Faculty of Engineering and Natural Sciences

Turkish Joke Generator

Prepared by

Furkan Aksoy ~ 210706029

Emre Sarı ~ 220706304

Mehmet Güzel ~ 210706030

Tamay Yazgan ~ 210706022

Ömer Faruk Özer ~ 210706028

Yaren Yıldız ~ 200706040

Table Of Contents

Software Requirements Specification (SRS) 3

- **1. Introduction** 3
  - **1.1 Purpose** 3
  - **1.2 Scope** 3
  - **1.2 Definitions, Acronyms, and Abbreviations** 4
  - **1.3 References** 4
- **2. Overall Description** 4
  - **2.1 Product Perspective** 4
  - **2.2 Product Functions** 5
  - **2.3 User Classes and Characteristics** 5
  - **2.4 Operating Environment** 5
  - **2.5 Design and Implementation Constraints** 5
  - **2.6 Assumptions and Dependencies** 6
- **3. Specific Requirements** 6
  - **3.1 Functional Requirements** 6
  - **3.2 Non-Functional Requirements** 7
  - **3.3 System Interfaces** 9
- **4. Appendices** 9
  - **4.1 Glossary** 9
  - **4.2 Document Revision History** 10

Software Requirements Specification (SRS)

Turkish Joke Generator

Project Duration: February 27, 2025 – April 24, 2025

Team Members:

- Furkan Aksoy
- Emre Sarı
- Tamay Yazgan
- Ömer Faruk Özer (Scrum Master)
- Mehmet Güzel
- Yaren Yıldız

  Advisor: Prof. Ensar Gül

1. Introduction

1.1 Purpose

This document provides a comprehensive set of requirements for the Turkish Joke Generator—a web application that leverages a fine-tuned GPT-2 model to generate random Turkish jokes. It serves as the foundation for design, development, testing, and eventual deployment of the system.

1.2 Scope

The Turkish Joke Generator is designed to deliver culturally rich, humorous content on demand. Users will be able to select a joke category (e.g., Nasreddin Hodja, Temel-Dursun, or general jokes) and generate a random joke through an intuitive web interface. The backend is built with Flask and integrated with a GPT-2 model trained on a dataset of 10,000 Turkish jokes via Google Colab.

1.2 Definitions, Acronyms, and Abbreviations

- **LLM:** Large Language Model
- **GPT-2:** Generative Pre-trained Transformer 2
- **API:** Application Programming Interface
- **UI/UX:** User Interface/User Experience
- **SRS:** Software Requirements Specification
- **REST:** Representational State Transfer

1.3 References

- Turkish Joke Generator Vision Scope Document
- Agile and Scrum best practices (Schwaber & Sutherland, The Scrum Guide)
- Flask, GitHub, and Google Colab official documentation

2. Overall Description

2.1 Product Perspective

The Turkish Joke Generator is a standalone web application that operates independently while utilizing a locally hosted GPT-2 model. It is designed as a single-page application that interacts with a RESTful API to deliver generated jokes. The system will run on standard web browsers and be deployed on a server environment that supports Python and Flask.

2.2 Product Functions

The application will perform the following functions:

- Generate random Turkish jokes on user request.
- Allow users to choose from different joke categories.
- Display generated jokes in a user-friendly format.
- Provide a RESTful API for joke retrieval.
- Accept and log user feedback and error reports.

2.3 User Classes and Characteristics

- **General Users:** Casual users seeking a quick laugh with minimal technical knowledge.
- **Cultural Enthusiasts:** Users interested in traditional Turkish humor.
- **Administrators/Developers:** Team members responsible for maintaining, updating, and debugging the system.

2.4 Operating Environment

- **Client Side:** Modern web browsers (Chrome, Firefox, Safari, Edge).
- **Server Side:** A Python environment running Flask on a Linux/Windows/Mac server.
- **Model Training:** Google Colab for initial training of the GPT-2 model.

2.5 Design and Implementation Constraints

- The system must be developed using Flask, HTML, CSS, and JavaScript.
- Model training and fine-tuning are restricted to Google Colab resources.
- Deployment must consider resource limitations (e.g., GPU availability).
- The project must adhere to the timeline (February 27 – April 24, 2025).

2.6 Assumptions and Dependencies

- The provided dataset (10,000 Turkish jokes) is comprehensive and sufficient.
- GPT-2 can be fine-tuned effectively for joke generation.
- Team members possess the necessary technical expertise.
- External collaboration tools (GitHub, Trello, WhatsApp) will remain operational.
- The system will run on a standard web server with internet access.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Joke Generation

- **FR1:** The system shall provide a "Get Random Joke" button that, when clicked, triggers the GPT-2 model to generate a joke.
- **FR2:** The system shall generate jokes in Turkish that are culturally coherent and contextually relevant.
- **FR3:** The system shall support multiple joke categories including Nasreddin Hodja, Temel-Dursun, and general jokes.
- **FR4:** The system shall allow users to select a desired joke category prior to generating a joke.
- **FR5:** The system shall display the generated joke in a clear and readable format on the user interface.

3.1.2 API Integration

- **FR6:** The system shall expose a RESTful API endpoint for retrieving generated jokes.
- **FR7:** The API shall accept parameters for joke category selection.

- **FR8:** The API shall return the generated joke along with metadata (timestamp, category) in JSON format.

3.1.3 User Interface

- **FR9:** The application shall feature a responsive design adaptable to desktop and mobile devices.
- **FR10:** The user interface shall include clear instructions, visual feedback (e.g., loading animations), and error notifications.
- **FR11:** The system shall provide a mechanism for users to easily navigate between different sections (e.g., joke generation, feedback submission).

3.1.4 Feedback and Error Reporting

- **FR12:** The system shall provide an interface for users to submit feedback and report errors.
- **FR13:** The feedback module shall capture user comments along with any error details.
- **FR14:** The system shall store feedback securely for review by administrators.

3.2 Non-Functional Requirements

3.2.1 Performance

- **NFR1:** The system shall generate and display a joke within 3 seconds of a user request under normal operating conditions.
- **NFR2:** The API shall support up to 50 concurrent requests without significant performance degradation.

3.2.2 Usability

- **NFR3:** The application shall have an intuitive interface that requires minimal user training.
- **NFR4:** The design shall ensure accessibility for users with disabilities, following basic WCAG guidelines.

3.2.3 Reliability and Availability

- **NFR5:** The system shall maintain an uptime of 99% during the project period.
- **NFR6:** The system shall handle unexpected errors gracefully, logging issues and displaying appropriate error messages to users.

3.2.4 Security

- **NFR7:** The application shall sanitize all user inputs to prevent injection attacks.
- **NFR8:** API endpoints shall be secured against unauthorized access, implementing standard authentication and authorization measures.
- **NFR9:** All data transmissions must occur over HTTPS to ensure encryption and data integrity.

3.2.5 Maintainability and Extensibility

- **NFR10:** The codebase shall be modular and well-documented, facilitating future maintenance and enhancements.
- **NFR11:** The system architecture shall be designed for scalability to incorporate additional features in subsequent releases.

3.3 System Interfaces

3.3.1 External Interfaces

- **Browser Interface:** The user interacts with the system through a modern web browser.
- **API Interface:** RESTful API endpoints facilitate joke retrieval and integration with third-party services if needed.

3.3.2 Internal Interfaces

- **Model Integration Interface:** The Flask backend communicates with the locally hosted GPT-2 model to fetch generated jokes.
- **Feedback Storage Interface:** A data storage mechanism (file system or database) to securely save user feedback and error logs.

3.4 Data Requirements

- **DR1:** The system shall use the curated dataset of 10,000 Turkish jokes for model training.
- **DR2:** User feedback and error logs shall be stored securely for administrative review.
- **DR3:** The system shall support logging of API calls and system errors for performance monitoring and troubleshooting.

4. Appendices

4.1 Glossary

- **Joke:** A humorous narrative intended for entertainment.
- **GPT-2:** A language generation model by OpenAI.
- **RESTful API:** An API that adheres to the constraints of REST architecture.

- **UI/UX:** The design and usability aspects of the application.

4.2 Document Revision History

- **Version 1.0:** Initial requirements draft prepared on [27.02.2025].
- **Version 1.1:** Revisions after team review and scope updates.[10.04.2025]