Test Case Report: Turkish Joke Generation System

Project Overview

This report outlines the testing approach and test cases for the Turkish Joke Generation System, a project utilizing the Trendyol-LLM-7b-chat-dpo-v1.0 model. The system consists of a Python backend script that interfaces with the model and a planned frontend component.

System Components

1. **Backend**: Python script using Hugging Face Transformers library to interact with Trendyol-LLM-7b-chat-dpo-v1.0
2. **Frontend**: Planned UI interface (to be developed)
3. **Model**: Trendyol/Trendyol-LLM-7b-chat-dpo-v1.0 (7B parameter Turkish language model)

Test Environment

Hardware Requirements

- **Minimum**: CUDA-compatible GPU with 8GB+ VRAM
- **Recommended**: CUDA-compatible GPU with 16GB+ VRAM
- **RAM**: 16GB+
- **Storage**: 15GB free space for model and dependencies

Software Requirements

- Python 3.8+
- PyTorch 2.0+
- Transformers library
- CUDA Toolkit (for GPU acceleration)
- Bit and Bytes library (for quantization)

Test Cases

1. Backend Functionality Tests

TC-B01: Basic Script Execution

Objective: Verify that the Python script executes without errors **Preconditions**: Python environment with all dependencies installed **Test Steps**:

1. Run the script with default parameters: `python3 joke_generator.py`
2. Observe output

Expected Results:

- Script runs without errors
- Default prompt "Fıkra anlat" is processed
- Response is generated and displayed
- Exit code 0

TC-B02: Custom Prompt Input

Objective: Verify that the script accepts custom prompts correctly **Test Steps**:

1. Run the script with a custom prompt: `python3 joke_generator.py "Nasreddin Hoca fıkrası anlat"`
2. Observe output

Expected Results:

- Script processes the custom prompt
- Response contains a joke about Nasreddin Hoca
- Exit code 0

TC-B03: System Prompt Override

Objective: Verify that the system prompt can be overridden **Test Steps**:

1. Run the script with a custom system prompt: `python3 joke_generator.py --system_prompt "Sen komik bir fıkra anlatıcısısın."`
2. Observe output

Expected Results:

- Custom system prompt is used instead of default
- Response style reflects the modified system prompt
- Exit code 0

TC-B04: Generation Parameters

Objective: Verify that generation parameters affect output appropriately **Test Steps**:

1. Run the script with modified temperature: `python3 joke_generator.py --temperature 0.8`
2. Run the same prompt with default temperature (0.3)
3. Compare outputs

Expected Results:

- Higher temperature produces more varied/creative responses

- Lower temperature produces more deterministic responses
- Exit code 0 for both runs

TC-B05: Token Limit Control

Objective: Verify that max_tokens parameter controls response length **Test Steps**:

1. Run the script with reduced token limit: `python3 joke_generator.py --max_tokens 100`
2. Run the script with increased token limit: `python3 joke_generator.py --max_tokens 2048`
3. Compare outputs

Expected Results:

- Shorter max_tokens produces more concise responses
- Longer max_tokens allows for more elaborate responses when appropriate
- Exit code 0 for both runs

TC-B06: Error Handling - Invalid Arguments

Objective: Verify script behavior with invalid arguments **Test Steps**:

1. Run the script with invalid temperature: `python3 joke_generator.py --temperature -0.5`
2. Run the script with invalid token count: `python3 joke_generator.py --max_tokens -100`

Expected Results:

- Script should provide helpful error message
- Script should not crash unexpectedly
- Non-zero exit code

TC-B07: Model Loading Performance

Objective: Evaluate performance of model loading process **Test Steps**:

1. Run the script with timing: `time python3 joke_generator.py "Kısa bir fıkra anlat"`
2. Note the time taken for model loading vs. generation

Expected Results:

- Model loading time is within acceptable limits (<60 seconds with GPU)
- Total execution time is reported
- Exit code 0

2. Model Output Quality Tests

TC-M01: Joke Relevance

Objective: Verify that generated content is relevant to the prompt **Test Steps**:

1. Run the script with specific joke request: `python3 joke_generator.py "Temel fıkrası anlat"`
2. Run the script with specific theme: `python3 joke_generator.py "Okul ile ilgili fıkra anlat"`

Expected Results:

- First output contains a joke about Temel
- Second output contains a joke related to school
- Content is appropriate to the requests

TC-M02: Content Appropriateness

Objective: Verify that generated jokes are family-friendly **Test Steps**:

1. Run the script with multiple prompts requesting jokes
2. Review outputs for inappropriate content

Expected Results:

- No explicit adult content
- No offensive stereotypes
- Humor is generally appropriate for broad audience

TC-M03: Turkish Language Quality

Objective: Verify quality of Turkish language in outputs **Test Steps**:

1. Run the script with prompts requiring proper Turkish grammar
2. Have a native Turkish speaker review outputs

Expected Results:

- Correct Turkish grammar and spelling
- Natural-sounding language
- Proper use of Turkish idioms and expressions

TC-M04: Humor Effectiveness

Objective: Evaluate the humor quality of generated jokes **Test Steps**:

1. Generate 10 different jokes with various prompts
2. Have 3+ Turkish speakers rate the jokes for humor (1-5 scale)

Expected Results:

- Average humor rating of 3+ across all jokes
- At least 30% of jokes rated 4 or higher

TC-M05: Cultural Context

Objective: Verify that jokes reflect Turkish cultural context appropriately **Test Steps**:

1. Run prompts for culturally-specific jokes: `python3 joke_generator.py "Bayram ile ilgili fıkra anlat"`
2. Review outputs for cultural relevance

Expected Results:

- Jokes demonstrate understanding of Turkish cultural references
- Cultural contexts are represented appropriately

3. Performance and Resource Tests

TC-P01: Memory Usage

Objective: Measure memory consumption during execution **Test Steps**:

1. Run the script with memory monitoring: `python3 -m memory_profiler joke_generator.py`
2. Record peak memory usage

Expected Results:

- Peak memory usage is within acceptable limits (<8GB with 4-bit quantization)
- No memory leaks observed

TC-P02: GPU Utilization

Objective: Measure GPU resource usage **Test Steps**:

1. Run the script while monitoring GPU with `nvidia-smi`
2. Record peak GPU memory usage and utilization

Expected Results:

- GPU memory usage is within limits of specified hardware
- GPU utilization patterns show efficient use of resources

TC-P03: Response Time

Objective: Measure time from prompt input to response completion **Test Steps**:

1. Run the script with various prompt lengths and measure time to complete
2. Test with both simple and complex requests

Expected Results:

- Response time is proportional to output length
- Response time for standard joke request is under 5 seconds on recommended hardware

TC-P04: Consecutive Executions

Objective: Verify performance during consecutive runs **Test Steps**:

1. Create a script that runs the joke generator 5 times in succession
2. Monitor system resources during execution

Expected Results:

- No degradation in performance across runs
- Memory is properly released between executions

4. Planned Frontend Integration Tests

TC-F01: API Communication

Objective: Verify backend-frontend communication **Preconditions**: Frontend prototype developed **Test Steps**:

1. Initialize the backend as a service
2. Send requests from frontend to backend
3. Verify response handling

Expected Results:

- Frontend successfully sends requests to backend
- Backend processes requests and returns responses
- Frontend displays responses correctly

TC-F02: User Input Handling

Objective: Verify frontend properly handles user inputs **Test Steps**:

1. Enter various types of prompts in the frontend UI
2. Test with empty inputs, very long inputs, and special characters

Expected Results:

- Frontend validates inputs appropriately
- Error messages are displayed for invalid inputs
- Special characters are handled correctly

TC-F03: Response Rendering

Objective: Verify frontend properly renders model outputs **Test Steps**:

1. Generate responses of varying lengths
2. Verify display on different screen sizes

Expected Results:

- Responses are properly formatted and displayed
- Long responses are handled with appropriate scrolling/pagination
- UI remains responsive with long outputs

TC-F04: Responsiveness

Objective: Verify frontend responsiveness during model processing **Test Steps**:

1. Submit a prompt requiring longer processing time
2. Observe frontend behavior during processing

Expected Results:

- Loading indicator is displayed during processing
- UI remains responsive
- User can cancel ongoing generations

Test Data

Sample Prompts for Testing

- "Fıkra anlat" (Basic request)
- "Nasreddin Hoca fıkrası anlat" (Character-specific)
- "Temel fıkrası anlat" (Character-specific)
- "Kısa bir fıkra anlat" (Length-specific)
- "Öğretmen ve öğrenci hakkında fıkra anlat" (Theme-specific)
- "Bayram ile ilgili komik bir fıkra anlat" (Cultural context)
- "Fıkrayı beş cümlede anlat" (Constraint-specific)

Risk Assessment

Identified Risks

1. **Hardware Limitations**: 7B parameter model requires significant GPU resources
2. **Content Appropriateness**: Risk of generating inappropriate content
3. **Performance Variability**: Response times may vary based on prompt complexity
4. **Turkish Language Quality**: Risk of grammatical errors in generated content

Mitigation Strategies

1. Implement 4-bit quantization (already in code) to reduce memory requirements
2. Include system prompt guidance for appropriate content generation

3. Set reasonable timeout limits for generation
4. Test extensively with native Turkish speakers for language quality assessment

Conclusion and Recommendations

Based on the test cases outlined in this report, we recommend:

1. **Gradual Deployment**: Initially deploy to a limited user group for feedback
2. **Monitoring System**: Implement usage monitoring to identify potential issues
3. **Content Filtering**: Consider implementing additional content filtering if inappropriate content is detected
4. **Performance Optimization**: Profile the application to identify optimization opportunities
5. **User Feedback Mechanism**: Implement a way for users to rate joke quality to improve the system

This test plan should be considered a living document and updated as the frontend development progresses and new requirements emerge.