**Final Report: Java Voice Assistant**

**Project Overview**

The Java Voice Assistant is an innovative application that enables hands-free interaction with a computer using voice commands. Leveraging the CMU Sphinx library for speech recognition and JavaFX for an intuitive GUI, the assistant executes system-level tasks and enhances accessibility. This project demonstrates the integration of speech-to-text technology and Java's capabilities to create a user-friendly, practical software solution.

**Objectives**

- Convert human speech into actionable commands.

- Provide a user-friendly interface for seamless interaction.

- Perform tasks like opening/closing applications, volume control, and system operations.

- Enhance accessibility and hands-free control for users.

**Features**

1. Speech Recognition: Converts spoken language to text using CMU Sphinx.

2. Application Control: Opens and closes applications like Chrome, Notepad, and YouTube.

3. Volume Control: Increases or decreases system volume using external tools like NirCmd.

4. System Operations: Handles tasks such as shutdown, restart, sleep, and changing wallpapers.

5. Graphical User Interface (GUI): Displays recognized commands and provides feedback.

**Methodology**

**1. Audio Signal Processing**

- Capture voice input via a microphone.

- Process the input with CMU Sphinx to convert audio into text.

- Match recognized text to predefined commands.

**2. Speech Recognition Setup**

- Configured CMU Sphinx with:

- Acoustic Model (en-us)

- Custom Dictionary (.dic)

- Language Model (.lm)

- Initialized `LiveSpeechRecognizer` for real-time speech processing.

**3. Command Execution**

- Used `switch-case` logic to map recognized commands to actions.

- Leveraged `ProcessBuilder` to execute system-level tasks like:

- Opening/closing applications.

- Adjusting volume.

- Performing shutdown, restart, or sleep operations.

**4. GUI Development**

- Built with JavaFX for an intuitive user experience.

- Included a button for starting the voice assistant and a text area for displaying feedback.

**5. Testing and Debugging**

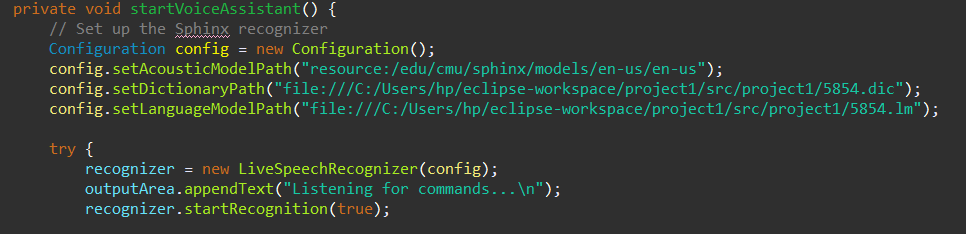
- Tested commands in various conditions.

- Debugged recognition issues and execution errors.

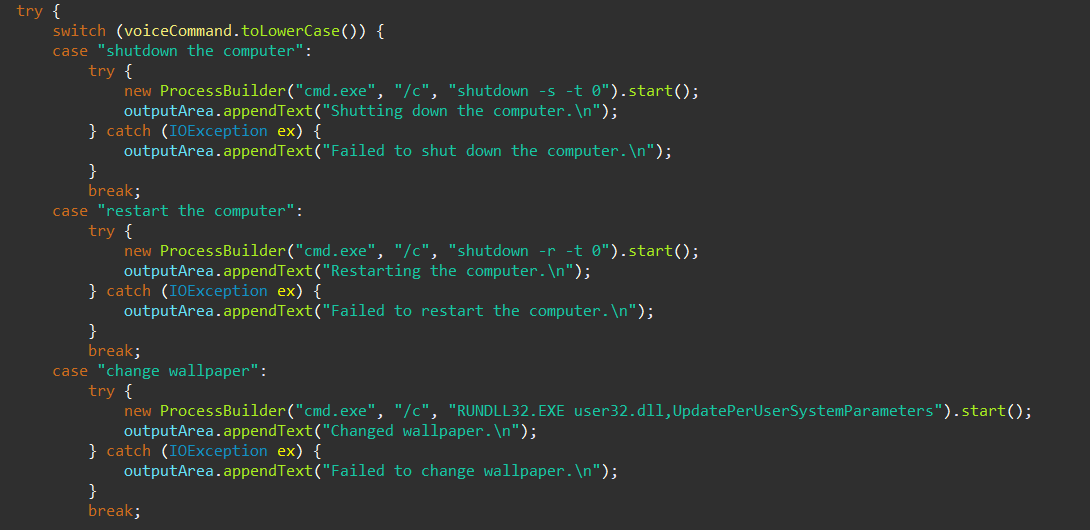
- Refined the dictionary and language model for accuracy.

**Code Highlights**

Speech Recognition Configuration



Command Execution



**Results**

The voice assistant successfully:

- Recognizes and executes user commands in real-time.

- Provides a responsive GUI for interaction and feedback.

- Performs tasks like application control, volume adjustment, and system operations seamlessly.

**Challenges & Solutions**

**1. Noise Interference:**

- Challenge: Speech recognition accuracy decreased in noisy environments.

- Solution: Optimized the microphone input and applied preprocessing.

**2. Command Recognition:**

- Challenge: Matching user speech to predefined commands.

- Solution: Improved the custom dictionary and language model.

**3. GUI Responsiveness:**

- Challenge: Ensuring the GUI remains responsive during speech processing.

- Solution: Used multithreading to separate GUI and speech recognition processes.

**Conclusion**

This project demonstrates the integration of advanced speech recognition with Java’s robust capabilities to create an accessible and practical voice assistant. Future improvements include adding more commands, improving recognition accuracy, and expanding functionality to support additional applications and devices.

**Tools Used:**

- JDK, CMU Sphinx, JavaFX

- NirCmd for volume control

**Resources:**

- Pre-trained acoustic models

- Custom dictionary and language models