接口对齐

把需要对齐的部分放在这里！

没问题！结合你们现有构想与我给出的最小可运行样例（h5-min：H5 + Flask），下面是一份**24h Hackathon 的“三人极清晰分工表”**。照表执行，就能在有限时间里做出能演的 MVP（状态面板 + 学伴推荐）。

# **总体目标（MVP）**

* **功能**：  
  1. 状态面板：显示“正在上课 / 现在有空”、可手动更新状态；
  2. 学习伙伴推荐：同课 + 当前都空闲 → Top-N 卡片；
  3. 课表录入（表单）并实时生效。
* **演示路径**：登录 → 更新状态 → 提交课表 → 看状态面板刷新 → 点“刷新推荐”得到候选。

# **仓库与基础**

* 代码基线：使用我提供的最小项目（h5-min）作为起点。
* 推荐目录：

campus-status/

├─ backend/ # Flask

│ ├─ app.py # 先从 h5-min/app.py 搬来

│ └─ dal.py # 可选：后期把读写/持久化从 app.py 拆出来

├─ static/

│ └─ index.html # 先从 h5-min/static/index.html 搬来

└─ README.md

*   
  统一分支：main（避免分支地狱），每 2–3 小时合并一次并跑一下 Demo。

# **角色分工**

## **Person 1：前端（H5）**

**职责目标**：单页状态面板 + 课表表单 + 推荐卡片，移动端优先。

### **今日任务（按优先级）**

1. **落地基础 UI**（基于 static/index.html）  
   * 登录输入框（保存 localStorage.user\_id）；
   * 状态按钮（free / studying / help / busy / tired / social）；
   * 两列卡片：📚正在上课 / ✅现在有空；
   * “添加课表”小表单 + 列表预览 + “提交课表”；
2. **打通 API**
   * 轮询 GET /api/current\_status（每 30s），渲染两列；
   * 点击按钮 → POST /api/update\_status；
   * 提交课表 → POST /api/add\_schedule；
   * 点击“刷新推荐” → GET /api/recommendations?user\_id=xxx；
3. **展示与可用性**
   * 推荐卡片：姓名 / 专业 / 共同课程 / 分数 / 理由；
   * 空态提示、加载态、防重复点击；
   * 小屏样式（<=700px 一列布局）。

### **交付验收（DoD）**

* 5s 内能完成“登录 → 设置状态 → 提交课表 → 看到刷新”；
* 轮询工作正常（30s）；
* 推荐卡片可读、有空态；
* 不崩：无 user\_id / 无课表 / 无候选均正常展示。

### **依赖/对齐**

* **与后端**：接口路径与字段名（见下文“统一契约”）；
* **与算法**：推荐返回字段：id/username/major/shared\_classes/score/reason。

### **备选/加分**

* “只看同课”前端过滤开关；
* 节流/防抖刷新；
* 简单颜色提示（help/Studying 高亮）。

## **Person 2：后端（Flask）**

**职责目标**：提供稳定 JSON API、最小持久化（先内存，若有余力再 JSON 文件）。

### **今日任务（按优先级）**

1. **启动服务**
   * 使用 backend/app.py（h5-min 基线），跑通 / 静态页；
2. **实现/校验 4 个接口**
   * GET /api/current\_status：返回 now / in\_class[] / free[]；
   * POST /api/add\_schedule：写入用户课表（内存或 JSON）；
   * POST /api/update\_status：写入用户 manual 状态；
   * GET /api/recommendations?user\_id=...：调用算法函数得到候选；
3. **时间与演示钩子（可选但很有用）**
   * 支持 ?now=YYYY-MM-DDTHH:MM 用于 Demo 回放；
4. **（有余力再做）持久化**
   * 新建 dal.py 把读写 JSON 抽离；
   * 原子写（写临时文件后 rename）、最小文件锁。

### **交付验收（DoD）**

* 四个接口返回 2xx，字段与契约一致；
* GET /api/current\_status 在 3 个内置用户下能返回可读内容；
* 错误入参返回 400 + json 错误信息；空用户/无候选返回 200 + 空数组；
* （若做持久化）重启不丢数据。

### **依赖/对齐**

* **与算法**：调用 get\_study\_partners(snapshot, user\_id, now) 等纯函数；
* **与前端**：CORS（若 H5 不同源）、统一字段名（见下）。

## **Person 3：算法（纯规则）**

**职责目标**：不做 I/O，只做打分与理由文案，保证“看起来聪明 + 可解释”。

### **今日任务（按优先级）**

1. **状态判断工具**
   * time\_to\_minutes(hhmm)、is\_now\_between(now,start,end)、current\_class\_for(user\_id, now)、next\_class\_for(user\_id, now)；
2. **学习伙伴推荐（MVP 核心）**
   * 过滤：与我**至少 1 门同课**；双方当前**free**；
   * 打分：10 × 共同课程数 +（对方 help +6、busy/tired -4）；
   * 产物：[{id, username, major, shared\_classes, score, reason}] 排序取 Top-5；
   * reason 模板：Shares {n} class(es); both free now（简单清晰）；
3. **（可选增强）同课连接/下一节同课**
   * 当前同课/下一节同课加权，返回 current\_class/next\_class 字段用于前端展示；
4. **自测脚本**
   * 不依赖 HTTP，直接喂入假数据跑出 Top-3 打印；
   * 覆盖边界：无同课/对方忙碌/无人可用 → 返回空数组。

### **交付验收（DoD）**

* 规则计算纯函数化、1 次调用 < 50ms；
* 任意用户调用可得到稳定 Top-N（或空数组）；
* reason 始终可读且与得分一致。

### **依赖/对齐**

* **与后端**：拿到 snapshot={users,schedules,statuses}（或直接访问内存全局）；
* **与前端**：输出字段名固定（见下）。

# **统一契约（三方必须对齐）**

* **时间**：HH:MM（24h）；**星期**：monday…friday 小写；
* **课表项**：{course, day, start, end, location?}；
* **状态枚举**：free | studying | help | busy | tired | social；
* **接口返回**：  
  + GET /api/current\_status →

{

"now":"2025-09-12T21:20",

"in\_class":[{"id":"alice","name":"Alice","major":"CS","current\_class":"CS151 @ UC 10:30-11:50","manual":"studying"}],

"free":[{"id":"bob","name":"Bob","major":"Math","next\_class":"CS151 13:00","manual":"help"}]

}

* +   
    GET /api/recommendations?user\_id=alice →

{"candidates":[

{"id":"bob","username":"Bob","major":"Math","shared\_classes":["CS151"],"score":16,"reason":"Shares 1 class(es); both free now"}

]}

*   
  **错误/空态**：  
  + 参数非法 → 400 {"ok":false,"error":"..."}；
  + 用户不存在/无候选 → 200 {"candidates":[]}。

# **时间表（建议节奏）**

| **时间** | **全队** | **Person 1（前端）** | **Person 2（后端）** | **Person 3（算法）** |
| --- | --- | --- | --- | --- |
| T+0–1h | 共同拉起项目、跑通 h5-min | 复制样式框架 | 跑起 Flask，返回静态页 | 写工具函数签名 |
| T+1–3h | 定好契约（上面一节） | 完成 UI 骨架、假数据渲染 | 实现 /current\_status | 实现 current\_class\_for 等 |
| T+3–6h | 联调 1：状态面板 | 接入轮询/状态按钮 | /add\_schedule /update\_status | 自测状态函数 |
| T+6–9h | 联调 2：推荐 | 推荐卡片展示 | /recommendations 路由 | get\_study\_partners 完成 |
| T+9–12h | 稳定性检查 | 空态/加载态/移动端适配 | 错误处理、简单日志 | 打分可解释性、排序稳定 |
| T+12–18h | Demo 数据注入 | UI 微调 | （可选）持久化 JSON | （可选）同课连接/下一节同课 |
| T+18–22h | 全链路演练 | 文案&细节优化 | 压测一轮（手点即可） | 自测脚本 + 边界用例 |
| T+22–24h | 最终演示 | 演示脚本/台词 | 演示时间钩子 ?now= | 兜底：手动状态触发推荐 |

# **风险与兜底**

* **推荐为空**：前端展示“暂无合适的学习伙伴”，并建议点击更新状态或添加课表；
* **时间不命中**：后端支持 ?now= 切到“都空闲”的时段；
* **联调失败**：保留“纯状态面板 MVP”演示（不依赖推荐）；
* **持久化来不及**：保留内存版，演示期间不重启服务即可。

# **一句话任务清单（可贴到 Issue）**

* 前端：UI骨架/轮询/状态按钮/表单/推荐卡片/空态与移动端
* 后端：4接口/参数校验/错误处理/（可选）持久化/演示时间钩子
* 算法：时间与状态函数/推荐过滤与打分/可解释 reason/自测脚本

需要我把 **JSON 持久化版**（dal.py + 路由改造）也直接生成给你们吗？半小时内即可在这套最小项目上升级到“重启不丢数据”的版本。

没问题！下面是**算法 ⇄ 后端 ⇄ 前端**需要提前对齐的关键点（含最小契约、样例与检查项），照这个执行能避免大多数“对不齐”的坑。

# **1) 身份与 ID**

* **user\_id / username / major** 三件套必须统一命名与来源。后端保存 users.json，算法按该键读取与返回；前端只传 user\_id（登录即输入名字即可）。
* **空/不存在的用户**：后端/算法统一返回 200 + []（而不是 4xx/5xx）。

# **2) 课表数据结构（前端 → 后端 → 算法）**

* 文件：schedules.json；结构（示例）：

{

"alice": {

"schedule": [

{"courseCode":"CS151","courseName":"Intro CS","day":"monday","startTime":"10:30","endTime":"11:50","location":"UC"}

]

}

}

*   
  **字段名与大小写**要完全对齐（courseCode/courseName/day/startTime/endTime/location）。算法按这些键读并计算。

# **3) 状态与枚举（前端按钮 → 后端 → 算法）**

* manualStatus 取值集合：studying | free | help | tired | busy | social | "want to hang out"；算法会把它们映射为学习/社交偏好（high/low/neutral）。
* status.json 示例：

{"alice":{"manualStatus":"studying"}}

*   
  **默认值**：未设置时 status:'unknown' + social\_preference:'neutral'。

# **4) 时间/时区与 Demo 钩子**

* **时间格式**统一 HH:MM（24h），算法用 time\_to\_minutes；**区间判定**是闭区间（含端点）。
* **星期字段**：小写英文全拼（monday…），算法按 strftime('%A').lower() 比对。
* **可注入 now**：后端支持 ?now=YYYY-MM-DDTHH:MM 便于演示重放；算法端函数也预留可选 now 参数（建议）。

# **5) API 契约（后端 ⇄ 前端；后端调用算法）**

最小 4 个接口：

1. POST /add\_schedule 保存课表（见第2点结构）。
2. POST /update\_status 写入 manualStatus。
3. GET /current\_status 返回两列表：in\_class[] / free[]（前端每 30s 轮询）。
4. GET /recommendations?user\_id=alice（或 /recommendations/{user\_id}）调用算法汇总推荐。

**推荐返回结构（统一前后端）**：

{

"candidates":[

{

"id":"bob","username":"Bob","major":"CS",

"score":23,

"shared\_classes":["CS151"],

"current\_class":"In CS151",

"reason":"Shares 1 class with you; Both interested in studying"

}

]

}

— 这些键名与算法产物一一对应。

# **6) 算法输出的可解释性字段**

* **reason**：用分号拼接前 1–2 条理由；前端原样展示。
* **current\_class / next\_class**：同课连接算法会返回这些字段，前端可选展示“正在同课/下一节同课”。

# **7) 过滤/排序/截断**

* 服务端排序按 score desc，前端不再二次排序；默认 Top-5（学习/社交）或 Top-3（同课连接）。
* 前端的“同课过滤”只做**展示过滤**，不改变服务端打分。

# **8) 轮询与性能**

* 前端轮询 current\_status **每 30s**；接口返回尽量小（只含必要字段）。
* 推荐接口**按需点击**触发（或较长间隔轮询），避免与状态接口同频。

# **9) 持久化与热更新**

* JSON 文件路径固定：data/users.json | data/schedules.json | data/status.json；算法每次调用 engine.refresh\_data() 热读。
* 后端写入需**原子写/文件锁**（至少保证不写坏 JSON）。

# **10) Demo 数据与空态**

* 准备 10–15 个假用户 + 多门重叠课程 + 多样 manualStatus，确保推荐永远“有东西看”。
* **空结果文案**：前端收到 [] 要展示“暂无合适人选”。（算法侧在无候选时返回空数组而非 None）。

## **最小对齐清单（给三人过一遍即可）**

* 字段名：courseCode/courseName/day/startTime/endTime/location 一致。
* manualStatus 的枚举值一致（前端按钮文案与后端写入一致）。
* 星期与时间格式一致（monday/HH:MM）。
* GET /current\_status 的 JSON shape 已确定并稳定；前端 30s 轮询。
* GET /recommendations 返回字段：id/username/major/score/shared\_classes/current\_class/reason。
* 空用户/空候选统一 200 + []；错误不影响 Demo 流。
* JSON 存储路径与文件名固定，算法侧每次刷新读取。

需要的话，我可以把**接口样例响应**和**前端渲染示例卡片**再补一版，你们直接抄用。