School Quiz Program

Charlie Cheung

Submission Date: xx/xx/xxxx

Ccc15@aber.ac.uk

Table of Contents

[Introduction 2](#_Toc165947117)

[Analysing The Problem 2](#_Toc165947118)

[Use Case Diagram 2](#_Toc165947119)

[Design 3](#_Toc165947120)

[Class Diagram 3](#_Toc165947121)

[Initial Class Diagram 3](#_Toc165947122)

[Final Class Diagram 3](#_Toc165947123)

[SchoolQuizApplication 4](#_Toc165947124)

[Module 4](#_Toc165947125)

[Bank 4](#_Toc165947126)

[Question 5](#_Toc165947127)

[SingleChoiceQuestion 5](#_Toc165947128)

[FillTheBlanks 6](#_Toc165947129)

[Quiz 6](#_Toc165947130)

[Scoreboard 7](#_Toc165947131)

[FileSchool 7](#_Toc165947132)

[Pseudo Code Example 8](#_Toc165947133)

[Testing 9](#_Toc165947134)

[Test Table 9](#_Toc165947135)

[FR1 9](#_Toc165947136)

[FR2 12](#_Toc165947137)

[FR2a 14](#_Toc165947138)

[FR2b 16](#_Toc165947139)

[FR3 17](#_Toc165947140)

[FR4 19](#_Toc165947141)

[FR5 20](#_Toc165947142)

[FR6 21](#_Toc165947143)

[FR6a 22](#_Toc165947144)

[FR6b 24](#_Toc165947145)

[FR7 27](#_Toc165947146)

[FR8 28](#_Toc165947147)

[FR9 29](#_Toc165947148)

[FR10 30](#_Toc165947149)

[Evaluation 30](#_Toc165947150)

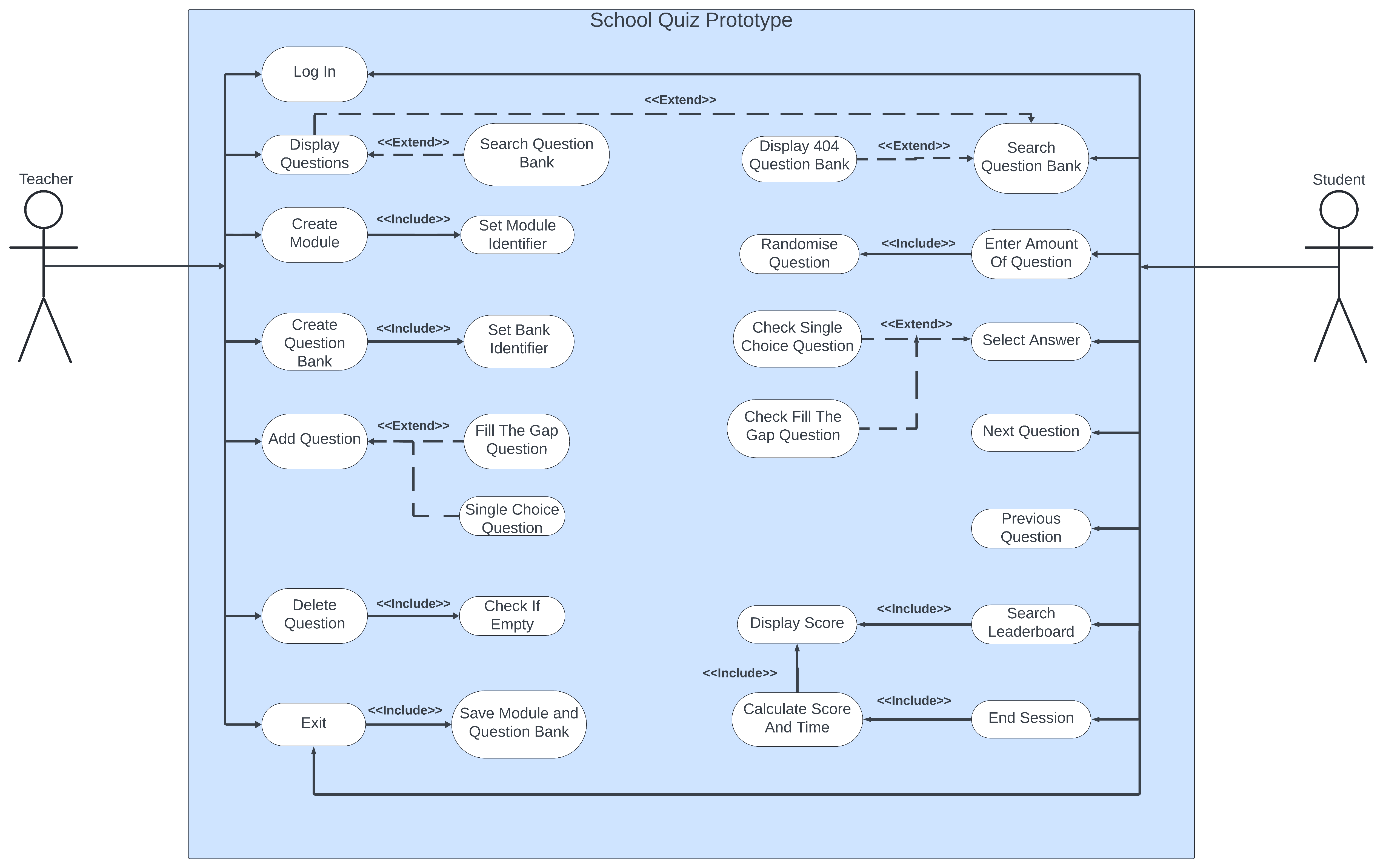
# Introduction

## Analysing The Problem

For my assignment I was briefed, to create a prototype quiz program for teachers to create sets of questions banks which are associated with a specific module. The program will be interacted through a text-based menu user interface, designed for teachers and students to navigate the program. Teachers should be able to remove questions banks and modules. Students can partake a quiz from a question bank created by the teacher. After finishing the quiz, the student will be displayed their score. Questions will be saved and loaded from files to allow quizzes to be saved and use later. Other features will be implemented such as randomise question and navigating questions.

## Use Case Diagram

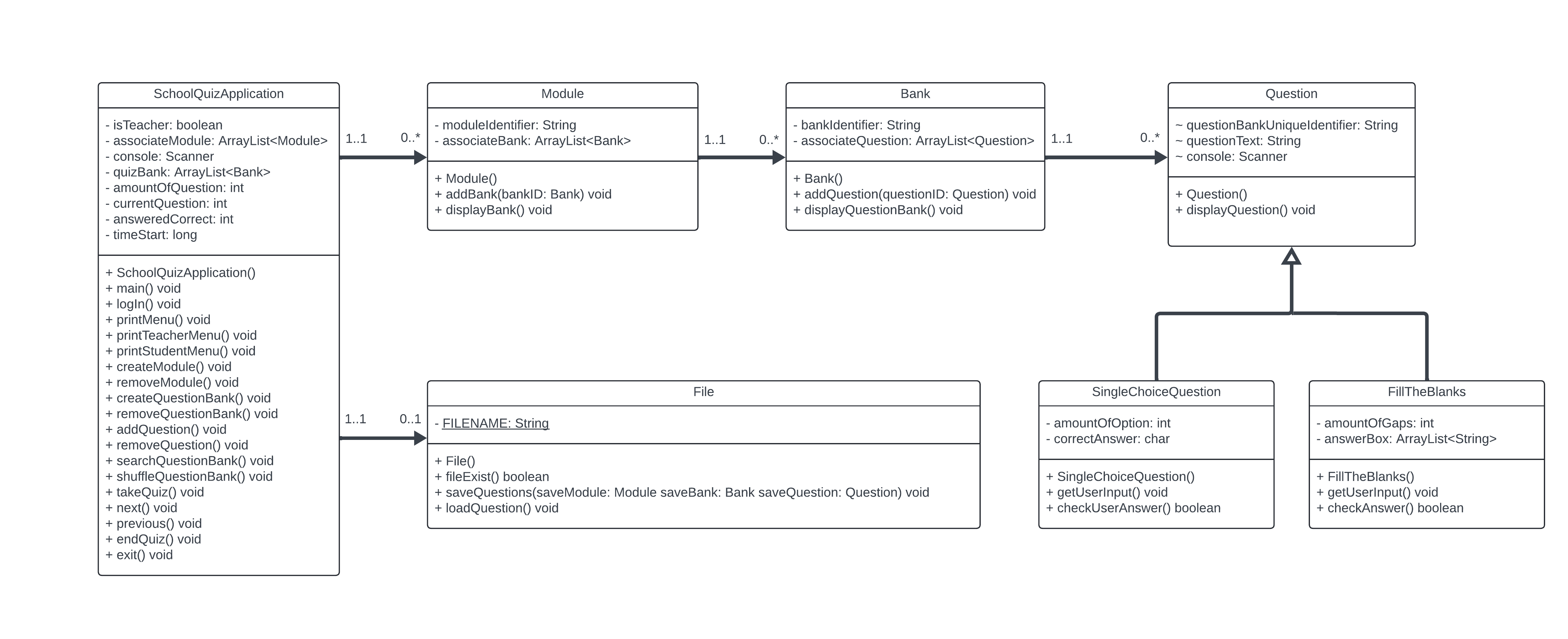
For my case diagram there will be two actors. A teacher and a student where the teacher can create the quiz and the student can partake the quiz. There will be one system which is the school quiz protype.



# Design

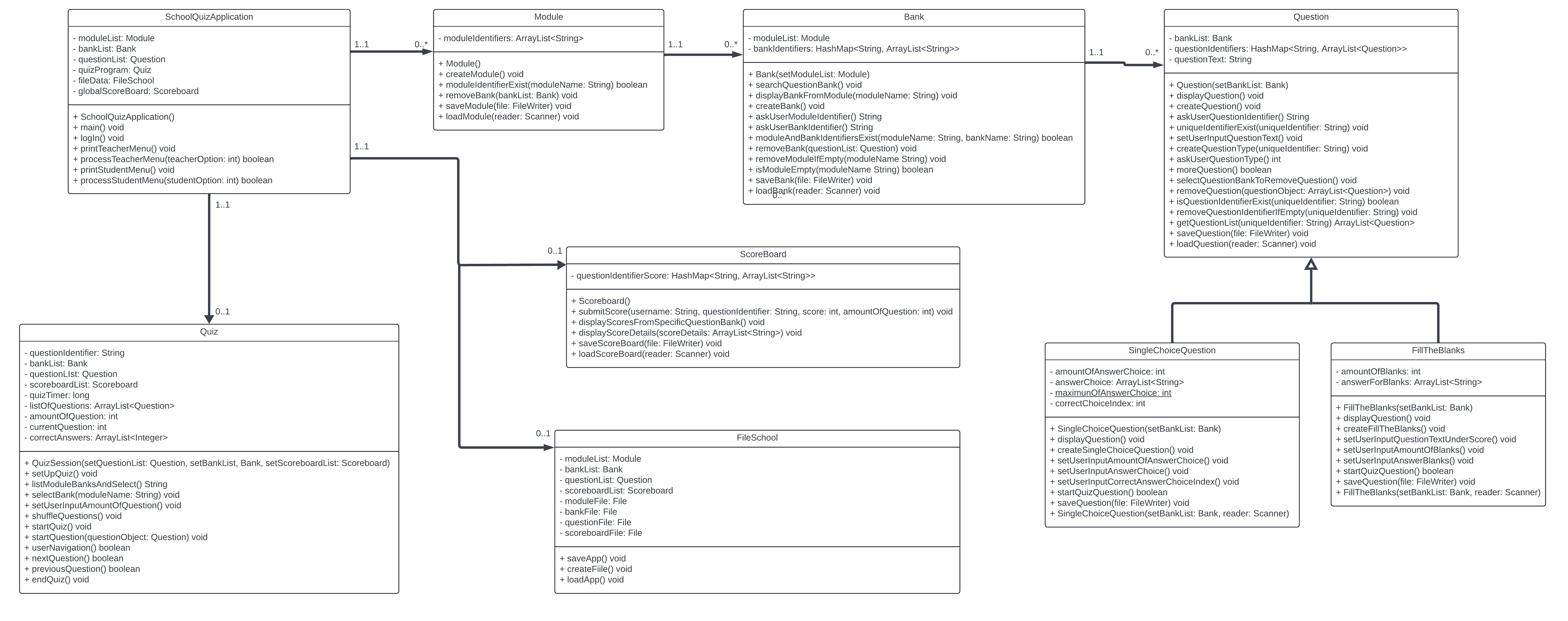
## Class Diagram

### Initial Class Diagram



Before I started creating my program, I’ve made a class diagram to give an insight of how my program will be structured and how my program will work. The code that I will be creating will follow this structure, as you can see above is my class diagram before coding and below is my final class diagram.

### Final Class Diagram



### SchoolQuizApplication

SchoolQuizApplication() - Is the constructor which instantiate all class to be ready and loads   
 module, bank question and scoreboard object with save files.

Login() – Asks the user to either login as a student or teacher.

printTeacherMenu() – Displays the teacher menu.

processTeacherMenu() – Processes teacher menu input.

printStudentMenu() – Displays the student menu

processStudentMenu() – Processes student menu input.

### Module

Module() – Constructor to instantiate the module class.

createModule() – Allows the user to create a module identifier.

moduleIdentifierExist() – Checks if the module exist.

removeModule() – Removes user selected module.

saveModule() – Saves the modules from moduleIdentifier to the module.txt file.

loadModule() – Loads modules from module.txt file to moduleIdentifier.

### Bank

Bank() – Constructor to instantiate the bank class.

searchQuestionBank() – Displays specific question banks.

displayBankFromModule() – Displays question banks from given module argument.

CreateBank() – User inputted bank links to user inputted module.

askUserModuleIdentifier() – User input module and must be valid.

askUserBankIdentifier() – User input bank and must be valid.

moduleAndBankIdentifier() – Checks if user inputted module and bank exist.

removeBank() – Selects a bank to be removed.

removeModuleIfEmpty() – Removes module if no bank is linked to module.

isModuleEmpty() – Returns true if module key exist.

saveBank() – Saves the banks from bankIdentifier to the bank.txt file.

loadBank() – Loads banks from bank.txt file to bankIdentifier.

### Question

Question() – Constructor to instantiate question class.

displayQuestion() – Displays question details.

createQuestion() – Create question.

askUserQuestionIdentifier() – User input question identifier.

addUniqueidentifierIfNew() - Creates new instance of question if new.

setUserInputQuestionText() – User input as the question text.

askUserQuestionType() – User select question type.

createQuestionType() – Create question type.

moreQuestion() – Checks if the user wants to add more question.

selectQuestionBankToRemoveQuesiton() – User select bank before removing question.

removeQuestion() – Removes the question from bank.

IsQuestionIdentifierExist() – Checks if question exist.

removeQuestionIdentifierIfEmpty() – Remove question if empty.

getQuestionList() – Return list of question.

saveQuestion() – Saves the questions from questionIdentifier to the question.txt file.

loadBank() – Loads questions from question.txt file to questionIdentifier.

### SingleChoiceQuestion

SingleChoiceQuestion - Constructor to instantiate SingleChoiceQuestion class.

displayQuestion() – Display SingleChoiceQuestion format.

createSingleChoiceQuestion() – Create single choice question.

setUserInputAmountOfAnswerChoice – User input the amount of answer options.

setUserInputAnswerChoice – User input answer options.

setUserINputCorrectAnswerChoiceIndex() – Store selected correct answer option.

startQuizSession() – Starts the single choice question.

saveQuestion() – Saves the single choice question to the question.txt file.

loadQuestion() – Loads single choice question from question.txt file.

### FillTheBlanks

FillTheBlanks() - Constructor to instantiate FillTheBlanks class.

displayQuestion() – Display FillTheBlanks format.

createFillTheBlanks() – Create fill the blanks question.

setUserInputQuestionTextUnderScore() – User input question text.

setUserInputAmountOfBlanks() – Checks the amount of blanks in question text.

setUserInputAnswerBlanks() – User input answer for each blank.

startQuizSession() – Starts the fill the blanks question.

saveQuestion() – Saves the fill the blanks question to the question.txt file.

loadQuestion() – Loads fill the blanks question from question.txt file.

### Quiz

Quiz() – Constructor to instantiate Quiz class.

setUpQuiz() – Set up quiz.

listModuleBanksAndSelect() – Displays banks user can select from.

SelectBank() – User selects banks.

setUserInputAmountOfQuestion() – User sets the amount of question.

shuffleQuestion() – Shuffle question order.

startQuiz() – User partake the quiz session.

StartQuestion() – Start question to the user.

userNavigation() – User can navigate between questions.

nextQuestion() – Moves to next question.

previousQuestion() – Moves to previous question.

endQuiz() – Displays user performance in quiz.

### Scoreboard

Scoreboard() - Constructor to instantiate Scoreboard class.

submitScore() – Submit the user score to scoreboard.

displayScoresFromAQuestionBank() – Display score of question bank.

displayScoreDetails() – Displays users performance in question bank.

saveScoreboard() – Saves the scoreboard to the scoreboard.txt file.

loadScoreboard() – Loads fill the scoreboard from scoreboard.txt file.

### FileSchool

FileSchool() - Constructor to instantiate FileSchool class.

saveApp() – Saves module, bank, question and scoreboard.

createFile() - Create file if they don’t exist.

loadApp() - load module, bank, question and scoreboard.

## Pseudo Code Example

My most complicated algorithm is when it comes to loading up questions tied to specific question identifier. Function is named loadQuestion().

A screenshot of a computer program

Description automatically generated

I found this complicated because different question type derived from Question class however my question object has a HashMap of question identifies as key which holds a list of questions as value.  
This led me having to figure out how can I instantiate a question type child class and pass it to the parent object and how will it figure out which question type as well.

# Testing

## Test Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| A1.1 | FR1 | User select Add Module | Enter 2 | Adding module method. | P | User must create a module before creating a bank. |
| A1.2 |  | User creates a module | Enter “Shrek” | Accept module identifier size 7 or less | P |  |
| A1.3 |  | User select Add Question Bank | Enter 3 | Adding a bank identifier. | P | Once module is created then bank can now link to it |
| A1.4 |  | User creates a bank | Enter  “Donkey” | Accept bank identifier size 15 or less | P |  |
| A1.5 |  | User must link valid bank to module | Enter “Shrek” | Accept “Shrek” since it exist | P | Bank must link to an existing module. |
| A1.6 |  | Display all question bank from a module | Enter “Shrek” | Should display all question bank from Shrek module | P |  |

## FR1

#### A1.1

A computer screen shot of a program

Description automatically generated

#### A1.2

A screenshot of a computer program

Description automatically generated

#### A1.3

A screenshot of a computer program

Description automatically generated

#### A1.4

A screen shot of a computer

Description automatically generated

#### A1.5

A screenshot of a computer

Description automatically generated

#### A1.6

A screenshot of a computer program

Description automatically generated

## FR2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| B1.1 | FR2 | User select Add Question | Enter 4 | Adding Question method should start | P |  |
| B1.2 |  | User enter a question identifier | Enter “Shrek”  “Donkey” | Bank “Donkey” exist and it links to “Shrek” module so it should be valid. | P |  |
| B1.3 |  | User enter question type | Enter 1 | User will be asked to create question text for single choice question | P |  |
| B1.4 |  |  | Enter 2 | User will be asked to create question text for fill the blanks question | P |  |

#### B1.1

A screenshot of a computer program

Description automatically generated

#### B1.2

A computer screen shot of white text

Description automatically generated

#### B1.3

A screenshot of a computer program

Description automatically generated

#### B1.4

A computer screen shot of a black screen

Description automatically generated

## FR2a

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| C1.1 | FR2a | User enter question text | Enter  “Capital of  China” | User question text will be stored in question. | P |  |
| C1.2 |  | User enter amount of question choice | Enter 3 | User amount of choice gets saved | P |  |
| C1.3 |  | User enter answer choice to amount | Enter “Beijing”  “Paris”  “London” | User set question choice to amount of question | P |  |
| C1.4 |  | User enter which question choice is correct | Enter  1 | User input number will be subtracted by one and be stored as correct answer index | P |  |

#### C1.1

A screen shot of a computer

Description automatically generated

#### C1.2

A screenshot of a computer

Description automatically generated

#### C1.3

A computer screen shot of a black screen with white text

Description automatically generated

#### C1.4

A screenshot of a computer

Description automatically generated

## FR2b

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| D1.1 | Fra2b | User enter question text with blanks | Enter “My name is \_\_\_” | User input special question text will be saved and blanks will be calculated | P |  |
| D1.2 |  | User enter answers for all the answer blanks | Enter  “Charlie  Cheung” | Each individual user input will be saved for each blank | P |  |

#### D1.1 and D1.2

A computer screen with white text

Description automatically generated

## FR3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| E1.1 | FR3 | User select Remove Question | Enter  7 | Remove Question method will be removed | P |  |
| E1.2 |  | User enter question identifier | Enter  “Shrek”  “Donkey | Bank exist and links to module then displays questions from question identifier | P |  |
| E1.3 |  | User select question to be removed | Enter  2 | Removes question 3 | P |  |

#### E1.1

A screenshot of a computer program

Description automatically generated

#### E1.2

A screenshot of a computer

Description automatically generated

#### E1.3

A screenshot of a computer program

Description automatically generated

## FR4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| F1.1 | FR4 | User select Search Question Bank | Enter  1 | Displays question bank from “Shrek” module | P |  |
| F1.2 |  | User enter module identifier | Enter “Shrek” | Displays question bank from “Shrek” module | P |  |

#### F1.1

A screenshot of a computer program

Description automatically generated

#### F1.2

A screenshot of a computer

Description automatically generated

## FR5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| G1.1 | FR5 | User select Remove Question Bank | Enter  6 | Asks the user to enter question identifier to delete | P |  |
| G1.2 |  | User enter module and bank | Enter  “Shrek”  “Donkey” | Removes Shrek:Donkey | P |  |

#### G1.1

A screenshot of a computer program

Description automatically generated

#### G1.2

A screen shot of a computer

Description automatically generated

## FR6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| H1.1 | FR6 | User select Change User | Enter  8 | Display user login | P |  |
| H1.2 |  | User select student user type | Enter  “S” | Displays student menu | P |  |

#### H1.1

A screenshot of a computer program

Description automatically generated

#### H1.2

A screenshot of a computer program

Description automatically generated

## FR6a

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| I1.1 | FR6a | User select Search Question Bank | Enter  1 | Askes the user to enter module | P |  |
| I1.2 |  | User enter existing module identifier | Enter  “Shrek” | Displays banks associated with module | P |  |
| I1.3 |  | User enter non-existing module identifier | Enter  “NotShrek” | Displays module does not exist | P |  |

#### I1.1

A screenshot of a computer program

Description automatically generated

#### I1.2

A screenshot of a computer program

Description automatically generated

#### I1.3

A screenshot of a computer program

Description automatically generated

## FR6b

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| J1.1 | FR6b | User select Start Quiz | Enter 2 | Asks the user to enter module | P |  |
| J1.2 |  | User enters module and bank identifiers | Enter  “Shrek”  “Donkey” | Ask the user how much question to partake | P |  |
| J1.3 |  | User enter number within amount range | Enter  1 | Display 1 question | P |  |
| J1.4 |  | User enter number more than amount of question | Enter  4 | Display all question available | P |  |

#### J1.1

A screenshot of a computer program

Description automatically generated

#### J1.2

A screenshot of a computer

Description automatically generated

#### J1.3

A screenshot of a computer

Description automatically generated

#### J1.3

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

## FR7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| K1.1 | FR7 | User select to End Quiz | Enter  3  Charung | Display time took, score and unanswered question | P |  |

#### K1.1

A screenshot of a computer

Description automatically generated

## FR8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| M1.1 | FR8 | Question automatically gets reordered |  | Displays the new question order | P |  |
| M1.2 |  | Displays one question at a time |  |  | P |  |

#### M1.1 and M1.2

A screenshot of a computer

Description automatically generated

## FR9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| N1.1 | FR9 | User select Next Question | Enter  1 | Moves to next question | P |  |
| N1.2 |  | User select Previous Question | Enter  2 | Moves to previous question | P |  |

#### N1.1 and N1.2

A screenshot of a computer

Description automatically generated

## FR10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| O1.1 | Fr10 | Display single choice question format |  |  | P |  |
| O1.2 |  | Display fill the blanks format |  |  |  |  |

#### O1.1 and O1.2

#### A screenshot of a computer Description automatically generated

# Evaluation