API Authentication Guide

Overview

This FastAPI server uses **API Key Authentication** for secure server-to-server communication. Only requests with valid API keys will be processed.

Authentication Flow

```
Node.js Server → [X-API-Key Header] → FastAPI Server → [Validation] → Response
```

Setup Instructions

1. Generate API Key (FastAPI Side)

Run the key generator script:

```
python generate_api_key.py
```

This will generate secure API keys and optionally save them to your `.env` file.

2. Configure Environment Variables

FastAPI Server (.env file):

```
SERVER_API_KEY=your_generated_api_key_here
```

Node.js Server (.env file):

```
FASTAPI_API_KEY=your_generated_api_key_here
```

3. Node.js Implementation

Using Axios:

```
const axios = require('axios');

const FASTAPI_URL = 'http://localhost:8000';
const API_KEY = process.env.FASTAPI_API_KEY;

async function callFastAPI(payload) {
  try {
    const response = await axios.post(`${FASTAPI_URL}/agent/kay-bot`, payload, {
      headers: {
        'Content-Type': 'application/json',
        'X-API-Key': API_KEY  // This is the secret key
      }
    });
    return response.data;
  } catch (error) {
    console.error('FastAPI call failed:', error.response?.data || error.message);
    throw error;
  }
}
```

Using Fetch:

```
const FASTAPI_URL = 'http://localhost:8000';
const API_KEY = process.env.FASTAPI_API_KEY;

async function callFastAPI(payload) {
  try {
    const response = await fetch(`${FASTAPI_URL}/agent/kay-bot`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'X-API-Key': API_KEY  // This is the secret key
      },
      body: JSON.stringify(payload)
    });

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    return await response.json();
  } catch (error) {
```

```
    console.error('FastAPI call failed:', error);
    throw error;
  }
}
```

Protected Endpoints

All endpoints except health checks require the `X-API-Key` header:

1. Kay Bot Endpoint

```
POST /agent/kay-bot
X-API-Key: your_api_key_here
Content-Type: application/json

{
    "age": "25",
    "gender": "Female",
    "name": "Sarah",
    "patient_id": "6899521238bcd98456d965e0",
    "message": "I'm feeling anxious today"
}
```

2. Chat Summary Endpoint

```
GET /agent/chat/summary/{patient_id}
X-API-Key: your_api_key_here
```

3. Authentication Test Endpoint

```
GET /agent/auth/test
X-API-Key: your_api_key_here
```

Public Endpoints (No Authentication Required)

Health Check

```
GET /agent/health
```

Database Health Check

```
GET /agent/health/db
```

Error Responses

Missing API Key (401 Unauthorized)

```
{
    "detail": "API key is required. Please provide X-API-Key header."
}
```

Invalid API Key (401 Unauthorized)

```
{
    "detail": "Invalid API key provided."
}
```

Testing

Use the provided test file `testing/api_auth_tests.http` to test authentication:

1. Update the `@validApiKey` variable with your actual API key
2. Run the tests using REST Client extension in VS Code
3. Verify that requests with valid keys succeed and invalid keys fail

Security Best Practices

✅ Do:

- Store API keys in environment variables
- Use HTTPS in production
- Rotate keys periodically

- Share keys through secure channels
- Monitor authentication failures

❌ Don't:

- Hardcode API keys in source code
- Commit API keys to version control
- Share keys via email or chat
- Use the same key across different environments
- Log API keys in application logs

Key Exchange Process

1. **FastAPI Developer**: Generates secure API key using `generate_api_key.py`
2. **Secure Sharing**: Share key through secure channel (secrets manager, encrypted message, etc.)
3. **Node.js Developer**: Stores key in environment variables
4. **Testing**: Both sides test authentication using provided test files
5. **Production**: Deploy with keys in production environment variables

Troubleshooting

Common Issues:

1. **401 Unauthorized**: Check that API key is correctly set in environment variables
2. **Missing Header**: Ensure `X-API-Key` header is included in requests
3. **Wrong Key**: Verify both servers are using the same API key
4. **Environment Variables**: Make sure environment variables are loaded correctly

Debug Steps:

1. Check health endpoint: `GET /agent/health` (should show API key status)
2. Test authentication: `GET /agent/auth/test` with valid key
3. Verify environment variables are loaded
4. Check server logs for authentication errors

Example Integration

```
// Complete Node.js integration example
const express = require('express');
const axios = require('axios');

const app = express();
app.use(express.json());

const FASTAPI_URL = process.env.FASTAPI_URL || 'http://localhost:8000';
const API_KEY = process.env.FASTAPI_API_KEY;

// Middleware to add API key to FastAPI requests
const callFastAPI = async (endpoint, payload) => {
  try {
    const response = await axios.post(`${FASTAPI_URL}${endpoint}`, payload, {
      headers: {
        'Content-Type': 'application/json',
        'X-API-Key': API_KEY
      }
    });
    return response.data;
  } catch (error) {
    console.error('FastAPI Error:', error.response?.data || error.message);
    throw error;
  }
};

// Example endpoint that calls FastAPI
app.post('/chat', async (req, res) => {
  try {
    const { patient_id, message, user_info } = req.body;

    const payload = {
      age: user_info.age,
      gender: user_info.gender,
      name: user_info.name,
      patient_id: patient_id,
      message: message
    };

    const aiResponse = await callFastAPI('/agent/kay-bot', payload);

    res.json({
      success: true,
      response: aiResponse.response,
      chat_saved: aiResponse.chat_saved
    });
  } catch (error) {
    res.status(500).json({
      success: false,
```

```
          error: 'Failed to get AI response'
        });
      }
    });

    app.listen(3000, () => {
      console.log('Node.js server running on port 3000');
    });
```

Support

If you encounter issues with API authentication:

1. Check the FastAPI server logs for detailed error messages
2. Verify the API key is correctly configured on both sides
3. Test with the provided test files
4. Ensure network connectivity between servers