**Feature: Voice and Command Recognition for AI Elephant with Feedback Reinforcement**

**Description**

A web interface where users can control an AI-powered elephant using voice commands in Malayalam, Hindi, or Gujarati. The interface will:

* Display a header with "AI Elephant" and a sub-header "Command Module."
* Allow users to enable a microphone for voice input.
* Process voice commands through an LLM "Supervisor" agent, which routes them to a "Language Understanding" sub-agent.
* Display a table of predefined commands with radio buttons for manual selection.
* Repeat selected commands at a configured interval.
* Collect user feedback (correct/incorrect recognition) to improve the model via reinforced learning.

**User Stories & Tasks**

**User Story 1: As a user, I want to see a clear web interface with branding and command module identification.**

**Tasks:**

1. Create a header with a small image (logo) and text "AI Elephant."
2. Add a sub-header below with the text "Command Module."
3. Ensure responsive design for different screen sizes.

**User Story 2: As a user, I want to enable a microphone to give voice commands.**

**Tasks:**

1. Implement a microphone toggle button (enable/disable).
2. Integrate the Web Speech API for voice recognition.
3. Display a visual indicator (e.g., animation) when the microphone is active.
4. Handle permission requests for microphone access.

**User Story 3: As a user, I want my voice command to be processed by an LLM Supervisor agent.**

**Tasks:**

1. Set up an API endpoint to receive voice input.
2. Send the voice input to the LLM "Supervisor" agent for initial classification.
3. Ensure the Supervisor identifies the input as a command and forwards it to the "Language Understanding" sub-agent.

**User Story 4: As a user, I want the system to recognize and execute commands in Malayalam, Hindi, and Gujarati.**

**Tasks:**

1. Create a structured dataset of commands (as per the table).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Action | Malayalam Commands | Hindi Commands (हिंदी) | Gujarati Commands (ગુજરાતી) |
| 1 | Turn Left | ഇടത്താനെ (Idathāne) | **बाएं** (Bāẽ) | **ડાબે** (Ḍābe) |
| 2 | Turn Right | വലത്താനെ (Valathāne) | **दाएं** (Dāẽ) | **જમણે** (Jamaṇe) |
| 3 | Walk Forward | നടയാനെ (Naṭayāne) | **चल** (Chal) | **ચાલ** (Chāl) |
| 4 | Walk Backward | സെറ്റാനെ (Seṭṭāne) | **पीछे** (Pīche) | **પાછળ** (Pāchaḷ) |
| 5 | Stop | നില്ലാനെ (Nillāne) | **ठहर** (Thahar) | **થોભ** (Thobh) |
| 6 | Lie down | കിടന്നാനെ (Kiṭannāne) | **लेट** (Leṭ) | **પડ** (Paḍ) |
| 7 | Sit | ഇരിയാനെ (Iriyāne) | **बैठ** (Baiṭh) | **બેસ** (Bes) |
| 8 | Lock the foot firmly on the ground | ഊന്നാനെ (Ūnnāne) | **जमीन दबा** (Jamīn dabā) | **જમીન દબાવ** (Jamīn dabāv) |
| 9 | Lift the trunk | ഭീരിയാനെ (Bhīriyāne) | **सूंड उठा** (Sūṇḍ uṭhā) | **સુંડ ઊંચી કર** (Sūṇḍ ū̃chī kar) |
| 10 | Bend down and take the leaves | എടാനെ (Eṭāne) | **झुक कर ले** (Jhuk kar le) | **ઝુકીને લે** (Jhūkīne le) |
| 11 | Lift leaves with the trunk | താങ്ങാനെ (Tāṅṅāne) | **सूंड से उठा** (Sūṇḍ se uṭhā) | **સુંડથી ઊંચક** (Sūṇḍthī ū̃chak) |
| 12 | Give blessing with the trunk | ആശീർവദിക്കാനെ (Āśīrvadikkāne) | **आशीर्वाद दो** (Āśīrvād do) | **આશીર્વાદ આપ** (Āśīrvād āp) |
| 13 | Move ears | ചെവിയാട്ടാനെ (Cheviyāṭṭāne) | **कान हिला** (Kān hilā) | **કાન હલાવ** (Kān halāv) |
| 14 | Move head | തലയാട്ടാനെ (Talayāṭṭāne) | **सिर हिला** (Sir hilā) | **ડોક હલાવ** (Ḍok halāv) |
| 15 | Lift one front leg | നട പൊക്കാനെ (Naṭa pokkāne) | **आगे पैर उठा** (Āge pair uṭhā) | **આગળનો પગ ઊંચો કર** (Āgaḷno pag ū̄cho kar) |
| 16 | Lift one back leg | അമരം പൊക്കാനെ (Amaram pokkāne) | **पीछे पैर उठा** (Pīche pair uṭhā) | **પાછળનો પગ ઊંચો કર** (Pāchaḷno pag ū̄cho kar) |
| 17 | Close eyes | കണ്ണ് അടയ്ക്കാനെ (Kaṇṇ aṭaykkāne) | **आंख बंद** (Āṅkh band) | **આંખો બંધ કર** (Āṅkho bandh kar) |
| 18 | Spray water from raised trunk | ഭീരി ഒഴിയാനെ (Bhīri oḻiyāne) | **पानी छिड़क** (Pānī chiṛak) | **પાણી છાંટ** (Pāṇī chhāṇṭ) |
| 19 | Stretch both sets of legs | നീട്ടി വെയ്യാനെ (Nīṭṭi veyyāne) | **पैर फैला** (Pair phailā) | **પગ લંબાવ** (Pag lambāv) |
| 20 | Make a sound | ഒന്നു വിളിച്ചെയാനെ (Onnu viḷiccheyāne) | **आवाज कर** (Āvāj kar) | **અવાજ કર** (Avāj kar) |
| 21 | Lift back leg for man to climb | മടക്കാനെ (Maṭakkāne) | **चढ़ने के लिए पैर उठा** (Chaṛhne ke lie pair uṭhā) | **ચડવા માટે પગ ઊંચો કર** (Chaḍvā māṭe pag ū̄cho kar) |
| 22 | Stand straight | നേരെ നില്ലാനെ (Nēre nillāne) | **सीधे खड़े हो** (Sīdhe khaṛe ho) | **સીધા ઊભા રહે** (Sīdhā ū̄bhā rahe) |
| 23 | Eat what is in your mouth | തിന്നോ ആനെ (Thinnō āne) | **खा लो** (Khā lo) | **ખા લે** (Khā le) |

1. Train/fine-tune the "Language Understanding" sub-agent to recognize the commands in all three languages.
2. Map recognized commands to corresponding actions.

**User Story 5: As a user, I want to see a table of predefined commands with radio buttons for selection.**

**Tasks:**

1. Display the command table with columns: No, Action, Trigger, Interval.
2. Add toggle button for each command row in ‘Trigger’. (Single action)
3. Ensure the table is scrollable if too long.

**User Story 6: As a user, I want the selected command to be repeated at a configured interval.**

**Tasks:**

1. Add an input field for setting the repeat interval (in seconds).
2. Implement a loop to send the selected command at the specified interval.
3. Stop Repeating once the toggle button is turned off.

**User Story 7: As a user, I want feedback when a command is executed.**

**Tasks:**

1. Display a confirmation message when a command is recognized.
2. Show the detected command text on the screen.
3. Provide visual feedback (e.g., highlight the selected command in the table).
4. There should be feedback button (👍/👎) to provide real time feedback on the command detection.

**User Story 8: As a user, I want error handling for unrecognized commands.**

**Tasks:**

1. Implement a fallback response for unrecognized commands.
2. Display a message like "Command not recognized, please try again."
3. Log unrecognized inputs for future improvements.

**User Story 9: As a user, I want to provide feedback on whether the system correctly interpreted my command.**

**Tasks:**

1. Accept user feedback through the feedback button (👍/👎) after each executed command.
2. Store feedback along with the original voice input and detected command.
3. Send feedback data to a backend service for reinforced learning.
4. Display a "Thank you for your feedback!" message.

**User Story 10: As an admin, I want to analyze user feedback to improve command recognition accuracy.**

**Tasks:**

1. Set up a database to store:
   * Voice recordings
   * Detected commands
   * User feedback (correct/incorrect)
   * Timestamp
2. Create an analytics dashboard to track recognition accuracy per language/command.
3. Implement a retraining pipeline to fine-tune the model based on feedback.

**User Story 11: As a system, I want to improve recognition based on user feedback.**

**Tasks:**

1. Use stored feedback to retrain the "Language Understanding" sub-agent periodically.
2. Apply reinforcement learning techniques to adjust command mappings.
3. Notify admins when accuracy improvements are detected.

**Additional Technical Tasks**

1. Set up backend API for LLM agent communication.
2. Implement WebSocket or polling for real-time command updates.
3. Add language detection logic to route commands correctly.
4. Write unit tests for command recognition and execution.
5. Optimize voice processing for low-latency response.
6. POST endpoint to log: {voice\_sample: "audio.wav", detected\_command: "Walk Forward", user\_feedback: true/false}
7. **Database Schema**

CREATE TABLE command\_feedback (

id INT AUTO\_INCREMENT,

voice\_sample BLOB,

detected\_command VARCHAR(100),

user\_feedback BOOLEAN,

language VARCHAR(20),

timestamp DATETIME

);

1. Reinforcement Learning Pipeline

* Weekly retraining job using new feedback data.
* Accuracy reports emailed to admins.

**Acceptance Criteria**

✅ Mic button enables/disables voice input with visual feedback.

✅ Recognized commands are displayed and executed.

✅ Command table allows selection via radio buttons.

✅ Selected commands repeat at the configured interval.

✅ Unrecognized commands trigger an error message.

✅ UI is responsive and user-friendly.

✅ Feedback buttons appear after command execution.

✅ Feedback is stored and used for model improvement.

✅ Admins can access accuracy reports.