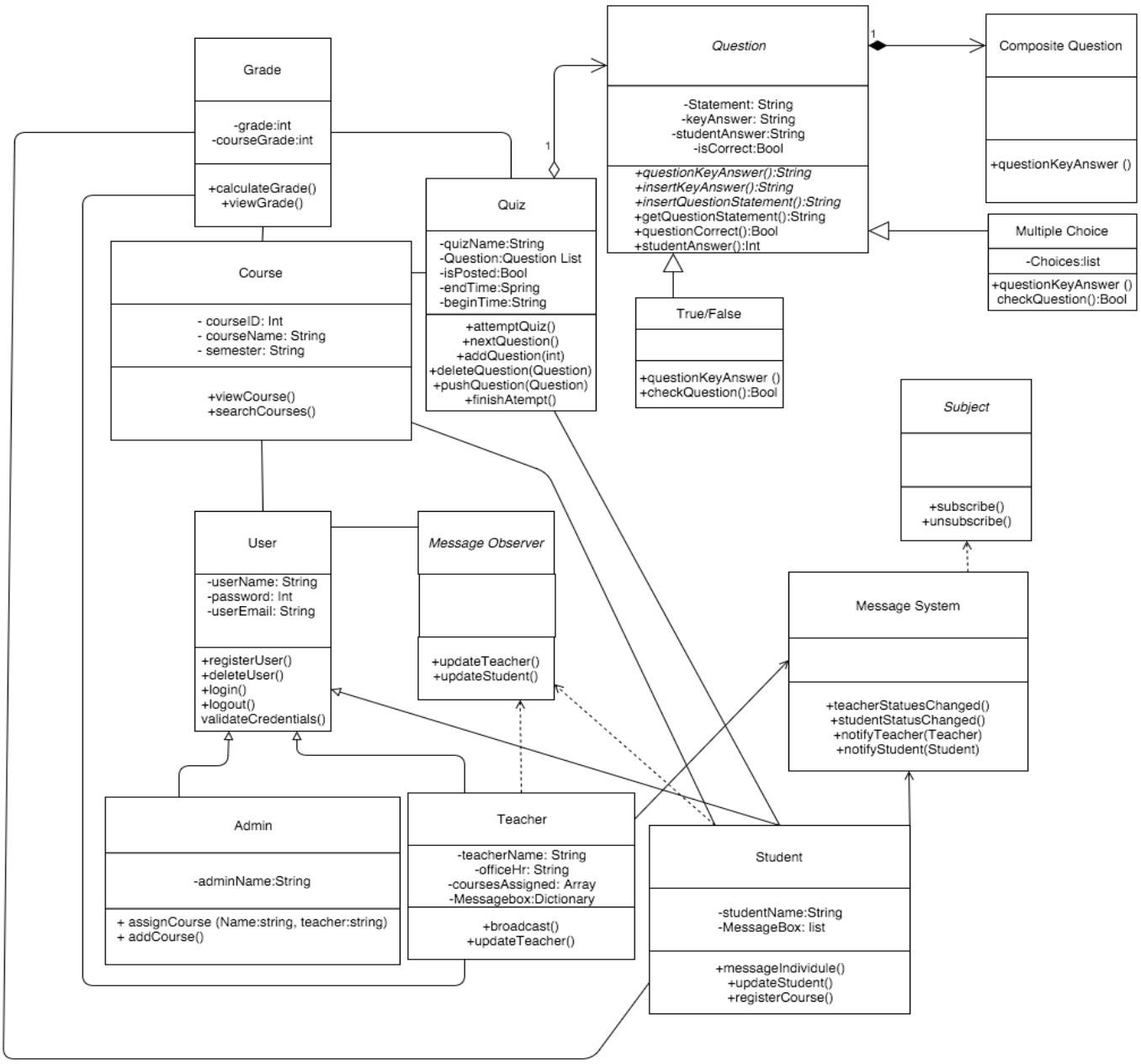
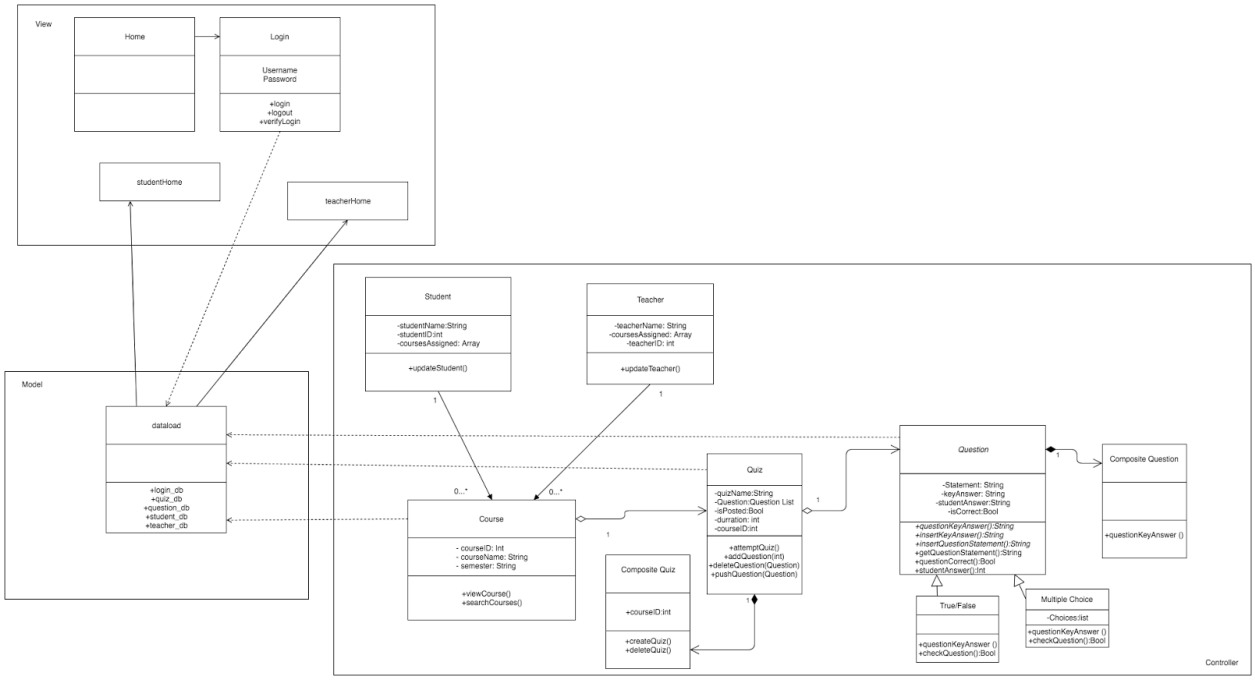
1. Features implemented:
    a. Login for both teacher and student
    b. Teachers can choose semesters, see all their courses for that semester, see all quizzes for that course, make quizzes, designate time limits to the quizzes, and add questions, and their answers, to their quizzes.
    c. Students can choose semesters, see all their courses for that semester, see all quizzes available for that course and view the questions for the quiz.
2. Features not implemented:
    a. Students cannot take the quiz
    b. There is no grade feedback
    c. There is no message system
    d. People must be manually added to the database, they cannot make an account
    e. Administrators were not considered.
3. Class Diagrams:
    a. Part 2



b. Final



c. Notable changes: no message systems, no grade feedback, no administrators

4. Design patterns:
    a. Composite was used in both the implementation of quiz (a composite of question) and in the implementation of course (a composite of quizzes)
    b. We used MVC to separate the logic in the app and make it more easily understandable.
    c. We intended to use the observer pattern in making the message system, but as that system was deemed too complex for the limited time available it was not used.
5. We learned that attempting to design a system while learning about different design patterns was difficult. Because we were learning about the different patterns, many of our group arguments came about because we didn't understand what could be done or how to use the patterns correctly. In having these discussions, we did become more familiar with the patterns and understood what they represented. By designing the system initially, we were able to implement the system with fewer in person meetings to clarify what was needed. We learned that doing the paperwork could be an extreme advantage for large, complex systems. But we also learned that if the people who do the paperwork don't understand what can be done the paperwork is less than helpful. We learned that communicating ideas doesn't always work even if you have the tools, and that many things came up in the actual coding that were design decisions that were not anticipated during the paperwork. Ultimately, while paperwork is necessary in many cases to facilitate the communication of ideas and project accomplishments, if it is done by someone with either a weak understanding of how to put together the paperwork, a weak understanding of the idea, or a weak understanding of the available implementation tools, then it seems to be more wrong than right. The different diagrams,

and even the process of spending so much time on the diagrams should be used as a guide to insuring that everyone understands the project and implementation, not as hard and fast rules unless it is clear that the people who did the paperwork know both how to represent their ideas and what is possible.