

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

The screenshot shows the Spartan Sync login interface. At the top center is the title "Spartan Sync" and a sub-header "Sign in to your account". Below this are two input fields: "Username" and "Password", each with a corresponding text input box. There is also a "Remember Me" checkbox. A large blue button labeled "Sign In" is positioned below the password field. Underneath the sign-in area, there are three tabs: "Student" (light blue), "Instructor" (purple, indicating it is selected), and "TA" (light green). Below the tabs, there is a link "Quick login with demo accounts". At the very bottom of the page, there is a question "Don't have an account?" followed by a blue link "Create Account", which is circled in red.

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)

Spartan Sync
demo-
student1@spartansync.demo

Assignments

Assignment Title	Description	Status	Action
Python Basics - Variables and Loops	INTRODUCTION TO PROGRAMMING Due Nov 26, 2025 04:40 AM - 100 pts	Submitted	View details
Kinematics Problem Set	INTRODUCTION TO PHYSICS Due Dec 03, 2025 04:40 AM - 100 pts	Overdue	View details
Literary Analysis Essay	COMPOSITION AND LITERATURE Due Dec 05, 2025 04:40 AM - 100 pts	Overdue	View details
Integration Techniques	CALCULUS II Due Dec 07, 2025 04:40 AM - 100 pts	Overdue	View details
Chemical Bonding Problems	GENERAL CHEMISTRY Due Dec 09, 2025 04:40 AM - 100 pts	Overdue	View details
Midterm Exam	INTRODUCTION TO PHYSICS Due Dec 10, 2025 04:40 AM - 150 pts	Graded	View details
Programming Midterm	INTRODUCTION TO PROGRAMMING Due Dec 12, 2025 04:40 AM - 150 pts	Graded	View details
Data Structures Implementation	INTRODUCTION TO PROGRAMMING Due Dec 20, 2025 04:40 AM - 150 pts	Pending	View details

STUDENT

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.

Spartan Sync
demo-cs
instructor@spartansync.demo

Python Basics - Variables and Loops
Write Python programs to solve the given problems using loops and conditionals.

Due: Nov 26, 2025 04:40 AM
Points: 100
Status: Published

[Delete Assignment](#)

Rubric

Code Correctness Programs run without errors	50 pts
Code Style Following Python conventions	30 pts
Documentation Comments and docstrings	20 pts

Add Rubric Criterion

Criterion Title:

Description:

Max Points:

Add Criterion

Submissions

demo-student1 Nov 21, 04:40 AM
Completed all problems. Please see attached work.

Code Correctness (50 pts)	45
Code Style (30 pts)	30
Documentation (20 pts)	12

Save Grade

Score: 87 / 100

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.

Spartan Sync
demo-student1@spartansync.demo

Python Basics - Variables and Loops
Write Python programs to solve the given problems using loops and conditionals.

Due: Nov 26, 2025 04:40 AM
Points: 100
Status: Published

Rubric

Code Correctness Programs run without errors	50 pts
Code Style Following Python conventions	30 pts
Documentation Comments and docstrings	20 pts

Your Submission

Last Updated Nov 21, 2025 04:40 AM
Score: 87 / 100

Submission Notes:
Completed all problems. Please see attached work.

Submit Assignment

Rubric Feedback

Code Correctness	45 / 50
Code Style	30 / 30
Documentation	12 / 20

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.

Spartan Sync
demo-
student1@spartansync.demo

Assignment Calendar
12/2025

Legend: PHYS101 — Introduction to Physics MATH201 — Calculus II CS101 — Introduction to Programming

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3 Kinematics Problem ... 100 pts PHYS101 • 04:40	4	5	6	7 Integration Techniq... 100 pts MATH201 • 04:40
8	9	10 Midterm Exam 150 pts PHYS101 • 04:40	11	12 Programming Midte... 150 pts CS101 • 04:40	13	14
15	16	17	18	19	20 Data Structures Impl... 150 pts CS101 • 04:40	21
22 Series and Sequences 100 pts MATH201 • 04:40	23	24 Force and Motion Lab 100 pts PHYS101 • 04:40	25	26	27	28
29	30	31 Calculus II Final Exam 200 pts MATH201 • 04:40				

-- Prev Next -- Export .ics

STUDENT

Spartan Sync
demo-
student1@spartansync.demo

Study Plan

Any concerns before generating a study plan?

Help! I have an exam coming up in CS. Make me a plan!

Generate Study Plan

Your Study Plan

Study Plan for Your Assignments & Exams:

Week 1 (Nov 25 - Dec 1)

- Focus: Kinematics Problem Set
- Break into 2-3 sessions, finishing by Dec 1
- Review materials & practice problems daily for 30-45 min.

Week 2 (Dec 2 - Dec 8)

- Dec 2-3: Finalize & submit Kinematics Problem Set
- Dec 3-5: Literary Analysis Essay
- Dec 5-7: Integration Techniques problems
- Small daily reviews of Chemistry basics.

Week 3 (Dec 8 - Dec 14)

- Dec 8-9: Chemical Bonding Problems
- Dec 10-12: Start Data Structures Implementation (split large project into modules)
- Dec 13-14: Series and Sequences practice.

Week 4 (Dec 15 - Dec 21)

- Continue Data Structures Implementation
- Review Force and Motion concepts for upcoming lab.
- Begin drafting portions of Research Paper Project.

STUDENT

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.

```
===== test session starts =====
platform darwin -- Python 3.9.6, pytest-8.4.2, pluggy-1.6.0
rootdir: /Users/aryangaur/CMPE131Project/CMPE131-LMS
configfile: pytest.ini
testpaths: tests
plugins: flask-1.3.0, anyio-4.12.0, cov-4.1.0
collected 87 items

tests/test_calendar.py ..
tests/test_forms.py .....
tests/test_main_extra.py .....
tests/test_messages.py .....
tests/test_models.py .....
tests/test_routes.py ......

----- coverage: platform darwin, python 3.9.6-final-0 -----
Name           Stmts   Miss  Cover   Missing
app/__init__.py      33      3  91%  12-13, 40
app/auth/__init__.py    3      0  100%
app/auth/routes.py     40      5  88%  55-59
app/config.py         8      0  100%
app/forms.py          54      0  100%
app/main/__init__.py    3      0  100%
app/main/gpt_client.py   9      0  100%
app/main/routes.py    486     90  81%  57-100, 110, 187, 258-303, 321, 346, 353-354, 419-420, 426, 484, 509, 521, 547,
552, 609, 615, 618-620, 634-643, 669, 698-707, 720-721, 746, 753, 778-783, 839-845, 860-863
app/models.py          99      3  97%  109, 115, 120

TOTAL              735    101  86%

===== 87 passed in 32.40s =====
» (venv) aryangaur@Aryans-MacBook-Pro CMPE131-LMS %
```

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.