Requirement ID	Description	Story Points	Priority	Sprint No.
1	User can create a new class profile by providing a course name and instructor name.	2	3	1
2	User can define a grading category (e.g., "Homework") and assign a percentage weight to it.	3	1	1
3	Validation: Ensure that the sum of all defined category weights for a class equals 100%.	5	2	1
4	User can add a new assignment to a category, specifying its name, points earned, and total points possible.	3	4	1
5	Core Calculation: Display the student's current overall weighted percentage grade for the class.	5	5	1
6	Display the percentage and letter grade for each individual grading category.	3	6	1
7	Data Persistence (Local): Implement logic to save the user's current class and grade data to local storage (browser/device).	8	8	1
8	Data Persistence (Local): Implement logic to load saved class and grade data upon application startup.	5	9	1
9	What-If Analysis: User can input a hypothetical score for an ungraded assignment to predict its impact on the final grade.	13	7	2
10	Final Grade Prediction: Calculate the minimum required score on a remaining (or final) assignment to achieve a user-specified target grade (e.g., 93% for an 'A').	13	10	2
11	Develop the primary UI/UX design (wireframe) for the input and display screens.	5	11	2
12	Implement the application's modern, responsive graphical user interface (GUI) using a chosen framework (e.g., Python/Tkinter, web framework).	8	12	2
13	Security: Implement user authentication (sign-up/log-in) using a cloud service (e.g., Firebase).	13	14	2
14	Cloud Storage: Store the user's class and grade data in a cloud database, replacing local storage.	8	15	2
15	Implement password hashing (e.g., bcrypt) for all stored user credentials.	5	13	2
16	Custom Grade Scale: Allow the user to define and save their own letter grade cutoffs (e.g., A = 93-100, A- = 90-92).	5	16	2
17	Error Handling: Gracefully handle the division-by-zero scenario when a category has 0 total points possible.	2	17	2
18	Validation: Implement input checks to prevent non-numeric or negative values in all score fields.	2	18	2
19	Implement basic unit tests for the core grade calculation logic.	3	19	2
20	Security: Ensure all data transmission (if cloud used) is encrypted (HTTPS/SSL).	2	20	3
21	Provide a summary report showing total points earned vs. total points possible across all assignments.	2	21	3
22	Refinement: Optimize grade calculation algorithms for handling a large number of assignments (>100).	3	22	3
23	Accessibility: Ensure the final GUI design meets WCAG minimum accessibility standards (e.g., keyboard navigation).	5	23	3
24	Develop a short video presentation demonstrating the final application's features and architecture.	3	24	3