TED UNIVERSITY

CMPE 492

Senior Project II

PROJECT NAME

Development of a Turkish Language Processing System with Integrated Speech Recognition and Synthesis

Test Plan Report

Submission Date: 17.04.2025

Team Members:

Doğa Paksoy	10078725538
Rana Çetinkaya	31726748334
Berrin Uzun	15982088236

Table of Contents

1 Introduction

1.1 Purpose of the Test Plan

The test plan's main goal is to provide a thorough testing approach for the Turkish Language Processing System, guaranteeing the precise and effective operation of the Speech-to-Text (STT) and Text-to-Speech (TTS) components. In order to contribute to a dependable, scalable, and user-friendly solution for voice-based interactions in Turkish, the strategy seeks to confirm that these modules not only function with excellent performance but also seamlessly connect with FNSS services.

1.2 Scope of Testing

This testing plan encompasses all essential system functionalities, including:

- Text-to-Speech (TTS) conversion accuracy and voice naturalness.
- Speech-to-Text (STT) transcription precision under real-time conditions.
- Processing of audio inputs and outputs in clean and noisy environments.
- Machine learning model performance, especially for Turkish dialects.
- Sound isolation and noise reduction techniques.
- Overall system integration and compatibility with FNSS infrastructure.

1.3 Objectives and Tasks

The testing effort's primary goals and related tasks are:

- To confirm that TTS output is correct, comprehensible, and natural.
- To confirm the accuracy of STT transcriptions for a variety of accents and casual speech.
- To evaluate offline model fine-tuning's effectiveness and impact.
- To measure the resilience of integration, latency, and overall system performance.
- To evaluate audio quality under various acoustic and environmental conditions.

1.4 Assumptions and Constraints

The following assumptions and constraints form the basis of the test design:

- The system's main language, including dialectal variations, is Turkish.
- Users can employ offline fine-tuned models as well as pre-trained models.

- Test audio samples could have interruptions, background noise, or unfavorable recording circumstances.
- It's possible that external services won't always be accessible when testing offline.

2 Test Items

This section outlines the key elements of the system that will undergo testing to verify their proper functionality, compliance with requirements, and effective collaboration.

TTS Model

Evaluates the fundamental text-to-speech model's capability to transform Turkish text into clear and natural audio. This assessment focuses on prosody, pronunciation precision, naturalness, and the model's ability to accommodate various Turkish accents and tones.

STT Module

Examines the system's proficiency in converting spoken Turkish into written text. This includes testing across diverse speaking styles, noise conditions, and dialects, utilizing metrics such as Word Error Rate (WER) for assessment.

Machine Learning Model Components

Confirms the validity of all AI models employed for TTS and STT, encompassing architecture selection, training effectiveness, inference performance, and adaptability to new Turkish data.

FNSS Integration Interface

Ensures seamless integration of the language system with FNSS platforms through APIs. This includes verifying real-time responsiveness, error management, and secure communication protocols.

Sound Isolation Pipeline

Assesses the efficacy of noise reduction algorithms in various environments, emphasizing the preservation of speech quality eliminating background noise during both input (STT) and output (TTS).

Performance Metrics Dashboard

Validates the dashboard that monitors and visualizes system metrics such as latency, accuracy, and user engagement, ensuring precise reporting and clear visual representation.

Reporting & Logging System

Verifies that the system systematically logs user activities, errors, and performance data, ensuring that logs are secure, accessible, and useful for debugging and auditing purposes.

3 Features to be Tested

The Turkish Language Processing System combines various sophisticated modules to ensure efficient speech-to-text (STT) and text-to-speech (TTS) operations, specifically optimized for the Turkish language and designed for compatibility with FNSS services. The following essential features will be evaluated:

Text-to-Speech (TTS) Conversion

- Naturalness and clarity of produced Turkish speech.
- Management of various Turkish dialects and accents.
- Performance in real-time response.
- Error management for invalid or improperly formatted text input.

Speech-to-Text (STT) Conversion

- Precision in Turkish speech transcription.
- Identification of informal phrases and diverse dialects.
- Immediate, low-latency operation.
- Consistency in varying background noise environments.
- Generation of error messages upon recognition failure.

Offline Fine-Tuning

- Effectiveness of model fine-tuning utilizing new dialects or domain-specific datasets.
- Comparison of model performance prior to and following tuning.
- System functionality and logging in offline mode.

FNSS API Integration

- Data transfer between the system and FNSS services.
- Functionality in both high and low network environments.
- Management of API response errors and alternative solutions.

Sound Isolation and Noise Reduction

- Quality of adaptive noise reduction in speech-to-text (STT) and text-to-speech (TTS) processes.
- The system's capability to minimize different forms of background noise.
- Evaluation of performance with noise reduction activated versus deactivated.

Performance Evaluation and Reporting

- Evaluation of latency, clarity, and accuracy for TTS and STT modules.
- Adherence to performance standards (such as response time).
- Generation of reports including key metrics such as response time, clarity, status, and timestamp for performance analysis.
- Options for data export and visualization of testing results.

Error Handling and Notification System

- Prompt and insightful error notifications for users.
- Delivery of notifications (such as visual alerts and logs).

4 Features Not to Be Tested

The following features are explicitly excluded from the current testing scope:

- Support for non-Turkish languages or multilingual capabilities.
- Legacy FNSS systems that are not integrated via defined API connections.
- Latency or performance issues originating from services outside of the internal FNSS network.

5 Testing Methodology

To guarantee the reliability, efficiency, and usability of the Turkish Language Processing System, a thorough testing strategy is implemented. The following testing methods will be utilized throughout the development process:

5.1 Unit Testing

Each system component will be individually tested to verify its correctness and ensure it functions as expected. This step is essential to confirm that all functions operate properly before the integration phase.

5.2 Integration Testing

Following unit testing, modules such as Speech-to-Text (STT), Text-to-Speech (TTS), and the FNSS API interface will be combined and tested to ensure smooth interaction. This phase verifies the data flow between components and identifies any interface discrepancies or communication problems.

5.3 System Testing

A comprehensive end-to-end test of the entire system—from user input to audio output—will be performed to validate overall functionality. This includes assessing input accuracy, processing stability, and output quality under various real-world scenarios.

5.4 Performance Testing

The system will be evaluated against performance metrics such as latency, throughput, memory usage, and CPU/GPU utilization. This testing is essential for ensuring real-time performance and responsiveness.

5.5 User Acceptance Testing (UAT)

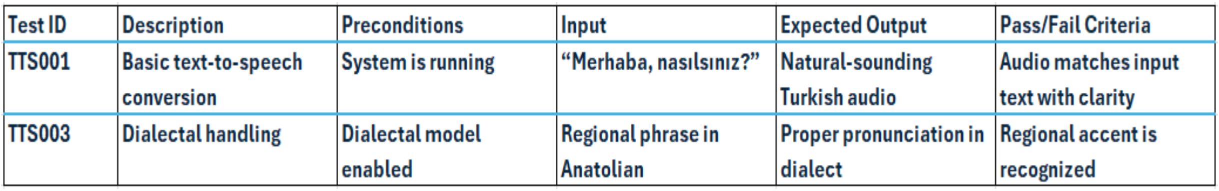
Input will be gathered from native Turkish speakers and FNSS personnel through practical use cases. This phase ensures that the system is user-friendly, culturally relevant, and meets operational requirements.

5.6 Beta Testing

The system will be implemented on a limited scale within a live FNSS environment. Real-time feedback and usage data will be collected to identify any remaining bugs, usability challenges, and areas for improvement before full-scale deployment.

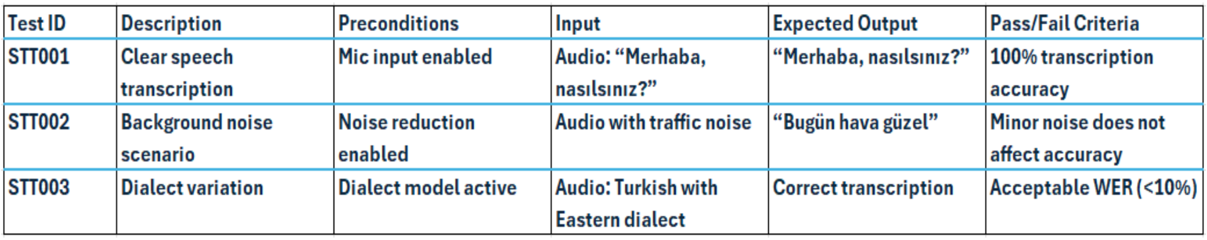
6 Test Cases

Text-to-Speech (TTS) Conversion



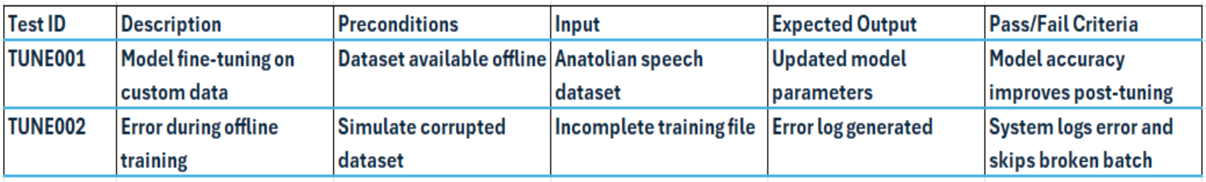
Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
TTS001	Basic text-to-speech conversion	System is running	"Merhaba, nasılsınız?"	Natural-sounding Turkish audio	Audio matches input text with clarity
TTS003	Dialectal handling	Dialectal model enabled	Regional phrase in Anatolian	Proper pronunciation in dialect	Regional accent is recognized

Speech-to-Text (STT) Conversion



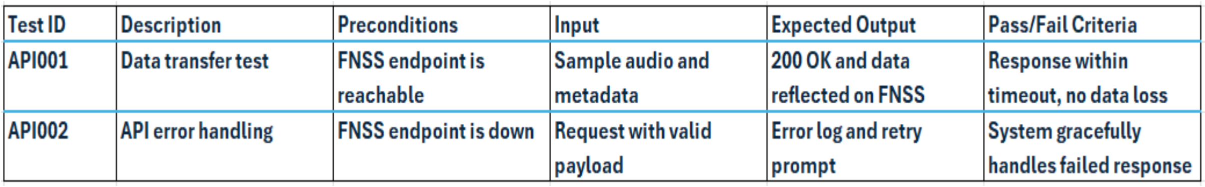
Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
STT001	Clear speech transcription	Mic input enabled	Audio: "Merhaba, nasılsınız?"	"Merhaba, nasılsınız?"	100% transcription accuracy
STT002	Background noise scenario	Noise reduction enabled	Audio with traffic noise	"Bugün hava güzel"	Minor noise does not affect accuracy
STT003	Dialect variation	Dialect model active	Audio: Turkish with Eastern dialect	Correct transcription	Acceptable WER (<10%)

Offline Fine-Tuning



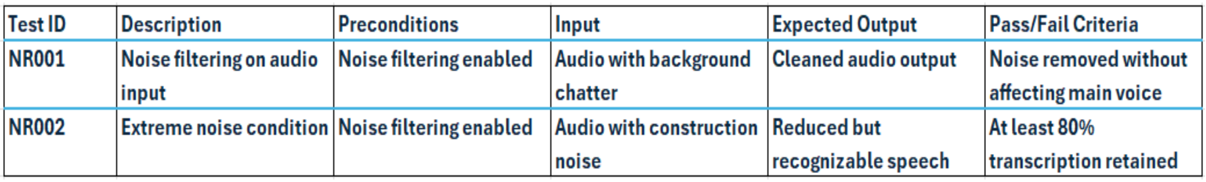
Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
TUNE001	Model fine-tuning on custom data	Dataset available offline	Anatolian speech dataset	Updated model parameters	Model accuracy improves post-tuning
TUNE002	Error during offline training	Simulate corrupted dataset	Incomplete training file	Error log generated	System logs error and skips broken batch

FNSS API Integration



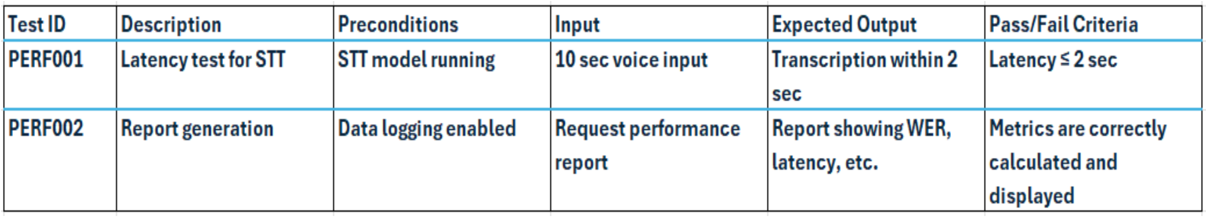
Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
API001	Data transfer test	FNSS endpoint is reachable	Sample audio and metadata	200 OK and data reflected on FNSS	Response within timeout, no data loss
API002	API error handling	FNSS endpoint is down	Request with valid payload	Error log and retry prompt	System gracefully handles failed response

Sound Isolation and Noise Reduction



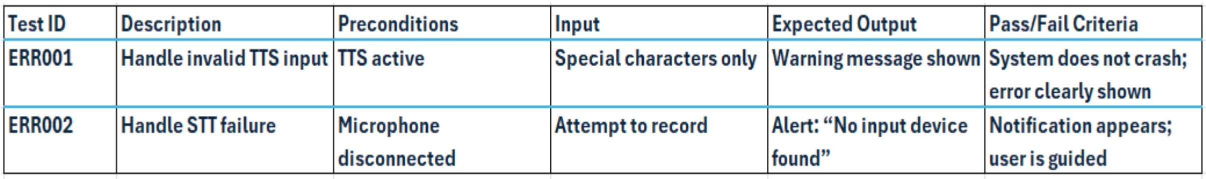
Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
NR001	Noise filtering on audio input	Noise filtering enabled	Audio with background chatter	Cleaned audio output	Noise removed without affecting main voice
NR002	Extreme noise condition	Noise filtering enabled	Audio with construction noise	Reduced but recognizable speech	At least 80% transcription retained

Performance Evaluation and Reporting



Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
PERF001	Latency test for STT	STT model running	10 sec voice input	Transcription within 2 sec	Latency ≤ 2 sec
PERF002	Report generation	Data logging enabled	Request performance report	Report showing WER, latency, etc.	Metrics are correctly calculated and displayed

Error Handling and Notifications



Test ID	Description	Preconditions	Input	Expected Output	Pass/Fail Criteria
ERR001	Handle invalid TTS input	TTS active	Special characters only	Warning message shown	System does not crash; error clearly shown
ERR002	Handle STT failure	Microphone disconnected	Attempt to record	Alert: "No input device found"	Notification appears; user is guided

7 Test Environment

The following will be present in the controlled environment where the testing will take place:

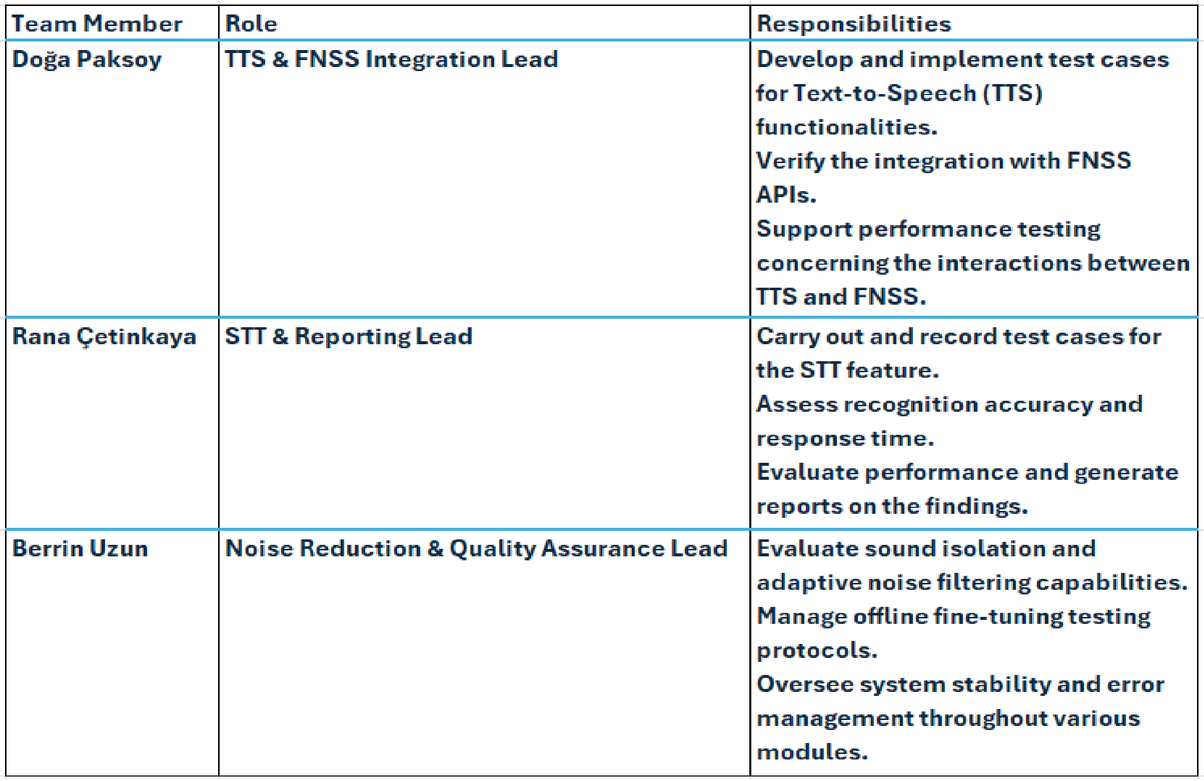
- **Hardware:** High-end computer systems with multi-core CPUs and NVIDIA A100 or T4 GPUs.
- **Software Stack:** TensorFlow, PyTorch, and a Python environment with Docker containerized deployment.
- **Evaluation Tools:**
  - ASR resources for measuring WER (Word Error Rate)
  - MOS measurements for assessing the perceived quality of audio
  - Capabilities for integration with FNSS testing infrastructure and servers
- **Datasets:** Various Turkish audio datasets that mimic real-world variability by including both clean and noisy samples.

8 Test Schedule

Testing Method	Start Date	End Date
5.2 Integration Testing	02.03.2025	15.03.2025
5.3 System Testing	16.03.2025	29.03.2025
5.4 Performance Testing	30.03.2025	12.04.2025
5.5 User Acceptance Testing (UAT)	13.04.2025	26.04.2025
5.6 Beta Testing	27.04.2025	10.05.2025

9 Roles and Responsibilities

The project team members will jointly oversee the testing process, with each individual assigned specific testing tasks that align with their expertise and knowledge of the system components. The following outlines the distribution of responsibilities:



Team Member	Role	Responsibilities
Doğa Paksoy	TTS & FNSS Integration Lead	Develop and implement test cases for Text-to-Speech (TTS) functionalities. Verify the integration with FNSS APIs. Support performance testing concerning the interactions between TTS and FNSS.
Rana Çetinkaya	STT & Reporting Lead	Carry out and record test cases for the STT feature. Assess recognition accuracy and response time. Evaluate performance and generate reports on the findings.
Berrin Uzun	Noise Reduction & Quality Assurance Lead	Evaluate sound isolation and adaptive noise filtering capabilities. Manage offline fine-tuning testing protocols. Oversee system stability and error management throughout various modules.

10 Control Procedures

Testing and quality assurance will be supported by the following control mechanisms:

- **CI/CD Automation:** GitLab CI/CD pipelines will be used to automate testing and deployment processes
- **Issue Tracking:** All defects and test anomalies will be tracked using JIRA
- **Bug Triage Meetings:** Weekly meetings will be held to prioritize and resolve reported issues
- **Regression Testing:** Comprehensive regression tests will be conducted after each major system update to ensure consistent performance

11 Risks and Mitigations

Risk	Mitigation Strategy
Inaccurate recognition of Turkish dialects or accents	Broaden the training datasets to encompass a variety of dialects; perform focused fine-tuning utilizing regional voice samples..
Background noise affecting STT accuracy	Incorporate sophisticated noise isolation and filtering methods; evaluate their performance in diverse acoustic settings.
User privacy concerns with voice data	Implement comprehensive end-to-end encryption, provide options for local or offline processing, and ensure clear user consent through transparent policies.
Integration failures with FNSS services	Create strong, well-documented APIs and conduct comprehensive integration and compatibility testing with the FNSS services.

Limited internet access affecting system usability	Ensure offline capabilities by utilizing lightweight models that can be deployed locally and do not depend on constant internet access.
Data loss or corruption	Establish routine data backups by employing secure storage options that offer redundancy and recovery features.
Low-quality audio input	Incorporate sound isolation techniques, utilize noise filters, and promote the use of suggested hardware such as noise-canceling microphones.
FNSS API changes	Adopt version-locking practices throughout the development process and ensure backward compatibility in API management.
Turkish dialectal variation	Integrate region-specific voice data into training datasets and consistently refresh the corpus to include dialects that are underrepresented.

12 References

[1]: Sommerville, I. (2015). *Software Engineering* (10th ed.). Pearson Education.