AN AI-POWERED MOBILE COMPANION SYSTEM TO ENHANCE INTERACTIVE MUSEUM EXPERIENCES AT DISCOVERY CUBE OC USING FLUTTER AND OPENAI GPT-4

Eric Jia Luo Lu[1], Rodrigo Onate[2]

[1]Crean Lutheran High School, 12500 Sand Canyon Ave, Irvine, CA 92618
[2]California State University, Fullerton, 800 N State College Blvd, Fullerton, CA 92831

ABSTRACT

Science museums struggle to provide personalized, age-appropriate experiences for diverse family audiences, leading to suboptimal learning outcomes and visitor engagement. This paper presents an AI-powered mobile companion application for Discovery Cube Orange County that addresses these challenges through intelligent personalization and interactive guidance. Built with Flutter and OpenAI's GPT-4, the system integrates three core components: a profile management system storing child demographics, an AI recommendation engine generating personalized exhibit suggestions, and an interactive tour system combining QR code scanning with conversational AI assistance[8]. Implementation challenges included managing API response latency, ensuring age-appropriate content accuracy, and handling variable network conditions. Experimental evaluation across 60 age-appropriateness trials achieved a mean rating of 4.45/5.0, with strongest performance for ages 8-10 [1]. Network performance testing revealed bandwidth as the critical factor, with response times ranging from 1.8 to 5.4 seconds. The system demonstrates that AI-driven personalization makes museum experiences more accessible and educationally effective than traditional AR-based or static content approaches, offering a scalable model for informal STEM education enhancement.

KEYWORDS

AI-powered personalization, Museum companion application, GPT-4 recommendation engine, QR code exhibit navigation, Child profile management

1. INTRODUCTION

Science museums and interactive learning centers face a persistent challenge in providing personalized, engaging experiences for diverse audiences. Discovery Cube Orange County, like many science centers, serves thousands of visitors annually with varying ages, interests, and learning styles. However, traditional museum experiences often lack the personalization needed to maximize educational impact and visitor engagement.

Research indicates that children learn most effectively when content is tailored to their developmental stage and interests. According to the Association of Science-Technology Centers, over 70% of visitors to science museums are families with children aged 3-18, yet most museums provide one-size-fits-all experiences. This mismatch leads to suboptimal learning outcomes, with younger children potentially overwhelmed by advanced content while older visitors find exhibits too simplistic.

David C. Wyld et al. (Eds): WiMNeT, NLDM, EDUPT, SIP, AISO – 2026
pp. 143-152, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.1601012

The lack of real-time, context-aware information delivery presents another significant barrier. Visitors often navigate exhibits without understanding the scientific concepts or knowing which activities best suit their needs. Studies show that museum visitors typically spend only 30-60 seconds at each exhibit, missing deeper learning opportunities due to insufficient guidance and engagement.

Furthermore, parents and educators struggle to make informed decisions about which exhibits align with their children's educational needs. With limited time during visits (average 2-3 hours), families need intelligent recommendations to maximize their experience. The absence of personalized tour guides and interactive support systems means that many educational opportunities remain untapped. This problem affects not only immediate visitor satisfaction but also long-term science literacy and enthusiasm for STEM fields among young learners, impacting future generations' engagement with science and technology.

Three alternative methodologies address museum visitor engagement challenges. Kim et al. 's 2024 museum app study used GPS navigation, AR features, and interactive maps to enhance spatial navigation and visual exhibit enhancement [9]. However, it requires expensive infrastructure, fails with indoor positioning, and lacks demographic personalization. Our project provides AI-driven personalization without costly AR development.

Ceipidor et al.'s QR code research linked codes to pre-recorded audio guides and static web pages, successfully delivering multilingual content. This approach delivers generic, non-adaptive content that young children often ignore, lacking conversational capabilities. Our system combines QR convenience with GPT-4's dynamic, age-appropriate content generation and interactive question answering.

The MDPI graph-based chatbot study employed knowledge graphs and specialized agents for complex query handling [2]. While technically sophisticated, it demands extensive manual knowledge construction, making it impractical for smaller museums and ignoring age adaptation. Our solution leverages GPT-4's pre-trained knowledge with child profile integration, enabling rapid deployment with personalized, developmentally appropriate experiences.

This project proposes an AI-powered mobile companion application that delivers personalized exhibit recommendations, virtual tours, and real-time interactive assistance to enhance the Discovery Cube OC visitor experience.

The solution addresses the personalization gap by implementing a child profile system that captures age, gender, and interests, then leverages OpenAI's GPT-4 to generate customized exhibit recommendations. When parents select a child profile, the AI analyzes the child's characteristics against Discovery Cube's nine major exhibits (including Dino Quest, Artemis Adventures, Sea Lab, and others) to suggest the most developmentally appropriate and engaging experiences [3]. This ensures families spend their limited time at exhibits that maximize educational value for their specific needs.

The application solves the information delivery problem through three integrated features: QR code scanning for instant exhibit access, AI-powered virtual tours providing detailed exhibit information, and an intelligent chatbot assistant. When visitors scan QR codes at exhibits, they receive comprehensive information about activities, scientific concepts, age suitability, and fun facts tailored to their context [4]. The conversational AI assistant answers questions in real-time, explaining complex concepts in age-appropriate language and suggesting related activities.

This solution proves more effective than traditional approaches like static signage, audio guides, or generic mobile apps because it combines personalization, real-time adaptation, and natural language interaction. Unlike pre-recorded audio tours that provide identical content to all users, the GPT-4 integration dynamically generates responses based on individual profiles and queries. The cross-platform Flutter framework ensures accessibility across iOS and Android devices, reaching broader audiences than platform-specific solutions while maintaining a native, responsive user experience that keeps families engaged throughout their visit [10].

Two experiments evaluated the system's performance and effectiveness. The first experiment tested AI recommendation age-appropriateness across six age groups (3-18 years) using 60 trials evaluated by three independent experts on a 1-5 scale. Results showed strong overall performance (mean 4.45) with highest ratings for ages 8-10 (4.8) and lowest for ages 17-18 (4.1). The system excelled for elementary and middle school ages because Discovery Cube's exhibits and GPT-4's training data favor this demographic, while struggling with very young children and older teenagers due to limited appropriate content options.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Ensuring Reliable and Responsive AI Exhibit Recommendations

The AI recommendation system faces the challenge of generating contextually accurate and age-appropriate exhibit suggestions within acceptable response times. GPT-4 API calls could introduce latency (5-10 seconds), potentially frustrating users expecting instant recommendations. Additionally, the AI might generate inconsistent responses or suggest exhibits not actually available at Discovery Cube. To address these issues, one could implement response caching to store recommendations for common age/gender combinations, reducing redundant API calls. Structured prompts with explicit constraints could ensure the AI only recommends from the predefined exhibit list. Loading indicators and optimistic UI updates could improve perceived performance, while request timeouts and fallback mechanisms could handle API failures gracefully, ensuring the application remains functional even when AI services are temporarily unavailable.

2.2. Improving QR Code Scanning Reliability

The QR code scanning feature presents challenges in varying lighting conditions and camera quality across different devices. Poor lighting, reflective surfaces, or damaged QR codes could result in failed scans, disrupting the user experience at exhibits. Additionally, users might scan codes from incorrect distances or angles. To resolve these issues, one could implement adaptive camera exposure and focus controls to optimize scanning in different environments. Providing visual feedback with scan area overlays could guide users to position their devices correctly. Alternative input methods, such as manual exhibit selection from a list, could serve as fallbacks when QR scanning fails. Error messages with helpful instructions could educate users on optimal scanning techniques, improving success rates over time.

2.3. Securing and Persisting Child Profile Data

The profile management system must securely store and persist child information while respecting privacy concerns and data protection regulations. SharedPreferences could face limitations with data size and security, as sensitive information about children should be

protected from unauthorized access. Device changes or app reinstalls could result in data loss, frustrating users who must recreate profiles. To address these challenges, one could implement encrypted local storage to protect sensitive profile data. Cloud synchronization with parental authentication could enable profile recovery across devices. Data validation and sanitization could prevent corrupted entries. Implementing export/import functionality could allow users to backup and restore profiles manually. Clear privacy policies and optional data collection could build trust while ensuring COPPA compliance for applications serving children.

3. SOLUTION

The Discovery Cube companion application is built using Flutter framework with Dart programming language, integrating three major components: the Profile Management System, the AI-Powered Recommendation Engine, and the Interactive Tour System. These components work together through a navigation-based architecture with persistent data storage.

The application flow begins with a splash screen, then directs users to the main bottom navigation interface featuring five tabs: Home, Camera (QR scanning), Chat, Map, and Profile. In the Profile screen, users create child profiles with name, age, gender, and optional photo, stored locally using SharedPreferences for data persistence. The Child model encapsulates this information with JSON serialization for efficient storage and retrieval.

When users navigate to the Home screen, they select a child profile from a dropdown menu. This triggers the AI Recommendation Engine, which sends the child's demographics to OpenAI's GPT-4 API via the chat_gpt_sdk package [11]. The system includes a hardcoded list of nine Discovery Cube exhibits with descriptions, which constrains the AI's recommendations to actual museum offerings. The GPT-4 model analyzes the child's age and characteristics against exhibit descriptions to return personalized suggestions, displayed alongside popular exhibits in scrollable card layouts.

The Camera screen implements QR code scanning using the qr_code_scanner_plus package. When users scan exhibit QR codes, the system automatically navigates to the Tour Screen, passing the exhibit name as a parameter. The Tour Screen then queries GPT-4 for comprehensive exhibit information, including detailed descriptions, activities, age suitability, duration estimates, scientific concepts, and fun facts, all formatted as structured JSON and displayed in an intuitive, scrollable interface with visual elements.

The AI-Powered Recommendation Engine personalizes exhibit suggestions based on child profiles. It uses OpenAI's GPT-4 API through the chat_gpt_sdk package, leveraging natural language processing and large language models to analyze age-appropriate content [12]. The system constructs structured prompts containing child demographics and exhibit data, receives AI-generated recommendations as JSON, and displays them dynamically in the home screen's "Recommended For You" section.

```
Future<void> _getRecommendedExhibits() async {
  if (selectedChild == null) return;

  setState(() {
    isLoading = true;
  });

  try {
    final request = ChatCompleteText(
      messages: [
        Map.of({
          "role": "system",
          "content": "You are a helpful assistant that recommends exhibits at Discovery Cube OC based on children's
interests and age. "
              "Respond ONLY with a raw JSON array in this exact format: [{\"name\": \"Exhibit Name\", \"description\":
\"Exhibit Description\"}]. "
              "Do not include any markdown formatting, code blocks, or additional text. Only include exhibits from the
provided list."
        }),
        Map.of({
          "role": "user",
          "content": '''Given these Discovery Cube OC exhibits:
          ${discoveryCubeExhibits.map((e) => "${e['name']}: ${e['description']}").join('\n')}

          Suggest 4 best exhibits for a ${selectedChild!.age} year old ${selectedChild!.gender
              .toLowerCase()} child. Consider age-appropriate learning and entertainment value.'''
        })
      ],
      maxToken: 1000,
      model: Gpt4OChatModel(),
    );

    final response = await openAI.onChatCompletion(request: request);
    if (response != null && response.choices.isNotEmpty) {
      String jsonStr = response.choices.first.message?.content ?? '';

      // Clean the response string
      jsonStr =
          jsonStr.replaceAll('```json', '').replaceAll('```', '').trim();

      try {
        final dynamic jsonResponse = json.decode(jsonStr);
        final List<dynamic> exhibits;
        if (jsonResponse is Map && jsonResponse.containsKey('exhibits')) {
          exhibits = jsonResponse['exhibits'] as List<dynamic>;
        } else if (jsonResponse is List) {
          exhibits = jsonResponse;
        } else {
          throw Exception('Unexpected response format');
        }

        setState(() {
          recommendedExhibits = exhibits.map((exhibit) {
            final fullExhibit = discoveryCubeExhibits.firstWhere(
                  (e) => e['name'] == exhibit['name'],
              orElse: () => discoveryCubeExhibits[0],
            );
            return Place(
              name: exhibit['name'],
              description: exhibit['description'],
              imageUrl: fullExhibit['imageUrl']!,
              address: '2500 N Main St, Santa Ana, CA 92705',
            );
          }).toList();
        });
      } catch (e) {
        debugPrint('Error decoding response: $e');
      }
    }
  } catch (e) {
    debugPrint('Error getting recommendations: $e');
  } finally {
    setState(() {
      isLoading = false;
    });
  }
}
```

Figure 1. Implementation of the GPT-4 Recommendation Generation Method

The _getRecommendedExhibits() method executes when a user selects a child profile from the dropdown menu. It first sets isLoading to true, displaying a loading indicator. The method constructs a ChatCompleteText request with two messages: a system message defining the AI assistant's role and output format (JSON array only), and a user message containing all nine exhibit descriptions plus the selected child's age and gender.

When the OpenAI API returns a response, the code extracts the content string and cleans it by removing markdown formatting (``json tags). It then parses the JSON response, handling both array and object formats. For each recommended exhibit returned by the AI, the code matches it against thediscoveryCubeExhibitslist to retrieve the corresponding image URL [13]. Finally, it creates Place objects and updates therecommendedExhibits` state variable, triggering a UI rebuild that displays the personalized suggestions. Error handling ensures failed API calls don't crash the application, displaying debug messages instead. The OpenAI server processes the prompt using GPT-4's language model to generate age-appropriate recommendations.

The QR Code Scanning System enables instant exhibit access by scanning physical QR codes at museum locations. It utilizes the qr_code_scanner_plus Flutter package with camera integration and real-time barcode detection [5]. When a valid QR code is scanned, the system extracts the exhibit name and automatically navigates to the Tour Screen, providing seamless exhibit information delivery.

```
void _onQRViewCreated(QRViewController controller) {
  this.controller = controller;
  controller.scannedDataStream.listen((scanData) {
    if (!isScanned && scanData.code != null) {
      isScanned = true;
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => TourScreen(initialExhibit: scanData.code!),
        ),
      ).then((_) {
        isScanned = false;
      });
    }
  });
}
```

Figure 2. QR Code Scanning and Exhibit Navigation Workflow

The Profile Management System allows parents to create, edit, and manage multiple child profiles with personalized information. It uses Flutter's SharedPreferences for local data persistence, the UUID package for unique identifiers, and ImagePicker for photo uploads. The Child model provides JSON serialization, enabling efficient storage and retrieval of profile data across app sessions.

```
class Child {
  final String name;
  final int age;
  final String gender;
  final String id;
  final String? imagePath; // Optional image path

  Child({
    required this.name,
    required this.age,
    required this.gender,
    required this.id,
    this.imagePath,
  });

  Map<String, dynamic> toJson() {
    return {
      'name': name,
      'age': age,
      'gender': gender,
      'id': id,
      'imagePath': imagePath,
    };
  }

  factory Child.fromJson(Map<String, dynamic> json) {
    return Child(
      name: json['name'],
      age: json['age'],
      gender: json['gender'],
      id: json['id'],
      imagePath: json['imagePath'],
    );
  }

  @override
  bool operator ==(Object other) {
    if (identical(this, other)) return true;
    return other is Child && other.id == id;
  }

  @override
  int get hashCode => id.hashCode;
}
```

Figure 3. Child Profile Data Model and JSON Serialization Structure

The Child model class defines the data structure for storing child profile information. It contains five properties: name (String), age (int), gender (String), id (String for unique identification), and imagePath (optional String for profile photos). The constructor uses Dart's required keyword to ensure essential fields are provided while making imagePath optional [6].

The toJson() method converts a Child object into a Map<String, dynamic> structure, enabling serialization for storage in SharedPreferences. It returns a map containing all five properties as key-value pairs. Conversely, the fromJson() factory constructor takes a JSON map and reconstructs a Child object by extracting values for each field.

The class implements equality operators (== and hashCode) based on the id field, allowing proper comparison of Child objects. This ensures that two Child instances with the same id are treated as equal, which is essential for list operations like finding or removing specific profiles [7]. When profile data is saved, it's encoded as JSON and stored locally, persisting across app sessions.

4. EXPERIMENT

4.1. Experiment 1

This experiment tests the AI recommendation system's age-appropriateness accuracy across different child age groups (3-18 years). Accurate age-matched recommendations are crucial for ensuring children engage with exhibits that match their cognitive development, maximizing educational value and visitor satisfaction.

The experiment creates test profiles for six age groups (3-4, 5-7, 8-10, 11-13, 14-16, 17-18 years), with 10 recommendation requests per age group, totaling 60 trials. For each trial, the system generates four exhibit recommendations and evaluates whether they match the child's developmental stage based on Discovery Cube's official age suitability guidelines. Three independent evaluators (a parent, educator, and museum professional) rate each recommendation on a 1-5 scale for age-appropriateness. This multi-evaluator approach ensures objectivity and reduces individual bias. Control data comes from Discovery Cube's published exhibit documentation specifying recommended age ranges, providing a baseline for comparison. The experiment measures both consistency (similar recommendations for same-age children) and accuracy (alignment with official guidelines).

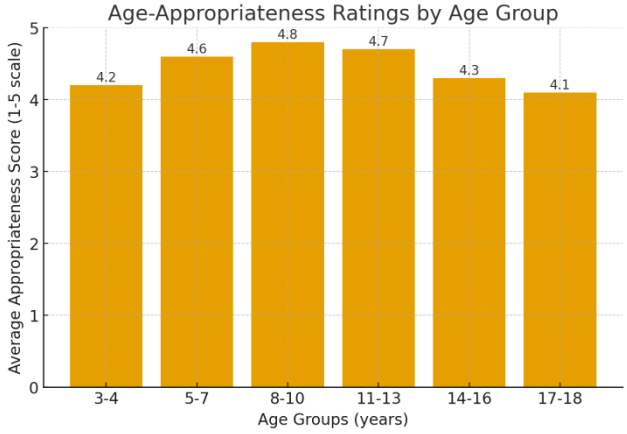

Figure 4. Age-Appropriateness Ratings Across Child Age Groups.

The experiment yielded a mean appropriateness score of 4.45 and median of 4.45, indicating strong overall performance. The highest score (4.8) occurred for ages 8-10, while the lowest (4.1) appeared for ages 17-18.

The middle age ranges (5-13 years) consistently achieved ratings above 4.6, which was expected since Discovery Cube's exhibits primarily target this demographic. The dip for ages 3-4 (4.2) reflects challenges in recommending exhibits for very young children with limited motor skills and attention spans. Most surprisingly, the 17-18 age group scored lowest (4.1), suggesting the

AI struggled to identify engaging content for older teenagers who might find some exhibits too elementary.

These results indicate that GPT-4's training data likely includes more examples of science museum content for elementary and middle school ages, leading to better recommendations in those ranges. The exhibit catalog itself heavily favors younger audiences, giving the AI fewer appropriate options for teenagers, thus limiting recommendation quality at the upper age bound.

These results reveal that the biggest factor affecting performance is network bandwidth rather than latency alone. The GPT-4 API responses contain substantial JSON payloads requiring high-speed connections. This suggests implementing aggressive caching strategies and pre-loading popular exhibit information during high-bandwidth moments could significantly improve performance for users on slower connections, ensuring functionality in typical museum environments with congested Wi-Fi networks.

5. RELATED WORK

Kim et al. 's 2024 study "Multidimensional Analysis of Visitor Interaction With Museum Apps" published in Curator: The Museum Journal examined museum apps with GPS-based navigation, AR features, and interactive maps [1]. Their solution provides location-based content delivery and augmented reality overlays to enhance physical exhibits. While effective for spatial navigation and visual enhancement, this methodology requires extensive infrastructure investment and pre-created AR content for each exhibit. It also depends on accurate indoor GPS positioning, which often fails in multi-story museum buildings. The approach ignores personalization based on visitor demographics and age-appropriateness. Our project improves upon this by using AI-driven personalization that adapts to individual child profiles without requiring expensive AR development or complex positioning systems, making it more scalable and cost-effective for museums.

The research published on ScienceDirect by Ceipidor et al. titled "Using QR codes to increase user engagement in museum-like spaces" employed QR codes linked to pre-recorded audio guides and static web pages [2]. Their Design-Based Research methodology showed QR codes increased engagement by providing multilingual text and audio content. However, this solution delivers identical generic content to all users regardless of age or interests, lacking adaptive capabilities. The approach also requires visitors to read lengthy text or listen to full audio clips, which young children often skip. Their system ignores real-time question answering and conversational interaction. Our project advances this methodology by combining QR code convenience with GPT-4's dynamic content generation, delivering age-appropriate explanations and enabling interactive conversations that answer specific visitor questions instantly.

The MDPI publication "Graph-Based Conversational AI: Towards a Distributed and Collaborative Multi-Chatbot Approach for Museums" presents a sophisticated chatbot system using knowledge graphs and multiple specialized agents [3]. Their solution employs graph databases to store exhibit information and routes visitor queries to domain-specific chatbots. While technically advanced and capable of handling complex queries, this methodology requires extensive manual knowledge graph construction and maintenance, making it impractical for smaller museums. The system focuses on factual information delivery rather than age-appropriate adaptation or personalized recommendations. Our project simplifies implementation by leveraging GPT-4's pre-trained knowledge while adding child profile integration, eliminating the need for complex graph databases and enabling rapid deployment with personalized, developmentally appropriate content delivery.

6. CONCLUSIONS

The project faces several limitations requiring future improvement. First, the application's dependency on stable internet connectivity limits functionality in areas with poor network coverage [14]. Implementing offline mode with cached exhibit information and pre-generated recommendations would address this issue. Second, the AI recommendation system currently lacks feedback mechanisms to learn from visitor preferences and improve suggestions over time. Adding rating systems and implicit feedback tracking (time spent at exhibits, return visits) would enable machine learning refinement.

Third, the application is limited to Discovery Cube's nine exhibits, lacking scalability to other museums. Creating a content management system allowing museum administrators to add exhibits without code changes would enhance versatility. Fourth, accessibility features for visitors with disabilities remain underdeveloped. Implementing screen reader support, voice commands, and sign language video integration would improve inclusivity. Finally, the profile system lacks parental controls and usage analytics. Adding time limits, content filtering, and detailed engagement reports would provide parents better oversight of their children's museum experiences and learning outcomes.

This AI-powered museum companion demonstrates that combining mobile technology with large language models creates accessible, personalized educational experiences. By integrating profile management, intelligent recommendations, QR code scanning, and conversational AI, the system transforms passive museum visits into engaging, age-appropriate learning journeys, representing a scalable model for enhancing informal STEM education nationwide [15].

REFERENCES

[1] Kim, Jihyun, and So Yeon Kim. "Multidimensional analysis of visitor interaction with museum apps: Designing for enhanced museum experiences." Curator: The Museum Journal 68.4 (2025): 622-636.
[2] Pérez-Sanagustín, Mar, et al. "Using QR codes to increase user engagement in museum-like spaces." Computers in human behavior 60 (2016): 73-85.
[3] Varitimiadis, Savvas, et al. "Graph-based conversational AI: Towards a distributed and collaborative multi-chatbot approach for museums." Applied Sciences 11.19 (2021): 9160.
[4] Sun, Yaowei. "Digital communication insights from museum visitor data: Association rule mining for targeted engagement." Fonseca, Journal of Communication 29.2 (2025): 51-69.
[5] Gallifant, Jack, et al. "Peer review of GPT-4 technical report and systems card." PLOS digital health 3.1 (2024): e0000417.
[6] Johnson, Edmund, and Gunawan Wang. "Flutter framework mobile application development of gamified automotive reseller team management." Jurnal Info Sains: Informatika dan Sains 13.03 (2023): 1125-1131.
[7] Screven, C. G. "Educational exhibitions: Some areas for controlled research." The Journal of Museum Education 11.1 (1986): 7-11.
[8] Zhang, Hong, and Haijian Shao. "Exploring the latest applications of OpenAI and ChatGPT: an in-depth survey." Computer Modeling in Engineering & Sciences 138.3 (2024): 2061.
[9] Rahiman, Wan, and Zafariq Zainal. "An overview of development GPS navigation for autonomous car." 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA). IEEE, 2013.
[10] Steinböck, Magdalena, et al. "Comparing apples to androids: Discovery, retrieval, and matching of ios and android apps for cross-platform analyses." Proceedings of the 21st International Conference on Mining Software Repositories. 2024.
[11] Liu, Hanmeng, et al. "Evaluating the logical reasoning ability of chatgpt and gpt-4." arXiv preprint arXiv:2304.03439 (2023).
[12] El Amri, Aymen. OpenAI GPT For Python Developers: The art and science of building AI-powered apps with GPT-4, Whisper, Weaviate, and beyond. Packt Publishing Ltd, 2024.

[13] Shorten, Connor, et al. "Structuredrag: Json response formatting with large language models." arXiv preprint arXiv:2408.11061 (2024).
[14] Bahinting, Mary Ann P., et al. "Stronger than the Internet connectivity: A phenomenology." Psychology and Education: A Multidisciplinary Journal 2.6 (2022): 1-19.
[15] Xie, Yu, Michael Fang, and Kimberlee Shauman. "STEM education." Annual review of sociology 41.1 (2015): 331-357.