TiEEdu

Project Pulse Guide

How admins, teachers and students use Project Pulse to give C programming tasks, follow every student's GitHub work, and review it, screen by screen.

For TiEEdu admins, teachers and students

Contents

1. How Project Pulse works, in one page
2. Admin: teacher logins and co-teachers
3. Teacher: set up a class (once)
4. Teacher: give a task (every task)
5. Teacher: follow the class on the dashboard
6. Teacher: review students, one click per requirement
7. Teacher: one student in full detail
8. Student: join and set up your repo (once)
9. Student: do a task and see your progress
10. Questions and fixes

1. How Project Pulse works, in one page

A teacher gives a C task. Each student keeps **one GitHub repo for the whole class** and puts each task in its own folder. Pulse reads every repo, so the teacher sees on one screen who started, who is behind, and every line of code. The teacher marks each requirement by hand. There is no AI.

```
c-lab-2026/            ← one repo per student, for the whole class
├── PULSE.txt          ← added once: proves the repo is the student's
├── task-1/  main.c    ← Task 1
├── task-2/  main.c    ← Task 2
└── task-3/  …         ← new task: the student makes this folder and pushes
```

	Once	For every task
Admin	Create each teacher's login	—
Teacher	Create the class, invite students	Pick a template or copy a task, set the deadline, publish; then review with one click per requirement
Student	Join, link one repo, add `PULSE.txt`	Make the folder `task-N/` and push code

What Pulse shows	How it knows
Started or not	Commits that changed the task's folder after the task was assigned
Work over time	Each of those commits: date, time, lines added and removed
The code and what changed	The files in the task's folder, compared with the starter code
Late, bulk and backdated work	Commits after the deadline; one huge commit; dates that don't match
Possible copying	Each student's folder compared with every other student's (renamed variables still match)
Requirements done	**The teacher's marks**: ✓ Done, ~ Partly or ✗ Missing
Test results (optional)	Only if the teacher turns on *Auto-run tests* for that task

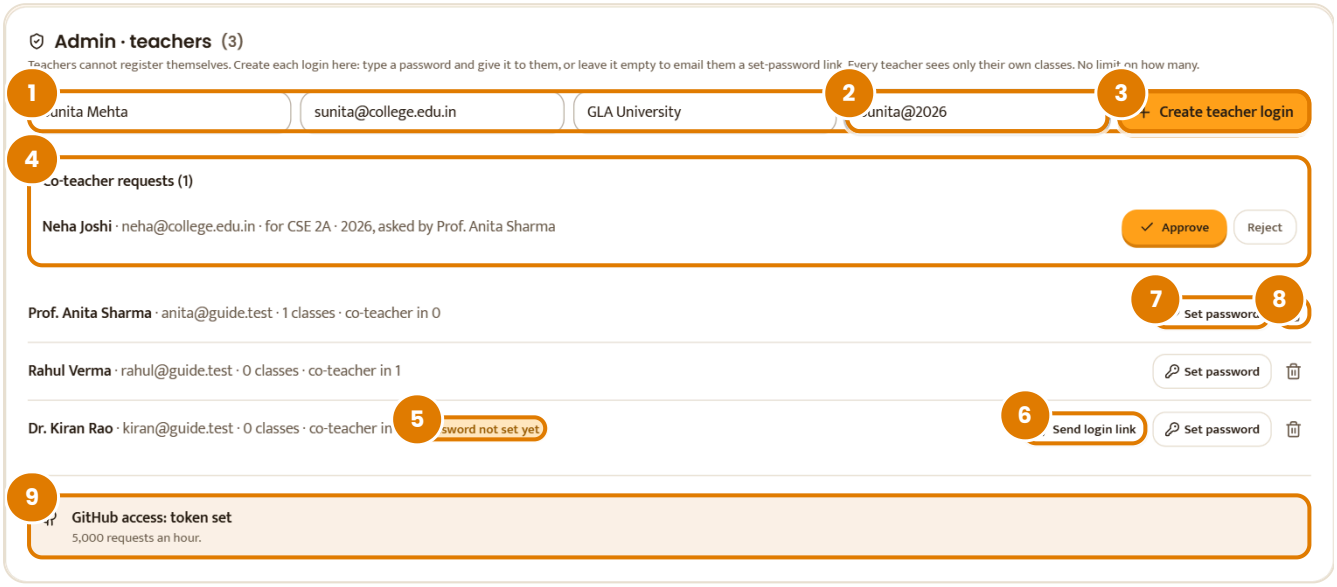
> **WHO SEES WHAT**
> Admins see everything. A teacher sees only their own classes. A student sees only their own work: never another student's name, code, score or rank, and never the teacher's private notes or copy flags.

WHEN IT UPDATES

Every 20 minutes automatically, when the teacher presses **Sync now**, and when a student presses **Re-check my repo**.

2. Admin: teacher logins and co-teachers

Teachers can't sign themselves up. Only an admin creates their login. Open **Project Pulse → Teacher portal** (tieedu.in/project-pulse/teacher) and scroll to **Admin · teachers**.



Admin · teachers, at the bottom of the Teacher portal (admins only).

1. Type the teacher's **name, email and college**.
2. **Password** (8+ characters): you set it and give it to the teacher yourself. Leave it empty to email them a one-time "Set my password" link (valid 7 days) instead.
3. **Create teacher login**. The teacher can now sign in at tieedu.in/login.
4. **Co-teacher requests** from teachers. **Approve** creates the person's login and adds them to that class; **Reject** discards it.
5. **Password not set yet**: this teacher hasn't used their emailed link.
6. **Send login link**: email a fresh set-password link and show it to copy.
7. **Set password**: give a teacher a new password (for example if they forgot theirs).
8. **Remove** a teacher. They lose the teacher portal at once; their classes stay.
9. Shows whether the GitHub token is set on the server (needed for 5,000 GitHub requests an hour).

HOW MANY TEACHERS

No limit. Each teacher has their own portal and sees only their own classes.

3. Teacher: set up a class (once)

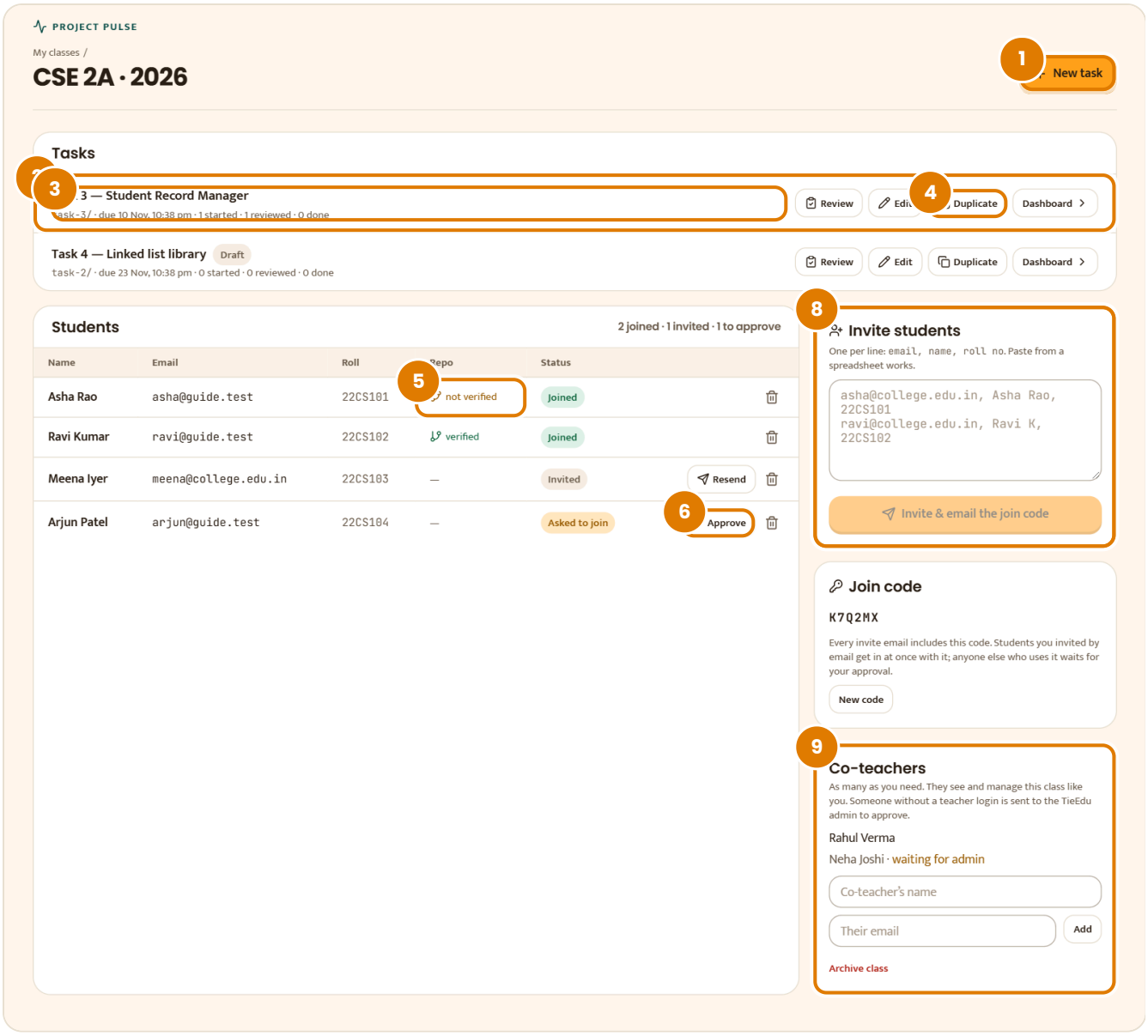
1. Sign in at **tieedu.in/login** with the email and password the admin gave you.
2. Open **Project Pulse → Teacher portal**.
3. In **New class**, type a name (for example “CSE 2A · 2026”) and a subject, then click **Create class**.



My classes: your teacher home.

1. A class card: open it to manage students and tasks.
2. Students who used the join code and are waiting for your approval.
3. **New class**: create another class.

Open the class. Everything about it is on one page:

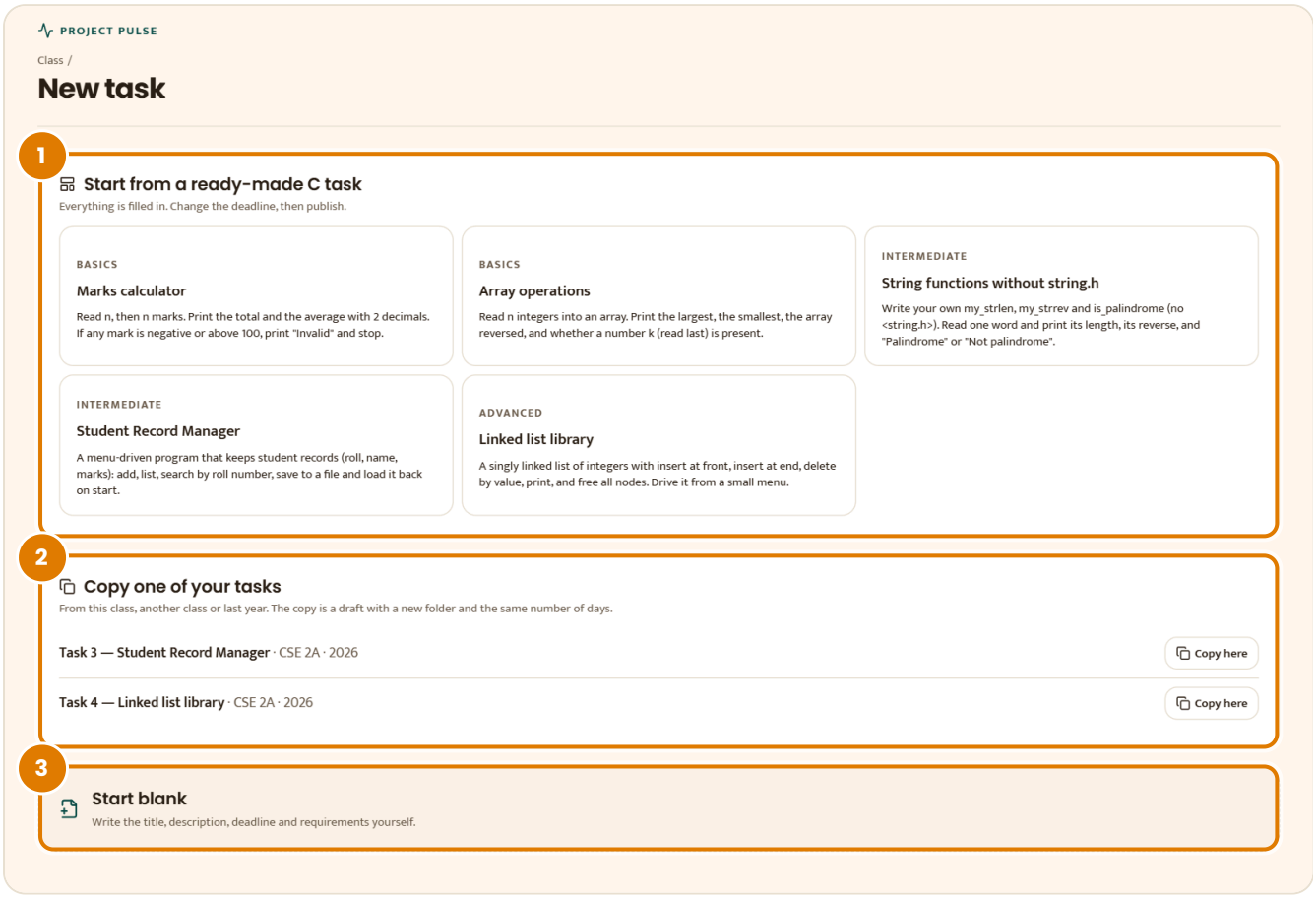


The class page.

1. **New task**: give the class a task (chapter 4).
2. Each task with its folder, deadline, and how many students have started, been reviewed or finished.
3. **Review**: go straight to Review mode for this task.
4. **Duplicate**: copy the task as a draft, with a new folder and the same number of days.
5. **Repo**: whether each student has linked their repo and proved it is theirs.
6. **Approve** a student who asked to join with the code.
7. **Invite students**: paste one line per student, `email, name, roll no`. Each gets an email with a one-time join link and the class code. The links also appear here to share on WhatsApp.
8. **Join code**: students can type it at tieedu.in/project-pulse/join. Invited emails get in at once; anyone else waits for your approval. **New code** stops the old one working.
9. **Co-teachers** (any number). An existing teacher is added at once; anyone else goes to the admin to approve.

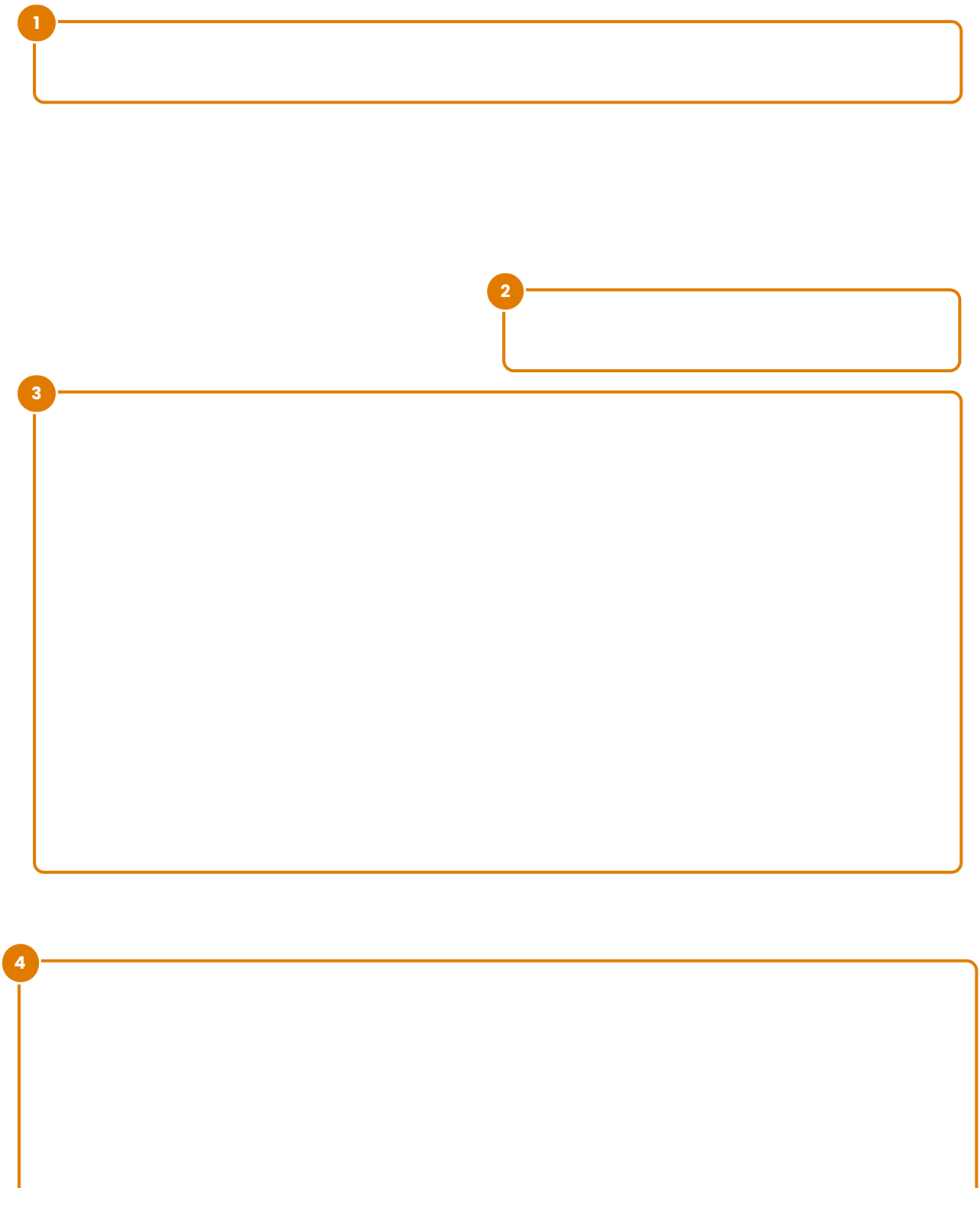
4. Teacher: give a task (every task, about 3 minutes)

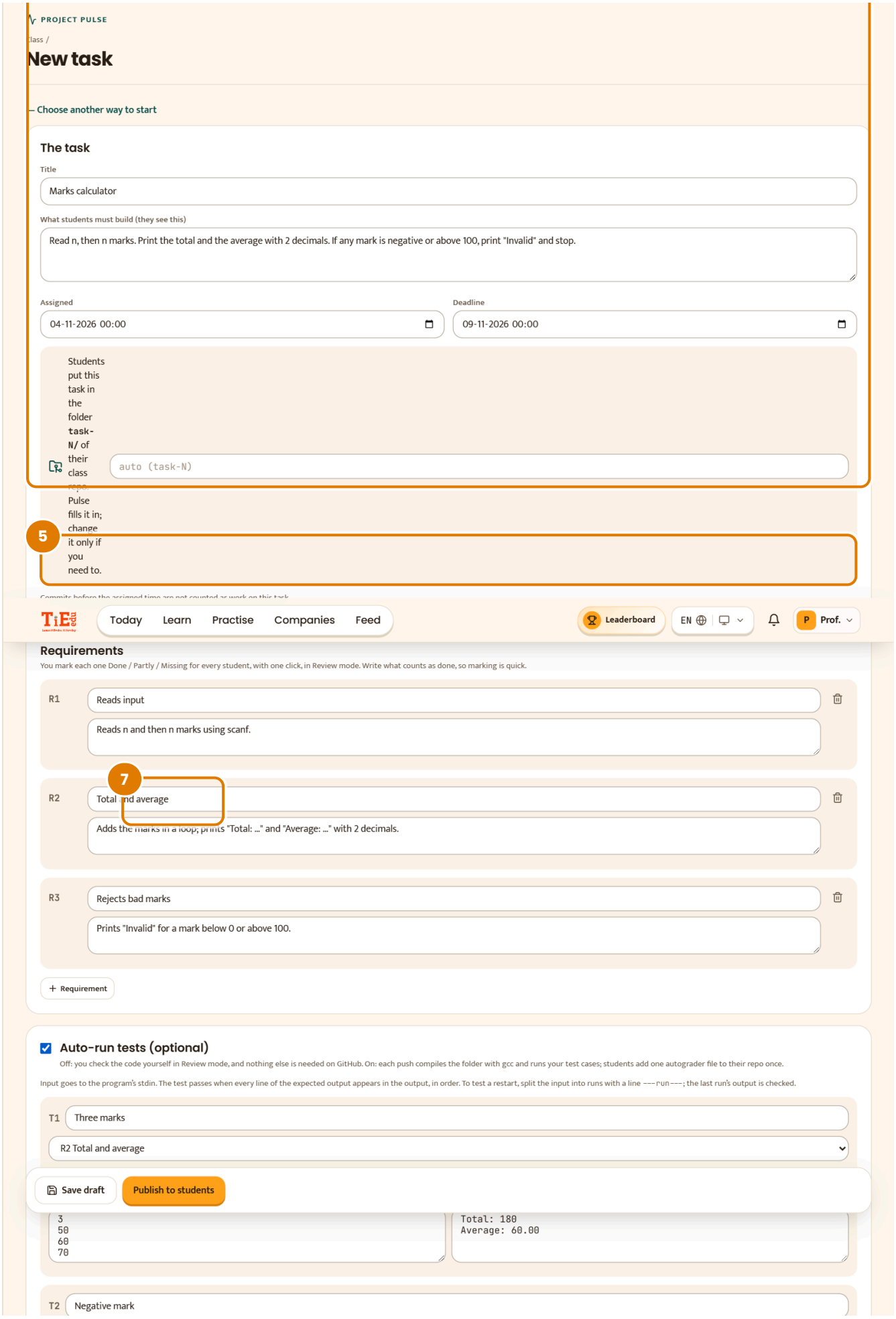
On the class page, click **New task**. You never make a GitHub repo.

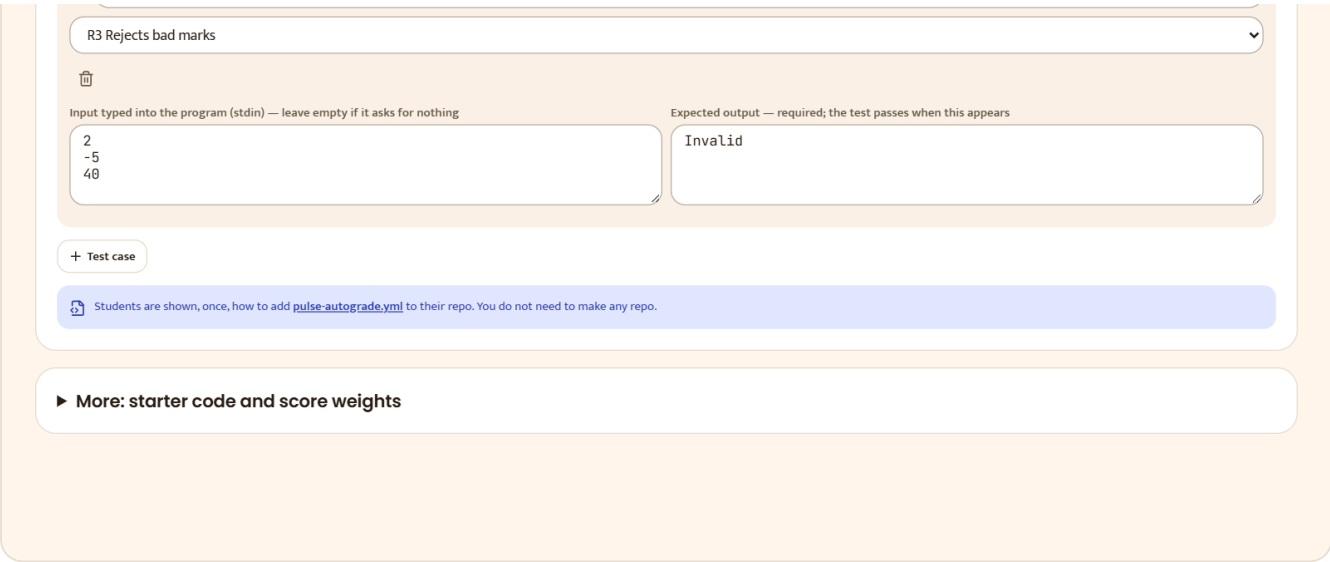


Three ways to start a task.

1. **Ready-made C tasks** (Basics to Advanced). Everything is filled in; change the deadline and publish.
2. **Copy one of your tasks** from this class, another class or last year.
3. **Start blank** and write it yourself.


1
2
3
4


PROJECT PULSE
New task
– Choose another way to start
The task
Title
Marks calculator
What students must build (they see this)
Read n, then n marks. Print the total and the average with 2 decimals. If any mark is negative or above 100, print "Invalid" and stop.
Assigned
04-11-2026 00:00
Deadline
09-11-2026 00:00
Students put this task in the folder task-N/ of their class repo. Pulse fills it in; change it only if you need to.
auto (task-N)
5
Today
Learn
Practise
Companies
Feed
Leaderboard
EN
Prof.
Requirements
You mark each one Done / Partly / Missing for every student, with one click, in Review mode. Write what counts as done, so marking is quick.
R1
Reads input
Reads n and then n marks using scanf.
7
R2
Total and average
Adds the marks in a loop, prints "Total: ..." and "Average: ..." with 2 decimals.
R3
Rejects bad marks
Prints "Invalid" for a mark below 0 or above 100.
+ Requirement
Auto-run tests (optional)
Off: you check the code yourself in Review mode, and nothing else is needed on GitHub. On: each push compiles the folder with gcc and runs your test cases; students add one autograder file to their repo once.
Input goes to the program's stdin. The test passes when every line of the expected output appears in the output, in order. To test a restart, split the input into runs with a line ---run---; the last run's output is checked.
T1
Three marks
R2 Total and average
Save draft
Publish to students
3
50
60
70
Total: 180
Average: 60.00
T2
Negative mark



The task editor (here started from the "Marks calculator" template, with tests switched on).

1. **Title** and what students must build. Students see this.
2. **Assigned** and **Deadline**. Commits before the assigned time don't count; commits after the deadline are marked late.
3. **Folder**: students put this task in `task-N/`. Pulse fills it in for you.
4. **Requirements**: one line each, with what counts as done. You mark these later with one click.
5. **Auto-run tests** (optional, off by default). Off: you just read the code. On: every push compiles the folder and runs your tests.
7. **Publish to students**. Until then the task is a draft only you can see.

TEST CASES, IF YOU USE THEM

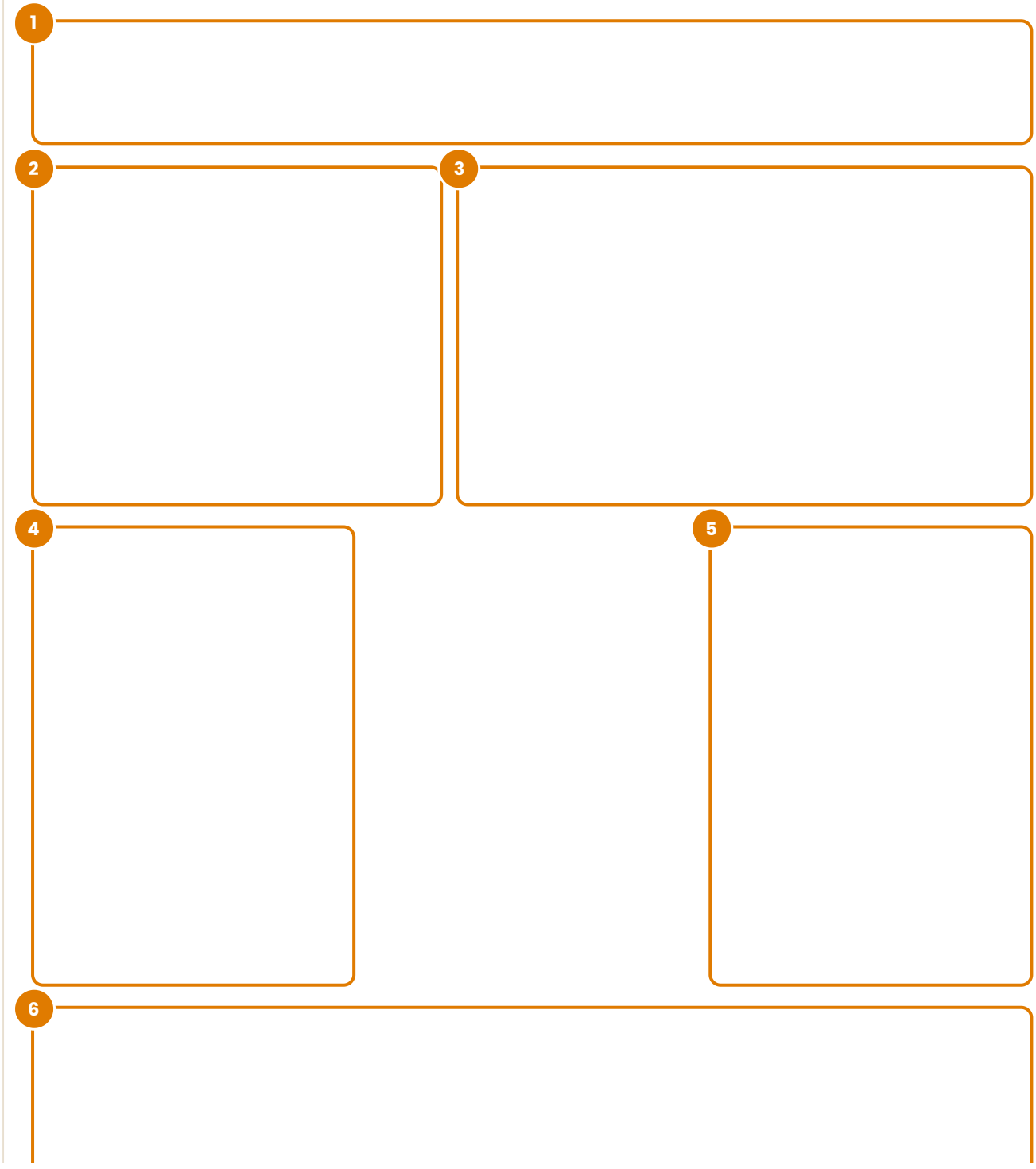
A test passes when every line of the expected output appears in the program's output, in order (extra prompts are fine). Leave the input empty if the program asks for nothing. To test a restart, split the input with a line `---run---`.

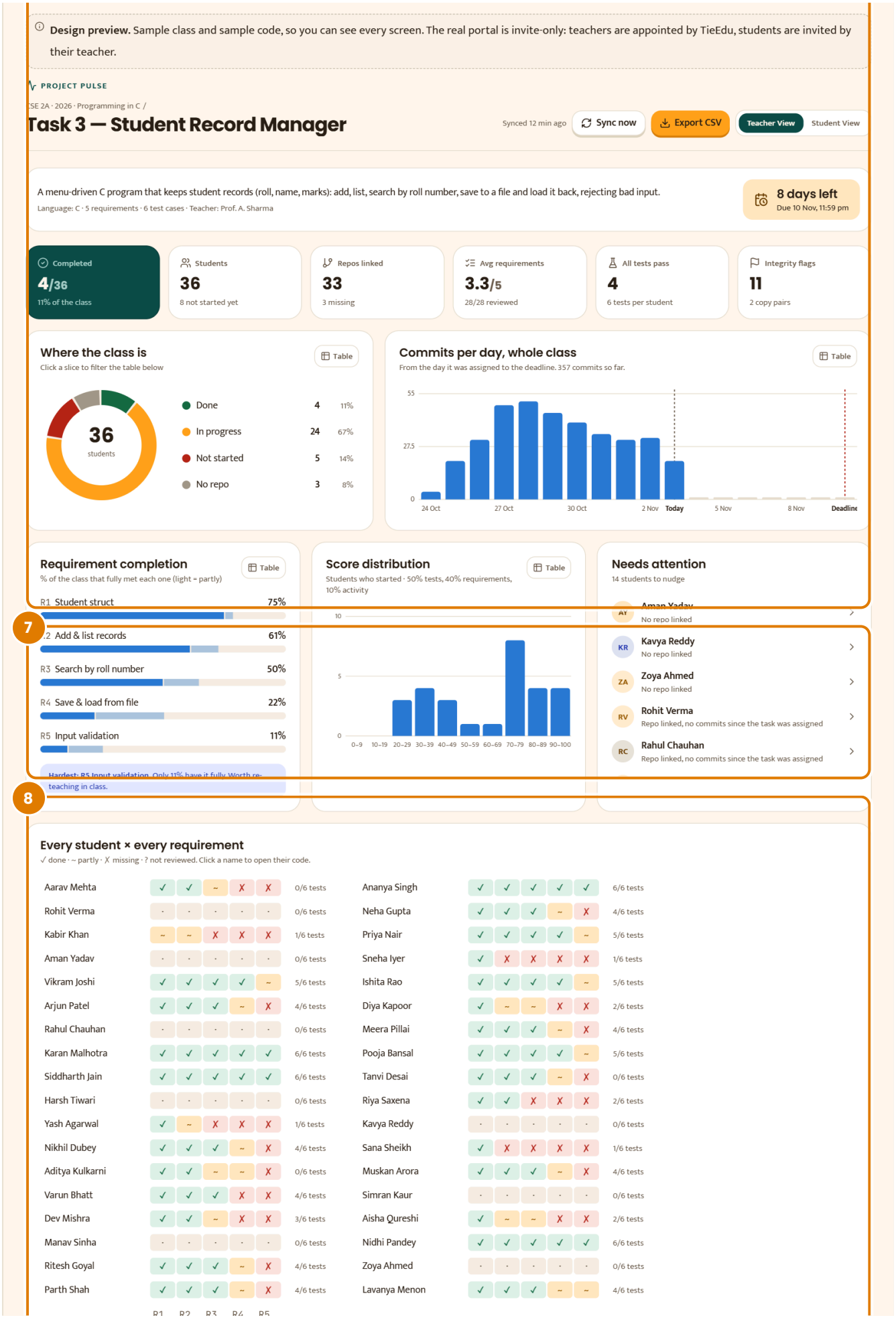
AFTER PUBLISHING

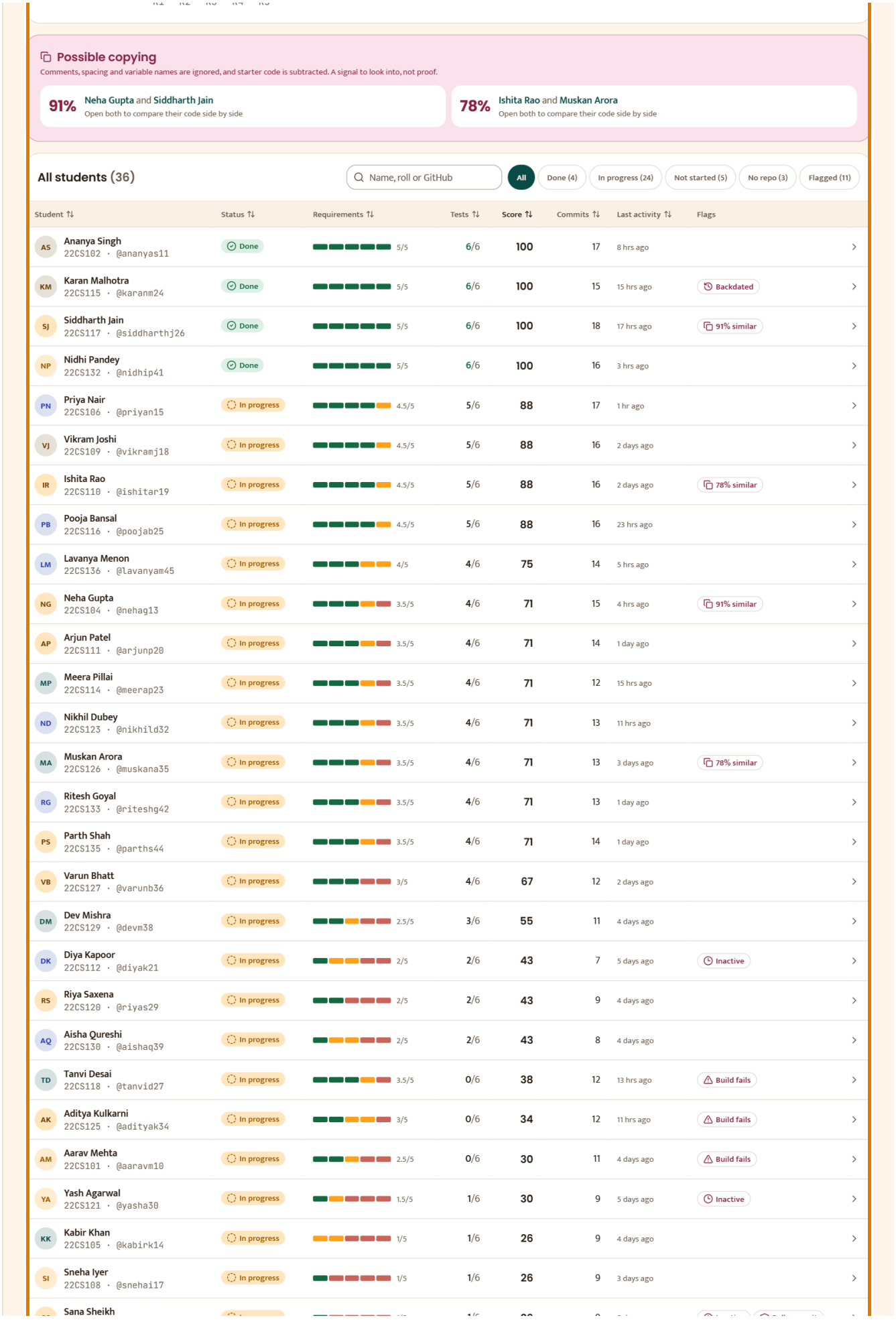
Tell students: "Task 3 is up. Put your code in the folder **task-3/** of your class repo and push." That's all they need.

5. Teacher: follow the class on the dashboard

Class page → click the task. Here is a full class (sample data):


1
2
3
4
5
6

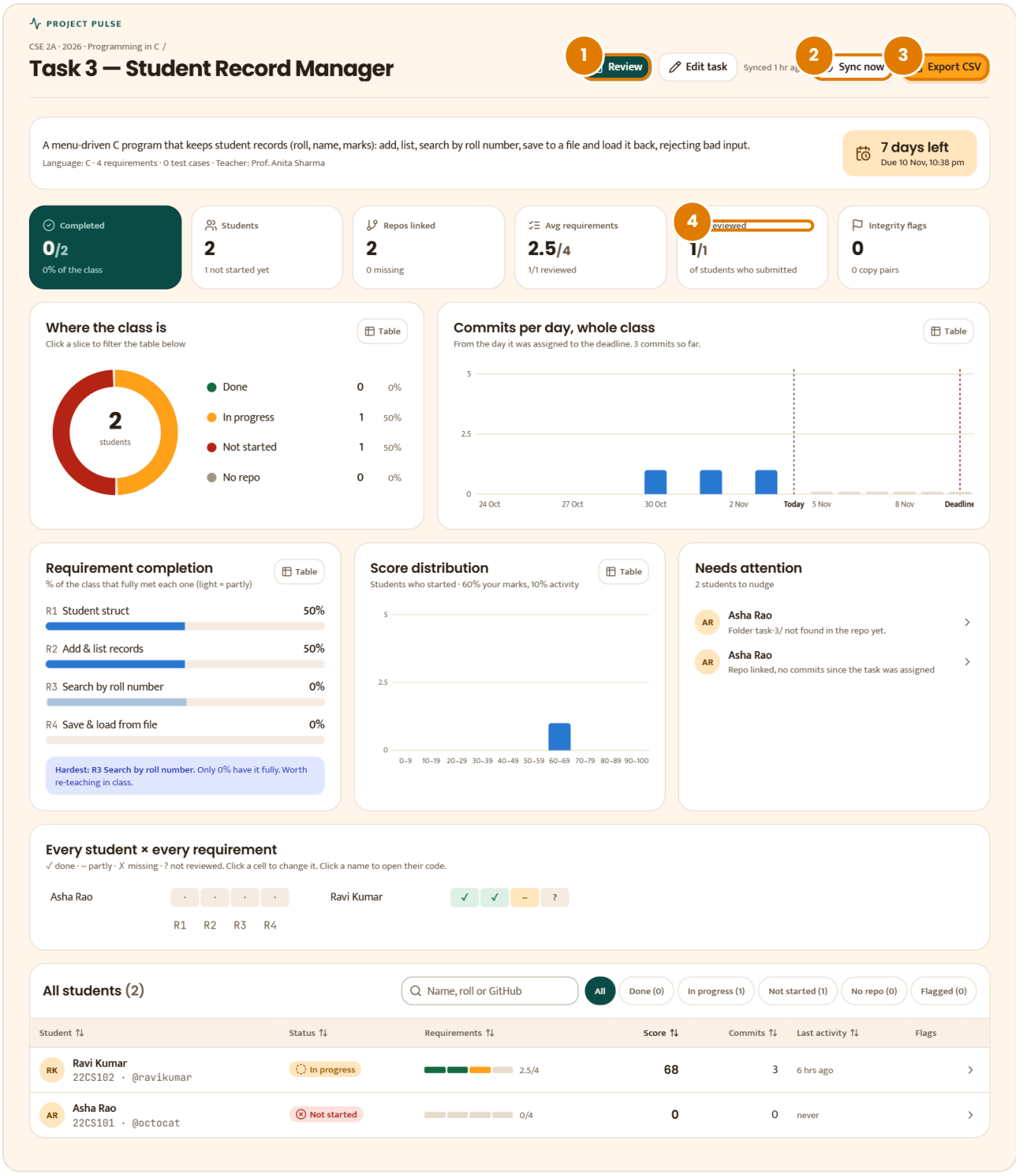

Design preview. Sample class and sample code, so you can see every screen. The real portal is invite-only: teachers are appointed by TieEdu, students are invited by their teacher.
PROJECT PULSE
CSE 2A · 2026 · Programming in C /
Task 3 — Student Record Manager
Synced 12 min ago
Sync now
Export CSV
Teacher View
Student View
A menu-driven C program that keeps student records (roll, name, marks): add, list, search by roll number, save to a file and load it back, rejecting bad input.
Language: C · 5 requirements · 6 test cases · Teacher: Prof. A. Sharma
8 days left
Due 10 Nov, 11:59 pm
Completed
4/36
11% of the class
Students
36
8 not started yet
Repos linked
33
3 missing
Avg requirements
3.3/5
28/28 reviewed
All tests pass
4
6 tests per student
Integrity flags
11
2 copy pairs
Where the class is
Click a slice to filter the table below
Table
36
students
Done 4 11%
In progress 24 67%
Not started 5 14%
No repo 3 8%
Commits per day, whole class
From the day it was assigned to the deadline. 357 commits so far.
Table
55
27.5
0
24 Oct
27 Oct
30 Oct
2 Nov
Today
5 Nov
8 Nov
Deadline
Requirement completion
% of the class that fully met each one (light = partly)
Table
R1 Student struct 75%
R2 Add & list records 61%
R3 Search by roll number 50%
R4 Save & load from file 22%
R5 Input validation 11%
Hardest: R5 Input validation. Only 11% have it fully. Worth re-teaching in class.
Score distribution
Students who started · 50% tests, 40% requirements, 10% activity
Table
10
5
0
0–9
10–19
20–29
30–39
40–49
50–59
60–69
70–79
80–89
90–100
Needs attention
14 students to nudge
No repo linked
KR Kavya Reddy
No repo linked
ZA Zoya Ahmed
No repo linked
RV Rohit Verma
Repo linked, no commits since the task was assigned
RC Rahul Chauhan
Repo linked, no commits since the task was assigned
7
8
Every student × every requirement
✓ done · ~ partly · ✗ missing · ? not reviewed. Click a name to open their code.
Aarav Mehta 0/6 tests
Rohit Verma 0/6 tests
Kabir Khan 1/6 tests
Aman Yadav 0/6 tests
Vikram Joshi 5/6 tests
Arjun Patel 4/6 tests
Rahul Chauhan 0/6 tests
Karan Malhotra 6/6 tests
Siddharth Jain 6/6 tests
Harsh Tiwari 0/6 tests
Yash Agarwal 1/6 tests
Nikhil Dubey 4/6 tests
Aditya Kulkarni 0/6 tests
Varun Bhatt 4/6 tests
Dev Mishra 3/6 tests
Manav Sinha 0/6 tests
Ritesh Goyal 4/6 tests
Parth Shah 4/6 tests
Ananya Singh 6/6 tests
Neha Gupta 4/6 tests
Priya Nair 5/6 tests
Sneha Iyer 1/6 tests
Ishita Rao 5/6 tests
Diya Kapoor 2/6 tests
Meera Pillai 4/6 tests
Pooja Bansal 5/6 tests
Tanvi Desai 0/6 tests
Riya Saxena 2/6 tests
Kavya Reddy 0/6 tests
Sana Sheikh 1/6 tests
Muskan Arora 4/6 tests
Simran Kaur 0/6 tests
Aisha Qureshi 2/6 tests
Nidhi Pandey 6/6 tests
Zoya Ahmed 0/6 tests
Lavanya Menon 4/6 tests
R1 R2 R3 R4 R5


Possible copying
Comments, spacing and variable names are ignored, and starter code is subtracted. A signal to look into, not proof.
91% Neha Gupta and Siddharth Jain
Open both to compare their code side by side
78% Ishita Rao and Muskan Arora
Open both to compare their code side by side
All students (36)
Name, roll or GitHub
All
Done (4)
In progress (24)
Not started (5)
No repo (3)
Flagged (11)
Student
Status
Requirements
Tests
Score
Commits
Last activity
Flags
Ananya Singh 22CS102 · @ananyas11 Done 5/5 6/6 100 17 8 hrs ago
Karan Malhotra 22CS115 · @karanm24 Done 5/5 6/6 100 15 15 hrs ago Backdated
Siddharth Jain 22CS117 · @siddharthj26 Done 5/5 6/6 100 18 17 hrs ago 91% similar
Nidhi Pandey 22CS132 · @nidhip41 Done 5/5 6/6 100 16 3 hrs ago
Priya Nair 22CS106 · @priyan15 In progress 4.5/5 5/6 88 17 1 hr ago
Vikram Joshi 22CS109 · @vikramj18 In progress 4.5/5 5/6 88 16 2 days ago
Ishita Rao 22CS110 · @ishitar19 In progress 4.5/5 5/6 88 16 2 days ago 78% similar
Pooja Bansal 22CS116 · @poojab25 In progress 4.5/5 5/6 88 16 23 hrs ago
Lavanya Menon 22CS136 · @lavanyam45 In progress 4/5 4/6 75 14 5 hrs ago
Neha Gupta 22CS104 · @nehag13 In progress 3.5/5 4/6 71 15 4 hrs ago 91% similar
Arjun Patel 22CS111 · @arjunp20 In progress 3.5/5 4/6 71 14 1 day ago
Meera Pillai 22CS114 · @meerap23 In progress 3.5/5 4/6 71 12 15 hrs ago
Nikhil Dubey 22CS123 · @nikhild32 In progress 3.5/5 4/6 71 13 11 hrs ago
Muskan Arora 22CS126 · @muskana35 In progress 3.5/5 4/6 71 13 3 days ago 78% similar
Ritesh Goyal 22CS133 · @riteshg42 In progress 3.5/5 4/6 71 13 1 day ago
Parth Shah 22CS135 · @parths44 In progress 3.5/5 4/6 71 14 1 day ago
Varun Bhatt 22CS127 · @varunb36 In progress 3/5 4/6 67 12 2 days ago
Dev Mishra 22CS129 · @devm38 In progress 2.5/5 3/6 55 11 4 days ago
Diya Kapoor 22CS112 · @diyak21 In progress 2/5 2/6 43 7 5 days ago Inactive
Riya Saxena 22CS120 · @riyas29 In progress 2/5 2/6 43 9 4 days ago
Aisha Qureshi 22CS130 · @aishaq39 In progress 2/5 2/6 43 8 4 days ago
Tanvi Desai 22CS118 · @tanvid27 In progress 3.5/5 0/6 38 12 13 hrs ago Build fails
Aditya Kulkarni 22CS125 · @adityak34 In progress 3/5 0/6 34 12 11 hrs ago Build fails
Aarav Mehta 22CS101 · @aaravm10 In progress 2.5/5 0/6 30 11 4 days ago Build fails
Yash Agarwal 22CS121 · @yasha30 In progress 1.5/5 1/6 30 9 5 days ago Inactive
Kabir Khan 22CS105 · @kabirk14 In progress 1/5 1/6 26 9 4 days ago
Sneha Iyer 22CS108 · @snehai17 In progress 1/5 1/6 26 9 3 days ago
Sana Sheikh



The task dashboard.

1. At a glance: completed, students, repos linked, average requirements met, reviewed, integrity flags.
2. **Where the class is**: click a slice to filter the table below.
3. **Commits per day** from assignment to deadline. Is the class working, or waiting for the last night?
4. **Requirement completion**: the hardest requirement is named, so you know what to re-teach.
5. **Needs attention**: no repo, not started, inactive, wrong folder name, waiting for review.
6. **Every student × every requirement** (✓ ~ ✗ ?). On your live dashboard, click a cell to change the mark.
7. **Possible copying**: pairs of students whose task folders match. A signal to check, not proof.
8. **All students**: search, filter and sort; click a student to open their work.

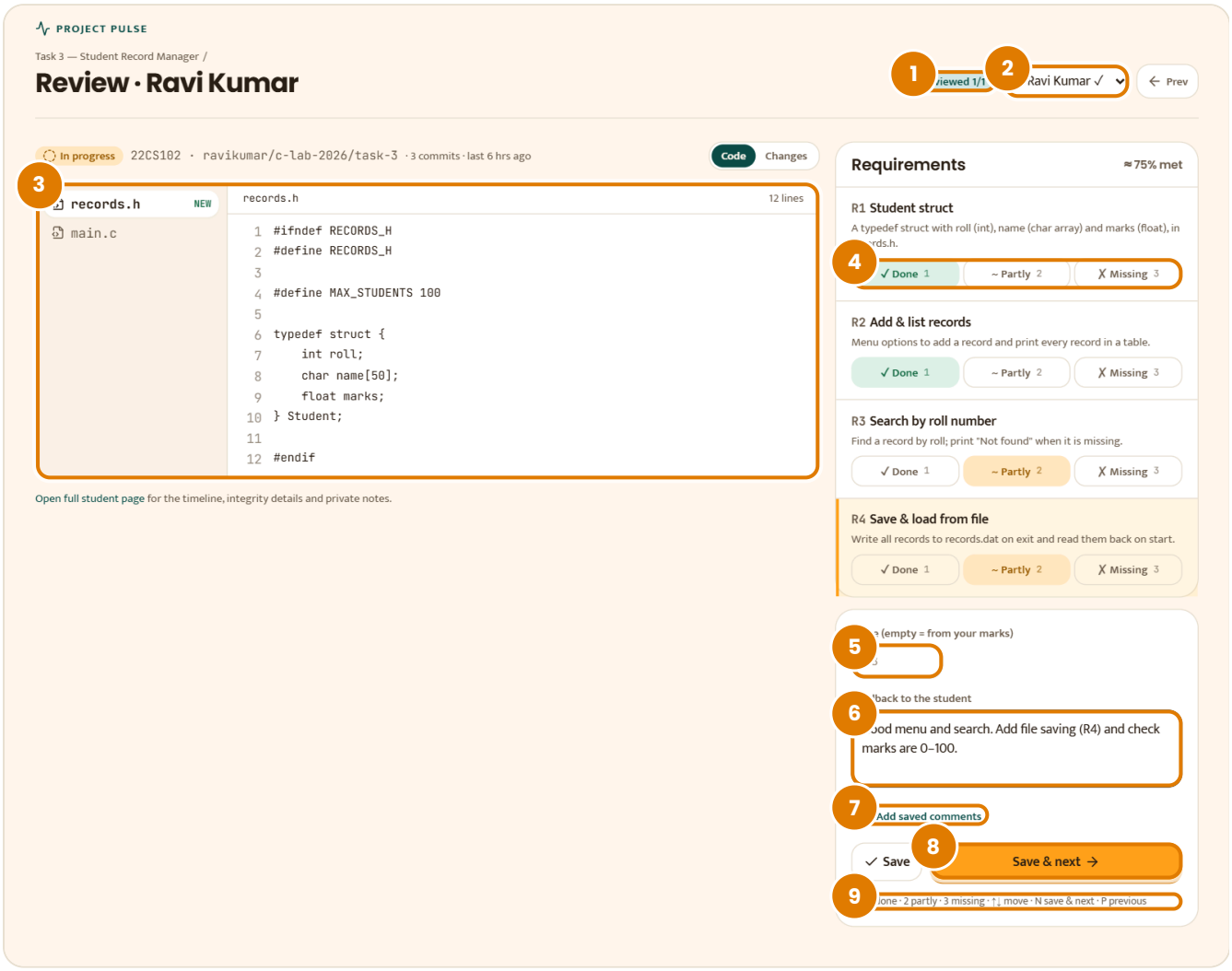


The buttons at the top of your live dashboard.

1. **Review**: open Review mode (chapter 6).
2. **Sync now**: check every student's repo right away.
3. **Export CSV**: the table, for your marks sheet.
4. **Reviewed**: how many students who submitted you've marked.

6. Teacher: review students, one click per requirement

Click **Review**. Students who have pushed code appear one at a time, the ones you haven't reviewed yet first.



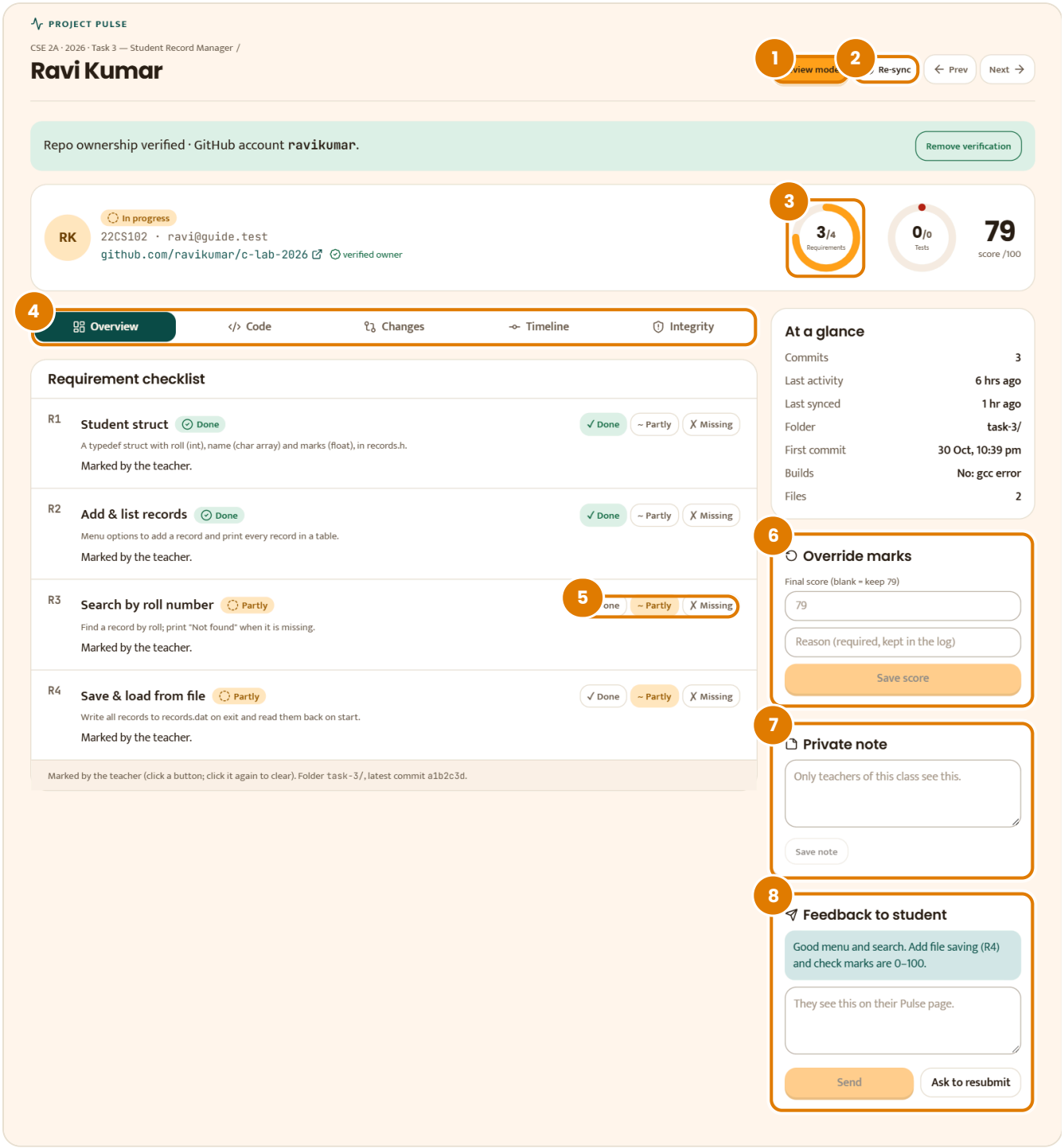
Review mode.

1. How many you've reviewed so far.
2. Jump to any student (✓ marks the ones already reviewed).
3. Their code in the task folder, with line numbers. **Changes** shows what they added.
4. Mark each requirement: **✓ Done**, **~ Partly** or **✗ Missing**. Click again to clear.
5. **Score**: leave empty to use your marks, or type your own.
6. **Feedback** the student will see on their page.
7. **Saved comments**: your own one-click lines, such as "Check fopen for NULL".
8. **Save & next**: saves and opens the next student.
9. Keyboard: **1** done · **2** partly · **3** missing (for the highlighted requirement, then it moves on) · **N** save & next · **P** previous.

TYPICAL SPEED

Read the code, press 1/2/3 for each requirement, press N. About a minute per student.

7. Teacher: one student in full detail

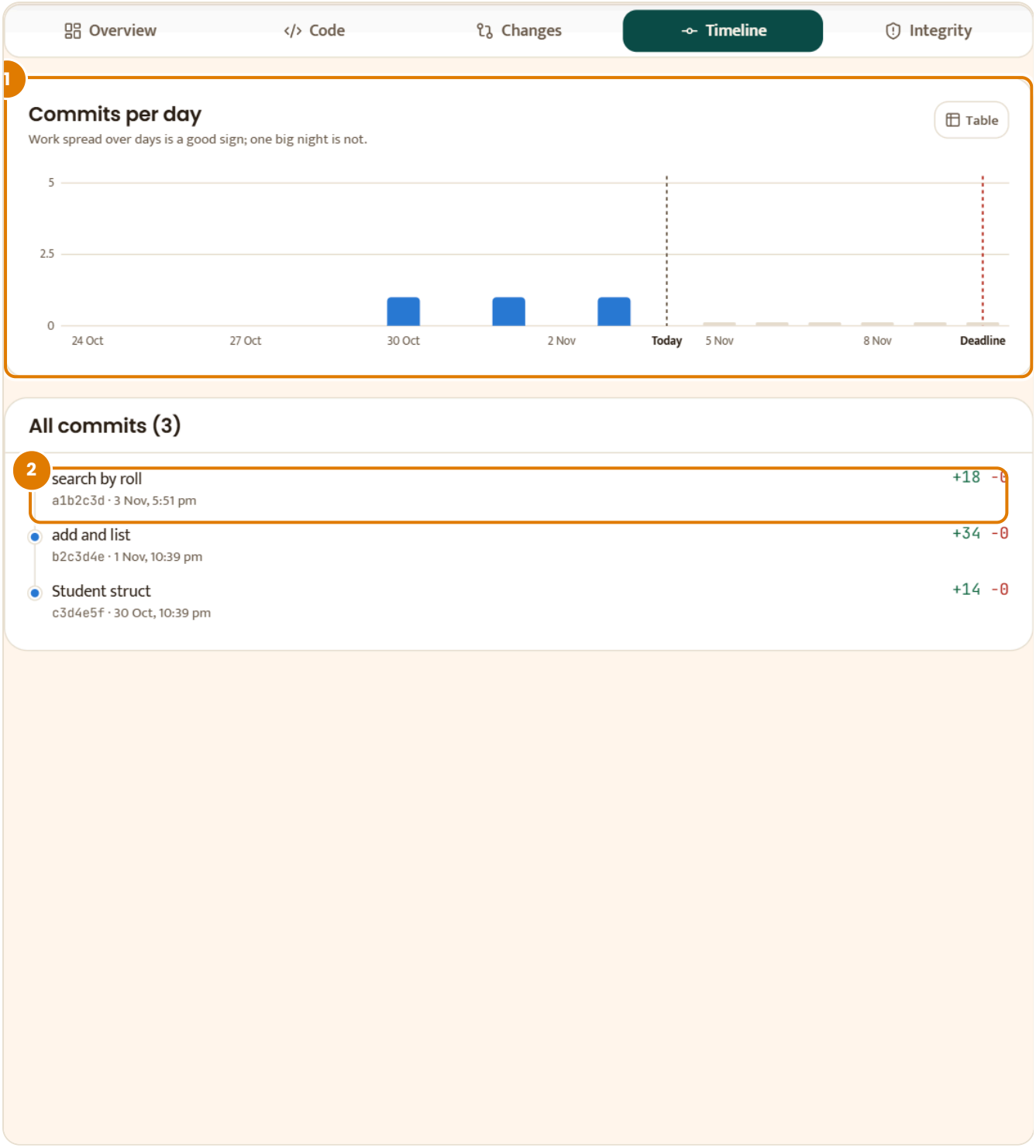


A student's page (open it from the table, the grid, or Review mode).

1. **Review mode** for this student.
2. **Re-sync**: check this student's repo now.
3. Requirements met so far (from your marks).
4. Tabs: **Overview**, **Code**, **Changes**, **Timeline**, **Integrity**.
5. Mark a requirement right here.
6. **Override marks**: set a final score with a reason (kept in the log).

7 **Private note**: only teachers of the class see it.

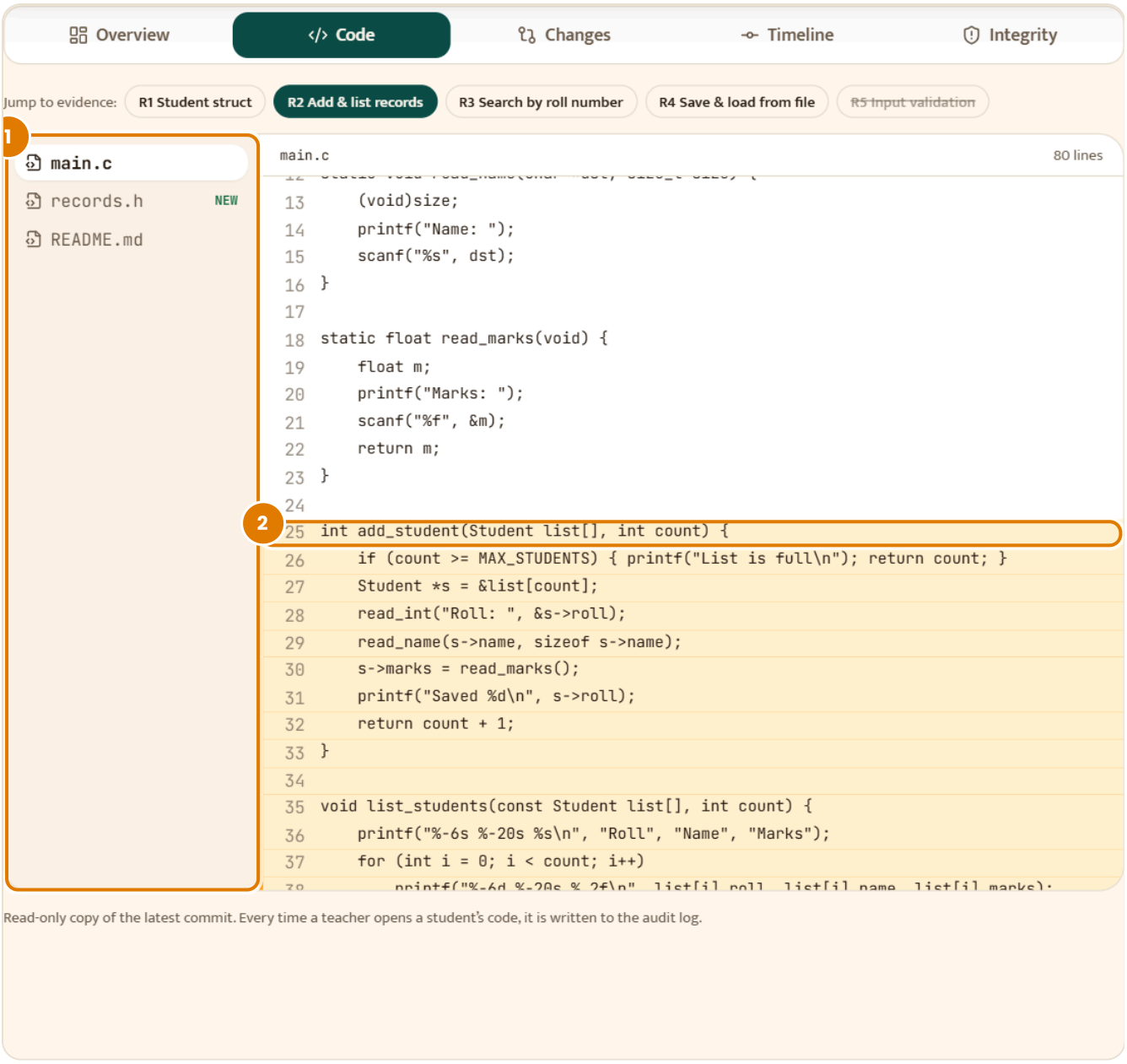
8 **Feedback to student**, and **Ask to resubmit**.



Timeline tab: when they actually worked.

1 Commits per day in the task folder.

2 Every commit: message, time, lines added and removed.



Code tab: the student's files, read-only.

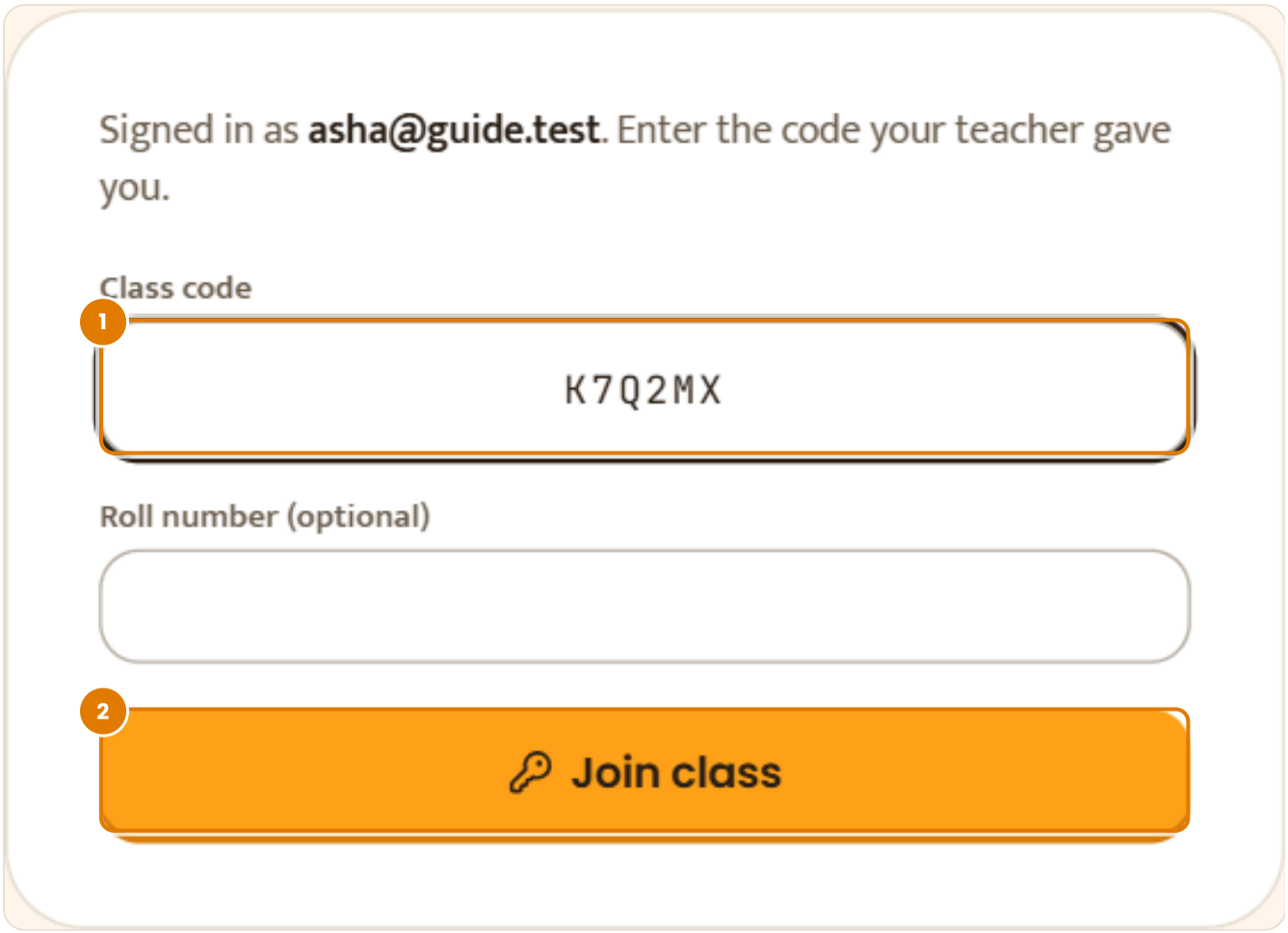
1. The files in their task folder.
2. Highlighted lines.

INTEGRITY TAB

Lists copying (with whom, and how similar), one bulk commit, backdated or late commits, and a repo that belongs to someone else. These are signals for you to check, never automatic penalties. If a student can't add PULSE.txt, check the repo with them and click **Verify myself**.

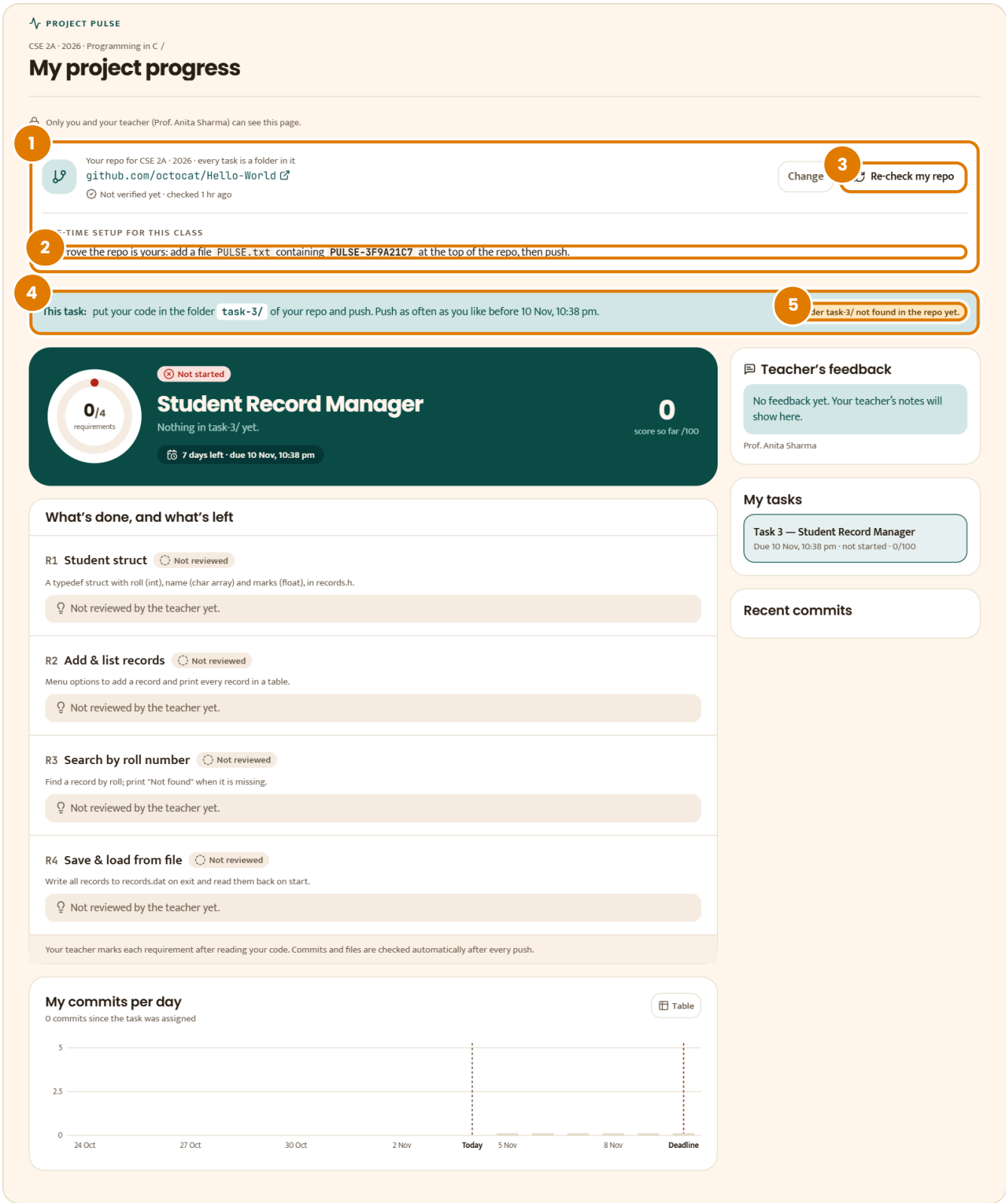
8. Student: join and set up your repo (once)

1. Open the invite email and click **Join the class**, or go to **tieedu.in/project-pulse/join** and type your class code.
2. Sign in, or sign up, with **the same email your teacher used**. You see “You’re in”.
3. On GitHub, make **one public repo** for the whole class, for example `c-lab-2026`.
4. Open **tieedu.in/project-pulse/student**, paste the repo link and click **Link repo**.
5. Add a file named `PULSE.txt` to the repo containing the code Pulse shows you, then push and click **Re-check my repo**.



Joining with a class code.

1. Type the class code from your teacher.
2. **Join class**. With a code you wait for your teacher’s approval; with the invite link you get in at once.



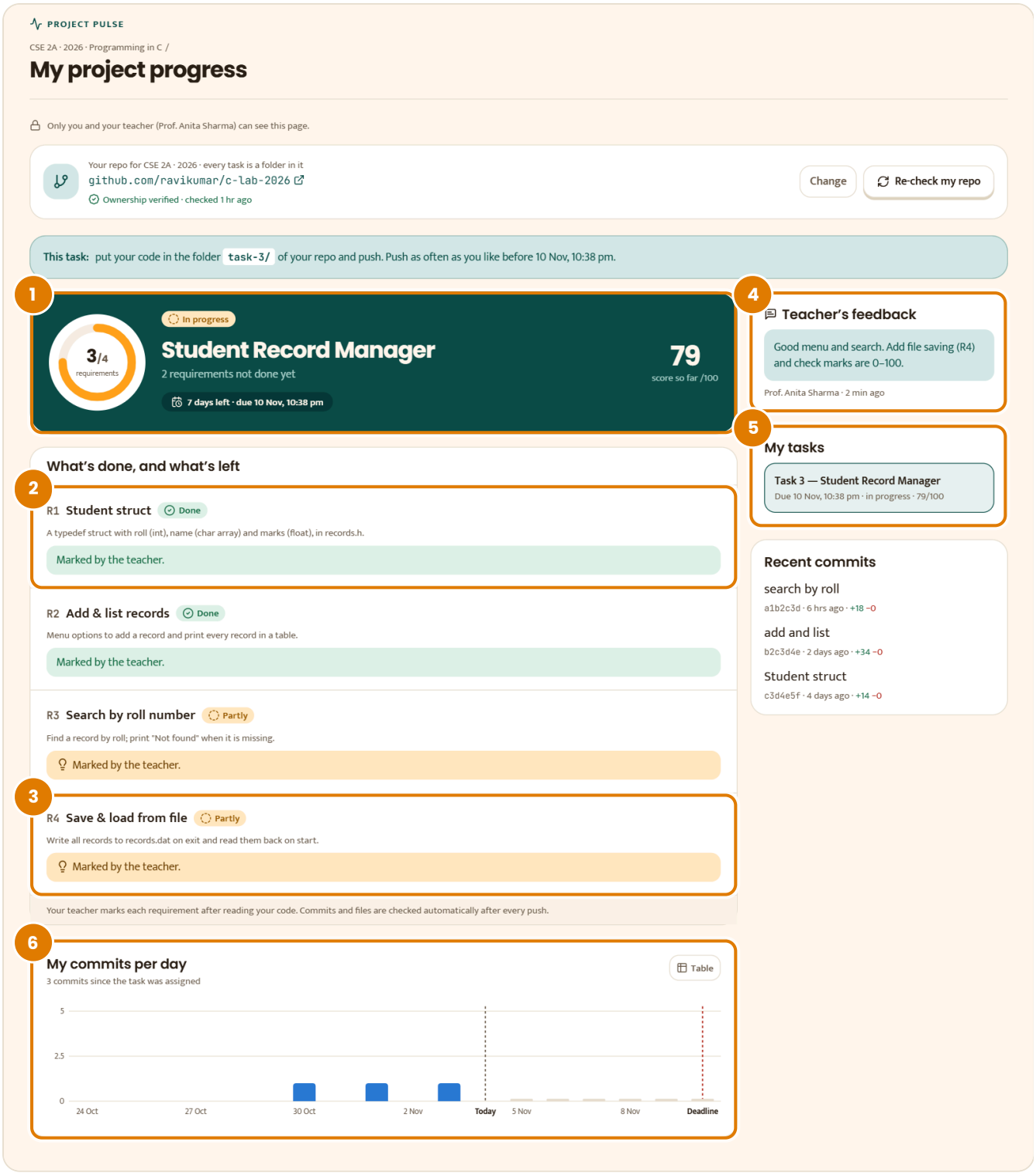
Your page before the repo is verified.

1. Your one repo for this class. **Change** links a different one.
2. One-time setup: add `PULSE.txt` with your personal code to prove the repo is yours. Until then your work doesn't count.
3. **Re-check my repo** after you push (once every 5 minutes).
4. **This task**: the exact folder to use, and the deadline.

5. Pulse tells you when the folder isn't there yet, or is named wrongly (for example `Task3/` instead of `task-3/`).

9. Student: do a task and see your progress

1. A new task appears in **My tasks** with its folder name, for example `task-3/`.
2. Put your code in that folder and push. Push as often as you like before the deadline.
3. Your teacher reviews it. You see their marks and feedback here.



Your progress on a task.

1. Status, score and days left.
2. A requirement your teacher marked **Done**.
3. A requirement not done yet, or **Not reviewed** while your teacher hasn't marked it.
4. **Teacher's feedback**.
5. **My tasks**: all your tasks in this class.
6. Your own commits per day.

PRIVATE

Only you and your teachers see this page. There is no leaderboard and no other student's data.

10. Questions and fixes

Problem	What to do
A teacher can't sign in	Admin: **Set password** next to their name, or **Send login link**.
A student says the invite link doesn't work	They must sign in with the invited email. Teacher: **Resend** on the class page, or share the join code.
A student shows "Not verified"	They need `PULSE.txt` with their code in the repo, then **Re-check**. Or the teacher clicks **Verify myself** after checking the repo with them.
A student shows "Not started" but says they pushed	Check the folder name: it must be exactly the task's folder (for example `task-3/`). Pulse shows the near-miss folder it found.
Two students linked the same repo	When the real owner verifies it, the other student is unlinked and the teacher is told.
Tests say "waiting for the autograder"	Only when Auto-run tests is on: the student must add the autograder file shown on their page, unchanged.
Data looks old	Teacher: **Sync now**. Student: **Re-check my repo**.