Testr - Part 4

Karthik Kannan
Mahesh Kumar Ravindranathan
Nikhil Mahendra

1. What features were implemented?

ID	Requirement	Topic Area	User	Priority
UR-001	Users should be able to login using credentials	Authentication	All	Low
UR-002	Register with a userid, password and an email id.	Authentication	All	High
UR-003	Instructors should be able to create, modify, delete tests containing various types of questions.	Test Creation	Instructors	Critical
UR-004	Instructors/Grader should be able to assign grades after the completion of each test.	Test Grading	Instructors	Critical
UR-008	Students should be able to take tests in the allotted time slots and submit it for grading.	Test Taking	Students	Critical
UR-009	Students should be able to view all the grades for the course	Test Grading	Student	Medium
UR-014	Students should be able to view their performance and areas of weakness	Performance Analysis	Student	High
UR-015	Instructors should be able to view overall performance of class as well as individual students	Performance Analysis	Instructor	High

2. Which features were not implemented from Part 2?

As there was a design change in the way that instructors are registered, some features with respect to administrator were not implemented. We added the option of the instructor registering with a passkey and their chosen login credentials instead of being added by the administrator.

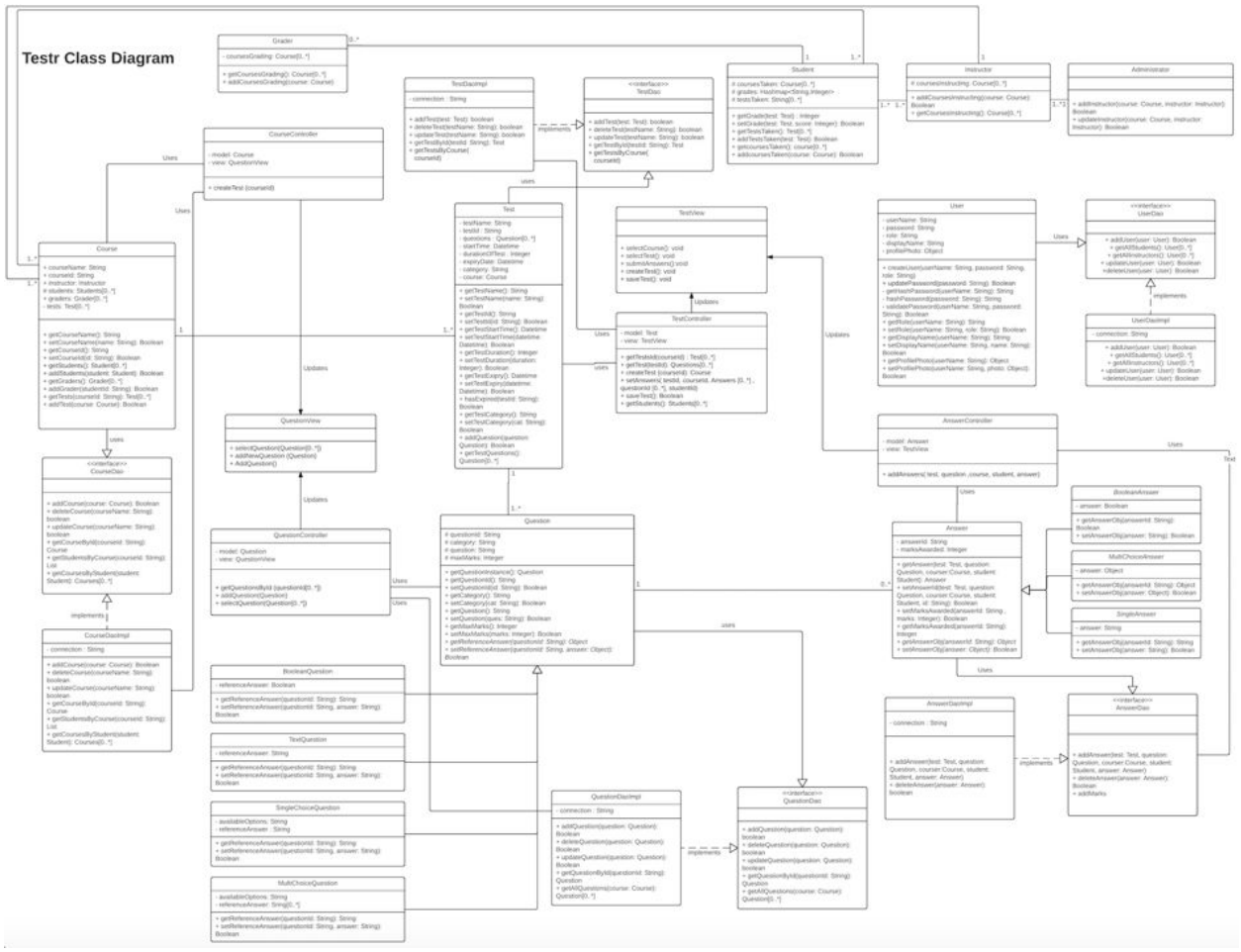
ID	Requirement	Topic Area	User	Priority
UR-006	Instructors should be able to modify grades once they have been given Feedback	Test Grading	Instructors	Critical

UR-007	Instructors should be elevate select students to the position of instructors so that they are able to create/grade tests	Administration	Instructors	High
UR-010	Students must be able to take demo tests whenever they want to check their preparedness in a subject	Test Taking	Student	Medium
UR-011	Administrator must be able to add instructors	Authorization	All	Medium
UR-012	Administrator must be able to change privileges	Authorization	All	Medium
UR-013	Administrator must verify an Instructor, before he is provided with the required privileges.	Authorization	All	Medium

3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

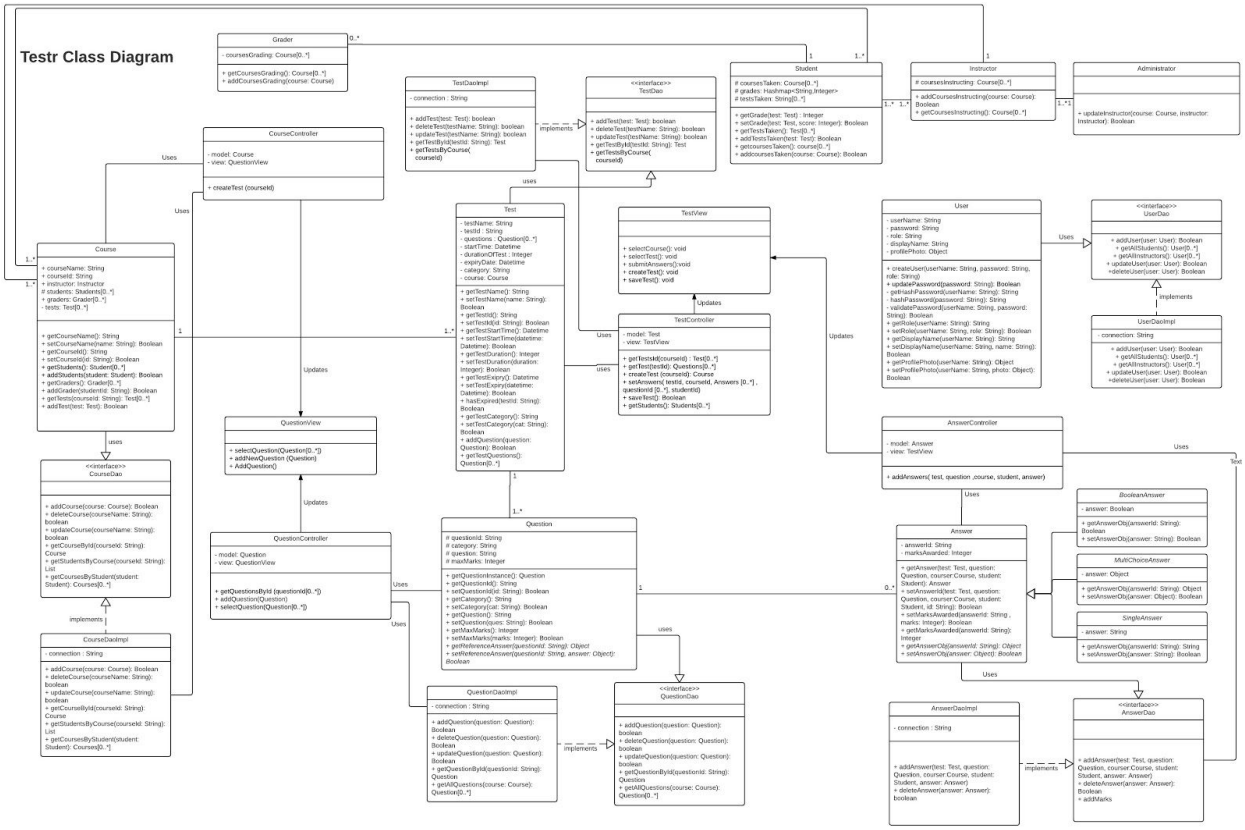
Initial Class Diagram:

https://www.lucidchart.com/invitations/accept/353d94b3-164e-4e8e-9765-44bfa66dcfd9



Final Class Diagram:

https://www.lucidchart.com/invitations/accept/09dacbd8-64e8-4a90-941a-a5c59bbcd9bc



There are minimal changes in the class diagram relating to:

- Question class creation is based on different answer types, and there is no need to have different types of question classes that inherited from the main question class.
- Administrator functions were reduced as instructor registration mechanism was changed.

Designing the system initially before implementing gave us an overview of all the components in the system, and how they interacted. With the help of class diagrams having the class signatures, the implementation logic of code for the corresponding parts were easy to add later.

4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

MVC Framework: The project was at its core, a web application and thus made heavy use of the Model View Controller framework. This helped in separating the GUI(view) of the application from the backend(model and controller). While not strictly a design pattern, the MVC framework acted as the backbone of the project.

Object Relational Mapping: ORM/ ODM was used to create persistent database

Observer Pattern: As a direct consequence of the MVC framework we used, the observer pattern played a major role in our system as a lot of our code was event driven. When events like a new test is posted, or grades for a test are posted, the students are notified.

Strategy Pattern: The behaviour of a sub-system can be switched based on certain conditions, which can be achieved with strategy pattern. Auto grading is based on the different types of answers, and their details are abstracted for grading. Based on the type of the answer, corresponding object's grade checking algorithm is executed.

Factory Pattern: This pattern is very helpful in creating generalized objects without specifying the logic pertinent to the individual or sub classes. When creating different questions, it is based on the type of answers and these answer objects are created based on the type.

Design Patterns which could have been used:

Singleton Pattern: This pattern restricts creating the creation of a class's instance to allow only one object to exist. After selecting to take a test, another object of the same test cannot be created for the same user and this could be handled with this pattern.

5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

- The process of designing the system before starting to code makes it easy to change when updating and adding features.
- We can obtain feedback for the initial design of the system, and update the design to include more efficient design principles or implementation mechanisms that could be suitable for our system.
- This is very helpful as it is easier to change the design than the code after implementation.
- It also helps us understand how changes in the code would affect other modules and reflect across the system.
- Visualizing the system with UML helps in understanding the data flow and interactions between different component and layers of the system.