Chat API Hang Bug Fix

Problem Summary

The AI chatbot API endpoints (`/api/chat` and `/api/enhanced-chat`) were hanging indefinitely with no response, causing the frontend to wait 30+ seconds with no feedback.

Root Cause Analysis

Primary Issue: Missing `await` on Stream Writes

The critical bug was in the `processStreamingChat` function where the `sendSSE` helper function was **not awaiting** the `writer.write()` calls. This caused several problems:

1. **Race Conditions**: Multiple writes to the stream could happen simultaneously without proper sequencing
2. **Backpressure Issues**: The stream's internal buffer could fill up, causing writes to block
3. **Silent Failures**: Errors during writes were not being caught or handled properly
4. **Stream Lock Issues**: The writer could be in an inconsistent state when errors occurred

Code Before Fix

```
const sendSSE = (data: any) => {
  writer.write(encoder.encode(`data: ${JSON.stringify(data)}\n\n`))
}

// Later in code:
sendSSE({ type: 'status', message: 'Searching documentation...' })  // NOT AWAITED!
```

Code After Fix

```
const sendSSE = async (data: any) => {
  try {
    await writer.write(encoder.encode(`data: ${JSON.stringify(data)}\n\n`))
  } catch (error) {
    console.error('[STREAMING] Failed to write to stream:', error)
    throw error
  }
}

// Later in code:
await sendSSE({ type: 'status', message: 'Searching documentation...' })  // PROPERLY
AWAITED!
```

Changes Made

1. Fixed Stream Write Operations (`src/app/api/chat/route.ts`)

Changed `sendSSE` to async function with proper error handling:
- Made `sendSSE` an `async` function
- Added `await` to `writer.write()` call

- Added try-catch block for write errors
- Added error logging for debugging

Updated all `sendSSE` calls to use `await`:
- Line 151: `await sendSSE({ type: 'status', message: 'Searching documentation...' })`
- Line 286: `await sendSSE({ type: 'status', message: 'Generating response...' })`
- Line 341: `await sendSSE({ type: 'content', content, done: false })`
- Line 359: `await sendSSE({ type: 'status', message: 'Executing tools...' })`
- Line 364: `await sendSSE({ type: 'tool_results', results: toolResults })`
- Line 377: `await sendSSE({ type: 'content', content: followUpResponse, done: true })`
- Line 404: `await sendSSE({ type: 'done', sessionId: sessionId || 'new' })`
- Line 411: `await sendSSE({ type: 'error', error: ... })`

2. Enhanced Error Handling

Added defensive error handling in `handleStreamingChat`:

```
.catch(error => {
  console.error('[HANDLE_STREAMING] Streaming chat error:', error)
  try {
    writer.write(encoder.encode(`data: ${JSON.stringify({
      type: 'error',
      error: error.message
    })}\n\n`))
  } catch (writeError) {
    console.error('[HANDLE_STREAMING] Failed to write error to stream:', writeError)
  }
})
.finally(() => {
  console.log('[HANDLE_STREAMING] Closing writer')
  try {
    writer.close()
  } catch (closeError) {
    console.error('[HANDLE_STREAMING] Failed to close writer:', closeError)
  }
})
```

3. Added Comprehensive Logging

Added detailed console logging at every step to help diagnose issues:
- Request receipt and parsing
- Stream creation and initialization
- Document search operations
- Operation log retrieval
- Tool availability checks
- Ollama API calls
- Stream reading and writing
- Error conditions

4. Enhanced Chat Route (`src/app/api/enhanced-chat/route.ts`)

Added similar logging improvements for consistency and debugging.

Why This Fix Works

1. Proper Async Flow

By awaiting all stream writes, we ensure:
- Operations complete in the correct order
- Backpressure is properly handled
- The stream buffer doesn’t overflow
- Errors are caught and handled appropriately

2. Error Visibility

With comprehensive logging, we can now:
- Track exactly where the execution flow is
- Identify which operation is causing delays
- Catch and log errors that were previously silent
- Debug issues in production

3. Defensive Programming

Try-catch blocks around critical operations prevent:
- Unhandled promise rejections
- Stream corruption
- Silent failures
- Cascading errors

Testing Recommendations

1. **Basic Chat Test**
   ```
   bash
   curl -X POST http://localhost:3000/api/chat \
     -H "Content-Type: application/json" \
     -d '{"message": "Hello", "stream": false}'
   ```

2. **Streaming Chat Test**
   ```
   bash
   curl -X POST http://localhost:3000/api/chat \
     -H "Content-Type: application/json" \
     -d '{"message": "Hello", "stream": true}'
   ```

3. **Enhanced Chat Test**
   ```
   bash
   curl -X POST http://localhost:3000/api/enhanced-chat \
     -H "Content-Type: application/json" \
     -d '{"message": "Hello"}'
   ```

4. **Monitor Logs**
   Check the console output for the detailed logging to verify:
   - All steps are executing
   - No errors are occurring
   - Response is being generated and streamed

Expected Behavior After Fix

1. **Immediate Response**: The API should start responding within 1-2 seconds
2. **Status Updates**: Client receives status messages during processing
3. **Streaming Content**: AI responses stream in real-time
4. **Proper Completion**: Stream closes cleanly with ‘done’ message
5. **Error Handling**: Any errors are caught and reported to the client

Additional Notes

- The fix maintains backward compatibility with non-streaming mode
- All existing functionality is preserved
- Performance should be improved due to proper async handling
- The logging can be reduced in production if needed

Related Files Modified

1. `src/app/api/chat/route.ts` - Main chat API with streaming support
2. `src/app/api/enhanced-chat/route.ts` - Enhanced chat API with logging
3. `package.json` / `package-lock.json` - Dependency updates (is-number)