**Vibe Studio - מפרט טכני (גרסה 2.0 - Gemini API)**

**גרסה:** 2.0

**תאריך:** 29 באוגוסט 2025 **נושא:** שכתוב המערכת לשימוש ב-Google Gemini API עם פלט מובנה (Structured Output)

**1. סקירה כללית ומטרות**

גרסה 2.0 של "Vibe Studio" מהווה קפיצת מדרגה בחוויית המשתמש, ומטרתה להפוך את תהליך יצירת היישומונים החינוכיים לחוויה חלקה, מהירה ו"קסומה" ככל האפשר. המערכת תעבור להשתמש באופן בלעדי ב-**Google Gemini API**, תוך מינוף יכולות מתקדמות של **הבנת שפה טבעית** ו**פלט מובנה (JSON)**.

המטרה המרכזית היא למכן את תהליך איסוף הנתונים על היישומון (מטא-דאטה), כך שהמורה יוכל להתמקד אך ורק בתיאור הרעיון הפדגוגי שלו, והמערכת תסיק מכך את כל הפרטים הנדרשים באופן אוטומטי.

**2. ארכיטקטורה וטכנולוגיות**

הארכיטקטורה הכללית נשמרת, אך עם שינוי מהותי ברכיב הבינה המלאכותית.

* **Frontend (צד לקוח):** ללא שינוי (HTML, CSS, JavaScript on Azure Static Web Apps).
* **Backend (צד שרת):** Firebase (us-central1).
* **פלטפורמת פריסה:** Netlify API.

**שינוי ליבה: מודל AI**

* **מודל AI:** המערכת תשתמש באופן בלעדי ב-**Google Gemini API**.
* **יכולת מפתח:** שימוש במצב **פלט מובנה (JSON Mode)** של ה-API. הדבר יבטיח שהתשובות מהמודל יהיו תמיד בפורמט JSON עקבי וניתן לעיבוד, ולא כטקסט חופשי.

**3. תרחיש משתמש מרכזי (הזרימה החדשה)**

תרחיש המשתמש מדגים את השיפור הדרמטי בחוויה:

1. **כתיבת הנחיה (Prompt):** מורה נכנס ל-Studio וכותב הנחיה בשפה טבעית:

"אני צריך משחקון **לכיתה ד'** בנושא **לוח הכפל**. המטרה היא תרגול מהיר של כפל עד 100. שיהיה כיף וצבעוני."

1. **קריאה ל-Gemini API:** המערכת שולחת את ההנחיה לפונקציית askVibeAI החדשה.
2. **קבלת קוד ומטא-דאטה:** הפונקציה מקבלת מ-Gemini API תגובת JSON המכילה **גם את קוד ה-HTML המלא** של המשחקון, ו**גם אובייקט מטא-דאטה** שהמודל חילץ מההנחיה.
3. **תצוגה מקדימה חיה:** קוד ה-HTML מוצג מיד בחלון התצוגה המקדימה.
4. **טופס פרסום מלא מראש:** במקביל, טופס הפרסום בצד **מתמלא אוטומטית** עם הנתונים שהמודל הסיק:
   * **שם האפליקציה:** תרגול לוח הכפל
   * **שכבת גיל:** כיתה ד'
   * **תחום:** מתמטיקה
   * **תת-תחום:** לוח הכפל
5. **אישור ופרסום:** המורה סוקר את הפרטים, יכול לערוך אותם במידת הצורך, ולוחץ על "פרסם". התהליך הידני של מילוי הטופס נחסך ממנו לחלוטין.

**4. דרישות פונקציונליות - שכתוב askVibeAI**

השינוי המרכזי במערכת מתרכז בשכתוב מלא של פונקציית הענן askVibeAI.

**4.1. הנדסת פרומפטים מתקדמת**

הפונקציה תהיה אחראית על בניית "פרומפט-על" מתוחכם שיישלח ל-Gemini API. הפרומפט יפותח וייבדק באמצעות **Gemini AI Studio** ויכלול:

* **הוראות** מערכת (System **Instructions):** סט כללים נוקשה שיגדיר את "אישיות" המודל. לדוגמה:

"אתה מומחה לפיתוח יישומונים חינוכיים אינטראקטיביים בקובץ HTML יחיד. עליך להקפיד תמיד על הכללים הבאים: (1) כל הקוד (HTML, CSS, JS) יהיה בקובץ אחד. (2) השתמש ב-Tailwind CSS לעיצוב. (3) ודא רספונסיביות מלאה. (4) הקוד יהיה קריא ועם הערות. (5) תמיד החזר תשובה בפורמט JSON על פי הסכמה שסופקה."

* **דוגמאות (Few-shot Examples):** הפרומפט יכלול 2-3 דוגמאות של קלט (הנחיית מורה) ופלט (JSON מושלם), כדי "לאמן" את המודל על הפורמט והאיכות הנדרשים.

**4.2. הגדרת סכמת פלט (JSON Schema)**

הדרישה המרכזית מה-API תהיה להחזיר תמיד אובייקט JSON שתואם לסכמה קבועה מראש. זהו החלק הקריטי ביותר בארכיטקטורה החדשה.

**סכמת ה-JSON שתוגדר בבקשה ל-Gemini API:**

{

"type": "object",

"properties": {

"htmlCode": {

"type": "string",

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

},

"metadata": {

"type": "object",

"properties": {

"appName": {

"type": "string",

"description": "A short, descriptive name for the app in Hebrew."

},

"gradeLevel": {

"type": "string",

"description": "The target grade level, e.g., 'כיתה ג'",

},

"domain": {

"type": "string",

"description": "The main educational subject in Hebrew, e.g., 'מתמטיקה'."

},

"subDomain": {

"type": "string",

"description": "The specific topic within the domain in Hebrew, e.g., 'שברים פשוטים'."

},

"pedagogicalExplanation": {

"type": "string",

"description": "A brief explanation of the pedagogical goal of the applet in Hebrew."

}

},

"required": ["appName", "gradeLevel", "domain", "subDomain", "pedagogicalExplanation"]

}

},

"required": ["htmlCode", "metadata"]

}

**4.3. זרימת לוגיקה חדשה ב-askVibeAI**

1. קבלת הנחיית טקסט פשוטה מהמורה (מה-Frontend).
2. בניית הבקשה המלאה ל-Gemini API, הכוללת את הנחיית המורה, הוראות המערכת, והגדרת סכמת ה-JSON הנדרשת.
3. שליחת הבקשה ל-Gemini API.
4. קבלת תגובת JSON מה-API.
5. וידוא שהתגובה תקינה ותואמת לסכמה.
6. החזרת אובייקט ה-JSON בשלמותו אל ה-Frontend.

**5. השפעה על ממשק המשתמש (Frontend)**

* **לוגיקת הצ'אט:** במקום לקבל רק טקסט (HTML), ה-JavaScript יקבל אובייקט JSON.
* **onSuccess:** כשהתשובה מתקבלת:
  + response.htmlCode יוזרק לתוך ה-iframe של התצוגה המקדימה.
  + response.metadata ישמש למילוי אוטומטי של שדות הטופס (שם, כיתה, תחום וכו').

שאר המערכת, כולל הגלריה, תהליך הפרסום ל-Netlify והאינטגרציה העתידית עם fireClass, נשאר ללא שינוי.