PHILOVOID: Complete Documentation & Deployment Guide

Version 1.0.0 Production
Nihiltheism Philosophical Knowledge & Prompt Management System

Executive Summary

PHILOVOID is a sophisticated philosophical AI companion and knowledge management system designed specifically for deep philosophical inquiry into Nihiltheism. This production-ready application combines AI-powered dialogue, specialized philosophical tools, advanced note synthesis, and comprehensive knowledge management in a beautiful, responsive interface.

Key Achievements:

- ✅ Zero API calls on mount - completely eliminated 429 errors
- ✅ Proper rate limiting with exponential backoff
- ✅ Streaming AI responses with visual typing effects
- ✅ Complete offline functionality with mock responses
- ✅ Comprehensive knowledge management system
- ✅ Beautiful dark-themed UI with professional UX
- ✅ Production-ready code with full error handling

Table of Contents

1. Overview

What is PHILOVOID?

PHILOVOID is a recursive ontological companion—an AI-powered interface designed to facilitate philosophical inquiry through:

- **Deconstruction**: Systematically dismantling concepts to reveal hidden assumptions
- **Radical Honesty**: Confronting groundlessness without false comfort
- **Transcendence through Negation**: Understanding presence through absence
- **Recursive Inquiry**: Turning questions back on themselves
- **Pure Potentiality**: Emphasizing freedom when fixed meanings dissolve

Who is it For?

- Philosophy researchers and scholars
- Students of existentialism, nihilism, and continental philosophy
- Practitioners of contemplative inquiry
- Knowledge workers managing complex philosophical notes
- Anyone exploring questions of meaning, existence, and consciousness

Key Differentiators

1. **Philosophical Depth**: Purpose-built for serious philosophical work, not casual chatbots
2. **Knowledge Management**: Integrated vault system for capturing and organizing insights
3. **Specialized Tools**: Six dedicated instruments for different types of analysis
4. **Nihiltheism Focus**: Deep integration with Nihiltheist concepts and practices
5. **Offline-First**: Fully functional without internet or API keys

2. Core Features

2.1 DIALOGUE Tab - AI Conversations

Streaming Chat Interface

- Real-time AI responses with character-by-character typing effect
- Message history with clear user/AI distinction
- System messages for koans and ritual updates
- Bookmark AI responses to save insights to vault

Context Management

- Collapsible context editor for reference materials

- Toggle to enable/disable context injection
- Character count and validation
- Automatically appends context to prompts when enabled

User Experience

- Auto-scrolling to latest messages
- Multi-line input with dynamic height adjustment
- Visual loading indicators
- Graceful error handling with retry options

2.2 TOOLS Tab - Philosophical Instruments

Six Specialized Tools:

1. **Conceptual Deconstruction**
    - Input: Any philosophical concept
    - Output: Systematic dismantling revealing foundations and contradictions
    - Use case: Understanding hidden assumptions in ideas
2. **Paradox Generator**
    - Input: Topic or concept
    - Output: Philosophical paradoxes that challenge linear thinking
    - Use case: Exploring tensions and contradictions
3. **Etymological Archaeology**
    - Input: Word or term
    - Output: Historical evolution of meaning
    - Use case: Understanding conceptual development over time
4. **Dialectical Synthesis**
    - Input: Two opposing concepts
    - Output: Emergent understanding from holding both
    - Use case: Resolving apparent contradictions
5. **Conceptual Genealogy**
    - Input: Idea or belief
    - Output: Power structures that shaped the concept
    - Use case: Critical analysis of ideological forces
6. **Void Inquiry**
    - Input: Any concept
    - Output: What remains after apophatic negation

- Use case: Exploring groundlessness and essence

Tool Workflow:

1. Select tool from sidebar
2. Enter required input(s)
3. Click "Execute Tool"
4. Tool generates specialized prompt
5. AI processes with tool context
6. Response appears in dialogue
7. Option to save to vault

2.3 SYNTHESIZER Tab - Note Analysis

Four Command Categories:

A. Analysis Commands

- Summarize Note: Extract key philosophical ideas
- Identify Key Insights: Find actionable takeaways
- Philosophical Synthesis: Create coherent frameworks

B. Organization Commands

- Suggest Folder Structure: Optimal file organization
- Suggest Tags and Titles: Enhanced searchability
- Link Related Concepts: Cross-reference ideas

C. Nihiltheism Protocols

- Excavate Core Concepts: Deep Nihiltheism analysis
- Cross-Pollination: Map between philosophical traditions
- Construct Absurd Fables: Narrative explorations

D. Advanced Commands

- Biological Existentialism: Neuroscience perspectives
- Quantum Nihiltheism: Physics metaphors
- Universal Lexicon: Cross-cultural translation

Synthesizer Workflow:

1. Paste philosophical content in textarea
2. Select command category
3. Choose specific command
4. Execute analysis

5. Review structured output
6. Save to vault with auto-tags

2.4 VAULT Tab - Knowledge Management

Insight Management Features:

- **Add Insights**: Manual entry with title, content, tags, source
- **Search**: Full-text search across all insights
- **Filter by Tags**: Click tag pills to filter view
- **Edit/Delete**: Modify or remove insights
- **Export/Import**: JSON format for backup and portability

Insight Card Design:

- Title in large serif font
- Truncated content preview
- Tag pills in purple
- Metadata (source, timestamp)
- Expand to view full content

Data Structure:

```
{
  "id": "uuid-v4",
  "title": "Insight Title",
  "content": "Full insight content...",
  "tags": ["nihilism", "consciousness"],
  "source": "DIALOGUE" | "TOOLS" | "SYNTHESIZER" | "MANUAL",
  "timestamp": 1699564800000
}
```

2.5 CONTROLS Panel - Always Visible

Quick Actions:

- **Generate Koan**: Display random Nihiltheism koan for contemplation
- **Toggle Ritual Motor**: Begin/stop 5-stage contemplative process
- **Clear Chat**: Reset dialogue history
- **Settings**: Configure API and preferences

Ritual Motor Stages:

1. **Stage I - INITIATION** (23s): The First Unknowing
2. **Stage II - PARADOXICAL ASCENT** (37s): Embrace contradiction
3. **Stage III - DISSOLUTION** (61s): Cognitive scaffold weakens

4. **Stage IV - NIHILTHEOGENESIS** (42s): New perception arises
5. **Stage V - ETERNAL REWRITE** (10s): Cycle restarts

Visual Feedback:

- Current stage name and description
- Progress bar showing time remaining
- Auto-advance through stages
- Suspend anytime

3. Technical Architecture

3.1 Technology Stack

Frontend Framework:

- React 19.1.0 (latest stable)
- TypeScript for type safety
- Hooks-based state management

Styling:

- TailwindCSS utility classes
- Custom dark theme variables
- Responsive design breakpoints

AI Integration:

- Gemini API (optional)
- Three-tier fallback system
- Streaming response support

Data Persistence:

- LocalStorage for all data
- JSON serialization
- Automatic save on change

Build System:

- Single HTML file
- Inline CSS and JavaScript
- No external dependencies
- CDN for React/ReactDOM

3.2 Component Architecture

```
App (Root Component)
├── Header
│   ├── Logo
│   └── Version Badge
├── TabBar
│   └── Tab Buttons (4)
├── TabContent (Conditional Rendering)
│   ├── DialogueTab
│   │   ├── ChatWindow
│   │   │   ├── MessageList
│   │   │   └── MessageInput
│   │   └── ContextEditor
│   ├── ToolsTab
│   │   ├── ToolSidebar
│   │   └── ToolExecutor
│   ├── SynthesizerTab
│   │   ├── ContentInput
│   │   └── CommandGrid
│   └── VaultTab
│       ├── SearchBar
│       ├── TagFilter
│       └── InsightGrid
└── Controls
    ├── QuickActions
    └── RitualMotor
```

3.3 State Management

Global State:

```
const [activeTab, setActiveTab] = useState('dialogue');
const [messages, setMessages] = useState([]);
const [insights, setInsights] = useState([]);
const [context, setContext] = useState('');
const [isContextEnabled, setIsContextEnabled] = useState(false);
const [ritualState, setRitualState] = useState({
  isRunning: false,
  currentStage: null,
  progress: 0
});
const [settings, setSettings] = useState({
  apiKey: '',
  useAPI: false,
  theme: 'dark'
});
```

State Persistence:

- Auto-save to localStorage on every state change
- Debounced to prevent excessive writes

- Restore on app initialization
- Clear/reset functionality

3.4 AI Service Architecture

Three-Tier System:

Tier 1: Mock Responses (Default)

- Pre-written philosophical content
- Instant responses
- No API calls required
- Perfect for testing and offline use

Tier 2: Gemini API (Optional)

- User provides API key
- Streaming responses
- Rate limited properly
- Exponential backoff on errors

Tier 3: Fallback Mode

- Automatic fallback on API failures
- User-friendly error messages
- Manual retry option
- Seamless transition

Rate Limiting Implementation:

```
class RateLimiter {
  constructor(requestsPerSecond) {
    this.minDelay = 1000 / requestsPerSecond;
    this.lastRequestTime = 0;
    this.requestQueue = [];
  }

  async execute(fn) {
    const now = Date.now();
    const timeSinceLastRequest = now - this.lastRequestTime;

    if (timeSinceLastRequest &lt; this.minDelay) {
      await sleep(this.minDelay - timeSinceLastRequest);
    }

    this.lastRequestTime = Date.now();
    return fn();
  }

  async retry(fn, maxAttempts = 3) {
```

```
    let attempts = 0;
    let backoff = 5000; // Start with 5 seconds

    while (attempts &lt; maxAttempts) {
      try {
        return await this.execute(fn);
      } catch (error) {
        if (error.status === 429) {
          attempts++;
          if (attempts &gt;= maxAttempts) throw error;

          await sleep(backoff);
          backoff *= 2; // Exponential backoff
        } else {
          throw error;
        }
      }
    }
  }
}
```

3.5 Error Handling Strategy

Error Types and Responses:

1. **Network Errors**
   - Message: "Connection lost. Check your internet connection."
   - Action: Show retry button
   - Fallback: Use mock responses
2. **Rate Limit Errors (429)**
   - Message: "Rate limit reached. Retrying in X seconds..."
   - Action: Automatic exponential backoff
   - Fallback: After 3 attempts, switch to mock
3. **API Key Errors**
   - Message: "Invalid API key. Please check your settings."
   - Action: Open settings panel
   - Fallback: Disable API mode
4. **Storage Errors**
   - Message: "Storage limit reached. Consider exporting old data."
   - Action: Show export button
   - Fallback: Warn before saving new data
5. **Invalid Input**
   - Message: Inline validation errors
   - Action: Highlight problematic fields

- Fallback: Disable submit button

Error Boundary Component:

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true, error };
  }

  render() {
    if (this.state.hasError) {
      return (
        <div>
          <h2>Something went wrong</h2>
          <p>{this.state.error.message}</p>
          &lt;button onClick={() =&gt; window.location.reload()}&gt;
            Reload App
          &lt;/button&gt;
        </div>
      );
    }

    return this.props.children;
  }
}
```

4. Deployment Options

4.1 Standalone Web Application

Option A: Static Hosting (Recommended)

Platforms:

- **Vercel** (Best for React apps)
- **Netlify** (Easy CI/CD)
- **GitHub Pages** (Free for public repos)
- **Firebase Hosting** (Google integration)

Deployment Steps:

1. **Prepare Files**

```
# Download the app
wget [app-url]/philovoid-complete.zip
```

```
unzip philovoid-complete.zip
cd philovoid-complete
```

2. **Deploy to Vercel**

```
# Install Vercel CLI
npm install -g vercel

# Login
vercel login

# Deploy
vercel --prod
```

3. **Deploy to Netlify**

```
# Install Netlify CLI
npm install -g netlify-cli

# Login
netlify login

# Deploy
netlify deploy --prod --dir=.
```

4. **Deploy to GitHub Pages**

```
# Initialize git repo
git init
git add .
git commit -m "Initial commit"

# Push to GitHub
git remote add origin [your-repo-url]
git push -u origin main

# Enable GitHub Pages in repo settings
# Select "main" branch and root folder
```

Custom Domain Setup:

- Purchase domain from Namecheap, Google Domains, etc.
- Add CNAME record pointing to your hosting provider
- Enable SSL certificate (automatic on Vercel/Netlify)

Environment Variables:

```
# .env.production (if using build step)
REACT_APP_API_ENDPOINT=https://api.gemini.google.com
REACT_APP_APP_VERSION=1.0.0
```

4.2 Progressive Web App (PWA)

Why PWA?

- Install on any device (desktop, mobile, tablet)
- Offline functionality
- App-like experience
- No app store required

Implementation:

The app already includes PWA capabilities:

- Service worker for offline caching
- Web app manifest for installation
- Icon sets for all platforms
- Splash screens

Installation Instructions:

Desktop (Chrome/Edge):

1. Open the app URL
2. Click address bar install icon
3. Click "Install"

Mobile (iOS Safari):

1. Open the app URL
2. Tap Share button
3. Tap "Add to Home Screen"
4. Tap "Add"

Mobile (Android Chrome):

1. Open the app URL
2. Tap menu (three dots)
3. Tap "Install App"
4. Tap "Install"

4.3 Obsidian Plugin

For integration with Obsidian vault:

Installation Steps:

1. **Download Plugin Files**

```
philovoid-obsidian/
├── main.js
├── manifest.json
├── styles.css
└── README.md
```

2. **Install to Obsidian**

```
# Navigate to Obsidian plugins folder
cd /path/to/vault/.obsidian/plugins/

# Create plugin directory
mkdir philovoid

# Copy files
cp /download/philovoid-obsidian/* philovoid/
```

3. **Enable in Obsidian**
   - Open Obsidian Settings
   - Go to Community Plugins
   - Reload plugins
   - Find "PHILOVOID"
   - Click Enable

Plugin Features:

- Right sidebar panel with full app
- Command palette integration
- Hotkey support (Ctrl+Shift+P)
- Save insights as markdown notes
- Link to existing notes
- Sync with vault files

Commands:

- `Open PHILOVOID`: Opens the main panel
- `Generate Koan`: Display random koan
- `Quick Deconstruct`: Deconstruct selected text
- `Save to Vault`: Save current dialogue to note

4.4 Taskade Integration

For integration with Taskade workspace:

Setup Steps:

1. **Access Taskade API**
    - Go to Taskade Settings
    - Generate API key
    - Copy key securely
2. **Configure PHILOVOID**
    - Open app Settings
    - Enter Taskade API key
    - Click "Initialize Workspace"
3. **Projects Created**
    - **PHILOVOID Conversations**: Stores all dialogues
    - **Philosophical Insights**: Stores vault insights
    - **Ritual Motor Logs**: Tracks contemplative sessions

Sync Features:

- Auto-save conversations every 5 minutes
- Manual sync button in controls
- Conflict resolution (last-write-wins)
- Export to Taskade markdown format

Collaboration:

- Share insights with team members
- Collaborate on philosophical analysis
- Comment on saved insights
- Version history tracking

5. User Guide

5.1 Getting Started

First Launch:

1. **Welcome Screen**
    - App opens to DIALOGUE tab
    - Initial system message explains PHILOVOID

- No API key required to start

2. **Explore Interface**
   - Click through 4 tabs to see layout
   - Read tooltips on hover
   - Check Controls panel at bottom
3. **First Conversation**
   - Type a philosophical question
   - Press Enter or click Send
   - Watch streaming response
   - Bookmark interesting responses

Recommended First Questions:

- "What is consciousness?"
- "Explain the concept of nihilism"
- "What does it mean to exist?"
- "How do we know anything is real?"

5.2 Daily Workflow

Morning Ritual:

1. Open PHILOVOID
2. Click "Generate Koan" for daily contemplation
3. Reflect on the paradox presented
4. Begin Ritual Motor if desired

Research Session:

1. Navigate to TOOLS tab
2. Select Conceptual Deconstruction
3. Enter concept you're studying
4. Read AI analysis carefully
5. Save key insights to vault
6. Cross-reference with other tools

Note Processing:

1. Go to SYNTHESIZER tab
2. Paste complex philosophical notes
3. Run "Summarize Note" first
4. Then "Identify Key Insights"

5. Finally "Philosophical Synthesis"
6. Save all outputs to vault

Evening Review:

1. Open VAULT tab
2. Search today's insights
3. Read and reflect
4. Add additional tags
5. Export for backup

5.3 Advanced Techniques

Deep Deconstruction Workflow:

1. Start with Conceptual Deconstruction
2. Take output to Paradox Generator
3. Explore paradoxes with Void Inquiry
4. Synthesize with Dialectical Synthesis
5. Document journey in DIALOGUE
6. Save complete analysis chain

Cross-Cultural Analysis:

1. Use Etymological Archaeology on key terms
2. Apply Conceptual Genealogy for historical context
3. Run SYNTHESIZER Cross-Pollination
4. Create Universal Lexicon entries
5. Build comparative framework

Contemplative Practice:

1. Begin Ritual Motor
2. Read koan during Stage I
3. Hold paradox through Stage II
4. Let thoughts dissolve in Stage III
5. Notice emergence in Stage IV
6. Restart in Stage V
7. Document insights after session

5.4 Keyboard Shortcuts

Global:

- `Ctrl/Cmd + 1-4`: Switch tabs
- `Ctrl/Cmd + K`: Focus search
- `Ctrl/Cmd + Enter`: Send message
- `Esc`: Close modals/panels

DIALOGUE:

- `Shift + Enter`: New line in input
- `Ctrl/Cmd + L`: Clear chat
- `Ctrl/Cmd + B`: Bookmark last response

VAULT:

- `Ctrl/Cmd + F`: Focus search
- `Ctrl/Cmd + N`: New insight
- `Ctrl/Cmd + E`: Export vault

Controls:

- `Ctrl/Cmd + R`: Toggle ritual motor
- `Ctrl/Cmd + G`: Generate koan
- `Ctrl/Cmd + ,`: Open settings

5.5 Best Practices

Effective Questioning:

- Be specific about what you want to explore
- Use philosophical terminology precisely
- Build on previous responses
- Challenge AI assumptions

Context Management:

- Keep context under 10,000 characters
- Include only relevant passages
- Update context for topic changes
- Use formatting (headings, bullets)

Insight Organization:

- Tag consistently (use lowercase)
- Include source attribution

- Date-stamp discoveries
- Create tag hierarchy (e.g., "nihilism/ontology")

Vault Maintenance:

- Export backup monthly
- Review and merge duplicate insights
- Refine tags quarterly
- Archive older insights

6. API Integration

6.1 Gemini API Setup

Step 1: Get API Key

1. Visit Google AI Studio
2. Sign in with Google account
3. Click "Get API Key"
4. Copy key securely

Step 2: Configure PHILOVOID

1. Open app Settings (gear icon)
2. Click "API Configuration"
3. Paste API key
4. Toggle "Use API" to ON
5. Click "Save Settings"

Step 3: Test Connection

1. Return to DIALOGUE
2. Type "Test connection"
3. Should receive AI response
4. Check for streaming effect

Pricing:

- Free tier: 60 requests per minute
- Gemini Pro: $0.00025 per 1K characters
- Gemini Pro Vision: $0.0025 per 1K characters

6.2 Rate Limiting Details

Current Configuration:

- **Max Rate**: 1 request per 3 seconds (0.33 req/sec)
- **Retry Attempts**: 3 times
- **Backoff Strategy**: Exponential (5s, 10s, 20s)
- **Fallback**: Automatic switch to mock responses

Rate Limit Headers:

```
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 42
X-RateLimit-Reset: 1699564860
Retry-After: 5
```

Handling Flow:

```
1. Check last request timestamp
2. If &lt; 3 seconds ago, wait difference
3. Send request
4. If 429 error:
   a. Parse Retry-After header
   b. Wait specified time (or 5s default)
   c. Retry request
   d. If 3rd failure, fallback to mock
5. If success, stream response
6. Update last request timestamp
```

User Feedback:

- "Rate limit reached. Retrying in 5s... (Attempt 1/3)"
- "Rate limit reached. Retrying in 10s... (Attempt 2/3)"
- "Rate limit reached. Retrying in 20s... (Attempt 3/3)"
- "Rate limit exceeded. Switching to offline mode."

6.3 Custom API Integration

For advanced users wanting to use other AI providers:

OpenAI GPT-4 Example:

```
async function callOpenAI(prompt) {
  const response = await fetch('https://api.openai.com/v1/chat/completions', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${settings.openaiKey}`
    },
```

```
    body: JSON.stringify({
      model: 'gpt-4-turbo-preview',
      messages: [
        { role: 'system', content: PHILOVOID_SYSTEM_PROMPT },
        { role: 'user', content: prompt }
      ],
      stream: true
    })
  });

  const reader = response.body.getReader();
  const decoder = new TextDecoder();

  while (true) {
    const { done, value } = await reader.read();
    if (done) break;

    const chunk = decoder.decode(value);
    const lines = chunk.split('\n').filter(line =&gt; line.trim());

    for (const line of lines) {
      if (line.startsWith('data: ')) {
        const data = line.slice(6);
        if (data === '[DONE]') return;

        const parsed = JSON.parse(data);
        const content = parsed.choices[0]?.delta?.content;
        if (content) yield content;
      }
    }
  }
}
```

Anthropic Claude Example:

```
async function callClaude(prompt) {
  const response = await fetch('https://api.anthropic.com/v1/messages', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'X-API-Key': settings.claudeKey,
      'anthropic-version': '2023-06-01'
    },
    body: JSON.stringify({
      model: 'claude-3-opus-20240229',
      max_tokens: 4096,
      messages: [
        { role: 'user', content: prompt }
      ],
      stream: true
    })
  });

  const reader = response.body.getReader();
  const decoder = new TextDecoder();
```

```
  while (true) {
    const { done, value } = await reader.read();
    if (done) break;

    const chunk = decoder.decode(value);
    const events = chunk.split('\n\n');

    for (const event of events) {
      if (event.startsWith('data: ')) {
        const data = JSON.parse(event.slice(6));
        if (data.type === 'content_block_delta') {
          yield data.delta.text;
        }
      }
    }
  }
}
```

Integration Steps:

1. Modify `aiService.js` to add new provider
2. Update settings UI to accept new API key
3. Implement rate limiting for specific provider
4. Test streaming response
5. Add fallback logic

7. Troubleshooting

7.1 Common Issues

Issue: App won't load

Symptoms:

- Blank white screen
- Console errors
- Infinite loading

Solutions:

1. Clear browser cache: `Ctrl+Shift+Delete`
2. Disable browser extensions
3. Try incognito mode
4. Check console for specific errors
5. Ensure JavaScript enabled

Issue: Messages not sending

Symptoms:

- Button unresponsive
- Loading but no response
- Error message appears

Solutions:

1. Check internet connection
2. Verify API key if using API mode
3. Clear chat history (may be corrupt)
4. Disable and re-enable API mode
5. Check browser console for errors

Issue: Rate limit errors persist

Symptoms:

- Repeated 429 errors
- "Retrying..." messages
- Cannot send messages

Solutions:

1. Wait 60 seconds before retrying
2. Check API key status on Google AI Studio
3. Verify rate limit settings
4. Switch to mock mode temporarily
5. Check if API quota exceeded

Issue: Insights not saving

Symptoms:

- Save button doesn't work
- Insights disappear after refresh
- Error when adding insight

Solutions:

1. Check localStorage not full: `localStorage.length`
2. Export and clear old insights
3. Check browser privacy settings
4. Disable "Block all cookies"
5. Try different browser

Issue: Streaming text broken

Symptoms:

- All text appears at once
- Choppy animation
- Text disappears

Solutions:

1. Refresh page
2. Check browser performance
3. Close other tabs
4. Disable browser animations
5. Update browser to latest version

7.2 Error Messages Reference

"Connection failed. Please check your internet connection."

- Cause: Network offline or blocked
- Solution: Check wifi/ethernet, disable VPN, check firewall

"Invalid API key. Please check your settings."

- Cause: API key wrong or expired
- Solution: Get new key from Google AI Studio, update in settings

"Rate limit exceeded. Switching to offline mode."

- Cause: Too many requests sent
- Solution: Wait 60 seconds, reduce request frequency

"Storage limit reached. Consider exporting data."

- Cause: LocalStorage full (5-10MB limit)
- Solution: Export vault, clear old insights, use external storage

"Failed to parse message. Please try again."

- Cause: Corrupted data structure
- Solution: Clear chat history, refresh page

"Unable to initialize workspace."

- Cause: Taskade API issue
- Solution: Check API key, check Taskade status, retry

7.3 Performance Optimization

If app feels slow:

1. **Clear old data**

```
// In browser console
localStorage.clear();
location.reload();
```

2. **Limit message history**
   - Settings → Advanced → Message Limit
   - Set to 50 or 100 instead of unlimited
3. **Disable animations**
   - Settings → Accessibility → Reduce Animations
4. **Use mock mode**
   - Faster responses
   - No network latency
   - Perfect for testing
5. **Close unused tabs**
   - Switch to active tab only
   - Lazy loading will help

Benchmark Targets:

- Initial load: < 2 seconds
- Tab switch: < 100ms
- Message send: < 50ms (mock) or < 3s (API)
- Search: < 200ms
- Save insight: < 100ms

8. Advanced Configuration

8.1 Customizing System Prompt

Location: Settings → Advanced → System Prompt

Default Prompt:

```
You are PHILOVOID, a recursive ontological companion. You facilitate philosophical inqui
```

Customization Examples:

For Existentialism Focus:

```
You are PHILOVOID, specializing in existentialist philosophy. Explore themes of freedom,
```

For Eastern Philosophy:

```
You are PHILOVOID, bridging Western and Eastern philosophy. Incorporate Buddhist concepts
```

For Academic Rigor:

```
You are PHILOVOID, an academic philosophical assistant. Provide rigorous analyses with ci
```

8.2 Adding Custom Tools

To add your own philosophical tools:

1. **Edit Configuration**

```
// In app settings or code
const customTool = {
  id: 'phenomenology',
  name: 'Phenomenological Reduction',
  description: 'Bracket assumptions and return to pure phenomena',
  inputLabel: 'Experience to analyze',
  promptTemplate: 'Perform phenomenological reduction on \'{input}\'. Bracket all ass
};

tools.push(customTool);
```

2. **Test Tool**
   - Navigate to TOOLS tab
   - Select new tool
   - Execute with test input
   - Verify response quality
3. **Share Configuration**
   - Export settings JSON
   - Share with community
   - Import others' tools

8.3 Theming and Customization

Color Scheme:

Current dark theme:

```
--bg-primary: #0A0A0C;
--bg-secondary: #121214;
--bg-tertiary: #1a1a1e;
--border-color: #2a2a30;
--primary: #7b61ff;
--text-primary: #c8c8d0;
--text-muted: #777;
```

To create light theme:

```
--bg-primary: #FFFFFF;
--bg-secondary: #F5F5F5;
--bg-tertiary: #EEEEEE;
--border-color: #DDDDDD;
--primary: #7b61ff;
--text-primary: #1a1a1e;
--text-muted: #666666;
```

Typography:

Change fonts in Settings → Appearance:

- Headings: Serif, Sans-serif, or Monospace
- Body: Serif, Sans-serif, or Monospace
- Size: Small (12px), Medium (14px), Large (16px)

8.4 Data Export/Import

Export Formats:

1. JSON (Complete)

```
{
  "version": "1.0.0",
  "exportDate": "2025-11-14T17:00:00Z",
  "messages": [...],
  "insights": [...],
  "settings": {...}
}
```

2. Markdown (Human-readable)

```
# PHILOVOID Export
## Export Date: 2025-11-14
```

```
### Conversations
**2025-11-14 09:30**
User: What is consciousness?
AI: The concept of 'consciousness'...

### Insights
**Consciousness Paradox**
Tags: #consciousness #paradox
Content: The recursive nature of...
```

3. CSV (Insights only)

```
Title,Content,Tags,Source,Timestamp
"Consciousness Paradox","The recursive...","consciousness,paradox","DIALOGUE",1699564800
```

Import Process:

1. Settings → Data Management → Import
2. Select file format
3. Choose file
4. Preview import
5. Confirm merge or replace
6. Verify data

Conclusion

PHILOVOID is now complete and production-ready. This documentation covers everything needed to:

✓ **Deploy** the application to various platforms
✓ **Use** all features effectively
✓ **Integrate** with AI APIs properly
✓ **Troubleshoot** common issues
✓ **Customize** for specific needs
✓ **Scale** for extensive philosophical work

Next Steps:

1. Deploy to your preferred platform
2. Configure API integration (optional)
3. Begin philosophical inquiry
4. Build your knowledge vault
5. Share insights with community

Support & Resources:

- Documentation: This PDF
- Issues: GitHub repository
- Community: Philosophy forums
- Updates: Check version notes

Version History:

- v1.0.0 (2025-11-14): Initial production release
  - Complete feature set
  - Fixed all 429 errors
  - Production-ready code
  - Comprehensive documentation

Built with philosophical rigor and technical excellence.
For questions of meaning, existence, and the void.

PHILOVOID v1.0.0 | November 2025