TED UNIVERSITY

CMPE 491

Senior Project I

PROJECT NAME

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

Project Analysis Report

Submission Date: 22.11.2024

Team Members:

Doğa Paksoy 10078725538

Rana Çetinkaya 31726748334

Berrin Uzun 15982088236

Table of Contents

1 Introduction

The "Development of a Turkish Language Processing System with Integrated Speech Recognition and Synthesis" project aims to create a comprehensive language processing system for Turkish, offering both text-to-speech (TTS) and speech-to-text (STT) functionalities. Utilizing machine learning techniques, this system will transform written text into natural-sounding speech and convert spoken language into accurate text. Designed for seamless integration with FNSS services, it aims to enhance operational efficiency and improve the user experience. Through language-specific tuning, the project will optimize the system's performance, while sound isolation techniques will ensure clear, noise-free results, providing a comprehensive solution for voice-based interactions.

2 Current System (if any)

3 Proposed System

The proposed system is an AI-driven solution for Turkish language processing, equipped with TTS and STT functionality. The spoken language gets converted into text by this machine learning/NLP-based approach, while natural speech in Turkish pronunciation will be generated from text input. The system is trained on a wide range of datasets that include different Turkish dialects and accents, ensuring great accuracy. The tool will be developed in a Python-based environment using TensorFlow and Keras. It incorporates sound isolation technology for better speech recognition. It would also provide a user-friendly interface to access functions of TTS/STT, with real-time error alerts. Integration with FNSS services via APIs enables the smooth interaction while offline functionality supports users in low-connectivity areas.

3.1 Overview

This project aimed to create a speech recognition and speech synthesis system for the Turkish language by providing a sound speech processing system that could understand and handle Turkish text and audio , providing seamless voice interactions. The system leverages recent developments in machine learning and deep neural networks. The model is trained on a wide-variety dataset that captures different accents and dialects of Turkish; thus, it produces accurate and culturally appropriate results.

The solution is built using Python-based libraries, including TensorFlow, Keras, and PyTorch, which help to better develop and train models efficiently while Sound Isolation Technology ensures speech recognition quality by reducing background noise. Additionally, the system is designed to manage real-time inputs for effective online and offline operation with simple integration into external systems for FNSS services, facilitating seamless data exchange and improved functionality for diverse user needs.

3.2 Functional Requirements

1. Text-to-Speech (TTS) Conversion: The system will transform Turkish text into speech that sounds natural, accommodating different dialects and allowing adjustments to voice settings such as pitch, speed, and volume.

2. Speech-to-Text (STT) Conversion: The system should reliably convert spoken Turkish into written text, accommodating dialects and informal expressions with real-time, low-latency processing.

3. Offline Fine-Tuning: The system must enable offline fine-tuning to enhance model performance for Turkish, permitting updates without interfering with functionality.

4. FNSS Services Integration: The system will support API-based data exchange, facilitate smooth integration with FNSS services, and guarantee compatibility with FNSS infrastructure.

5. Sound Isolation and Noise Reduction: The system will utilize adaptive noise reduction to enhance speech clarity across different acoustic settings.

6. Performance and Quality Benchmarking: The system will incorporate tools for evaluating the accuracy, latency, and clarity of TTS and STT, facilitating both manual and automated assessments.

7. Reporting: The system will produce reports on the performance of TTS and STT, including options for export and the ability to filter by metrics and time periods.

8. Admin Dashboard: The system will offer a dashboard for tracking usage, setting up TTS/STT configurations, and retrieving live performance information.

9. Error Handling and Notifications: The system must identify errors in TTS/STT functions, delivering clear error messages and alerts for significant problems.

3.3 Nonfunctional Requirements

1. Performance: The system must perform TTS and STT conversions instantly, with very low latency to provide a smooth user experience.

2. Accuracy: The system must effectively process Turkish dialects and informal phrases, using an extensive dataset to ensure high-quality text-to-speech (TTS) and speech-to-text (STT) outputs.

3. Reliability: The system must operate reliably with different audio and text inputs, guaranteeing consistent and trustworthy TTS and STT functions.

4. Usability: The system must prioritize ease of use, incorporating an intuitive interface that includes straightforward instructions, notifications, and alerts regarding system performance and any errors.

5. Maintainability: The system will feature a modular architecture to facilitate straightforward updates, troubleshooting, and future improvements. Both the code and documentation must be clear and well-structured.

6. Security: The system must adhere to data privacy laws by utilizing encryption and role-based access controls to safeguard sensitive information.

7. Scalability: The system must enable horizontal scaling to accommodate growing demand, with an architecture crafted for seamless integration of model updates.

8. Compatibility: The system will connect with FNSS services and be compatible with common platforms (Windows, macOS, iOS, Android) as well as standard file formats (WAV, MP3, TXT).

9. Compliance: The system must comply with data privacy laws (such as GDPR and KVKK) as well as industry standards, fulfilling all necessary accessibility and security requirements.

10. Localization: The system must provide customization tailored for Turkish dialects, idioms, and phrases, with the user interface and documentation accessible in both Turkish and English.

3.4 Pseudo Requirements

1. Hardware Requirements
   - ➔ Requirements: The system needs to function on standard FNSS servers equipped with sufficient GPU and memory to handle real-time TTS and STT processing.
   - ➔ Details: The FNSS infrastructure must offer adequate hardware resources to guarantee optimal performance without any latency problems.
2. Network Dependence
   - ➔ Requirements: The system needs a reliable internet connection to enable TTS/STT features that depend on cloud processing or external APIs.
   - ➔ Details: Consistent internet connectivity is essential for all outside processing, particularly during high-traffic periods.
3. Data Availability
   - ➔ Requirements: Access to high-quality datasets in the Turkish language is essential for effectively training and fine-tuning the models.
   - ➔ Details: Datasets should incorporate a wide variety of dialects and expressions to improve system precision and cultural significance.
4. User Proficiency
   - ➔ Requirements: Individuals engaging with the system ought to possess a fundamental level of digital skills.
   - ➔ Details: Only a small amount of training will be necessary since the FNSS dashboard and its configuration settings are crafted to be user-friendly.

5. Operational Environment
   - ➔ Requirements: The system must be implemented in a safe, regulated setting to avoid any unauthorized access.
   - ➔ Details: FNSS needs to guarantee limited access to the system in order to safeguard confidential information and avoid any disruptions.
6. Compliance with Future Standards
   - ➔ Requirements: The system operates under the assumption that existing regulatory frameworks, such as GDPR and KVKK, will continue to be stable.
   - ➔ Details: Any significant changes in regulations might necessitate adjustments to maintain ongoing compliance.
7. Third-Party Integrations
   - ➔ Requirements: The functionality relies on the compatibility and continuous support from the external libraries or APIs utilized in TTS/STT processing.
   - ➔ Details: Modifications in external resources could impact system performance and would necessitate prompt updates.
8. Data Retention Policy
   - ➔ Requirements: FNSS is anticipated to implement data retention and deletion policies to adhere to privacy regulations.
   - ➔ Details: Well-defined policies will aid in legal compliance and guarantee that essential data is preserved for operational needs.
9. Testing and Validation Scope
   - ➔ Requirements: Benchmark testing will take place in settings with normal background noise, avoiding extreme conditions.
   - ➔ Details: This guarantees that the system operates effectively in typical settings, although there may be some constraints in environments with significant noise.
10. Language and Dialect Limitations
    - ➔ Requirements: The system expects gradual changes in Turkish dialects and aims to limit the need for frequent model updates.
    - ➔ Details: This premise enables consistent functioning while adapting to moderate changes in language.

3.5 System Models

The project will include distinct system models to comprehensively illustrate its structure and functionality:

- Use case Model
- Object and class Model
- Sequence Diagram
- Activity Diagram

3.5.1 Scenarios

Scenario 1: Text-to-Speech (TTS) Conversion

Description: Adapts pitch, speed, and volume settings to translate Turkish text into natural speech.

Flow: The audio is played after the system processes the Turkish text using TTSModel. If the conversion fails, an error message appears.

Scenario 2: Speech-to-Text (STT) Conversion

Description: Adapts to dialects and informal expressions when translating spoken Turkish into accurate text.

Flow: The system records audio using STTModel to transcribe it, and then shows the text in real-time. There are instructions for troubleshooting in case transcription doesn't work.

Scenario 3: Offline Fine-Tuning

Description: Through offline fine-tuning, TTS and STT models are optimized for the subtleties of the Turkish language.

Flow: The system modifies model parameters while performing planned maintenance by accessing a dataset in Turkish. The admin receives notification, and any tuning errors are recorded.

Scenario 4: Integration with FNSS Services

Description: Enables smooth data exchange with FNSS services via APIs.

Flow: After establishing an API session with FNSS and confirming data integrity, the system sends a success message. The admin receives an alert if integration fails.

Scenario 5: Sound Isolation and Noise Reduction

Description: Adaptive noise reduction in TTS and STT processes improves speech clarity.

Flow: Noise filters are used in real-time audio processing to isolate background noise and enhance clarity. A quality improvement notice is sent if noise reduction is unsuccessful.

Scenario 6: Performance and Quality Benchmarking

Description: Uses automated tools to evaluate the clarity, latency, and accuracy of TTS and STT.

Flow: Reports containing the results of benchmark tests are generated from the Admin Dashboard. An error message is recorded if testing is failed.

Scenario 7: Reporting

Description: Generates comprehensive reports with filtering options on TTS and STT performance.

Flow: The system gathers and presents performance metrics after the user requests a report. An admin notification and error message are displayed if report generation is failed.

Scenario 8: Admin Dashboard Management

Description: Offers resources for monitoring performance, modifying configurations, and tracking TTS/STT usage.

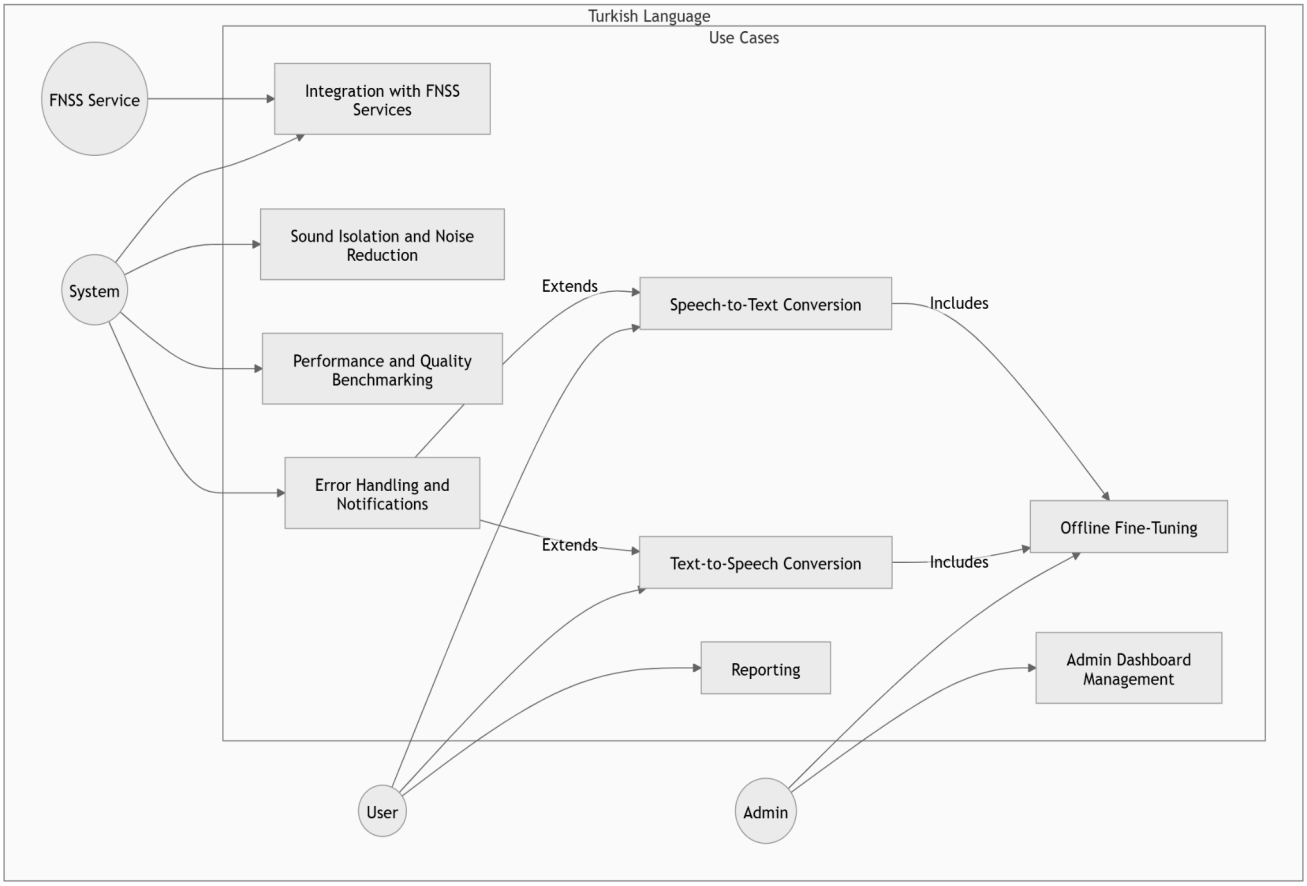
Flow: The admin views the dashboard, adjusts preferences, and focuses on notifications. An error message is sent if the configuration fails.

Scenario 9: Error Handling and Notifications

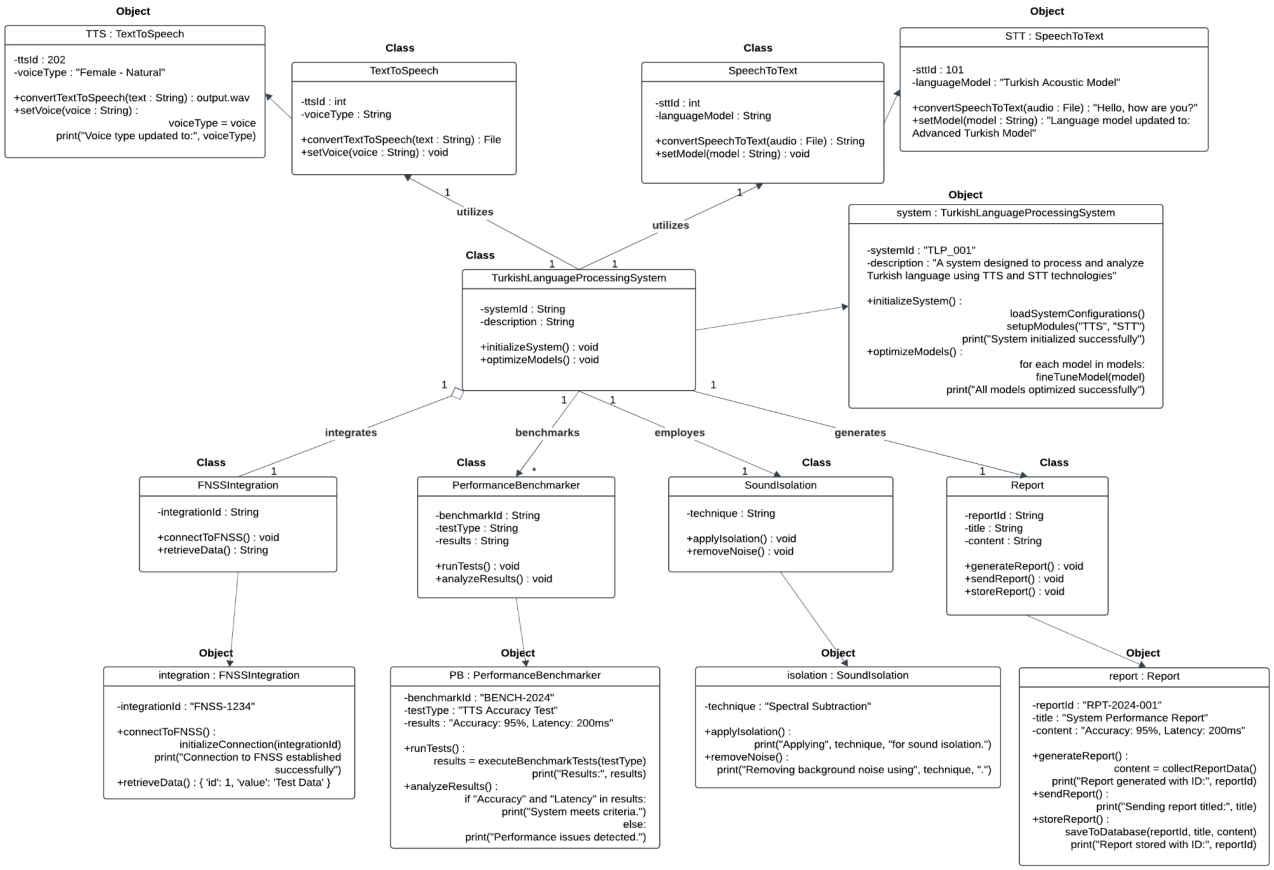
Description: Identifies and fixes TTS/STT function errors, alerting the administrator to serious problems.

Flow: Critical alerts are sent to the admin, and errors are shown on the interface. Notifications are escalated according to system rules if problems continue.

3.5.2 Use Case Model

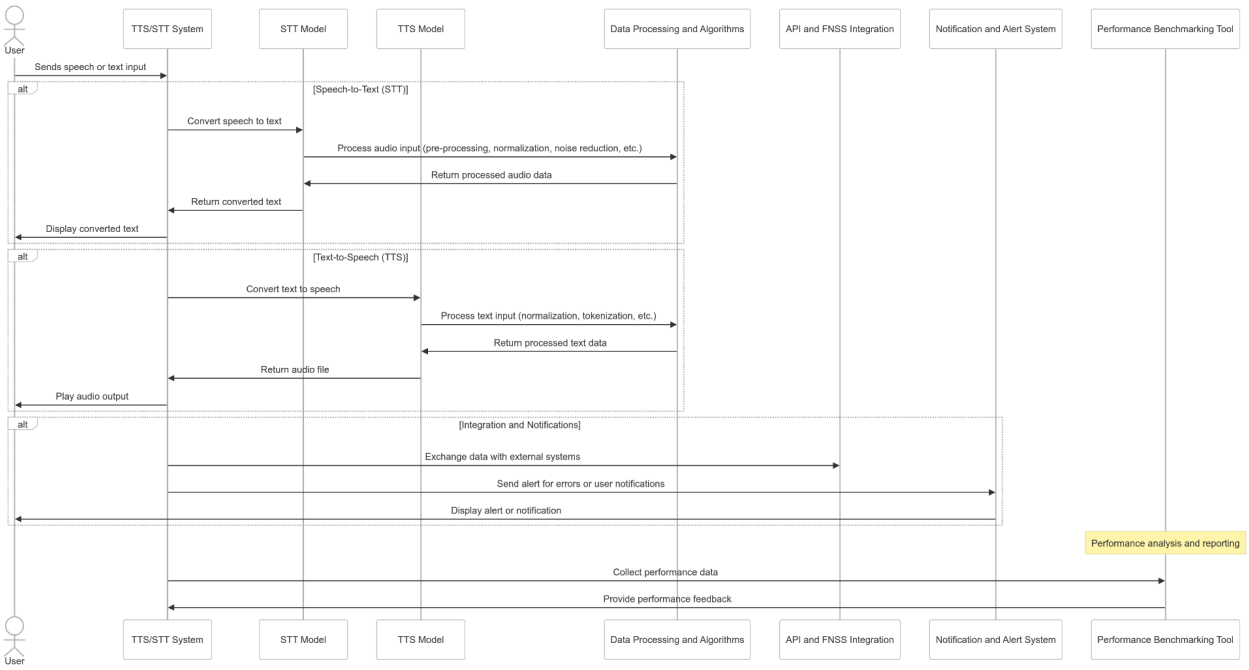


3.5.3 Object and Class Model

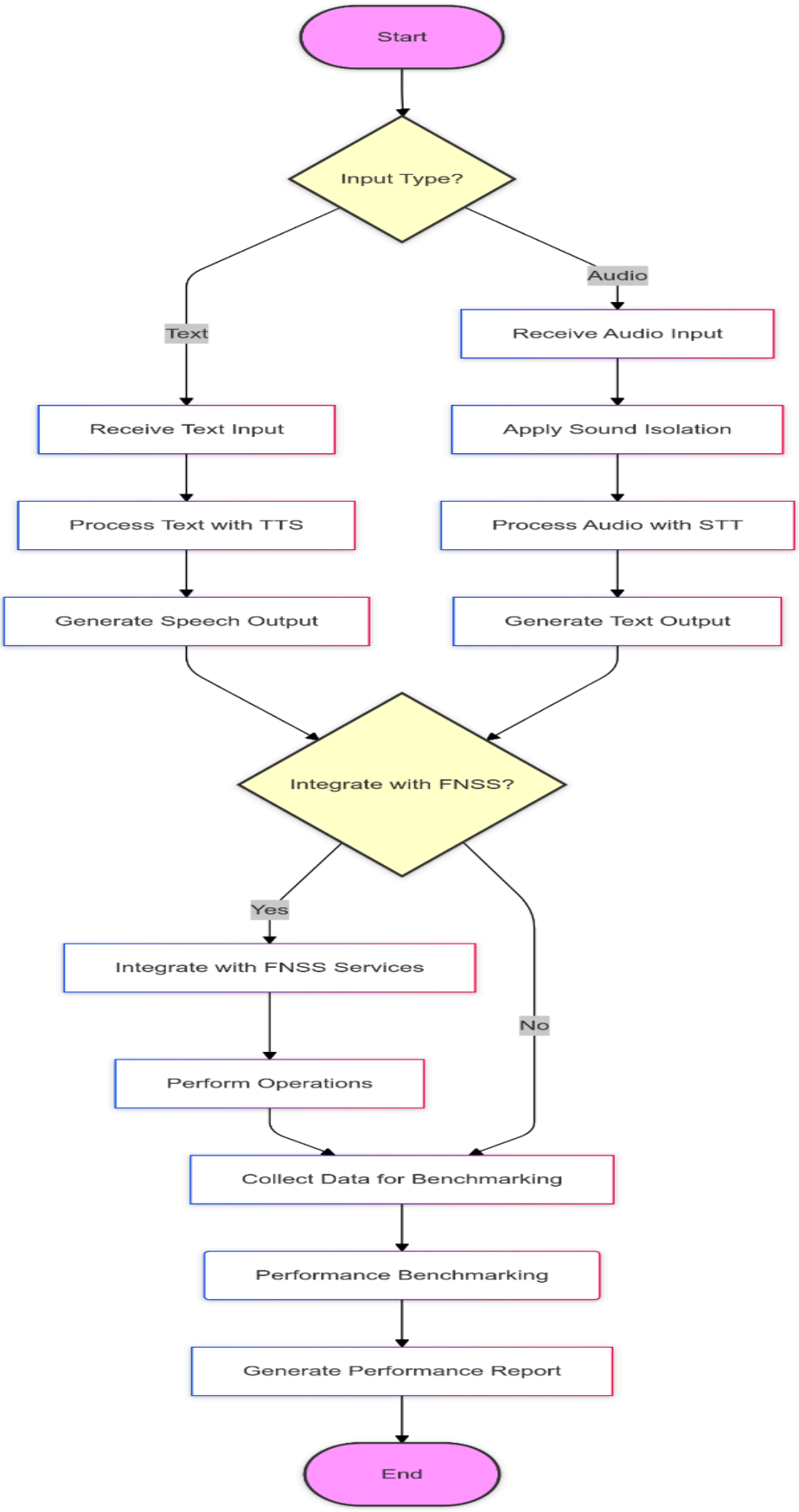


3.5.4 Dynamic Models

a) Sequence Diagram:



b) Activity Diagram:



3.5.5 User interface - navigational paths and screen mock-ups

User Interface:

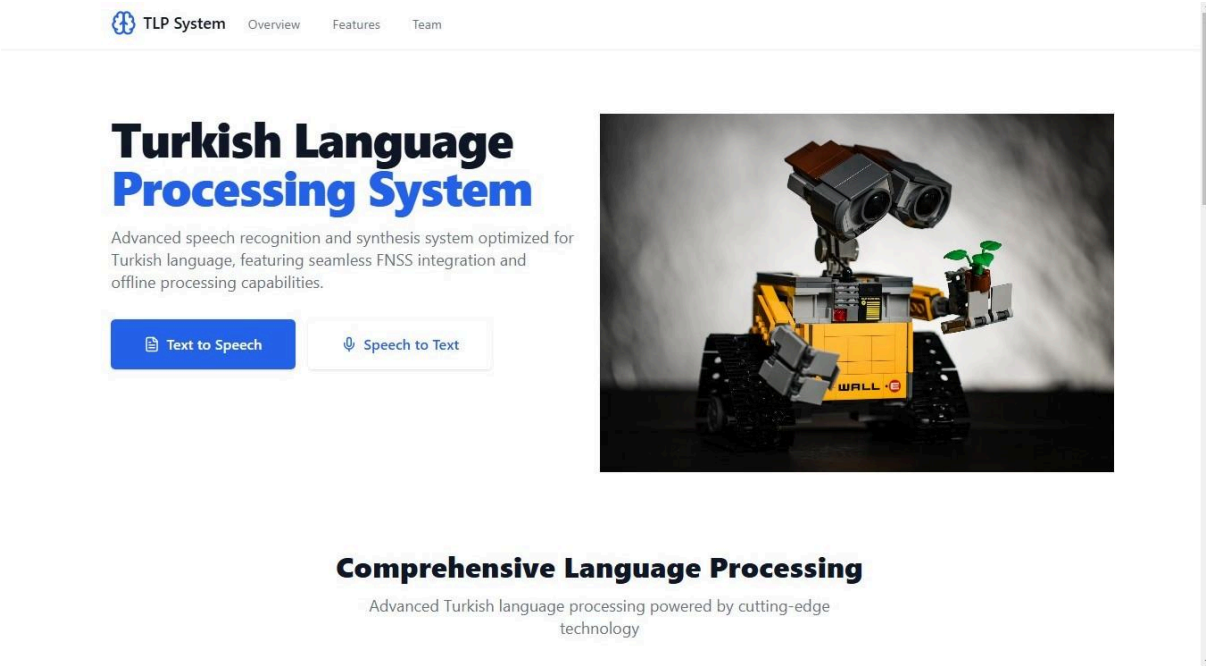


Figure 1: Main Page. Image is taken from [1]

Navigational paths and screen mock-ups:

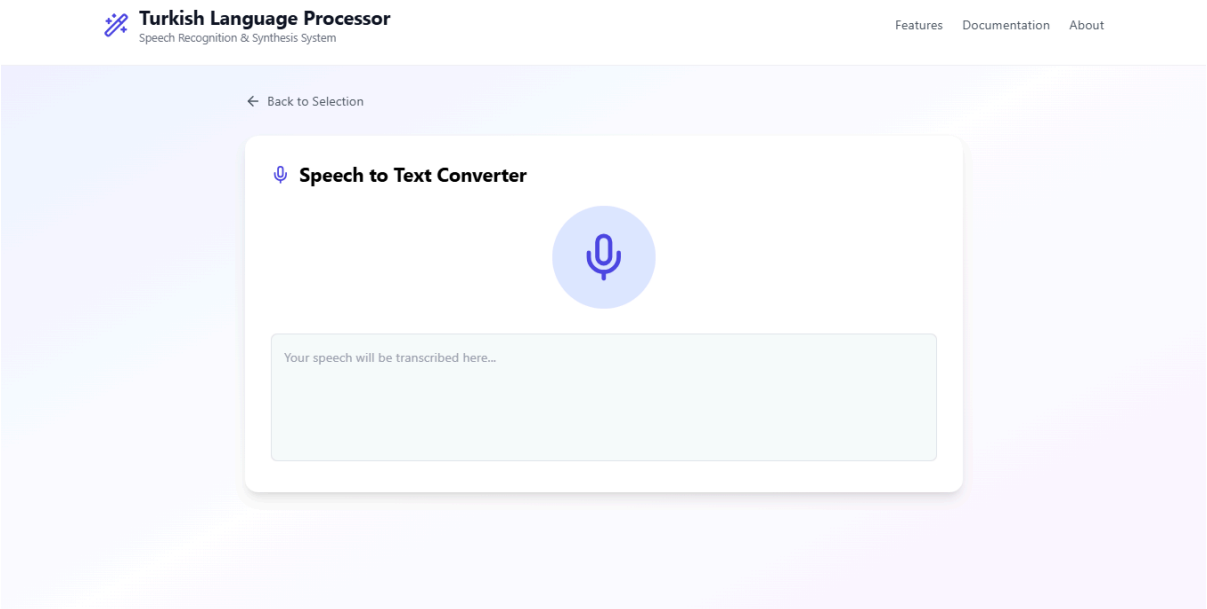


Figure 2: STT Converter Page. Image is taken from [2]

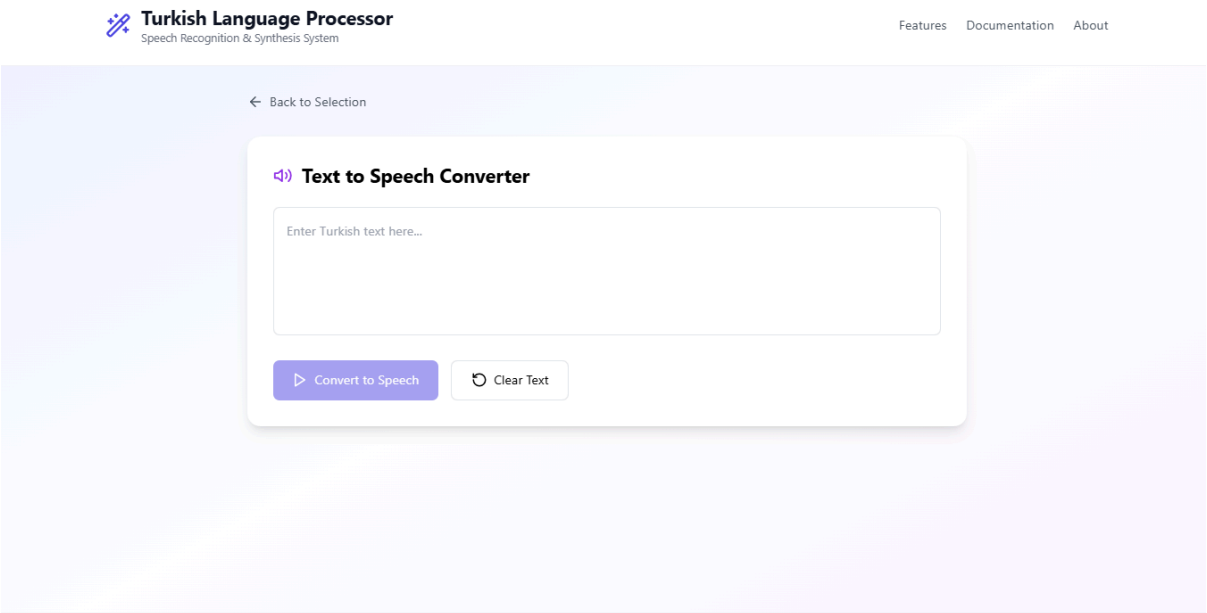


Figure 3: TTS Converter Page. Image is taken from [3]

4 Glossary

Activity Diagrams: Activity diagrams illustrate the actions involved in a process or data manipulation. (Sommerville, I., (2015), Software Engineering, Pearson Education)

Android: A mobile operating system created by Google that is open-source, designed for use on devices like smartphones and tablets.

API: A collection of protocols and tools that enables various software applications to connect and work together.

Cloud Processing: Utilizing internet-hosted remote servers to handle data processing and application execution, minimizing the reliance on local computing resources.

Deep Neural Networks: A kind of machine learning model that employs several layers of nodes to analyze vast quantities of data and produce predictions or decisions.

Fine-Tuning: The procedure of fine-tuning a pre-trained machine learning model to enhance its effectiveness for a particular task or dataset.

GDPR and KVKK: The General Data Protection Regulation (GDPR) in the European Union and the Personal Data Protection Law (KVKK) in Turkey are both privacy laws that regulate the management and safeguarding of personal data.

GPU: A Graphics Processing Unit is a hardware element designed to enhance computational speed, particularly for machine learning operations that involve extensive datasets and models.

iOS: The mobile operating system created by Apple Inc. for its devices, including the iPhone and iPad.

MacOS: The operating system created by Apple Inc. for its desktop and laptop systems.

Machine Learning: A branch of artificial intelligence that enables systems to learn from data and enhance their performance over time without the need for direct programming.

MP3: A well-known audio file format that employs lossy compression for saving music and other audio information.

Object and Class Model: A depiction of items and their connections within a system, utilizing classes to outline properties and functions in object-oriented programming.

Sequence Diagram: Sequence diagrams illustrate the interactions occurring between users and the system, as well as the interactions among the components of the system. (Sommerville, I., (2015), Software Engineering, Pearson Education)

Speech-to-Text (STT): A technology that transforms oral communication into written form.

TensorFlow: An open-source machine learning framework developed by Google for building and training machine learning models, particularly deep learning models.

Text-to-Speech (TTS): A system that transforms written words into auditory speech.

WAV: A widely used audio file format that retains high-quality audio data without compression.

Windows: A personal computer operating system created by Microsoft.

Use Case Model: Use case diagrams illustrate the relationships between a system and its surrounding environment. (Sommerville, I., (2015), Software Engineering, Pearson Education)

5 References

[1]: Main Page. Retrieved from: https://bolt.new

[2]: STT Converter Page. Retrieved from: https://bolt.new

[3]: TTS Converter Page. Retrieved from: https://bolt.new

Software Engineering, Ian Sommerwille, 10th Ed.