Grade Calculator Project Requirements Document

Version: 1.0 **Date:** October 19, 2025

1.0 Introduction

1.1 Purpose

This document outlines the functional and non-functional requirements for the development of the Agile Grade Calculator Application. This single-user application will enable students to accurately track their academic performance, manage weighted grading structures, and predict final scores needed for desired outcomes. This SRS serves as the authoritative guide for all development, testing, and acceptance activities for Project 3.

1.2 Scope

The application will manage one or more user-defined course profiles. For each course, the user can input assignments, organize them into weighted categories, calculate the current grade, and perform advanced projections ("What-If" analysis). The final system must incorporate a modern GUI and include provisions for user authentication and persistent cloud-based data storage.

2.0 Functional Requirements

2.1 Class and Category Management (Setup and Weighting)

01 (Class Profile Creation): The system shall allow the user to create a new class profile by providing a course name (e.g., "EECS 581") and associated details. 02 (Category Definition): The system shall allow the user to define grading categories (e.g., "Homework," "Exams") and assign a percentage weight to each category. 03 (Total Weight Validation): The system shall enforce that the sum of all category weights within a single class profile is exactly 100%.

2.2 Assignment Input and Score Management

01 (Assignment Input): The user shall be able to add a new assignment to any defined category, specifying its name, points earned, and total points possible. 02 (Score Validation): The system shall implement input checks to prevent the entry of non-numeric or negative values in all score and weight fields. 03 (Assignment Editing): The user shall be able to edit or delete existing assignment entries within a category.

2.3 Calculation and Prediction

01 (Current Weighted Grade): The system shall calculate and display the student's current overall weighted percentage grade for the class based on completed assignments. 02 (Category Grade Display): The system shall display the percentage and letter grade for each individual grading category. 03 ("What-If" Analysis): The system shall allow the user to input a hypothetical score for an ungraded assignment to instantly see the predicted final course grade. 04 (Score Needed Calculation): The system shall calculate the minimum required score on a remaining assignment (e.g., Final Exam) to achieve a user-defined target grade. 05 (Error Handling): The system shall gracefully handle the division-by-zero scenario that occurs if a category has 0 total points possible.

2.4 User and Data Management

01 (User Authentication): The system shall implement secure user sign-up, log-in, and session management using a cloud-based authentication service. 02 (Cloud Data Persistence): The user's class and grade data shall be securely saved to and retrieved from a cloud database (replacing initial local storage). 03 (Custom Grade Scale): The system shall allow the user to define and save their own percentage-to-letter grade cutoffs (e.g., A = 93-100).

3.0 System Architecture

This section describes the high-level structure of the system as a guide for development and future extensions.

3.1 Components

- **User Manager:** Handles authentication, user sessions, and manages the collection of ClassProfile objects associated with the current user.
- **Data Service:** Abstracts the cloud database interaction (read/write/update) for data persistence.
- **Grade Logic Engine:** Contains the core business logic within the ClassProfile, Category, and Assignment classes, managing all calculation, validation, and prediction functions.
- **User Interface (GUI):** Renders the input forms and displays results, communicating with the Grade Logic Engine.

3.2 Key Data Structures

The core data will be structured hierarchically using Python classes:

- User (Root object for a session)
- ClassProfile (Manages a list of Category objects)
- Category (Manages a list of Assignment objects and calculates category score)

- Assignment (The unit of score data: points earned/possible)

3.3 Data Flow

1. **Authentication:** User logs in via the GUI, and the User Manager validates credentials via the cloud service.
2. **Data Loading:** Upon successful login, the Data Service loads the user's ClassProfile objects, hydrating the Grade Logic Engine.
3. **Input:** User interacts with the GUI to enter scores or weights.
4. **Processing:** Input Handler sends data to the Grade Logic Engine (Assignment, Category methods), which validates the input and updates the internal state.
5. **Output:** The Grade Logic Engine calculates the current grade, which the GUI instantly retrieves and displays.

3.4 Non-Functional Assumptions

- **Development Language:** The business logic will be implemented primarily in Python.
- **Technology Stack:** The application will use a modern framework for the GUI and a cloud platform for authentication and data storage.
- **Security:** The system assumes the security mechanisms of the chosen cloud authentication and database providers (e.g., Firebase Auth/Firestore) are sufficient.
- **Weighting:** All grade calculation assumes a weighted average system where category weights must sum to 100%.