# AG UI Protocol Implementation Plan for ZintelliX Architecture

## Current Architecture Overview

```
┌─────────────────────────────────────────────────────────────┐
│  Frontend: ZintelliX App (React)                            │
│  - CopilotKit UI Component                                  │
│  - Uses useAgent hook                                       │
└───────────────────┬─────────────────────────────────────────┘
                    │ HTTP POST /api/copilotkit
                    │ (with JWT token)
                    ↓
┌─────────────────────────────────────────────────────────────┐
│  Zintellix Backend (Node.js/Express)                       │
│  - routes/copilotkit.js                                     │
│  - HttpAgent from @ag-ui/client                            │
│  - Forwards requests to ZNAI backend                       │
└───────────────────┬─────────────────────────────────────────┘
                    │ HTTP POST {ZIN_AI_URL}/agent
                    │ (AG UI Protocol format)
                    ↓
┌─────────────────────────────────────────────────────────────┐
│  ZNAI Backend (FastAPI/LangGraph on AWS)                   │
│  - @app.post("/agent")                                      │
│  - Uses orchestrator.run()                                    │
│  - ❌ CURRENTLY: Returns JSON (WRONG)                      │
│  - ✅ NEEDED: SSE Streamning with Terminal Events           │
└─────────────────────────────────────────────────────────────┘
```

## Problem Statement

Current Issue: "Run ended without emitting a terminal event"

Root Cause:
- AG UI Protocol requires **Server-Sent Events (SSE)** streaming format
- Must emit `terminal` event to signal completion
- Current `/agent` endpoint returns simple JSON response

## Implementation Plan

~~~markdown
### Phase 1: Update ZNAI Backend (FastAPI) - CRITICAL

#### Step 1.1: Add Required Imports

**File**: `zinai.txt` (or your main FastAPI file)

**Location**: Top of file, with other imports

**Add**:
```python
from fastapi.responses import StreamingResponse  # Add this
import concurrent.futures  # Add this
import asyncio  # Ensure this is imported
```

**Current imports should include**:
```python
from fastapi import FastAPI, UploadFile, File, Form, HTTPException, Request
from fastapi.responses import JSONResponse
```

#### Step 1.2: Replace `/agent` Endpoint with Streaming Version

**File**: `zinai.txt`

**Location**: After your existing endpoints (around line 297), before `if __name__ ==
"__main__"`

**Replace entire `/agent` endpoint with**:

```python
@app.post("/agent")
async def agent_endpoint(request: Request):
   """
   CopilotKit HttpAgent endpoint with AG UI Protocol compliance
   Implements Server-Sent Events (SSE) streaming with terminal events
   """
   async def generate():
       try:
           # Parse request body
           body = await request.json()
~~~

```
            # Extract input from AG UI Protocol format
            messages = body.get("messages", [])
            state = body.get("state", {})

            # Extract user message
            user_message = None
            if messages:
                for msg in reversed(messages):
                    if msg.get("role") == "user":
                        user_message = msg.get("content", "")
                        break

            if not user_message:
                user_message = body.get("input", body.get("question", ""))

            if not user_message:
                # Emit error and terminal events
                yield f"data: {json.dumps({'type': 'error', 'error': 'No user message
found'})}\n\n"
                yield f"data: {json.dumps({'type': 'terminal'})}\n\n"
                return

            # Extract user_id (can be enhanced with JWT parsing)
            user_id = body.get("user_id", "default_user")

            # Extract from JWT token if available
            auth_header = request.headers.get("Authorization", "")
            if auth_header and auth_header.startswith("Bearer "):
                # TODO: Decode JWT to extract user_id
                # For now, use default
                pass

            # Validate user exists
            if not user_config_manager.user_exists(user_id):
                available_users = user_config_manager.get_available_users()
                user_id = available_users[0] if available_users else "user1"

            print(f"\n{'='*80}")
            print(f"🤖 AG UI PROTOCOL REQUEST")
            print(f"👤 User: {user_id}")
            print(f"❓ Question: {user_message}")
            print(f"{'='*80}\n")
```

```python
# Emit start event (AG UI Protocol requirement)
yield f"data: {json.dumps({'type': 'start'})}\n\n"

# Run orchestrator in executor (handle blocking call)
loop = asyncio.get_event_loop()
with concurrent.futures.ThreadPoolExecutor() as executor:
    result = await loop.run_in_executor(
        executor,
        lambda: orchestrator.run(
            user_id=user_id,
            question=user_message,
            file_path=None
        )
    )

# Convert to JSON-serializable format
serializable_result = make_json_serializable(result)

# Extract content from result
if isinstance(serializable_result, dict):
    content = (
        serializable_result.get("content") or
        serializable_result.get("answer") or
        serializable_result.get("output") or
        json.dumps(serializable_result)
    )
else:
    content = str(serializable_result)

# Ensure content is a string
if not isinstance(content, str):
    content = json.dumps(content)

# Emit data event with content (AG UI Protocol requirement)
yield f"data: {json.dumps({
    'type': 'data',
    'content': content,
    'messages': [{
        'role': 'assistant',
        'content': content
    }]
```

~~~~
            })}\n\n"

            # CRITICAL: Emit terminal event to signal completion
            # This is what was missing and causing the error!
            yield f"data: {json.dumps({
                'type': 'terminal',
                'state': {
                    'result': serializable_result
                }
            })}\n\n"

            print(f"✅ AG UI Protocol stream completed successfully")

        except Exception as e:
            print(f"❌ Error in agent endpoint: {e}")
            import traceback
            traceback.print_exc()

            # Emit error event
            yield f"data: {json.dumps({
                'type': 'error',
                'error': str(e)
            })}\n\n"

            # CRITICAL: Still emit terminal event even on error
            yield f"data: {json.dumps({'type': 'terminal'})}\n\n"

    # Return StreamingResponse with SSE format
    return StreamingResponse(
        generate(),
        media_type="text/event-stream",
        headers={
            "Cache-Control": "no-cache",
            "Connection": "keep-alive",
            "X-Accel-Buffering": "no",  # Disable buffering for nginx/proxy
        }
    )
```

#### Step 1.3: Verify Host Configuration

**File**: `zinai.txt`
~~~~

~~~
**Location**: Bottom of file, `uvicorn.run()` call

**Ensure**:
```python
uvicorn.run(
   "main:app",
   host="0.0.0.0",   # ✅ Must be 0.0.0.0 (not 127.0.0.1)
   port=8000,
   reload=True,
   log_level="info"
)
```

### Phase 2: Verify Zintellix Backend Configuration

#### Step 2.1: Check Environment Variable

**File**: `.env` (in Zintellix backend)

**Ensure**:
```env
ZIN_AI_URL=http://15.207.21.33:8000
# Or your actual ZNAI backend URL
```

#### Step 2.2: Verify CopilotKit Route

**File**: `routes/copilotkit.js`

**Current configuration is correct**:
- ✅ `HttpAgent` configured with `graphId: "agent"`
- ✅ JWT token forwarding implemented
- ✅ Route mounted at `/api/copilotkit`

**No changes needed** - this is already correct!

### Phase 3: Testing & Verification

#### Step 3.1: Test ZNAI Backend Directly

**Test 1: Basic Connectivity**
~~~

~~~
```bash
curl http://15.207.21.33:8000/agent
# Should connect (may return method not allowed for GET, but should connect)
```

**Test 2: POST Request with Streaming**
```bash
curl -X POST http://15.207.21.33:8000/agent \
 -H "Content-Type: application/json" \
 -H "Authorization: Bearer YOUR_JWT_TOKEN" \
 -N \
 -d '{
   "messages": [
     {"role": "user", "content": "Hello, test message"}
   ],
   "state": {}
 }'
```

**Expected Output** (SSE format):
```
data: {"type":"start"}

data: {"type":"data","content":"...","messages":[...]}

data: {"type":"terminal","state":{...}}
```

#### Step 3.2: Test Full Flow

1. **Start ZNAI Backend**:
  ```bash
  # On AWS instance
  python zinai.txt  # or however you run it
  ```

2. **Start Zintellix Backend**:
  ```bash
  # On your local machine
  npm start
  # or
  node index.js
~~~

~~~~markdown
  ```

3. **Test CopilotKit Info Endpoint**:
  ```bash
  curl http://localhost:3000/api/copilotkit/info
  # Should return: {"agents": ["agent"]}
  ```

4. **Test from Frontend**:
  - Open your ZintelliX frontend app
  - Navigate to ZinAI Workspace component
  - Send a message through CopilotKit UI
  - Check browser console for errors
  - Check backend logs for request flow

### Phase 4: Debugging & Troubleshooting

#### If Still Getting "Terminal Event" Error:

1. **Check ZNAI Backend Logs**:
  - Look for "🤖 AG UI PROTOCOL REQUEST" log
  - Verify "✅ AG UI Protocol stream completed successfully"
  - Check for any exceptions

2. **Verify SSE Format**:
  - Each event must be: `data: {json}\n\n`
  - Must have exactly two newlines (`\n\n`)
  - Terminal event must be emitted

3. **Check Network Tab**:
  - Open browser DevTools → Network
  - Find request to `/api/copilotkit`
  - Check response headers:
    - `Content-Type: text/event-stream`
    - `Cache-Control: no-cache`
    - `Connection: keep-alive`

4. **Verify Response Format**:
  - Response should be streaming (not single JSON)
  - Should see multiple `data:` lines
  - Last line should be terminal event
~~~~

~~~markdown
#### Common Issues:

**Issue 1: Still getting JSON response**
- **Cause**: Not using `StreamingResponse`
- **Fix**: Ensure using `StreamingResponse` with `text/event-stream`

**Issue 2: Terminal event not emitted**
- **Cause**: Exception before terminal event
- **Fix**: Ensure `yield terminal` is in `finally` block or always executed

**Issue 3: Connection refused**
- **Cause**: Host binding or firewall
- **Fix**: Ensure `host="0.0.0.0"` and AWS security group allows port 8000

**Issue 4: CORS errors**
- **Cause**: CORS not configured
- **Fix**: Already configured in ZNAI backend (line 28-34 in zinai.txt)

### Phase 5: Production Deployment

#### Step 5.1: Environment Variables

**Zintellix Backend `.env`**:
```env
ZIN_AI_URL=https://your-zna-backend-domain.com
# or
ZIN_AI_URL=http://your-aws-ip:8000
```

#### Step 5.2: Security Considerations

1. **JWT Validation**: Implement JWT validation in ZNAI backend `/agent` endpoint
2. **Rate Limiting**: Add rate limiting to `/agent` endpoint
3. **HTTPS**: Use HTTPS in production
4. **CORS**: Restrict CORS origins to your frontend domain

#### Step 5.3: Monitoring

1. **Logging**: Monitor logs for:
  - Request counts
  - Error rates
  - Response times
~~~

~~~
  - Terminal event emissions

2. **Alerts**: Set up alerts for:
  - Missing terminal events
  - High error rates
  - Slow response times

## Summary Checklist

### ZNAI Backend (FastAPI)
- [ ] Add `StreamingResponse` import
- [ ] Add `concurrent.futures` import
- [ ] Replace `/agent` endpoint with streaming version
- [ ] Verify `host="0.0.0.0"` in uvicorn.run()
- [ ] Test endpoint returns SSE format
- [ ] Verify terminal event is emitted

### Zintellix Backend (Node.js)
- [ ] Verify `ZIN_AI_URL` in `.env`
- [ ] Verify `routes/copilotkit.js` configuration
- [ ] Test `/api/copilotkit/info` endpoint
- [ ] Test full flow from frontend

### Frontend
- [ ] Verify CopilotKit provider configuration
- [ ] Verify `runtimeUrl` points to Zintellix backend
- [ ] Test agent interaction

## Expected Flow After Implementation

```
1. User types message in CopilotKit UI
  ↓
2. Frontend → POST /api/copilotkit (Zintellix backend)
  ↓
3. Zintellix backend → HttpAgent → POST {ZIN_AI_URL}/agent
  ↓
4. ZNAI backend → StreamingResponse with SSE:
  - data: {"type":"start"}
  - data: {"type":"data","content":"..."}
  - data: {"type":"terminal","state":{...}}
  ↓
~~~

~~~
5. HttpAgent receives stream → Forwards to CopilotKit runtime
  ↓
6. CopilotKit runtime → Frontend displays response
  ✅ Success!
```

## Key Points

1. **AG UI Protocol requires SSE streaming** - Not JSON responses
2. **Terminal event is mandatory** - Signals completion
3. **Event format**: `data: {json}\n\n` (exactly two newlines)
4. **Media type**: `text/event-stream`
5. **Headers**: `Cache-Control: no-cache`, `Connection: keep-alive`

## Next Steps

1. **Implement Phase 1** (ZNAI Backend changes)
2. **Test Phase 3** (Verification)
3. **Debug Phase 4** (If issues occur)
4. **Deploy Phase 5** (Production)

This plan addresses the root cause: **missing SSE streaming and terminal events** in
your ZNAI backend.
~~~