AI Adapter Fix Guide - Step by Step

🔧 What I Fixed

All Adapters Updated:

1. **OpenAI Adapter** (ChatGPT)
   - ✅ Added proper error checking
   - ✅ Added default model fallback
   - ✅ Added max_tokens config
   - ✅ Better response validation
2. **Anthropic Adapter** (Claude)
   - ✅ Fixed system message handling
   - ✅ Added proper model default
   - ✅ Better error messages
   - ✅ Response validation
3. **Gemini Adapter** (Google)
   - ✅ Fixed endpoint URL generation
   - ✅ Proper API key in query param
   - ✅ Better response parsing
   - ✅ Model name handling
4. **DeepSeek Adapter**
   - ✅ Added max_tokens config
   - ✅ Better error handling
   - ✅ Response validation
5. **aiProviders Service**
   - ✅ Better error logging
   - ✅ Detailed console output
   - ✅ Proper error propagation

🚀 How to Apply Fixes

Step 1: Update Database (Important!)

The Gemini endpoint was wrong in the database. Fix it:

```bash
# Connect to database
docker-compose exec postgres psql -U aiplatform -d aiplatform

# Run this SQL:
UPDATE ai_providers
SET api_endpoint = 'https://generativelanguage.googleapis.com/v1beta/models',
    config = '{"temperature": 0.7, "max_tokens": 1000}'
WHERE name = 'chatgpt';

UPDATE ai_providers
SET config = '{"temperature": 0.7, "max_tokens": 1000}'
WHERE name = 'deepseek';

# Check it worked:
SELECT name, api_endpoint, config FROM ai_providers;

# Exit:
\q
```

Step 2: Update All Adapter Files

Copy the updated code for each adapter from the artifacts above:

- `backend/src/services/adapters/openai.js`
- `backend/src/services/adapters/anthropic.js`
- `backend/src/services/adapters/gemini.js`
- `backend/src/services/adapters/deepseek.js`
- `backend/src/services/aiProviders.js`

Step 3: Restart Backend

```bash
docker-compose restart backend

# Watch logs
docker-compose logs -f backend
```

🧪 Testing Each Adapter

Option A: Use the Test Script

```bash
chmod +x test-adapter.sh
./test-adapter.sh
```

This interactive script will:

1. Ask for your login credentials
2. Show available providers
3. Let you choose which to test
4. Create a test project and chat
5. Send a message and show the response
6. Clean up after testing

Option B: Manual Testing via API

1. Login

```bash
TOKEN=$(curl -s -X POST http://localhost/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}' \
  | grep -o '"token":"[^"]*"' | cut -d'"' -f4)

echo "Token: $TOKEN"
```

2. Add API Key (if not already added)

```bash
# For ChatGPT (provider_id: 1)
curl -X POST http://localhost/api/providers/keys \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "provider_id": 1,
    "key_value": "sk-your-actual-openai-key-here"
  }'
```

3. Create Test Project

bash

```
PROJECT_ID=$(curl -s -X POST http://localhost/api/projects \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"name":"Test","description":"Testing adapters"}' \
  | grep -o '"id":[0-9]*' | cut -d':' -f2)

echo "Project ID: $PROJECT_ID"
```

4. Create Chat

bash

```
CHAT_ID=$(curl -s -X POST http://localhost/api/chats/project/$PROJECT_ID \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"title":"Test Chat","provider_id":1}' \
  | grep -o '"id":[0-9]*' | cut -d':' -f2)

echo "Chat ID: $CHAT_ID"
```

5. Send Message

bash

```
curl -X POST http://localhost/api/chats/$CHAT_ID/messages \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"content":"Hello! Say OK if you can read this."}' \
  | jq .
```

Expected response:

```json
{
  "message": {
    "id": 2,
    "role": "assistant",
    "content": "OK! I can read your message.",
    "token_count": 15,
    "created_at": "2024-12-09T..."
  },
  "chat_status": "active"
}
```

🔍 Debugging Tips

Check Backend Logs

```bash
# Follow logs in real-time
docker-compose logs -f backend

# You should see:
# - "Calling chatgpt API at https://..."
# - Request payload (JSON)
# - "chatgpt response received"
# - Or error details if it failed
```

Common Issues & Fixes

Issue 1: "Invalid API key"

Symptoms:

```json
{"error": "invalid_api_key"}
```

Fix:

- Check your API key is correct
- Check it has proper permissions
- For OpenAI: Key should start with `sk-`
- For Anthropic: Key should start with `sk-ant-`
- For Gemini: Get key from Google AI Studio

Issue 2: "Invalid response format"

Symptoms:

```
json

{"error": "api_error", "message": "Invalid response format from ..."}
```

Fix:

- This means the API returned unexpected data
- Check backend logs for the actual response
- The adapter might need adjusting for your specific API version

Issue 3: "Rate limit exceeded"

Symptoms:

```
json

{"error": "rate_limited"}
```

This is normal! The system handles it:

- Chat status becomes "pending_rate_limit"
- Shows reset time to user
- Background job will reactivate after reset

Issue 4: "No available API keys"

Symptoms:

```
json

{"error": "no_available_keys"}
```

Fix:

```bash
# Check if keys exist
curl http://localhost/api/providers \
  -H "Authorization: Bearer $TOKEN" | jq .

# Add a key
curl -X POST http://localhost/api/providers/keys \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"provider_id": 1, "key_value": "your-key"}'
```

📝 Adapter-Specific Notes

ChatGPT (OpenAI)

Models supported:

- `gpt-3.5-turbo` (default, cheapest)
- `gpt-4` (smarter, more expensive)
- `gpt-4-turbo`

API Key format: `sk-proj-...` or `sk-...`

Get key: https://platform.openai.com/api-keys

Claude (Anthropic)

Models supported:

- `claude-3-5-sonnet-20241022` (default)
- `claude-3-opus-20240229`
- `claude-3-haiku-20240307`

API Key format: `sk-ant-...`

Get key: https://console.anthropic.com/

Gemini (Google)

Models supported:

- `gemini-pro` (default)
- `gemini-pro-vision` (for images)

API Key format: 39-character string

Get key: https://makersuite.google.com/app/apikey

Note: Gemini uses API key in URL, not header

DeepSeek

Models supported:

- `deepseek-chat` (default)
- `deepseek-coder`

API Key format: Similar to OpenAI

Get key: https://platform.deepseek.com/

✅ Verification Checklist

After applying fixes:

- [ ] All adapter files updated
- [ ] Database config updated (SQL above)
- [ ] Backend restarted
- [ ] API key added for at least one provider
- [ ] Test script runs successfully OR
- [ ] Manual API test returns AI response
- [ ] Backend logs show successful API call
- [ ] No errors in browser console (if using web UI)

🎯 Next Steps

Once adapters are working:

1. **Add more API keys** - Add keys for all providers you want to use
2. **Test all providers** - Run test script for each
3. **Monitor usage** - Check Admin panel for key usage
4. **Set up backups** - Run `./backup.sh` daily
5. **Update models** - Change model names in database if needed

🆘 Still Having Issues?

If adapters still don't work after these fixes:

1. **Check backend logs:**

```bash
docker-compose logs backend | grep -A 20 "Error"
```

2. **Verify database config:**

```bash
docker-compose exec postgres psql -U aiplatform -d aiplatform \
  -c "SELECT name, api_endpoint, model_name, adapter_module FROM ai_providers;"
```

3. **Test API key separately:**

```bash
# For OpenAI:
curl https://api.openai.com/v1/chat/completions \
  -H "Authorization: Bearer sk-your-key" \
  -H "Content-Type: application/json" \
  -d '{"model":"gpt-3.5-turbo","messages":[{"role":"user","content":"test"}]}'
```

4. **Check adapter file exists:**

```bash
ls -la backend/src/services/adapters/*.js
```

Let me know which adapter is still failing and I'll help debug further!