1. Features Implemented

- Login for both teacher and student

  - We implemented a modal that you can choose to log in with either student or teacher. It will send a “type” within user class to Mysql to record this user is student or teacher.

				userID	userName	password	userEmail	type
☐	Edit	Copy	Delete	0	tommy	1234		student
☐	Edit	Copy	Delete	2	A	1234		student
☐	Edit	Copy	Delete	3	bob	12		teacher

- Teacher can view the courses they have been assigned by Admin and see all the quizzes they have made under the course.

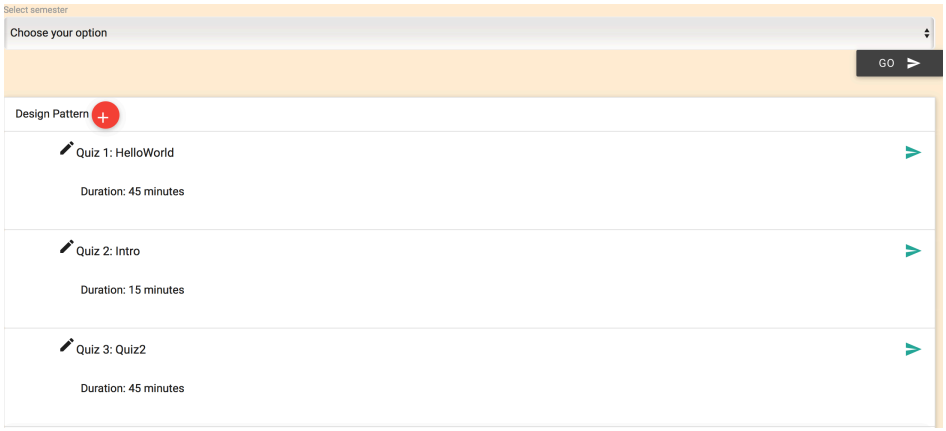
  - In the course table(class), we have set a teacher ID along with it, because every course will only have one teacher teaching at the same time.

  - So we can get all the courses the teacher has by **teacherID**.

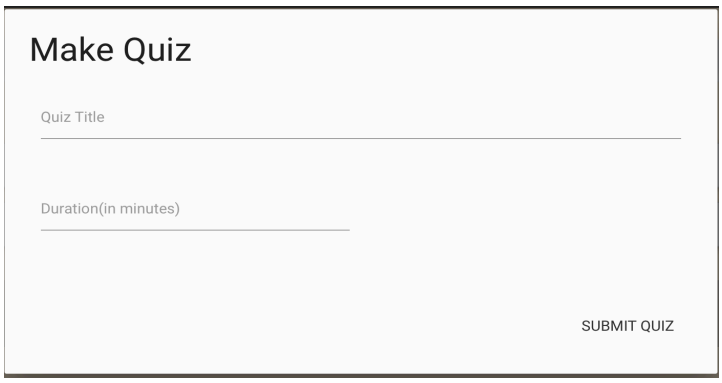
				courseID	courseName	semester	teacherID
☐	Edit	Copy	Delete	1	Algorithm	Spring 2016	2
☐	Edit	Copy	Delete	2	Design Pattern	Spring 2016	3
☐	Edit	Copy	Delete	3	UCDD2	Fall 2016	2
☐	Edit	Copy	Delete	1073	OOP	Spring 2016	3

- With the **courseID**, we can get the quizID and all the quiz Information from the quiz table.

quizID	quizName	isPosted	Duration	courseID
1000	Quiz1	1	30	1
1001	Quiz2	1	40	1
2001	HelloWorld	1	45	2
2002	Intro	1	15	2
5	OOP concept	0	20	1073
6	Quiz2	0	45	2
7	quiz3	0	30	2
8	Quiz4	0	40	2



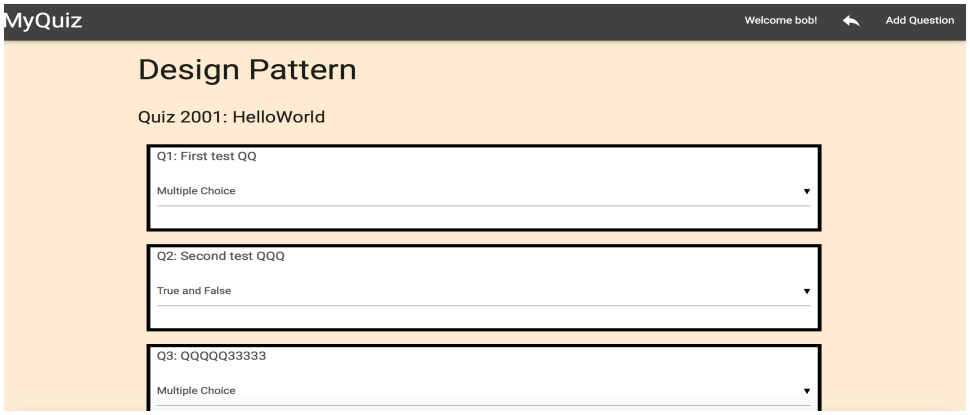
- Teacher can **makeQuiz** (makeQuiz function is under teacher class) under the course with the red button besides the course.

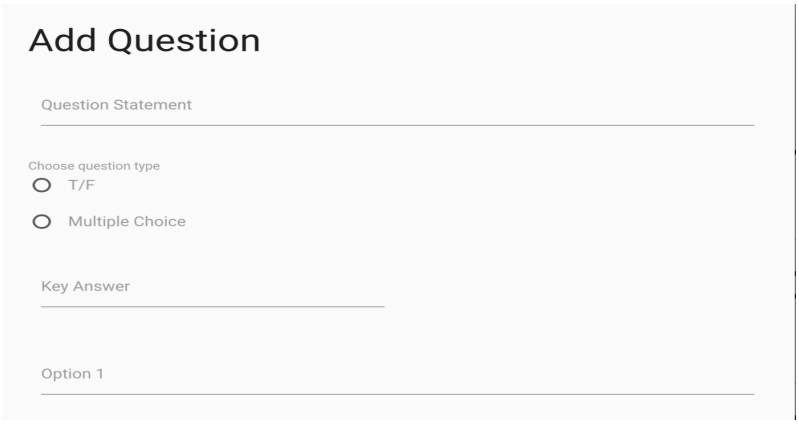


- Teacher can go to the quiz page and **add question (addQuestion function)** within the quiz class.

  ➢ Each question has a corresponding quizID so that we can reference the question to the quiz.

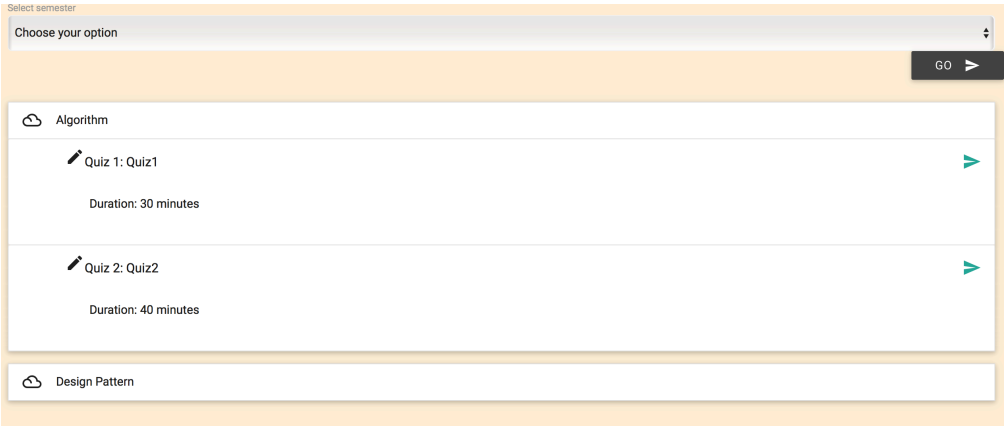
questionID	questionStatement	keyAnswer	studentAnswer	isCorrect	questionType	quizID	option1	option2	option3	option4	option5
200101	First test QQ	1	0	0	1	2001	AA	BB	CC	DD	EE
200102	Second test QQQ	2	0	0	0	2001	AA	BB			
100101	First SSS	3	0	0	1	1001	AAA	BBB	CCC	DDD	
200103	QQQQQ33333	2	0	0	1	2001	aa	bb	cc	dd	
5	Is it sunny outside today?	0	0	-1	0	5	true	false			
6	Is it sunny outside today?	1	0	-1	0	6	true	false			
7	is it sunny outside today?	1	0	-1	0	7	true	false			
8	Is it sunny outside today2?	3	0	-1	1	7	1	23	4	4	1
9	Is it sunny outside today?	1	0	-1	0	2001	true	false			





- Students can choose semesters, see all their courses for that semester, see all quizzes available for that course and link to the quizzes.

➢ The table and student class structure is similar to the teacher, using studentID to match the the courses they have.



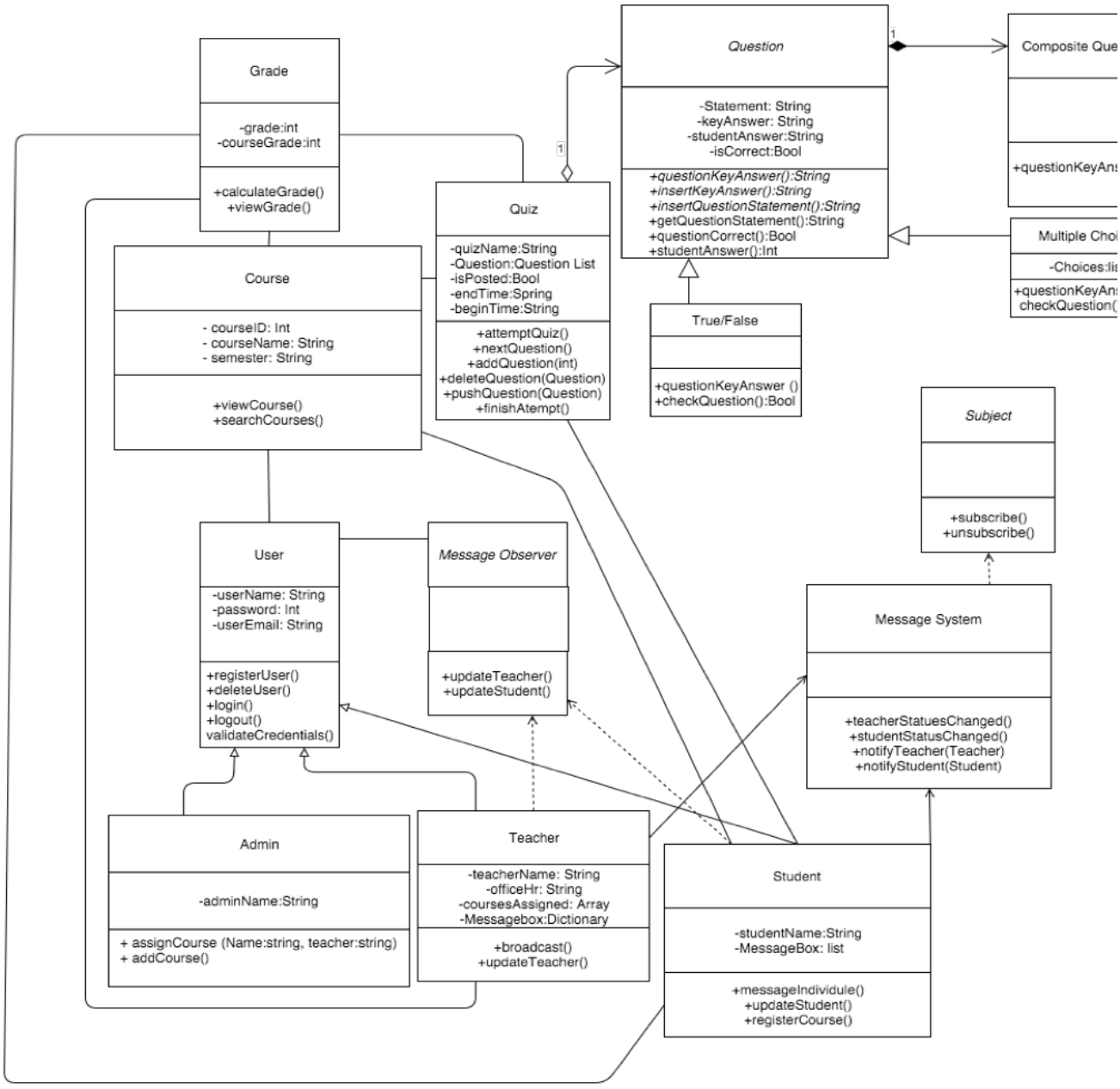
- Student can go to the quiz to **attempt Quiz** (in the quiz class)

2. Features not implemented:

- Students cannot take the quiz and answer the question.
- The grade class was not implemented.
- There is no message system.
- There is no register function.
- Administrator class was not implemented.

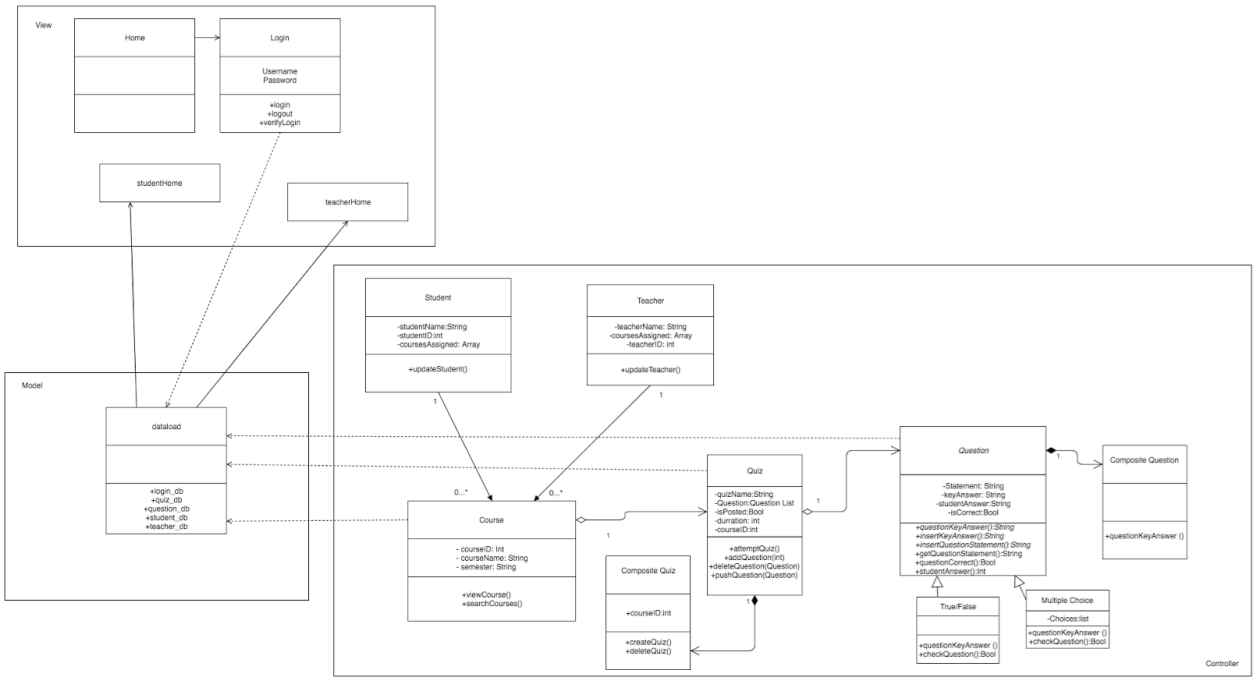
3. Class a. Part 2

Diagrams:


Grade
-grade:int
-courseGrade:int
+calculateGrade()
+viewGrade()
Question
-Statement: String
-keyAnswer: String
-studentAnswer:String
-isCorrect:Bool
+questionKeyAnswer():String
+insertKeyAnswer():String
+insertQuestionStatement():String
+getQuestionStatement():String
+questionCorrect():Bool
+studentAnswer():Int
Composite Que
+questionKeyAns
Quiz
-quizName:String
-Question:Question List
-isPosted:Bool
-endTime:Spring
-beginTime:String
+attemptQuiz()
+nextQuestion()
+addQuestion(int)
+deleteQuestion(Question)
+pushQuestion(Question)
+finishAtempt()
Multiple Choi
-Choices:lis
+questionKeyAns
checkQuestion(
Course
- courseID: Int
- courseName: String
- semester: String
+viewCourse()
+searchCourses()
True/False
+questionKeyAnswer ()
+checkQuestion():Bool
Subject
+subscribe()
+unsubscribe()
User
-userName: String
-password: Int
-userEmail: String
+registerUser()
+deleteUser()
+login()
+logout()
validateCredentials()
Message Observer
+updateTeacher()
+updateStudent()
Message System
+teacherStatuesChanged()
+studentStatusChanged()
+notifyTeacher(Teacher)
+notifyStudent(Student)
Admin
-adminName:String
+ assignCourse (Name:string, teacher:string)
+ addCourse()
Teacher
-teacherName: String
-officeHr: String
-coursesAssigned: Array
-Messagebox:Dictionary
+broadcast()
+updateTeacher()
Student
-studentName:String
-MessageBox: list
+messageIndivdule()
+updateStudent()
+registerCourse()

1.

b. Final



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

4. Design patterns

- Composite was used in both the implementation of quiz (a composite of question) and in the implementation of course (a composite of quizzes)

- We used MVC to separate the logic in the app and make it more easily understandable.

- 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. 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.