Homework Tracker

- Sean Anderson
- Mauricio Aquino
- Joseph Mazzocco
- Eric Quan
- Bryce Young

Summary of Software Prototype

- Goal
  - Help students manage homework across the courses that they are taking
  - Reduce the time and stress that comes from homework management
  - Simplify the process of tracking homework.
- Functionality
  - Presents class and assignment data in a clear and visual way
  - Keeps track of when assignments are due
  - Notifies the user, via the Windows 10 notification tray, when an assignment's deadline is within the current week.

Demo Time!

Application Test Plan

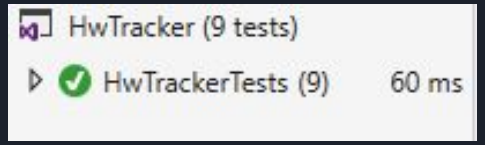
```
/// <summary>
/// Tests if assignments are properly constructed
/// </summary>
[TestMethod]
public void TestAssignmentConstructor()
{
    string name = "Test";
    int points = 100;
    DateTime testDate = DateTime.Now;

    Assignment testAssignment = new Assignment(name, points, testDate);

    Assert.AreEqual(name, testAssignment.Name);
    Assert.AreEqual(points, testAssignment.Points);
    Assert.AreEqual(testDate, testAssignment.DueDate);
}
```

- 3 Unit Tests
  - Assignment Constructor
    - Make sure Name, Points, and DueDate are properly initialized
  - GradeWeightCategory TotalPoints
    - Make sure the the points in a category is the sum of its Assignments points
  - Notification Constructor
    - Make sure Title and Message are properly initialized
- 2 Integration Tests
  - Integrate Course, NotificationGenerator, NotificationQueue
    - Makes sure that the NotificationGenerator properly creates and adds Notifications to the NotificationQueue given a list of Courses
  - Integrate GradeWeightCategory, Assignment
    - Makes sure that the GradePercentage of an Assignment is recalculated properly when new Assignments are added and the weight of the category changes

Results of Testing



- Unit Tests
  - Assignment Constructor: Success!
    - Name, Points, and DueDate were properly initialized in the test Assignment
  - GradeWeightCategory TotalPoints: Success!
    - A test Category with Assignments worth 10, 20, and 30 points had TotalPoints == 60
  - Notification Constructor: Success!
    - Title and Message were properly initialized
- 2 Integration Tests:
  - Integrate Course, NotificationGenerator, NotificationQueue: Success!
    - When given a list of Courses that had two Assignments due within a week, the NotificationGenerator added two Notifications to the NotificationQueue
  - Integrate GradeWeightCategory, Assignment: Success!
    - When the weight of the category changed and new Assignments were added, the grade percentage in the test Assignment properly updated.

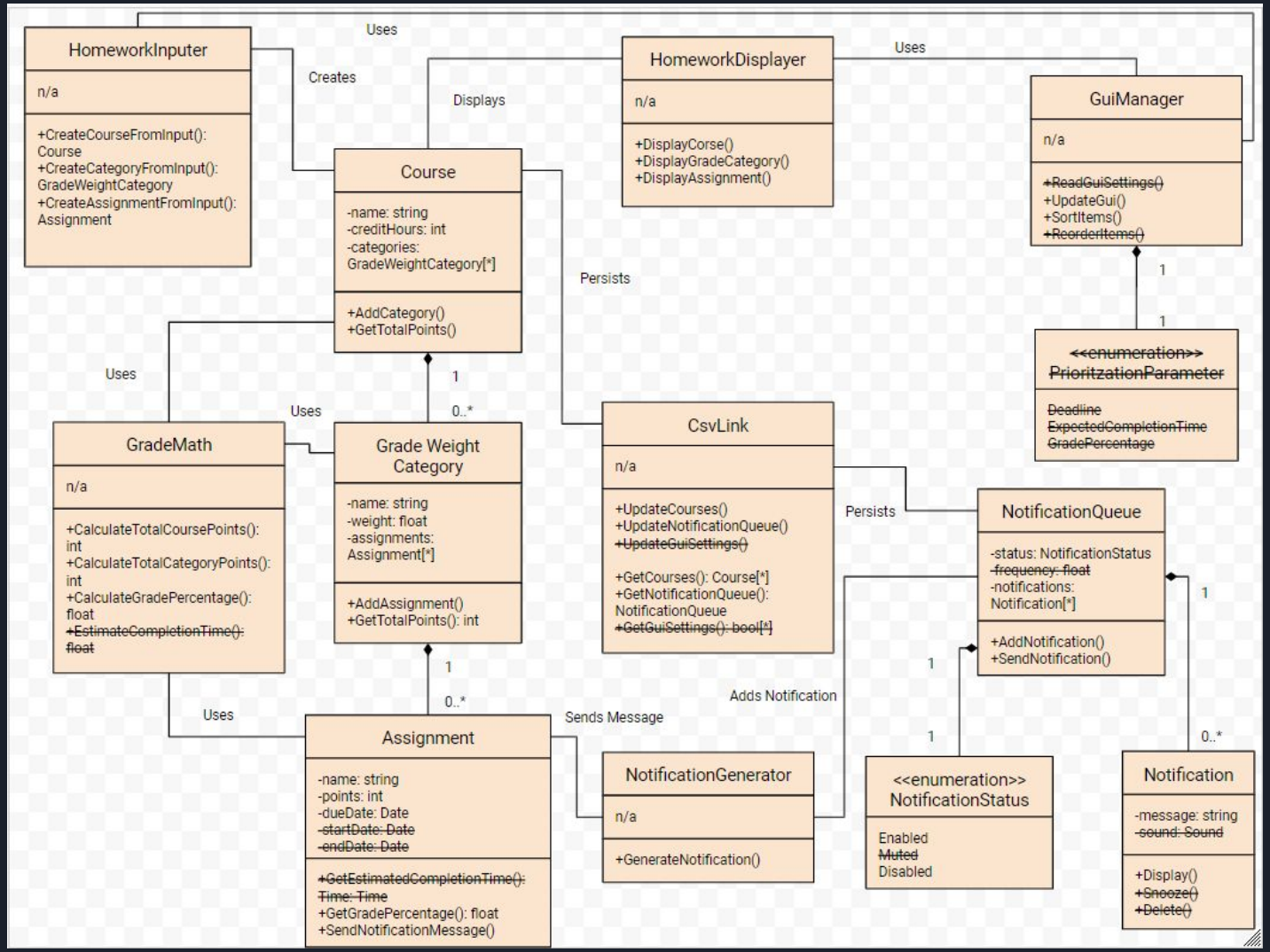
Updates to the SRS

- N/A

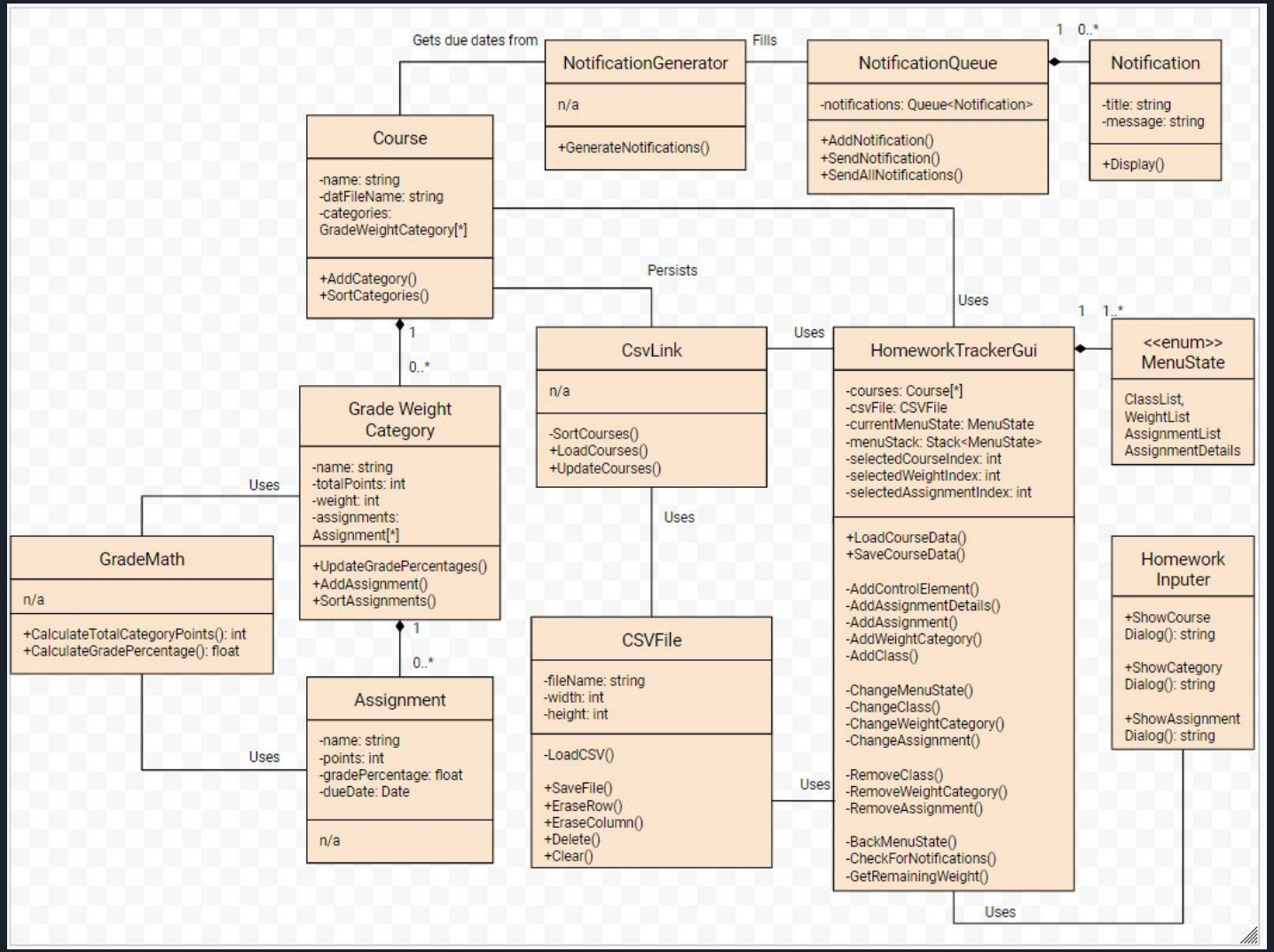
Updates to the SDD

- Class Diagram Changes
  - Because of Windows Forms, we had to merge display classes
  - Still had members that pertained to features that we were not planning to implement in the prototype
  - Edited backend classes to better support our functionality
  - Added classes that we ended up needing
  - Changed some associations between classes
  - Changed which data we are persiting

Original Class Diagram



New Class Diagram



Lesson 1: The Role of Requirements and Scope

- Requirements gathering
  - Makes sure the customer knows and is getting what they want
  - Ensures the customer is satisfied
  - Defines the scope of the project
- Scope
  - Allows developers to focus their effort were it matters
  - Lets developers know what they need to do, and refuse what they do not need to do

Lesson 2: The Importance of Good Software Design

- It is important to think and plan before coding
- Proper design encourages flexibility and code reuse
- Helps ensure requirements are fulfilled
- A good design often leads to good software

Lesson 3: The Merits and Downfalls of the Waterfall Methodology

- Merits
  - The solution is well defined before implementation takes place
  - Planning is a big part of the beginning of the project
  - Separation of work into stages helps break the project into more manageable pieces
- Downfalls
  - Not easy to make changes
  - Once the design phase is over, a bad design will lead to a bad implementation
  - Can make the implementation of the software rushed

Lesson 4: Requirements can change

- Customer/clients needs may change overtime
- Requirements can can change throughout the software development process
  - Flaws in requirements
  - Need for additional requirements
  - Unrealistic expectations

Lesson 5: Don’t take on more than you can handle

- Be realistic about the requirements you plan to fulfill and the time it takes to implement them
- Do not reinvent the wheel
  - Designing login/encryption
- Give yourself enough time during implementation
- Do well in requirements and design phase so you discover issues early and implementation goes smoothly

Lesson 6: Don’t Just jump into Implementation

- Do not jump straight into the solution
  - Reason for this class and for Software Engineering
- Allows us to find things we need to add, modify, or remove
- Helps us to understand how components work together and individually
- Helps us to find flaws, issues, and risks with our software
- Program more effectively