**Nubot Robot Project - Task Breakdown and Expected Outcomes**

**Project Summary:** Nubot is an interactive child-friendly robot powered by a Raspberry Pi. It engages children through speech recognition, text-to-speech response, emotion detection, and simple games. The system is designed to run offline as much as possible.

### **1. Hardware Setup**

**Tasks:** - Install Raspberry Pi OS - Connect and test microphone, speaker, and LEDs - Setup GPIO for input/output control

**Expected Outcome:** - Raspberry Pi is operational with input and output peripherals working

**Resources:** - Raspberry Pi 4/5 - Microphone (USB or analog) - Speaker (USB or 3.5mm audio) - LEDs + resistors - Breadboard and jumper wires - MicroSD card with Raspberry Pi OS

### **2. Offline Speech Recognition Integration**

**Tasks:** - Install Vosk speech recognition library - Configure models for English and Arabic - Develop Python module for capturing and converting speech to text

**Expected Outcome:** - Real-time offline voice-to-text conversion in both English and Arabic

**Resources:** - Vosk (https://alphacephei.com/vosk/) - Python - Pre-trained language models (EN, AR)

### **3. Offline Text-to-Speech Integration**

**Tasks:** - Install and configure Coqui TTS or eSpeak/Festival - Generate child-friendly audio responses

**Expected Outcome:** - Voice responses from the system using text-to-speech offline

**Resources:** - Coqui TTS (https://github.com/coqui-ai/TTS) - eSpeak NG / Festival - Prewritten response scripts

### **4. Emotion Detection System**

**Tasks:** - Integrate pitch/tone-based emotion detection or simple classifier - Connect LED color output to emotional states

**Expected Outcome:** - System detects basic emotional tone and reflects it with light or message

**Resources:** - pyAudioAnalysis or custom ML model - Python libraries: librosa, numpy, scipy

### **5. Game: Guess the Animal Sound**

**Tasks:** - Store audio clips of animal sounds - Play clip and prompt user to guess - Match answer using voice recognition

**Expected Outcome:** - Interactive voice-based animal sound guessing game

**Resources:** - Audio files (e.g., .wav/.mp3) - Python audio module (e.g., pydub or playsound) - Voice input logic

### **6. Game: Tic Tac Toe**

**Tasks:** - Create 3x3 game grid logic - Enable move selection by voice - Add win/loss detection and voice feedback

**Expected Outcome:** - Fully playable voice-controlled Tic Tac Toe game

**Resources:** - Python or web interface (HTML/JS) - JSON state tracking - Offline STT/TTS modules

### **7. Game: Story Spinner**

**Tasks:** - Randomly select and tell short stories via TTS - Allow replay or next interaction by voice

**Expected Outcome:** - System can tell stories interactively

**Resources:** - Local text/story files - TTS module - JSON logic file

### **8. Multilingual Interaction (Arabic/English)**

**Tasks:** - Detect language intent or configure user language - Route STT and TTS through selected language models

**Expected Outcome:** - Full support for bilingual interaction

**Resources:** - Vosk language models - Coqui TTS bilingual setup

### **9. Conversational Logic & Navigation System**

**Tasks:** - Develop JSON-based decision flow - Prompt user every 10s and await “yes” or question - Route interaction accordingly

**Expected Outcome:** - Structured conversational system with logic-based navigation

**Resources:** - Python - JSON logic files - Voice prompt recordings

### **10. Notification & Alert System**

**Tasks:** - Detect trigger conditions (emotion/stress) - Display notification, light alert, or store event locally

**Expected Outcome:** - Basic alert system integrated into emotion flow

**Resources:** - Local JSON/event logger - LEDs, buzzer, or display

### **11. System Testing & Validation**

**Tasks:** - Test each module independently - Conduct full interaction flow test - Validate voice inputs, games, and emotion feedback

**Expected Outcome:** - Fully working, integrated and bug-free prototype

**Resources:** - Test scripts - Sample dialogues - Volunteers (children) for feedback

### **12. Packaging and Documentation**

**Tasks:** - Organize code and files - Create setup instructions and usage manual - Prepare demonstration video or guide

**Expected Outcome:** - Documented, reusable, and demonstrable system package

**Resources:** - Markdown or Word document - Video recording software - GitHub or USB archive

**End of Report**