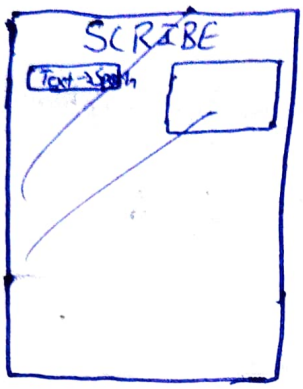

Smart Pen → Optical or magnetic tracking
- → pressure sensors
- → Wireless communication: include Bluetooth Low Energy (BLE) or another low power wireless tech.
- → otherwise wire
- → Buttons → for clearing etc.
- → ~~Visual feedback: LED~~
- → Vibrate when error

UI

Writing pad
- → memory
- → connectivity modules
- → Audio output
- → Customisation
- → Visual feedback: LEDs

1) Using Raspberry Pi 5
- Raspberry pi 5 board
- 7-8 inch touchscreen display
- Micro SD card with ~~8~~ OS installed.
- Power supply (USB-C)
- Bluetooth & Wifi modules
- Optical / magnetic sensors (for pen tracking)
- Pressure sensors (for pen & pad)
- Vibration motor (For haptic feedback)
- Microphone & speaker (for audio feedback)
- Jumper wires, resistors
- Rechargable battery

(2) Designing the smart pen

- Optical sensor for pen tracking (like in optical mouse)
  - Use optical sensor module (ADNS-2620) to track movement of pen on writing surface
  - Connect sensor to R.P GPIOs pins
  - Python code using the "spidev" library to read data from optical sensor & determine pos^n & movement of pen.

- Pressure sensor
  - Integrate a pressure sensor (FSR or load cell) into the pen to detect how hard the child is pressing down the pen while writing
  - Use an ADC to interface the analog sensor with RP's digital GPIOs pins [MCP3008]
  - Python code to read pressure sensor values

- Haptic feedback
  - Add a small vibration motor to the pen to provide haptic feedback (for error detection or task completion).
  - We can control the vibration motor using a GPIO pin & a transistor ckt (or a motor driver module like L293D).
  - Python code to control the motor by sending signals, such as incorrect writing or anything else.

(3) Building the smart writing pad

- **Touchscreen Display**
  - → Use 7-8 inch RP touchscreen display to act as a writing pad (stylus optional).
  - → Install touchscreen libraries ["evdev", "libinput"] to detect handwriting gestures.
  - → Handwriting recognition using Tensorflow / Tesseract OCR/opencv.

- **Audio feedback system**
  - → Connect a small speaker to the RP (3.5 mm audio jack)
  - → Install python libraries ("pygame" or "pydub") to provide audio feedback for writing tasks (reading out text, alerting errors).

- **Visual feedback**
  - → Use display to provide visual cues (highlighting errors, displaying errors, showing incorrect words / letters).
  - → Use python libraries GUI (Tkinter, PyQt, Kivy) for designing a child-friendly UI.

(4) SD (software development)

- Handwriting recognition
  - AI/ML → to train a model
  - opencv to recognise handwriting inputs

- Feedback & error detection
  - Implement error detection techniques to compare child's input with expected answers (ex: use string comparison techniques to check spelling & grammar).
  - Provide corrective feedback, such as vibrations or audio prompts, when the user makes mistake.

- Voice-to-text and Text-to-speech
  - Use Google Speech recognition API or python libraries like "SpeechRecognition" for voice-to-text functionality.
  - Use "pyttsx3" to implement text-to-speech for reading out written content.