Final Project - OO Architecture (Gradebook)

The purpose of an electronic gradebook used in grade schools in public school systems is to provide an interface for teachers to easily input and change grades for their classes and to streamline the collection and distribution of grades to students, parents and the school database. Every teacher has about 5-7 classes for a school year, with any number of students in their classes and a number of assignments with different weights.

Once a particular term is over, teachers then submit their grades to a centralized system, where it is distributed to a centralized school database and to students and their parents. The grades are pulled from teachers at the end of each term and are split by the student's ID number and then the grades are accumulated to be presented to the students and their parents.

There are typically 3 to 4 terms in a school year, and they will be aggregated to form a final grade for the student. Honors students and AP students grades are calculated differently from regular courses. They are then stored in a database containing students and their cumulative grades during their entire time in that particular school system.

For this project, I want to create a gradebook system that simulates the presence of multiple teachers and the school system pulling each of their grades at the end of each term. The UI will be written in JavaFX, and be linked to a database that collects all the grades. Interaction with the database will be done with Hibernate.

To simulate printing report cards at the end of each term, a signal will be sent to the database that will pull all the grades, convert them to a csv file, and then pass them to a message queue to be processed and distributed to parents.

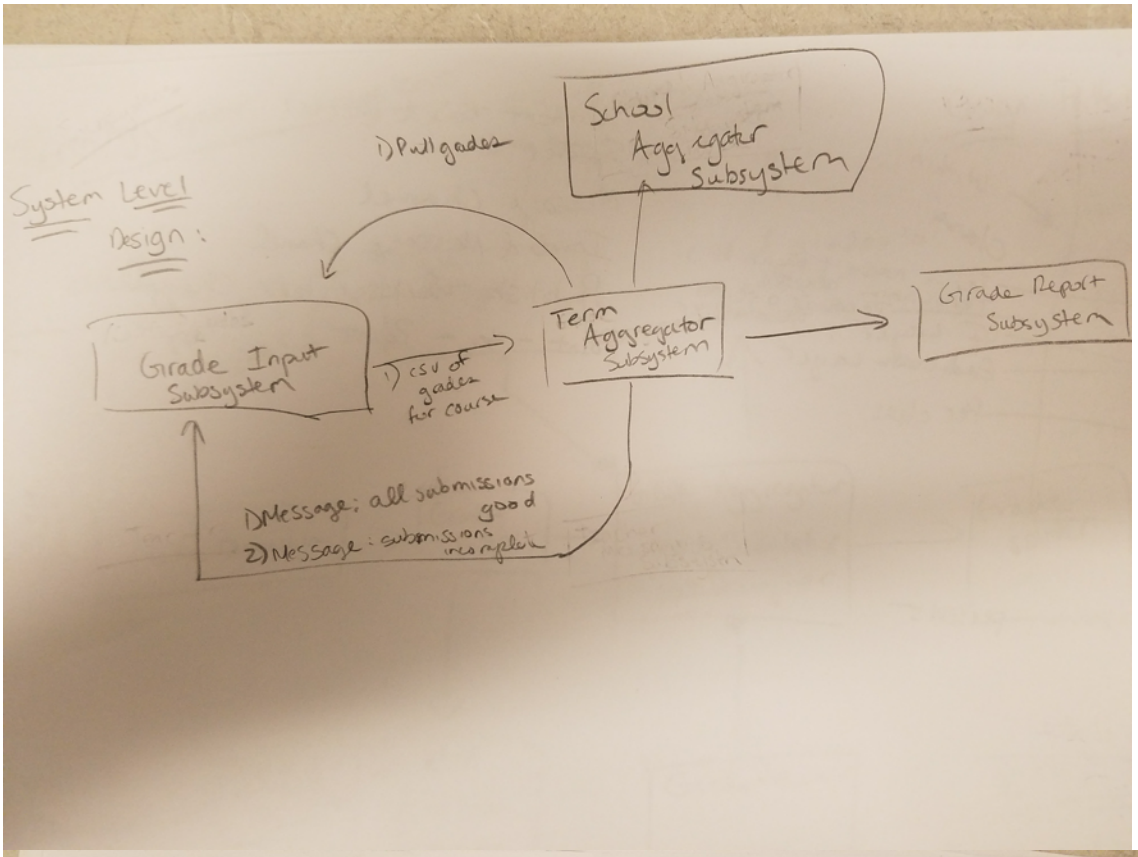
Letters by email will be sent to the following students who face the following conditions:

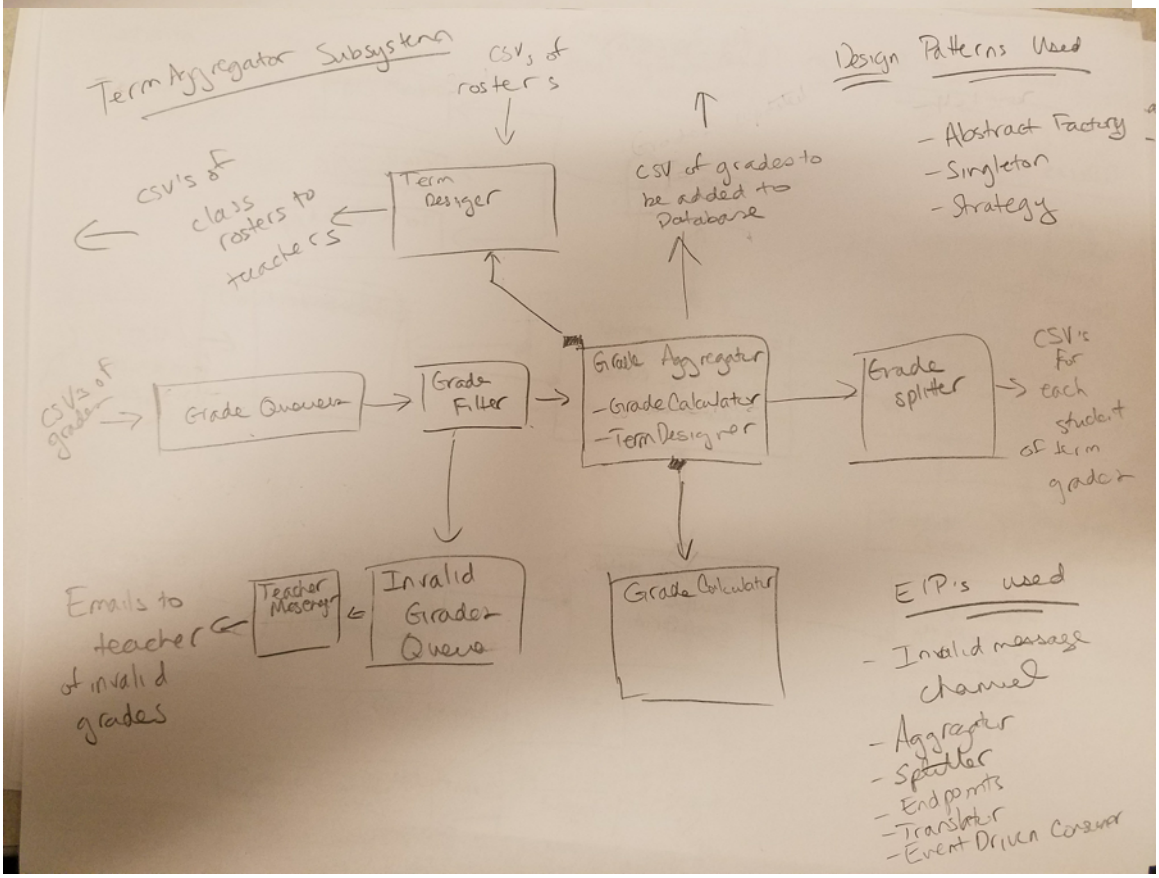
- They are on Principal's Honor Roll
- They are on Honor Roll
- They are failing a class
- They have a cumulative GPA of less than 2.0

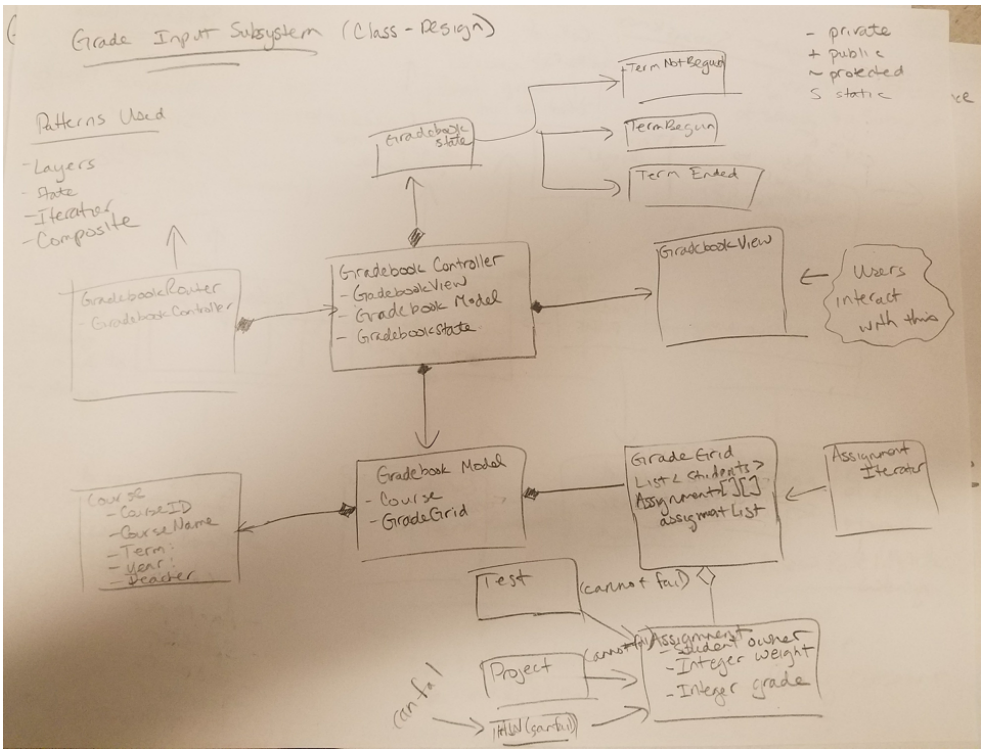
Teacher's gradebooks that are invalid under the following conditions:

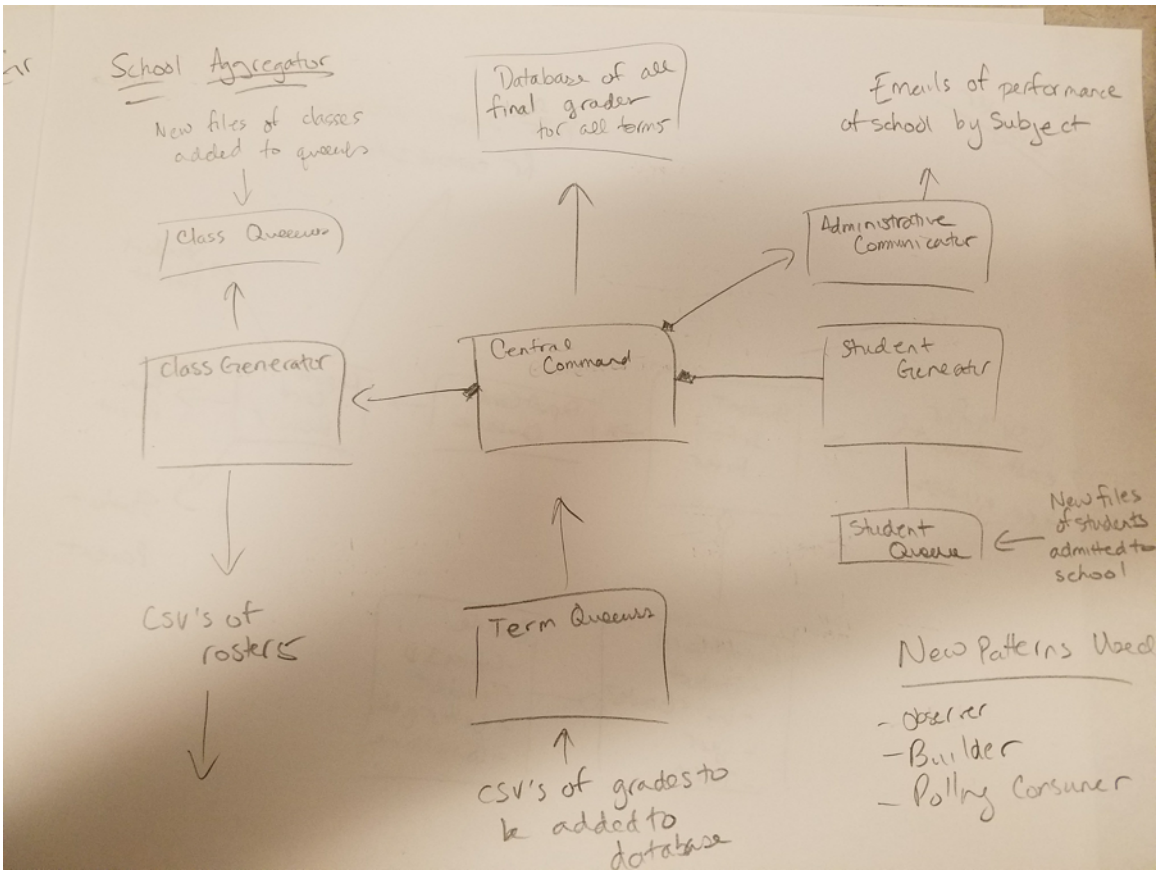
- No grades have been entered for a student for a given term and GPA cannot be calculated
    - Message will be sent to a queue, and then teachers will have to within a certain time period manually push their grades to the queue with a cumulative term grade inputted.

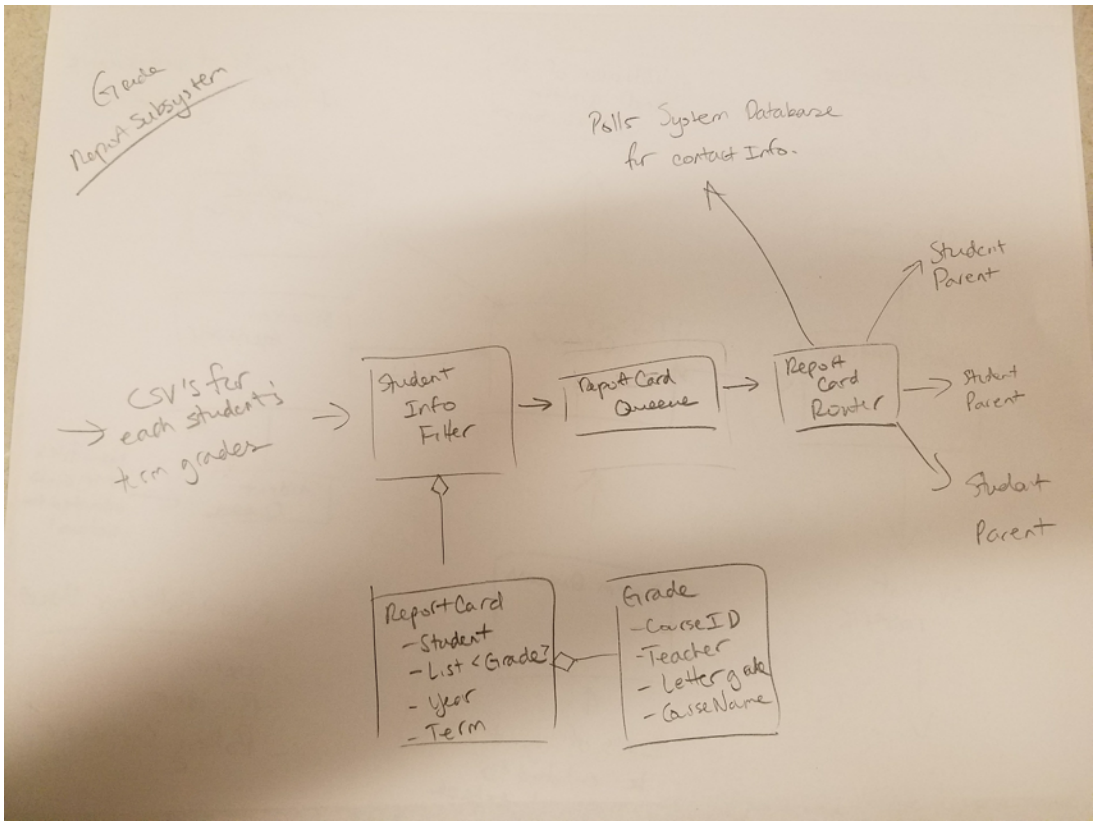
The following is the class/system design:


System Level Design:
1) Pull grades
School Aggregator Subsystem
Grade Input Subsystem
1) csv of grades for course
Term Aggregator Subsystem
Grade Report Subsystem
2) Message: all submissions good
2) Message: submissions incomplete


Term Aggregator Subsystem
CSV's of rosters
CSV's of class rosters to teachers
Term Designer
CSV of grades to be added to Database
Design Patterns Used
- Abstract Factory
- Singleton
- Strategy
CSVs of grades
Grade Queuer
Grade Filler
Grade Aggregator
-GradeCalculator
-TermDesignator
Grade splitter
CSV's for each student of term grades
Emails to teacher of invalid grades
Teacher Messager
Invalid Grades Queue
Grade Calculator
EIP's used
- Invalid message channel
- Aggregator
- Splitter
- Endpoints
- Translator
- Event Driven Consumer


Grade Input Subsystem (Class - Design)
- private
+ public
~ protected
S static
Patterns Used
- Layers
- State
- Iterator
- Composite
Term Not Begun
Gradebook State
Term Begun
Term Ended
Gradebook Controller
- GradebookView
- Gradebook Model
- GradebookState
Gradebook View
Users interact with this
Gradebook Model
- Course
- GradeGrid
Grade Grid
List<Students>
Assignment[][]
assignment List
Assignment Iterator
Course
- CourseID
- CourseName
- Term:
- year:
- teacher
Test
(cannot fail)
Project
Assignment
- student owner
- Integer weight
- Integer grade
(can fail)


School Aggregator
New files of classes added to queues
Class Queues
Class Generator
CSV's of rosters
Database of all final grades for all terms
Emails of performance of school by Subject
Administrative Communicator
Central Command
Student Generator
Student Queue
New files of students admitted to school
Term Queues
CSV's of grades to be added to database
New Patterns Used
- Observer
- Builder
- Polling Consumer



Some use cases include:

- Send an email to parents when student has principal's honor roll, honor roll or are failing
- Send an email to teachers when their gradebook is incomplete/invalid
- Update cumulative GPA over entire time at school.
- Analyze performance in classes based on subject.

Ideally, database should be queryable and I should be able to get a number of interesting facts.

The following enterprise integration patterns (EIPs) and design patterns will be used:

Design Patterns:

a. Singleton
b. Layers
c. State
d. Iterator
e. Composite
f. Strategy
g. Observer
h. Builder
i. Abstract Factory

Enterprise Integration Patterns

a) Polling Consumer
b) Event Driven Consumer

c) Invalid Channel
d) Message Router
e) Aggregator
f) Splitter

This project is quite ambitious, so if it I notice that it ends up taking up too much time, I will hardcode all data into text files, cut out the database, Hibernate and UI and work with plain text.