Agile Grade Calculator Initial Architecture

Version: 1.0 **Date:** October 26, 2025
Group 18: Carson Abbott, Owen Deines, Kaleb Howard, Jakob Huffman

1.0 Project Synopsis

An application that allows students to accurately track their academic performance, manage weighted grading structures, and predict final scores needed for desired outcomes.

2.0 Architecture

This section outlines the initial architecture of the Agile Grade Calculator system.

2.1 Components

The system consists of four primary components: User Manager, Data Service, Grade Logic Engine, and User Interface (GUI). Together, these components coordinate to deliver a seamless experience that allows students to manage classes, track grades, and predict academic outcomes.

- The User Manager is responsible for authentication, session management, and association of user-specific data. It handles all interactions related to user identity, including login, logout, and current user sessions. Each authenticated user is linked to a collection of ClassProfile objects representing their active or past courses. The User Manager acts as the gateway between user identity and application data, ensuring that users can securely access only their own records.
- The Data Service provides an abstraction layer for persistent data storage and retrieval. It isolates the application logic from the underlying cloud database by exposing a clean interface for reading, writing, and updating objects such as User, ClassProfile, Category, and Assignment. This design choice enables easy migration between different cloud platforms (e.g., Firebase, AWS, or Azure) without requiring changes to the core logic. The Data Service also handles synchronization, ensuring that updates from the Grade Logic Engine are consistently reflected in the database and vice versa.
- The Grade Logic Engine contains the core logic of the application. It operates within the ClassProfile, Category, and Assignment classes, managing calculations such as category averages, weighted totals, and required future scores. It also performs input validation, ensuring that entered scores, weights, and totals are within acceptable bounds.
- The graphical interface provides the visual and interactive layer through which the user interacts with the system. It renders forms for score entry, displays calculated grades, and presents predictive analytics in a clear and responsive layout.

2.2 Key Data Structures

The core data model is implemented through a hierarchical set of Python classes designed to mirror the logical structure of academic grading systems:

- User - The root object for each authenticated session. It maintains references to all ClassProfile instances associated with the logged-in account.
- ClassProfile - Represents a single course or subject. It manages a list of Category objects (e.g., Homework, Exams, Projects) and aggregates their weighted contributions to determine the overall grade.
- Category - Defines a grading category containing multiple Assignment objects. It maintains its weight percentage and computes the category average by summing weighted assignment scores.
- Assignment - It stores the number of points earned, the number of points possible, and any related metadata such as due date or completion status.

This hierarchical model enables flexible grade calculation and future extensibility. For example, adding new structures like “Extra Credit” or “Attendance” without major redesign is easier to achieve.

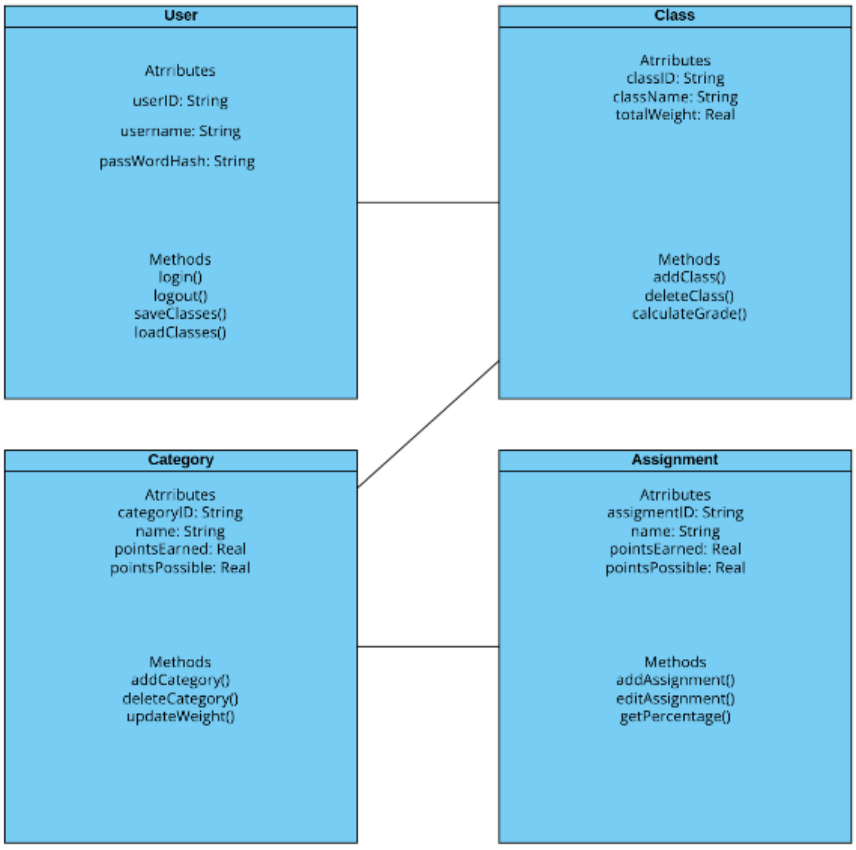

Figure 1: *Structure UML Diagram*

2.3 Data Flow

1. Authentication: The user logs in through the GUI. The User Manager validates credentials via the cloud authentication service.
2. Data Loading: After successful login, the Data Service retrieves the user's stored data and initializes the Grade Logic Engine with corresponding ClassProfile and Category objects.
3. Input: The user interacts with the GUI to modify grades, add assignments, or adjust category weights.
4. Processing: The Grade Logic Engine validates input and performs necessary recalculations, updating category and class averages.
5. Output: Updated results are returned to the GUI for real-time display, allowing the user to view their current grade and predicted outcomes.

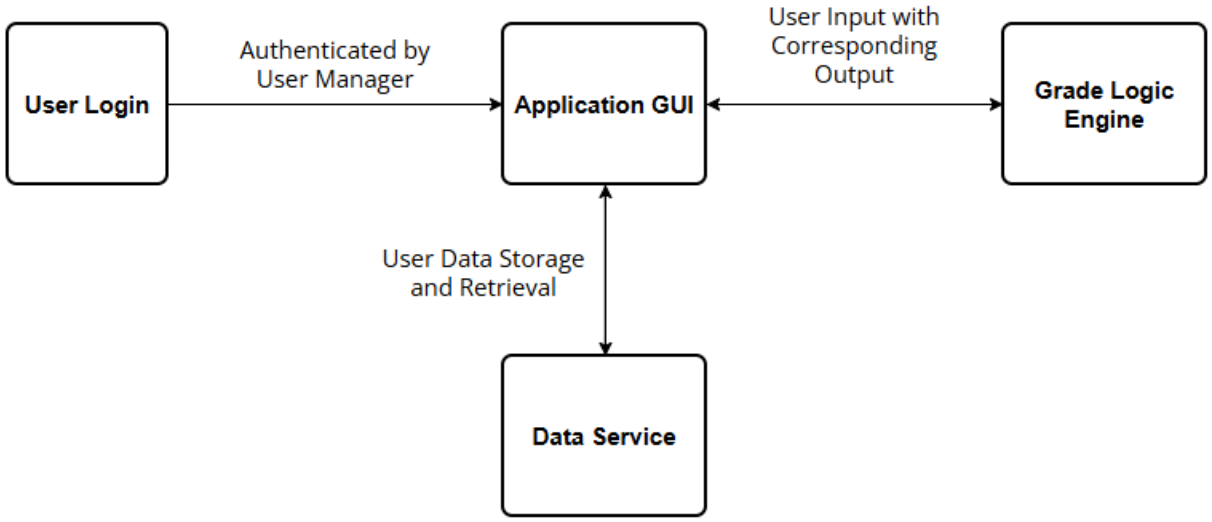

Figure 2: *Data Flow Diagram*

2.4 Non-Functional Assumptions

- Development Language: Core logic is implemented in Python.
- Technology Stack: A modern web or desktop framework will be used for the GUI, and a cloud platform (such as Firebase) will handle authentication and storage.
- Security: The system leverages built-in security of the cloud provider.
- Weighting Policy: Grade computations assume weighted averages where total category weights equal 100%.