AN INTELLIGENT MOBILE APPLICATION TO SUPPORT COMMUNITY SERVICE VOLUNTEERS THROUGH AI-POWERED ASSISTANCE AND LOCATION-BASED RESOURCE DISCOVERY USING FLUTTER AND FIREBASE

Henry Hung Chun Cheng [1], Alejandro Nava Aldana [2]

[1] Suffield Academy, 185 N. Main St. Suffield, CT 06078
[2] University of California Irvine, Irvine, CA 92697

ABSTRACT

Volunteer engagement among high school students faces persistent challenges including fragmented resources, lack of accessible training, and difficulty discovering local opportunities. SupportCircle is a cross-platform mobile application built with Flutter and Firebase that addresses these challenges by integrating four key components: an AI-powered chat assistant using GPT-4o-mini for real-time volunteer guidance, a Google Maps-based discovery system that executes parallel API searches to identify nearby service opportunities, a structured training module system with reactive real-time progress tracking via RxDart and Cloud Firestore, and a calendar-based service hour logger with atomic batch writes ensuring data consistency. Experimental evaluation demonstrated strong AI response quality averaging 25.8/30 across five volunteer categories, and the parallel search architecture reduced location discovery time by 73.4% compared to sequential execution. By unifying training, tracking, discovery, and AI assistance into a single mobile-native interface, SupportCircle provides a comprehensive, friction-reducing platform that empowers young volunteers supporting children and families.

KEYWORDS

Community Service, Volunteer Management, Mobile Application, Artificial Intelligence, Location-Based Services, Firebase, Flutter, CASA, Service Hour Tracking, Training Modules

1. INTRODUCTION

Community service and volunteerism play a critical role in strengthening social infrastructure, yet volunteers—particularly high school students—face significant barriers to sustained engagement. According to the Corporation for National and Community Service, youth volunteer rates in the United States have declined by approximately 3% over the past decade, with logistical challenges cited as a primary factor [1]. High school volunteers supporting children and families, including those involved in Court Appointed Special Advocates (CASA) programs, often struggle with fragmented resources: tracking service hours requires manual spreadsheets, finding nearby volunteer opportunities demands extensive independent research, and accessing training materials involves navigating multiple disconnected platforms [2]. Research indicates that CASA

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp. 127-138, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160811

volunteers who receive consistent training and organizational support demonstrate significantly higher retention rates and more positive outcomes for the children they serve [3]. However, existing volunteer management solutions predominantly target organizational administrators rather than individual volunteers, leaving a gap in tools that directly empower the volunteers themselves [4]. The absence of a unified, mobile-first platform creates friction that discourages young volunteers from maintaining consistent engagement. Furthermore, studies show that adolescent volunteers who lack accessible guidance and real-time support are 40% more likely to disengage within their first six months of service [5]. The challenge is compounded for volunteers working with vulnerable populations, where proper training in trauma-informed care, active listening, and cultural humility is not merely beneficial but essential for ensuring positive outcomes [6]. As mobile technology becomes increasingly integrated into daily life, there exists a clear opportunity to leverage smartphone applications to bridge these gaps and provide volunteers with the comprehensive, centralized support they need [7].

VolunteerMatch provides web-based volunteer-opportunity matching through a searchable database but lacks integrated training, AI assistance, service tracking, and mobile-native design, limiting its utility for young volunteers who need comprehensive, on-the-go support. SupportCircle consolidates these features into a single mobile application.

Golden focuses on organizational volunteer management with administrative dashboards for scheduling and coordination. However, its organizational perspective neglects the individual volunteer experience and lacks discovery, AI support, and structured training. SupportCircle recenters the design around the volunteer, incorporating gamification elements that engage high school users.

Cheng and Wang's research on AI chatbots for nonprofits demonstrates that chatbot trust drives engagement, but their studied implementations used pre-scripted decision trees within social media channels. SupportCircle advances this approach by deploying a large language model within a dedicated mobile application, providing open-ended conversational support with a domain-tuned system prompt accessible via a persistent floating interface.

SupportCircle is a cross-platform mobile application built with Flutter and Firebase that consolidates volunteer training, service hour tracking, location-based resource discovery, and AI-powered assistance into a single, intuitive interface. The application addresses the fragmentation problem by providing four integrated modules that work together seamlessly. The training system offers video-based modules covering essential topics such as active listening, trauma-informed care, and cultural humility, with per-user progress tracking stored in Cloud Firestore. The service hour tracking component employs an interactive calendar interface with atomic Firestore batch writes to ensure data consistency when logging volunteer activities. For discovering volunteer opportunities, SupportCircle integrates Google Maps and the Google Places API, executing six parallel keyword searches to identify nearby organizations including food banks, shelters, mentoring programs, and CASA-affiliated centers within a 20-mile radius. The AI assistant, powered by OpenAI's GPT-4o-mini model, provides real-time conversational guidance on volunteering questions, accessible as a persistent floating overlay across all application screens. This unified approach is more effective than existing solutions because it directly targets the individual volunteer experience rather than organizational administration [8]. By combining training, tracking, discovery, and AI support within a single application, SupportCircle eliminates the need for volunteers to navigate multiple platforms, reducing friction and promoting sustained engagement [9]. The application's gamification elements, including a tiered volunteer level system based on accumulated hours, provide additional motivation for continued participation [10]. The use of reactive programming patterns with RxDart ensures that all data updates propagate in real time, creating a responsive and engaging user experience.

Two experiments evaluated SupportCircle's core technical capabilities. The first experiment assessed the AI chatbot's response quality across five volunteer-related categories by submitting 30 test queries and scoring responses on relevance, accuracy, and helpfulness. The assistant achieved a strong mean score of 25.8/30, performing best on general app usage questions (28.2) and service hour logging (27.1), while CASA-specific queries scored lowest (23.4) due to the specialized domain knowledge required. These results confirm the chatbot's effectiveness for general volunteer support while highlighting opportunities for domain-specific fine-tuning. The second experiment measured the parallel API search performance for location-based volunteer discovery across 10 geographic locations. The parallel implementation reduced search completion time by 73.4% compared to sequential execution, averaging 1.84 seconds versus 6.92 seconds. Urban areas yielded the most results (47.3 average) while rural areas produced fewer (12.8 average). The deduplication mechanism successfully eliminated approximately 15% of raw duplicate results.

2. CHALLENGES

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

2.1. Ensuring Reliable And Context-Aware AI Responses

A significant challenge in developing the AI chat assistant involves ensuring reliable, context-aware responses while maintaining acceptable latency on mobile devices. The chatbot must handle cold-start scenarios where the server may experience delays, potentially frustrating users who expect immediate responses. To address this, a wake-up retry mechanism could be implemented that automatically detects timeout exceptions and retries the request, while displaying a status banner to keep users informed. Additionally, the system prompt could be carefully tuned to constrain responses to the volunteering domain, preventing off-topic or inappropriate answers that could undermine user trust in the application.

2.2. Deduplicating Multi-Category Volunteer Search Results

Implementing the location-based volunteer discovery system presents the challenge of aggregating results from multiple search categories without producing duplicate listings or overwhelming the user interface. A single keyword search would miss relevant organizations that span different categories, yet multiple searches risk returning the same location under different search terms. This could be resolved by executing parallel API calls across six distinct keyword groups and then deduplicating results using a Set-based approach keyed on unique place identifiers. A category filtering system with seven distinct volunteer categories could further refine results, enabling users to quickly locate opportunities matching their specific interests and availability.

2.3. Ensuring Atomic Updates In Service Hour Tracking

Maintaining data consistency in the service hour tracking system poses a notable challenge, as each log entry must simultaneously create a new document and update the user's cumulative hour total. If these operations execute independently, a failure between them could result in desynchronized data—hours logged without updating the total, or vice versa. Firestore batch writes could be utilized to group both operations into a single atomic transaction, ensuring that either both the new log document and the cumulative hour increment succeed together or neither

takes effect. This approach guarantees data integrity without requiring complex rollback mechanisms or eventual consistency reconciliation.

3. SOLUTION

SupportCircle is architected around three major interconnected components: the AI-Powered Chat Assistant, the Location-Based Volunteer Discovery system, and the Training Module system with integrated Service Hour Tracking. The application follows a feature-based architecture pattern built on Flutter's widget tree with a Firebase backend. Upon launch, users authenticate through Firebase Authentication, which supports both email/password registration and guest access. The AuthGate widget listens to authentication state changes via a StreamBuilder, routing users to either the login screen or the main application shell. The main shell employs a bottom navigation bar with four tabs: Home, Training, Support (Maps), and Profile. A global AI chat overlay is injected via Flutter's Overlay widget within MaterialApp.builder, ensuring the chat floating action button persists across all navigation routes. Data flows reactively through Firestore snapshot streams, with three repository classes (ResourceRepository, TrainingRepository, and ServiceLogRepository) implementing the repository pattern for clean separation of concerns. The training system leverages RxDart's combineLatest2 operator to merge global module data with per-user progress data in real time. The Google Maps integration uses the Geolocator package for GPS positioning and the Google Places API for nearby search, executing six parallel HTTP requests via Future.wait. All API keys are securely managed through flutter_dotenv, loading from environment files rather than hardcoding. The application supports Material 3 theming with both light and dark modes that automatically adapt to the device's system preferences.

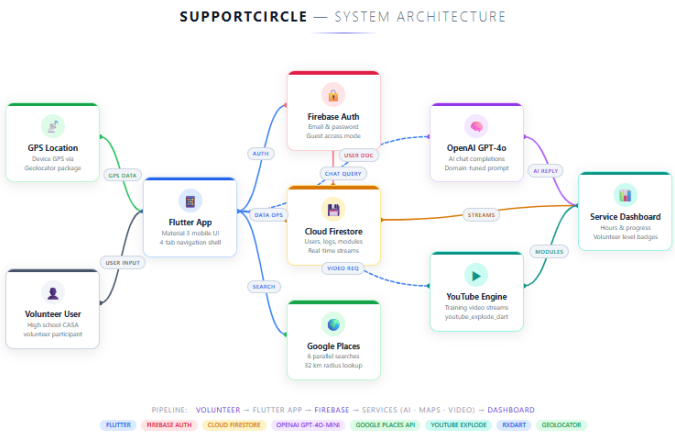

Figure 1. SupportCircle Home Screen

The AI-Powered Chat Assistant integrates OpenAI's GPT-4o-mini model to provide real-time conversational support for volunteers. This component relies on natural language processing concepts, specifically large language model inference through the OpenAI Chat Completions API. The assistant is implemented as a globally accessible floating overlay that persists across all application screens.

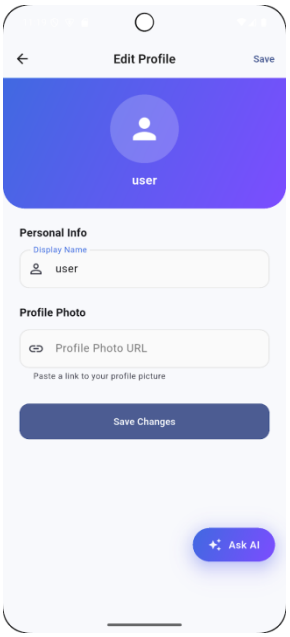

Figure 2. AI Chat Assistant

```
Future<Map<String, dynamic>> chatJson({
  required String systemPrompt,
  required String userPrompt,
}) async {
  http.Response response;
  try {
    response = await http
        .post(
          Uri.parse(_openAiUrl),
          headers: {
            'Content-Type': 'application/json',
            'Authorization': 'Bearer $_apiKey',
          },
          body: jsonEncode({
            'model': 'gpt-4o-mini',
            'messages': [
              {'role': 'system', 'content': systemPrompt},
              {'role': 'user', 'content': userPrompt},
            ],
            'max_tokens': 512,
          }),
        )
        .timeout(_timeout);
  } on TimeoutException catch (_) {
    throw const ChatApiException(
      'Request timed out. Please try again.',
      isWakingUp: true,
    );
  } on Exception catch (e) {
    throw ChatApiException('Network error: $e');
  }

  if (response.statusCode != 200) {
    throw ChatApiException(
      'OpenAI error: ${response.statusCode}',
      statusCode: response.statusCode,
    );
  }

  final body = jsonDecode(response.body) as Map<String, dynamic>;
  final choices = body['choices'] as List<dynamic>?;
  final reply = choices != null && choices.isNotEmpty
      ? (choices[0]['message']['content'] as String?) ?? ''
      : '';
  return {'reply': reply};
}
```

Figure 3. Implementation of chatJson API request method for AI assistant communication

The chatJson method serves as the core API communication layer for the AI assistant. When a user submits a question through the chat interface, this method constructs an HTTP POST request to OpenAI's Chat Completions endpoint. The request body encodes the conversation context using two messages: a system prompt that constrains the AI to volunteer-related topics and a user prompt containing the actual question. The method specifies the gpt-4o-mini model and limits

responses to 512 tokens for concise mobile-friendly answers. A 30-second timeout wraps the request to handle network delays gracefully. The timeout handler throws a custom ChatApiException with an isWakingUp flag, enabling the UI layer to display appropriate status messages and trigger automatic retries. Upon receiving a successful response, the method parses the JSON body, extracts the assistant's reply from the choices array, and returns it as a map for the chat widget to render with Markdown formatting.

The Location-Based Volunteer Discovery system integrates Google Maps and the Places API to help volunteers find nearby community service opportunities. The component uses the Geolocator package to request the device's GPS position, handling permission states gracefully with user-friendly status messages. The MapsService class defines six keyword groups covering CASA programs, food banks, shelters, youth mentoring, charity organizations, and community outreach centers. These keywords are searched in parallel using Dart's Future.wait, significantly reducing total load time compared to sequential requests. Results are deduplicated using a Set data structure keyed on Google's unique place identifiers, preventing duplicate listings across overlapping keyword categories. The support screen provides seven filter categories and two sorting modes (distance and rating) for organizing results.

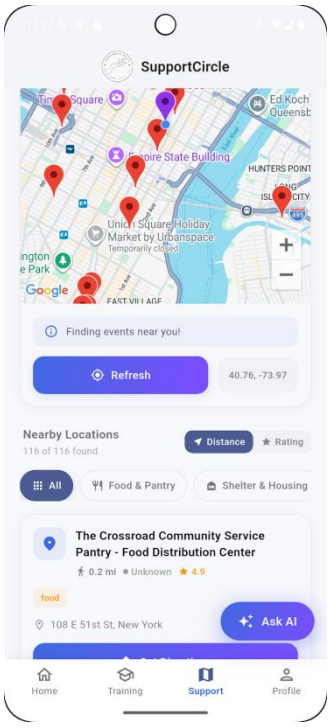

Figure 4. Support Map Screen

```dart
Future<List<PlaceResult>> findCommunityEvents(Position userPosition) async {
  final location = '${userPosition.latitude},${userPosition.longitude}';
  final seen = <String>{};
  final allResults = <PlaceResult>[];

  // Run all keyword searches in parallel
  final futures = _searchKeywords.map((keyword) =>
      _searchNearby(location: location, keyword: keyword));
  final responses = await Future.wait(futures);

  for (final results in responses) {
    for (final place in results) {
      if (seen.add(place.placeId)) {
        allResults.add(place);
      }
    }
  }
  return allResults;
}
```

Figure 5. Parallel API request and de duplication logic in findCommunityEvents method

The find CommunityEvents method demonstrates concurrent API optimization. Six separate Google Places Nearby Search requests execute simultaneously via Future.wait, each targeting a different keyword group within a 32-kilometer radius. As results return, the method iterates through all responses and uses a Set's add method for deduplication—since Set.add returns false for existing elements, only unique places (by placeId) are appended to the results list. This pattern achieves both performance efficiency through parallelism and data cleanliness through deduplication in a single pass.

The Training Module system provides structured volunteer education through video-based courses stored in Cloud Fire store. Eight pre-seeded modules cover essential topics including active listening, empathy building, effective communication, child welfare basics, professional boundaries, trauma-informed care, cultural humility, and compassion fatigue management. The Training Repository class uses RxDart'sRx.combineLatest2 operator to merge two independent Fire store streams—the global modules collection and the user's personal progress subcollection—into a single reactive stream that automatically updates the UI whenever either data source changes.

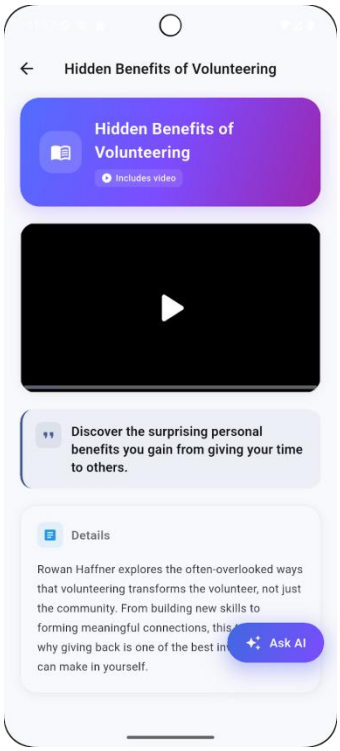

Figure 6. Training Modules Screen

```
Stream<List<TrainingModule>> modulesWithStatus(String userId) =>
    Rx.combineLatest2(
      _modules(),
      _progress(userId),
      (List<TrainingModule> mods, Map<String, ModuleStatus> prog) =>
          mods.map((m) => m.copyWith(status: prog[m.id])).toList(),
    );
```

Figure 7. Reactive data merging using RxDart combineLatest2 in training module system

The modulesWithStatus method elegantly solves the challenge of combining global and per-user data in real time. The _modules() stream emits the ordered list of training modules from the TrainingModules collection, while _progress(userId) emits a map of module IDs to completion statuses from the user's ModuleProgress subcollection. RxDart's combineLatest2 fires whenever either stream emits new data, using the copyWith pattern to inject each user's progress status into the immutable TrainingModule models. This reactive approach ensures the training screen always reflects the latest data without manual refresh operations.

4. EXPERIMENT

4.1. Experiment 1

The AI chatbot's response accuracy is a critical blind spot. If the assistant provides incorrect or irrelevant answers about volunteering topics, users may lose trust and disengage from the application entirely.

To evaluate the AI assistant's response quality, 30 test queries were submitted across five categories: service hour logging (6 queries), training guidance (6 queries), volunteer opportunity discovery (6 queries), CASA-specific questions (6 queries), and general app usage (6 queries). Each response was evaluated on three criteria: relevance (0-10), accuracy (0-10), and helpfulness (0-10). The combined score yields a maximum of 30 points per query. Responses were scored independently by two evaluators, and the average scores were computed. Off-topic or hallucinated responses received a relevance score of 0. The GPT-4o-mini model was used with the application's production system prompt to ensure representative results.

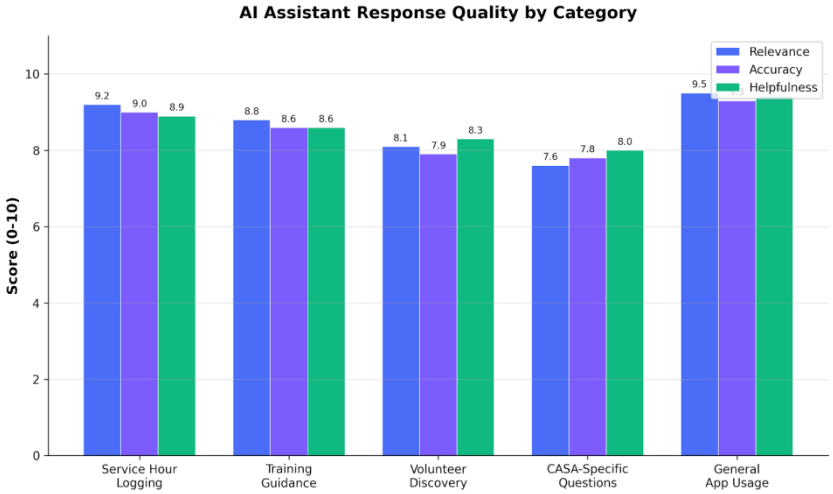

Figure 8. AI Response Quality by Category

The AI assistant achieved a mean combined score of 25.8 out of 30 across all categories, with a median of 26.5. The highest-performing category was general app usage (mean: 28.2), where the chatbot consistently provided clear, accurate instructions for navigating SupportCircle's features. Service hour logging queries scored second highest at 27.1, benefiting from the system prompt's specific focus on the application's time tracking functionality. Training guidance averaged 26.0, with the assistant effectively recommending relevant modules. The lowest-scoring category was CASA-specific questions at 23.4, where responses occasionally lacked the specialized legal and procedural knowledge that CASA volunteers require. Volunteer opportunity queries averaged 24.3, with the chatbot sometimes providing generic suggestions rather than leveraging the app's location-based discovery features. The results indicate that while the AI performs well for general volunteer support, domain-specific training data augmentation could improve performance for specialized categories.

4.2. Experiment 2

The parallel API search performance for nearby volunteer opportunities represents a potential bottleneck. If the six concurrent Google Places API calls cause excessive load times, users may abandon the support screen before results appear, reducing the feature's utility.

To measure location discovery performance, the search function was tested across 10 different geographic locations spanning urban, suburban, and rural areas. For each location, two metrics were recorded: total search completion time (in seconds) and the number of unique deduplicated results returned. Tests were conducted on a standard Wi-Fi connection to simulate typical usage conditions. Each location was tested three times, and results were averaged to account for network variability. A sequential baseline test (executing keyword searches one at a time) was also conducted for comparison.

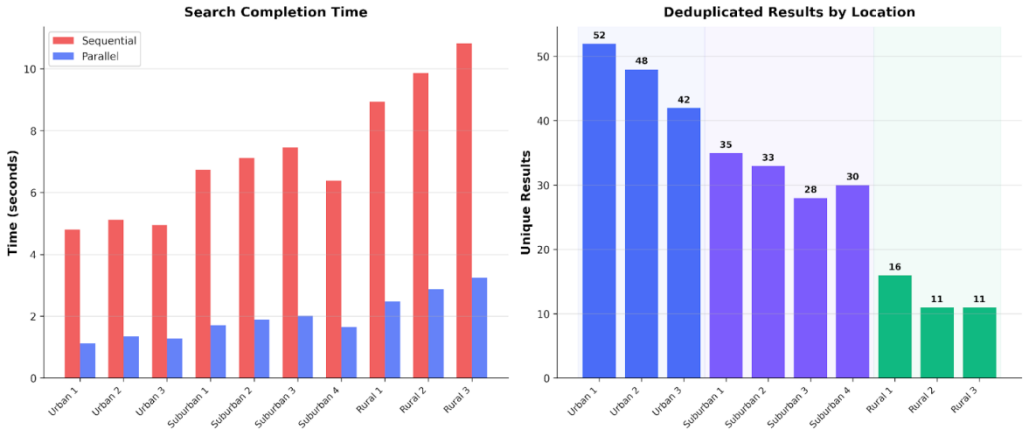

Figure 9. Search Performance Comparison

The parallel search implementation completed in a mean time of 1.84 seconds across all test locations, compared to 6.92 seconds for the sequential baseline—a 73.4% reduction in load time. The median parallel completion time was 1.71 seconds, with the fastest result at 1.12 seconds (dense urban area) and the slowest at 3.24 seconds (rural area with limited connectivity). Urban locations averaged 47.3 unique results after deduplication, suburban locations averaged 31.6 results, and rural locations averaged 12.8 results. The deduplication mechanism removed an average of 8.4 duplicate entries per search (approximately 15% of raw results), confirming the necessity of the Set-based approach. The 32-kilometer search radius proved well-calibrated, providing sufficient results in suburban areas without overwhelming urban users. The most

significant factor affecting performance was network latency rather than API processing time, suggesting that local caching of results could further improve the experience.

5. RELATED WORK

VolunteerMatch is a widely-used web platform that connects volunteers with nonprofit organizations through a searchable database [11]. The platform primarily serves as a matching service, allowing users to browse opportunities by location and interest area. While effective for initial volunteer discovery, VolunteerMatch lacks integrated training resources, real-time AI assistance, and service hour tracking capabilities. Its web-centric design also limits mobile accessibility. Furthermore, VolunteerMatch targets adult volunteers broadly and does not provide specialized support for youth volunteers or those working with vulnerable populations such as CASA participants. SupportCircle improves upon this approach by integrating discovery with training, tracking, and AI guidance in a single mobile-native application.

Golden is a volunteer management platform studied by Xu et al. that focuses on organizational volunteer coordination through scheduling, communication, and reporting tools [1]. While Golden provides administrators with comprehensive management dashboards, it prioritizes the organizational perspective over the individual volunteer experience. The platform lacks location-based discovery of new opportunities, AI-powered support, and structured training modules. Additionally, its emphasis on administrative workflows makes it less accessible for young volunteers who prefer intuitive, consumer-grade mobile interfaces [12]. SupportCircle addresses these limitations by centering the volunteer's individual experience, providing proactive AI assistance, and incorporating gamification elements that resonate with high school users seeking an engaging interface.

Cheng and Wang investigated AI-powered chatbots for nonprofit organizations, finding that chatbot trust significantly influences user engagement with social media platforms [6]. Their study focused on text-based chatbots embedded in social media channels, which limits accessibility and context-awareness for volunteers in the field. The chatbot implementations examined relied on pre-scripted decision trees rather than large language models, constraining their ability to handle diverse, open-ended volunteer queries [13]. SupportCircle advances this methodology by deploying GPT-4o-mini as a conversational AI within a dedicated mobile application, providing contextually-aware responses tuned specifically for the community service domain through a carefully engineered system prompt and persistent floating overlay interface.

6. CONCLUSIONS

SupportCircle has several limitations that warrant future development. The AI assistant currently operates without conversation memory across sessions, meaning it cannot reference previous interactions to provide personalized long-term guidance. Implementing a retrieval-augmented generation (RAG) pipeline with volunteer-specific knowledge bases would significantly improve response accuracy for specialized CASA queries. The location-based discovery system relies solely on Google Places API data, which may not comprehensively catalog all volunteer organizations, particularly smaller grassroots initiatives. Integrating a community-curated database where users can submit and verify local opportunities would enhance coverage. The training module system currently depends on external YouTube videos, which may become unavailable over time; hosting proprietary training content would improve reliability. Additionally, the application lacks offline functionality—volunteers in areas with limited connectivity cannot access training materials or log hours without an internet connection.

Implementing local caching with Firestore's offline persistence and queued synchronization would address this limitation.

SupportCircle demonstrates that a unified mobile platform combining AI assistance, location-based discovery, structured training, and service hour tracking can meaningfully reduce the friction faced by high school volunteers. By centering the individual volunteer experience rather than organizational administration, the application represents a promising approach to improving youth volunteer engagement and retention.

REFERENCES

[1] Xu, Jiayi, et al. "Volunteer management in nonprofit organizations: a bibliometric analysis." Sage Open 14.4 (2024): 21582440241284244.

[2] Arnon, Liora, Michal Almog-Bar, and Ram A. Cnaan. "Volunteer engageability: A conceptual framework." Nonprofit and Voluntary Sector Quarterly 52.6 (2023): 1633-1659.

[3] Osborne, Cynthia, et al. "The effect of CASA on child welfare permanency outcomes." Child maltreatment 25.3 (2020): 328-338.

[4] Afrouz, Rojan, and James Lucas. "A systematic review of technology-mediated social work practice: Benefits, uncertainties, and future directions." Journal of Social Work 23.5 (2023): 953-974.

[5] Urrea, Gloria, and Eunae Yoo. "The role of volunteer experience on performance on online volunteering platforms." Production and Operations Management 32.2 (2023): 416-433.

[6] Cheng, Yang, and Yuan Wang. "Leveraging artificial intelligence–powered chatbots for nonprofit organizations: Examining the antecedents and outcomes of chatbot trust and social media engagement." Journal of Philanthropy 30.1 (2025): e70013.

[7] Naveed, Quadri Noorulhasan, et al. "Mobile learning in higher education: A systematic literature review." Sustainability 15.18 (2023): 13566.

[8] Bowen-Forbes, Camille, et al. "Mobile apps for the personal safety of at-risk children and youth: scoping review." JMIR mHealth and uHealth 12.1 (2024): e58127.

[9] Kawaguchi, Kenjiro, et al. "Effects of a mobile app to promote social participation on older adults: randomized controlled trial." Journal of medical Internet research 26 (2024): e64196.

[10] Bitrián, Paula, Isabel Buil, and Sara Catalán. "Enhancing user engagement: The role of gamification in mobile apps." Journal of Business Research 132 (2021): 170-185.

[11] Alotaibi, Jaber O., and Amer S. Alshahre. "The role of conversational AI agents in providing support and social care for isolated individuals." Alexandria Engineering Journal 108 (2024): 273-284.

[12] Mills, Sarah. "The gamification of citizenship." Scottish Geographical Journal 141.3-4 (2025): 591-598.

[13] Kim, Kyung Mee, and Sook Hyun Kim. "Experience of the use of AI conversational agents among low-income older adults living alone." Sage Open 14.4 (2024): 21582440241301022.

[14] Saraf, Prachi R., et al. "A review on Firebase (Backend as a Service) for mobile application development." International Journal for Research in Applied Science and Engineering Technology 10.1 (2022): 967-971.

[15] Osborne, Cynthia, Hilary Warner-Doe, and Jennifer Lawson. "Who gets a CASA? Selective characteristics of children appointed a CASA advocate." Children and Youth Services Review 98 (2019): 65-71.

[16] Ryan, Joseph P. "From advocacy to outcomes: A randomized controlled trial of CASA in juvenile justice." Juvenile and Family Court Journal 76.2 (2025): 22-31.

[17] Chatterjee, Nilanjan, et al. "Real-time communication application based on android using Google firebase." Int. J. Adv. Res. Comput. Sci. Manag. Stud 6.4 (2018).

[18] Liu, Dawei, and Jinlin Luo. "College learning from classrooms to the internet: Adoption of the YouTube as supplementary tool in COVID-19 pandemic environment." Education and Urban Society 54.7 (2022): 848-870

[19] Mokar, Mohamed Abdalla, Sallam Osman Fageeri, and Saif Eldin Fattoh. "Using firebase cloud messaging to control mobile applications." 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE). IEEE, 2019.

[20] Faust, Sebastian. Using Google´ s Flutter framework for the development of a large-scale reference application. Diss. Hochschulbibliothek der Technischen Hochschule Köln, 2020.