maltepe university
istanbul
www.maltepe.edu.tr

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

User Stories & Use Cases Document

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

The purpose of this document is to define and elaborate on the user stories and use cases for the Turkish Joke Generator project. It describes the interactions between different user roles and the system, ensuring that all functional requirements are captured. This document will serve as a reference for developers, testers, and project stakeholders during the design, development, and validation phases.

2. Purpose and Scope

The Turkish Joke Generator is a web application designed to generate culturally relevant Turkish jokes using a fine-tuned GPT-2 model. The system supports various user interactions—from generating jokes on demand to providing feedback—which are critical for continuous improvement and enhanced user experience. This document addresses both functional and non-functional interactions and defines the expected behavior for different scenarios.

3. Actors and User Roles

Primary Actors

- **End User (General User):**
  A user who visits the website to generate and view Turkish jokes. Typically interested in quick, humorous content.
- **Admin/Developer:**
  A team member or administrator responsible for maintaining the system, monitoring feedback, and addressing errors or issues that affect joke quality and system performance.

Secondary Actors

- **External Systems:**
  Systems that interact with our application via the API (e.g., potential third-party applications integrating our joke generation feature).

4. User Stories

4.1 Story 1: Joke Generation

- **As a user,** I want to click a "Get Random Joke" button so that I can instantly receive a funny Turkish joke.
- **Acceptance Criteria:**
  - The button is clearly visible on the homepage.
  - Upon clicking, the system displays a randomly generated joke within 3 seconds.
  - The displayed joke is coherent, culturally relevant, and formatted correctly.
- **Notes:**
  This user story is central to the application’s functionality and drives the overall user engagement.

4.2 Story 2: Joke Category Selection

- **As a user,** I want to select a specific joke category (e.g., Nasreddin Hoca, Temel, or General) so that the generated joke aligns with my preferred style.
- **Acceptance Criteria:**
  - A category selection dropdown or buttons are available.
  - Selecting a category filters the joke generation process.
  - The output joke reflects the chosen category's characteristics.
- **Notes:**
  Category selection enhances personalization and cultural context, ensuring users receive jokes that resonate with them.

4.3 Story 3: Feedback Submission

- **As a user,** I want to provide feedback on a joke so that I can help improve the system.
- **Acceptance Criteria:**
  - A feedback form is accessible on the joke display page.
  - The form collects basic information (e.g., rating, comments).
  - Submitted feedback is stored securely for administrative review.
- **Notes:**
  Collecting feedback is essential for continuous model improvement and quality assurance.

4.4 Story 4: Error Reporting

- **As an admin,** I want to review error reports submitted by users so that I can address issues affecting the joke generation process.
- **Acceptance Criteria:**
  - Error logs and user feedback are aggregated in a secure admin dashboard.

    - Admins can filter and sort error reports by type and severity.
    - Detailed error information is available for troubleshooting.
- **Notes:**
  This story ensures that system performance issues are promptly addressed, maintaining a high-quality user experience.

4.5 Story 5: System Navigation and Usability

- **As a user,** I want an intuitive navigation interface so that I can easily access different features of the web application.
- **Acceptance Criteria:**
    - Clear navigation menus or links are provided.
    - The layout is responsive and user-friendly across devices.
    - All key functionalities (joke generation, category selection, feedback) are easily accessible.
- **Notes:**
  A focus on usability is crucial for ensuring that all users, regardless of technical expertise, can navigate the platform effortlessly.

4.6 Story 6: Model Selection & Comparison

As an end user,
I want to switch between the GPT-2 and LSTM models,
so that I can compare the style and quality of jokes generated by each.
Acceptance Criteria:

- A toggle or dropdown allows selection of “GPT-2” or “LSTM” before generation.
- The selected model is used for subsequent joke requests.
- The UI indicates which model is active and logs the choice for analytics.

4.7 Story 7: Performance Metrics Dashboard

As an administrator,

I want to view performance metrics for both models,

so that I can assess quality, latency, and user satisfaction over time.

Acceptance Criteria:

- A protected dashboard displays metrics such as average response time, error rate, perplexity, and user ratings.
- Charts and tables allow filtering by date range and model type.
- Admins can export reports in CSV or PDF formats.

5. Use Cases

5.1 Use Case 1: Generate Random Joke

- **Primary Actor:** End User
- **Preconditions:**
  - The user is on the home page.
  - The system is online and the GPT-2 model is integrated.
- **Main Flow:**

1. The user clicks the "Get Random Joke" button.
2. The system sends a request to the Flask API.
3. The API invokes the GPT-2 model to generate a joke.
4. The system displays the generated joke on the page.

- **Alternate Flow:**
  - If the model fails to generate a joke within the expected time, display a loading indicator and an error message after a timeout.
- **Postconditions:**
  - The user sees a new, randomly generated Turkish joke.
- **Extensions:**
  - The system logs the joke generation process for performance tracking.

- **Acceptance Criteria:**
  The joke is displayed in under 3 seconds, and the output meets cultural and linguistic expectations.

5.2 Use Case 2: Select Joke Category

- **Primary Actor:** End User
- **Preconditions:**
  - The user is on the joke generation page.
  - Category options are loaded and visible.
- **Main Flow:**

1. The user selects a category from the provided options.
2. The system adjusts the generation parameters to focus on the selected category.
3. The user clicks "Generate Joke" and receives a category-specific joke.

- **Alternate Flow:**
  - If the category selection is invalid or not loaded, the system defaults to a general joke generation mode.
- **Postconditions:**
  - The joke generated aligns with the selected category.
- **Acceptance Criteria:**
  The output clearly reflects the cultural and thematic elements of the chosen category.

5.3 Use Case 3: Submit Feedback

- **Primary Actor:** End User
- **Preconditions:**
  - The user has read the generated joke.
- **Main Flow:**

1. The user clicks on a "Submit Feedback" link or button.
2. A feedback form is presented.

3. The user enters their comments and selects a rating.
4. The user submits the feedback.
5. The system stores the feedback and acknowledges receipt.

- **Alternate Flow:**
  - If the feedback submission fails, the user is prompted to try again or contact support.
- **Postconditions:**
  - The feedback is recorded for further review by the admin.
- **Acceptance Criteria:**
  The feedback is successfully stored and an acknowledgment message is displayed.

5.4 Use Case 4: Review Error Reports (Admin)

- **Primary Actor:** Admin/Developer
- **Preconditions:**
  - The admin is logged into the system.
- **Main Flow:**

1. The admin accesses the error reports dashboard.
2. The system displays a list of error reports and user feedback.
3. The admin filters or sorts reports based on severity or date.
4. The admin selects a report to view detailed information.
5. The admin initiates corrective actions based on the report.

- **Alternate Flow:**
  - If no error reports exist, the dashboard indicates that all systems are functioning normally.
- **Postconditions:**
  - The admin has reviewed the error reports and planned necessary interventions.

- **Acceptance Criteria:**
  The dashboard provides clear, actionable information that assists the admin in maintaining system quality.

5.5 Use Case 5: Switch Between Models

Primary Actor: End User
Preconditions: Both models are loaded and available.
Main Flow:

1. User opens model selector and chooses “LSTM”.
2. User clicks “Generate Joke”.
3. System sends request to /generate/random?model=lstm.
4. LSTM model generates and returns a joke.
5. UI highlights that LSTM output is shown.

Alternate Flow:

- If LSTM server is unreachable, prompt fallback to GPT-2.

Postconditions: Joke from selected model displayed; selector state saved.

5.6 Use Case 6: Compare Model Outputs

Primary Actor: End User
Preconditions: User has generated at least one joke from each model.
Main Flow:

1. User clicks “Compare Models” button.
2. System retrieves last GPT-2 and LSTM jokes from cache.
3. UI displays both jokes side-by-side with generation timestamps.
4. User reviews differences and may provide comparative feedback.

Alternate Flow:

- If one model has no recent output, prompt user to generate from that model first.

Postconditions: Side-by-side comparison shown.

5.7 Use Case 7: View Performance Metrics Dashboard

Primary Actor: Admin

Preconditions: Admin is authenticated; metrics pipeline running.

Main Flow:

1. Admin navigates to Dashboard → Metrics.
2. System displays charts: response times, error rates, perplexity scores, feedback ratings.
3. Admin filters by date range and model (GPT-2 vs. LSTM).
4. Admin exports report as CSV or PDF.

Alternate Flow:

- If data pipeline is delayed, show last available snapshot with timestamp.

Postconditions: Metrics reviewed; insights recorded.

6. Conclusion

This User Stories & Use Cases Document captures the essential interactions between users and the Turkish Joke Generator system. By detailing both the user stories and use cases, the document ensures that functional requirements are clearly defined and can be effectively translated into development tasks. These interactions form the backbone of our agile development cycle, guiding our efforts to build a robust, user-friendly, and culturally relevant web application.

7. References

- **Schwaber, K., & Sutherland, J. (2020).** *The Scrum Guide.* Provides detailed guidelines for agile development and user story formulation.
- **Schwalbe, K. (2015).** *Information Technology Project Management.* Cengage Learning.

Offers foundational concepts for capturing functional requirements and use cases.

- **Flask Official Documentation:** Flask
  Reference for developing RESTful APIs and integrating backend services.
- **Hugging Face Transformers Documentation:** Hugging Face
  Guidelines and best practices for implementing NLP models, which inform our use cases related to text generation and sentiment analysis.
- **OpenAI API Documentation:** OpenAI
  Essential for understanding the integration of GPT models in generating creative content.