

# Strategic Architecture and Product Vision for FRN: The Common Ground Platform

## 1. Executive Vision: Engineering the Anti-Echo Chamber

The digital landscape of the mid-2020s is defined by a crisis of fragmentation. Social platforms, originally architected to connect, have inadvertently optimized for division through engagement algorithms that prioritize high-arousal emotions—typically outrage and polarization. The application "FRN" enters this ecosystem not merely as another social network, but as a socio-technical intervention designed to reverse-engineer the dynamics of online discourse. The core vision for FRN is to gamify the discovery of shared humanity amidst disagreement, transforming the adversarial "Zero-Sum" nature of internet debate into a "Non-Zero-Sum" coordination game where the objective is not to defeat the opponent, but to construct a shared reality.

To achieve this, FRN must transcend the traditional chat application model. It must function as a **Common Ground Engine**. Unlike platforms that match users based on superficial static attributes for romantic or platonic compatibility, FRN creates *dynamic pairings* based on the potential for constructive friction. The platform's architecture will enforce a "Rogerian" mode of interaction—where understanding precedes rebuttal—mediated by an active Artificial Intelligence that serves as a neutral third participant. This AI does not merely moderate; it actively contextualizes, pulling threads of shared identity (like a mutual love for specific cuisines or hobbies) from user profiles to weave a safety net under high-wire intellectual debates.

The user experience is designed to transition the user from a state of passive consumption (the "Feed") to active, structured engagement (the "Board"). By integrating real-time world data via trending topics, the app ensures that debates are grounded in the immediate reality of the zeitgeist, rather than abstract ideological battles. The ultimate vision is a platform where the "win state" is defined by the **Synthesis Score**—a metric reflecting the distance traveled from divergence to convergence.

---

## 2. Theoretical Frameworks: The Psychology of Digital Accord

The design of the "2-Column" interface and the overall interaction model must be rigorously grounded in psychological and game-theoretical principles. To simply split a screen is a UI decision; to split a screen to induce empathy is a psychological strategy.

## 2.1 Rogerian Argumentation as Interface Design

The prevailing mode of online debate is Aristotelian—adversarial, logic-focused, and aimed at victory. However, for FRN's goal of "finding common ground," the most relevant framework is **Rogerian Argumentation**, derived from the work of psychologist Carl Rogers.<sup>1</sup> Rogers argued that in situations of profound disagreement, the primary barrier to communication is the tendency to evaluate rather than listen. He proposed that a party should not be permitted to state their position until they have restated the opponent's position to the opponent's satisfaction.<sup>1</sup>

In the context of FRN's "2-Column" UI, this theoretical framework translates into specific constraints. The interface must actively discourage the "rebuttal mind"—the state where a user listens only to find flaws.<sup>2</sup> Instead of a continuous stream of text, the UI can enforce a "**Validation Gate**." Before User A can type in their column, they might be required to highlight a segment of User B's text and tag it with a semantic marker of understanding (e.g., "I see your point about X"). This turns the act of listening into a game mechanic, where "unlocking" the ability to speak is contingent on demonstrating comprehension. This moves the interaction from a "Winner/Loser" dynamic to a collaborative puzzle where mutual understanding is the victory condition.<sup>1</sup>

## 2.2 Game Theory: Shifting Payoff Matrices

Current social media debates function as **Zero-Sum Games**, where one user's status gain (via a "dunk" or witty retort) is the other's loss. To engineer common ground, FRN must restructure the interaction as a **Stag Hunt** or **Coordination Game**.<sup>3</sup> In a Stag Hunt, two hunters must cooperate to catch a high-value stag; if either defects to catch a low-value rabbit, the cooperation fails.

In FRN, the "Stag" is the discovery of a profound shared truth or a high "Synthesis Score." The "Rabbit" is the cheap dopamine hit of an insult. The UI must visualize this payoff matrix.<sup>3</sup> If the AI detects that both users are engaging constructively (high semantic overlap, neutral sentiment), a central "Common Ground" progress bar fills, unlocking special features or "Credibility Tokens." If one user defects (uses toxicity), the bar shatters, and the session ends with low rewards for both. This explicitly aligns the users' selfish incentives with the platform's goal of civility. The presence of a "dominant strategy" in bad conversations—where aggression wins—must be neutralized by algorithmic penalties for dominance and rewards for parity.<sup>3</sup>

## 2.3 Social Identity Theory and Context Injection

Henri Tajfel's **Social Identity Theory (SIT)** posits that individuals categorize themselves into "in-groups" and "out-groups" to enhance self-esteem, leading to inevitable bias against the out-group.<sup>4</sup> When two users debate a polarized topic (e.g., gun control), they immediately categorize each other as "Enemy."

FRN's "Profile Context Injection" feature is the antidote to this. By having the AI inject "already known profile stuff" [User Query], the system artificially constructs a **Cross-Cutting Identity**. If User A (Liberal) and User B (Conservative) are arguing, the AI interrupts the binary "Us vs. Them" dynamic by inserting a notification: "Context Alert: You both listed 'Sourdough Baking' as a primary hobby." This forces the users to re-categorize the opponent not just as "Political Enemy" but simultaneously as "Fellow Baker." Psychological research confirms that salience of cross-cutting identities reduces intergroup aggression.<sup>5</sup> The UI acts as a persistent reminder of the "complex self," preventing the reduction of the human opponent to a single, hated opinion.

---

### 3. User Experience Architecture: The Dual-Perspective Interface

The request specifies a need to "improve the 2 columns for finding common ground." This requires moving beyond a simple split-screen layout into a **Shared Workspace Metaphor**.

#### 3.1 The Dual-Column Board Layout

Standard chat apps use a single vertical timeline. This visually implies a singular narrative that one user dominates. FRN will use a **Split-Column Board**:

- **Column A (Left - Self):** The user's workspace.
- **Column B (Right - Other):** The partner's workspace.
- **The "Bridge" (Center Gutter):** A dynamic, AI-controlled active zone.

Visualizing Convergence:

Most split-screen UIs keep content static.<sup>7</sup> FRN's Bridge will be dynamic. When the AI analyzes the text in Column A and Column B and detects semantic similarity (using Vector Embeddings), it extracts the shared concept and places it physically in the Bridge as a "Synthesis Card."

- *Example:* User A types, "I want safer streets." User B types, "Security is paramount."
- *Action:* The AI extracts "Public Safety" and spawns a card in the center.
- *Interaction:* Both users can double-tap this card to "Lock" it. When locked, the card glows and expands, visually pushing the two columns closer together. This uses the Gestalt principle of **Common Fate**—elements moving together are perceived as related.<sup>8</sup>

## 3.2 The "Sort and Improve" Mechanic

The user asked to "sort" the columns. In a live debate, points are often made out of order. The FRN Board will allow **Post-Hoc Sorting**:

- **Drag-and-Drop Argumentation:** Users can drag a paragraph from their column and align it horizontally with a specific paragraph in the opponent's column. This visual alignment signifies "I am responding specifically to this point."
- **Threaded sorting:** The AI can auto-sort the columns not by time, but by **Topic Cluster**. If the debate ranges from "Economics" to "Ethics," the AI can regroup messages under these headers in real-time, cleaning up the messy linear flow of human speech into a structured "Balance Sheet" of the argument.

## 3.3 Gamification and "The Shuffler"

To support the "shuffler" requirement for profiles [User Query], the matching phase acts as a "Deck Building" phase.

- **The Random Persona Generator:** Before entering the debate, the user sees a "Profile Shuffler." This slot-machine style UI spins to generate a temporary, anonymized handle (e.g., "LogicalPanda," "StoicEagle").<sup>9</sup>
- **Why Anonymize?:** To reduce **Stereotype Threat** and initial bias. The user can hit "Shuffle" until they find a persona that feels right, giving them a sense of agency while maintaining the "veil of ignorance" necessary for unbiased debate.

---

# 4. Technical Architecture: Modernizing the Codebase

The existing repository vikram-sra/frn requires a significant architectural overhaul to support complex real-time AI states, dual-column synchronization, and bot polymorphism. We will adopt a **Clean Architecture** approach tailored for **React Native**, ensuring separation of concerns between the UI (Presentation), the Business Logic (Domain), and the Data Fetching (Data).

## 4.1 Architectural Pattern: Feature-Sliced Clean Architecture

The codebase should be refactored from a monolithic structure to a modular one.<sup>10</sup>

Proposed Folder Structure:

```
src/
  └── app/ # Entry Point & Global Config
    ├── App.tsx
    └── navigation/ # React Navigation Stacks (Auth, Main, Chat)
      └── di/ # Dependency Injection Container (InversifyJS or simple context)
  └── core/ # Shared Utilities & Kernel
```

```

|   └── theme/ # Colors, Typography, Spacing
|   └── constants/ # App-wide constants
|   └── errors/ # Error handling classes
|   └── data/ # Data Layer (The "How")
|       └── datasources/ # Raw data fetchers
|           └── api/ # OpenAI, Google Trends, NewsAPI clients
|           └── local/ # Realm/SQLite/AsyncStorage
|           └── sockets/ # WebSocket client for real-time chat
|       └── repositories/ # Concrete implementations of Domain Interfaces
|           └── ChatRepositoryImpl.ts
|           └── ProfileRepositoryImpl.ts
|       └── dtos/ # Data Transfer Objects (API responses)
|   └── domain/ # Business Logic Layer (The "What")
|       └── entities/ # Pure Types (User, Message, DebateSession)
|       └── repositories/ # Interfaces (IChatRepository, IProfileRepository)
|       └── usecases/ # Single Responsibility Actions
|           └── AnalyzeSentimentUseCase.ts
|           └── FindCommonGroundUseCase.ts
|           └── GenerateBotResponseUseCase.ts
|   └── presentation/ # UI Layer (The "Show")
|       └── components/ # Reusable Atoms/Molecules (Buttons, Inputs)
|       └── features/ # Feature-based Modules
|           └── chat/ # The "2-Column" Logic
|               └── screens/ # DualColumnBoard.tsx
|               └── components/ # BridgeCard.tsx, Column.tsx
|               └── state/ # Local State (Zustand/Bloc/Riverpod)
|               └── discovery/ # Trending Topics & Category Selection
|               └── profile/ # Profile Manager & Shuffler
|           └── hooks/ # Custom React Hooks (useDebateAI, useTrendFeed)
|       └── services/ # Background Services
|   └── BotOrchestrator.ts # "Wizard of Oz" Bot Logic

```

## 4.2 State Management Strategy

For an app with high-frequency updates (typing in two columns, real-time AI analysis), useState is insufficient.

- **Global State (User/Settings):** Use **Zustand** for its minimal boilerplate and ease of use compared to Redux.<sup>12</sup> It allows for transient updates without re-rendering the entire tree.
- **Server State (Trends/News):** Use **TanStack Query (React Query)**. This is critical for the "Trending Topics" feature, as it handles caching, background refetching, and error states automatically, preventing the app from freezing while fetching external data.
- **Complex UI State (The Debate Board):** Use a **State Machine** (XState or a simple Reducer). The debate has distinct states: Waiting, Drafting, Analyzing, BridgeUnlocked, BotTyping. Managing these explicitly prevents "impossible states" (e.g., bot typing while

user is disconnected).

## 4.3 SOLID Principles in Functional Components

To ensure maintainability:

- **Single Responsibility:** The ChatScreen should not calculate common ground. It should only render the data. The calculation happens in FindCommonGroundUseCase.
  - **Dependency Inversion:** The UI should not import OpenAI directly. It should consume an IAIService interface. This allows us to swap the real OpenAI service for a MockBotService during the prototyping phase (Point 7) without changing a single line of UI code.<sup>13</sup>
- 

# 5. The AI Engine: The "Third Participant"

The requirement for AI integration involves three distinct capabilities: **Moderation/Censoring**, **Context Injection**, and **Bot Simulation**.

## 5.1 Text Analysis and Censoring Stack

We must balance cost, latency, and accuracy.

- **Layer 1: On-Device Pre-Filtering (Latency < 50ms):** Integrate **TensorFlow Lite (TFLite)** with a mobile-optimized BERT model for toxicity detection.<sup>14</sup> This runs locally on the user's phone. If the user types a slur, the UI turns red immediately, preventing the network request. This is privacy-preserving and free.
- **Layer 2: Cloud Moderation (Deep Analysis):** For nuanced analysis, use the **OpenAI Moderation API**. It is free to use and detects categories like "Hate/Threatening," "Self-Harm," and "Sexual".<sup>15</sup>
  - *Integration:* Every message passes through an Edge Function that calls the Moderation API. If flagged, the message is blocked, and the user receives a "Cool Down" warning.

## 5.2 The Context Injection Engine (RAG Lite)

To satisfy the requirement of "AI reads your profile and keeps putting in known profile stuff," we need a lightweight **Retrieval-Augmented Generation (RAG)** system.

**Mechanism:**

1. **Profile Vectorization:** When a user creates a profile (interests: "Italian Food," "Sci-Fi"), these strings are converted into embeddings (numerical representations) and stored locally.
2. **Conversation Monitoring:** As the debate progresses, the AI buffers the last 5 messages.
3. **Semantic Search:** The AI checks the semantic distance between the *current debate topic* and the *user's profile vectors*.

4. **Injection Prompt:** If the debate gets heated (negative sentiment), the AI searches for convergent vectors (shared interests).
- *Trigger:* Sentiment Score < 0.3 (Negative).
  - *Action:* The AI injects a system message.
  - *Prompt to LLM:* "The debate is getting heated. User A likes [Pizza], User B likes [Pizza]. Generate a playful interruption reminding them of this shared ground."

**Data Table: Comparison of Moderation/Analysis Tools**

Tool	Capability	Latency	Cost	Best For
<b>TensorFlow Lite</b>	Basic Toxicity, Keyword Spotting	Ultra-Low (On-device)	Free	Instant UI feedback (Red glowing border)
<b>OpenAI Moderation</b>	Deep Safety Analysis	Medium (API call)	Free	Blocking harmful messages before delivery
<b>Google Perspective</b>	Nuanced Attribute Scores	Medium	Free (Quota)	Detailed "Toxicity Score" visualization
<b>VADER Sentiment</b>	Emotional Tone (Pos/Neg)	Low (Local Logic)	Free	"Temperature Bar" visualization

### 5.3 Bot Prototyping: The "Wizard of Oz" Strategy

The user requests a "Bot Dropdown" to override the random selector. This requires a **Strategy Pattern** for the chat participant.

- **Interface:** IChatParticipant
- **Implementation A:** RealUserParticipant (Connects to WebSocket).
- **Implementation B:** BotParticipant (Connects to OpenAI API).

The Bot Logic:

When the user selects "Contrarian Bot" from the dropdown, the system initializes BotParticipant with a specific System Prompt:

"You are a debate opponent. Your goal is to disagree with the user's premise: {{User\_Topic}}. However, you share the following interest with the user: {{User\_Shared\_Interest}}. Occasionally mention this to soften the blow. Keep responses under 200 characters."

---

## 6. Dynamic World State: Trending Topics Ingestion

The user explicitly requested replacing static lists with "new trending topics from the internet."

### 6.1 The Data Pipeline

Directly accessing Google Trends via client-side API is difficult due to CORS and lack of official API. We will use a **Serverless Aggregator** (e.g., Firebase Cloud Function) to fetch and sanitize data.<sup>16</sup>

**Sources:**

1. **Google Trends RSS:** <https://trends.google.com/trends/trendingsearches/daily/rss?geo={Location}>. This is the most reliable free source for real-time trends.<sup>18</sup>
2. **NewsAPI.org:** Used to enrich the trend with a headline and image. Google Trends gives the keyword; NewsAPI gives the context.<sup>19</sup>

### 6.2 The "Debate Prompt" Generator

Raw trends (e.g., "Taylor Swift") make bad debate topics. We need an AI transformation layer.

- **Input:** "Taylor Swift" + "NFL".
- **Process:** Pass to LLM (GPT-4o-mini).
- **Instruction:** "Turn this trend into a polarized debate question."
- **Output:** "Does celebrity coverage distract from the sport of football?"

This transformed question becomes the "Room Title."

---

## 7. Identity Management: Profile & Shuffler

### 7.1 Profile Page UI Specification

The profile page serves as the "Staging Area" for identity.

- **Layout:** Minimalist card centered on screen.
- **Username Field:** Read-Only text input.
  - **Action:** Beside it, a "Dice" icon button (The Shuffler).

- *Logic*: Tapping Dice calls a generator function: Adjective + Noun (e.g., "SkepticalBadger").
- **Read-Only Fields**: To signify immutability during the session, these fields (Username, ID) should have a distinct visual style (e.g., light gray background, lock icon) to distinguish them from editable fields like "Location" or "Categories".
- **Main Categories**: A multi-select "Chip" interface (Politics, Sports, Tech).

## 7.2 Random Username Algorithm

We will avoid external API calls for usernames to keep it snappy. We will use a local library or a custom dictionary array.

- **Dictionaries**: Adjectives (colors, emotions) and Animals/Objects.
  - **Mechanism**: Array.random() selection.<sup>20</sup>
  - **Collision Avoidance**: Append a 3-digit random number (e.g., "BlueFox#492").
- 

# 8. Implementation Plan: The Roadmap (plan.md)

Based on the research above, here is the detailed project plan for the FRN implementation.

## Phase 1: Core Architecture & Refactoring

- **Objective**: Stabilize the vikram-sra/frn repo and prepare for AI features.
- [ ] **Scaffold Clean Architecture**: Create src/domain, src/data, src/presentation directories.
- [ ] **State Management**: Install zustand for user state and @tanstack/react-query for fetching trends.
- [ ] **Navigation Flow**: Implement the stack: Profile -> Category/BotSelection -> ChatBoard.

## Phase 2: Identity & Profile System

- **Objective**: Implement the "Shuffler" and Profile Management.
- [ ] **Username Generator**: Create services/UsernameGenerator.ts using adjective/noun arrays.
- [ ] **Profile UI**: Build the Profile Screen with Read-Only fields (Username) and Editable fields (Interests, Location).
- [ ] **Shuffler Logic**: Wire the "Dice" button to the generator service.

## Phase 3: Dynamic Trends Engine

- **Objective**: Replace static lists with live world data.
- [ ] **RSS Service**: Create a utility to fetch Google Trends RSS (use a CORS proxy or Cloud Function).

- [ ] **AI Transformer**: Implement a simple prompt to convert Trends -> Debate Questions using OpenAI.
- [ ] **Feed UI**: Display these questions as swipeable cards in the CategoryScreen.

## Phase 4: Bot Ecosystem & Prototyping (Point 7)

- **Objective**: Enable "Wizard of Oz" testing with selectable bots.
- [ ] **Bot Dropdown**: Add a dropdown to CategoryScreen visible *only* in debug/prototype mode.
  - *Options*: Random Match, Contrarian Bot, Mediator Bot.
- [ ] **Flow Logic**:
  - *If User selects "Random Match"*: Navigate -> ShufflerScreen (Wait for human) -> ChatBoard.
  - *If User selects "Bot"*: Navigate -> ChatBoard immediately (Override Shuffler).
- [ ] **Bot Persona**: Implement BotParticipant class that sends the user's profile to the LLM to generate context-aware replies.

## Phase 5: The "Common Ground" Board (UI/UX)

- **Objective**: Build the 2-Column Interface with AI Mediation.
- [ ] **Split Layout**: Use Flexbox to create two equal vertical columns with a scrollable container.
- [ ] **The Bridge**: Create the center gutter component for "Common Ground" cards.
- [ ] **Context Toast**: Implement the UI overlay for "AI Context Injection" (e.g., "You both like Pizza!").
- [ ] **Text Analysis**: Wire up TensorFlow Lite for local toxicity checks and OpenAI for sentiment analysis.

## Phase 6: Polish & Analytics

- [ ] **Sentiment Bar**: Add a visual "Temperature" gauge at the top of the chat.
- [ ] **Gamification**: Add "Synthesis Points" animation when common ground is found.

## 9. Detailed User Flow Specification (Point 7 Focus)

To explicitly address the flow requirement: "*Combination of Category of user dropdown + Trending topic selected would show me shuffler screen or jump straight to cg board.*"

### State Diagram:

1. **Screen 1: Category & Topic Selection**
  - **Dropdown A (Category)**: User selects "Politics".
  - **Dropdown B (Bot/Mode)**:
    - *Option 1: "Match with Human" (Default)*

- Option 2: "Debate with Bot (Contrarian)"
  - **List (Trends):** User selects "Universal Basic Income".
  - **Action:** User taps "Start Debate".
2. **Logic Gate (The Router)**
- **Condition:** Check Dropdown B value.
  - **Path A (Human Match):** Navigate to **Screen 2: The Shuffler/Waiting Room.**
    - *UI:* "Searching for opponent..." + Random Username Spinner.
    - *Exit:* When match found -> Go to **Screen 3.**
  - **Path B (Bot Match):** Navigate directly to **Screen 3: The Common Ground Board.**
    - **Action:** Initialize BotParticipant with "Contrarian" system prompt.
    - **Context:** Pass "Universal Basic Income" and User Profile to Bot.
3. **Screen 3: The Common Ground Board**
- **Action:** Debate begins.
- 

## 10. Conclusion

This report establishes a robust foundation for transforming FRN into a market-leading "Common Ground" platform. By moving beyond simple chat mechanics to a **psychologically informed, AI-mediated architecture**, FRN can solve the "Context Collapse" problem plaguing modern social media. The technical roadmap prioritizes modularity through Clean Architecture, ensuring that the complex state requirements of real-time bot interaction and dynamic context injection are manageable. The proposed "Wizard of Oz" prototyping strategy allows for immediate testing of these high-value features without the need for an initial critical mass of users, de-risking the development process while validating the core value proposition: that technology can be used to bridge, rather than widen, the human divide.

### Works cited

1. "You Don't Seem Evil," Finding Common Ground Amid Profound Disagreement - Kim Scott, accessed January 9, 2026, <https://kimmalonescott.medium.com/you-dont-seem-evil-finding-common-ground-amid-profound-disagreement-5138faead372>
2. Principles of Dialogue and Reasoned Argument - Psychology Today, accessed January 9, 2026, <https://www.psychologytoday.com/us/blog/finding-common-ground/202511/principles-of-dialogue-and-reasoned-argument>
3. The Game Theory on Why Many Conversations Are Bad and Democracy Likely Doomed, accessed January 9, 2026, <https://o-g-rose-writing.medium.com/the-game-theory-on-why-many-conversations-are-bad-and-democracy-likely-doomed-c04fe80bc45d>
4. Social Identity Theory In Psychology (Tajfel & Turner, 1979), accessed January 9, 2026, <https://www.simplypsychology.org/social-identity-theory.html>
5. Social psychological debates about identity (Chapter 1) - Identity Process Theory,

- accessed January 9, 2026,  
<https://www.cambridge.org/core/books/identity-process-theory/social-psychological-debates-about-identity/6DE9B7225D3A5ECD244AE6DBAAA0B13F>
- 6. Social identity theory - Wikipedia, accessed January 9, 2026,  
[https://en.wikipedia.org/wiki/Social\\_identity\\_theory](https://en.wikipedia.org/wiki/Social_identity_theory)
  - 7. Best Practices for Split Screen Design | by Nick Babich - UX Planet, accessed January 9, 2026,  
<https://uxplanet.org/best-practices-for-split-screen-design-ad8507d92e66>
  - 8. Gestalt Principles for Visual UI Design - UX Tigers, accessed January 9, 2026,  
<https://www.uxtigers.com/post/gestalt-principles>
  - 9. ChristianGreinke/beautiful-username-generator - GitHub, accessed January 9, 2026, <https://github.com/ChristianGreinke/beautiful-username-generator>
  - 10. React Native project structure: a best practices guide - Tricentis, accessed January 9, 2026, <https://www.tricentis.com/learn/react-native-project-structure>
  - 11. React Native Clean Architecture — ResoCoder's way | by Mike Vas | Nerd For Tech, accessed January 9, 2026,  
<https://medium.com/nerd-for-tech/react-native-clean-architecture-resocoders-way-589c6e3d3fc2>
  - 12. Which architecture is best Bloc or Riverpod as a fresher ? : r/FlutterDev - Reddit, accessed January 9, 2026,  
[https://www.reddit.com/r/FlutterDev/comments/1oz92em/which\\_architecture\\_is\\_best\\_bloc\\_or\\_riverpod\\_as\\_a/](https://www.reddit.com/r/FlutterDev/comments/1oz92em/which_architecture_is_best_bloc_or_riverpod_as_a/)
  - 13. React & React Native: Mastering Design Patterns & SOLID Principles for Effective Software Development | Komodo Digital, accessed January 9, 2026,  
<https://www.komododigital.co.uk/insights/react-and-react-native-design-patterns-solid-principles/>
  - 14. Add on-device Text Classification to your app with TensorFlow Lite and Firebase - Android Codelab, accessed January 9, 2026,  
<https://firebase.google.com/codelabs/textclassification-android>
  - 15. What Does Moderation API in OpenAI Address Mean: Complete Guide to Content Safety in July 2025, accessed January 9, 2026,  
<https://www.cursor-ide.com/blog/openai-moderation-api-address-meaning>
  - 16. Introducing the Google Trends API (alpha): a new way to access Search Trends data, accessed January 9, 2026,  
<https://developers.google.com/search/blog/2025/07/trends-api>
  - 17. Google Trends API May Still Be The Wrong Tool For You - Exploding Topics, accessed January 9, 2026, <https://explodingtopics.com/blog/google-trends-api>
  - 18. Convert RSS to JSON API - ApyHub, accessed January 9, 2026,  
<https://apyhub.com/utility/converter-rss-json>
  - 19. News API – Search News and Blog Articles on the Web, accessed January 9, 2026, <https://newsapi.org/>
  - 20. Random Username Generator in SwiftUI | by Nayana N P - Medium, accessed January 9, 2026,  
<https://medium.com/@nayananp/random-username-generator-in-swiftui-7ac389780112>