**Vibe Studio - Technical Specifications**

**Version:** 2.0 (Defines v1.0 implementation and v2.0/v3.0 roadmap) **Last Updated:** August 29, 2025 **Status:** Version 1.0 Implemented; future versions in planning.

**Part 1: Current System (Version 1.0)**

This section describes the Vibe Studio system as it is currently implemented and operational. The core of this version is a streamlined, AI-powered workflow for creating single-file educational applets.

**1.1. Overview & Goals**

Vibe Studio is a "No-Code" web platform that empowers teachers to create interactive learning applets through natural language conversations with an AI. The primary goal of v1.0 is to create a "magical" user experience where a teacher can describe a pedagogical idea, and the system generates a functional applet and all its necessary metadata automatically, requiring only final approval from the teacher.

**1.2. Core Technologies**

* **Frontend:** HTML, Tailwind CSS, Vanilla JavaScript. Hosted on **Azure Static Web Apps**.
* **Backend:** **Firebase** (US-Central1)
  + Authentication: Firebase Authentication (Email Magic Link).
  + Database: Firestore.
  + Serverless Logic: Cloud Functions for Firebase.
  + File Storage: **Firebase Cloud Storage** for hosting the generated applets.
* **AI Model:** **Google Gemini API** (specifically, a model that supports structured JSON output, like gemini-1.5-flash).

**1.3. User Workflow (The "Magic Flow")**

The user flow is designed to be as simple and intuitive as possible:

1. **Prompt:** A teacher enters the studio and describes their desired applet in a single text area (e.g., *"Create a simple quiz for 4th graders about the multiplication table up to 100. Make it fun and colorful."*).
2. **AI Processing:** The prompt is sent to the askVibeAI Cloud Function. This function communicates with the Gemini API, requesting both the HTML code and the extracted metadata in a structured JSON format.
3. **Instantaneous Results:** The system receives the JSON response. Instantly, the UI updates in two places:
   * **Live Preview:** The generated HTML applet appears in a preview window, fully functional.
   * **Pre-filled Form:** The publishing form is automatically populated with the metadata the AI extracted (App Name: Multiplication Table Quiz, Grade Level: 4th Grade, Domain: Math, etc.).
4. **Review & Publish:** The teacher reviews the auto-filled details, interacts with the live preview, and simply clicks "Publish". The applet is then saved and becomes available in the gallery.

**1.4. Technical Architecture**

The key to v1.0 is the askVibeAI Cloud Function.

**askVibeAI Cloud Function**

This function is the system's brain. It leverages the Gemini API's structured output (JSON mode) capability.

* **Input:** Receives a simple text prompt and the current UI language (en or he) from the client.
* **Logic:**
  1. Constructs a detailed system prompt that instructs the Gemini model to act as an expert educational app developer and to adhere to strict rules (e.g., single-file HTML, use Tailwind CSS, respond only in JSON).
  2. Defines a rigid JSON schema that the model's output must follow.
  3. Sends the user's prompt along with the system instructions and JSON schema to the Gemini API.
* **Output:** Returns the complete JSON object received from Gemini directly to the client.

**Gemini JSON Output Schema**

{

"type": "object",

"properties": {

"htmlCode": {

"type": "string",

"description": "The complete, single-file HTML code for the educational applet."

},

"metadata": {

"type": "object",

"properties": {

"appName": { "type": "string" },

"gradeLevel": { "type": "string" },

"domain": { "type": "string" },

"subDomain": { "type": "string" },

"pedagogicalExplanation": { "type": "string" }

}

}

}

}

**Applet Hosting**

Generated single-file applets are stored in **Firebase Cloud Storage**. Access is provided via secure **Download Tokens**, which generate a stable, public URL for each index.html file without exposing the entire storage bucket.

**Part 2: Future Roadmap**

This section outlines planned features and integrations to expand the Vibe Studio ecosystem.

**2.1. Version 2.0: Seamless fireClass Integration**

The goal of v2.0 is to bridge the gap between the applet creation in Vibe Studio and its usage in the fireClass classroom management system, creating a single, cohesive ecosystem.

**Feature: Auto-Sync to Personal Library**

* **Goal:** Eliminate the manual step where a teacher creates an app in Vibe Studio and then has to copy the link and add it to their fireClass library.
* **Mechanism:**
  1. The fireClass system will need to expose a new, secure Cloud Function endpoint, e.g., addLinkToLibrary.
  2. This endpoint will accept applet metadata (name, URL, description, etc.) and an authenticated teacher's ID token.
  3. After an applet is successfully published in Vibe Studio, the publishHtml function will make a server-to-server API call to this new fireClass endpoint.
  4. The new applet will automatically and instantly appear in the teacher's personal library within fireClass, ready to be used in a lesson.

sequenceDiagram

participant VSUI as Vibe Studio UI

participant VSCF as Vibe Studio Cloud Functions

participant FC as fireClass Cloud Functions

participant FCFS as fireClass Firestore

VSUI->>VSCF: Publish Applet (HTML + Metadata)

VSCF->>VSCF: Save HTML to Firebase Storage, get URL

VSCF->>FC: call addLinkToLibrary(URL, Metadata, AuthToken)

FC->>FCFS: Write new entry to /teachers/{uid}/personal\_links

FC-->>VSCF: Success

VSCF-->>VSUI: Publish Complete

**Feature: Public Library Submission**

* **Goal:** Allow high-quality, teacher-created applets to be shared with the entire community.
* **Mechanism:**
  1. A "Suggest for Public Library" button will be added to the gallery view in Vibe Studio for each applet a teacher has created.
  2. Clicking this button will trigger a backend process (e.g., a Cloud Function) that sends a notification to a system administrator (e.g., via email or by creating a document in a pending\_reviews Firestore collection).
  3. The notification will contain the applet's details and a direct link for the admin to review.
  4. The admin can then choose to copy the applet to the central fireClass public library.

**2.2. Version 3.0: Empowering Advanced Users**

The goal of v3.0 is to re-introduce support for advanced teachers who wish to create more complex, multi-file applications that may include assets like images, audio files, or external JavaScript libraries.

**Feature: Manual ZIP Upload to Netlify**

* **Goal:** Provide a development path for complex projects that cannot be contained within a single HTML file.
* **Mechanism:**
  1. A new "Upload Manually" tab will be added to the Vibe Studio UI.
  2. This interface will allow a teacher to upload a .zip file containing their entire project folder.
  3. A new Cloud Function, e.g., publishZipToNetlify, will be created.
  4. This function will:
     + Receive the zipped file.
     + Unpack it in a temporary environment.
     + Use the Netlify API (with a separate API key stored in Firebase Secrets) to deploy the folder's contents as a new site.
     + Save the resulting Netlify URL to the app gallery, alongside the AI-generated apps.
* **Resulting Architecture:** This will create a hybrid hosting model. Simple, AI-generated applets will continue to be hosted on Firebase Storage for simplicity and cost-effectiveness, while complex, manually-uploaded projects will be hosted on Netlify, leveraging its powerful build and deployment platform.