

- 1) User is at the Course Schedule Page and simply enters the course numbers

Automated Registration Scheduler

Intelligent Course Selection for Spring 2026

Start Automated Planning

Enter your desired courses in order of priority. The bot will optimize for the highest historical professor GPA while avoiding time conflicts.

Enter Course IDs (comma-separated)

COP 4600, CDA 3103, CEN 4020, PHY 2048

Start Automation

Current Schedule

Daily Schedule Overview

- Monday 🌟 No classes scheduled.
- Tuesday 🌟 No classes scheduled.
- Wednesday 🌟 No classes scheduled.
- Thursday ⚡ No classes scheduled.
- Friday 🎉 No classes scheduled.

- 2) The program looks at each prof available for the course, compares the prof with the USF infocenter data, selects the best non-conflicting(w/ previous class) prof with highest gpa.

Step 2 of 4: Reviewing CDA 3103

Computer Organization

Automated Decision:

The best section found is CRN 14088 with Prof. Liu, F. (Avg GPA: 3.50).

All Available Sections vs. Automated Choice

Auto-Selected	Sec 003 Instructor: Liu, F. ["M", "W"] 12:30 PM - 01:45 PM Best available non-conflicting section (Highest Avg GPA). CRN: 14088	⭐ 3.50
Available	Sec 001 Instructor: Zhang, X. ["Tu", "Th"] 02:00 PM - 03:15 PM CRN: 14080	⭐ 3.20
Time Conflict	Sec 002 Instructor: Yi, H. ["Tu", "Th"] 08:00 AM - 09:15 AM Time Conflict with COP 4600 (Operating Systems). CRN: 14087	⭐ 2.50

Current Schedule

COP 4600 (Operating Systems)
["Tu", "Th"] 08:00 AM - 09:15 AM
Instructor: Pamplona Segundo, V. Avg GPA: 3.80 ⭐
CDA 3103 (Computer Organization)
["M", "W"] 12:30 PM - 01:45 PM
Instructor: Liu, F. Avg GPA: 3.50 ⭐

Daily Schedule Overview

Monday 🌟
No classes scheduled.
Tuesday 🌟
No classes scheduled.
Wednesday 🌟
No classes scheduled.
Thursday ⚡
No classes scheduled.
Friday 🎉
No classes scheduled.

- 3) It prioritizes the courses entered first, meaning if there is a time conflict between the course entered first and the course entered second, it will select the best prof (by avg gpa from infocenter) and then it will select the best non-conflicting prof from course 2 and this continues until the last course. It finally displays a proposed plan -

Final Automated Schedule Plan

The automated plan is complete. 4 courses were successfully scheduled. Review the results below and the final timetable on the right.

Course Selection Results

✓ COP 4600 - Operating Systems

Scheduled 001 with Pamplona Segundo, V. (Avg GPA: 3.80 ★) | CRN 14808 | Tu, Th 08:00 AM - 09:15 AM

✓ CDA 3103 - Computer Organization

Scheduled 003 with Liu, F. (Avg GPA: 3.50 ★) | CRN 14088 | M, W 12:30 PM - 01:45 PM

✓ CEN 4020 - Software Engineering

Scheduled 001 with Fang, S. (Avg GPA: 3.70 ★) | CRN 14913 | Tu, Th 03:30 PM - 04:45 PM

✓ PHY 2048 - General Physics I with Calculus

Scheduled 005 with Voronin, D. (Avg GPA: 3.90 ★) | CRN 10345 | Tu, Th 06:00 PM - 07:15 PM

Current Schedule

COP 4600 (Operating Systems)

[Tu', Th'] 08:00 AM - 09:15 AM
Instructor: Pamplona Segundo, V. | Avg GPA: 3.80 ★

CDA 3103 (Computer Organization)

[M', W'] 12:30 PM - 01:45 PM
Instructor: Liu, F. | Avg GPA: 3.50 ★

CEN 4020 (Software Engineering)

[Tu', Th] 03:30 PM - 04:45 PM
Instructor: Fang, S. | Avg GPA: 3.70 ★

PHY 2048 (General Physics I with Calculus)

[Tu', Th] 06:00 PM - 07:15 PM
Instructor: Voronin, D. | Avg GPA: 3.90 ★

Daily Schedule Overview

Wednesday ⏲

12:30 PM - 01:45 PM: CDA 3103 (003)
Prof: Liu, F. | GPA: 3.50 ★

Thursday ⏲

08:00 AM - 09:15 AM: COP 4600 (001)
Prof: Pamplona Segundo, V. | GPA: 3.80 ★

03:30 PM - 04:45 PM: CEN 4020 (001)
Prof: Fang, S. | GPA: 3.70 ★

06:00 PM - 07:15 PM: PHY 2048 (005)
Prof: Voronin, D. | GPA: 3.90 ★

Code snippets - to mimic actual registration system

```
COURSE_OFFERINGS = {
    "COP 4600": {
        "title": "Operating Systems",
        "sections": [
            {"crn": "14808", "section": "001", "days": "TuTh", "time": "08:00 AM - 09:15 AM", "instructor": "Pamplona Segundo, V.", "status": "Open", "enrolled": 0, "capacity": 20},
            {"crn": "14809", "section": "002", "days": "TuTh", "time": "09:30 AM - 10:45 AM", "instructor": "Zhang, Y.", "status": "Full", "enrolled": 40, "capacity": 40},
            {"crn": "14810", "section": "003", "days": "MW", "time": "09:30 AM - 10:45 AM", "instructor": "Karthik, R.", "status": "Open", "enrolled": 30, "capacity": 30},
            {"crn": "14811", "section": "004", "days": "MW", "time": "02:00 PM - 03:15 PM", "instructor": "Smith, A.", "status": "Open", "enrolled": 20, "capacity": 20}
        ]
    },
    "CDA 3103": {
        "title": "Computer Organization",
        "sections": [
            {"crn": "14080", "section": "001", "days": "TuTh", "time": "02:00 PM - 03:15 PM", "instructor": "Zhang, X.", "status": "Open", "enrolled": 35, "capacity": 35},
            {"crn": "14087", "section": "002", "days": "TuTh", "time": "08:00 AM - 09:15 AM", "instructor": "Yi, H.", "status": "Open", "enrolled": 25, "capacity": 25},
            {"crn": "14088", "section": "003", "days": "MW", "time": "12:30 PM - 01:45 PM", "instructor": "Liu, F.", "status": "Open", "enrolled": 28, "capacity": 28}
        ]
    },
    "CEN 4020": {
        "title": "Software Engineering",
        "sections": [
            {"crn": "14913", "section": "001", "days": "TuTh", "time": "03:30 PM - 04:45 PM", "instructor": "Fang, S.", "status": "Open", "enrolled": 30, "capacity": 30},
            {"crn": "14914", "section": "002", "days": "MW", "time": "06:00 PM - 07:15 PM", "instructor": "Anderson, J.", "status": "Full", "enrolled": 40, "capacity": 40}
        ]
    },
    "PHY 2048": {}
}
```

Build the schedule -

```

def build_daily_schedule_data(schedule_data, user_courses):
    """
    Organizes scheduled sections into a dictionary grouped by day and sorted by time.
    """

    # Daily schedule keys 1-5 for Mon-Fri
    daily_schedule = {d: [] for d in range(1, 6)}
    online_schedule = []

    for course_id, section in schedule_data.items():
        # Add priority index for reference
        section['priority'] = user_courses.index(course_id) + 1

        if 'Online' in section['days']:
            online_schedule.append(section)
            continue

        # Calculate start minutes for sorting
        section['start_minutes'] = time_to_minutes(section['start_time'])

        for day_abbr in section['days']:
            day_key = DAY_MAP.get(day_abbr)
            if day_key:
                daily_schedule[day_key].append(section)

    # Sort classes within each day by start time (in-place)
    for day_key in daily_schedule:

```

Checks for conflicts -

