AURAK
الجامعة الأمريكية في رأس الخيمة
AMERICAN UNIVERSITY OF RAS AL KHAIMAH

**Introduction to Artificial Intelligence**
**CSAI 350**
**Project Report**

# Student At-Risk Recovery Planner using A* Search

**Group Members**

| Student Name | Student ID |
|---|---|
| Nebiyu Gemedu | 2023006249 |
| Mukerem Shifa | 2023006222 |
| Amir Beshir | 2023006017 |
| Adam El Mouddene | 2023006464 |
| Abdalla Eid | 2022005820 |

**Instructor:** Dr. Khaled Abdalgader Balawafi
**Submission Date:** 30/04/2026

# 1. Introduction

This report describes the Student At-Risk Recovery Planner. The goal was to build a Python dashboard that takes a student's current academic situation and returns a step-by-step plan that moves the student out of the at-risk zone. The plan is generated by an A* search algorithm written from scratch. We added Greedy Best-First search and Uniform Cost Search as baselines so we could see what A* buys us. The dashboard was built with Streamlit because it gave us cleaner risk plots and comparison tables than Tkinter.

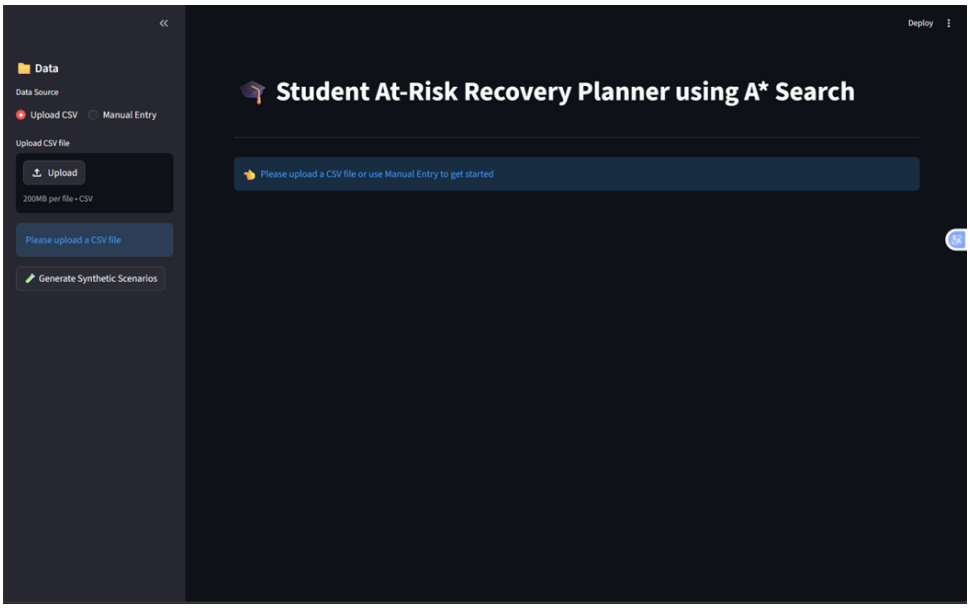


**Figure 1:** Dashboard landing page before any data is loaded, showing the CSV upload control and the option to generate synthetic scenarios.

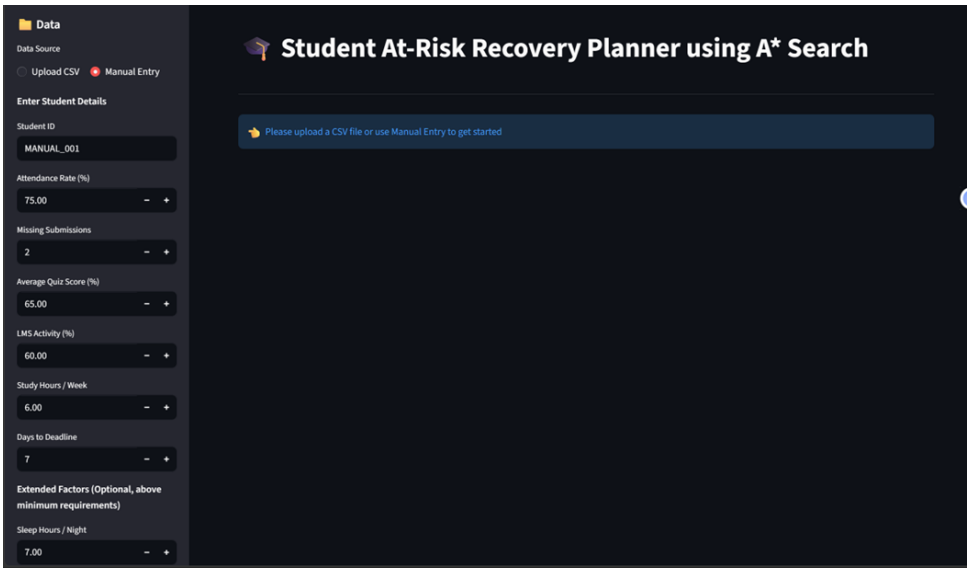


**Figure 2:** Manual Entry mode of the dashboard, used as an alternative to CSV upload.

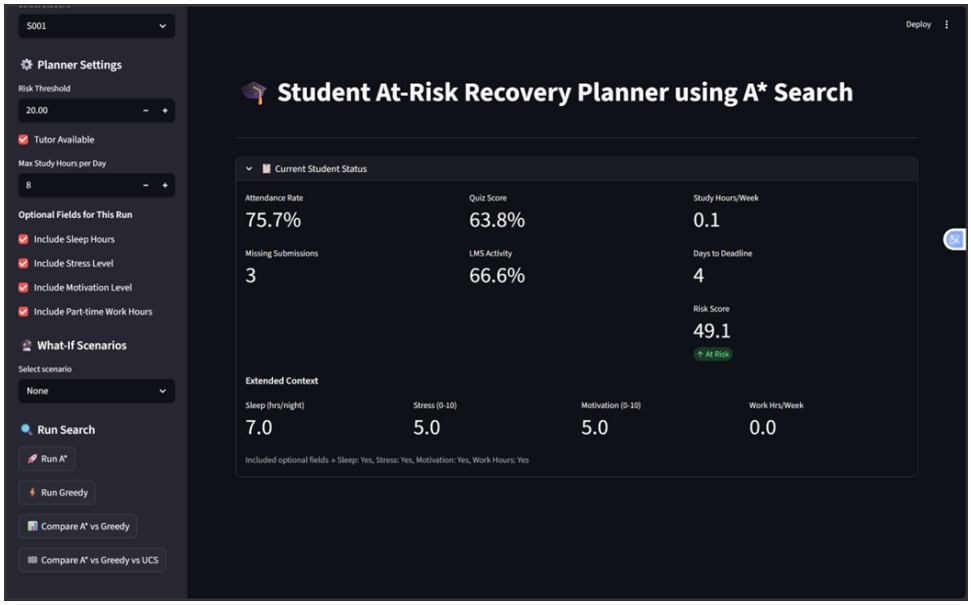


**Figure 3:** Planner sidebar and student status UI

## 2. Problem Formulation

We modeled the student situation as a search problem with four parts.

**State:** Each state is a frozen dataclass with eleven fields. The six required ones are:

- `attendance_rate` (0–100),
- `missing_submissions`,
- `avg_quiz_score` (0–100),
- `lms_activity` (0–100),
- `study_hours_per_week`, and
- `days_to_deadline`.

We added `fatigue_level`, `sleep_hours_per_night`, `stress_level`, `motivation_level`, and `part_time_work_hours` as extended context, since these affect how stressful a plan really is. Fatigue updates with every action and feeds back into the cost.

**Actions:** We defined six actions:

- `Attend_Class`,
- `Study_1_Hour`,
- `Submit_Assignment`,
- `Practice_Quiz`,
- `Meet_Tutor`, and `Rest`.

Each one has its own delta on the state fields and its own base cost. `Submit_Assignment` reduces `missing_submissions` by one and bumps `lms_activity` by three. `Rest` is the only action that lowers fatigue. Every action consumes one day from `days_to_deadline`. Some actions

have preconditions: `Submit_Assignment` is applicable only when `missing_submissions > 0`, `Meet_Tutor` is blocked if the tutor is unavailable, and `Study_1_Hour` is capped by a daily study limit.

**Goal:** A state is a goal when `risk_score(state)` is at or below the user-set threshold AND `missing_submissions` equals zero. The missing-submissions condition is enforced separately so the planner cannot game the threshold by ignoring overdue work.

**Cost:** Every step adds a base cost (1 to 5 depending on the action), plus a fatigue penalty for fatigue-sensitive actions equal to `current_fatigue / 10`, plus a deadline penalty equal to `max(0, 10 - days_to_deadline) * 2`. So the same action gets more expensive as the deadline approaches and as the student gets more tired.

## 3. Heuristic and Cost Design

The risk score is a weighted sum of the bad signals: low attendance, missing submissions, low quiz score, low LMS activity, overwork, sleep deficit, stress, low motivation, and excess work hours. Missing submissions get the heaviest weight (10 per submission) because they are also a hard goal condition.

For the heuristic we needed something admissible so A* would still return optimal plans. We used:

$$h(n) \;=\; \mathit{missing\_submissions} \times 4 \;+\; \frac{\max(0,\; \mathit{risk} - \mathit{threshold})}{10} \times 2$$

The first term uses 4 because the cheapest way to clear a submission is `Submit_Assignment`, with base cost 4. The second term assumes that every 10 risk points can be removed by some action with cost 2, the cheapest improving action. Both estimates underestimate the real remaining cost because they ignore deadline penalties and fatigue, which is what we want for admissibility.

## 4. Results

We tested on `STUDEN_1.CSV`. A typical at-risk student (S017: attendance 76.4%, 2 missing submissions, quiz score 55.6%, lms activity 55.6%, study hours 11.7, days to deadline 12) was solved by A* in 6 steps with a total cost of 45.0. The plan started with two `Submit_Assignment` actions and then mixed in `Attend_Class` and `Meet_Tutor`.

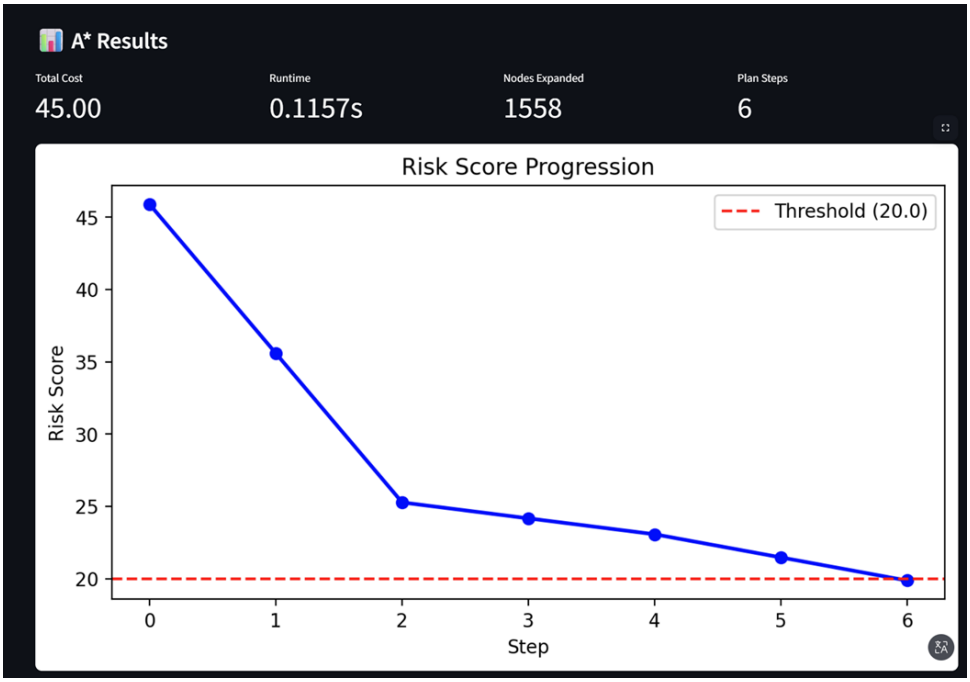


**Figure 4:** A* run on student S017. The metrics row reports total cost 45.00, runtime 0.1157s, 1558 nodes expanded, and 6 plan steps. The line chart shows the risk score dropping from about 45.9 down to the threshold of 20.0 over the six steps.

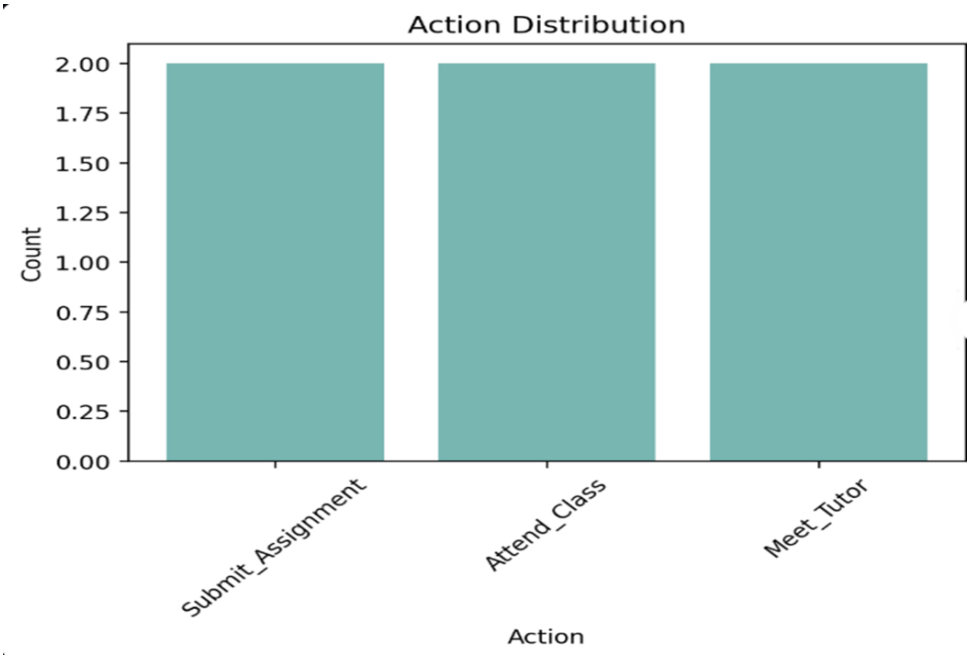


**Figure 5:** Action counts for the A* plan on S017. `Submit_Assignment`, `Attend_Class`, and `Meet_Tutor` each appear twice in the optimal plan.

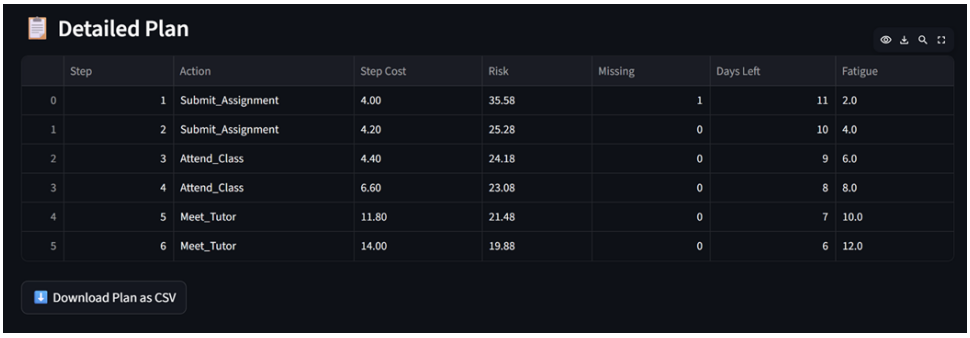


**Figure 6:** Step-by-step plan produced by A* for S017. The plan starts with two `Submit_Assignment` steps to clear missing work, then alternates `Attend_Class` and `Meet_Tutor`. Risk drops from 35.58 to 19.88 and missing submissions reach zero by step 2.

Greedy and UCS behaved very differently. Greedy ran fastest because it ignores $g(n)$, but its plans were sometimes more expensive. UCS matched A* on cost but expanded many more nodes. The numbers below come from a representative run on randomly selected student S017:

| Method | Total Cost | Runtime (s) | Nodes Expanded | Plan Length |
|---|---|---|---|---|
| A* | 45.00 | 0.1157 | 1558 | 6 |
| Greedy | 48.00 | 0.0010 | 6 | 6 |
| UCS | 45.00 | 0.1968 | 2628 | 6 |

**Table 1:** Baseline comparison on student S017.

A* matched UCS on cost and beat it heavily on runtime and node count. Greedy was faster than A* for the particular example, but its plan was about 6.7% more expensive.


**Figure 7:** Baseline comparison panel for student S017. The table reports total cost, runtime, nodes expanded, and plan length for A*, Greedy, and UCS. A* and UCS tie on cost at 45.00, while Greedy comes in at 48.00. The bar chart highlights Greedy as the only met

## 5. What-If Cases

We included three what-if scenarios.

**Tutor Unavailable:** With `Meet_Tutor` disabled, A* shifted to using 5 `Attend_Class` steps and one `Practice_Quiz` step. Total cost rose by roughly 50% on S017 because `Meet_Tutor` was a +5 quiz boost for cost 5, and that ratio is hard to replace.


**Figure 8:** A* result for the "Tutor Unavailable" what-if on S017. With `Meet_Tutor` disabled, total cost rises to 67.60 over 8 steps, and the risk curve takes more iterations to drop below the threshold of 20.0.

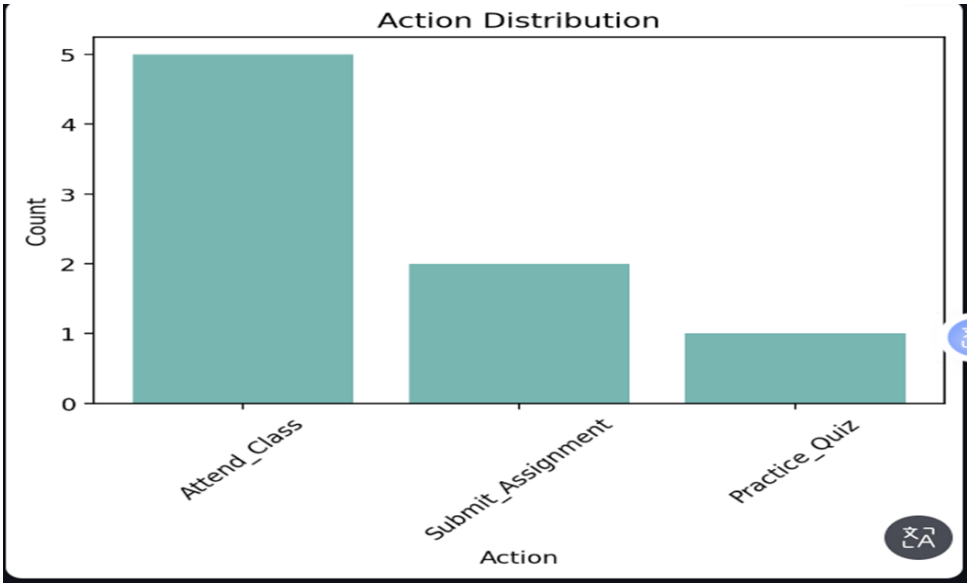


**Figure 9:** Action counts for the Tutor Unavailable plan. The planner replaces the missing `Meet_Tutor` steps with five `Attend_Class` steps and one `Practice_Quiz`.


**Figure 10:** Step-by-step plan for the Tutor Unavailable scenario. The plan is two steps longer than the baseline and step costs grow as fatigue and the deadline penalty stack up.

**Deadline Reduced to 2 Days.** Plans became infeasible for students with three or more actions (since each action consumes one day from `days_to_deadline`) because the deadline term has passed and there are not enough days to complete the work. A* correctly returned no plan rather than a fake one. For lighter loads the plan got shorter but more expensive due to the deadline penalty.

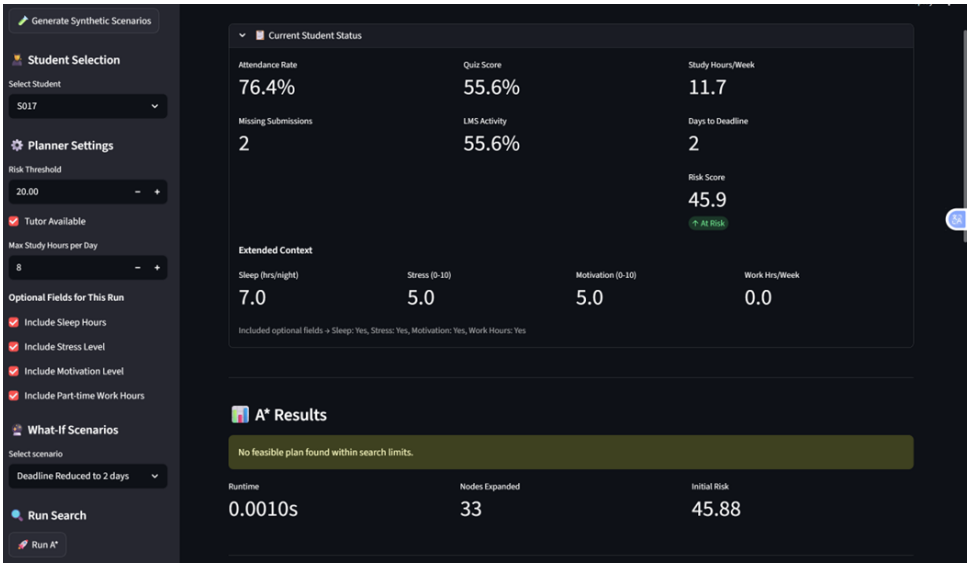


**Figure 11:** Result of the Deadline Reduced to 2 Days what-if on S017. With only two days available, the planner cannot finish the required actions in time and correctly returns no plan rather than a partial one. The dashboard reports 33 nodes expanded before giving up.

**Max Study Hours = 2:** We capped `Study_1_Hour` at two applications per day. For S017 the cap did not bind, because the baseline optimal plan does not use `Study_1_Hour` at all (it relies on `Submit_Assignment`, `Attend_Class`, and `Meet_Tutor`). A* therefore returned the same 6-step plan with total cost 45.00. The cap only changes the result for students whose optimal plan would otherwise contain three or more `Study_1_Hour` steps in one day; for those students the planner is forced to substitute `Practice_Quiz` or `Attend_Class`, which raises the total cost.

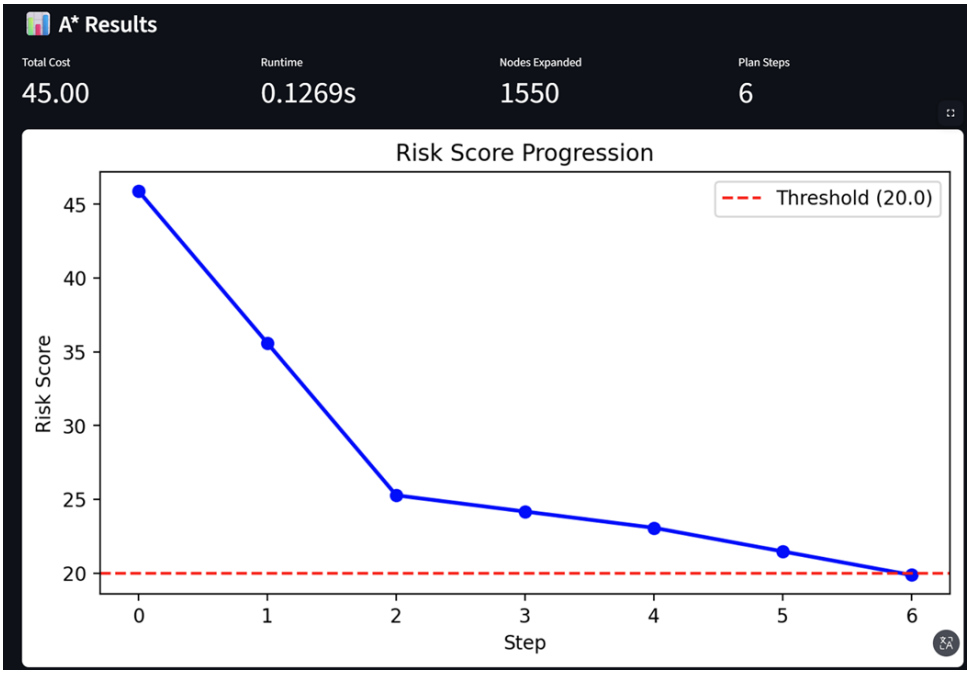


**Figure 12:** A* result for the Max Study Hours = 2 what-if on S017. Total cost stays at 45.00 over 6 steps because the optimal baseline plan does not use any `Study_1_Hour` actions, so the daily study cap does not bind for this student.

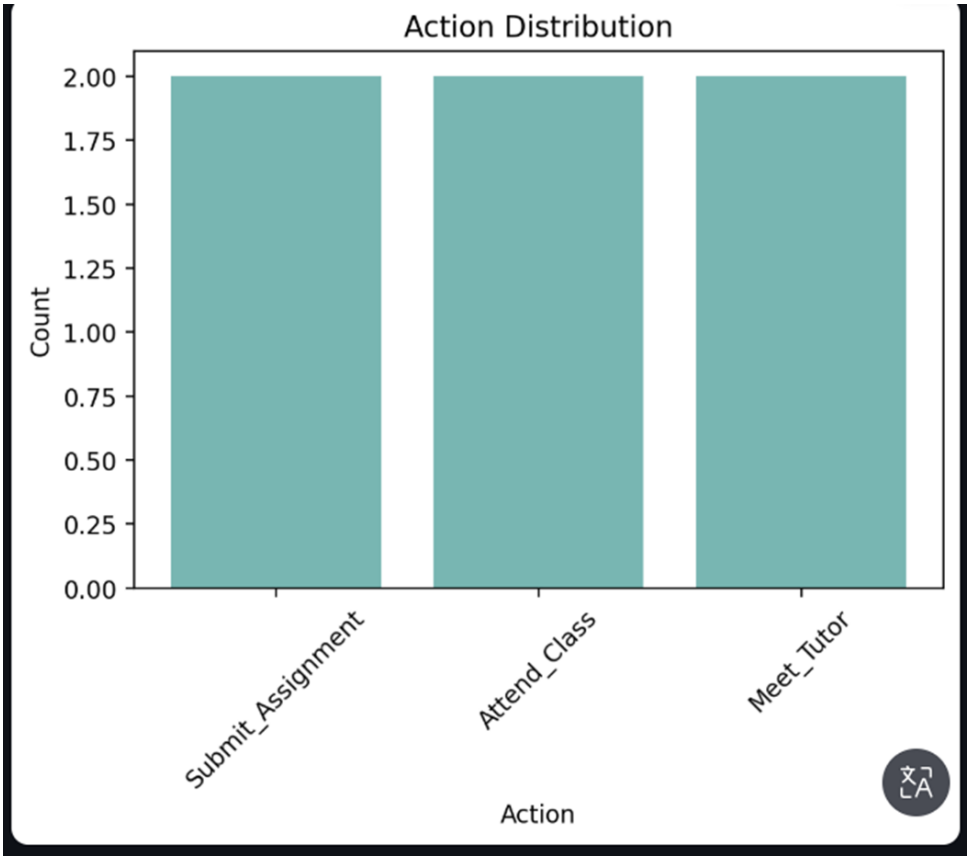


**Figure 13:** Action counts for the Max Study Hours = 2 plan, identical in shape to the baseline because the cap never had to be enforced.

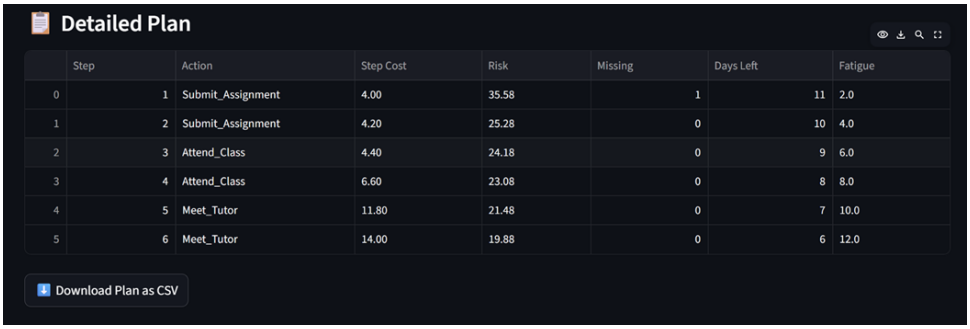


**Figure 14:** Step-by-step plan under the Max Study Hours = 2 constraint, confirming that the planner produced the same six-step sequence as the baseline.

## 6. Acknowledgement of AI Tools

We used ChatGPT and Claude during development for debugging help, refactoring suggestions, and explaining matplotlib syntax. All algorithm design decisions, the cost function, the heuristic, and the final code structure are our own work.