Level Risks**
    - Instability in model response from fikra_llm.py
    - Blocking or latency caused by main.py during API inference
- **Prompt Filtering Risks**
    - Inappropriate user inputs passed directly into model

3. Risk Assessment Table

ID	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 (e.g., sexist, political)	Medium	High	High	Apply filtering algorithms; test with sample users; implement moderation flags
R3	Generated jokes are not funny or meaningful	High	Medium	High	Improve model quality via fine-tuning and human-in-the-loop evaluations
R4	Frontend crashes due to browser or layout bugs	Low	Medium	Medium	Conduct cross-browser testing and use responsive web design practices
R5	Model responds slowly or stops during peak usage	Medium	Medium	Medium	Optimize model response time; monitor backend performance with APM tools

R6	Legal issues from using copyrighted joke content	Low	High	Medium	Use only public domain or explicitly permitted joke datasets
R7	Exposure to malicious or offensive user inputs	Medium	High	High	Apply input validation and profanity filtering before passing to the model
R8	Security vulnerabilities in data handling	Medium	High	High	Ensure secure HTTP protocols, encrypt sensitive data, and limit data retention
R9	Backend server is unreachable or crashes	Medium	Medium	Medium	Set up backup servers or fallback responses on UI
R10	Logical errors during coding cause system bugs	Medium	High	High	Implement regular code reviews and automated unit testing
R11	Tool or library version conflicts across team	Medium	Medium	Medium	Use version control and document all project dependencies
R12	Team miscommunication or unclear task delegation	Medium	Medium	Medium	Use tools like Trello, GitHub issues, and regular stand-ups

Table 1: Risk Table

4. Risk Mitigation Strategies

Each risk is actively managed with the following actions:

- **Model & Data Risks**:
  - Perform ethical and humor validation steps
  - Apply filtering layers and content classifiers
- **UI/UX Issues**:
  - Run UI tests on multiple devices and screen sizes
  - Include feedback collection mechanisms for users
- **Technical Infrastructure**:
  - Monitor CPU/RAM usage and optimize endpoints

    - Ensure high availability via Docker and load balancing
- **Security**:
    - Sanitize inputs, follow GDPR-like policies for data
    - Perform penetration testing simulations
- **Team Workflow**:
    - Weekly status meetings
    - Clearly documented project responsibilities and task assignments

5. Monitoring & Reporting

The team uses the following process for ongoing risk monitoring:

- Weekly risk status reviews in group meetings
- Trello board updates and GitHub issue tracking
- Post-sprint feedback analysis for new emerging risks
- Clear documentation of resolved and new risks

6. Conclusion

By systematically managing potential risks, the *GiggleLab* team ensures a robust and responsible approach to delivering a fun, safe, and technically sound product. Regular reviews, proactive planning, and documented strategies strengthen the project's resilience against unforeseen issues.