🚀 SipSpot 项目快速启动指南

这是一份零到一的完整启动指南，帮助你快速搭建和运行SipSpot项目。

📋 前置要求

在开始之前，请确保你的电脑已安装：

- ✅ **Node.js** (v16+): 下载地址
- ✅ **Python** (v3.9+): 下载地址
- ✅ **MongoDB**: 下载地址 或使用 MongoDB Atlas
- ✅ **Git**: 下载地址
- ✅ **代码编辑器**: VS Code 推荐

验证安装

```bash
node --version    # 应显示 v16.0.0 或更高
npm --version     # 应显示 8.0.0 或更高
python --version  # 应显示 Python 3.9.0 或更高
mongod --version  # 应显示 MongoDB 版本信息
```

🎯 30分钟快速启动

第一步：创建项目结构（2分钟）

```bash
# 创建主项目文件夹
mkdir sipspot
cd sipspot

# 创建子项目文件夹
mkdir backend frontend ai-service
```

你的项目结构应该是这样的：

```
sipspot/
├── backend/      # Node.js后端
```

```
├── frontend/     # React前端
└── ai-service/   # Python AI服务
```

第二步：设置后端（8分钟）

1. 初始化Node.js项目

```bash
cd backend
npm init -y
```

2. 安装依赖

```bash
npm install express mongoose dotenv cors bcryptjs jsonwebtoken multer cloudinary axios express-validator express-rate-limi

npm install -D nodemon
```

3. 创建环境配置

创建 `backend/.env` 文件：

```env
```

```
NODE_ENV=development
PORT=5000

# MongoDB - 选择一个配置
# 本地MongoDB
MONGODB_URI=mongodb://localhost:27017/sipspot

# 或使用 MongoDB Atlas
# MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/sipspot

JWT_SECRET=your_super_secret_key_change_this_in_production
JWT_EXPIRE=7d

# Cloudinary配置（可选，用于图片上传）
CLOUDINARY_CLOUD_NAME=your_cloud_name
CLOUDINARY_API_KEY=your_api_key
CLOUDINARY_API_SECRET=your_api_secret

# AI服务URL
AI_SERVICE_URL=http://localhost:8000

# CORS
CLIENT_URL=http://localhost:3000
```

4. 复制代码文件

从我提供的 `Backend_Code_Examples.md` 中：

- 复制 `models/` 文件夹下的所有模型文件
- 复制 `controllers/` 文件夹下的所有控制器
- 复制 `middleware/` 文件夹下的中间件
- 复制 `server.js` 主文件

你的后端结构应该是：

```
backend/
├── src/
│   ├── models/
│   │   ├── User.js
│   │   ├── Cafe.js
│   │   └── Review.js
│   ├── controllers/
```

```
│   │   ├── authController.js
│   │   └── cafeController.js
│   ├── middleware/
│   │   └── auth.js
│   └── routes/
│       ├── auth.js
│       ├── cafes.js
│       └── reviews.js
├── server.js
├── package.json
└── .env
```

5. 创建路由文件

routes/auth.js

javascript

```javascript
const express = require('express');
const router = express.Router();
const { register, login, getMe } = require('../controllers/authController');
const { protect } = require('../middleware/auth');

router.post('/register', register);
router.post('/login', login);
router.get('/me', protect, getMe);

module.exports = router;
```

routes/cafes.js

javascript

```javascript
const express = require('express');
const router = express.Router();
const {
  getCafes,
  getCafe,
  createCafe,
  updateCafe,
  deleteCafe,
```

```
  getNearby
} = require('../controllers/cafeController');
const { protect } = require('../middleware/auth');

router.get('/', getCafes);
router.get('/nearby', getNearby);
router.get('/:id', getCafe);
router.post('/', protect, createCafe);
router.put('/:id', protect, updateCafe);
router.delete('/:id', protect, deleteCafe);

module.exports = router;
```

6. 更新 package.json

在 `package.json` 中添加 scripts：

```
json

{
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  }
}
```

7. 启动后端服务

```
bash

# 确保MongoDB正在运行
# 如果使用本地MongoDB：mongod

# 启动后端
npm run dev
```

你应该看到：

```
✅ MongoDB连接成功
🚀 服务器运行在端口 5000
```

测试后端：

```
bash

curl http://localhost:5000/api/auth/register
```

curl http://localhost:5000/api/auth/register

第三步：设置前端（10分钟）

1. 创建React项目

bash

```
cd ../frontend
npx create-react-app .
```

2. 安装依赖

bash

```
npm install axios react-router-dom react-query antd @ant-design/icons leaflet react-leaflet

npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

3. 配置Tailwind CSS

编辑 `tailwind.config.js`：

javascript

```
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

编辑 `src/index.css`：

css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

4. 创建环境配置

4. 创建环境配置

创建 `frontend/.env`:

```
env

REACT_APP_API_URL=http://localhost:5000/api
```

5. 复制代码文件

从 `Frontend_Code_Examples.md` 中复制:

- `services/` 文件夹
- `contexts/` 文件夹
- `hooks/` 文件夹
- `components/` 文件夹
- `pages/` 文件夹
- `App.jsx`

你的前端结构应该是:

```
frontend/
├── src/
│   ├── components/
│   │   ├── Navbar.jsx
│   │   ├── CafeCard.jsx
│   │   └── CafeList.jsx
│   ├── pages/
│   │   ├── Home.jsx
│   │   └── Login.jsx
│   ├── services/
│   │   └── api.js
│   ├── contexts/
│   │   └── AuthContext.jsx
│   └── App.jsx
├── package.json
└── .env
```

6. 启动前端

```
bash

npm start
```

浏览器应该自动打开 `http://localhost:3000`

第四步：设置AI服务（10分钟）

1. 创建Python虚拟环境

```bash
cd ../ai-service

# 创建虚拟环境
python -m venv venv

# 激活虚拟环境
# Windows:
venv\Scripts\activate
# Mac/Linux:
source venv/bin/activate
```

2. 安装依赖

创建 `requirements.txt`:

```txt
fastapi==0.104.1
uvicorn==0.24.0
pydantic==2.5.0
python-dotenv==1.0.0
google-generativeai==0.3.1
httpx==0.25.1
```

安装:

```bash
pip install -r requirements.txt
```

3. 获取Gemini API密钥

1. 访问 Google AI Studio
2. 点击 "Get API Key"

3. 创建或选择一个项目
4. 复制API密钥

4. 创建环境配置

创建 `ai-service/.env`:

```
env

GEMINI_API_KEY=your_gemini_api_key_here
PORT=8000
BACKEND_URL=http://localhost:5000
```

5. 复制代码文件

从 `AI_Service_Code_Examples.md` 复制 `main.py`

6. 启动**AI**服务

```
bash

uvicorn main:app --reload --port 8000
```

你应该看到:

```
INFO:     Uvicorn running on http://0.0.0.0:8000
```

测试AI服务:

```
bash

curl http://localhost:8000/health
```

✅ 验证所有服务

现在你应该有三个服务在运行:

1. 后端: http://localhost:5000 ✅
2. 前端: http://localhost:3000 ✅
3. **AI**服务: http://localhost:8000 ✅

完整测试流程

1. 注册用户

1. 注册用户

```bash
curl -X POST http://localhost:5000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "username": "testuser",
    "email": "test@example.com",
    "password": "password123"
  }'
```

2. 登录获取Token

```bash
curl -X POST http://localhost:5000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "test@example.com",
    "password": "password123"
  }'
```

复制返回的 token

3. 创建咖啡店

```bash
curl -X POST http://localhost:5000/api/cafes \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_TOKEN_HERE" \
  -d '{
    "name": "星享咖啡",
    "description": "一家温馨的精品咖啡店",
    "location": {
      "type": "Point",
      "coordinates": [116.4074, 39.9042]
    },
    "address": "北京市朝阳区三里屯路11号",
    "city": "北京",
    "price": 3,
    "amenities": ["WiFi", "插座", "安静"]
  }'
```

4. 测试AI分析

```bash
curl -X POST http://localhost:8000/analyze \
  -H "Content-Type: application/json" \
  -d '{
    "content": "这家咖啡店环境很好，咖啡味道浓郁，服务周到！",
    "cafe_name": "星享咖啡",
    "rating": 5.0
  }'
```

🎨 前端测试

打开浏览器访问 `http://localhost:3000`，你应该能够：

1. ✅ 看到首页
2. ✅ 注册新用户
3. ✅ 登录
4. ✅ 浏览咖啡店列表
5. ✅ 查看咖啡店详情
6. ✅ 发布评论

📦 MongoDB数据准备（可选）

如果你想快速测试，可以导入一些示例数据：

创建 `backend/seed.js`：

```javascript
const mongoose = require('mongoose');
require('dotenv').config();

const Cafe = require('./src/models/Cafe');

const sampleCafes = [
  {
    name: "星巴克臻选店",
    description: "全球知名咖啡连锁品牌的高端门店",
    location: {
```

```
    type: "Point",
    coordinates: [116.4074, 39.9042]
   },
   address: "北京市朝阳区三里屯路19号",
   city: "北京",
   price: 4,
   amenities: ["WiFi", "插座", "空调"],
   rating: 4.5,
   reviewCount: 128
  },
  {
   name: "精品咖啡工作室",
   description: "独立精品咖啡店，手冲咖啡专家",
   location: {
    type: "Point",
    coordinates: [116.3998, 39.9167]
   },
   address: "北京市东城区五道营胡同12号",
   city: "北京",
   price: 3,
   amenities: ["WiFi", "安静", "户外座位"],
   rating: 4.8,
   reviewCount: 86
  }
];

async function seedDatabase() {
  try {
   await mongoose.connect(process.env.MONGODB_URI);
   console.log('MongoDB连接成功');

   await Cafe.deleteMany({});
   console.log('清除旧数据');

   await Cafe.insertMany(sampleCafes);
   console.log('示例数据导入成功');

   process.exit(0);
  } catch (error) {
   console.error('错误:', error);
   process.exit(1);
  }
}

seedDatabase();
```

```
seedDatabase();
```

运行：

```bash
node seed.js
```

🐛 常见问题排查

MongoDB连接失败

问题：`MongooseServerSelectionError: connect ECONNREFUSED`

解决：

```bash
# 检查MongoDB是否运行
sudo systemctl status mongodb  # Linux
brew services list             # Mac

# 启动MongoDB
sudo systemctl start mongodb   # Linux
brew services start mongodb    # Mac
mongod                         # 手动启动
```

CORS错误

问题：`Access to XMLHttpRequest has been blocked by CORS policy`

解决：在 `backend/server.js` 中确认CORS配置：

```javascript
app.use(cors({
  origin: 'http://localhost:3000',
  credentials: true
}));
```

Gemini API错误

问题：`API key not valid`

解决：

1. 检查 `.env` 文件中的 `GEMINI_API_KEY`

2. 确认API密钥有效
3. 检查API配额

端口被占用

问题: `Error: listen EADDRINUSE: address already in use`

解决:

```bash
# 查找占用端口的进程
lsof -i :5000  # Mac/Linux
netstat -ano | findstr :5000  # Windows

# 杀死进程
kill -9 <PID>  # Mac/Linux
taskkill /PID <PID> /F  # Windows
```

📚 下一步

恭喜！你已经成功搭建了SipSpot的基础架构。接下来可以:

1. 完善功能
   - 添加更多页面组件
   - 实现图片上传功能
   - 添加用户个人主页
2. 优化体验
   - 实现地图展示
   - 添加搜索筛选
   - 优化移动端适配
3. 部署上线
   - 前端部署到Vercel
   - 后端部署到Render
   - AI服务部署到Railway
4. 学习资源
   - 参考 `SipSpot_Project_Plan.md` 了解完整架构