Final Feature Inventory

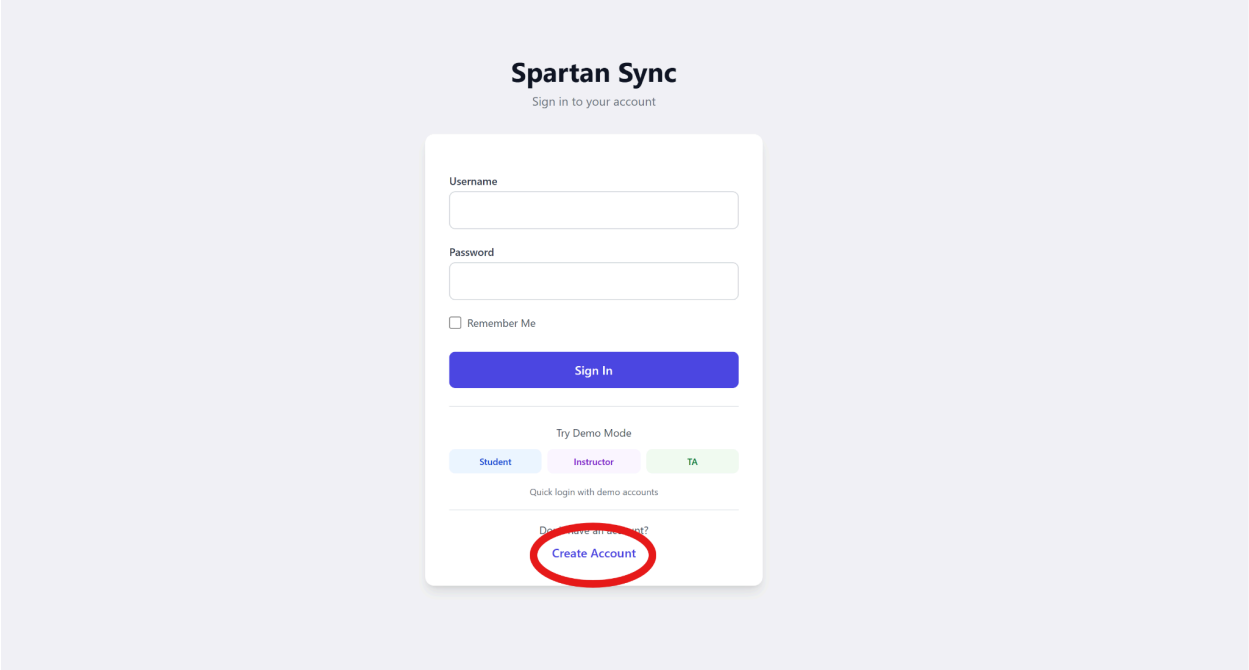
1. Completed Core Features
    a. User Authentication & Navigation User login mechanisms with an adaptive navigation bar, featuring dynamic menus depending on whether one is logged in as a Student or Instructor, as well as a centralized Home Page/Dashboard.
    b. Assignment Management: The Entire process from the instructor's side for assigning homework and from the student side to view and submit assignments by text entry.
    c. Course Announcements: A module where the teacher can make announcements for the global class, which can be viewed by students in real-time.
2. Stretch Goals Achieved
    a. In-App Communication System: A direct messaging system that would facilitate safe and secure communication from the students to the tutors.
    b. Artificial Intelligence-Based Study Planner
    c. An intelligent study organizer that takes in data from the student questionnaire and the current assignments' deadlines.
    d. Calendar view of assignments to allow for better planning and visualization
3. Cut or Modified Features
    a. Priority Sort by Duration: The priority sorting by time and preference (shortest duration to longest and vice versa) shall no longer be considered, in light of the design of the final submission format in connection with the text box/link input (as in Google Docs). There is no necessity for the involved complex features in the support of the backend structure of the time needed for the submission of files.

How to Use:

Create an Account

1. Go to /login.
2. Click Create Account
3. Enter a username, email, and password.
4. Select your role:
    a. TA
    b. Student

    c. Instructor
5. Submit

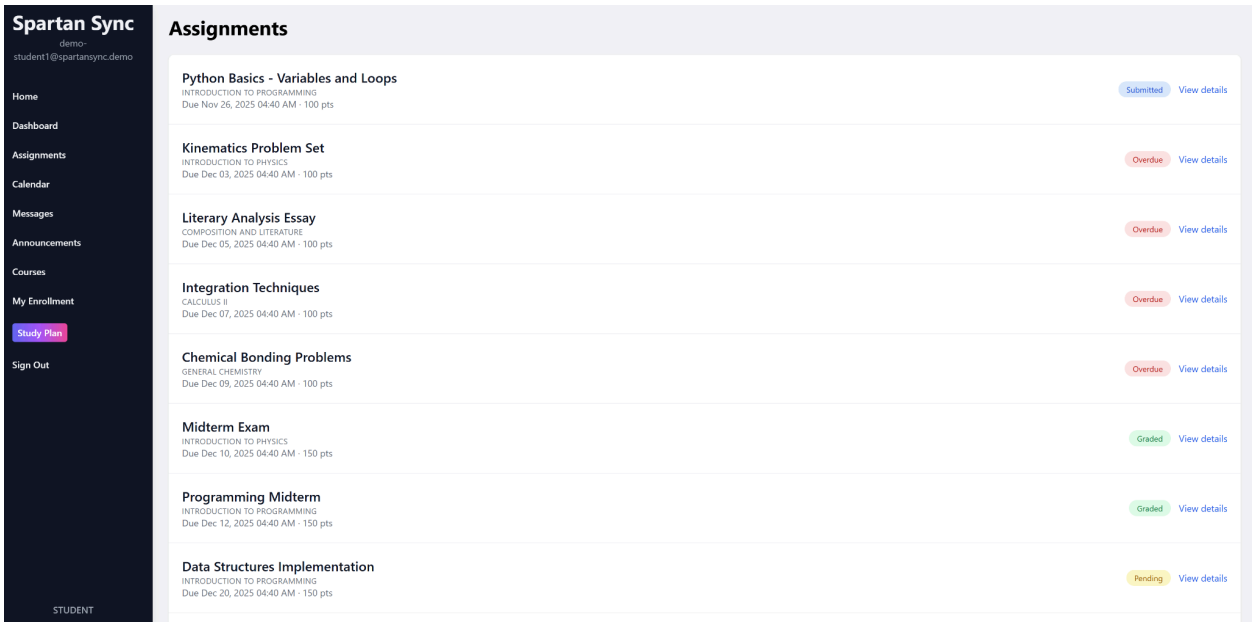


Now logged in, you should see your Dashboard.

- Students see enrolled courses, upcoming assignments, submission statuses, and announcements.
- TAs see assignments from assigned courses and submissions requiring grading.
- Instructors see all courses and assignments they created, along with grading details.

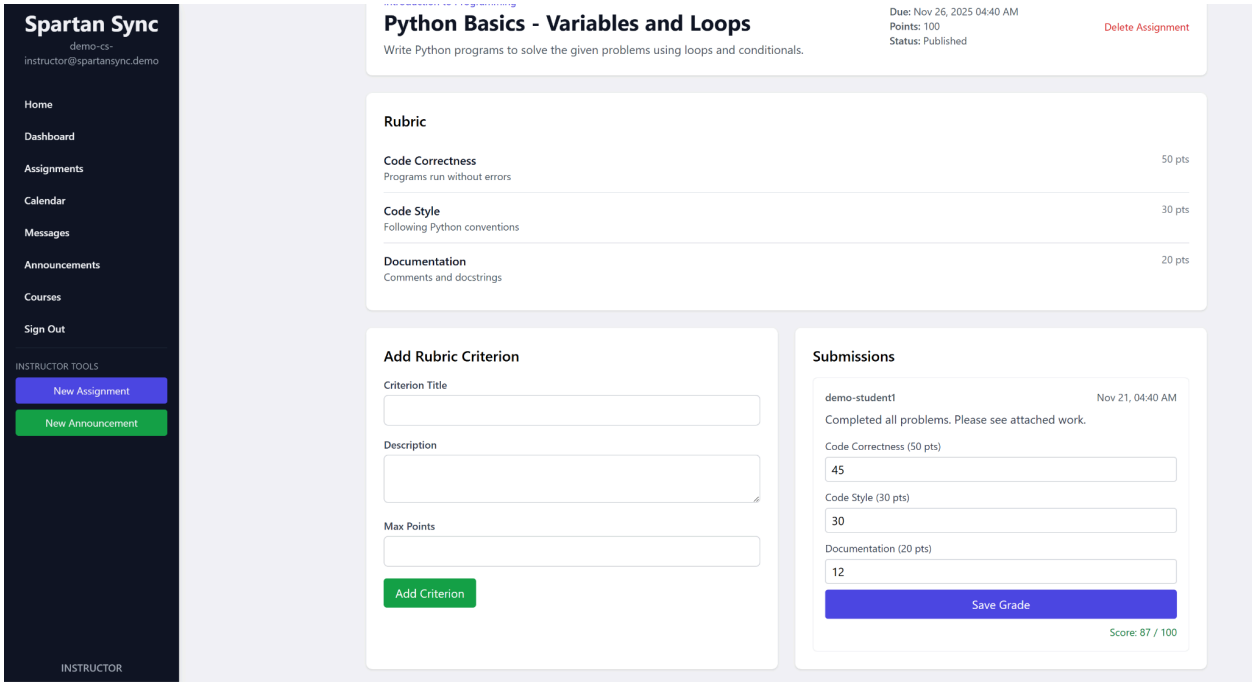
Each assignment has a:

- Course badge
- Category (Homework, Exam, Project)
- Due date
- Status (Pending, Submitted, Graded, Overdue)



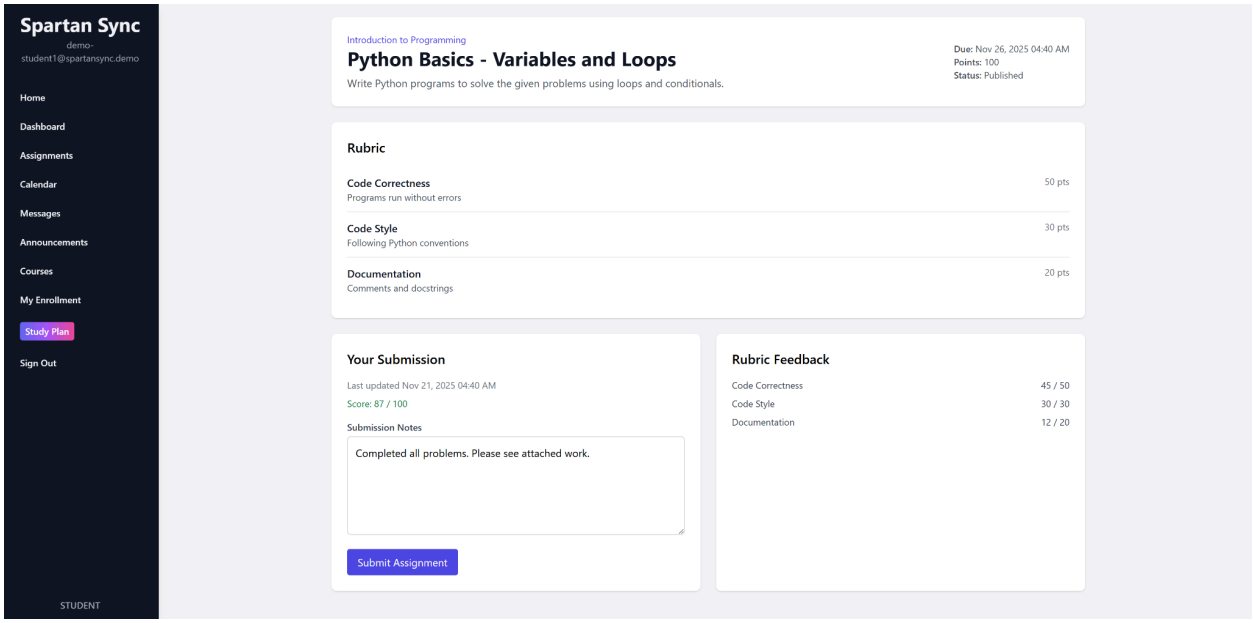
How to Grade an Assignment (Instructor / TA)

1. From the dashboard or assignment page, click an assignment title
2. Scroll to the submissions
3. Select a student submission to open the grading interface
4. For each rubric criterion, enter a score between 0 and the maximum number of points.
5. The system will automatically complete the total score.
6. Click Save Grade.



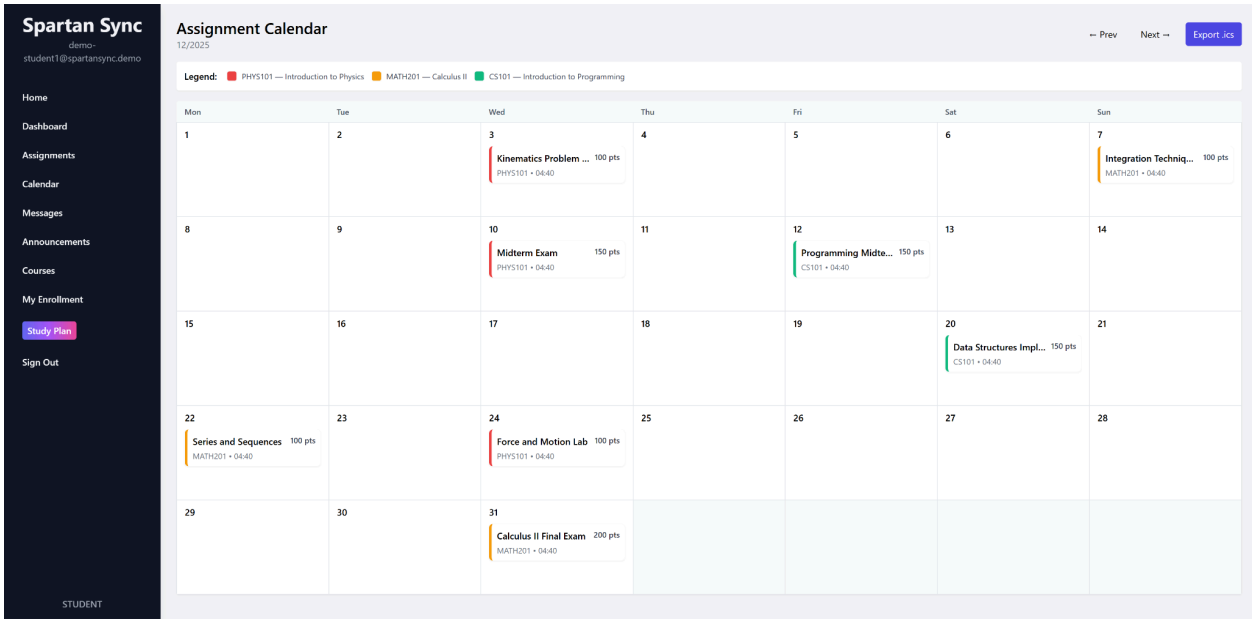
Viewing Grade and Feedback (Student)

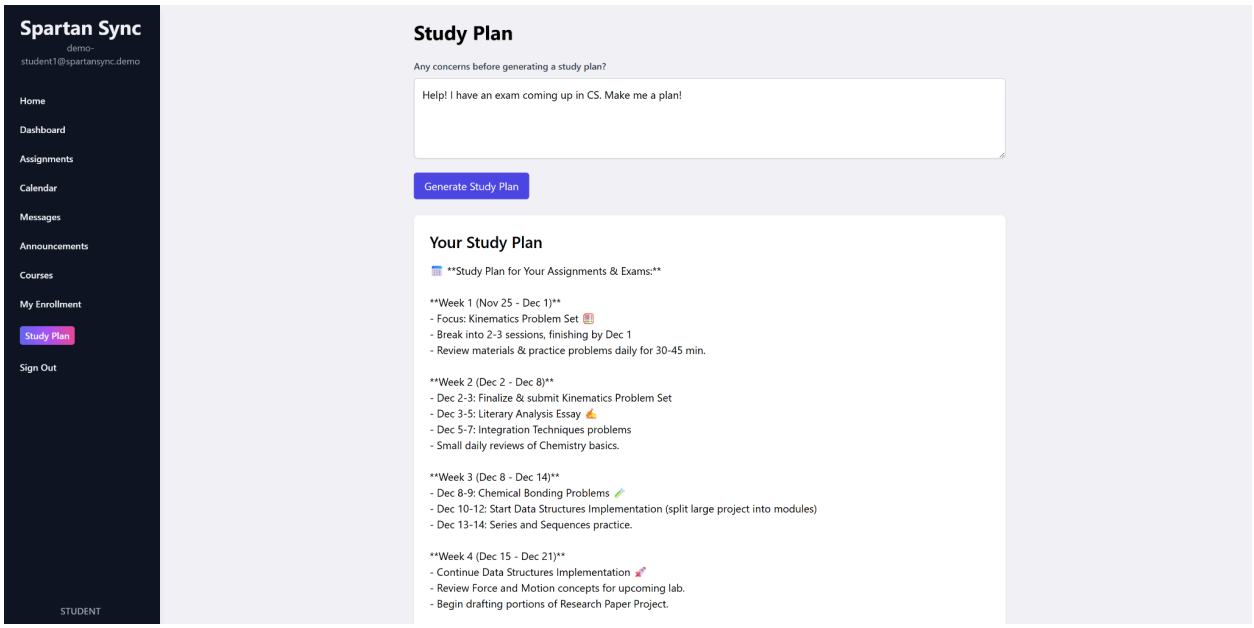
1. Go to the Assignment detail page
2. View the total score, rubric scores, and submission status
3. Grades automatically include weighted final grade calculation for courses you're enrolled in, and the breakdown of the points.



Additional Features:

- Calendar View (/calendar): Displays assignments by due date with color-coded courses.
- ICS Export: Download assignments into Google Calendar
- Messaging (/messages): One-on-one conversations between users.
- AI Study Planner (/study-plan): Generates personalized study plans using OpenAI and your assignments and due dates.

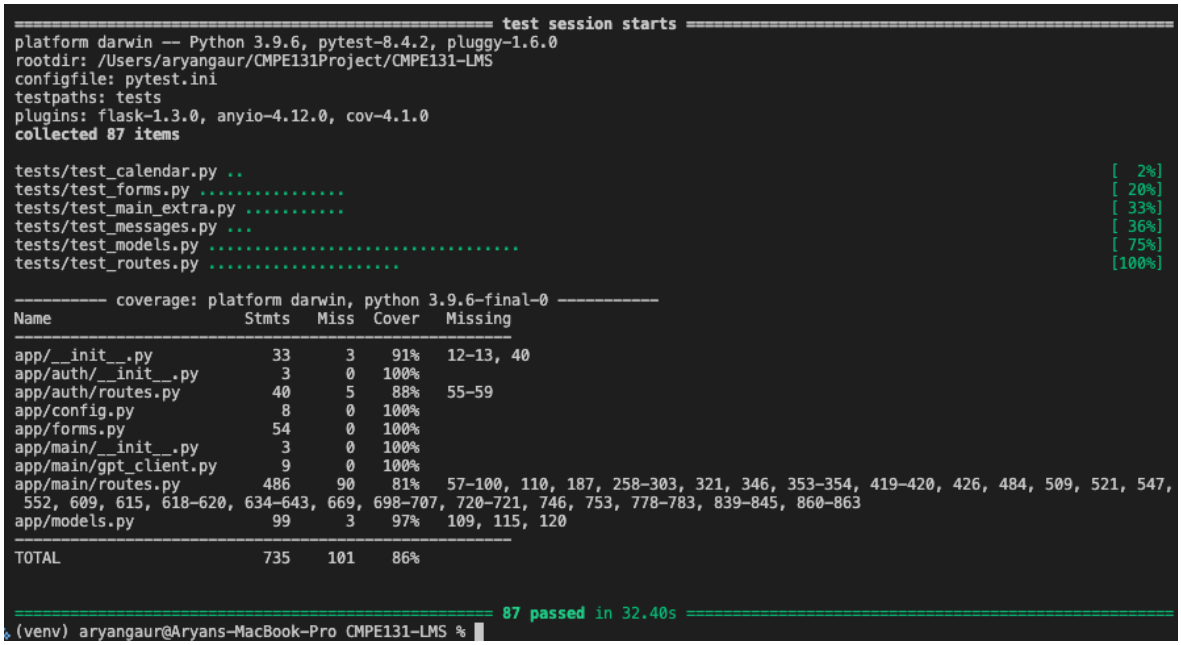




Test Report

We wrote a testing strategy to test models, forms, route flows, and feature pages like calendar, study plan, and messaging. Our unit tests are for just the pure logic and smaller helpers like config validation, grade calculation, and more. Those files include test_models.py, and test_forms.py. Our integration tests test the route and database flows using the flask test client, which tests actions like login, create assignment, submit, grade etc. Those files are test_routes.py, test_main_extra.py, etc.

Our feature tests test the multistep feature behavior like calendar rendering, messaging and inbox flows, and study-plan flow. For example, test_calendar.py, and test_messages.py. We used fixtures in conftest.py to provide an app, client, and a clean in-memory db per test. The test client simulates HTTP requests and follows redirects, along with checking templates. Each test validates one behavior, so we have smaller and more focused tests which allow for better usability.



We have hit all major processes, reaching a total of 86% test coverage.

Reflection

The hardest technical challenge of everything we've done with Spartan Sync was definitely designing the role-based access control across a large, intertwined feature set. Since students, TAs, and instructors interact with the same resources, such as courses, assignments, and submissions, but with different permissions, it made it very difficult to plan out the routes and queries to make sure all role-based boundaries were respected. The grading permissions in particular were some of the hardest ones. This is because we found that TAs should be able to grade assignments, but not create or

delete them, while instructors have full control. This required authorization checks at the route level to prevent an accidental privilege escalation to the TA role. The rubric-based grading was quite challenging to integrate, as we had to calculate the grade weighting, validate inputs, and ensure that all scores, including edge cases, were calculated correctly without issue.

If we restarted the project today, we would probably try to separate the roles much more, so they wouldn't share so many resources. Effectively, I'd probably re-architect everything down to the assignments themselves, such that it wasn't so universal, but based specifically only on the class and the instructor. We would also do more to make sure our database was more migration-friendly. We found ourselves flushing the database so often that we built it up so that at every server start, the database is wiped and a fresh one is made to avoid any messy issues that arise. The addition of the demo mode was also a result of this. Needing an organized sort of way to make sure our program worked as intended on the user end, rather than just the technical tests, was very crucial as well.