
Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Bachelor Thesis

# Voice Assistants for Personalised Learning: Development and Evaluation of a Prototype in an Agile Development Context

Author
Pablo Larraz

Supervisors
Professor R. Schimkat
Javier Fabra

Bachelor
Computer Science

University of Technology, Business and Design Konstanz
Konstanz
2025

# Abstract

The rise of voice user interfaces and the demand for agile-ready graduates present an opportunity to rethink how Scrum is taught. This thesis explores the use of voice assistants as a tool for personalised, hands-on learning of core Scrum ceremonies. Drawing on research in cognitive styles, feedback theory, and adaptive learning, it defines requirements for an educational system that offers real-time, low-pressure practice. A cross-platform prototype was developed with a React Native frontend and a Python FastAPI backend, integrating local LLM inference (via Ollama), LangChain for prompt orchestration, and Pinecone for semantic retrieval. The assistant simulates four Scrum situations: Daily Stand-up, Sprint Planning, Product Backlog Management, and Sprint Review, delivering immediate feedback, personalised summaries, and progress tracking through a colour-coded history. Evaluation through structured self-tests showed the system to be responsive, coherent, and capable of adapting to individual learning patterns. While limited in scope, this work demonstrates the potential of voice-based systems to support reflective and adaptive agile learning.

# Index

# 1. Introduction

## 1.1 Background and Motivation

In recent years, digital learning has become a key tool in education, driven by its increasing accessibility and the growing demand for a flexible experience. Among these, voice interfaces have emerged as a promising innovation, enabling more natural and intuitive interactions. These systems, initially popularised for general consumer use, are gradually being adapted for educational purposes, offering new modalities for engagement, accessibility, and personalisation, granting the student a huge advantage.

On the other hand, the landscape of employability and professional development is evolving. Agile methodologies, particularly Scrum, have gained widespread adoption across industries, becoming a core component of collaborative project management and software development practices. For students, gaining familiarity with Scrum has become key in their future job applications. However, becoming good in Scrum involves more than memorising topics, as its practical use prevails over the learning of the concepts, which is the most common case in education centres.

The combination of voice technologies and agile education presents a unique opportunity: using voice assistants to facilitate active, role-based learning of Scrum. This thesis explores how voice-based systems can simulate Scrum environments and provide structured support for students, enabling them to practice ceremonies, internalise workflows, and receive feedback in a natural, adaptive, and low-pressure context, where the student will be able to thrive.

## 1.2 Research Objectives and Questions

The main objective of this thesis is to investigate how voice assistants can enhance the learning of Scrum methodology by simulating interactive scenarios and providing personalised feedback. The document focuses on the pedagogical potential of conversational interfaces, their adaptability to individual learner preferences, and their capacity to support simulated agile education.

This goal is explored through the following research questions:

- How can voice-based systems be designed to adapt to individual learning preferences and cognitive styles in order to improve learning outcomes?
- What are the advantages and limitations of voice interaction in enhancing learner engagement, motivation, and knowledge retention compared to traditional learning methods?
- What technologies and design strategies enable effective real-time feedback in voice-based learning systems?

These questions guide both the theoretical part and the practical design of the prototype.

## 1.3 Scope and Limitations

This thesis sits at the crossroads of educational technology, voice interfaces, and agile learning, with Scrum as the test case. I use voice assistants to recreate four key Scrum rituals: daily stand-ups, product backlog grooming, sprint reviews, and sprint planning, so that students can practise them through guided conversation. It keeps things voice-only: no screens, visuals, or haptics. It also targets learners in universities and tech programs, not professional Scrum teams.

To stay portable, the prototype runs a compact language model on local hardware. That choice keeps response times short but can limit the depth and accuracy of reasoning. Time and resource bounds also mean the evaluation is small-scale, so it cannot track long-term learning or guarantee a good adaptation by groups of students.

Even with these limits, the project lays the groundwork for future research on voice-driven, hands-on learning in agile and other collaborative fields.

## 1.4 Use of AI Tools

Early on in this thesis, I leaned on ChatGPT (OpenAI, 2025) for help. I used it to bounce around rough ideas, sketch out early outlines for the chapters, dig up promising papers on voice-enabled learning tech and Agile practices, and get quick pointers or snippets of code. Everything it suggested was just a starting point, that was later expanded, or totally re-written by me to keep things accurate and on-target.

# 2. Theoretical and Conceptual Foundations

## 2.1 Historical Background and Technological Evolution

### 2.1.1 Evolution

Voice user interfaces (VUIs) have been chasing natural conversation since the early 1950s. Bell Labs' AUDREY prototype (1952) was the first to recognise spoken digits with respectable accuracy, albeit only from its inventor's voice. IBM raised the bar a decade later: the suitcase-sized Shoebox, shown at the 1962 Seattle World's Fair, understood 16 words, proof that voice commands might one day replace key-pads (Wikipedia contributors, 2024a). During the 1970s, the U.S. Defence Advanced Research Projects Agency funded the *Speech Understanding Research* programme, which produced systems such as Harpy (and sister projects like Hearsay-II) capable of vocabularies above 1,000 words, an early glimpse of multi-module blackboard architectures (Wikipedia contributors, 2025). Mass-market use, however, had to wait. The first commercial wave arrived with telephone interactive-voice-response (IVR) platforms in the mid-1980s, followed by PC dictation suites in the 1990s. Dragon Dictate (1990) and Dragon NaturallySpeaking 1.0 (1997) offered continuous speech input, while IBM ViaVoice (1997) shipped multilingual packages for Windows and Mac, though each demanded lengthy enrolment sessions and hefty CPUs (Wikipedia contributors, 2024b).

A major shift came once speech moved to the cloud. Siri debuted on the iPhone 4S in 2011, fusing on-device intent parsing with server-side recognition, and proved that voice could be a core feature of a mainstream OS. Amazon Echo (2014) introduced Alexa with far-field microphones and elastic GPU clusters so that the assistant could always listen without overheating or overloading the local device. Google Assistant followed at I/O 2016, adding true multi-turn context to everyday devices (Kostaciński, 2022). The 2020s mark the era of transformer-scale language models that merge speech recognition, intent detection and text generation into fluid, chat-like exchanges. Amazon's paid tier, Alexa Plus (2025), now rewrites prompts on the fly, remembers personal preferences and even drafts songs, signalling the shift from command lists to personalised conversation (Davis, 2025). Meanwhile, microphones and embedded NLP chips have permeated almost anything with a battery. The in-car voice-assistant market was valued at roughly US$3.45 billion in 2024 and is projected to quadruple by 2033 (Quantum Market Research, 2025). Wearables tell a similar story: Meta's Oakley HSTN AI glasses (launched June 2025) combine a five-mic array with on-device Meta AI so athletes can get verbal coaching mid-workout.

Industry analysts and UX commentators agree that VUIs are no longer a novelty but a cross-platform expectation, from smart speakers to web apps and industrial tools. This widespread use lays the groundwork for education: learners can rehearse Scrum stand-ups, practise language drills or follow lab protocols wherever they are, without needing to find a screen or keyboard. This rapid evolution underpins why educational applications are now

feasible, modern VUIs are ubiquitous and powerful enough to support complex tasks like Scrum training.

### 2.1.2 Pedagogical Role and Applications

Modern VUIs fit perfectly into learner-centred teaching because they let students talk over concepts instead of bumping into menus and buttons. Recent survey data from Boston Children's Digital Wellness Lab shows that even primary-age learners already treat assistants as semi-teachers, with many families reporting that children use them to quiz facts or ask for explanations during homework sessions when the interaction channel is natural speech, cognitive load shifts away from operating the system and back onto the material itself (Bickham et al., 2024, p.6).

In practice, VUIs can serve different purposes:

- **Guided tutor.** A voice agent can walk learners through step-by-step tasks or problem sets, offering hints the moment it detects hesitation.
- **Accessibility gateway.** A 2024 systematic review of 56 studies confirms that voice assistants meaningfully lower access barriers for students with visual impairments, enabling them to navigate resources that screen-readers still mangle (Millán Martínez et al., 2024).
- **Language-practice partner.** In second-language teaching, computer-assisted pronunciation tools that rely on automatic speech recognition consistently improve segmental accuracy and learner confidence (Thi-Nhu Ngo et al., 2024, p. 4).
- **Hands-free companion for technical training.** Voice-activated lab aides now integrate with ELN/LIMS software to record data, facilitating learners' lives, efficiency and reinforcing safety protocols (CSols Inc., 2024).

### 2.1.3 Current Capabilities

First, they can change content dynamically by having individual profiles and past interactions, guiding each student along a path that reflects their current knowledge, interests and long-term goals. Second, VUIs provide real-time feedback during drills, whether it is spelling, grammar or math, instantly correcting errors and walking learners through each step of a problem (Schön et al., 2023). Third, they integrate seamlessly with learning management systems, classroom IoT devices and smart whiteboards, ensuring that voice-driven activities stay synchronised across all platforms (Mierow, 2019).

Beyond these core functions, several emerging capabilities are reshaping how VUIs support education. Emotion-aware interaction by analysing tone, pace and word choice, assistants can now infer a learner's emotional state, offering encouraging prompts if frustration is detected or slowing down when the flow of the conversation sinks, though this remains an experimental feature (Ma et al., 2023, p. 528). Multimodal support now that VUIs work closely with gesture, touch and on-screen visuals, creating a complementary learning experience that lets students switch seamlessly between voice commands, hand movements

and display feedback (Dritsas et al., 2025, p. 2). Multi-user recognition and task juggling in scenarios like classrooms, modern systems can distinguish between different speakers, manage multiple simultaneous requests and route each student's query to the right profile.

### 2.1.4 Limitations and Challenges

On the other hand, voice interfaces present some obstacles that hold back their full potential.

- **Technical constraints:** During my own testing, I could see how speech recognition engines slip when the room is loud, when learners have different accents, or when they switch languages mid-sentence. Misfires break the conversational flow and make students repeat themselves.
- **Pedagogical gaps:** Most educational assistants run on scripted dialogue paths and shallow learner models. They struggle to read frustration, boredom, or confusion, so they cannot always adjust pacing or depth in real-time.
- **Privacy and ethics:** Because many systems send raw audio to cloud servers for processing, every utterance becomes data to be stored, analysed, and possibly shared. Protecting children's voices and complying with data-protection rules remains a thorny issue for schools and universities.
- **Limited domain coverage:** Outside the topics the model was trained on, answers can turn vague or wrong. A flawed response can diminish trust and push learners back to traditional resources.
- **Risk of over-personalisation:** If the assistant jumps in with hints at every stumble, students may lean on it too heavily and miss out on critical thinking.
- **Interface invisibility:** With no visible menu to guide them, users often wonder what to do next. Unless the system actively signposts options or follow-up questions, confusion and drop-offs follow.

These challenges show the need for stronger on-device models, user-centred design, and transparent consent flows. Until accuracy, adaptability, and ethics progress together, voice tools will remain helpful supplements rather than standalone tutors. Even so, the ongoing shift toward emotionally aware, context-rich dialogue suggests a next generation of VUIs that can better match diverse learner needs while safeguarding privacy and encouraging genuine inquiry.

(Silva et al., 2024)

## 2.2 Cognitive Styles & Personalisation in Educational Tech

### 2.2.1 Definition of Cognitive Styles

Cognitive styles refer to the preferred ways individuals process information, solve problems, and engage in learning. Unlike intelligence, which is often measured through standardised assessments, cognitive styles are about how people think rather than how well they think.

Understanding these styles is essential for developing educational technologies that can adapt to the diverse needs of learners (Wikipedia contributors, 2021).

Several models have been proposed to categorise cognitive styles. For instance, the **Kolb Learning Style Model** identifies four learner types based on a two-dimensional model:

- **Concrete Experience vs. Abstract Conceptualisation**
- **Active Experimentation vs. Reflective Observation**.

The resulting types, **Diverging, Assimilating, Converging, and Accommodating**, represent different preferences in how learners perceive and process information (Kolb, 1984, p. 83-85).

Another influential model is the **Felder-Silverman Learning Styles Model**, which classifies learners along four dimensions:

- **Sensing vs. Intuitive**: concrete facts vs. abstract concepts
- **Visual vs. Verbal**: graphical vs. textual representation
- **Active vs. Reflective**: trial-and-error vs. introspective thinking
- **Sequential vs. Global**: linear steps vs. holistic patterns

(Felder & Silverman, 1988, p. 676)

### 2.2.2 Limitations of fixed learning styles

A substantial body of empirical work now shows that matching instruction to a learner's declared "visual," "auditory," or similar style rarely yields measurable gains. Pashler et al. (2008) meta-review found "virtually no evidence" that style-by-treatment interactions improve outcomes once rigorous experimental criteria are applied, concluding that resources are better spent on evidence-based practices rather than on "meshing" instruction to style labels (Pashler et al., 2008, p. 116). Earlier, Coffield (2004, p. 138)'s systematic review of seventy-one style inventories noted serious measurement flaws, poor reliability, weak predictive validity, and contradictory typologies, and warned that the sheer proliferation of taxonomies risks obscuring genuine learning processes. More recent syntheses add that reported preferences are unstable, shifting with subject matter and context, so inventories capture transient "learner states" rather than enduring traits (Straub, 2025). Because static style labels neglect how competence, motivation, and affect fluctuate during study, adaptive-learning scholars now advocate dynamic learner modelling. Real-time analytics drawn from click-streams, response latencies, and affect detectors can trace evolving engagement and adjust pedagogy in real time; studies in intelligent tutoring systems such as ASSISTments show that modelling affect alongside knowledge better predicts performance than knowledge tracing alone (Corrigan et al., 2015, p. 101). Parallel-process growth-mixture models of language-learning motivation likewise demonstrate that motivational trajectories, not fixed categories, explain variance in achievement over semesters (Yu et al., 2022).

### 2.2.3 A Dynamic Framework of Personalised Education

Tetzlaff et al. (2021) argue that effective personalisation requires us to check in with learners continuously and adjust our approach based on real-time data. They describe three levels of adaptation:

- **Long term (weeks–months):** Plan the learning journey around each student's existing knowledge and overarching goals.
- **Medium term (days–weeks):** As learners work through a topic, tweak the sequence or depth of activities whenever misconceptions pop up.
- **Short term (seconds–minutes):** During practice, offer hints or rephrase questions on the fly if a learner hesitates or makes errors.

In practice, this means treating personalisation as an ongoing cycle rather than a linear sequence: gather data, update the learner's profile, and use that information to inform the next step. This keeps the learning path flexible and responsive instead of a one-time placement (Tetzlaff et al., 2021, p. 877).

### 2.2.4 Adaptivity

Adaptivity refers to the software's ability to reshape its behaviour as it gets to know the user, and in voice-based tools, this trait manifests in several ways. Personalised pathways emerge as conversational AI modulates difficulty or shifts topics after each turn, while dynamic dialogues tackle multi-turn context tracking to support follow-up questions and explanations right away, maintaining coherent conversation over time (Halkiopoulos & Gkintoni, 2024, p. 5). Temporal adaptation allows certain agents to shorten or pause sessions when they sense attention diminishes, protecting working-memory resources (Chen et al., 2021, p. 2). Complementing these features, voice-tone sensitivity enables systems such as SensEmo to gauge frustration or confidence from acoustic and physiological cues, adjusting pacing or encouragement in real time and reportedly boosting quiz scores by as much as 40 per cent (Choksi et al., 2024, p.8).

### 2.2.5 Personalisation as a Relational Aspect

Fielding points out that education is fundamentally about relationships. Drawing on John Macmurray's ideas, he argues that personalisation works best when it is a team effort built on open dialogue, mutual support and shared responsibility. Ignoring these human aspects risks reducing learners to mere data points and weakening democratic values in the classroom (Fielding, 2012, p. 690). Building on this, John Macmurray's own educational writings highlight the moral and communal foundations of learning. In his lecture "Learning to be Human," he argues that nurturing emotional development and authentic community bonds is essential for genuine personalisation and human flourishing in education.

Mills et al. (2014) provide a complementary policy perspective by analysing how mandated differentiation in Queensland's public schools often falters without clear definitions and

teacher support. Their study warns that differentiation risks becoming an audit ritual rather than a substantive educational shift unless educators receive practical guidance and professional learning (Mills et al., 2014, p. 340-342).

### 2.2.6 Educational Recommender Systems

Recommender systems help learners navigate large amounts of content by suggesting what to study next based on data. Kundu et al. (2021) identify five common approaches:

- Non-personalised (stereotyped): Everyone follows the same sequence of materials. This can work as a simple starting point, but quickly feels irrelevant to individual needs.
- Product-association: Similar to online shopping recommendations, this method introduces related resources that others have found useful.
- Content-based: Uses a learner's profile, such as their chosen Scrum role or prior quiz results, to match materials with relevant keywords or topics.
- Collaborative filtering: Finds learners with similar interaction patterns (e.g., quiz scores or time spent on activities) and recommends content that peers have used successfully.
- Hybrid systems: Combine two or more of the above strategies to balance their strengths and weaknesses, for instance, using content-based filtering to handle new users and collaborative filtering as more interaction data becomes available.

Hybrid recommenders are often the most effective because they adapt as the learner's history grows while still leveraging collective insights from the wider cohort. To keep recommendations fair and understandable, these systems should provide simple explanations and allow learners to give feedback on each suggestion. This transparency helps maintain trust and lets learners feel in control of their own path.

Trackable recommender pipelines allow the system to surface just-in-time resources, whether it is a quick hint on a difficult concept or a follow-up activity after a lesson, based on the learner's interaction data. By combining data-driven suggestions with clear, user-friendly explanations and feedback loops, these systems maintain trust and help learners feel guided rather than dictated to.

Synthesis Integrating these strands, effective personalisation demands that we:

1. **Monitor learner state continuously** (dynamic framework).
2. **Translate policy aspirations** into clear, resourced classroom practices (differentiation literature).
3. **Protect relational and democratic values** by fostering dialogue and agency alongside data analytics.
4. **Deploy transparent recommender pipelines** that adapt at macro, meso and micro horizons without entrenching bias.

Taken together, the section reframes personalisation as a data-informed, ethically grounded and time-sensitive process, moving the thesis beyond static learning styles toward a whole, responsive vision.

(Kundu et al., 2021, p. 90)

# 2.3 Feedback

## 2.3.1 Why feedback matters

Feedback links a learner's current performance to the next target, closing the gap between *what is* and *what should be*. Hattie & Timperley's seminal model frames effective feedback around three questions: *Where am I going?*, *How am I doing?*, and *Where to next?*, and shows effect sizes almost double the average impact of regular instruction. A more recent meta-analysis reaches similar conclusions, stressing that quality matters more than quantity. For voice-first systems, that quality boils down to prompt responses**,** context-aware, and pitched to the individual (Wisniewski et al., 2020).

## 2.3.2 Forms of feedback in ed-tech

- **Timing.** Immediate feedback excels in procedural drills, while a delay can deepen reflection during concept learning. Controlled experiments still find a small but significant edge for "right-away" feedback on conceptual tasks, provided the follow-up test interval is held constant.

- **Corrective vs. elaborative.** Simple "right/wrong" cues are fast, but elaborative explanations boost transfer because they unpack why an answer works.

- **Scaffolded feedback.** Gradually fading hints mirror apprenticeship, easing the learner toward independence.

- **Affective feedback.** When a voice assistant adapts its tone, pacing and wording to the learner's emotional state, engagement and persistence rise; systems that add real-time affect detection and supportive back-channels significantly increase users' self-disclosure and perceived emotional support (Cho et al., 2022).

Integrating several of these layers lets a voice agent address both the cognitive *and* emotional sides of learning.

(van der Kleij, 2013, p. 55)

## 2.3.3 Scaffolding through voice

Scaffolding, grounded in Vygotsky's Zone of Proximal Development, is essentially helpful with an expiry date. Recent reviews confirm that the right amount of scaffolding accelerates

problem-solving skills, while over-help breeds dependence. A voice assistant can operationalise scaffolding by prompting when the learner hesitates, rephrasing or simplifying instructions, offering examples or analogies tied to the learner's context or fading these supports as accuracy improves (Raslan, 2024, p. 64-65).

Timely, personalised, and context-sensitive feedback is the heartbeat of effective learning, and voice interfaces can deliver it at conversational speed. Coupled with adaptive mechanisms and scaffolded support, a well-designed VUI can replicate much of the learning guidance once limited to human tutors, while real-time affect detection opens the door to genuinely empathetic teaching agents. Continued work on privacy, deeper learner models, and richer dialogue management will decide how far these promises travel from research lab to everyday classroom.

## 2.4 Agile Methodology in Educational Contexts

### 2.4.1 Understanding Scrum in Agile

Agile methodologies are iterative frameworks designed to improve flexibility and responsiveness in software development and project management. Among the most used is Scrum, a lightweight framework that promotes incremental development through short time-boxed iterations called sprints. It consists of structured roles (Product Owner, Scrum Master, Development Team), events (Daily Stand-up, Sprint Planning, Sprint Review, and Sprint Retrospective), and artefacts (Product Backlog, Sprint Backlog, and Increment). The core principles emphasise collaboration, adaptability, transparency, and continuous feedback, which align closely with modern teaching goals in education.

Moving Scrum into a course lets students go further than memorising a few buzzwords. When they have to run a short sprint themselves, prioritising work, adjusting to surprises, and talking through what went well or badly, they get a feel for how real software teams behave. The exercise also forces them to practise soft abilities such as clear speaking, quick problem-solving and sharing leadership, which are hard to pick up from lectures alone.

(Schwaber & Sutherland, 2020)

### 2.4.3 Role of Voice Assistants in Scrum-Based Learning

Voice assistants can effectively simulate Scrum environments and provide structured support for students engaging in role-based agile learning. By leveraging natural language interaction, these systems facilitate hands-on experience with Scrum ceremonies, allowing learners to internalise agile principles through guided participation. In this project, the voice assistant supports four key Scrum situations:

**1. Daily Stand-up Meetings**

In this scenario, the voice assistant initiates the stand-up and sequentially asks each team member the three standard questions:
“What did you do yesterday?”, “What will you work on today?”, and “Is there anything blocking your progress?”
The student assumes the role of a Scrum team member and practices articulating daily updates. This promotes structured communication and reflection, while the assistant can detect blockers and summarise responses for feedback purposes.

**2. Product Backlog Management**

Here, the assistant acts as a facilitator, reading out current backlog items and prompting the student with questions about prioritisation and refinement, such as whether to adjust the priority or rephrase an item. This scenario enhances the student's understanding of backlog refining, prioritisation strategies, and stakeholder-driven adjustments in a simulated but realistic setting.

**3. Sprint Review**

The assistant opens the Sprint Review session by asking reflective questions like: “What goals did the team achieve this sprint?” and “What challenges did you face?”. The student, acting as a developer or team representative, presents completed work and anticipates feedback from stakeholders. This encourages the development of communication and presentation skills, while the assistant helps structure the session and ensures coverage of all relevant points.

**4. Sprint Planning**

In the Sprint Planning simulation, the assistant guides the student through task selection and goal setting. It prompts questions such as: “What items should be selected from the backlog?” and “What is the sprint goal?”. The student, acting as a Developer, defines the sprint scope and breaks down tasks. The assistant ensures that planning is thorough, reinforcing prioritisation, estimation, and goal-oriented thinking.

These scenarios enable learners to engage in immersive and adaptive Scrum practice sessions. By simulating realistic team interactions, the voice assistant enhances experiential learning, reinforces agile thinking, and supports the development of communication, planning, and reflection skills essential in both educational and professional agile environments.

### 2.4.4 Educational Value of Agile Simulation

That kind of guided simulation gives a safe space to experiment. Students can try different tactics, hear instant feedback, and, if they get something wrong, adjust on the next attempt without real-world consequences. It also nudges them to think about their own process, how

clearly they spoke, whether their plan matched the goal, and how quickly they adapted, skills that matter well beyond Scrum itself.

# 3. Related Work

Voice assistants are increasingly embedded in both formal and informal learning. Alexa alone lists many educational skills in the German store, ranging from flash-card quizzing to full conversational tutoring. Yet large-scale classroom adoption lags behind marketing hype because teachers must balance added value with concerns about privacy, moderation and reliability. Research gaps remain around long-term learning outcomes, fair access, and sustainable skill-development ecosystems.

## 3.1 Voice-Assistant Ecosystem: Platforms and Development Models

Amazon's Alexa Skills Kit (ASK) lets educators create custom skills without extensive coding, while the newer Alexa Skills Inventor brings block-based programming into K-12 CS lessons, lowering the entry barrier for students themselves.

Google's Actions on Google offers comparable capabilities, though it has fewer education-specific templates and has shifted toward Android-first integrations.

Both ecosystems share a revenue-free, data-exchange business model, skills are free to publish, but rely on cloud meters for usage. This model has limited third-party momentum, and Amazon publicly admitted that sustained developer engagement is a challenge.

**Taxonomy of Educational Skills**

| Category | Typical pedagogical goal | Example skill |
|---|---|---|
| **Micro-content & Flash Briefings** | Deliver daily nuggets, retrieval practice | Teacher-curated Flash Briefing for art-history facts (Hunt & Hudson, 2024, p. 142) |
| **Quizzing & Formative Assessment** | Check recall, give instant feedback | ASK tutorial "Quiz Builder" sample (Davie & Hilber, 2018) |
| **Language & Pronunciation Tutors** | improve oral fluency, vocabulary | Japanese self-study skills |
| **Classroom Management / SEL** | timers, mindfulness, and emotion regulation | *AskMyClass* for social-emotional activities |

| | | |
|---|---|---|
| **Accessibility Helpers** | hands-free navigation, multimodal prompts | Eye-Gaze on Alexa for motor impairments (Amazon staff, 2023) |

## 3.2 Empirical Evidence of Learning Impact

### 3.2.1 Language Learning

A 2022 exploratory case study tracked six adult learners of Japanese who integrated Alexa into out-of-class drill sessions for six weeks. Usage logs showed that Alexa correctly recognised about 78 % of the learners' Japanese utterances; when errors occurred, students either reformulated the command or abandoned the attempt. Post-intervention surveys (TAM-style) revealed high ratings for perceived usefulness, enjoyment, and convenience as a pronunciation and vocabulary supplement, tempered by mild frustration with occasional misrecognitions (Dizon et al., 2022).

### 3.2.2 STEM & Computational Thinking

The *Alexa Skills Inventor* programme reports thousands of teachers and tens of thousands of students around the globe building block-code skills that reinforce conditional logic and slot-filling dialogue design (Amazon Science Staff, 2023).

### 3.2.3 Social-Emotional & Executive-Function Support

Teachers who use AskMyClass timers, breathing exercises, and study breaks report that pupils refocus up to three times faster after high-energy activities and that transitions cause fewer disruptions (Casey, 2019). An experimental study with 40 adults found that an Alexa prototype which adds short active-listening back-channels—"Mm-hmm," "I see"—elicits deeper self-disclosure and greater perceived emotional support (Cho et al., 2022, p.9). The same technique could be repurposed for reflective journaling tasks in the classroom.

## 3.3 Teaching Opportunities and Design Principles

Voice interfaces eliminate the friction of typing, letting learners verify understanding on the fly and switch rapidly through speaking-practice loops. Because smart speakers remain always on, they can slip unobtrusive micro-learning moments into the gaps between other activities, which is why their wake words must be tuned so the devices do not intrude on classroom chatter. When a screen is available, as on an Echo Show or Nest Hub, visual examples such as images and concise captions can reinforce spoken content and tap the power of dual coding. Translating these affordances into effective pedagogy means maintaining minimal cognitive load in each turn of dialogue, asking for explicit

confirmations during assessment tasks, and offering opt-in logging to uphold privacy commitments.

## 3.4 Limitations, Risks and Ethical Considerations

- **Privacy and Data Sovereignty**: Persistent microphones record classroom ambience; U.S. lawmakers questioned Amazon's data-retention policies for Echo Dot Kids Edition as early as 2018 (Lapowsky, 2018). The GDPR and Germany's SchulCloud guidelines mandate local-server options or anonymised transcripts, features not yet natively supported.

- **Skill Discoverability & Maintenance**: Although Amazon advertises many educational skills, classroom-ready offerings are hard to surface, and many suffer from abandonment or outdated content (Pierce, 2024).

- **Speech-Recognition Bias**: Classroom trials with Amazon Echo found that young children's unclear articulation and non-native English accents often led Alexa to misrecognise answers, forcing pupils to repeat or rephrase (Davie & Hilber, 2018, p. 206). Such systematic errors can frustrate learners and create inequitable feedback loops.

- **Teacher Cognitive Load**: Integrating voice workflows demands lesson-design changes and may divert attention if the device misfires; professional-development programmes remain sparse (Davie & Hilber, 2018).

## 3.5 Synthesis of Related Work

Across recent studies, voice-assistant skills have matured from quirky demos into useful tools for language drills, quick formative checks, and accessibility support. Still, their long-term value in education will depend on three factors: clear data-privacy policies, smarter ways for learners to discover the right skill at the right moment, and learning designs grounded in solid evidence. How well the field tackles these issues will decide whether assistants like Alexa become core classroom infrastructure or stay on the sidelines.

# 4. Methodology

## 4.1 Prototype Ideas

Once all the research has been done, many ideas have arisen on how the voice assistant can support personalisation and pedagogy, granting the student a rewarding experience when learning Scrum.

**User profile & preference customisation**

Learners can pick the assistant's voice persona, set preferred session lengths, or toggle between different feedback styles. By letting students shape the tone and pace themselves, the system feels more like a trusted mentor than a generic app, reinforcing autonomy and motivation.

*This idea comes from (Tetzlaff et al., 2021).*

**Adaptive content and difficulty**

Every response is evaluated in real time, so when answers are consistently strong, the assistant raises the bar with multi-team scenarios or time-boxed challenges, while persistent errors trigger simpler questions or extra hints.

*This idea comes from (Corrigan et al., 2015).*

**Continuous and personalised learning profile**

Each practice session updates a long-term learner model stored locally. The richer this profile grows, the sharper the assistant's predictions become, suggesting the next best ceremony to rehearse, recalling past blockers, or reminding the student of prior feedback when a similar mistake reappears.

*This idea comes from (Corrigan et al., 2015).*

**Immediate formative feedback**

After every spoken answer, the assistant responds instantly with a concise judgement, pointing out omissions or praising clarity. Fast feedback reinforces correct habits before errors harden into routines.

*This idea comes from (Wisniewski et al., 2020).*

**Contextual hints and guidance**

If the system detects hesitation, long pauses or filler words, it prompts a micro feedback such as "Try framing your update in one sentence". These nudges keep the conversation flowing and help nervous students regain focus without derailing the exercise.

*This idea comes from* (Raslan, 2024).

**Detailed explanations and summaries**

At the end of each simulation, the assistant delivers a structured debrief: strengths, areas for improvement, a brief rationale linked to Scrum theory, and an overall score. Students leave every session knowing exactly what to fix next.

*This idea comes from* (Wisniewski et al., 2020).

**Scaffolded feedback**

Complex ceremonies are broken into smaller steps. First, define the Sprint Goal, then select backlog items, and then estimate effort. As competence rises, the scaffold is removed, encouraging independence while preventing early overload.

*This idea comes from* (Raslan, 2024).

**Pacing and timing adaptation**

Speech rate, pause length, and question cadence adjust automatically to the user's tempo. A reflective learner gets slower TTS and longer think-time; a rapid-fire user experiences brisk, stand-up-like exchanges. This alignment respects individual processing speed yet honours Scrum's time-boxed nature.

*This idea comes from* (Chen et al., 2021).

**Knowledge-gap targeting**

Analytics highlight persistent blind spots, perhaps the learner often omits Done criteria or misorders backlog priorities. The assistant gets the previous performance's history and tunes the situations and feedback accordingly.

This idea comes from (Corrigan et al., 2015).

**Emotion recognition from voice**

Basic sentiment cues trigger empathetic responses, slower pacing, encouraging phrases, or an offer to pause. By acknowledging affect, the assistant sustains engagement and reduces drop-out risk during tough topics.

*This idea comes from* (Choksi et al., 2024).

**Tone and style adaptation**

The assistant's own voice shifts with context: neutral and formal during assessment, upbeat for praise, calm when the learner sounds tense. Matching style to the moment humanises the interaction and models professional communication.

*This idea comes from* (Ma et al., 2023).

**Active listening cues**

Short acknowledgements while the student speaks signal that the system is attentive, making the voice assistant feel more human.

*This idea comes from* (Cho et al., 2022).

**Natural dialogue management**

Learners can deviate from the assistant's answers before smoothly returning to the script. Fluid back-and-forth makes practice feel like a real coaching session, not a menu of rigid commands.

*This idea comes from* (Dritsas et al., 2025).

**Clarification and confirmation**

When speech recognition is unsure, the system politely verifies, preventing misunderstandings and reinforcing accurate Scrum vocabulary.

*This idea comes from* (Davie & Hilber, 2018).

**Turn-taking cues and timing**

Audio chimes or subtle on-screen signals indicate whose turn it is to speak, preventing overlap and training students in concise, orderly stand-up behaviour.

*This idea comes from* (Cho et al., 2022).

**Visuals feedback**

While feedback is spoken, a compact on-screen checklist ticks off Scrum pillars met, or a small bar shows time spent vs. the 15-minute stand-up limit. Dual-coding boosts memory and supports users who benefit from seeing as well as hearing feedback.

*This idea comes from* (Dritsas et al., 2025).

**Customisable voice and display settings**

Users can slow the TTS rate, enable live captions, or switch to a high-contrast theme, ensuring accessibility across accents, abilities, and environments and making the tool inclusive by design.

*This idea comes from* (Millán Martínez et al., 2024).

**Progress-tracking dashboard**

A timeline of sessions, scores, and qualitative comments lets learners (and optionally instructors) monitor improvement across all four ceremonies, celebrating milestones and highlighting plateau points that need extra attention.

*This idea comes from* (Wisniewski et al., 2020).

**Performance analytics & insights**

Beyond raw scores, pattern mining surfaces trends, "You consistently skip risks during Sprint Reviews", and recommends targeted readings or follow-up drills. Data thus evolves from mere record-keeping into actionable coaching intelligence.

*This idea comes from* (Corrigan et al., 2015).

**Automatic transcript & export**

Every dialogue turn is transcribed and can be exported as a PDF. Instructors can review authentic evidence of verbal skill growth. Transcripts also support accessibility and let learners reread nuanced feedback that may be missed in real time.

*This idea comes from* (Davie & Hilber, 2018).

**Colour-graded performance history**

Session summaries appear in a colour spectrum (e.g., green → yellow → red) that instantly signals overall quality and improvement trends. The history view can be sorted by ceremony, date, or score, letting learners spot plateaus or breakthroughs at a glance while giving instructors an at-a-glance diagnostic tool.

*This idea comes from a conversation with Professor Schimkat.*

**History-aware feedback engine**

Before generating each new response, the assistant retrieves a short window of past feedback and scores, then tailors its comments to the learner's recurring strengths and weaknesses—e.g., "You've improved on specifying Done criteria since last week, but backlog prioritisation is still inconsistent." This continuity makes the feedback feel personalised and reinforces long-term reflection.

*This idea comes from* (Corrigan et al., 2015).

Collectively, these features form an adaptive, emotionally aware mentor that embodies Scrum's own cycle of inspection and adaptation while aligning with modern educational best practice. Although the brainstorming stage generated an extensive catalogue of potential features, the initial prototype deliberately concentrates on a core subset that could be delivered reliably within the project timeframe. Specifically, it implements immediate formative feedback after each learner response, detailed end-of-session summaries, explicit

turn-taking cues and timing to keep the dialogue organised, a colour-graded performance history for quick visual tracking of progress, and a history-aware feedback engine that tailors guidance to patterns from previous sessions. All remaining ideas are recorded for future iterations, subject to additional development time and user-testing cycles.

## 4.2 Requirements Analysis

To ensure accessibility and ease of use, the system includes a simple user interface that allows the student to choose among different Scrum scenarios, such as Daily Stand-up Meetings and Product Backlog Management. Once a situation is selected, the assistant initiates the corresponding interaction flow by prompting the user with predefined questions. The assistant listens to and processes the student's spoken responses, analyses their content having previous interactions in mind, then provides formative feedback and a grade based on the metrics given. This feedback includes recognition of strengths, identification of potential improvements, and suggestions for better alignment with Scrum principles. Through this interactive and adaptive dialogue, the system not only reinforces Scrum knowledge but also supports the development of communication and self-assessment skills critical to agile teamwork. The system will give the students a summary and grade regarding their performance, and it will be stored, so the student can check it at all times in a colour-graded history.

The requirements obtained are based on the literature on educational technology, conversational interfaces and agile learning practices. This ensures that it remains focused on pedagogical effectiveness, usability, and relevance to real-world agile practices.

**Functional Requirements (FR)**

- **FR-1** – The system shall present a menu of Scrum practice situations to the user at the start of the session.
- **FR-2** – The system shall load the corresponding prompt flow and assistant persona based on the selected situation.
- **FR-3** – The system shall initiate the dialogue by asking predefined questions related to the selected Scrum situation.
- **FR-4** – The system shall capture the student's response via voice, convert it to text, and normalise it for analysis.
- **FR-5** – The system shall evaluate the student's answer based on semantic similarity to key concepts and communication clarity, based on specific metrics and previous simulations.
- **FR-6** – The system shall provide feedback in natural language that includes strengths, weaknesses, and concrete suggestions for improvement.
- **FR-7 –** The system shall provide a colour-graded history feature, where the student can see the progress.

**Non-Functional Requirements (NFR)**

- **NFR-1 –** The user interface shall be minimal and intuitive, enabling users with no prior training to select scenarios and begin interactions within seconds.
- **NFR-2 –** The system shall respond to user input in an acceptable time to ensure natural conversational flow.
- **NFR-3 –** The system shall support mobile platforms.
- **NFR-4 –** The voice assistant shall use clear, neutral speech synthesis that adheres to accessibility guidelines for educational tools.
- **NFR-6 –** The system shall be modular, allowing future integration of additional Scrum ceremonies or other agile practices with minimal restructuring.
- **NFR-7 –** The system shall handle background noise and maintain voice recognition accuracy.
- **NFR-8 –** Feedback messages shall be pedagogically sound, using consistent tone and structure to support formative learning.

## 4.3 Technology Stack and Tools

To implement the voice-assisted Scrum simulation system, a multi-layered technology stack was selected. The system combines mobile development, conversational AI, and backend natural language processing tools. The selection was based on the need for real-time voice interaction, cross-platform compatibility, and seamless integration of intelligent feedback mechanisms.

**Frontend with Expo (React Native)**

The user interface is developed using React Native within the Expo framework, allowing for rapid and simple mobile deployment. Expo provides a high-level environment for building native apps using TypeScript and includes integrated support for features like audio recording and text-to-speech, which is key for voice-based interaction.

**Backend with Python and FastAPI**

The backend is built using Python and the FastAPI framework, which offers high performance, simple syntax, and native support for asynchronous operations. FastAPI handles API requests from the mobile client, receives voice-to-text transcriptions, and coordinates the logic for scenario management and feedback generation.

**Voice Communication with Ngrok and Localhost Tunnelling**

During development and testing, ngrok is used to expose the locally running Python backend to the mobile app over the internet. This allows the mobile application to communicate with the server even during local development by creating a secure HTTPS tunnel, simulating production-like conditions for real-time interactions.

**Conversational Intelligence with LangChain and Ollama**

To manage and orchestrate dynamic conversations, the system uses LangChain, a powerful framework for building applications with LLMs. LangChain enables the creation of customizable prompt templates and dialogue chains tailored to each Scrum scenario. For local inference, the system integrates Ollama, a lightweight LLM runner efficient enough to be executed on personal machines. This ensures data privacy and reduces latency compared to cloud-based alternatives.

**Memory and Semantic Search with Pinecone**

To support context-aware conversations and feedback generation, the system uses Pinecone as a vector database for storing and retrieving embeddings of user responses and ideal answers. This enables semantic similarity searches that go beyond simple keyword matching, allowing for better analysis and feedback based on meaning.

## 4.4 Real-World Cloud Deployment Environment

In a production setting, the voice-assisted Scrum simulation system must be deployed to a robust, secure, and highly available cloud infrastructure rather than relying on local tunnelling. Continuous deployment, automatic scaling, and stringent security controls together ensure the system can protect sensitive learner data and deliver smooth, real-time voice interactions at scale. A typical real-world environment would look like the following:

### 4.4.1 Containerization and Orchestration

- **Docker Containers:** Package each component into individual Docker images. Ensuring consistency across development, testing, and production.
- **Kubernetes:** Deploy containers to a managed Kubernetes cluster for automated scaling, rolling updates, and self-healing.

### 4.4.2 Managed Data Services

- **Vector Database:** Use Pinecone's managed cloud offer for storing and querying embeddings at scale without manual cluster management.
- **Object Storage:** Store raw audio recordings, TTS outputs, and session logs securely with versioning and lifecycle policies.

### 4.4.3 Model Hosting

- **Model Hosting:** Hosting language models on scalable GPU instances or serverless inference endpoints to meet fluctuating demand with low latency.

### 4.4.4 Authentication

- **Identity Provider:** Manage user sign-up, sign-in, and role-based access control for both students and instructors.

#### 4.4.5 CI/CD Pipeline & Infrastructure as Code

- **GitHub Actions:** Automate build, test, and container image publishing upon code commits.
- **Terraform:** Define infrastructure declaratively to provision clusters, databases, and networking resources reproducibly.

## 4.5 AI Model Selection

I chose the Qwen 1.7B model because it offered better reasoning ability than smaller models, without making the system too slow. In practice, on my development machine, it still ran in near real-time. By using local deployment and optimisation (batching requests, running on a GPU, etc.), I managed to keep response times in a good mark even with the larger model.

# 5. Prototype Design and Implementation

## 5.1 Use Cases and Scenarios

### 5.1.1 Daily Stand-up Meeting

1. **Precondition**: The user has opened the app and selected “Daily Stand-up.”
2. **Flow**:
    - Assistant: “What did you work on yesterday?”
    - Student responds.
    - Assistant: “What will you work on today?”
    - Assistant: “Any blockers?”
    - The assistant provides a summary and targeted feedback.

### 5.1.2 Product Backlog Management

1. **Precondition**: User selects “Backlog Management.”
2. **Flow**:
    - The assistant reads current items.
    - Prompts prioritisation/refinement questions.
    - Provides rationale and suggestions based on semantic analysis.

### 5.1.3 View Interaction History

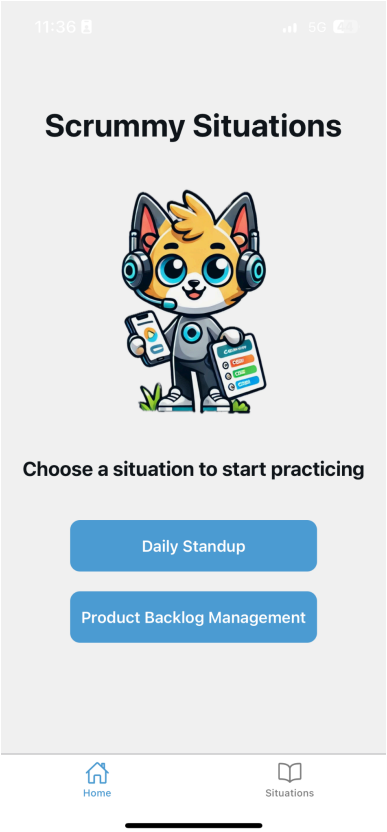
1. **Precondition**: User has completed ≥1 session.
2. **Flow**:
    - The student taps the “History” tab.
    - Assistant displays a chronological list of past sessions, each showing date, scenario type, and high-level feedback.

- The student selects a session to view the full transcript and detailed feedback, including strengths, areas for improvement, and scores.

- The assistant may highlight trends (e.g., recurring blockers) and suggest targeted practice.

## 5.2 Screens

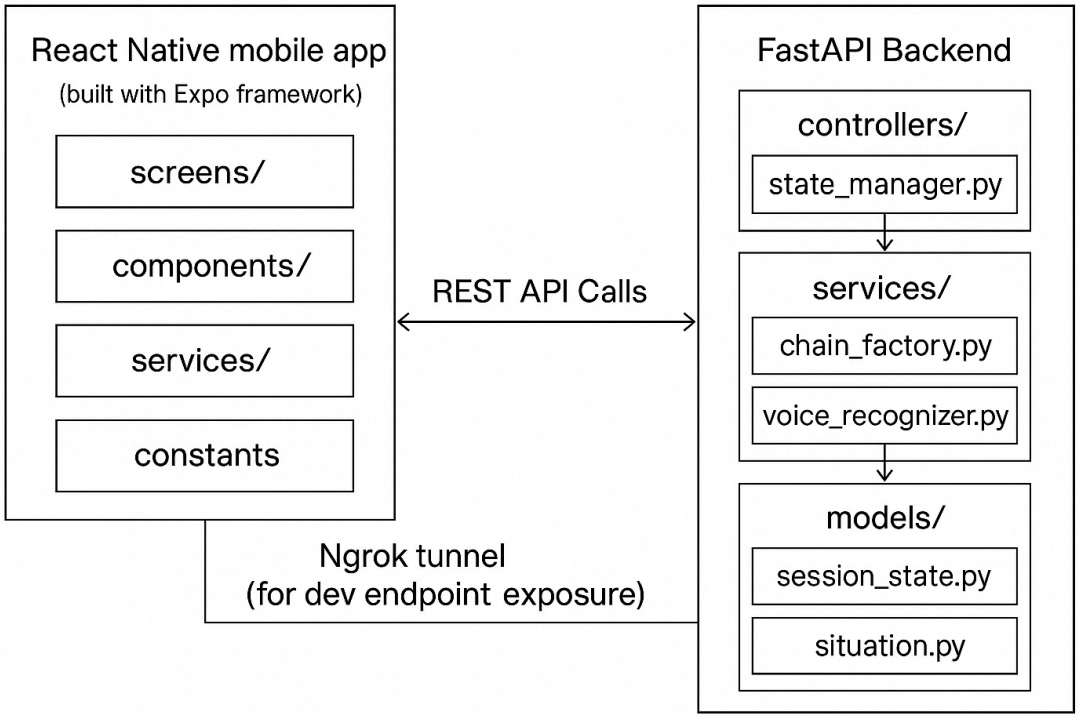
The following screen is presented to give the reader an idea of what the appearance of application looks like. In the annexe, the reader can find screenshots of the whole app.




1. The Home Screen, where the situations will be displayed, and the student will be able to choose which one to practice next.

## 5.3 System Architecture

The system architecture is divided into two main layers: the **React Native frontend** developed with Expo, and the **Python backend** built using FastAPI. These components interact via a RESTful API, with local tunnelling enabled by **ngrok** during development to simulate a production-like setup on real devices.




**Frontend (React Native with Expo)**

The mobile frontend presents an intuitive interface for students to select Scrum situations and interact with the assistant. It includes:

- Screens/: Pages such as the scenario selection, interaction tabs and history.
- Components/: Reusable UI components.
- Services/: Responsible for key functionalities like audio recording (AudioRecorder.ts), voice synthesis (TTSService.ts) and saving and retrieving the history (HistorySaver.ts) using Expo modules.

The app communicates with the backend through HTTP requests. During development, **ngrok** exposes the local backend (localhost:8000) to the device.

**TTSService**

The text-to-speech service is invoked every time the client receives a new text that must be reproduced by the phone. It uses the Expo Speech library, allowing it to regulate some voice properties. It gets the text already cleaned of other characters and reproduces it. It also uses some callback functions to manage ui feature to show the text as it sounds on the phone.

```typescript
async speakText(raw: string | object, events: SpeakEvents = {}): Promise<void> {
  let text: string;
  if (typeof raw === "string") {
    text = raw;
  } else {
    try {
      text = JSON.stringify(raw);
    } catch {
      text = String(raw);
    }
  }
 return new Promise((resolve) => {
    Speech.speak(text, {
      voice: "com.apple.ttsbundle.siri_Martha_en-GB_compact",
      language: "en-GB",
      pitch: 0.8,
      rate: 0.6,
      volume: 1.0,
      onBoundary: (evt: { charIndex: number | undefined; }) => {
        if (events.onBoundary && evt?.charIndex !== undefined) {
          events.onBoundary(evt.charIndex);
        }
      },
    });
  });
```

**AudioRecorderService**

This service is responsible for recording the audio of the student and sending it to the backend. It also uses the Expo library. This class is built using reactive to programming, helping the ui to handle the state. At first, some configurations are made, and the reactive programming is initiated. Some properties had to be changed in order to make the iPhone work with the library. It has to be set to PlaybackMode every time the recording is done, so the response can be heard properly. Permissions are checked, and we have the base functions of recording and stopping recording. Later, the request is sent via POST to the backend. The reader can see the code in the annexe.

To see the reactive programming, some lines of the code are shown, where it can be seen that some ui components depend on this service. First, we have the state handlers and their subscription to the objects.  And we set the useState components so they react to a change of the state in the objects of the service.

```TypeScript
const [isRecording, setIsRecording] = useState(false);
const [loading, setLoading] = useState(false);
const [speaking, setSpeaking] = useState(false);
const s1 = recorder.isRecording$.subscribe(setIsRecording);
const s2 = recorder.loading$.subscribe(setLoading);
const s3 = recorder.response$.subscribe(async (r) => {
```

**HistorySaverService**

To maintain a consistent history tab, a history service has been designed. Only basic read and write functionalities have been developed. For storing the entries, the local storage will be used. Additionally, the entries have been modelled into an interface to have better control of what is being saved. The code can be found in the annexe.

**Backend (Python with FastAPI)**

The backend is modular and divided into:

- Controllers/: Manage session logic and route orchestration (state_manager.py).
- Services/: Core functional logic, including:
    - chain_factory.py to generate conversational chains using LangChain.
    - voice_recognizer.py to handle audio transcription.
- Models/: Define data structures such as session_state.py and situation.py, which track user sessions and define the prompt flows, respectively.

Only 3 routes have been created for different purposes. The first one is for regular requests regarding the usage of the voice assistant. The second one is for resetting the state of the application when a simulation is over. Lastly, we have one for developing purposes, in checking the health of the application.

```Python
@app.post("/index")
async def index( audio_input: UploadFile = File(...), situation_file: str = Query(...)):
    return await next_state(situation_file, audio_input)

@app.post("/reset")
```

```
def reset():
    return reset_state()

@app.get("/health")
def health():
    return {"status": "ok",
            "message": "The server is running and healthy."}
```

The situation has also been modelled, so it is externalised from the code. All the JSON files containing the questions and context should be in a specified directory, from which the application will read and take. The code is shown in the annexe.

The session will have some key components that have also been coded, so it has a regular flow. This way, the application can know which questions have been asked so far and how many are left with the corresponding answers. The code is shown in the annexe.

Now we can see the function where the state changes. It will initialise a session if it is not one already happening. When suited, it will transcribe the audio of a file sent in the request. Then it will send the next question or the final feedback of the simulation.

```Python
async def next_state(situation_file: str, audio_input: UploadFile):
    result = None
    if not session.initialized:
        initialize_situation(situation_file)
        result = session.situation.greeting
    else:
        user_text = await transcribe_audio(audio_input)
        session.answers.append(user_text)
        print(session.answers)

    questions = session.situation.questions

    if session.question_index < len(questions):
        next_question = questions[session.question_index]["question"]
        session.question_index += 1
        result = f"{result}\n{next_question}" if result else next_question
    else:
        input_map = chain_input_formatting(session.situation)
        feedback = session.chain.invoke(input_map)
        result = f"{result}\n{feedback['answer'].strip()}" if result else feedback["answer"].strip()
        reset_state()
```

```
    return {"result": result}
```

**Conversational Intelligence Layer**

The backend integrates:

- **LangChain**: To structure multiple-step conversations and control prompt flow per Scrum situation.
- **Ollama**: For local execution of LLMs.
- **Pinecone**: To manage embeddings and perform semantic similarity searches, enabling the system to provide context-aware and meaningful feedback based on user responses.

This modular and scalable architecture ensures maintainability, supports real-time interactions, and allows future enhancements such as emotion-aware feedback or additional educational modules.

**PDF to Scrum Index**

This method has been used only once in the project, and its main goal was to populate the pinecone index so that later on, the system can have a better understanding of what the responses should look like. It slices the PDF and uploads it to the Pinecone API.

```
Python
embeddings = OllamaEmbeddings(model="nomic-embed-text")
index_name = os.getenv("PINECONE_ENVIRONMENT")
vectorstore = PineconeVectorStore(index_name=index_name, embedding=embeddings)


pdf_path = ""
namespace = ""
chunk_size = 150
chunk_overlap = 10

loader = PyMuPDFLoader(pdf_path)
docs = loader.load()
splitter = RecursiveCharacterTextSplitter(
    chunk_size=chunk_size,
    chunk_overlap=chunk_overlap
```

```
)
chunks = splitter.split_documents(docs)


vectorstore_from_docs = PineconeVectorStore.from_documents(
   chunks,
   index_name=index_name,
   embedding=embeddings
)
```

**ChainFactoryService**

This function is a small factory that builds a retrieval-augmented chat pipeline: it first creates an embedding model and a local LLM (qwen3:1.7b) via Ollama using the .env file, then connects to a Pinecone vector index and wraps that index as a retriever that will return the top two semantically similar documents for any query. Next it assembles a PromptTemplate from the template string and the list of expected input keys you pass in, and finally it stitches everything together with LangChain's ConversationalRetrievalChain.from_llm, producing an object you can call with a dictionary, the chain will fetch the relevant docs, insert them into the prompt, run the LLM to craft a context-aware answer, and return the response in one shot.

Python

```python
def get_chain(template: str, input_variables: list[str]):

    embeddings = OllamaEmbeddings(model="nomic-embed-text")
    model_name = os.getenv("MODEL_NAME")
    llm = OllamaLLM(model=model_name)

    index_name = os.getenv("PINECONE_ENVIRONMENT")
    vectorstore = PineconeVectorStore(index_name=index_name,
embedding=embeddings)
    retriever = vectorstore.as_retriever(search_kwargs={"k": 6})

    prompt = PromptTemplate(
        input_variables=input_variables,
        template=template
    )

    return ConversationalRetrievalChain.from_llm(
```

```
        llm=llm,
        retriever=retriever,
        combine_docs_chain_kwargs={"prompt": prompt}
    )
```

**VoiceRecognitionService**

This method is invoked by the application every time it gets a response from the user. It gets an audio file in the data of the request. Such an audio file will be stored and transcribed using audio recognition from Google.

```
Python
async def transcribe_audio(file: UploadFile) -> str:
    suffix = Path(file.filename).suffix.lower() or ".wav"
    with tempfile.NamedTemporaryFile(delete=False, suffix=suffix) as tmp_input:
        contents = await file.read()
        tmp_input.write(contents)
        tmp_input.flush()
        input_path = tmp_input.name

    try:
        wav_fd, wav_path = tempfile.mkstemp(suffix=".wav")
        os.close(wav_fd)
        AudioSegment.from_file(input_path).export(wav_path, format="wav")

        with sr.AudioFile(wav_path) as source:
            audio_data = recognizer.record(source)

        try:
            text = recognizer.recognize_google(audio_data, language="en-US")
            print(text)
```

**Deployment**

The deployment script automates the whole stack in four moves. First, the ensure_ollama_running function checks whether the local Ollama language-model server is alive; if it is not, it spins up Ollama serve in the background and waits five seconds to confirm the PID. Second, get_ngrok_url either retrieves the current HTTPS endpoint from the ngrok API (port 4040) or, if none exists, launches a fresh tunnel to expose port 8000 and then captures its public URL. This guarantees that every run gets a valid, publicly reachable address for the FastAPI backend. Third, the script force-kills any leftover Uvicorn processes, activates the project's Python virtual environment, and boots the backend

(backend.src.main:app) on 0.0.0.0:8000. Fourth, it writes the newly minted ngrok URL to frontend/.env as EXPO_PUBLIC_API_URL, so the React-Native (Expo) frontend always talks to the correct host, then launches expo start. In short, the script makes sure the model is running, provisions a fresh tunnel for the backend, injects that endpoint into the environment file, and finally brings up the mobile frontend, all in a single command. The script can be found in the annexe.

## 5.4 Situations Structure

Each Scrum ceremony is still described by a single JSON file in /resources, but the schema now supports richer rubrics and a single student_responses queue.

**greeting**
A short phrase the assistant speaks at the start of the session to set the tone and identify the ceremony.

**questions**
An array of objects that the assistant will ask in order. Each object contains:

- Question – the sentence to speak aloud.
- Key – a unique identifier (e.g., priority, changes, clarity) used later to reference that answer.

**responses**
A model answer written with the same keys. It shows the LLM what a perfect (10 / 10) response looks like.

**context**
A concise description of the role, audience, and tone so the LLM knows how to frame its feedback.

**metrics**
A Markdown block listing the scoring rubric (five metrics worth 0–2 points each, plus optional penalties). Because the rubric lives in the JSON file, you can revise evaluation criteria without touching the code.

**prompt_template**
The master prompt that will be sent to the LLM. It interpolates:

- {context} – the situation description.
- {source_documents} – optional retrieved references.
- {responses} – the perfect 10/10 answer.
- {metrics} – the rubric to apply.
- {student_responses} – all user answers combined into one string.

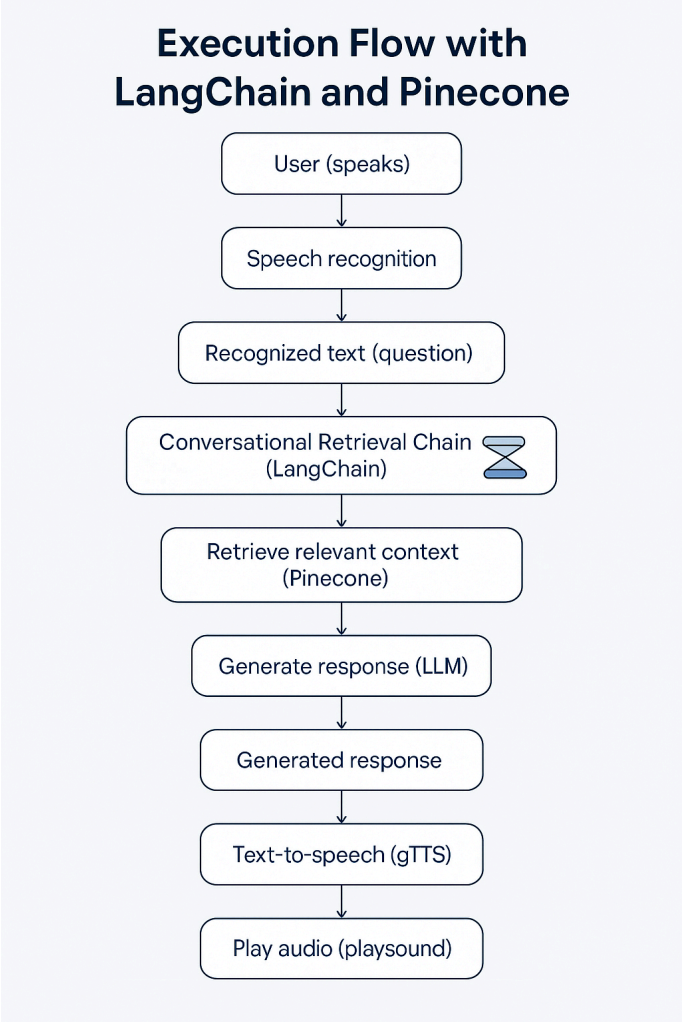
- {history} – the last n pieces of saved feedback, for personalisation.

It finishes with strict instructions to return only four sections: **Strengths**, **Areas for improvement**, **Summary**, and **Grade**.

**input_variables**
A list of every placeholder that must be filled when assembling the prompt. In the new design this typically includes: question, chat_history, context, responses, source_documents, metrics, student_responses and history.

## 5.5 Conceptual System Pipeline

This diagram gives a simplified picture of what happens to a spoken input. In reality, the mobile front end is doing extra work, recording audio, showing UI cues, and sending data over the network, but the core "model loop" in the backend stays the same: once the phone turns speech into text, that text and a few recent turns of the conversation are fed into a LangChain pipeline. The chain first queries Pinecone for relevant snippets of reference material, then assembles a prompt that blends those snippets with several user-specific inputs (the recognised question, the learner's role, and any prior feedback for this session) so the language model has full context. A lightweight LLM reads that composite prompt and generates a tailored answer. The backend sends the answer text back to the app, the app converts it to speech, and the user hears the response. So, while the production wiring has more moving parts, the heart of the system is still this two-step model process: retrieve the right context and, with a rich prompt, generate a clear personalised reply.



# 6. Evaluation

Because the prototype was finalised only shortly before the submission deadline, a classroom‑scale study with real student teams could not be scheduled. Instead, I performed a series of structured self-tests that aimed to answer two practical questions:

1. **Does the assistant already provide formative value, like usable, specific feedback, when a single learner rehearses the four Scrum ceremonies alone?**
2. **Does the student get a personalised experience while learning with the assistant?**

The section documents the setup, the findings, and the lessons learned.

## 6.1 Evaluation Objectives

- **E-1. End-to-end stability:** Verify that every simulated Scrum ceremony, from prompt selection to feedback delivery, runs without crashes or noticeable delays.
- **E-2. Perceived feedback quality:** Determine whether a single learner finds the assistant's guidance and commentary genuinely helpful during solo practice.
- **E-3. Speech-recognition robustness:** Explore how accurately the system transcribes spoken answers across varying accents and background-noise levels.
- **E-4. Rubric flexibility:** Check whether the scoring model treats conceptually correct but differently worded responses as valid, instead of penalising indirect phrasing.
- **E-5. Personalised Experience:** Highlight results of specific features that grant the student a more individualised practice.

(These objectives have been selected by me with the help of ChatGPT.)

## 6.2 Discussion

- **A solid tool for ceremony rehearsal (Objective E-1 & E-2).**

  In every run, the assistant asked the full question set, recorded answers, retrieved guide excerpts, and returned structured feedback without crashing. Latency felt natural on a mid-range phone, confirming that the local 1.7 B-parameter model plus Pinecone retrieval is fast enough for solo practice.

- **Speech recognition is the weakest link (E-3).**

  The Google Speech API handles neutral or light-accented English well, but WER more than doubles with stronger accents or background chatter. Misrecognitions propagate into the evaluation stage, lowering scores unfairly.

- **Feedback is accurate when the answer matches the prompt, less so when phrasing diverges (E-4).**

Because the prompt template feeds both the *ideal* response and the student's answers into the chain, the model can highlight missing elements with surprising precision. However, the strict slot-based rubric expects explicit mention of, for example, a "blocker" during Stand-up. When the same issue is implied rather than named, the grade drops.

- **The applications provide a clear personalised approach (E-5).**

  The model gives feedback based on past interactions, so the student can know what to improve with more context. It also allows the student to review past interactions with a colour grade history.

## 6.3 Limitations to the Evaluation

1. **Evaluator bias.** All sessions were performed by the author, who is intimately familiar with the rubric. Real drills may phrase answers less predictably and trigger different failure modes.
2. **Small sample.** The prototype was not tested with a real heavy workload
3. **Single-device test.** Performance may differ on older phones.
4. **No effect data.** The prototype logs latency and accuracy but not frustration or cognitive load; thus, user experience is inferred only from diary notes.

## 6.4 Implications for Future User Study

- **Broaden accent coverage.** Integrate open-source ASR (e.g. Whisper large-v3) fine-tuned on non-native English or allow manual correction before grading.
- **Add semantic leniency.** Replace binary keyword checks with embedding-based similarity on each rubric item to reward conceptually correct paraphrases.
- **Pilot with 8-10 students.** A short in-class pilot will reveal group dynamics, turn-taking issues, and motivational effects that self-tests cannot capture.
- **Instrument effect.** Simple Likert emotion check-ins or voice-tone sentiment analysis would quantify engagement.

## 6.5 Summary

Although only self-tested, the prototype already functions as a hands-on tutor for Scrum ceremonies: it guides the learner through realistic prompts, mines the Scrum Guide for reference answers, and delivers formative feedback in under two seconds. The main blockers en route to classroom deployment are speech-recognition robustness and rubric flexibility, both solvable with larger ASR models and embedding-level grading. The present evaluation, therefore, positions the assistant as a promising first iteration that justifies a forthcoming multi-student study.

# 7. Conclusion

This thesis set out to discover whether a voice assistant could deliver truly personalised learning of Scrum and, by extension, demonstrate the broader potential of conversational agents in higher-education settings. The journey, from historical analysis of voice technology, through learning-science theory, to prototype design and self-evaluation, produced three clear insights that jointly answer the research questions.

First, a dynamic learner model is indispensable. By updating after every utterance, tracking response latency, semantic accuracy and past errors, the system adapted pace, question depth and feedback tone in real time, offering far richer personalisation than static "learning-style" inventories can provide. Second, voice interaction adds unique pedagogical value: hands-free exchanges reduce interface friction, continuous speech data fuel adaptivity, and the conversational format naturally mirrors Scrum's inspect-and-adapt rhythm. Nevertheless, we observed limits: cloud-based ASR faltered with strong accents or noisy rooms, and rubric rules that relied on explicit keywords sometimes underrated conceptually correct paraphrases. These constraints highlight that technology must be paired with careful instructional design, faded scaffolds, accent-tuned recognition, and semantic grading, to avoid over-guidance or unfair assessment. The significance of these findings is twofold. Pedagogically, they suggest that voice assistants can shoulder much of the formative-feedback load, freeing instructors for higher-order coaching while still giving students immediate, actionable guidance. Methodologically, the study shows that agile principles can be both content and method: the assistant iterates, inspects and adapts just as students are expected to do, illustrating a virtuous alignment between learning medium and learning objective. That promise is tempered by limitations. Our evaluation was confined to structured self-tests rather than live, multi-user classroom deployments; affective states such as frustration or boredom were only inferred; and accent bias in speech recognition remains unresolved. Addressing these gaps requires broader trials, richer affect telemetry and more robust ASR models.

Future research should therefore pursue three avenues: deploy the system in real classrooms to measure long-term retention and team dynamics; integrate accent-aware Whisper-grade recognition and embedding-based semantic grading to boost fairness and flexibility; and incorporate emotion detection so pacing and encouragement respond to the learner's actual state. Extending the blueprint beyond Scrum, to laboratory safety, language pragmatics or clinical hand-offs, will test its generality and further illuminate how voice, feedback theory and adaptive modelling can converge to democratise personalised learning.

In sum, the work demonstrates that voice assistants, designed around dynamic learner models and agile feedback loops, can act as scalable, low-pressure mentors. Scrum provided the proving ground; the pedagogical principles uncovered here reach far wider.

# 8. References


Amazon Science Staff. (2023, July 6). *Alexa Skills Inventor boosts AI education, drives student engagement*. Amazon Science. Retrieved June, 2025, from https://www.amazon.science/news-and-features/alexa-skills-inventor-boosts-ai-education-drives-student-engagement

Amazon staff. (2023, December 1). *How people with disabilities can use Alexa to help them lead more independent lives*. About Amazon. Retrieved June, 2025, from https://www.aboutamazon.co.uk/news/devices/alexa-accessibility-features

Bickham, D.S., Schwamm, S., Izenman, E.R., Yue, Z., Carter, M., Powell, N., Tiches, K., & Rich, M. (2024, February). *Use of voice assistants & generative AI by children and families*. Digital Wellness Lab, Boston Children's Hospital. Retrieved April, 2025, from https://digitalwellnesslab.org/wp-content/uploads/Digital-Wellness-Lab-Pulse-Report-Feb-2024.pdf

Casey, T. (2019, December 19). *AskMyClass turns Alexa into a next-gen teacher's assistant*. Forbes. Retrieved July, 2025, from https://www.forbes.com/sites/theodorecasey/2019/12/19/askmyclass-turns-alexa-into-a-next-gen-teachers-assistant/

Chen, S., Fang, Y., Shi, G., Sabatini, J., Greenberg, D., Frijters, J., & Graesser, A. C. (2021). Automated disengagement tracking within an intelligent tutoring system. *Frontiers in Artificial Intelligence*, *3*. https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2020.595627/full

Cho, E., N., Sundar, S.S., & Abdullah, S. (2022). Alexa as an active listener: How back-channeling can elicit self-disclosure and promote user experience. *Proceedings of the ACM on Human-Computer Interaction*, *6*(CSCW 2). https://dl.acm.org/doi/10.1145/3555164

Choksi, K., Chen, H., Joshi, K., Jade, S., Nirjon, S., & Lin, S. (2024, June 13). *SensEmo: Enabling affective learning through real-time emotion recognition with smartwatches*. arXiv. Retrieved June, 2025, from https://arxiv.org/abs/2407.09911

Coffield, F. (2004). *Learning styles and pedagogy in post-16 learning: A systematic and critical review*. Learning & Skills Research Centre. https://www.voced.edu.au/content/ngv:13692

Corrigan, S., Barkley, T., & Pardos, Z. A. (2015). Dynamic approaches to modeling student affect and its changing role in learning and performance. *Lecture Notes in Computer Science*, *9146*, 92-103. https://www.researchgate.net/publication/286523134_Dynamic_Approaches_to_Modeling_Student_Affect_and_its_Changing_Role_in_Learning_and_Performance

CSols Inc. (2024, September 19). *Comparing voice-activated lab assistants*. Retrieved June, 2025, from https://www.csolsinc.com/resources/comparing-voice-activated-lab-assistants

Davie, N., & Hilber, T. (2018). Opportunities and challenges of using Amazon Echo in education. *14th International Conference Mobile Learning 2018*. https://files.eric.ed.gov/fulltext/ED590385.pdf

Davis, W. (2025, February 26). *Amazon announces AI-powered Alexa Plus*. The Verge. Retrieved April, 2025, from https://www.theverge.com/news/619755/amazon-alexa-ai-upgrade-artificial-intelligence-smart-assistant

Dizon, G., Tang, D., & Yamamoto, Y. (2022). A case study of using Alexa for out-of-class, self-directed Japanese language learning. *Computers & Education: Artificial Intelligence*, *3*. https://www.researchgate.net/publication/361845759_A_case_study_of_using_Alexa_for_out-of-class_self-directed_Japanese_language_learning

Dritsas, E., Trigka, M., Troussas, C., & Mylonas, P. (2025, January 14). Multimodal interaction, interfaces, and communication: A survey. *Multimodal Technologies and Interaction*, *9*(1). https://www.mdpi.com/2414-4088/9/1/6

Felder, R. M., & Silverman, L. K. (1988). Learning and teaching styles in engineering education. *Engineering Education*, *78*(7), 674-681. https://www.researchgate.net/publication/257431200_Learning_and_Teaching_Styles_in_Engineering_Education

Fielding, M. (2012, December). Education as if people matter: John Macmurray, community and the struggle for democracy. *Oxford Review of Education*, *38*(6), 675-692. https://www.jstor.org/stable/42001785

Halkiopoulos, C., & Gkintoni, E. (2024). Leveraging AI in e-learning: Personalized learning and adaptive assessment through cognitive neuropsychology—A systematic analysis. *Electronics*, *13*(18). https://www.mdpi.com/2079-9292/13/18/3762

Hunt, T., & Hudson, M. (2024). "Alexa, play today's Flash Briefing": Utilizing conversational agents in educator preparation. *PDS Partners: Bridging Research to Practice*. https://www.emerald.com/pdsp/article/19/2/140/1215155/Alexa-play-today-s-Flash-Briefing-utilizing

Kolb, D. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.

https://www.researchgate.net/publication/235701029_Experiential_Learning_Experience_As_The_Source_Of_Learning_And_Development

Kostaciński, M. (2022, June 29). *Voice user interfaces—the future of UI*. Fabrity. Retrieved April, 2025, from https://fabrity.com/blog/voice-user-interfaces-the-future-of-ui/

Kundu, S. S., Sarkar, D., Jana, P., & Kole, D. K. (2021). Personalization in education using recommendation system: An overview. In *Computational intelligence in digital pedagogy* (pp. 85-111). Deyasi, A.; Mukherjee, S.; Mukherjee, A.; Bhattacharjee, A. K.; Mondal, A. https://link.springer.com/chapter/10.1007/978-981-15-8744-3_5

Lapowsky, I. (2018, May 9). *Congress, privacy groups question Amazon's Echo Dot for Kids*. Wired. Retrieved June, 2025, from https://www.wired.com/story/congress-privacy-groups-question-amazons-echo-dot-for-kids/

Ma, Y., Zhang, Y., Bachinski, M., & Fjeld, M. (2023, November). Emotion-aware voice assistants: Design, implementation, and preliminary insights. *Association for Computing Machinery*. https://dl.acm.org/doi/pdf/10.1145/3629606.3629665

Mierow, M. (2019, September 25). *The Alexa Education Skill API (Preview) allows you to easily create voice interfaces for education technology applications*. Amazon Developer Blog. Retrieved June, 2025, from https://developer.amazon.com/en-US/blogs/alexa/post/92af8bc8-d076-4df4-9121-d2e968fea00a/the-alexa-education-skill-api-preview-allows-you-to-easily-create-voice-interfaces-for-education-technology-application

Millán Martínez, M.W., Lagunes Barradas, V., & Rojano-Cáceres, J.R. (2024, February). *Accessible applications for people with visual disabilities through voice assistants: A systematic review of the literature*. ResearchGate. Retrieved May, 2025, from https://www.researchgate.net/publication/378461383_Accessible_applications_for_pe

ople_with_visual_disabilities_through_voice_assistants_a_systematic_review_of_the_literature

Mills, M., Monk, S., Keddie, A., Renshaw, P., Christie, P., Geelan, D., & Gowlett, C. (2014). Differentiated learning: From policy to classroom. *Oxford Review of Education*, *40*(3), 331-348. https://www.tandfonline.com/doi/full/10.1080/03054985.2014.911725

OpenAI. (2025). *ChatGPT o3* [Large Language Model]. https://chat.openai.com/

Thi-Nhu Ngo, T., Hao-Jan Chen, H., & Kuo-Wei Lai, K. (2024). The effectiveness of automatic speech recognition in ESL/EFL pronunciation: A meta-analysis. *ReCALL*, *36*(1), 4-21. https://www.cambridge.org/core/journals/recall/article/effectiveness-of-automatic-speech-recognition-in-eslefl-pronunciation-a-metaanalysis/A915444CF252B61D14961D2FE733822D

Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R.A. (2008). Learning styles: Concepts and evidence. *Psychological Science in the Public Interest*, *9*(3), 105-119. https://www.researchgate.net/publication/233600402_Learning_Styles_Concepts_and_Evidence

Pierce, D. (2024, 10 30). *The Alexa Skills revolution that wasn't*. The Verge. Retrieved June, 2025, from https://www.theverge.com/24283253/amazon-alexa-skills-app-store-10th-anniversary

Quantum Market Research. (2025, May 1). *In-car voice assistant market forecast: Size, trends, and opportunity in 2033*. LinkedIn. Retrieved April, 2025, from https://www.linkedin.com/pulse/in-car-voice-assistant-market-forecast-size-trends-s0t3e/

Raslan, G. (2024). The impact of the zone of proximal development concept (scaffolding) on the students' problem-solving skills and learning outcomes. In *BUiD Doctoral Research Conference 2023: Multidisciplinary studies* (pp. 59-66). Al Marri, K.; Mir, F. A.; David, S. A.; Al-Emran, M. https://link.springer.com/chapter/10.1007/978-3-031-56121-4_6

Schön, E.-., Neumann, M., Hofmann-Stölting, C., Baeza-Yates, R., & Rauschenberger, M. (2023, July 27). How are AI assistants changing higher education? *Frontiers in Computer Science*. https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2023.1208550/full

Schwaber, K., & Sutherland, J. (2020, November). *The Scrum Guide: The definitive guide to Scrum—The rules of the game*. Scrum Guides. Retrieved June, 2025, from scrumguides

Silva, F. N. d., Coelho, R. C., Araújo, W. R. V. C. d., & Godoy, C. M. G. d. (2024). Exploring the applications and impacts of voice assistants: A comprehensive review focused on education and academic services. *RENOTE – Revista Novas Tecnologias na Educação*, *22*(1), 320-329. https://seer.ufrgs.br/index.php/renote/article/view/141558

Straub, E. O. (2025, July 11). *Roundup on research: The myth of "learning styles"*. Online Teaching. Retrieved July, 2025, from https://onlineteaching.umich.edu/articles/the-myth-of-learning-styles/

Tetzlaff, L., Schmiedek, F., & Brod, G. (2021). Developing personalized education: A dynamic framework. *Educational Psychology Review*, *33*, 863-882. https://link.springer.com/article/10.1007/s10648-020-09570-w

van der Kleij, F. (2013). *Computer-based feedback in formative assessment*. Enschede, The Netherlands.

https://www.researchgate.net/publication/304628867_Computer-based_feedback_in_formative_assessment

Wikipedia contributors. (2021, October 4). *Cognitive style*. Wikipedia. Retrieved May, 2025, from https://en.wikipedia.org/wiki/Cognitive_style

Wikipedia contributors. (2024, August 25). *Timeline of speech and voice recognition*. Wikipedia. Retrieved May, 2025, from https://en.wikipedia.org/wiki/Timeline_of_speech_and_voice_recognition

Wikipedia contributors. (2024, September 11). *IBM ViaVoice*. Wikipedia. Retrieved May, 2025, from https://en.wikipedia.org/wiki/IBM_ViaVoice

Wikipedia contributors. (2025, January 10). *Voice computing*. Wikipedia. Retrieved May, 2025, from https://en.wikipedia.org/wiki/Voice_computing

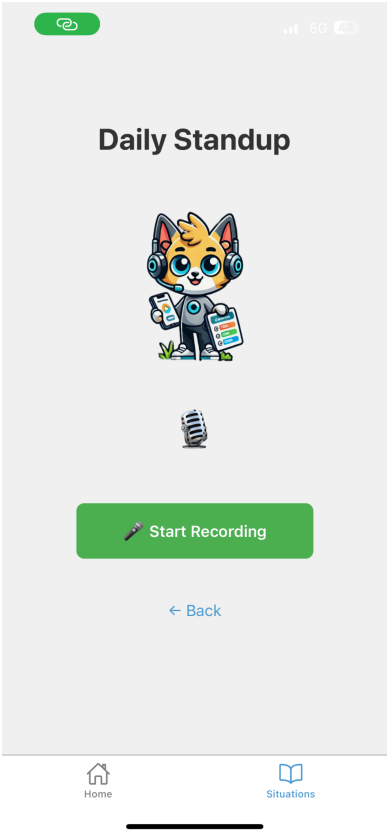
Wisniewski, B., Zierer, K., & Hattie, J. (2020). The power of feedback revisited: A meta-analysis of educational feedback research. *Frontiers in Psychology*, *10*. https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2019.03087/full

Yu, H., Peng, H., & Lowie, W. M. (2022, July 21). Dynamics of Language Learning Motivation and Emotions: A Parallel-Process Growth Mixture Modeling Approach. *Frontiers in Psychology*, *13*. https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2022.899400/full

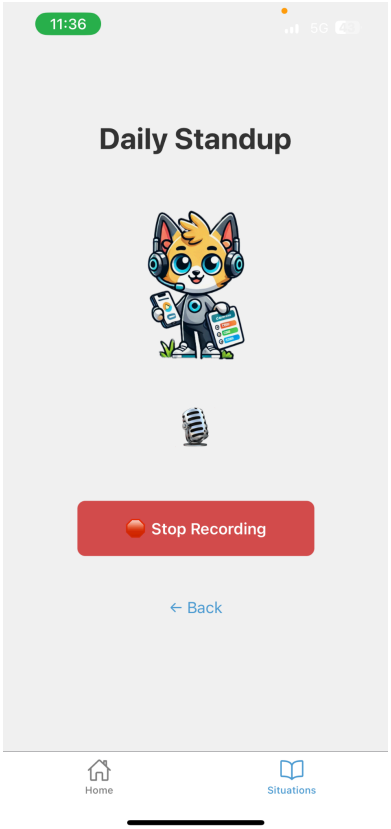
# 9. Annexe

## Screens

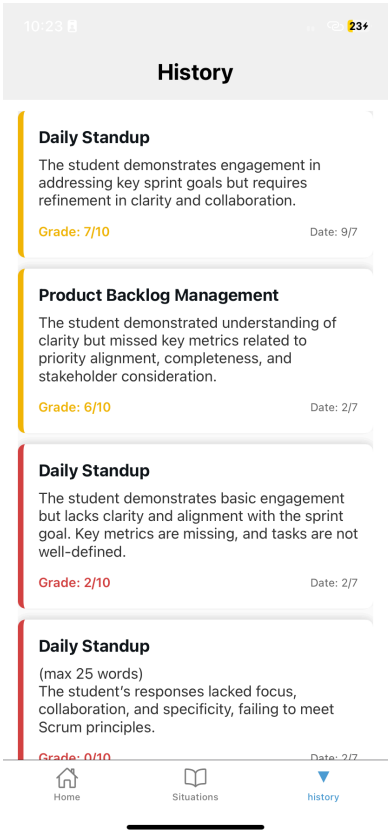
Situation Screen where the simulation takes place. The student will start speaking by pressing the green button.

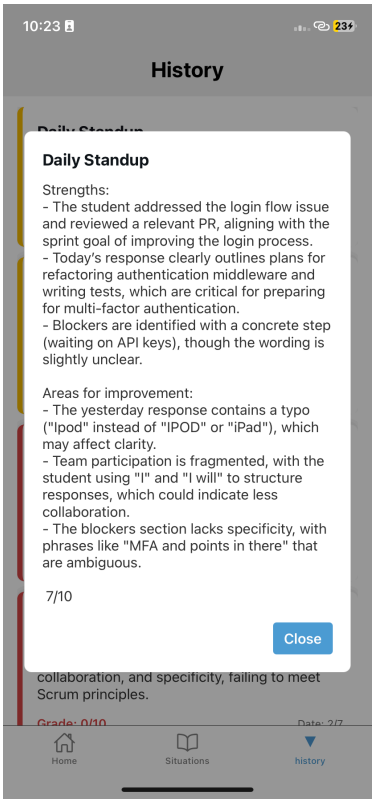


Recording Screen where the student will be speaking to the assistant.




The history screen where the student could see the recent situations, practices with a grade, and a summary of the performance.






Screen to include the full feedback received from the voice assistant so the student can access it at any time and check full progress.

## AudioRecorderService

```JavaScript
export class ReactiveAudioRecorder {
  private recording: Audio.Recording | null = null;
  private serverUrl: string;
  private situationFile: string;

  public isRecording$ = new BehaviorSubject<boolean>(false);
  public loading$ = new BehaviorSubject<boolean>(false);
  public response$ = new Subject<any>();

  constructor(serverUrl: string, situationFile: string) {
    this.serverUrl = serverUrl;
    this.situationFile = situationFile;
  }

  private async setRecordingMode() {
    await Audio.setAudioModeAsync({
      allowsRecordingIOS: true,
      playsInSilentModeIOS: true,
      staysActiveInBackground: false,
      shouldDuckAndroid: false,
      playThroughEarpieceAndroid: false,
    });
  }

  private async setPlaybackMode() {
    await Audio.setAudioModeAsync({
      allowsRecordingIOS: false,
    });
  }
  async start() {
    try {
      this.loading$.next(true);

      const permission = await Audio.requestPermissionsAsync();
      if (permission.status !== "granted") {
        throw new Error("Permiso para grabar no concedido");
      }

      await this.setRecordingMode();

      this.recording = new Audio.Recording();
      await this.recording.prepareToRecordAsync(
        Audio.RecordingOptionsPresets.HIGH_QUALITY
      );
      await this.recording.startAsync();

      this.isRecording$.next(true);
    } catch (err) {
      console.error("Error al iniciar grabación:", err);
    } finally {
```

```
    this.loading$.next(false);
  }
}

async stopAndUpload() {
  try {
    this.loading$.next(true);
    this.isRecording$.next(false);

    await this.setPlaybackMode();
    await this.recording?.stopAndUnloadAsync();

    const uri = this.recording?.getURI();
    this.recording = null;

    if (!uri) return;
    const result = await this.sendToServer(uri);
    this.response$.next(result);
    console.log("Respuesta del servidor:", result);
  } catch (err) {
    console.error("Error al detener o subir audio:", err);
  } finally {
    this.loading$.next(false);
  }
}

private async sendToServer(uri: string) {
  const urlWithParam = `${this.serverUrl}/index?situation_file=${encodeURIComponent(
    this.situationFile
  )}`;

  const formData = new FormData();
  formData.append("audio_input", {
    uri,
    name: "audio.wav",
    type: "audio/wav",
  } as any);

  const response = await fetch(urlWithParam, {
    method: "POST",
    body: formData,
  });

  return await response.json();
}
```

## HistorySaver

JavaScript

```
export interface HistoryEntry {
  situation: string;
  feedback: string;
  grade: number | string;
  timestamp: string;
}
static async saveEntry(entry: HistoryEntry): Promise<void> {
    try {
      const raw = await AsyncStorage.getItem(HistorySaver.STORAGE_KEY);
      const hist: HistoryEntry[] = raw ? JSON.parse(raw) : [];
      hist.push(entry);
      await AsyncStorage.setItem(
        HistorySaver.STORAGE_KEY,
        JSON.stringify(hist)
      );
    } catch (e) {
      console.error(" Error saving history entry", e);
    }
  }
  static async getEntries(): Promise<HistoryEntry[]> {
    try {
      const raw = await AsyncStorage.getItem(HistorySaver.STORAGE_KEY);
      return raw ? (JSON.parse(raw) as HistoryEntry[]) : [];
    } catch (e) {
      console.error(" Error retrieving history entries", e);
      return [];
```

```
  }

}
```

## Situation Model

```python
class Situation(ABC):
    def __init__(self, file_name: str):
        base_dir = os.path.dirname(os.path.abspath(__file__))
        resource_path = os.path.join(base_dir, "../../resources/situations", file_name)
        resource_path = os.path.normpath(resource_path)
        with open(resource_path, 'r') as f:
            data = json.load(f)

        self._greeting = data["greeting"]
        self._questions = data["questions"]
        self._context = data["context"]
        self._prompt_template = data["prompt_template"]
        self._input_variables = data["input_variables"]
```

## Session Model

```python
class SessionState:
    def __init__(self):
        self.initialized = False
        self.situation = None
        self.chain = None
        self.question_index = 0
        self.answers = []
```

## Deployment Script

```shell
#!/bin/bash
ensure_ollama_running() {
  if ! pgrep -x "ollama" > /dev/null; then
    echo "🔄 Running Ollama..."
    nohup ollama serve > /dev/null 2>&1 &
    sleep 5
```

```
    if pgrep -x "ollama" > /dev/null; then
      echo " Ollama is running."
     else
    echo " Ollama is already running."
  fi
}

get_ngrok_url() {
  # Get ngrok public HTTPS URL
  NGROK_URL=$(curl --silent http://127.0.0.1:4040/api/tunnels | grep -o
'"public_url":"https://[^"]*"' | head -n 1 | sed
's/.*"public_url":"\(https[^"]*\)".*/\1/')

  if [ -z "$NGROK_URL" ]; then
    pkill ngrok 2>/dev/null
    nohup ngrok http 8000 > /dev/null 2>&1 &
    sleep 5
    NGROK_URL=$(curl --silent http://127.0.0.1:4040/api/tunnels | grep -o
'"public_url":"https://[^"]*"' | head -n 1 | sed
's/.*"public_url":"\(https[^"]*\)".*/\1/')
  fi

  echo "$NGROK_URL"
}

ensure_ollama_running
# Get or start ngrok tunnel
NGROK_URL=$(get_ngrok_url)

if [ -z "$NGROK_URL" ]; then
  echo "Failed to get ngrok URL. Exiting."
  exit 1
fi

echo "Ngrok tunnel is running at: $NGROK_URL"

# Kill any existing backend (uvicorn) process
echo " Restarting backend..."
pkill -f "uvicorn backend.src.main:app" 2>/dev/null

# Activate virtualenv and start backend
echo " Starting backend..."
source backend/.venv/bin/activate
uvicorn backend.src.main:app --host 0.0.0.0 --port 8000 &

# Write .env file with ngrok URL for frontend
echo " Writing .env file for frontend..."
echo "EXPO_PUBLIC_API_URL=$NGROK_URL" > frontend/.env
```

```
# Start frontend
echo "🌐 Starting frontend..."
cd frontend
npx expo start
```

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Abschlussarbeit selbstständig und ohne fremde Hilfe verfasst sowie keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle wörtlich oder sinngemäß aus anderen Quellen übernommenen Stellen sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht.

Konstanz, 25.07.2025
Ort, Datum Pablo Larraz 
(Eigenhändige Unterschrift, Vor- und Nachname)