PROJECT REQUIREMENTS DOCUMENT

Bloom Academia - 30-Day MVP

Version 1.0 | January 2026

Project Name	Bloom Academia
Timeline	30 Days (January 11 - February 10, 2026)
Target Event	Gemini 3 Hackathon Submission

1. EXECUTIVE SUMMARY

This document outlines requirements for building a Minimum Viable Product (MVP) of Bloom Academia, an AI-powered school that revolutionizes traditional schooling through personalized, voice-based instruction with real-time visual teaching aids.

1.1 Vision Statement

To solve the global education crisis by providing every child with access to a world-class, AI-powered teacher that adapts to their unique learning style. In Sha Allah, we aim to build a billion-dollar company that makes quality and complete education accessible to 244+ million out-of-school children worldwide.

1.2 Core Innovation

- Real-time voice-based teaching (not just content delivery)
- Interactive visual whiteboard synchronized with voice instruction
- Truly personalized learning paths that adapt in real-time
- Complete school infrastructure replacement (not supplemental)

2. PROBLEM STATEMENT

2.1 Global Education Crisis

- 244+ million children worldwide lack access to quality education (UNESCO)
- Insufficient schools and qualified teachers in underserved regions
- High cost of traditional education infrastructure
- Geographic barriers preventing school access

2.2 Limitations of Traditional Schools

- One-size-fits-all teaching approach
- Limited 1-on-1 attention (30+ students per teacher)
- Rigid schedules that don't accommodate individual learning paces
- High operational costs (buildings, staff, materials)

3. SOLUTION OVERVIEW

An AI-powered school platform featuring real-time voice-based teaching with synchronized visual whiteboard instruction. Students engage in natural conversations with an AI teacher that adapts to their learning style, pace, and comprehension level.

3.1 Key Features

Voice-Based Teaching

- Real-time voice conversations
- Natural language understanding and response
- Interrupt handling (students can ask questions mid-explanation)
- Emotional tone detection and appropriate responses

Interactive Whiteboard

- Full-screen canvas for visual instruction
- Real-time SVG diagram generation by AI
- Synchronized animations with voice narration
- Mathematical equations, diagrams, and visual aids

4. TECHNICAL ARCHITECTURE

4.1 Technology Stack

Layer	Technology
Frontend	Next.js 14+ with React 18 Tailwind CSS + shadcn/ui Fabric.js or Konva (whiteboard) Framer Motion (animations)
AI Model	Gemini 3 Flash
Database	Supabase (PostgreSQL + Auth)
Hosting	Vercel (automatic deployments)
Backend	• **Next.js API Routes** • **Supabase** • **WebSocket connections** (for Gemini Live API streaming)
STT	Soniox WebSocket for live transcription
TTS	Google cloud TTS Neural2 Streaming

4.2 System Architecture

Voice Interaction Flow:

1. **Student speaks** (Web Audio API captures audio)
2. **Audio streams via WebSocket to Soniox STT API**
   - Real-time transcription using `@soniox/speech-to-text-web`
   - Model: `stt-rt-preview` with endpoint detection
   - Returns transcribed text when student finishes speaking
3. **Next.js API routes receive transcribed text** (`/api/teach`)
   - Fetches user profile from Supabase (learning style, strengths, struggles)
   - Loads recent session conversation history
   - Builds comprehensive AI context from 3-layer memory system
4. **Gemini 3 Flash API processes request**
   - Model: `gemini-3-flash-preview` (text-only, NOT Live API)
   - Receives enriched prompt with user context + transcribed message
   - Generates teaching response text + SVG code (if appropriate)

5. **Response processing in parallel:**
   - **(a) Text → Google Cloud TTS:** Converts response to speech audio (Neural2 voice)
   - **(b) SVG extraction:** Parses SVG code from response for whiteboard rendering
   - **(c) Memory update:** Saves interaction to session history in Supabase
6. **Frontend receives complete response:**
   - Plays audio (student hears teacher)
   - Renders SVG on whiteboard (student sees visual aid)
   - Updates UI state with conversation
7. **Progress/interaction data saved to Supabase** for memory system

5. DEVELOPMENT TIMELINE & FEATURES

5.1 Week 1: Foundation (Days 1-7)

Days 1-2: Landing Page

- Design and develop marketing website
- Hero section with compelling copy and demo video placeholder
- Email capture for waitlist
- Deploy to production

Days 3-5: Voice Pipeline Integration

Goal: Implement complete voice interaction system (STT → AI → TTS)

Day 3: API Setup & Soniox Integration

- Set up all API credentials:
  - Gemini API key from Google AI Studio
  - Soniox API key for speech-to-text
  - Google Cloud service account for TTS
  - Apply for hackathon credits (Gemini API)
- Implement Soniox WebSocket integration:
  - Create `/api/stt/temp-key` endpoint for temporary API keys
  - Build frontend audio capture using Web Audio API
  - Integrate `@soniox/speech-to-text-web` SDK
  - Test real-time transcription with endpoint detection

Day 4: Gemini 3 Flash & Memory System

- Implement Gemini 3 Flash integration:
  - Install `@google/genai` SDK
  - Create Gemini client wrapper (`lib/ai/gemini-client.ts`)
  - Build context builder with memory system
  - Create `/api/teach` main endpoint
  - Test text-based teaching responses
- Set up Supabase database:
  - Run schema.sql (users, sessions, interactions, progress tables)
  - Implement user profile queries
  - Test session memory storage

Day 5: Google Cloud TTS & Complete Pipeline

- Implement Google Cloud Text-to-Speech:
  - Install `@google-cloud/text-to-speech`
  - Create TTS synthesis function (`lib/tts/google-tts.ts`)

    - Configure Neural2 voice (en-US-Neural2-F)
    - Test audio generation from text responses
- **Test complete voice loop:**

1. Student speaks → Soniox transcribes
2. Text sent to Gemini 3 Flash → AI responds
3. Response converted to speech → Student hears
4. Interaction saved to memory system

Days 6-7: Basic UI Framework

- Set up Next.js project with Tailwind CSS
- Create blank canvas interface for whiteboard
- Add voice indicator UI (microphone pulsing animation)

5.2 Week 2: Core Teaching System (Days 8-14)

Days 8-10: Whiteboard Canvas

- Set up Fabric.js or Konva for canvas rendering
- Implement SVG rendering system
- Build animation system (fade, draw, highlight effects)
- Test with sample mathematical diagrams

Days 11-13: AI Teaching Prompts

- Design comprehensive system prompt for teaching behavior
- Implement structured JSON output parsing
- Test different explanation styles and approaches
- Refine SVG generation prompts for mathematical diagrams

Day 14: Voice-Whiteboard Synchronization

- Implement timing system for content display
- Queue whiteboard updates based on voice timing
- Trigger animations at correct moments in speech

5.3 Week 3: Curriculum Implementation (Days 15-21)

Days 15-18: Complete Fractions Lesson

- Design complete lesson flow from introduction to mastery
- Create 5-7 interactive practice problems
- Build visual aids (pizza diagrams, number lines, shaded shapes)
- Test end-to-end lesson experience
- Iterate based on internal testing feedback

Days 19-21: Additional Math Lessons

- Comparing Fractions lesson
- Adding Fractions lesson
- Introduction to Decimals lesson

- Basic Multiplication lesson

5.4 Week 4: Polish & Launch (Days 22-30)

Days 22-24: User System & Progress Tracking

- Set up Supabase authentication (email/password, Google OAuth)
- Create user profiles with learning preferences
- Build progress tracking dashboard
- Implement learning path visualization

Days 25-26: UI/UX Polish

- Smooth animations and transitions with Framer Motion
- Professional styling and branding
- Responsive design testing (desktop, tablet, mobile)
- Performance optimization

Days 27-28: Beta Testing

- Recruit 5-10 students from friends/family
- Conduct supervised testing sessions
- Collect feedback and observations
- Record video footage for demo
- Fix critical bugs identified

Days 29-30: Demo Production & Submission

- Edit compelling 3-5 minute demo video
- Prepare hackathon submission materials
- Create investor pitch deck
- Final testing and deployment
- Submit to Gemini 3 hackathon

6. SUCCESS METRICS

Technical Metrics:

- Voice interaction latency < 3 seconds
- Whiteboard-voice synchronization accuracy > 95%
- System uptime > 99% during demo period
- Zero critical bugs in core teaching flow

User Experience Metrics:

- 5-10 beta testers complete at least one full lesson
- 80%+ of testers report understanding the taught concept
- Positive feedback on voice interaction naturalness
- Visual aids rated as helpful by majority

Business Metrics:

- Compelling 3-5 minute demo video produced
- Landing page captures minimum 100 email addresses
- Hackathon submission completed on time
- Investor pitch deck prepared and polished

7. RISKS & MITIGATION

Risk	Impact	Mitigation
API Latency (2-3 sec)	Medium - UX concern	Position as thoughtful teaching pace
Internet Dependency	High - Business model	Acknowledge in pitch; show offline roadmap
Technical Complexity	High - Timeline risk	Build iteratively; test independently
Timeline Slippage	High - Delivery risk	Ruthless prioritization; daily check-ins

8. BUSINESS MODEL & VISION

8.1 Revenue Streams

B2C (Business to Consumer):

- Freemium model (basic lessons free, advanced paid)
- Premium subscriptions: $50-200/month (private school replacement)
- Family plans and multi-child discounts

B2G (Business to Government):

- Per-student annual licensing: $100-500/student/year
- Pilot program contracts with ministries of education
- Long-term service agreements

Foundation/Grant Funded:

- Free access for underserved communities
- Partnerships with educational NGOs
- Philanthropic funding from tech foundations

8.2 Competitive Advantages

1. True Personalization: Adapts to each student's learning style in real-time
2. Voice + Visual: Mimics best aspects of human teachers
3. Complete Infrastructure: Not just supplemental, full school replacement
4. Mission-Driven: Focused on solving global access problem
5. Cost Efficiency: Fraction of traditional school operational costs
6. Scalability: Unlimited students with consistent quality

9. AI INTEGRATION SPECIFICATIONS

9.1 Gemini 3 Pro Configuration

Model: gemini-3-pro
API: Gemini Live API (Native Audio)
Connection: WebSocket bidirectional streaming
Latency Target: 1-3 seconds response time

9.2 System Prompt Architecture

Core Identity:
"You are an expert mathematics teacher conducting a voice-based lesson with visual aids. Your goal is to help students truly understand concepts, not just memorize. You are patient, encouraging, and adaptive to each student's learning style."

Teaching Principles:

- Socratic Method: Ask guiding questions, don't just tell
- Multiple Explanations: Try different approaches until understanding clicks
- Visual Reinforcement: Whiteboard should support and clarify speech
- Patient Pacing: Never move on until student demonstrates understanding
- Mistake Analysis: When wrong, explain WHY, not just correct
- Encouragement: Build confidence through positive reinforcement

10. DATABASE SCHEMA

users table:

- id (UUID, primary key)
- name, email, age, grade_level
- learning_style (visual/auditory/kinesthetic)
- strengths, struggles (TEXT[])
- preferences (JSONB)
- total_learning_time (INTEGER in seconds)

lessons table:

- id, title, subject, grade_level, topic
- description, prerequisites (UUID[])
- estimated_duration (INTEGER in minutes)

progress table:

- user_id, lesson_id (foreign keys)
- mastery_level (0-100%)
- attempts, common_mistakes (TEXT[])
- time_spent, completed (BOOLEAN)

In Sha Allah, this project will transform education globally and create a billion-dollar company that serves humanity.

** * * END OF DOCUMENT * * **