实战项目建议：个人技术名片网站的自动化部署

这个项目的核心思想是：**用项目管理工具来规划，用代码来生成一个静态网页，再用CI/CD自动把它发布到互联网上。**

1. 设计理念

- **低门槛**：不强制要求学员会复杂的前端或后端，甚至不需要云服务器和Docker。只需要Python基础和Git。
- **高回报**：最终产物是一个公开的、可以访问的个人网页（托管在GitHub Pages上），可以直接作为求职时的在线简历或作品集入口。
- **流程完整**：完美覆盖从“需求提出”到“自动上线”的闭环，让学员亲身体验整个流程。

2. 项目目标

创建一个简单的单页个人网站，包含以下信息：

- 姓名和头衔 (如: "张三 - AI Engineer")
- 个人简介
- 技能列表 (如: Python, TensorFlow, Git)
- 项目经历链接 (可以链接到GitHub仓库)
- 联系方式 (GitHub/LinkedIn链接)

最终目标是：**当代码被合并到 `main` 分支时，这个网站会自动更新并发布到 `https://<你的用户名>.github.io/<仓库名>/`**

3. 技术栈

- **项目管理**：GitHub Issues
- **代码生成**：Python (无需Web框架，只用一个简单的脚本生成HTML文件)
- **版本控制**：Git & GitHub
- **CI/CD**：GitHub Actions
- **部署托管**：GitHub Pages (免费，集成度高)

4. 实战步骤分解 (可对应课程的实操幻灯片)

第一阶段：项目规划与初始化 (Project Management & Init)

1. **创建项目 (对应幻灯片8, 10)**
    - 在GitHub上创建一个新的公开仓库，例如 `my-dev-card`。
    - **任务**: 初始化仓库时，选择添加一个 `README.md` 和 `.gitignore` (选择Python模板)。

2. **用Issues定义任务 (对应幻灯片7, 10)**
    - 进入仓库的 "Issues" 标签页。
    - **任务**: 创建至少3个Issue，模拟项目需求：
        - `#1: 创建生成网页内容的基础Python脚本`
        - `#2: 添加个人信息和技能列表到网页`
        - `#3: 配置GitHub Actions实现自动化部署`

第二阶段：本地开发与代码协作 (Git & Collaboration)

3. **克隆并创建功能分支 (对应幻灯片11, 12, 13)**
    - `git clone` 仓库到本地。
    - **任务**: 针对Issue #1，创建一个新的 `feature/add-script` 分支。
        - `git checkout -b feature/add-script`

4. **编写核心代码 (最简单的版本)**
    - 创建一个Python脚本 `build_page.py` 。这个脚本的功能极其简单：读取一个文本文件（或直接在代码里定义）中的个人信息，然后生成一个 `index.html` 文件。
    - **示例 `build_page.py`**:

```python
def main():
    html_content = """
    <html>
    <head><title>My Tech Card</title></head>
    <body>
        <h1>张三</h1>
        <p>AI Engineer passionate about MLOps.</p>
        <h2>Skills</h2>
        <ul>
            <li>Python</li>
            <li>Git</li>
            <li>CI/CD</li>
        </ul>
    </body>
    </html>
    """
    with open("index.html", "w", encoding="utf-8") as f:
        f.write(html_content)
    print("index.html generated successfully!")

if __name__ == "__main__":
    main()
```

    - **任务**: 在本地运行 `python build_page.py` ，确认 `index.html` 文件被正确创建。

5. **提交并创建Pull Request (对应幻灯片13)**

- **任务**: 提交代码，并在commit message中关联Issue。
  - `git add .`
  - `git commit -m "feat: Add basic script to generate homepage (closes #1)"`
- **任务**: 推送到远程并创建Pull Request。在PR的描述中，简单说明你做了什么。请另一位同学（或讲师）进行简单的Code Review。

第三阶段：CI/CD 自动化 (Automation)

6. **合并PR，开始自动化任务**
   - 在PR被Review通过后，合并到 `main` 分支。
   - **任务**: 现在，针对Issue #3，创建新的 `feature/add-ci` 分支。
7. **编写GitHub Actions工作流 (对应幻灯片21, 25, 26)**
   - 在项目根目录下创建 `.github/workflows/` 文件夹。
   - 在其中创建一个 `deploy.yml` 文件。
   - **任务**: 编写 `deploy.yml`，内容如下：

```yaml
name: Deploy Personal Tech Card

on:
  push:
    branches:
      - main # 只在main分支更新时触发

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.10'

      - name: Generate HTML file
        run: python build_page.py

      - name: Setup Pages
        uses: actions/configure-pages@v3

      - name: Upload artifact
        uses: actions/upload-pages-artifact@v2
        with:
```

```
          path: '.' # 上传整个目录，GitHub Pages会找到index.html

      - name: Deploy to GitHub Pages
        id: deployment
        uses: actions/deploy-pages@v2
```

- **注意**: 这份配置利用了GitHub最新的官方Actions，非常简洁，无需手动处理部署秘钥。

8. **配置GitHub Pages**
   - **任务**: 进入仓库的 `Settings` -> `Pages` 。在 `Build and deployment` -> `Source` 下，选择 `GitHub Actions` 。
9. **最终验证**
   - **任务**: 将 `feature/add-ci` 分支的代码提交并创建PR，然后合并到 `main` 分支。
   - 合并后，进入仓库的 "Actions" 标签页，你会看到流水线正在运行。
   - 等待流水线成功后，访问 `https://<你的用户名>.github.io/<仓库名>/` ，就能看到你的个人主页了！

总结这个实战项目的优势：

1. **串联核心知识点**： `GitHub Issues` (PM) -> `Git Branch/PR` (协作) -> `GitHub Actions` (CI/CD) -> `GitHub Pages` (部署)。
2. **结果导向**：学员能立刻看到自己的成果（一个上线的网站），成就感极强。
3. **过程可控**：每一步都非常清晰，不易出错，讲师容易引导和排错。
4. **高度相关**：最终产出的 `GitHub Portfolio` 和 `CI/CD` 经验，都是求职时非常有价值的展示项，完美契合课程目标。