网页端

教育 + ml + 游戏？

多领国

-排行榜

-ar / visualizer？

**Study & Social tracker**

-llm - decision

-calendar - birthdays / appointments

-friend profile - info / curr status / 喜欢的东西

-“这位朋友很久没联系啦” /课表/在学什么/

-map - \*校园里 - 这位在uc，要不要一起吃饭？

* 自己的status
  + status/番茄钟/正在学习/朋友圈 现在在干什么/in class/out of class/in library/
  + -DM/ friend cycle system
  + Post 学习笔记
* 他人的status
  + 看到别人的status，在干什么， 看到空闲状态
  + 看到别人的课表
* 社交属性
  + gifting system
  + 找队友 / 找人自习
  + 课表 / 课友
  + 自动提醒：下周考试/朋友下周考试，送加油卡

Ai分析

-朋友behavior

-class schedule -

-上课内容 - 找课友 / 队友

- 基于课表/专业/空闲时间 自动匹配学习小组

* 目标

登陆 → Dashboard

* 看到自己状态 + 更新按钮
* 好友状态（如“Bob 在 UC”）
* 提示：“这位朋友很久没联系啦”
* 提醒：“你下周有考试”

Friends → 好友列表 + DM + 送礼物

Schedule → 输入课程，AI 推荐课友

Posts → 发学习笔记

Events → 输入生日/考试，自动提醒

Map（简单版）→ 用“location”字段显示“UC / Library”

## **The Power of Context**

By focusing on students within the same university, you have rich contextual data - class schedules, majors, dorms, campus events. This makes the ML much more accurate and relevant. The platform can suggest study groups based on shared classes AND compatible communication styles, or recommend hangouts based on both social patterns and geographic proximity on campus.

## **Natural Social Discovery**

Instead of artificial matchmaking, this creates organic connections: "Sarah from your Chemistry class is usually free Tuesday evenings and enjoys the same study spots as you" or "Three people in your major are planning to work on similar projects and have complementary skills based on their past collaboration patterns."

## **Dual Value Proposition**

Students get both academic support (study groups, homework help, class coordination) AND social enhancement (friend discovery, activity planning, relationship insights). This creates much stronger user retention than single-purpose apps.

## **Novel ML Applications**

* **Academic-Social Cross-Analysis**: Predict which study methods work best for different personality types
* **Campus Influence Mapping**: Identify knowledge experts in different subjects within the social network
* **Optimal Group Formation**: ML considers both academic compatibility and social dynamics for project teams
* **Event/Activity Recommendations**: Based on social patterns, academic load, and campus involvement

## **Hackathon-Ready Features**

Start with class schedule integration, basic study group formation, and simple social pattern analysis. The university-specific focus makes it much easier to demo with realistic scenarios.

This genuinely feels like something students would actually use daily. What university context are you thinking of building this around?

## **Ultra-Simple MVP: "Campus Status Board"**

**Core Feature**: A single webpage where students can see who's in class right now and who's free. That's it. But make it work really well.

# **Detailed 24-Hour Hackathon Outline**

## **Team Setup & Roles**

**Person 1 (Frontend)**: HTML/CSS/JavaScript interface

**Person 2 (Backend)**: Data handling and simple server

**Person 3 (Smart Logic)**: "ML" matching rules and algorithms

## **Hours 1-2: Project Setup**

**Everyone Together:**

* Create shared GitHub repo
* Set up basic folder structure:

campus-status/

├── frontend/

├── backend/

├── data/

└── README.md

* Choose tech stack: HTML/CSS/JS + Python Flask + JSON files

**Person 1**: Create basic HTML structure with placeholder data

**Person 2**: Set up Flask server with hello world endpoint

**Person 3**: Design data structures (write down what data you need)

## **Hours 3-6: Core Foundation**

**Person 1 Tasks:**

* Build main status board HTML page
* Create schedule input form
* Basic CSS styling (make it look like a campus app)
* Show list of "Currently in class" and "Currently free"

**Person 2 Tasks:**

* Create Flask endpoints:
  + POST /add\_schedule - save user schedule
  + GET /current\_status - get everyone's current status
  + POST /update\_status - manually update status
* Set up JSON file storage for schedules

**Person 3 Tasks:**

* Write time calculation logic (given schedule + current time → status)
* Create sample data for 10-15 fake students with realistic schedules
* Design simple matching rules

## **Hours 7-12: Make It Interactive**

**Person 1 Tasks:**

* Connect frontend to backend with JavaScript fetch()
* Add manual status buttons (tired, studying, free, want help, can help)
* Auto-refresh status every 30 seconds
* Add simple filtering (show only people in my classes)

**Person 2 Tasks:**

* Implement all the Flask endpoints properly
* Add user authentication (super simple - just name input)
* Create endpoint for "find study partners"
* Handle schedule time parsing

**Person 3 Tasks:**

* Build "smart" recommendations:
  + People in same class who are free
  + Study partners based on shared courses
  + Social matches based on status + availability
* Test with sample data

## **Hours 13-18: Smart Features**

**Person 1 Tasks:**

* Build recommendation display ("People you might want to talk to")
* Add user profile view (shows their schedule, status history)
* Mobile-responsive design
* Add simple chat/contact feature

**Person 2 Tasks:**

* Advanced endpoints:
  + GET /recommendations/{user\_id}
  + GET /class\_activity/{class\_name}
* Add basic data persistence
* Error handling and validation

**Person 3 Tasks:**

* Refine matching algorithms:
  + Weight recent interaction success
  + Consider personality compatibility
  + Time-based availability patterns
* Add "insights" (like "You study best with Alice on Tuesdays")

## **Hours 19-22: Polish & Demo Prep**

**Everyone:**

* Bug fixing and testing
* Create compelling demo scenario with realistic data
* Practice presentation (5-7 minutes)
* Write simple documentation

**Specific Polish Tasks:**

* Make it look professional with better CSS
* Add loading states and smooth transitions
* Create demo accounts with interesting stories
* Prepare backup plans if features break

## **Hours 23-24: Final Demo Prep**

* Final testing with fresh eyes
* Practice demo multiple times
* Prepare to explain the "novel" aspects clearly
* Have backups ready

## **Tech Implementation Details**

**Frontend (Person 1):**

****<!-- Basic structure -->

<div id="status-board">

<div class="currently-in-class"></div>

<div class="currently-free"></div>

<div class="recommendations"></div>

</div>

**Backend (Person 2):**

****from flask import Flask, jsonify, request

app = Flask(\_\_name\_\_)

@app.route('/current\_status')

def get\_status():

# Read schedules, calculate current status

return jsonify(status\_data)

**Smart Logic (Person 3):**

****def find\_study\_partners(user\_id):

# Simple rules that feel smart

# Same class + available + compatible status

pass



## **Demo Script Outline**

1. **Problem**: Students can't easily find classmates to study with or hang out
2. **Solution**: Real-time campus awareness with smart recommendations
3. **Demo**: Show live status updates, find study partner, show recommendations
4. **Novel Aspect**: Combines real-time scheduling with social intelligence
5. **Future**: Could add more ML, expand to multiple campuses

Want me to dive deeper into any specific technical implementation?