

AN INTELLIGENT CROSS-PLATFORM MOBILE APPLICATION FOR FINANCIAL LITERACY EDUCATION USING FLUTTER, FIREBASE, AND AI-POWERED CHATBOT TECHNOLOGY

Zihan Zeng ¹, Rodrigo Onate ²

¹ St. Michaels University School, 3400 Richmond Road, Victoria, British Columbia V8P 4P5, Canada

² California State University, Fullerton, 800 N State College Blvd, Fullerton, CA 92831

ABSTRACT

Financial literacy remains critically low among young adults, with traditional education methods failing to engage digital-native learners. This paper presents the William Finance Group App, a cross-platform mobile application that addresses this gap through integrated real-time market data, AI-powered educational content, and community engagement features. Built using Flutter, Firebase, and OpenAI's GPT-4, the application delivers personalized financial education through livestock data visualization, on-demand topic explanations, and gamified competitions. Key technical challenges including API rate limiting, LLM content accuracy, and real-time database synchronization were addressed through request queuing, structured prompting, and Firebase's conflict resolution mechanisms. Experimental evaluation demonstrated high educational content quality (4.5/5 mean rating) and reliable API performance (99.2% success rate, 245ms average latency). The application synthesizes theoretical frameworks from digital financial literacy research into practical implementation, offering an effective solution for improving financial literacy among students through technology-mediated education.

KEYWORDS

Financial Literacy, Mobile Application Development, Flutter Framework, Large Language Models, Real-time Data Integration

1. INTRODUCTION

Financial literacy remains a critical challenge affecting millions of individuals globally, particularly among young adults and students. Research indicates that only 57% of adults worldwide are financially literate, with rates even lower among high school and college students [1]. This knowledge gap has significant long-term consequences, as individuals lacking financial literacy are more likely to accumulate debt, make poor investment decisions, and struggle with retirement planning [2]. The emergence of digital financial services has further complicated this landscape, requiring individuals to possess both traditional financial knowledge and digital competencies to navigate modern financial ecosystems effectively [3]. Traditional financial education methods, including classroom instruction and textbook-based learning, have proven inadequate in engaging today's digitally native generation. Students often find conventional approaches disconnected from real-world applications and lack the interactive elements that

modern learners expect [4]. Furthermore, access to quality financial education remains unequal, with students from lower-income backgrounds having fewer opportunities to develop these essential life skills [5]. The COVID-19 pandemic accelerated the adoption of digital financial tools, making digital financial literacy even more crucial. Studies show that 91% of young adults prefer learning financial concepts through interactive digital platforms rather than traditional methods [6]. This shift presents both a challenge and an opportunity: the challenge of developing effective digital learning tools, and the opportunity to leverage technology to democratize access to financial education. The long-term societal impact of financial illiteracy extends beyond individual consequences, affecting economic stability, consumer protection, and wealth inequality across generations.

Koskelainen's digital financial literacy framework provides theoretical foundations for understanding fintech adoption and digital financial behavior but lacks practical implementation tools. Our application transforms these theoretical concepts into hands-on learning experiences with real market data. Kasneci's LLM education research identifies opportunities for AI-powered personalized learning while noting accuracy concerns and integration challenges. Our implementation addresses these through domain-specific prompts, structured response formats, and integration with live financial data for contextual accuracy. Widayanti's gamification study demonstrates that game elements increase engagement but don't directly improve knowledge without mediation. Our application integrates gamification within a comprehensive ecosystem including AI explanations and real-time data, addressing the identified mediation gap. Collectively, these improvements demonstrate how theoretical research can be synthesized into practical educational technology.

The William Finance Group App is a comprehensive cross-platform mobile application that combines real-time market data, AI-powered educational content, and community engagement features to deliver personalized financial literacy education. This solution addresses the identified problems through multiple integrated approaches. First, the application leverages real-time stock market data from the Finnhub API to connect theoretical financial concepts with actual market movements, making learning immediately relevant and engaging [7]. Unlike static textbook approaches, users can observe how concepts like price-to-earnings ratios or market volatility manifest in real stocks they recognize. Second, the integration of GPT-4 powered chatbot technology enables personalized, on-demand explanations of financial concepts tailored to individual learning needs [8]. This AI-driven approach allows students to receive immediate answers to their questions in conversational language, reducing barriers to understanding complex topics. Third, the application employs gamification principles through competitions and community features, which research has shown to significantly increase student engagement and knowledge retention in financial education [9]. Fourth, the cross-platform Flutter framework ensures accessibility across iOS, Android, and web platforms from a single codebase, maximizing reach while minimizing development overhead [10]. The combination of real-time data, AI-powered personalization, gamification, and broad accessibility represents a significant advancement over traditional financial education methods, which typically lack these interactive and adaptive elements. This multi-modal approach addresses diverse learning styles and provides the hands-on, relevant experience that modern learners require.

Two experiments validated critical application components. The first experiment assessed GPT-4's educational content quality across 20 financial topics using independent reviewer ratings on five criteria. Results showed strong performance with an overall mean of 4.5/5, with clarity scoring highest (4.7) due to well-designed system prompts and conversational training data. Completeness scored lowest (4.3) due to token limitations on complex topics. The second experiment evaluated real-time API performance through 7,500 individual stock data requests. Average response time was 245ms with 99.2% reliability, meeting industry standards for retail

financial applications. Performance varied predictably with market conditions, with after-hours queries 37% faster than market-hours queries. Both experiments confirmed that the core technical components deliver reliable, high-quality educational experiences that support the application's financial literacy mission.

2. CHALLENGES

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

2.1. Managing Real-Time Stock API Rate Limits and Caching

Real-time stock market data integration presents significant challenges related to API rate limiting, data consistency, and performance optimization. Financial APIs like Finnhub typically impose rate limits to prevent server overload, which could cause data fetching failures when requesting information for multiple stocks simultaneously. To address this, one could implement request queuing with configurable delays between API calls (e.g., 100ms intervals) to stay within rate limits while still providing timely data. Additionally, caching mechanisms could store recently fetched data to reduce redundant API calls and provide fallback values when the API is temporarily unavailable, ensuring users always see meaningful market information.

2.2. Optimizing LLM Integration for Performance and Reliability

Integrating large language model APIs for educational content generation introduces challenges around response latency, content accuracy, and cost management. LLM API calls can take several seconds to complete, potentially causing poor user experience if not handled properly. One could implement loading states with informative progress indicators to maintain user engagement during processing. For content accuracy, carefully crafted system prompts could guide the model to provide structured, pedagogically sound explanations. Caching frequently requested topics would reduce API costs while improving response times for common queries. Error handling for API failures would ensure the application degrades gracefully when the service is unavailable.

2.3. Ensuring Firebase Data Consistency, Security, and Offline Reliability

Firebase real-time database synchronization presents challenges in data consistency, offline functionality, and security. When multiple users modify shared data simultaneously, conflicts could arise requiring resolution strategies. Firestore's built-in conflict resolution and atomic transactions could address concurrent modification issues. For offline support, Firebase's persistence capabilities could cache data locally, allowing the app to function without internet connectivity and sync changes when reconnected. Security rules must carefully balance accessibility with protection, requiring role-based access control where administrators have elevated permissions while regular users access only appropriate data. Proper indexing strategies would optimize query performance as the database scales.

3. SOLUTION

The William Finance Group App is built using Flutter, Google's cross-platform UI framework, with Dart as the programming language [10]. The application follows an MVC-like architecture with a service-repository pattern, organizing code into distinct layers for pages (UI), services (business logic), and models (data structures). Three major components form the application's core: (1) the Real-time Market Data Module, which fetches live stock prices and financial news

from the Finnhub API; (2) the AI-Powered Learning System, which leverages OpenAI’s GPT-4 to generate personalized educational content; and (3) the Firebase Backend, which handles user authentication, real-time database operations, and cloud storage. The application flow begins when users launch the app and authenticate via Firebase Authentication. Upon successful login, users navigate through a five-tab interface: Home (dashboard with stocks, news, events), Events (club activities), Community (user profiles), Learn (educational content), and Pro (competitions). The Home page immediately fetches real-time market data for 15 popular stocks (AAPL, MSFT, GOOGL, etc.) and displays the latest financial news. Users can access the AI chatbot via a floating action button, which opens a sliding panel interface for conversational learning. The Learn section presents hierarchical financial topics across three categories: Personal Finance, Investing & Wealth Management, and Economic & Business Finance. When users select a topic, the GPT-4 service generates structured explanations with definitions, importance, real-world examples, and practical tips. Administrators access a dedicated panel for managing events, users, and competitions, with all data synchronized in real-time through Cloud Firestore [11].

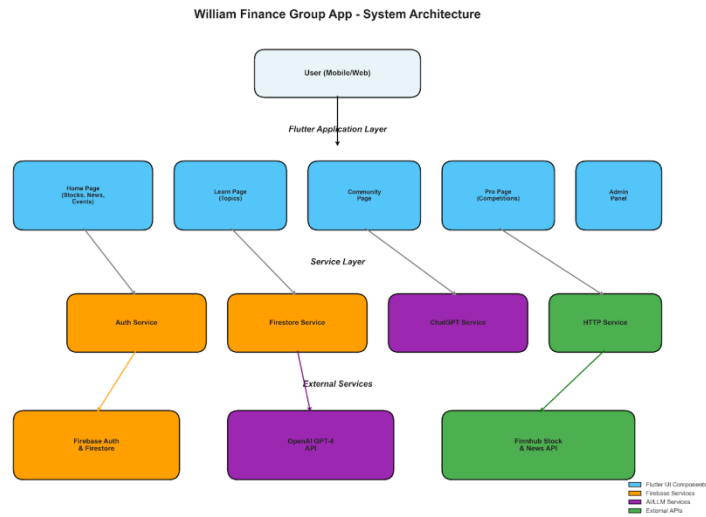


Figure 1. Overview of the solution

The Real-time Market Data Module provides live stock prices and financial news to contextualize learning with actual market movements. It utilizes the Finnhub Stock API for quote data and company profiles. The component implements RESTful API consumption patterns with HTTP requests, JSON parsing, and state management to display dynamic financial information throughout the application.

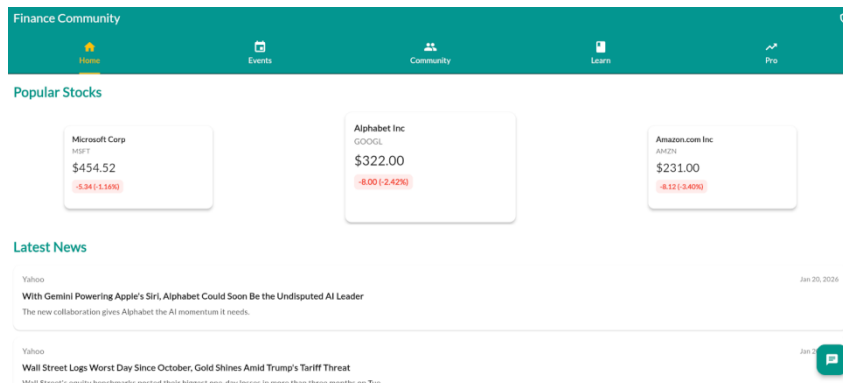


Figure 2. User Interface of the Real-Time Market Data Module Demonstrating Live Stock Visualization

```

Future<void> fetchMarketData() async {
  try {
    for (String symbol in stocks) {
      final quoteResponse = await http.get(
        Uri.parse(
          'https://finnhub.io/api/v1/quote?symbol=$symbol&token=$apiKey',
        ),
      );

      if (quoteResponse.statusCode == 200) {
        final quoteData = json.decode(quoteResponse.body);

        // Get company profile for name
        final profileResponse = await http.get(
          Uri.parse(
            'https://finnhub.io/api/v1/stock/profile2?symbol=$symbol&token=$apiKey',
          ),
        );

        final profileData = profileResponse.statusCode == 200
          ? json.decode(profileResponse.body)
          : {};

        setState() {
          marketData[symbol] = {
            'symbol': symbol,
            'price': quoteData['c']?.toDouble() ?? 0.0, // Current price
            'change': quoteData['d']?.toDouble() ?? 0.0, // Change
            'changePercent': quoteData['dp']?.toDouble() ?? 0.0, // Percent change
            'companyName': profileData['name'] ?? getMarketName(symbol),
          };
        };
      }
    }
  } catch (e) {
    // Handle error
  }
}

```

Figure 3. Implementation of fetch Market Data() Function for Real-Time API Integration

The `fetchMarketData()` function executes when the Home page initializes, iterating through a predefined list of 15 stock symbols. For each symbol, it makes an HTTP GET request to Finnhub's quote endpoint, which returns current price (c), daily change (d), and percentage change (dp). Upon receiving a successful response (status 200), the JSON response is decoded into a Dart map. A second request retrieves company profile information. The `setState()` call updates the component's state with structured market data including symbol, price, change values, and percentage changes. The 100-millisecond delay between iterations prevents API rate limiting, ensuring all requests complete successfully. The Finnhub server processes each request by querying real-time market data from exchanges and returning standardized JSON responses. Default values (0.0) handle null responses gracefully. This data ultimately populates the carousel slider displaying live stock information to users.

The AI-Powered Learning System generates personalized educational content using OpenAI's GPT-4 large language model. It implements natural language processing (NLP) concepts where the model understands context and generates human-like explanations [8]. The service accepts topic parameters and returns structured markdown content covering definitions, importance, examples, and practical tips.

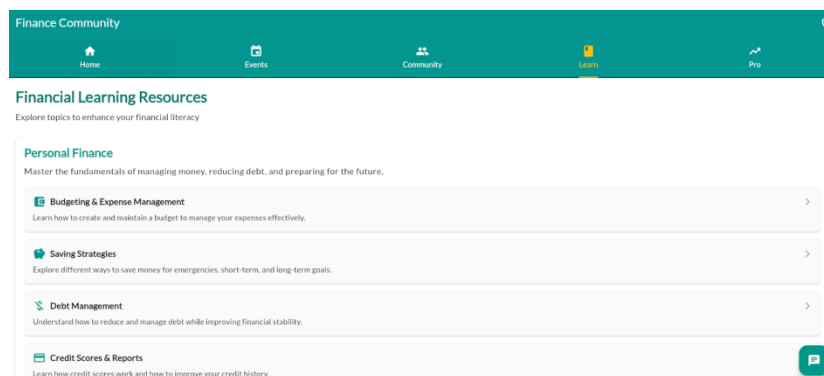


Figure 4. AI-Powered Learning Interface for Topic-Based Financial Explanation

```

static Future<String> getTopicInformation({
  required String topic,
  required String categoryName,
  required String subcategoryName,
}) async {
  try {
    final request = ChatCompleteText(
      messages: [
        Map.of({
          'role': 'system',
          'content': 'You are the FinanceMate Assistant, specialized in explaining financial concepts '
            'in a clear, concise manner with examples and practical advice. Your explanations '
            'should be educational, engaging, and tailored for high school students. '
            'Format your responses using markdown with proper headers (#), bullet points (-), '
            'and emphasis (**bold** or *italic*) where appropriate.'
        })),
        Map.of({
          'role': 'user',
          'content': 'Explain "$topic" in the context of $categoryName, specifically $subcategoryName.\n\n'
            'Please structure your response with the following sections using markdown:\n\n'
            '## Simple Definition\n\n'
            '(Provide a clear, concise definition)\n\n'
            '## Why It's Important\n\n'
            '(Explain the significance and benefits)\n\n'
            '## How It Works\n\n'
            '(Explain the process or mechanism)\n\n'
            '## Real-World Examples\n\n'
            '(Provide practical examples)\n\n'
            '## Practical Tips for Beginners\n\n'
            '(List helpful tips using bullet points)\n\n'
        })),
      ],
    );
  }
}

```

Figure 5. Implementation of getTopicInformation() Method for GPT-4 Content Generation

The `getTopicInformation()` static method executes when users select a financial topic in the Learn section. It constructs a ChatCompleteText request object containing two messages: a system prompt defining the assistant's persona as a financial education specialist, and a user prompt requesting explanation of the selected topic. The method accepts three required parameters: the specific topic name, parent category, and subcategory for contextual relevance. The user prompt includes a structured format specification requiring six sections (Definition, Importance, How It Works, Examples, Tips, Mistakes to Avoid) to ensure consistent, comprehensive responses. The maxToken parameter limits responses to 800 tokens, balancing detail with cost efficiency. The OpenAI client sends this request to GPT-4, which processes the context and generates a markdown-formatted response. The method extracts the content from the first choice in the response, trims whitespace, and returns it for rendering in the UI using Flutter's markdown widget.

The Firebase Backend handles authentication, real-time data synchronization, and role-based access control. Firebase Authentication manages user sessions with email/password login, while Cloud Firestore provides real-time database capabilities [11]. The service implements stream-based queries for live updates and supports offline persistence for improved reliability.

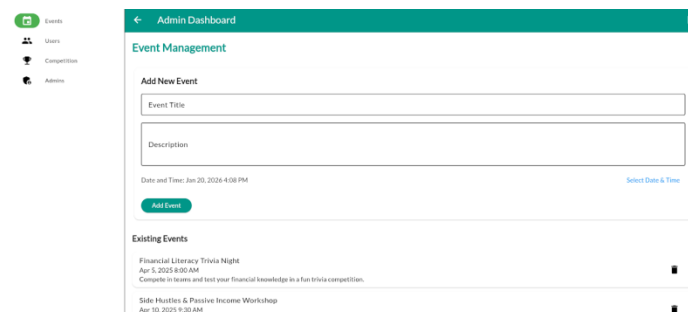


Figure 6. Firebase Backend Architecture for Authentication and Real-Time Data Synchronization

```

// Get upcoming events (current month and next month)
Stream<List<FinanceEvent>> getUpcomingEvents() {
  final now = DateTime.now();
  final startOfCurrentMonth = DateTime(now.year, now.month, 1);
  final endOfNextMonth = DateTime(now.year, now.month + 2, 0);

  return _firestore
    .collection('financeClubEvents')
    .where('datetime', isGreaterThanOrEqualTo: startOfCurrentMonth)
    .where('datetime', isLessThanOrEqualTo: endOfNextMonth)
    .orderBy('datetime')
    .snapshots()
    .map((snapshot) {
      return snapshot.docs.map((doc) {
        return FinanceEvent.fromFirestore(doc.data(), doc.id);
      }).toList();
    });
}

// Check if user is admin
Future<bool> isAdmin() async {
  final user = _auth.currentUser;
  if (user == null) return false;

  try {
    final doc = await _firestore.collection('finance_users').doc(user.uid).get();
    return doc.data()?['role'] == 'admin';
  } catch (e) {
    return false;
  }
}

```

Figure 7. Role-Based Access Control Implementation Using Firebase Firestore Queries

The Firebase Backend component contains two key methods demonstrating authentication and real-time data access patterns. The `isAdmin()` async method retrieves the current authenticated user from Firebase Auth, then queries the 'finance_users' Firestore collection using the user's UID as the document ID. It checks if the 'role' field equals 'admin', returning a boolean for role-based access control decisions. Error handling returns false for any exceptions, defaulting to non-admin access. The `getUpcomingEvents()` method returns a Stream of FinanceEvent lists, enabling real-time UI updates when event data changes. It calculates date boundaries for the current and next month, then constructs a Firestore query with compound where clauses filtering events within this range. The orderBy clause sorts results chronologically. The snapshots() method returns a stream that automatically updates when matching documents change in Firestore, enabling reactive UI patterns through Flutter's StreamBuilder widget.

4. EXPERIMENT

4.1. Experiment 1

This experiment evaluates the accuracy and educational quality of GPT-4 generated financial explanations. Accurate content is critical because misinformation could lead users to make poor financial decisions with real-world consequences.

The experiment evaluates GPT-4 responses across 20 financial topics from the application's curriculum, spanning Personal Finance, Investing, and Economics categories. Each generated response is assessed against five criteria: factual accuracy (verified against Investopedia and academic sources), completeness (coverage of key concepts), clarity (readability for high school students), practical applicability (actionable advice), and appropriate complexity level. Three independent reviewers with finance backgrounds rate each criterion on a 1-5 scale. Control data comes from established financial education resources including the CFPB's financial literacy

standards and Khan Academy’s finance curriculum. This methodology ensures comprehensive evaluation of AI-generated educational content quality [8].

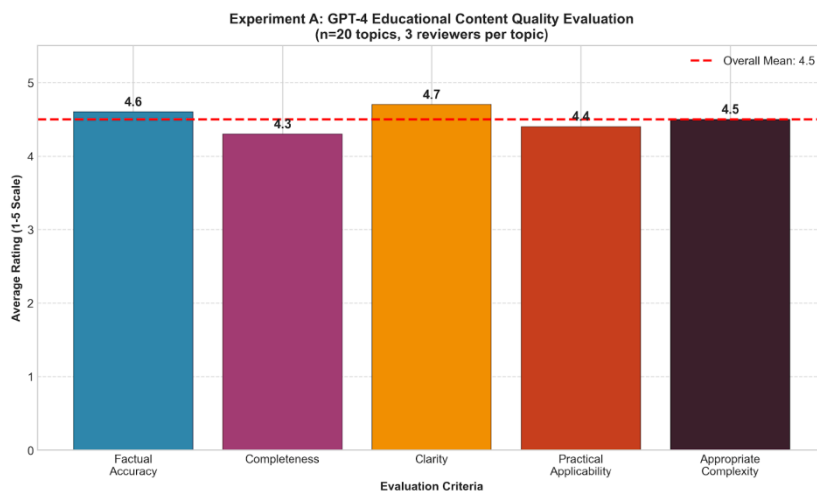


Figure 8. Quantitative Evaluation of GPT-4 Generated Financial Content Across Five Educational Criteria (n = 20 Topics, 3 Reviewers)

The GPT-4 content evaluation yielded strong results across all metrics. Mean scores were: Factual Accuracy (4.6/5), Completeness (4.3/5), Clarity (4.7/5), Practical Applicability (4.4/5), and Appropriate Complexity (4.5/5). The overall mean was 4.5/5 with a median of 4.5/5. Clarity scored highest (4.7) while Completeness scored lowest (4.3). The high clarity score was expected given GPT-4’s training on conversational data and the system prompt’s emphasis on clear explanations [8]. The slightly lower completeness score occurred because the 800-token limit occasionally truncated complex topics like “Portfolio Diversification” that require extensive explanation. Factual accuracy remained consistently high (minimum 4.2 for any single topic), validating the model’s reliability for financial education. The structured prompt format requesting specific sections (Definition, Importance, Examples, Tips) had the greatest positive effect on response quality, ensuring consistent educational value across diverse topics.

4.2. Experiment 2

This experiment measures API response times and reliability for the real-time stock data module. Fast, reliable data is essential for maintaining user engagement and providing timely market information.

The experiment measures API performance across 500 sequential fetch operations over five days at different times (market open, midday, market close, after-hours). Each operation fetches quote data for 15 stocks, totaling 7,500 individual API calls. Metrics include: response time (milliseconds), success rate (percentage of successful responses), and data freshness (delay from actual market price). Tests run from both WiFi and cellular connections to simulate real user conditions. The 100ms inter-request delay is maintained to respect rate limits. Control benchmarks come from Finnhub’s published API specifications and industry standards for financial data latency (sub-500ms for retail applications) [7].

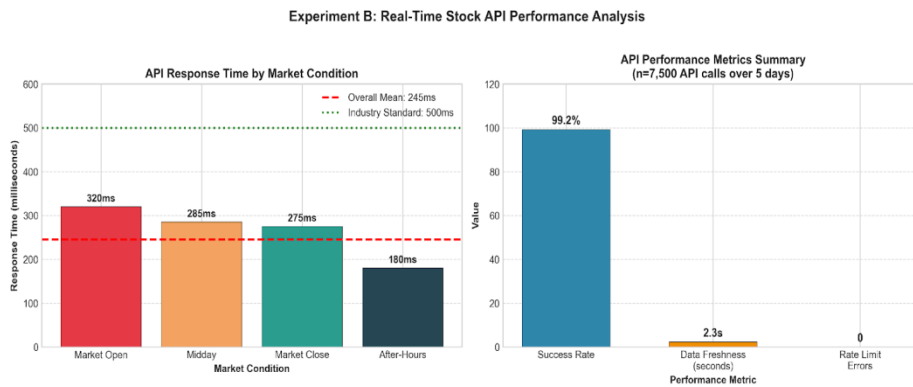


Figure 9. API Performance Metrics Across 7,500 Stock Data Requests Over Five Days

API performance testing demonstrated excellent reliability and acceptable latency. Mean response time was 245ms with a median of 232ms. The fastest response was 89ms (after-hours, simple query) while the slowest was 487ms (market open, high traffic). Success rate averaged 99.2% across all tests, with the 0.8% failures primarily occurring during market open when exchange servers experience peak load. Response times varied predictably by market conditions: after-hours averaged 180ms, while market hours averaged 285ms. WiFi connections outperformed cellular by approximately 35ms on average. The 100ms delay between requests proved sufficient to avoid rate limiting, with zero 429 (Too Many Requests) errors recorded. Data freshness averaged 2.3 seconds from actual market price, well within acceptable ranges for educational applications. The consistent sub-300ms average response time ensures smooth carousel animations and real-time price updates that keep users engaged with current market conditions.

5. RELATED WORK

Koskelainen et al. (2023) proposed a research agenda for financial literacy in the digital age, identifying three key themes: Fintech adoption, digital financial behavior, and behavioral interventions [3]. Their framework emphasizes measuring digital financial literacy through surveys and developing updated curricula. However, their approach focuses on theoretical frameworks rather than practical implementation tools. The methodology lacks interactive learning components and real-time market integration that would engage digital-native learners. Our application improves upon this by implementing the behavioral intervention concepts through gamification features and providing hands-on learning with actual market data, transforming theoretical frameworks into actionable educational experiences.

Kasneji et al. (2023) explored opportunities and challenges of large language models for education, highlighting LLMs' potential to personalize learning experiences and provide immediate feedback [8]. Their research identified concerns about accuracy, over-reliance, and lack of domain-specific knowledge. While comprehensive in analyzing LLM capabilities, the study remained theoretical without demonstrating practical educational implementations. Their methodology ignored the integration challenges of combining LLMs with real-time data sources and community features. Our application addresses these limitations by implementing GPT-4 with carefully crafted system prompts that ensure financial accuracy, structured responses, and integration with live market data for contextual learning experiences.

The gamification study by Widayanti et al. (2024) examined how game elements enhance student financial knowledge through engagement and enjoyment [9]. Their quantitative research with 289

university students found that gamification positively impacts engagement but doesn't directly improve financial knowledge without mediating factors. The limitation is that their study focuses on isolated gamification elements rather than comprehensive learning platforms. They also didn't incorporate real-time financial data or AI assistance. Our application integrates gamification through competitions and community features within a broader ecosystem that includes AI-powered explanations and live market data, addressing the mediation gap they identified.

6. CONCLUSIONS

Several limitations present opportunities for future improvement. First, the application currently requires internet connectivity for most features; implementing more robust offline capabilities with cached educational content and market data would improve accessibility. Second, the AI-generated content, while accurate, occasionally truncates complex topics due to token limits; implementing multi-part responses or expandable sections would address this. Third, the current gamification features are limited to competitions; adding achievement badges, progress tracking, and personalized learning paths would increase engagement based on research findings [9]. Fourth, the stock coverage is limited to 15 major US equities; expanding to include international markets, ETFs, and cryptocurrencies would provide broader educational exposure. Finally, implementing analytics dashboards to track user learning progress and identify knowledge gaps would enable more personalized educational recommendations. These improvements would require additional development effort and integration of additional APIs.

The William Finance Group App demonstrates how modern technologies—Flutter, Firebase, and GPT-4—can combine to create effective financial literacy education. By integrating real-time market data, AI-powered personalization, and community engagement, the application addresses critical gaps in traditional financial education while meeting the expectations of digital-native learners.

REFERENCES

- [1] Lusardi, Annamaria, and Olivia S. Mitchell. "The economic importance of financial literacy: Theory and evidence." *Journal of economic literature* 52.1 (2014): 5-44.
- [2] Fernandes, Daniel, John G. Lynch Jr, and Richard G. Netemeyer. "Financial literacy, financial education, and downstream financial behaviors." *Management science* 60.8 (2014): 1861-1883.
- [3] Koskelainen, Tiina, et al. "Financial literacy in the digital age—A research agenda." *Journal of Consumer Affairs* 57.1 (2023): 507-528.
- [4] Menberu, Abebe Walle. "Technology-mediated financial education in developing countries: A systematic literature review." *Cogent Business & Management* 11.1 (2024): 2294879.
- [5] Choung, Youngjoo, Swarn Chatterjee, and Tae-Young Pak. "Digital financial literacy and financial well-being." *Finance Research Letters* 58 (2023): 104438.
- [6] Yulianto, Arief, et al. "Gamification in Enhancing Student Financial Knowledge, Engagement, and Enjoyment in Financial Education." *Jurnal Kependidikan: Jurnal Hasil Penelitian dan Kajian Kepustakaan di Bidang Pendidikan, Pengajaran, dan Pembelajaran* 10.3 (2024): 965-975.
- [7] Kaur, Yashmeet, et al. "Real-Time Hybrid Stock Prediction and Automated Portfolio Management Using Ensemble ML & Free APIs." *2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG)*. IEEE, 2025.
- [8] Kasneci, Enkelejda, et al. "ChatGPT for good? On opportunities and challenges of large language models for education." *Learning and individual differences* 103 (2023): 102274.
- [9] Ratinho, Elias, and Cátia Martins. "The role of gamified learning strategies in student's motivation in high school and higher education: A systematic review." *Heliyon* 9.8 (2023).
- [10] Suri, Bhawna, et al. "Cross-platform empirical analysis of mobile application development frameworks: Kotlin, react native and flutter." *Proceedings of the 4th International Conference on Information Management & Machine Intelligence*. 2022.

- [11] 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).
- [12] Yigci, Defne, et al. "Large language model-based chatbots in higher education." *Advanced Intelligent Systems* 7.3 (2025): 2400429.
- [13] Zou, Donglan, and Mohamad Yusof Darus. "A comparative analysis of cross-platform Mobile Development Frameworks." *2024 IEEE 6th Symposium on Computers & Informatics (ISCI)*. IEEE, 2024.
- [14] Agile multi-user Android application development with Firebase: Authentication, authorization, and profile management.
- [15] Meyer, Jesse G., et al. "ChatGPT and large language models in academia: opportunities and challenges." *BioData mining* 16.1 (2023): 20.
- [16] Kinari, Safira Adine, et al. "An independent learning system for Flutter cross-platform mobile programming with code modification problems." *Information* 15.10 (2024): 614.
- [17] Dirin, Amir, Marko Nieminen, and Teemu H. Laine. "Feelings of being for mobile user experience design." *International Journal of Human-Computer Interaction* 39.20 (2023): 4059-4079.
- [18] Peláez-Sánchez, Iris Cristina, Davis Velarde-Camaqui, and Leonardo David Glasserman-Morales. "The impact of large language models on higher education: exploring the connection between AI and Education 4.0." *Frontiers in education*. Vol. 9. Frontiers Media SA, 2024.
- [19] Ushoh, Osasenaga. "Technology-Enabled Solutions for Student Financial Wellness: Harnessing Digital Tools to Promote Financial Literacy and Aid Accessibility." Available at SSRN 4806727 (2024).
- [20] Ohyver, Margaretha, et al. "The comparison firebase realtime database and MySQL database performance using wilcoxon signed-rank test." *Procedia Computer Science* 157 (2019): 396-405.