B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

OOMD Mini Project Report

Developing writing pen and writing pad for children with learning disability

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

BJ Keertana (1BM23CS05)
Dharunya Balavelavan (1BM23CS090)
Disha H Jain (1BM23CS095)
Disha DS (1BM23CS094)

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2025-26

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **BJ Keertana(**1BM23CS059), **Dharunya Balavelavan (1BM23CS090), Disha H Jain(1BM23CS095), Disha DS (**1BM23CS094)
students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled "**Developing writing pen and writing pad for children with learning disability**" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester August 2025- December 2025. I also declare that to the best of our knowledge and belief, the OOMD mini Project report is not from part of any other report by any other students.

Signature of the Candidate

BJ Keertana(1BM23CS059)

Dharunya Balavelavan (1BM23CS090)

Disha H Jain(1BM23CS095)

Disha DS (1BM23CS094)

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOMD Mini Project titled “**Developing writing pen and writing pad for children with learning disability”** has been carried out by BJ Keertana(1BM23CS059), Dharunya Balavelavan(1BM23CS090), Disha H Jain (1BM23CS095), Disha DS (1BM23CS094) during the academic year 2025-2026.

Signature of the Faculty in Charge
Vikranth B M
Assistant Professor

Table of Contents

Sl No	Title	Pageno
1	Ch 1: Problem statement	6
2	Ch 2: Software Requirement Specification	6
3	Ch 3: Class Diagram	10
4	Ch 4: State Diagram	12
5	Ch 5: Interaction diagram	18
6	Ch 6: UI Design with Screenshots	25

Chapter 1: Problem Statement

Children with learning disabilities—such as dysgraphia, ADHD, and mild motor-coordination challenges—often struggle with handwriting, letter formation, and maintaining writing consistency. Traditional pen-and-paper methods do not provide real-time guidance, feedback, or correction, making it difficult for children to practice independently. Teachers and parents also lack tools to monitor progress accurately or customize exercises for each child's learning needs. To address these challenges, an intelligent Writing Pen and Writing Pad System is required. The system should consist of a smart digital writing pen that captures writing movements (pressure, tilt, speed, stroke direction) and a digital writing pad that displays the child's writing in real-time. The system must provide guidance, corrective feedback, progress tracking, and personalized practice exercises to support children with learning disabilities. The pen and pad should work together using embedded sensors, Bluetooth/USB connectivity, and a companion software application. The software should analyze handwriting patterns, highlight mistakes (such as improper letter angle, spacing, or size), and provide visual/audio cues to help children correct their writing. Teachers and parents should be able to create tasks, view reports, and track improvements over time. The system should be easy to use, child-friendly, durable, and accessible. It must support multiple difficulty levels, customizable learning modes, and multilingual character sets (English alphabets, numbers, regional languages if needed).

Chapter 2: Software Requirement Specification

Software Requirements Specification (SRS) for Developing Writing Pen and Writing Pad for Children with Learning Disability

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the detailed functional and non-functional requirements for the development of a writing assistance system comprising a smart writing pen and writing pad designed specifically for children with learning disabilities. The goal of this system is to aid children who face difficulties in handwriting, such as those with dysgraphia or fine motor coordination issues, by offering real-time feedback, correctional guidance, and progress tracking. This document serves as a foundation for system design, development, testing, and validation, ensuring that all stakeholders—developers, educators, and parents—have a clear understanding of the system's objectives and capabilities.

1.2 Scope

The proposed system, “Developing Writing Pen and Writing Pad for Children with Learning Disability,” consists of two main hardware components—a smart pen and a smart writing pad—integrated with a companion software application. The pen will be equipped with motion and pressure sensors to detect handwriting movements, while the pad will digitally capture the written strokes. Together, these devices will transmit data to the companion application, which will analyze handwriting patterns and provide corrective feedback. The application will display progress data, generate reports for teachers and parents, and allow personalized exercises for children. The system will primarily benefit children with learning disabilities by improving their handwriting skills through an engaging and interactive experience, while teachers and parents can use the platform to monitor growth and adapt teaching methods accordingly.

1.3 Overview

This document is organized into several sections.
Section 1 introduces the purpose, scope, and structure of the project.
Section 2 describes the overall product, including its perspective, functions, and user characteristics.
Section 3 outlines specific requirements, both functional and non-functional, while
Section 4 explains system features and interactions. The document concludes with appendices that describe possible future enhancements and a glossary of key terms.

2. General Description

The writing pen and pad system is an integrated assistive product combining both hardware and software components. The smart pen captures writing dynamics such as stroke direction, pressure, and motion using embedded sensors, while the smart pad records handwriting and transmits the data to the companion software via Bluetooth or USB connection. The companion application, installed on a mobile device or computer, receives the data, processes it using pattern recognition algorithms, and generates instant visual and auditory feedback. The system architecture follows a flow in which the pen transmits raw motion data to the pad, the pad digitizes it, and the application analyzes it to provide real-time guidance and record performance statistics.

3. Functional Requirements

The system shall capture handwriting data from the smart pen and pad in real time. It shall analyze the pressure, direction, and speed of each stroke to evaluate the handwriting pattern. It shall identify and correct letter formation errors through visual and auditory feedback, guiding the user toward proper writing techniques. The application shall allow teachers or therapists to assign personalized writing exercises to each child and store their progress in a local or cloud database. Parents and teachers shall be able to view performance analytics, export reports in standard formats such as PDF or CSV, and operate the system in offline mode when necessary. The application shall also support multiple user profiles so that different children can use the same system independently.

4. Performance Requirements

The performance of the system is critical since real-time responsiveness directly affects the learning experience of children. The system shall provide handwriting feedback within a maximum latency of 500 milliseconds from the time the child writes a stroke to the moment feedback is displayed. This responsiveness ensures smooth, interactive guidance that prevents user frustration. The handwriting capture rate should be at least 100 samples per second, allowing accurate tracking of fast or irregular handwriting patterns.

The system shall be capable of processing handwriting data continuously for sessions lasting up to one hour without overheating or noticeable performance degradation. It must support at least ten active user profiles on a single device without affecting speed or memory performance. The application shall be able to store and retrieve large volumes of data efficiently, maintaining response times under two seconds for any user action such as loading exercises, viewing reports, or switching profiles.

5. Interface Requirements

The system will interact with hardware components through Bluetooth Low Energy or USB connections. The smart pen will send motion and pressure data to the pad, which will then communicate with the companion software. The sensors embedded in the pen, such as accelerometers and gyroscopes, will provide accurate motion tracking. The software interface will include both mobile and desktop applications that synchronize data with a cloud service. The user interface will consist of three modes: a child view offering playful visual cues, a teacher view with analytical dashboards, and a parent view summarizing the child's progress. The software interface will include the companion application that runs on Android, iOS, and Windows platforms. It will communicate with the hardware through BLE protocols and with the cloud server via secure REST APIs for synchronization and storage. The software shall support firmware updates for both the pen and pad, allowing future upgrades without requiring hardware replacement. The application will also provide an export interface to generate performance reports in PDF or CSV format that teachers and parents can download.

6. Design Constraints

The design of the smart writing pen and writing pad system is subject to several hardware, software, environmental, regulatory, and usability constraints that must be considered throughout development. From a hardware perspective, the system is limited by the physical size, weight, and power capacity of the writing pen and pad. Since the product is intended for children aged five to twelve, the devices must be lightweight, ergonomic, and safe to use. The pen must not exceed a weight of 40 grams, and the pad must remain thin and portable for classroom use. From a software design standpoint, the constraints include limited processing power and memory in the embedded system of the pen and pad. The firmware must be efficient enough to perform data acquisition, filtering, and transmission in real time with minimal computational overhead. The companion application must function smoothly across different platforms such as Android, iOS, and

Windows, requiring cross-platform development frameworks and careful optimization to ensure consistent performance.

7. Non-Functional Attributes

The first key attribute is performance. The system must respond to user input in real time, with handwriting feedback delivered within a maximum latency of 500 milliseconds. Data transfer between the pen, pad, and companion application must be smooth and continuous, ensuring that no stroke or gesture is missed. The handwriting capture rate should be high enough (at least 100 samples per second) to provide precise motion tracking, and the software must handle multiple user profiles and long sessions without lag or overheating.The second attribute is reliability. Since the system will be used by children and teachers in classrooms, it must operate consistently under varying conditions.

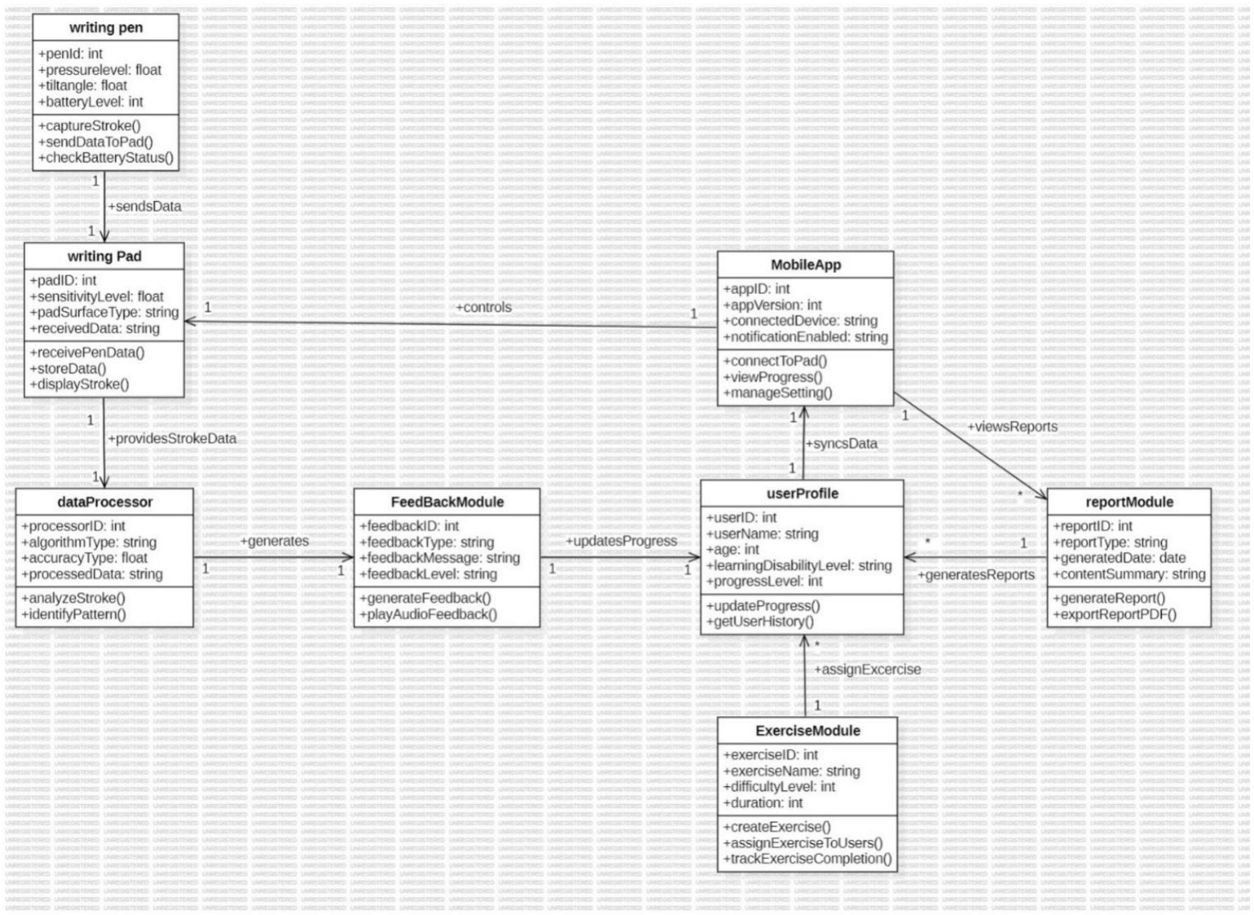
Hardware components such as the sensors and communication modules must function correctly even under minor physical disturbances. The software should have a recovery mechanism that automatically restores unsaved data after an unexpected shutdown or connectivity loss. The system should achieve an uptime of at least 95 percent and maintain accuracy across long-term usage.

Usability is another critical non-functional attribute, as the target users are children with learning difficulties. The system interface must be intuitive, engaging, and accessible. The design must use clear icons, simple instructions, and vibrant visuals while avoiding excessive text or complex menus. Voice prompts and audio cues should be incorporated to assist users who struggle with reading or attention. The application should also allow teachers and parents to navigate dashboards easily, view progress reports, and customize learning exercises without requiring technical expertise.

8. Preliminary Schedule and Budget

The development of the smart writing pen and writing pad system for children with learning disabilities is planned to be completed over a period of approximately nine months, divided into five key phases: requirement analysis, system design, development, testing, and deployment. The first month will focus on collecting user requirements and feasibility analysis. The next two months will be dedicated to hardware and software design, including sensor selection, circuit design, and UI mockups. The following three months will cover prototype development and integration of both hardware and software components. Two months will be allocated for functional and usability testing, including pilot trials with children, teachers, and therapists. The final month will focus on documentation, training, and deployment. The estimated total budget for the project is approximately ₹18–20 lakhs (USD 21,000–24,000). Out of this, around ₹6 lakhs is allocated for hardware development, ₹5 lakhs for software and machine learning modules, ₹3 lakhs for testing and quality assurance, ₹2 lakhs for UI design and documentation, and ₹2–3 lakhs for training, deployment, and administrative expenses. An additional ₹1–1.5 lakhs will be reserved as a contingency fund to handle unexpected costs. Overall, the project’s schedule and budget are structured to ensure timely delivery, technical efficiency, and cost-effectiveness while maintaining high usability.

Chapter 3: Class Modeling



writingPen — The hardware source of raw handwriting data (strokes, pressure, tilt, battery). Its relevance is isolating sensor/firmware concerns so all downstream logic works from consistent, timestamped input and so firmware updates or new pen models don't require changes to analysis or UI code.

writingPad — The capture/display surface and local buffer. It mediates between hardware and software: provides low-latency visual feedback to the child, pre-processes and stores stroke data, and ensures reliable delivery of session data to the analysis pipeline or cloud.

dataProcessor — The core analysis engine. It converts raw stroke events into meaningful features (segmentation, character recognition, tremor/noise detection) and identifies error patterns. Centralizing algorithms here keeps pedagogical logic separate from UI and hardware and makes it easy to swap or upgrade analytical methods.

FeedBackModule — Translates analysis results into child-friendly actions (visual cues, audio prompts, haptics). Its relevance is turning technical outputs into actionable, motivating guidance that supports learning in real time while allowing experimentation with feedback styles.

userProfile — Stores user-specific context (age, disability level, progress history). It enables personalization—adaptive difficulty, tailored feedback, and longitudinal tracking—and links session metrics to an identifiable learning trajectory for teachers/parents.

ExerciseModule — Manages practice content and assignments. It creates and schedules exercises, tracks completion, and ties curriculum to user performance so practice is structured, progressive, and measurable.

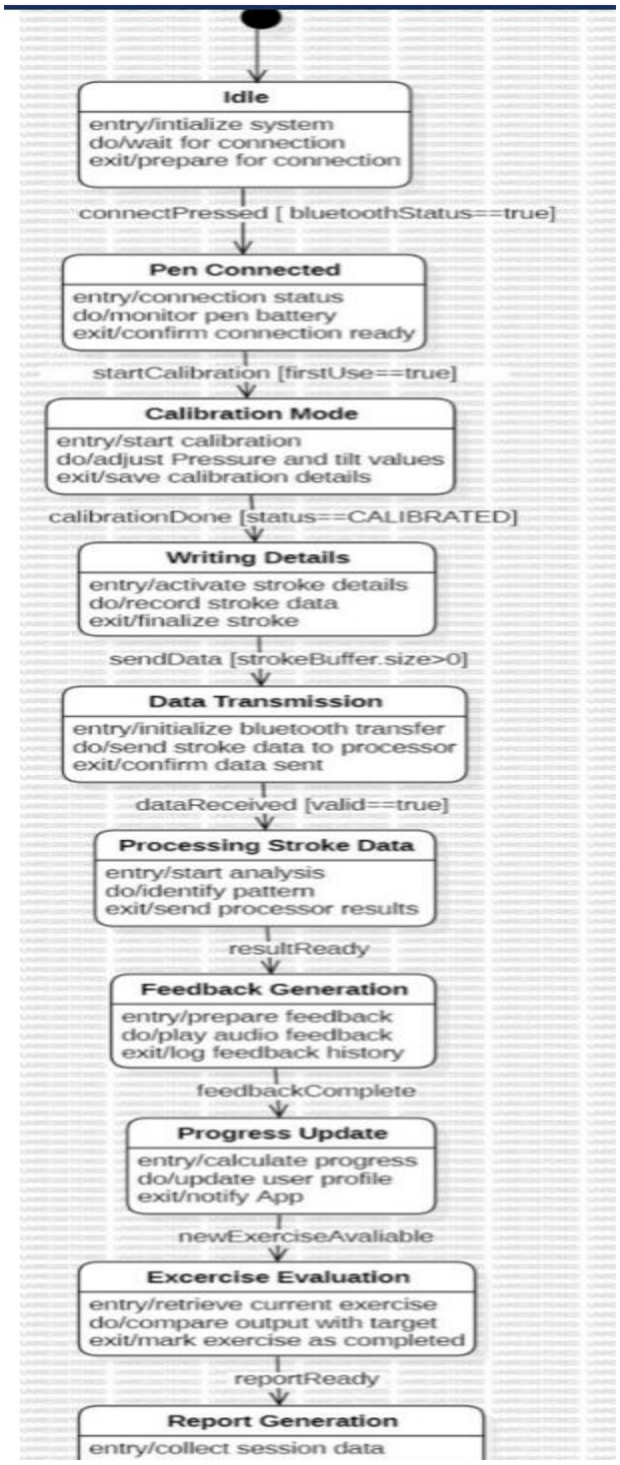
reportModule — Aggregates session and progress data into readable summaries (PDFs, charts). It's relevant for communicating outcomes to parents, teachers, or clinicians, supporting accountability and data-driven decisions about interventions.

MobileApp — The user-facing interface and orchestration layer. It connects devices, syncs data, shows dashboards, applies settings, and facilitates teacher/parent interactions—functionality needed to configure the system, review progress, and deploy updates or assignments.

Each class separates a distinct responsibility—hardware capture, local UI/storage, analysis, feedback, user management, exercises, reporting, and user interface—which together create a modular, maintainable system that supports real-time correction, personalization, and long-term progress tracking for children with learning disabilities.

Chapter 4: State Modeling

Smart Pen System Statechart


Idle
entry/intialize system
do/wait for connection
exit/prepare for connection
connectPressed [bluetoothStatus==true]
Pen Connected
entry/connection status
do/monitor pen battery
exit/confirm connection ready
startCalibration [firstUse==true]
Calibration Mode
entry/start calibration
do/adjust Pressure and tilt values
exit/save calibration details
calibrationDone [status==CALIBRATED]
Writing Details
entry/activate stroke details
do/record stroke data
exit/finalize stroke
sendData [strokeBuffer.size>0]
Data Transmission
entry/initialize bluetooth transfer
do/send stroke data to processor
exit/confirm data sent
dataReceived [valid==true]
Processing Stroke Data
entry/start analysis
do/identify pattern
exit/send processor results
resultReady
Feedback Generation
entry/prepare feedback
do/play audio feedback
exit/log feedback history
feedbackComplete
Progress Update
entry/calculate progress
do/update user profile
exit/notify App
newExerciseAvaliable
Excercise Evaluation
entry/retrieve current exercise
do/compare output with target
exit/mark exercise as completed
reportReady
Report Generation
entry/collect session data

1. Idle

Purpose: System startup / standby state where the device awaits user action or connection. **Entry:** Initialize system (load settings, check hardware).

Do: Wait for connection attempts; monitor for button presses or incoming pairing requests. **Exit:** Prepare connection resources (allocate Bluetooth sockets, power up comms).

Relevance: A safe default state that conserves resources and prevents any accidental data flow until an explicit connection occurs.

2. Pen Connected

Purpose: The pen and pad are paired and ready for operation.

Entry: Confirm connection status (handshake).

Do: Monitor pen battery, connection quality, and readiness for calibration or writing.

Exit: Confirm connection is ready to move into calibration or writing.

Relevance: Ensures stable communication and that hardware is healthy before collecting writing data.

3. Calibration Mode

Purpose: Set up pressure/tilt baselines and other sensor offsets for accurate stroke capture. **Entry:** Start calibration routine (triggered on first use or when user requests).

Do: Adjust and measure pressure/tilt min-max values; map sensor-to-coordinate transforms. **Exit:** Save calibration details to persistent settings.

Relevance: Calibration improves accuracy and personalization; without it, analysis may misinterpret strokes.

4. Writing Details

Purpose: Active capture of stroke-level details while the child writes.

Entry: Activate stroke recording and buffers.

Do: Record raw stroke sequences (time, x/y, pressure, tilt) into a stroke buffer.

Exit: Finalize stroke and prepare buffer for transmission when a stroke is completed. **Relevance:** This is the main data-collection state — correct recording here is essential for everything downstream (feedback, analytics, reports).

5. Data Transmission

Purpose: Transfer the digitized stroke(s) from pad to processing component (app or cloud).

Entry: Initialize Bluetooth/communication transfer.

Do: Send stroke buffer packets reliably (with acknowledgements or retries).

Exit: Confirm data has been sent and optionally clear local buffer.

Relevance: Reliable transfer prevents data loss and allows analysis to happen off-device or on a stronger compute module.

6. Processing Stroke Data

Purpose: Analyze the transmitted stroke data to extract features and detect errors.

Entry: Begin analysis pipeline (feature extraction, segmentation).

Do: Identify writing patterns, classify letters, detect tremors, spacing issues, formation errors.

Exit: Send structured analysis results (scores, error labels) to feedback generation.

Relevance: This is the "brain" step — quality of analysis directly affects the usefulness of feedback and exercise adaptation.

7. Feedback Generation

Purpose: Create and deliver child-friendly corrective or reinforcing feedback.

Entry: Prepare feedback messages and cues.

Do: Play audio prompts, show visual hints, or send haptic cues; log feedback in history.

Exit: Finalize feedback and notify the progress module that the feedback cycle is complete.

Relevance: Immediate, understandable feedback is what helps the child correct mistakes in real time — it operationalizes the analysis.

8. Progress Update

Purpose: Update user profile and progress metrics after a feedback/analysis cycle.

Entry: Compute adjustments to progress level and learning metrics.

Do: Update persistent user profile, notify the app/UI of updated progress.

Exit: Signal if new exercises should be unlocked/created.

Relevance: Maintains the learner's longitudinal record and enables adaptive learning (difficulty scaling, streaks, achievements).

9. Exercise Evaluation

Purpose: Compare the child's output against exercise targets to mark completion and quality.
Entry: Retrieve the current exercise goals/targets.
Do: Compare measured features (accuracy, time, pressure) with target thresholds.
Exit: Mark the exercise completed/failed and produce evaluation metadata.
Relevance: Determines mastery and readiness to progress; drives personalized next steps and reporting.

10. Report Generation

Purpose: Aggregate session data into a human-readable performance report (for parents/teachers).
Entry: Collect session logs, metrics, and exercise evaluations.
Do: Generate charts, summaries, and a report file (PDF/JSON).
Exit: Save/export the report for sharing or archival.
Relevance: Provides evidence of progress for stakeholders and supports decisions by teachers/therapists.

Events / Transitions

connectPressed [bluetoothStatus == true]

Trigger: User presses connect and Bluetooth is available.
Guard: bluetoothStatus == true ensures hardware is on and ready.
Why: Moves the system out of Idle to establish a reliable connection; prevents attempts when hardware is off.

startCalibration [firstUse == true]

Trigger: System flags first-time setup or user explicitly requests calibration.

Guard: firstUse == true indicates calibration is required.

Why: Ensures baseline sensor behavior is learned before capturing data, improving downstream accuracy.

calibrationDone [status == CALIBRATED]

Trigger: Calibration routine completes successfully.

Guard: status == CALIBRATED confirms successful calibration.

Why: Only proceed to normal writing when sensors are properly calibrated.

sendData [strokeBuffer.size > 0]

Trigger: Stroke data has been recorded and is ready to transmit.

Guard: There must be non-empty stroke data.

Why: Prevents sending empty payloads and triggers reliable transfer of meaningful content.

dataReceived [valid == true]

Trigger: Receiving component confirms the data arrived intact.

Guard: valid == true checks integrity (checksums, expected format).

Why: Ensures analysis operates on correct data; otherwise, repeat/repair transmission.

resultReady

Trigger: Processing module finishes analysis and returns results.

Guard: None shown (assumed implicit success).

Why: Signals feedback module to generate user-facing guidance.

feedbackComplete

Trigger: Feedback playback/display finishes and is logged.

Guard: None shown.

Why: Allows the system to update progress and move on to evaluations or next strokes — keeps the loop synchronous and traceable.

newExerciseAvailable

Trigger: Progress update determines that a new exercise should be created or unlocked. **Guard:** None shown.

Why: Enables adaptive curriculum flow based on the learner's current proficiency.

reportReady

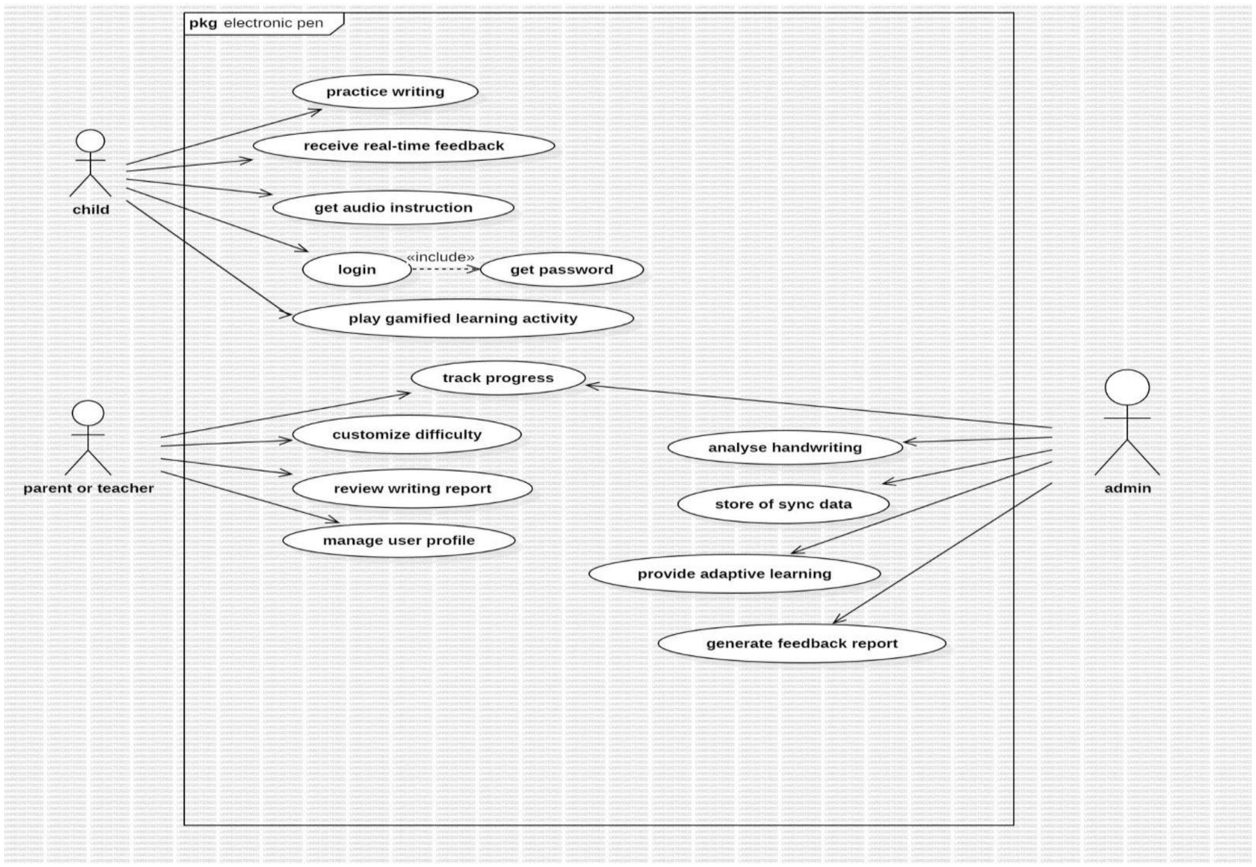
Trigger: Exercise evaluation or session aggregation completed and report files are generated. **Guard:** None shown.

Why: Notifies the app or backend that a report can be delivered or exported for stakeholders.

This statechart models the life cycle of the smart-pen system from startup to reporting: the device begins in Idle, moves to Pen Connected after Bluetooth pairing, optionally enters Calibration Mode to tune sensors, then captures strokes in Writing Details. Collected strokes are transferred in Data Transmission, analyzed in Processing Stroke Data, and transformed into child-facing actions in Feedback Generation. Outcomes update the learner record in Progress Update, which may trigger Exercise Evaluation and eventually Report Generation for teachers/parents. Each transition is triggered by explicit events (button presses, data availability, analysis completion) often guarded by conditions (e.g., bluetoothStatus == true, strokeBuffer.size > 0, status == CALIBRATED) to ensure reliability and correctness.

Chapter 5: Interaction Modeling

1. Use Case Diagram



1. Child (Primary User / Learner)

The child is the main end-user of the electronic writing pen system. This actor represents young learners, especially those with learning disabilities, who directly interact with the writing pad and receive real-time guidance.

Relevance:

· They initiate writing activities, receive live feedback, and improve literacy/motor skills.

· Their behaviour drives the system's primary functionalities such as handwriting capture, feedback generation, and exercise execution.

2. Parent or Teacher (Secondary Support User)

The parent or teacher oversees the child's progress and customizes the learning experience.

Relevance:

· They monitor reports, adjust difficulty levels, and manage user settings.

· They ensure that the child receives personalized learning paths and that progress is aligned with educational goals.

3. Admin (System Administrator / Backend Controller)

The admin maintains system functionality, data synchronization, analytics, and backend algorithms.

Relevance:

· Ensures the system runs smoothly by handling data storage, handwriting analysis, and generating global reports.

· Manages AI models, adaptive learning logic, and system-wide configurations.

· Maintains data integrity and supports teachers/parents with accurate analytics.

Relevance of Each Use Case

Use Cases for Child

1. Practice Writing

Allows the child to perform handwriting activities using the smart pen.

Relevance: Core function for developing handwriting skills and capturing stroke data.

2. Receive Real-Time Feedback

The child receives immediate visual/audio/haptic guidance.

Relevance: Helps correct mistakes instantly, improving learning efficiency and motor coordination.

3. Get Audio Instruction

Provides spoken instructions or cues during writing.

Relevance: Supports children who respond better to auditory guidance or have reading limitations.

4. Login

Children access their personalized learning environment.

Relevance: Ensures security, progress tracking, and personalized content.

5. Get Password

Triggered when the child forgets login credentials.

Relevance: Supports account recovery while maintaining secure access.

6. Play Gamified Learning Activity

The child engages with interactive, game-like exercises.

Relevance: Increases motivation and makes learning enjoyable, especially for children with attention difficulties.

7. Track Progress

Allows the child to view their improvement over time.

Relevance: Encourages self-motivation and helps learners understand their strengths and weaknesses.

Use Cases for Parent/Teacher

8. Customize Difficulty

Parents/teachers adjust exercise complexity based on the child’s ability.

Relevance: Ensures exercises match the child's developmental level.

9. Review Writing Report

Allows viewing of analytics, error analysis, accuracy trends, etc.

Relevance: Provides insights for intervention, reinforcement, or modification of learning strategies.

10. Manage User Profile

Parents/teachers modify user settings, preferences, and personal data.

Relevance: Keeps child data updated and relevant for personalization.

Use Cases for Admin

11. Analyse Handwriting

Admin side processes the child’s handwriting using AI algorithms.

Relevance: Provides backend intelligence for accurate feedback and assessment.

12. Store or Sync Data

Admin manages storage and synchronization of user data, handwriting samples, and progress logs.

Relevance: Ensures data availability across devices and prevents data loss.

13. Provide Adaptive Learning

The system adjusts the difficulty and content dynamically based on the child's performance.

Relevance: Delivers personalized learning paths to help children progress at their own pace.

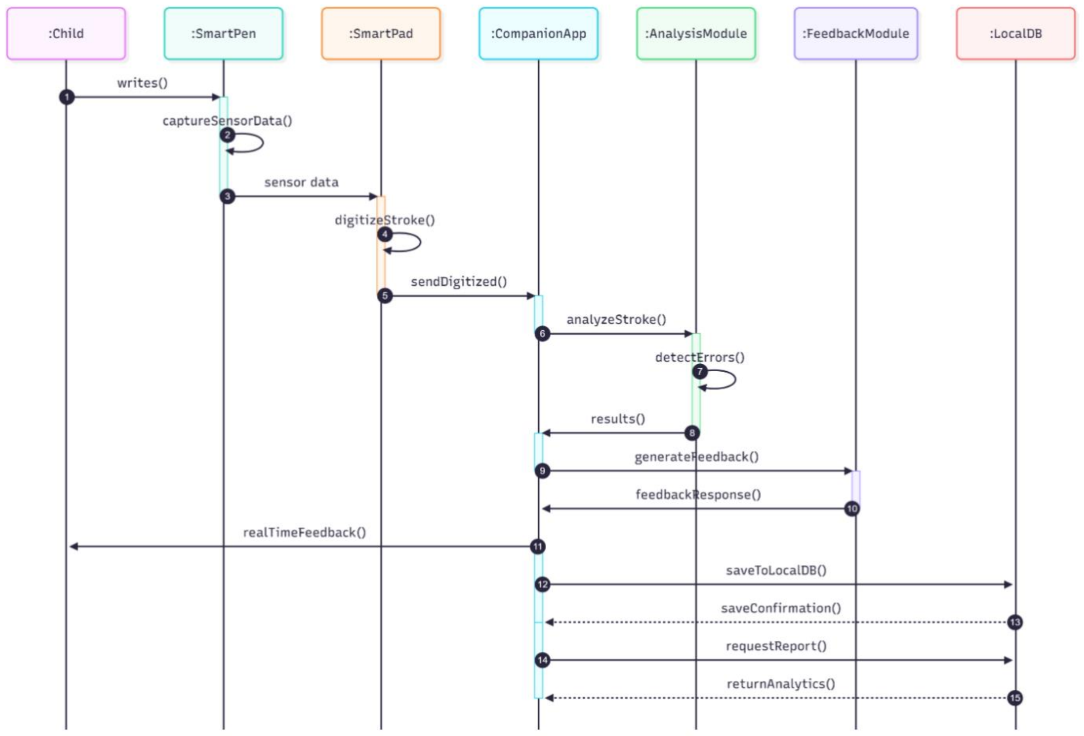
14. Generate Feedback Report

Admin triggers or maintains the module that produces performance reports.

Relevance: Supplies parents and teachers with understandable and actionable reports.

The use case diagram models an intelligent writing pen system designed for children with learning disabilities. The child serves as the primary user, performing writing activities and receiving real-time corrective feedback. The parent/teacher oversees progress, customizes difficulty, and manages learner profiles to support educational goals. The admin maintains backend services including handwriting analysis, data synchronization, adaptive learning, and report generation. Each use case directly contributes to improving the child's writing skills, supporting personalized and engaging learning experiences, and ensuring the system operates efficiently.

2.Sequence Diagram



Smart Handwriting Assistance System Workflow

1. Child → SmartPen: writes() The interaction begins when the child starts writing. This triggers the SmartPen to begin sensing motion, pressure, and tilt.

2. SmartPen: captureSensorData() The SmartPen enters a state where it collects raw sensor data, which includes stroke coordinates, pressure values, and tilt orientation.

3. SmartPen → SmartPad: sensor data After capturing, the SmartPen sends the raw sensor readings to the SmartPad for further handling. This ensures data is transmitted quickly with minimal computation on the pen itself.

4. SmartPad: digitizeStroke() The SmartPad receives the raw input and digitizes it—converting it into structured stroke data suitable for algorithms. This may include smoothing, coordinate conversion, and time-stamping.

5. SmartPad → CompanionApp: sendDigitized() The digitized stroke data is forwarded to the Companion App, which acts as the center of processing, UI display, and data management. **6. CompanionApp → AnalysisModule: analyzeStroke()** The Companion App sends the digitized stroke data to the Analysis Module to perform handwriting analysis.

7. AnalysisModule: detectErrors() The Analysis Module performs deeper analysis to identify handwriting issues, such as: Incorrect letter formation Uneven spacing Excessive pressure Slow or shaky strokes

8. AnalysisModule → CompanionApp: results() The processed results (errors, metrics, stroke classifications) are returned to the Companion App.

9. CompanionApp → FeedbackModule: generateFeedback() The Companion App requests feedback generation based on the analysis result—for example: "Write the letter straighter" "Reduce pressure" "Try again—spacing too large"

10. FeedbackModule → CompanionApp: feedbackResponse() The Feedback Module creates audio/visual/haptic feedback and sends it back.

11. CompanionApp → Child: realTimeFeedback() The child receives the feedback in real time on the smart screen or through audio cues, helping them correct the stroke immediately.

12. CompanionApp → LocalDB: saveToLocalDB() After feedback is delivered, the App stores the session data (strokes, feedback, errors) in the local database.

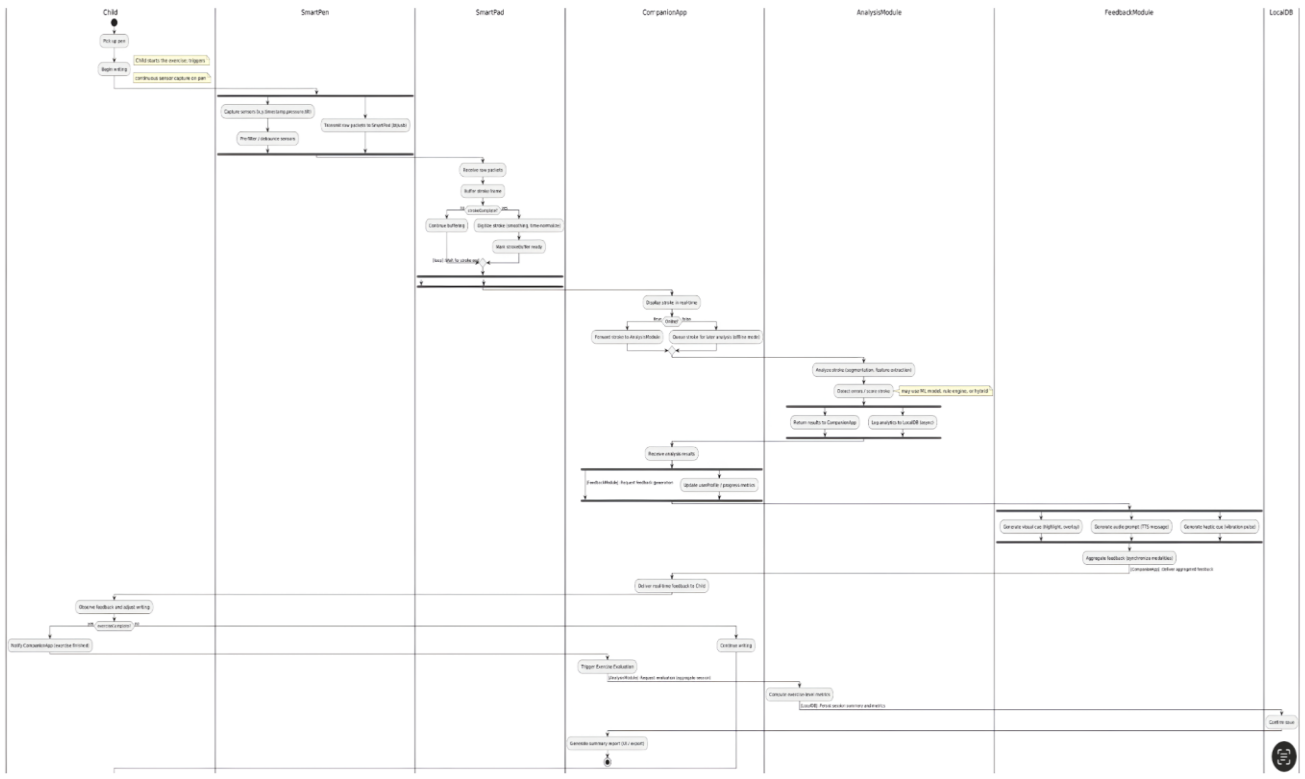
13. LocalDB → CompanionApp: saveConfirmation() The database confirms that the session data has been safely stored.

14. CompanionApp → LocalDB: requestReport() The user or system requests analytics or a performance summary.

15. LocalDB → CompanionApp: returnAnalytics() The database returns aggregated statistics (accuracy, error frequency, performance trends), which may later be used for reports.

This sequence diagram represents the complete workflow of the smart handwriting assistance system. When the child writes, the SmartPen captures sensor data and sends it to the SmartPad, which digitizes the stroke and forwards it to the Companion App. The App sends this data to the Analysis Module, which detects handwriting errors and returns results. The Feedback Module then generates corrective feedback that the child receives instantly. The session data is stored in the Local Database, and analytics can be retrieved later for reviewing progress or generating reports. Together, these interactions create a seamless loop of capture, analysis, feedback, storage, and evaluation

3. Activity Diagram


Child
SmartPen
SmartPad
CompanionApp
AnalysisModule
FeedbackModule
LocalDB
Pick up pen
Begin writing
Child starts the exercise; triggers
continuous sensor capture on pen
Capture sensors (x,y,timestamp,pressure,tilt)
Transmit raw packets to SmartPad (bt/usb)
Pre-filter / debounce sensors
Receive raw packets
Buffer stroke frame
strokeComplete?
Continue buffering
Digitize stroke (smoothing, time-normalize)
Mark strokeBuffer ready
[loop] Wait for stroke end
Display stroke in real-time
Online?
true
false
Forward stroke to AnalysisModule
Queue stroke for later analysis (offline mode)
Analyze stroke (segmentation, feature extraction)
Detect errors / score stroke
may use ML model, rule engine, or hybrid
Return results to CompanionApp
Log analytics to LocalDB (async)
Receive analysis results
[FeedbackModule] Request feedback generation
Update userProfile / progress metrics
Generate visual cue (highlight, overlay)
Generate audio prompt (TTS message)
Generate haptic cue (vibration pulse)
Aggregate feedback (synchronize modalities)
[CompanionApp] Deliver aggregated feedback
Deliver real-time feedback to Child
Observe feedback and adjust writing
exerciseComplete?
yes
no
Notify CompanionApp (exercise finished)
Continue writing
Trigger Exercise Evaluation
[AnalysisModule] Request evaluation (aggregate session)
Compute exercise-level metrics
[LocalDB] Persist session summary and metrics
Confirm save
Generate summary report (UI / export)

1. Child Swimlane

This swimlane represents the user (the child) who interacts directly with the writing pen and pad. Key responsibilities include:

· Starting the writing action by picking up the pen and beginning to write.

· Receiving real-time feedback from the system during writing.

· Adjusting handwriting based on feedback provided.

· Completing exercises and notifying the system when an activity is finished.

Relevance: This swimlane models the human–system interaction and shows how the child's writing action triggers the entire sensing, analysis, and feedback workflow.

2. SmartPen Swimlane

This swimlane represents the hardware pen device, equipped with pressure, tilt, and motion sensors. Key responsibilities include:

· Capturing raw sensor data, including stroke coordinates, pressure levels, and tilt.

· Pre-filtering and debouncing sensor readings to remove noise.

· Transmitting raw sensor packets to the SmartPad over Bluetooth or USB.

Relevance: The SmartPen is the entry point for handwriting data. It initiates the data pipeline and ensures high-frequency sensor capture.

3. SmartPad Swimlane

This swimlane represents the digital pad surface used for handwriting input. Key responsibilities include:

· Receiving raw sensor packets from the SmartPen.

· Buffering stroke data until the stroke is complete.

· Digitizing the stroke, including smoothing, noise reduction, and time normalization.

· Sending digitized stroke data to the CompanionApp.

· Saving a local backup of the raw stroke asynchronously.

Relevance: The SmartPad serves as a preprocessing and bridge device between hardware input and software processing. It ensures that the system has clean, well-structured stroke data.

4. CompanionApp Swimlane

This swimlane represents the smartphone/tablet application that manages user interaction and processing coordination. Key responsibilities include:

· Displaying the writing in real time for visual feedback.

· Forwarding digitized strokes to the AnalysisModule (when online).

· Queueing strokes for later analysis in offline mode.

· Receiving analysis results and triggering feedback generation.

· Updating user profile and progress metrics.

· Delivering real-time feedback to the child.

· Handling exercise completion, and triggering evaluation and reporting.

Relevance: This swimlane acts as the central controller, managing system logic, data flow, and communication between all modules.

5. AnalysisModule Swimlane

This swimlane represents the AI or rule-based engine responsible for handwriting interpretation. Key responsibilities include:

· Analyzing the stroke, including segmentation and feature extraction.

· Detecting handwriting errors such as shakiness, improper shape, spacing, or pressure issues.

· Scoring the stroke to measure performance.

· Returning analysis results to the CompanionApp.

· Logging results to the LocalDB for analytics and long-term tracking.

Relevance: The AnalysisModule is the "brain" of the system. It transforms raw writing into actionable insights that drive personalized feedback.

6. FeedbackModule Swimlane

This swimlane covers the subsystem responsible for generating corrective and supportive feedback. Key responsibilities include:

· Generating visual cues (highlighting mistakes or showing correct directions).

· Generating audio feedback through TTS (e.g., "Try reducing pressure").

· Generating haptic cues such as vibrations when available.

· Aggregating feedback from all modalities into a synchronized message.

Relevance: This module provides immediate learning assistance to the child, helping improve handwriting skills through multi-sensory feedback.

7. LocalDB Swimlane

This swimlane represents the local storage system on the device. Key responsibilities include:

· Storing raw stroke backups for reliability.

· Saving analytics logs from the AnalysisModule.

· Persisting session summaries and exercise metrics.

· Confirming successful saves to the CompanionApp.

Relevance: Provides persistence, progress tracking, and offline capabilities. Ensures that data is never lost and is available for reports and long-term analysis.

The swimlanes in the activity diagram clearly separate responsibilities among the Child, SmartPen, SmartPad, CompanionApp, AnalysisModule, FeedbackModule, and LocalDB. Each swimlane handles a distinct part of the workflow—from sensing handwriting and digitizing strokes to analyzing errors, generating feedback, storing data, and evaluating exercises. This division ensures clarity, modularity, and efficient coordination within the smart writing system, enabling real-time corrective support for children with learning disabilities.