**Voicera**

**Towards Personalized Learning - Every Student Matters - Every Voice Counts**

*"In every classroom, there’s a quiet student with questions they’re too shy to ask… and another who’s still trying to understand yesterday’s lesson. Teachers want to reach them all — but time and language barriers get in the way.*

*That’s where* ***Voicera*** *steps in. Your AI Teacher Assistant that listens, translates, explains, and answers — anytime, in any language, in a voice that feels familiar. It learns each student’s pace, gives them the space to grow, and frees teachers from repeating the same answers again and again.*

*Voicera doesn’t replace the teacher — it amplifies their reach, ensuring no learner is left behind.*

**three core features**

1. ***AI Voice Teacher Assistant*** *– Reads, explains, and customizes lessons in real-time for each student.*
2. ***Personalized Learning Paths*** *– AI dynamically adjusts difficulty and pace for slow or advanced learners.*
3. ***Voice Engagement Analytics*** *– Real-time reports on student understanding, participation, and engagement for teachers/admin.*

1. Concept Storyboard – Enterprise Voicera Flow

**Step 1: Institution Registration**

* Admin of a school/college/university signs up.
* Creates an **institution profile** (name, logo, type, location, accreditation details).

**Step 2: Program & Course Setup**

* Admin adds **programs** (e.g., BSc Computer Science, MBA, BA English).
* Within each program, they add **courses** (e.g., "Data Structures," "Marketing 101").

**Step 3: Teacher & Student Onboarding**

* Teachers are assigned to courses.
* Students enroll in specific courses (via code, email invite, or bulk upload).

**Step 4: Repository Building (Institutional Digital Assets)**

* Every course gets an **AI-powered repository**.
* Repository stores lecture notes, assignments, references, and AI-generated summaries.
* AI automatically tags and organizes files for **program-wise and course-wise navigation**.

**Step 5: AI Web Scraping & Research Assistance**

* Students can ask **course-related queries**.
* AI scrapes **reliable internet sources** (e.g., academic journals, educational websites) and organizes findings under the correct course folder.
* AI provides **summarized + referenced** answers for easier learning.

**Step 6: Continuous Growth of Institutional Knowledge**

* Each year’s content adds to the **permanent, searchable knowledge base** of the institution.
* Teachers can update courses, reuse materials, and train AI on institutional-specific content.

**Valued Features**

1. **Institution, Program, Course Registration**
2. **Teacher & Student Enrollment**
3. **AI-Powered Course Repository** (document uploads + AI tagging)
4. **Basic Web Scraping Q&A** (retrieves info from internet, organizes into course folders)
5. **Search & Retrieval** within repository (keyword + tag-based search)

**Core Features**

* Institution registration & authentication
* Program & course creation
* Teacher & student enrollment
* Document uploads + auto-tagging (AI classification by program/course/topic)
* AI-driven Q&A with internet research (scraping + summarizing)
* Basic dashboard for admins, teachers, students
* Speech-to-text lecture capture
* AI Voice Chatbot
* Learning Leads generation
* Agentic AI to drive learning
* Multiple language voice chat facility
* AI-powered auto quizzes from lecture content
* Predictive analytics for student performance
* Integration with video conferencing tools (Zoom, MS Teams)
* Mobile app version

**Technical Part for AI developers**

**1 — High level architecture (components)**

* **ERPNext (Frappe) core** — multi-tenant app extended with new doctypes, UI pages, REST endpoints and background jobs. Single source of truth for institutions, programs, courses, users, enrolment, grades.
* **AI Backend (microservice)** — hosts LLM orchestration, agent controller, RAG retriever, scraping scheduler and connectors. Exposes REST / websocket APIs to ERPNext.
* **Vector DB** — Pinecone / Weaviate / Milvus / Qdrant for embeddings & semantic retrieval.
* **Object store** — S3 (or MinIO) for uploaded files & scraped content.
* **Speech Stack** — STT (Deepgram / Whisper / Google Speech), TTS (ElevenLabs / Amazon Polly / Azure), telephony/WhatsApp (Twilio / Vonage / Gupshup).
* **Agent Orchestrator** — LangChain-style agent workflow: tools to call ERPNext (update grades, fetch course materials), web scraper, retriever, and messaging tools.
* **Analytics & Monitoring** — Prometheus/Grafana for infra, application logs (ELK), model usage dashboards, human-in-loop alerts.
* **Auth & Security** — ERPNext auth + API tokens between ERPNext and AI backend; encryption at rest/in transit.

**2 — Key ERPNext data model (recommended doctypes)**

Create a custom Frappe app voicera with these doctypes:

1. **Institution** (existing ERPNext Tenant)
   * name, contact, domain, admin\_user, allowed\_domains, GDPR policy.
2. **Program**
   * program\_code, title, description, institution.
3. **Course**
   * course\_code, title, program, credits, syllabus (link to Repository).
4. **Teacher**
   * link to ERPNext User, courses\_assigned, voice\_profile\_id (for TTS voice personalization).
5. **Student**
   * link to ERPNext User, enrollment list, learning\_profile (speed, preferences).
6. **Enrollment**
   * student, course, status, cohort.
7. **RepositoryItem**
   * title, course, type (video/audio/pdf/slide/url/scraped), uploaded\_by, storage\_path, embeddings\_status, chunks\_meta.
8. **Lecture**
   * course, teacher, recording\_link, transcript, tags, published\_flag.
9. **VoiceInteraction**
   * user, session\_id, transcript\_in, response\_out, confidence, source\_refs (IDs of RepositoryItem or URLs), timestamp.
10. **AgentTask**
    * id, task\_type (e.g., “generate\_lesson\_plan”, “answer\_query”, “create\_quiz”), status, result, teacher\_review\_required.
11. **ScrapeJob**
    * query, course, status, seed\_urls, last\_run, results (links to RepositoryItem).
12. **ConsentLog / AuditTrail**
    * student\_id, recording\_consent, data\_retention\_policy, timestamps.

3 — RAG pipeline (how teacher uploads become searchable)

1. **Ingest**
   * Teacher uploads file -> stored in S3 -> creates RepositoryItem record in ERPNext.
2. **Preprocess**
   * Convert -> plain text (OCR for images/slides, audio -> transcript using STT).
   * Chunk large text into ~500-1000 token chunks; create metadata (course, lecture time, teacher).
3. **Embeddings**
   * Generate embeddings (OpenAI / Cohere / local model) for each chunk.
   * Store embeddings in vector DB with metadata linking to RepositoryItem and chunk offsets.
4. **Indexing / Search**
   * Semantic retriever + keyword filter (course/program-level scope).
5. **Answering**
   * For an incoming chat/query: retriever fetches top-K chunks (course-scoped), LLM synthesizes an answer with citations and confidence; voice pipeline TTS renders it for student.
6. **Teacher Review**
   * If confidence < threshold, mark AgentTask for teacher review before publishing to class knowledge base.

4 — Agentic AI design (student learning driver)

* **Agent Identity:** “Voicera Agent” with objectives per student (improve comprehension, schedule revision, prep quiz).
* **Tools for agent:**
  + erpnext\_api (get student profile, update progress, create tasks)
  + retriever (RAG over course repo + scraped web)
  + scraper\_tool (ad-hoc web scraping following policies)
  + scheduler (set reminders, schedule tutor sessions)
  + message\_sender (voice/text via Twilio or In-app chat)
* **Agent Flow Example:** Student asks “Explain Fourier Transform” → Agent uses retriever scoped to the student’s course; if insufficient, uses scraper\_tool to fetch more reputable sources (papers, course notes); synthesizes answer; delivers voice answer; logs result; if uncertainty, flags teacher review; schedules follow-up quiz in 2 days.
* **Human-in-loop rules:** teacher can set thresholds (confidence threshold, auto-publish on approval), review edits, and override agent behavior.

5 — Web scraping & content organization

* **Scraper engine** (Python Playwright / Scrapy) running as an async job. Features:  
  + Domain whitelist + blacklist per institution.
  + Priority: academic sources, open content (Google Scholar, edu domains, government sites, reputable blogs).
  + Auto citation metadata capture (title, author, pubdate, URL).
  + Sanitization & content classification (is it tutorial, research, blog, video transcript).
  + Rate limit & obey robots.txt (policy + legal compliance).
* **Storage & Indexing:** store scraped pages as RepositoryItem with provenance metadata and embeddings.
* **Auto-tagging:** NLP classifier to tag by topic, difficulty, course mapping.

6 — Voice chatbot experience (student-facing)

* **Channels:**
  + In-app web chat + voice button (record voice question -> STT -> query pipeline)
  + WhatsApp/Phone (optional) for access outside app via Twilio
* **Session flow:**
  + Student opens chat, says or types question
  + STT converts speech -> agent/RAG pipeline retrieves answer -> LLM crafts response -> TTS returns audio + transcript + citations
  + Student can ask follow-ups; context preserved in session.
* **UX features:**
  + “Ask teacher” toggle to escalate
  + Confidence badge and “sources” button to view links/snippets
  + Play/pause audio, download transcript, save to personal notes
* **Teacher tools:** view common questions, intervene, push curated answers to class repository.

**Multi-tenant & scoping rules**

* Retrieval must be scoped by institution → program → course by default.
* Admins can enable cross-course/global search for research-driven programs.
* Per-institution allowlist/denylist for web domains.

**Data governance & privacy**

* **Consent**: explicit consent flow for audio recording and scraping student-generated content.
* **Retention policy**: configurable per institution (e.g., delete transcripts after X years).
* **Encryption**: S3 server-side encryption + DB encryption for PII; TLS in transit.
* **Access control**: ERPNext roles for teacher/admin/student; audit trail on who viewed/edited AI outputs.
* **Explainability**: return source references with each AI answer; store LLM prompt templates for audits.

**9 — Monitoring & safety**

* **Human-in-loop alerts** when agent confidence low or when answers touch sensitive topics.
* **Toxicity & hallucination checks**: post-process with safety models; if risk, escalate to teacher.
* **Metrics:** active sessions, average response time, accuracy/confidence distribution, teacher intervention rate, top queries by topic.

**10 — Integrations & technology choices (recommended)**

* **ERPNext / Frappe** — Python-based custom app
* **AI models**:  
  + LLM: OpenAI / Anthropic / local Llama2/GPT-NeoX variant
  + Embeddings: OpenAI embeddings or open-source (e.g., sentence-transformers)
* **Vector DB**: Pinecone / Weaviate / Qdrant
* **STT**: Deepgram / Whisper (server), Google Speech for scale
* **TTS**: ElevenLabs for teacher-like voices / custom voice cloning if needed
* **Telephony/WhatsApp**: Twilio (programmable voice + WhatsApp)
* **Scraper**: Playwright / Scrapy + newspaper3k for article parsing
* **Agent framework**: LangChain or custom orchestration layer
* **Storage**: S3 or MinIO; Postgres + Elasticsearch/Pinecone
* **Deployment**: Kubernetes (for scalability) or managed services

**11 — MVP scope (ERPNext-focused) — must-haves**

Deliverables in MVP:

1. ERPNext app with doctypes: Institution, Program, Course, Teacher, Student, Enrollment, RepositoryItem.
2. Upload pipeline: upload doc -> text extraction -> chunk & embedding -> vector DB.
3. Basic voice chatbot UI inside ERPNext (web chat with voice in/out) connecting to AI backend.
4. RAG answering scoped to course repository; returns citations.
5. Simple web-scraper job that takes a query or seed URL, scrapes, and stores content as RepositoryItem.
6. Agent basic: follow-up scheduling + auto-quiz suggestion + teacher review queue.
7. Consent & audit logging.
8. Deployment scripts and basic monitoring.

**12 — Roadmap & timeline (phased)**

Assumes a 4–6 developer team (backend engineers, AI engineer, frontend dev, QA, product). Rough sprint plan:

* **Phase 0 (2 weeks)** — Detailed specs, data model, infra setup, security checklist.
* **Phase 1 (4 weeks)** — ERPNext core doctypes + auth + UI for institution/program/course + CSV enrollment imports.
* **Phase 2 (4 weeks)** — Repository ingestion (uploads, OCR/transcribe), chunking, embeddings pipeline, vector DB integration, semantic search API.
* **Phase 3 (4 weeks)** — Voice chat UI + STT/TTS integration + basic RAG answer flow.
* **Phase 4 (4 weeks)** — Web scraping engine, provenance capture, auto-classification and storage.
* **Phase 5 (4 weeks)** — Agent orchestration: simple agent tasks (answering, schedule follow-up, teacher review queue).
* **Phase 6 (2–4 weeks)** — Pilot with 1–2 institutions, fix issues, performance tuning, finalize teacher-in-loop UX.
* **Phase 7 (ongoing)** — Add multilingual TTS, advanced agent capabilities, analytics, scaling optimizations.