Project Name: Homework Tracker // Group 6

1. Overview

Homework Tracker is a software solution that helps students manage homework across the courses that they are taking. It aims to reduce the time and stress that comes from homework management and to simplify the process of prioritizing homework. Specifically, homework tracker keeps track of when assignments are due and notifies the user, via the Windows 10 notification tray, when an assignment's deadline is within 24 hours. In addition to notifications, homework tracker estimates how long it will take the user to complete an assignment based on previous completion times and helps the user figure out which assignment to focus on by allowing the user to sort assignments based on various prioritization parameters.

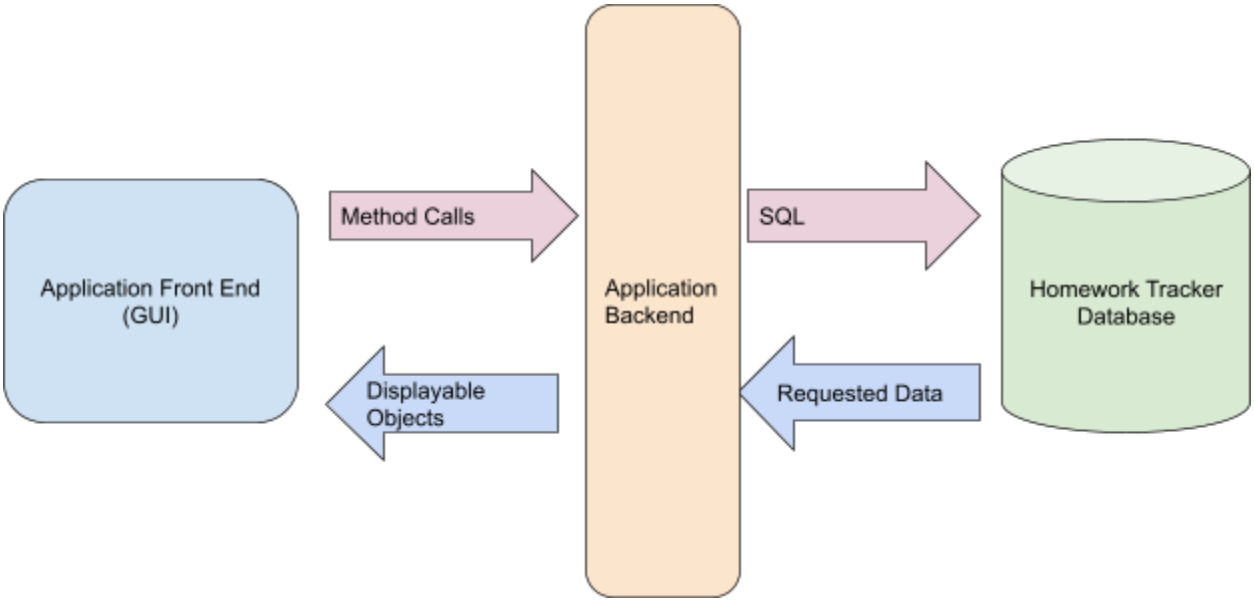
2. Stakeholders

- The Development Team:
  - Role:
    - The development team is responsible for building the project and making sure that it satisfies all requirements.
  - Concerns:
    - This application will be very UI/UX_1 intensive, and the team has very little experience in designing UI/UX_1.
    - The time given to build the application is very limited, which may lead to an underwhelming application.
- The Customer:
  - Role:
    - The customer is the reason that this solution is being developed and will be responsible for evaluating the work produced by the development team.
  - Concerns:
    - At the end of the development process, the customer will expect a high quality application that helps users with homework management.
    - The customer will be wary of the risks highlighted in our SRS. For example, the customer will be very concerned with whether or not Homework Tracker is easy to use.
    - Customer would be concerned about our team finishing the requirements outlined in our SRS.
- The Users:
  - Role:
    - The users are the people who Homework Tracker is being created for and will be the primary people who interact Homework Tracker.

- Concerns:
  - The users will expect Homework Tracker to help them effectively manage their homework.
  - The users will want an easy to use UI/UX$_1$.
  - The users will want a reliable application so that do not lose their data.
  - The users will want a secure application so that their data is safe.

3. System Architecture

- Diagram:



- Description:
  - Our application will use a three-tiered architecture. We use a GUI$_2$ to interact with the user. When the user makes choices that require database information, like viewing assignments and changing settings, our GUI$_2$ will request that information from our application backend. Our application backend will then communicate with the database using SQL$_3$, and the database will return the requested information. After this, the application backend wraps the requested data in an object and sends it to the GUI$_2$ to be displayed.

4. Detailed Design

4.1.1 Logical Viewpoint: Classes and their Relationships

- **Course**
  - Description:
    - Models a class the user is taking. A Course has a name, total point amount, a credit hour amount, and a list of GradeWeightCategories. A Course contains data pertaining to a class the user is taking.
  - Relationship to other classes:
    - A Course is composed of many GradeWeightCategories
    - Course uses GradeMath
    - Courses are persisted by DatabaseLink
    - Courses are created by HomeworkInputer
    - Courses are displayed by HomeworkDisplayer

- **GradeWeightCategory**
  - Description:
    - Models a grade weight category in a course. A GradeWeightCategory has a name, a list of Assignments, and a weight, which is a floating point number that multiplies the points of the Assignments in this GradeWeightCategory. A GradeWeightCategory contains assignments that are worth a certain percentage of the overall Course grade.
  - Relationship to other classes:
    - GradeWeightCategories are owned by a Course
    - A GradeWeightCategory is composed of many Assignments
    - GradeWeightCategory uses GradeMath
    - GradeWeightCategories are persisted by DatabaseLink
    - GradeWeightCategories are created by HomeworkInputer
    - GradeWeightCategories are displayed by HomeworkDisplayer

- **Assignment**
    - Description:
        - Models an assignment in a grade weight category. An assignment has a name, a due date, a point amount, a start time, and an end time. An Assignment contains data that pertains to homework that the user needs to complete.
    - Relationship to other classes:
        - Assignments are owned by a GradeWeightCategory
        - Assignment uses GradeMath
        - Assignments are persisted by DatabaseLink
        - Assignments are created by HomeworkInputer
        - Assignments are displayed by HomeworkDisplayer

- **HomeworkInputer**
    - Description:
        - A class that gets homework data from the user. A HomeworkInputer is responsible for getting and validating user input via the GUI_2 and creating the appropriate object from the input.
    - Relationship to other classes:
        - HomeworkInputer creates Courses
        - HomeworkInputer creates GradeWeightCategories
        - HomeworkInputer creates Assignments
        - HomeworkInputer is used by GUIManager

- **Notification**
    - Description:
        - A class models the notifications that will be sent to the user, via the Windows 10 notification tray. A Notification has a message which is a string, and a sound to be played. A notification can be displayed, snoozed, or deleted.
    - Relationship to other classes:
        - Notifications are created by NotificationGenerator
        - Notifications are owned by NotificationQueue
        - Notifications are persisted by DatabaseLink

- **NotificationGenerator**
    - Description:
        - A class that creates notifications. A NotificationGenerator creates a new Notification when an Assignment indicates that it's deadline is close.
    - Relationship to other classes:
        - NotificationGenerator receives messages from Assignments
        - NotificationGenerator creates Notifications
        - NotificationGenerator adds Notifications to the NotificationQueue

- **NotificationStatus**
    - Description:
        - An enumeration that tracks the user's current notification preference. Notification has three possible values: Enabled, Muted, and Disabled.
    - Relationship to other classes:
        - NotificationStatus is owned by NotificationQueue
        - NotificationStatus is persisted by DatabaseLink

- **NotificationQueue**
    - Description:
        - A class that holds notifications and sends them to the user. A NotificationQueue has a NotificationStatus, a list of Notifications, and a notification frequency which is a floating point number that controls how often notifications are sent.
    - Relationship to other classes:
        - A NotificationQueue is composed of many Notifications
        - NotificationQueue owns NotficationStatus
        - NotificationQueue is persisted by DatabaseLink

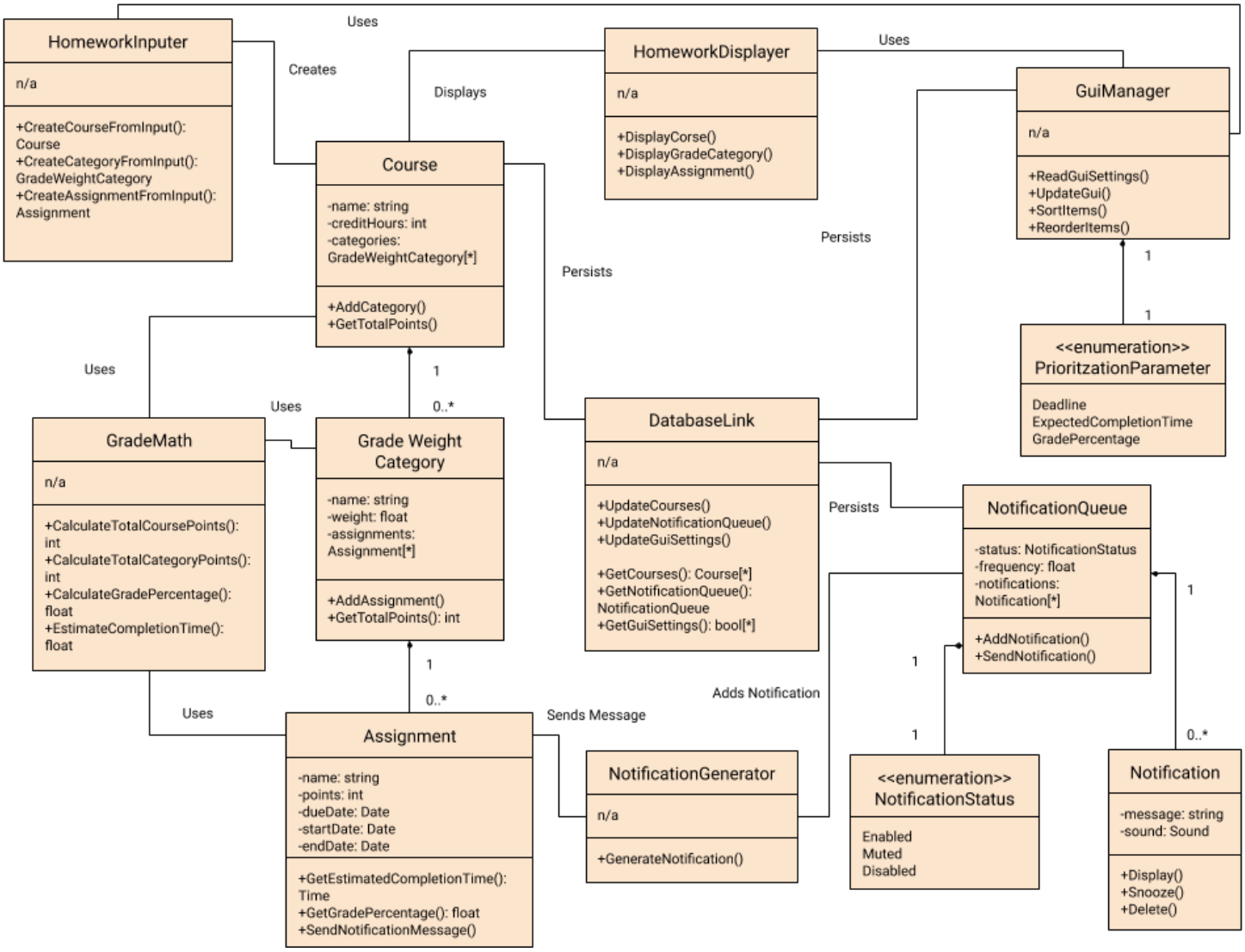
- **PrioritizationParameter**
    - Description:
        - An enum that tracks the user's current prioritization preference. The notification status will be an enum with three possible values: Deadline, ExpectedCompletionTime, and GradePercentage.
    - Relationship to other classes:
        - PrioritizationParameter is owned by GUIManager
        - PrioritizationParameter is persisted by DatabaseLink

- **HomeworkDisplayer**
    - Description:
        - A class that displays homework related objects to the GUI_2. A HomeworkDisplayer is displays Courses, GradeWeightCategories, and Assignments to the GUI_2.
    - Relationship to other classes:
        - HomeworkDisplayer displays Courses
        - HomeworkDisplayer displays GradeWeightCategories
        - HomeworkDisplayer displays Assignments
        - HomeworkDisplayer is used by GUIManager

- **GUIManager**
    - Description:
        - A class that controls the GUI_2. GUIManager controls the appearance of the GUI_2 and displays all objects that need to be displayed in the GUI_2.
    - Relationship to other classes:
        - GUIManager uses HomeworkDisplayer
        - GUIManager uses HomeworkInputer
        - GUIManager is persisted by DatabaseLink

- **DatabaseLink**
    - Description:
        - A class that persits the application's data in the database. A DatabaseLink is responsible for updating the database and reading from the database.
    - Relationship to other classes:
        - DatabaseLink persists Course
        - DatabaseLink persists GradeWeightCategory
        - DatabaseLink persists Assignment
        - DatabaseLink persists Notification
        - DatabaseLink persists NotificationStatus
        - DatabaseLink persists NotificationQueue
        - DatabaseLink persists PrioritizationParameter
        - DatabaseLink persists GUIManager
- **GradeMath**
    - Description:
        - A collection of static methods for doing calculations pertaining to grades. GradeMath will be responsible for calculating: weighted grades, grade percentages, total class points, and estimated completion times.
    - Relationship to other classes:
        - Course uses GradeMath
        - GradeWeightCategory uses GradeMath
        - Assignment uses GradeMath

4.1.2 Logical Viewpoint: Class Diagram

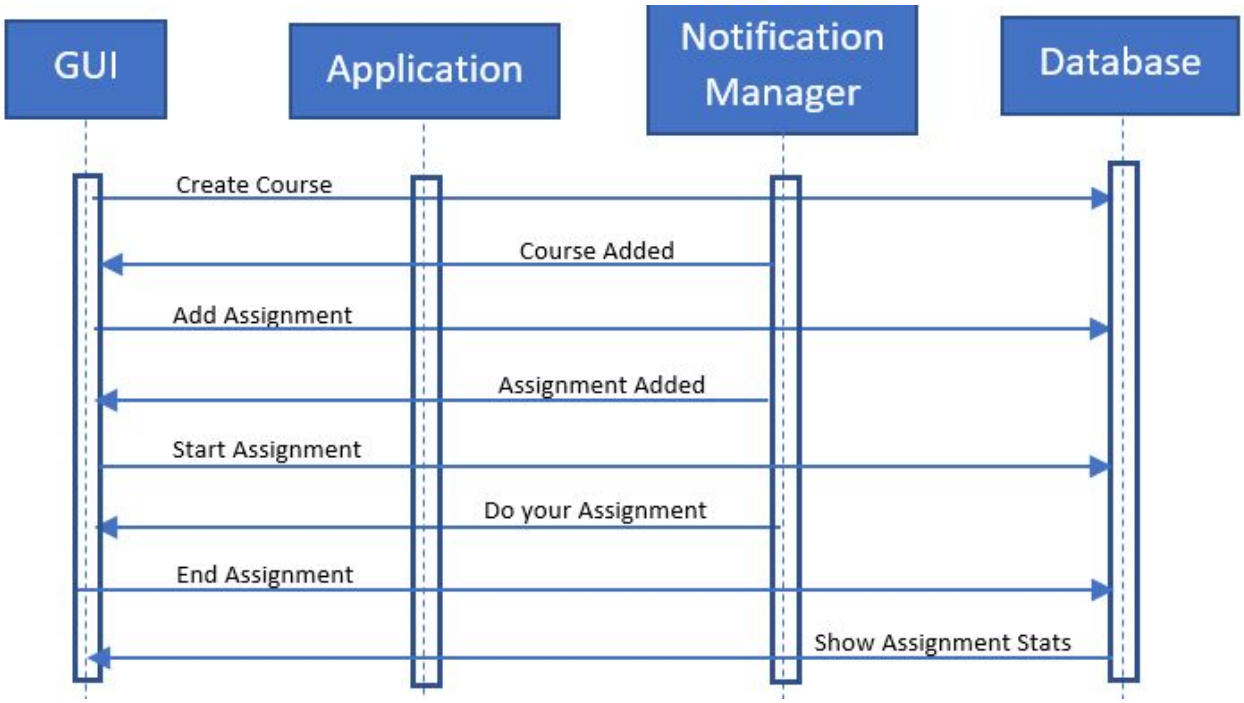


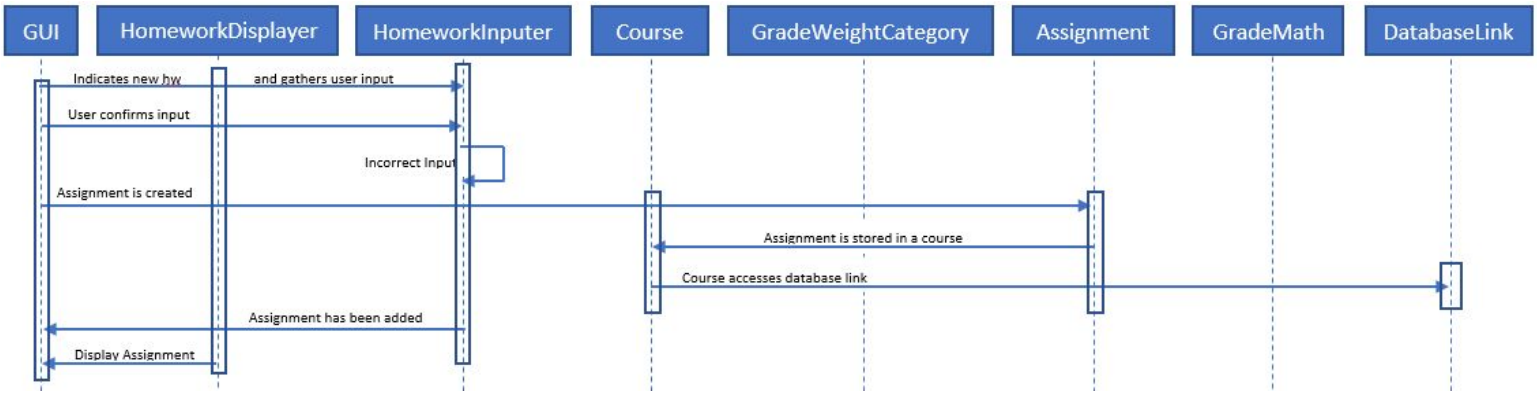
4.2.1 Interaction Viewpoint: Main Functionalities

- Login Interaction:
    - Description: When users open the application, they will be automatically logged into their account. There is no need for a username and password because there is only one account and data does not need to be private.
    - Relationships:
        - Required by classes
- Edit Class Interaction:
    - Description: In order to receive notifications for class assignments, users will need to add and edit classes.
    - Relationships:
        - Dependent on login
        - Used by weight categories
        - Used by notifications
        - Used by database
- Edit Weight Category Interaction:
    - Description: Assignments are stored in a weight category there they will need to be modified, added and removed by the user.
    - Relationships:
        - Dependent on classes
        - Dependent on weight category
- Add Class Interaction:
    - Description: Users will add classes into weight categories and assign a point value
    - Relationships:
        - Used by notifications
- Send Notifications Interaction:
    - Description: The system must send notifications to user. The frequency of such notifications is calculated using the weight category and the assignment point value to get an overall assignment value. The higher the assignment value, the more frequently notifications are sent.
    - Relationships:
        - Uses assignments
        - Uses weight categories
        - Uses classes

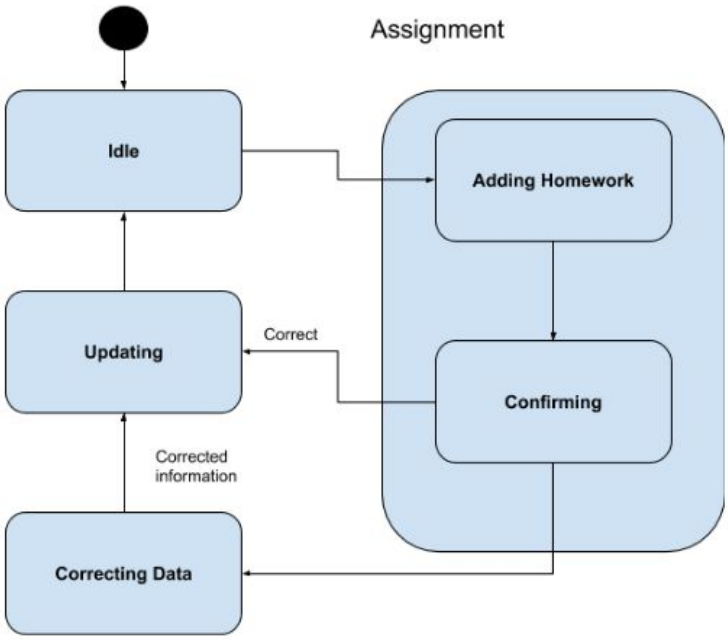
- Modify Database Interaction
    - Description: All changes to the data must be made to a database. This interaction is responsible for making sure each change to the program state gets saved so it can be loaded next runtime of the app. All classes, weights, and assignments are saved in the database along with their corresponding values. The most recently sent notifications for each assignment are also saved in the database to keep track of when the next one should be sent.
    - Relationships:
        - Dependent on login
        - Uses classes
        - Uses weights
        - Uses assignments
        - Uses notifications
- GUI_2 Updating
    - Description: Every time a modification is made, the GUI_2 needs to be updated. First, the data modified data is saved to the database and a refresh signal that contains a list of classes to update is sent to the GUIManager. That data is loaded from the database and the GUI_2 is updated.
    - Relationships:
        - Uses database

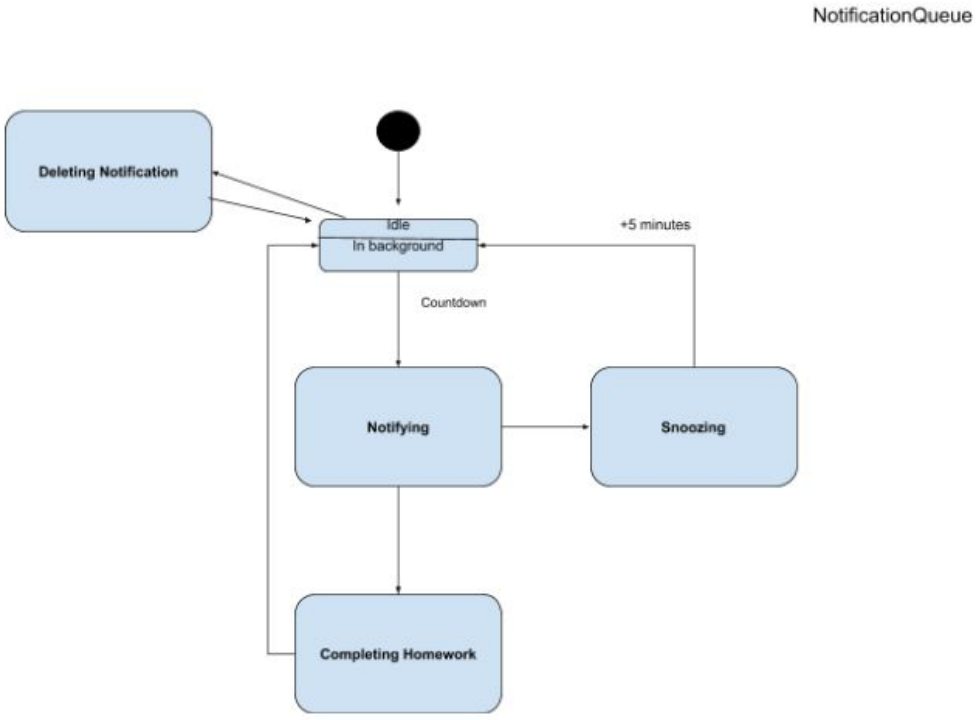
4.2.2 Interaction Viewpoint





4.3 State Dynamic Viewpoint

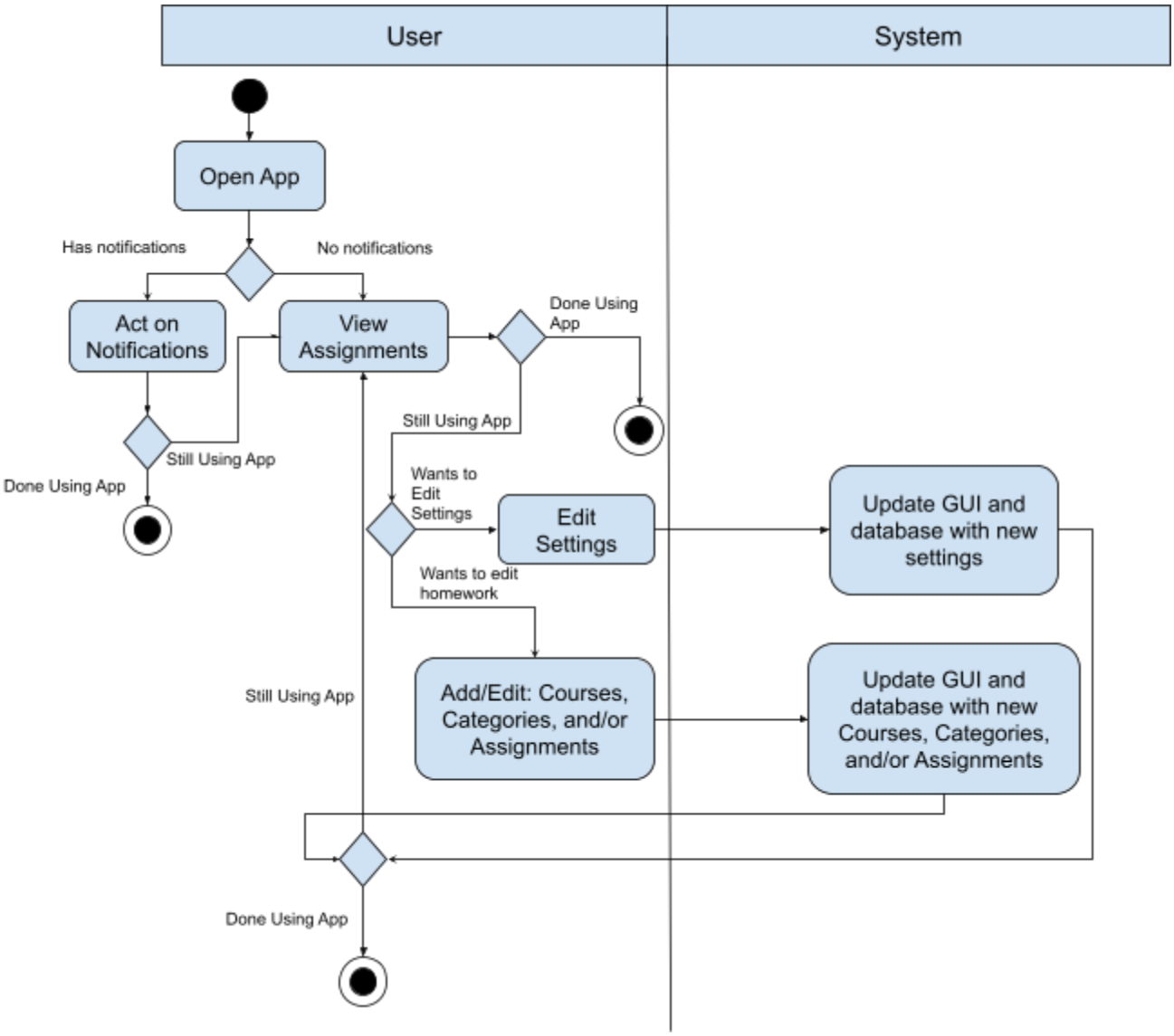




- Description
    - The state diagram shows the different states of each class and how it responds to various events. For example, adding an assignment. When the user opens the software, it will begin in an idle state, waiting for the user to add an assignment. Once added, the software will be in a state of confirming, waiting for user confirmation. If the information is correct, it will update. If not, the software will let you make changes, then will correct the data. Once corrected, it will update. After everything is done, it will go back to idle.

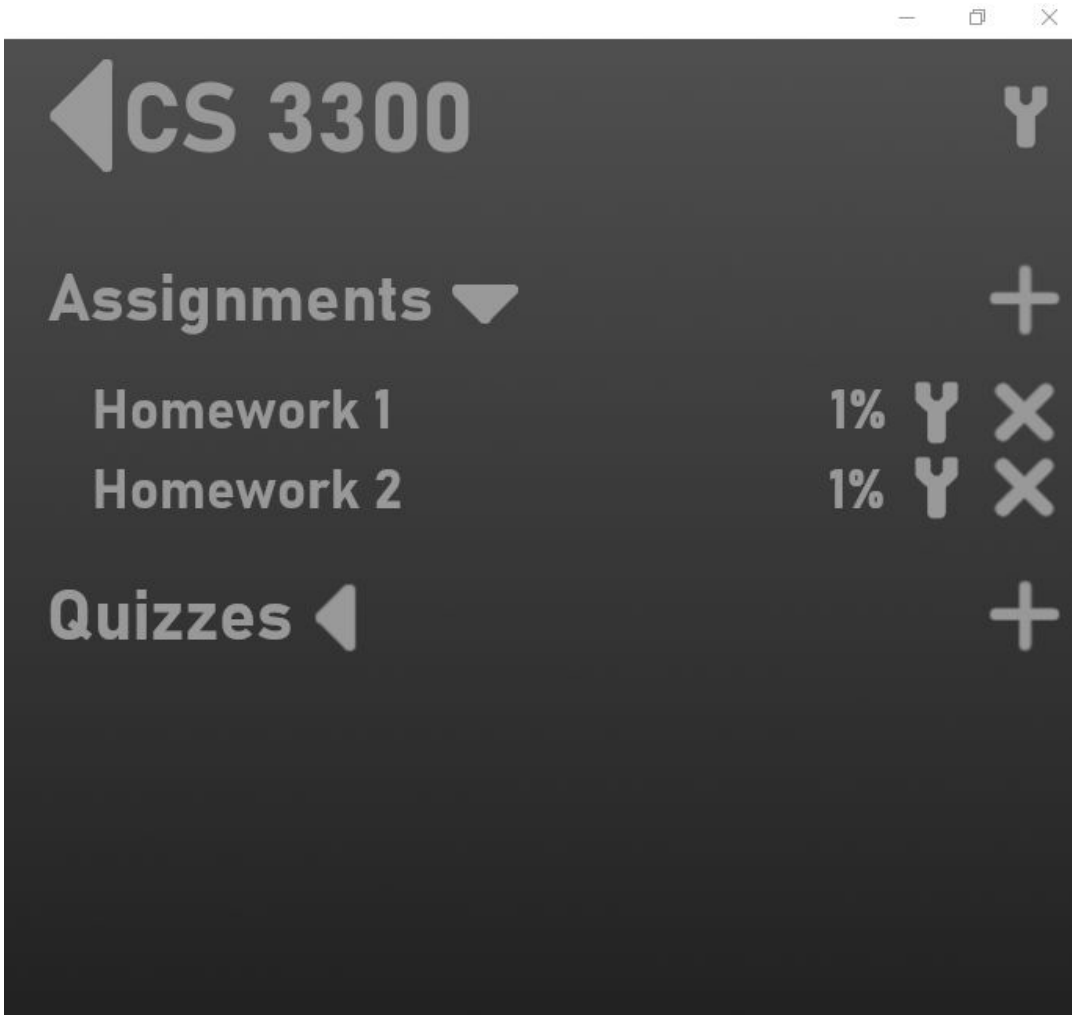
4.4 SRS Diagram Updates:

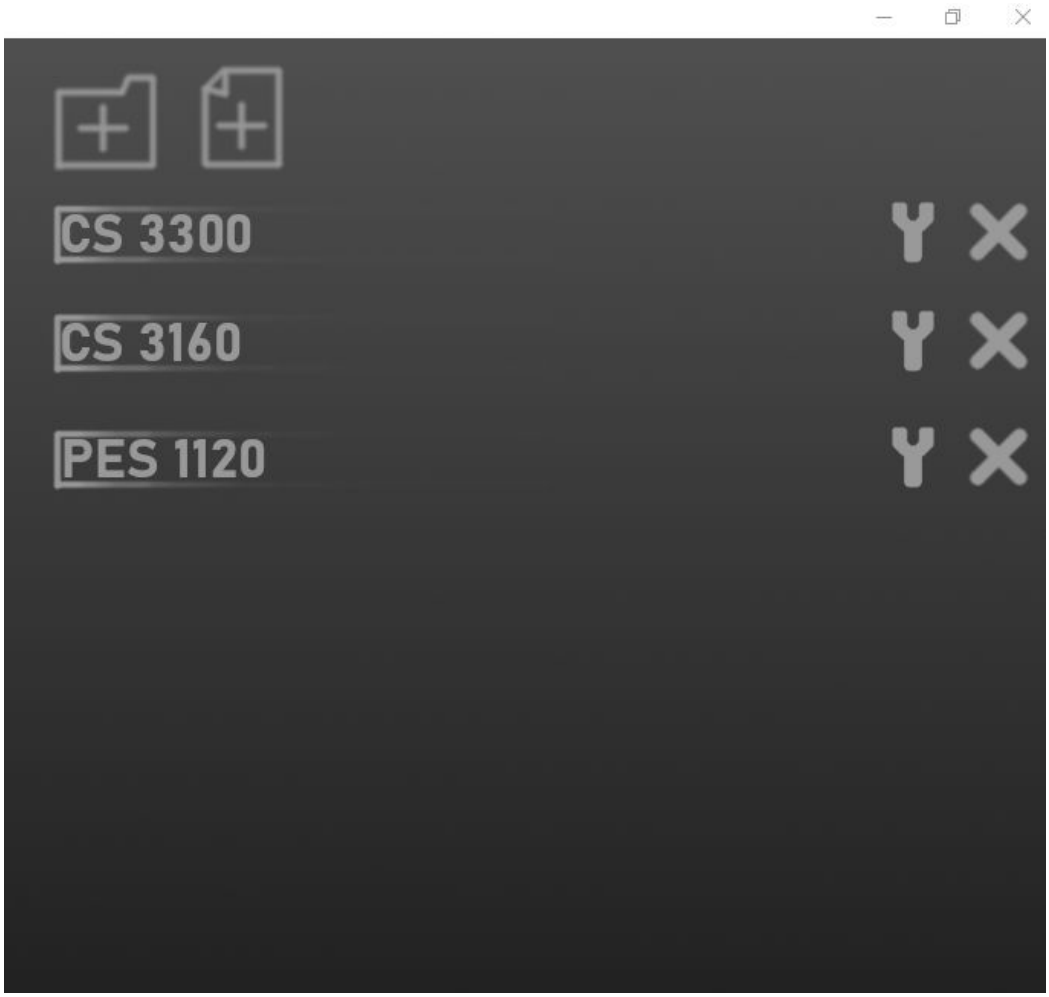
- Updated AD_4:



- Description:
    - Our old AD_4 only showed the process of adding a new assignment. Now that we have a better overview of our applications behavior, we have updated our activity diagram to show the general use of our application. The user first acts on notifications if they have them, and then view their assignments. Then the user can add/edit their homework if they need to, and the system will update the database and the GUI_2. Then the user returns to viewing their assignments. At any point in this process the user may close the application and be done using it.

5. Human Interface Design





6. Updates to SRS

N/A

7. Glossary/References

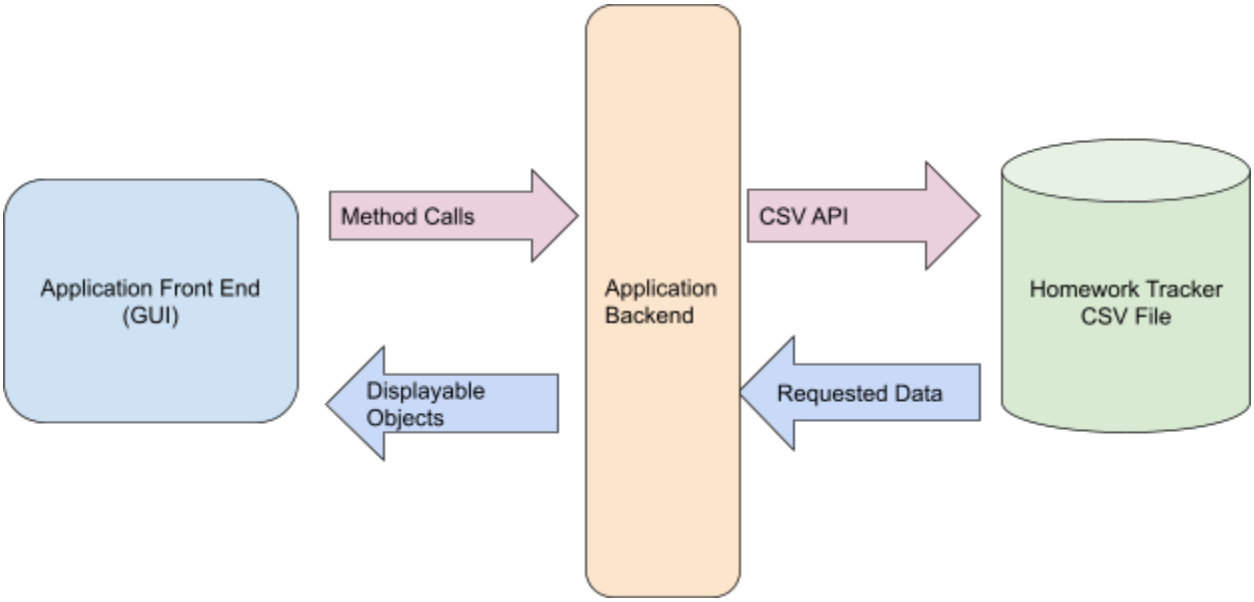
- Acronym Definitions:
  - UI/UX[1]: User Interface and user Experience
  - GUI[2]: Graphical User Interface
  - SQL[3]: Standard Query Language
  - AD[4]: Activity Diagram
  - CSV[5]: Comma Separated Values
  - API[6]: Application Programming Interface
- References:

  N/A

8. Prototype Design

8.1 Prototype Architecture

- Diagram:



- Description:
    - Our prototype will use a three-tiered architecture. We use a GUI_2 to interact with the user. When the user makes choices that require CSV_5 file information, like viewing assignments and changing settings, our GUI_2 will request that information from our application backend. Our application backend will then communicate with the CSV_5 file using a CSV_5 API_6, to get the requested information. After this, the application backend wraps the requested data in an object and sends it to the GUI_2 to be displayed.

8.2.1.1 Prototype Logical Viewpoint: Classes and their Relationships

- **Course**
  - Description:
    - Models a class the user is taking. A Course has a name, total point amount, a credit hour amount, and a list of GradeWeightCategories. A Course contains data pertaining to a class the user is taking.
  - Relationship to other classes:
    - A Course is composed of many GradeWeightCategories
    - Course uses GradeMath
    - Courses are persisted by CsvLink
    - Courses are created by HomeworkInputer
    - Courses are displayed by HomeworkDisplayer

- **GradeWeightCategory**
  - Description:
    - Models a grade weight category in a course. A GradeWeightCategory has a name, a list of Assignments, and a weight, which is a floating point number that multiplies the points of the Assignments in this GradeWeightCategory. A GradeWeightCategory contains assignments that are worth a certain percentage of the overall Course grade.
  - Relationship to other classes:
    - GradeWeightCategories are owned by a Course
    - A GradeWeightCategory is composed of many Assignments
    - GradeWeightCategory uses GradeMath
    - GradeWeightCategories are persisted by CsvLink
    - GradeWeightCategories are created by HomeworkInputer
    - GradeWeightCategories are displayed by HomeworkDisplayer

- **Assignment**
  - Description:
    - Models an assignment in a grade weight category. An assignment has a name, a due date, a point amount, ~~a start time, and an end time~~. An Assignment contains data that pertains to homework that the user needs to complete.
  - Relationship to other classes:
    - Assignments are owned by a GradeWeightCategory
    - Assignment uses GradeMath
    - Assignments are persisted by CsvLink
    - Assignments are created by HomeworkInputer
    - Assignments are displayed by HomeworkDisplayer

- **HomeworkInputer**
    - Description:
        - A class that gets homework data from the user. A HomeworkInputer is responsible for getting and validating user input via the GUI_2 and creating the appropriate object from the input.
    - Relationship to other classes:
        - HomeworkInputer creates Courses
        - HomeworkInputer creates GradeWeightCategories
        - HomeworkInputer creates Assignments
        - HomeworkInputer is used by GUIManager

- **Notification**
    - Description:
        - A class models the notifications that will be sent to the user, via the Windows 10 notification tray. A Notification has a message which is a string~~, and a sound to be played~~. A notification can be displayed, ~~snoozed,~~ or dismissed (which is handled through the Windows 10 notification tray).
    - Relationship to other classes:
        - Notifications are created by NotificationGenerator
        - Notifications are owned by NotificationQueue
        - Notifications are persisted by CsvLink

- **NotificationGenerator**
    - Description:
        - A class that creates notifications. A NotificationGenerator creates a new Notification when an Assignment indicates that it's deadline is within 24 hours.
    - Relationship to other classes:
        - NotificationGenerator receives messages from Assignments
        - NotificationGenerator creates Notifications
        - NotificationGenerator adds Notifications to the NotificationQueue

- **NotificationStatus**
    - Description:
        - An enumeration that tracks the user's current notification preference. Notification has three possible values: Enabled, ~~Muted~~, and Disabled.
    - Relationship to other classes:
        - NotificationStatus is owned by NotificationQueue
        - NotificationStatus is persisted by CsvLink

- **NotificationQueue**
    - Description:
        - A class that holds notifications and sends them to the user. A NotificationQueue has a NotificationStatus, a list of Notifications, ~~and a notification frequency which is a floating point number that controls how often notifications are sent~~.
    - Relationship to other classes:
        - A NotificationQueue is composed of many Notifications
        - NotificationQueue owns NotficationStatus
        - NotificationQueue is persisted by CsvLink

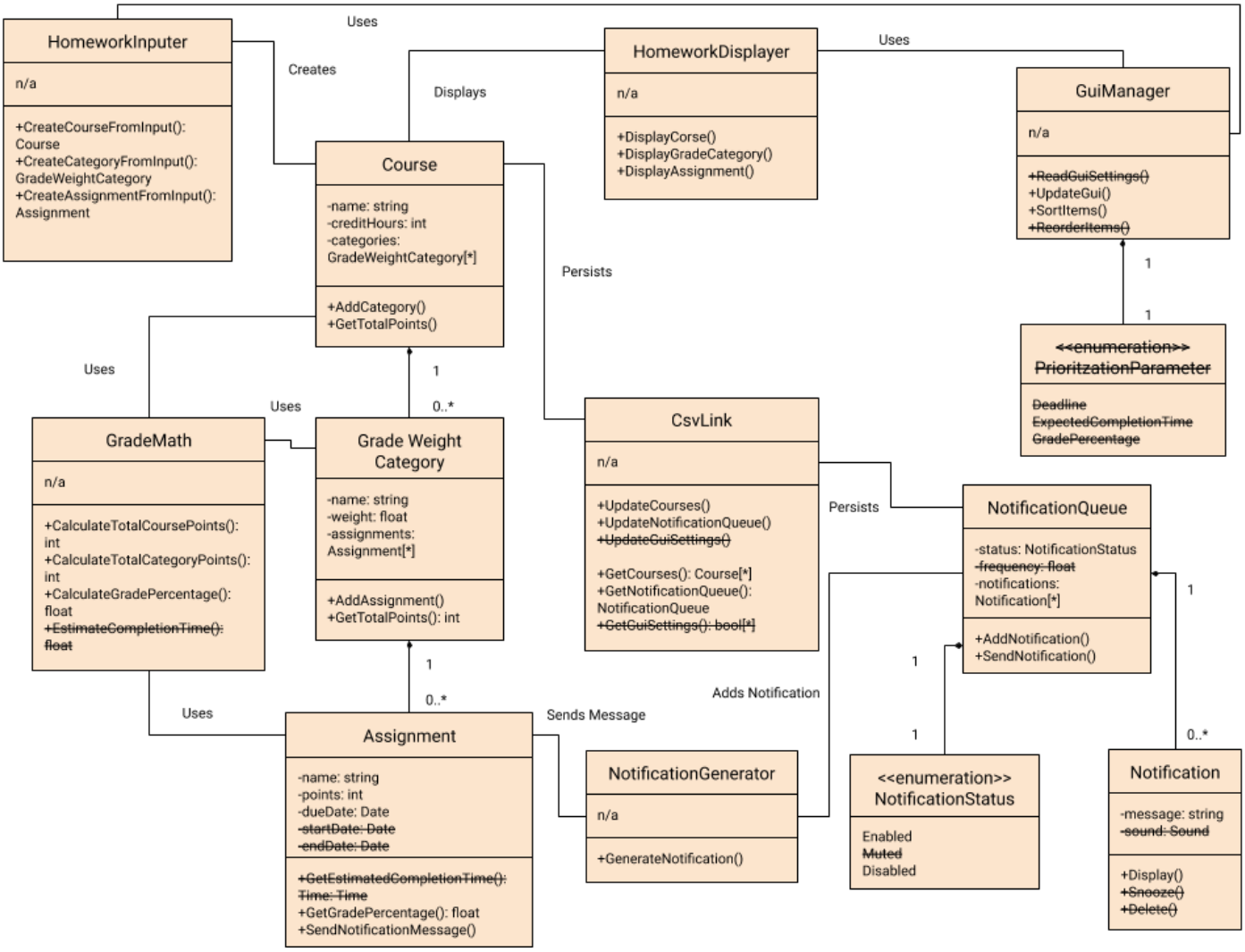
- ~~**PrioritizationParameter**~~
    - ~~Description:~~
        - ~~An enum that tracks the user's current prioritization preference. The notification status will be an enum with three possible values: Deadline, ExpectedCompletionTime, and GradePercentage.~~
    - ~~Relationship to other classes:~~
        - ~~PrioritizationParameter is owned by GUIManager~~
        - ~~PrioritizationParameter is persisted by CsvLink~~

- **HomeworkDisplayer**
    - Description:
        - A class that displays homework related objects to the GUI_2. A HomeworkDisplayer is displays Courses, GradeWeightCategories, and Assignments to the GUI_2.
    - Relationship to other classes:
        - HomeworkDisplayer displays Courses
        - HomeworkDisplayer displays GradeWeightCategories
        - HomeworkDisplayer displays Assignments
        - HomeworkDisplayer is used by GUIManager

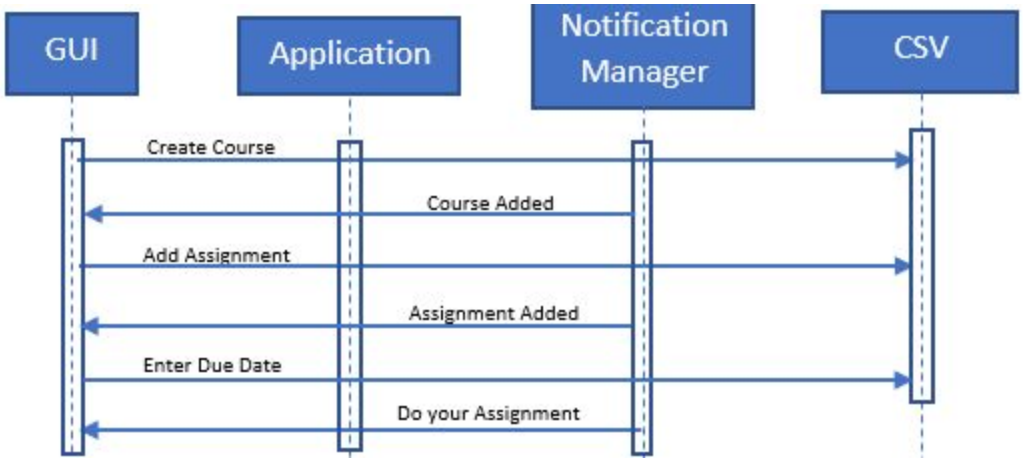
- **GUIManager**
    - Description:
        - A class that controls the GUI_2. GUIManager ~~controls the appearance of the GUI_2 and~~ displays all objects that need to be displayed in the GUI_2.
    - Relationship to other classes:
        - GUIManager uses HomeworkDisplayer
        - GUIManager uses HomeworkInputer
        - ~~GUIManager is persisted by CsvLink~~

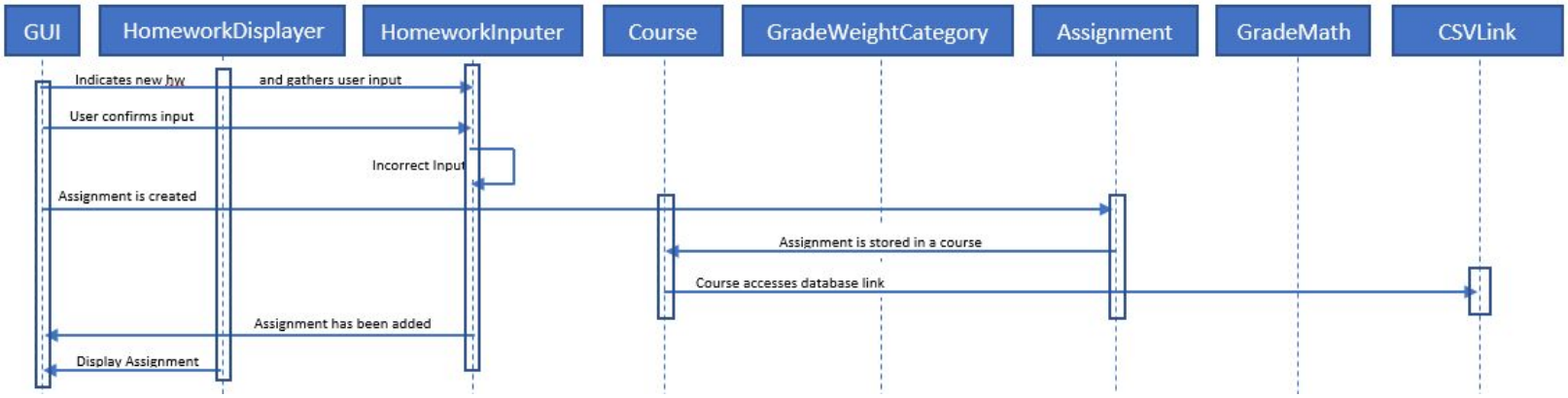
- **CsvLink**
    - Description:
        - A class that persits the application's data in the CSV_5 file. A CsvLink is responsible for updating the CSV_5 file and reading from the CSV_5 file.
    - Relationship to other classes:
        - CsvLink persists Course
        - CsvLink persists GradeWeightCategory
        - CsvLink persists Assignment
        - CsvLink persists Notification
        - CsvLink persists NotificationStatus
        - CsvLink persists NotificationQueue
        - ~~CsvLink persists PrioritizationParameter~~
        - ~~CsvLink persists GUIManager~~
- **GradeMath**
    - Description:
        - A collection of static methods for doing calculations pertaining to grades. GradeMath will be responsible for calculating: weighted grades, grade percentages, total class points, ~~and estimated completion times~~.
    - Relationship to other classes:
        - Course uses GradeMath
        - GradeWeightCategory uses GradeMath
        - Assignment uses GradeMath

8.2.1.2 Prototype Logical Viewpoint: Classes Diagram

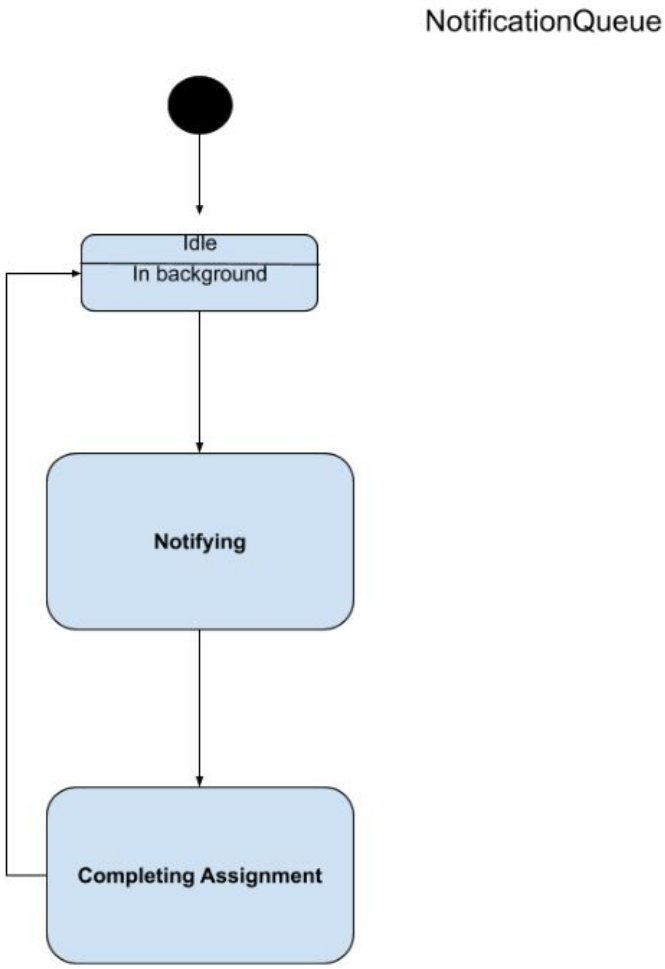


8.2.2 Prototype Interaction Viewpoint





8.2.3 Prototype State Dynamic Viewpoint



- There was no change to the assignment state diagram for the prototype.

8.2.4 Prototype SRS Diagram Updates

No change for prototype, see section 4.4.