# Stage 1 REST API — Quick Guide

- PHP + MySQL backend with 14 endpoints (7 protected)

- Run locally via PHP built-in or NGINX script

- Includes docs page and two test suites

# Agent State Controller

- Manages agent lifecycle: idle, running, paused, error

- CRUD for agents, tasks, tools; logs for execution history

- Bearer token auth for mutating operations

- Front controller via PHP (built-in) or NGINX + PHP-FPM

- Docs page and two test suites included

# Prerequisites

- PHP 8+, MySQL (or MariaDB)

- (Optional) NGINX + PHP-FPM

- VS Code (for editing), curl (for tests)

# Endpoints Overview

- Agents: list, get, create, update, delete

- Tasks: list, get, create, update, delete

- Tools: list, create

- Logs: list

# Protected Endpoints (Bearer)

- POST/PUT/DELETE for Agents and Tasks

- POST Tools, Logs write endpoints

- Provide header: `Authorization: Bearer <token>`

# Sample Request (Protected)

Create Agent:

```
curl -X POST http://localhost:8000/agents \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"name":"DemoAgent","description":"test"}'
```

# Sample Response

```
{ "id": 1, "name": "DemoAgent", "status": "idle" }
```

# NGINX Config Anatomy

- listen 8000; root code/
- `try_files $uri /index.php?$args`
- fastcgi_pass 127.0.0.1:9000
- minimal, self-contained config

# PHP-FPM Quick Start

- macOS: `brew services start php`
- Linux/WSL: `sudo systemctl start php-fpm`
- Script auto-starts if not running

# Windows / WSL Note

- Recommended: WSL (Ubuntu) then follow Linux steps

- Native: nginx.exe + php-cgi.exe (not primary path)

# Schema Overview

- agents, tasks, tools, logs, api_tokens

- Foreign keys: tasks→agents, logs→(agents,tasks)

- Indexes on status and timestamps

# Seed Data (Optional)

- `./setup.sh` prompts to load seed.sql
- Provides sample agents/tasks/tools/logs

# 1) Configure Database (.env)

Create `code/.env` :

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_NAME=agent_management
DB_USER=root
DB_PASS=123456
```

Tip: Keep this file local. Values above are for local dev only.

# 2) Initialize Database

From `code/database/` :

```
DB_HOST=127.0.0.1 DB_PORT=3306 DB_USER=root DB_PASS=123456 ./setup.sh
```

- Creates DB and tables, optional seed data

```
[foxj7@MacBook-Pro code % cd database
[foxj7@MacBook-Pro database % l
total 40
-rw-r--r--@ 1 foxj7  staff  1157 Oct 26 21:39 database.md
-rw-r--r--@ 1 foxj7  staff  1944 Oct 25 00:37 schema.sql
-rw-r--r--@ 1 foxj7  staff  1378 Oct 25 19:57 seed.sql
-rwxr-xr-x@ 1 foxj7  staff   686 Oct 26 21:13 setup.sh
-rwxr-xr-x@ 1 foxj7  staff  1219 Oct 25 19:24 test.sh
[foxj7@MacBook-Pro database % ./setup.sh
[Enter MySQL password for root:
Creating database and tables...
Database schema created successfully
Load seed data? (y/n): y
Seed data loaded successfully
foxj7@MacBook-Pro database % 
```

# 3) Start Server (PHP built-in)

**From** `code/` :

```
php -S localhost:8000
```

**Visit:**

- http://localhost:8000/docs

```
[foxj7@MacBook-Pro project-stage1-submission % cd code
[foxj7@MacBook-Pro code % php -S localhost:8000
[Sun Oct 26 22:11:40 2025] PHP 8.4.14 Development Server (http://localhost:8000) started
[Sun Oct 26 22:11:44 2025] 127.0.0.1:52862 Accepted
[Sun Oct 26 22:11:44 2025] 127.0.0.1:52862 [404]: GET /tests - No such file or directory
[Sun Oct 26 22:11:44 2025] 127.0.0.1:52862 Closing
[Sun Oct 26 22:11:44 2025] 127.0.0.1:52864 Accepted
[Sun Oct 26 22:11:48 2025] 127.0.0.1:52864 [200]: GET /docs
[Sun Oct 26 22:11:48 2025] 127.0.0.1:52864 Closing
```

# 4) Start Server (NGINX script)

**From project root:**

```
./deploy/start_nginx.sh
```

- Auto-generates config, starts services if needed
- Visit http://localhost:8000/docs

**Stop:**

```
./deploy/stop_nginx.sh
```

# 5) Generate Auth Token

From `code/` :

```
./generate_token.sh
```

- Inserts token into DB

- Use "Authorization: Bearer <token>" in protected calls

```
-rw-r--r--@ 1 foxj7  staff    260 Oct 25 19:33 tools.php
[foxj7@MacBook-Pro code % ./generate_token.sh
[Enter MySQL password for root:
Enter token description (optional):
mysql: [Warning] Using a password on the command line interface can be insecure.

Token created successfully!
===============================
Token: b89991ec21e19bad1ebcea8918926f0a05b252c9ec265c32402dca9d98d11bae
===============================


Use in requests:
Authorization: Bearer b89991ec21e19bad1ebcea8918926f0a05b252c9ec265c32402dca9d98d11bae
foxj7@MacBook-Pro code %
```

# 6) Test — Shell Suite

From `code/` :

```
./tests/test_api.sh
```

- Runs cURL tests against all endpoints
- Shows HTTP codes and pass/fail
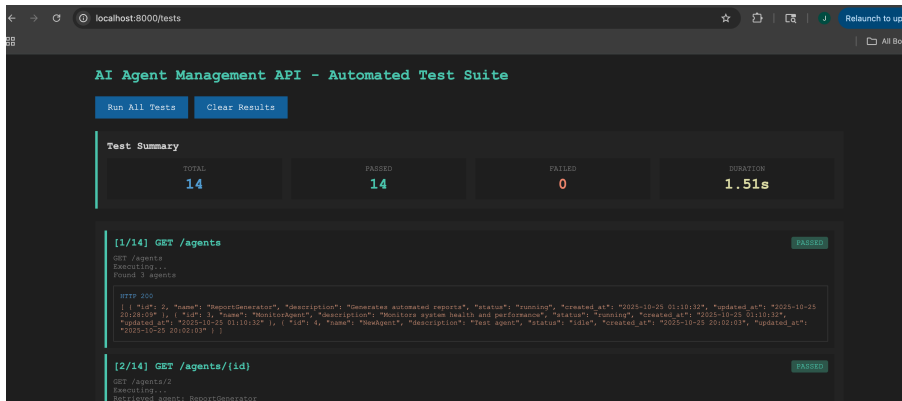


cURL run



DB verification
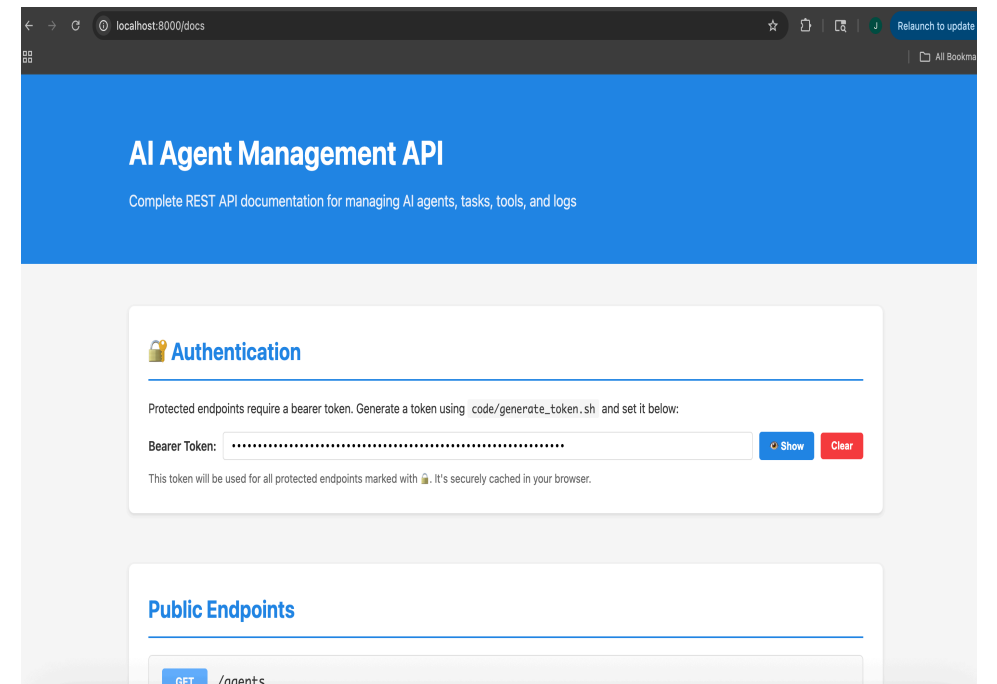
# 7) Test — Browser Suite

Open:

```
open code/tests/test.html        # macOS
# or
xdg-open code/tests/test.html    # Linux
```

- Click "Run All Tests"

- See per-endpoint results

# 8) Docs Page

- http://localhost:8000/docs

- Try endpoints interactively

- Paste the Bearer token for protected routes

# 9) Troubleshooting (Quick)

- DB errors: check `code/.env` values
- 502 in NGINX: ensure PHP-FPM is running
  - macOS: `brew services start php`
  - Linux/WSL: `sudo systemctl start php-fpm`
- Port busy: script frees 8000 automatically

# 10) What's Included

- REST API (PHP) with MySQL (PDO)
- Auth via `api_tokens` table
- `./deploy/start_nginx.sh` + `./deploy/stop_nginx.sh`
- Docs page + two test suites

# DB Troubleshooting

- Access denied: check `DB_USER/DB_PASS`

- Unknown DB: run setup.sh

- Not running: start MySQL (brew/systemctl)

# Ports & Tips

- Default port: 8000 (script frees if busy)
- Use `php -S` for quick dev, NGINX for parity