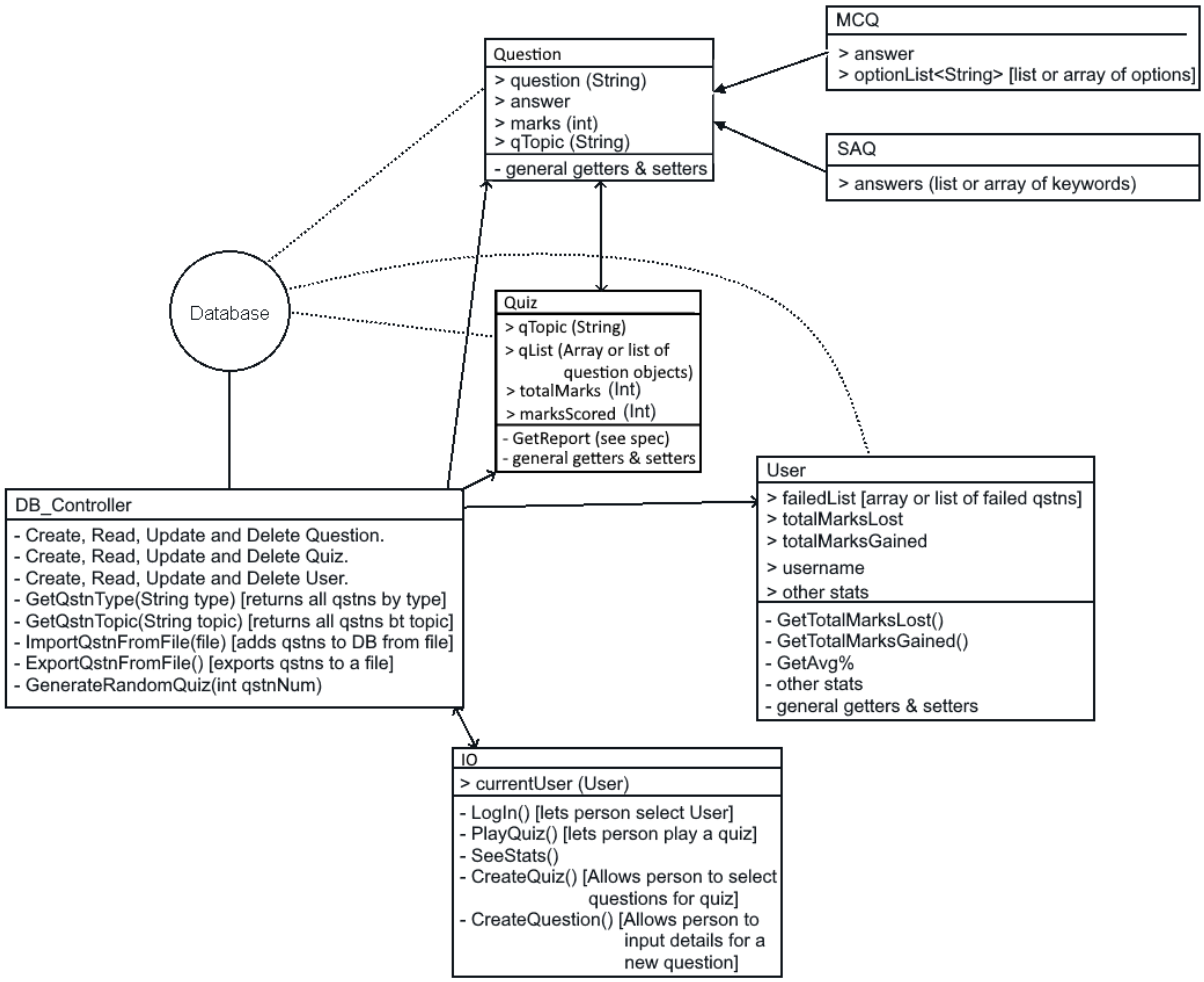
Report (Using Reflective Theory on each class)

Determining the Outline

The first practical on Tuesday I got to meet my group and we immediately set up a system to be able to divide the work up equally. .We chose that our main communication would be Whatsapp and we'd keep a regular update on our google doc where we made a table of the client requirements. In our next practical, we had a discussion about what we thought the overall design should be and agreed that we'd have a seperate table to be able to control the updating of the database. Going into it, I was excited about getting a chance to see a work environment largely based on working with others. I was hoping it would be easier than our other projects since we'd have support and help from each other. The experience was promising, I got along with my teammates and was able to have an understanding of the project. Next time, instead of tasking everyone with different parts of the project, I would make sure to divide it up in smaller parts and break it down as we go through it. I was tasked with starting the Quiz class. We drafted an initial UML diagram that we could use to start our project.



Quiz Class

Going off of this design, I started my quiz class. My initial start was creating a class and field to specify the quiz topic, total marks, marks scored and link it to a question list. I created the fields and generated getters and setters. I assigned quizID as a randomly generated ID and the rest as columns. Going into this I was really confused on how I was going to complete this without a question class. The good thing about the experience was that I learned how to create tables and columns. The first problem I ran into was when I wanted to create a question list, it immediately gave me an error on my arraylist when I assigned it as a column. I asked a few demonstrators and they informed me that I can't have a list as a column in my database which made sense. If I had to return to this experience, I would create an IO for only my code so I could test it along the way. Eventually, when week 6 videos were released I realized that we didn't have a many to many relationship between our quiz and question class. After going back to week 5 Hibernate Relationships slides, my teammate that was tasked with the question class and I were able to create a many to many relationship. I was able to collaborate and solve the issue.

Testing Class

I moved onto testing. I created a testing class and tested some of the methods in our MCQuestion and SAQuestion class which had some technical errors that I informed my teammate about and he was able to fix it. I waited for the DBController to merge so I could test the rest. When it did, I tested Creating, Reading and Updating a question and a quiz. It gave me some errors but I was able to tweak the code to run smoothly. After that I tested Randomly Generating a quiz which didn't give me the results I wanted so I informed my team mate. I was thinking we would have more errors than we did. The experience was good because I was able to determine our errors and see our code in action.

User class

After I was done with that, my teammate that was tasked with the User class got sick so she had to pull out of the project. I volunteered to undertake the user class and started right away. I started by looking at our UML diagram. I realized we also needed a many to many relationship between the user and question class. I created a many to many relationship and then started on the fields. After creating a Username field, I assigned it as a unique ID and assigned the fields totalMarksLost and totalMarksGained as columns. I was thinking the User class would be easy because the idea behind it was mostly like my Quiz class. The good thing about the experience was I was able to get through it pretty fast. After I was done, I had a discussion with my group where we determined to have only one user which eliminated the need for a User class. Next time, I would have that discussion earlier.

Import/Export from a file

After creating the User class, I decided to do importing and exporting from/to a file. This task took a lot of trial and error. I had some errors with mapping and later linking the options to multiple choice. I started by looking at the queries.setup package in the lecture code examples. I took some time to understand and analyze the code and implemented it according to my code. I was really frustrated because I didn't understand the error messages. After fixing the mapping

between the question and quiz class, my code ran smoothly and was able to generate a table of questions and populate it. This taught me that a small error in an unrelated place could affect my code and next time I would make sure all the major parts are working before starting to implement a new method.

Overall

Overall, my expectations were that I would be able to collaborate and have help along the way. The experience taught me that it's always a good idea to overlook some of the teammates and to make sure all the parts are being met. The experience taught me how to collaborate and what working together with other people loosely looks like. It was also not ideal to try to complete methods without having the rest of the code. We weren't able to complete our IO in time and next time I would divide the workload differently and create an IO for each class separately and put them together in the end. I could extend this project next time by adding more features and more users. Implementing a Graphical User Interface could also be the next step.

File/Method Created or Modified	Date	Description of work
Organizing the team workload and distributing tasks	01/03/2002	Talked about the outline
Talked about the Entity Relationship diagram	04/03/2022	Created and discussed the rough outline of the project
Created a Quiz.java class	04/03/2022	Created a class that will store the questions
Created fields Quiz ID, qTopic, totalMarks and marksScored in Quiz.java	04/03/2002	Created fields that will store quiz ID and quiz topic
Created getters and setters in Quiz.java	08/03/2022	Created methods that have a purpose of getting and setting a function
Attempted to create an arraylist in Quiz.java	09/03/2022	Created a field in Quiz.java that has a list of questions
Created a constructor for Quiz.java	11/03/2022	Created a constructor
Created Testing.java	11/03/2022	Created a class that is used for testing and is used to test the various classes
Created a multiple choice question and set attributes in Testing.java	11/03/2022	Initialized a multiple choice question and tested it using getters and setters
Created a short answer question and set attributes in Testing.java	11/03/2022	Initialized a short answer question and tested it using getters and setters

Discussed how to do the mapping and attempted to map many to many with Quiz.java and Question.java	12/03/2022	Created a mapping between Question.java and Quiz.java
Created the User.java class	16/03/2022	Created a class that allows to create multiple users
Assign username as ID in User.java	16/03/2022	Created a field called username and assigned it as a unique ID
Joined User.java with Questions.java with a many to many relationship	16/03/2022	Joined User with a many to many relationship to question class
Merged DB-Controller with master	17/03/2022	Merged the DB-Controller branch with master to test the various methods
Modify the CRUD methods in QuestionController.java to take parameters	17/03/2022	Modify QuestionController.java to take parameters instead of hardcoding fields
Modify RandomGenerate.java	17/03/2022	Fix errors in RandomGenerate.java, instead of using Question.getid it uses question.getId
Create a method in Testing.java to read from a csv file	17/03/2022	Created a method that reads from csv file using a constructor and the scanner object
Create a method in Testing.java to update the csv file	17/03/2022	Created a method that updates a csv file using Session object

Add a CSV file	17/03/2022	Add and populate the csv file with sample data
Fix many to many mapping of questions and quiz class	17/03/2022	Instead of mappedby = QuestionID , changed it to questions
Changed the parameter for adding to the csv file from questions to mcquestions	17/03/2022	Changed the parameter for adding to the csv file from questions to mcquestions in order to map options
Added a constructor to MCquestion	17/03/2022	Added a constructor that takes in parameters of a list of options
Populate questions.csv	17/03/2022	Populate questions.csv with sample data
Map option to question while reading and updating the csv file	17/03/2022	The csv file reads the options and updates the database
Add a check to see if type is MCQ to add options	17/03/2022	Adds options only to the MCQ Questions in the CSV file
Use Setters to Read the CSV file instead of the constructor	17/03/2022	Uses setters in the CSV file to read the data and update it
Delete unused constructor	17/03/2022	Delete the question constructor that is not used

Add a to string method to Question.java and Quiz.java	22/03/2022	Add a tostring method that prints out a human readable version of the question and quiz class
Add a getTotalMarks(); method in Quiz.java	22/03/2022	Add a method that calculates and returns the total marks of a quiz
Create getter and setters for the USER class	22/03/2022	Create methods that get and set the username and the overall marks of a user
Created a new column in Question.java called check answer	23/03/2022	Created a new column that checks if the answer is correct
Create updateCheckAnswer() method in Question.java	23/03/2022	Created a method that checks if the answer is correct and if it is updates the answer column to 1
Delete USER class	23/03/2022	Decided to have one user and deleted the User class
Create addToCSVFile.java	24/03/2022	Create a class that reads questions.csv file and updates the database.
Move reading and updating from a csv file to a new class	24/03/2022	Move the reading and updating method from testing.java to addToCSVFile.java
Test Question Controller in Testing.java	24/03/2022	Initialize and test the question controller class

Add an OptionCreat() method in QuestionController.java	24/03/2022	Add an OptionCreat() method that links the options to the questions.
Test Quiz Controller in Testing.java	24/03/2022	Initialize and test quizcontroller.java
Test Random Generator in Testing.java	24/03/2022	Initialize and test RandomGenerator.java
Write a README.md	25/03/2022	Write a readme that outlines how to run the code