School Quiz Program

Charlie Cheung

Submission Date: xx/xx/xxxx

Ccc15@aber.ac.uk

Table of Contents

[Introduction 2](#_Toc165908876)

[Analysing The Problem 2](#_Toc165908877)

[Use Case Diagram 2](#_Toc165908878)

[Design 3](#_Toc165908879)

[Class Diagram 3](#_Toc165908880)

[Initial Class Diagram 3](#_Toc165908881)

[Final Class Diagram 4](#_Toc165908882)

[SchoolQuizApplication 4](#_Toc165908883)

[Module 4](#_Toc165908884)

[Bank 5](#_Toc165908885)

[Question 5](#_Toc165908886)

[SingleChoiceQuestion 6](#_Toc165908887)

[FillTheBlanks 6](#_Toc165908888)

[Quiz 6](#_Toc165908889)

[Scoreboard 7](#_Toc165908890)

[FileSchool 7](#_Toc165908891)

[Pseudo Code Example 8](#_Toc165908892)

[Testing 8](#_Toc165908893)

[Test Table 8](#_Toc165908894)

[FR1 9](#_Toc165908895)

[A1.1 9](#_Toc165908896)

[A1.2 9](#_Toc165908897)

[A1.3 10](#_Toc165908898)

[A1.4 10](#_Toc165908899)

[Evaluation 10](#_Toc165908900)

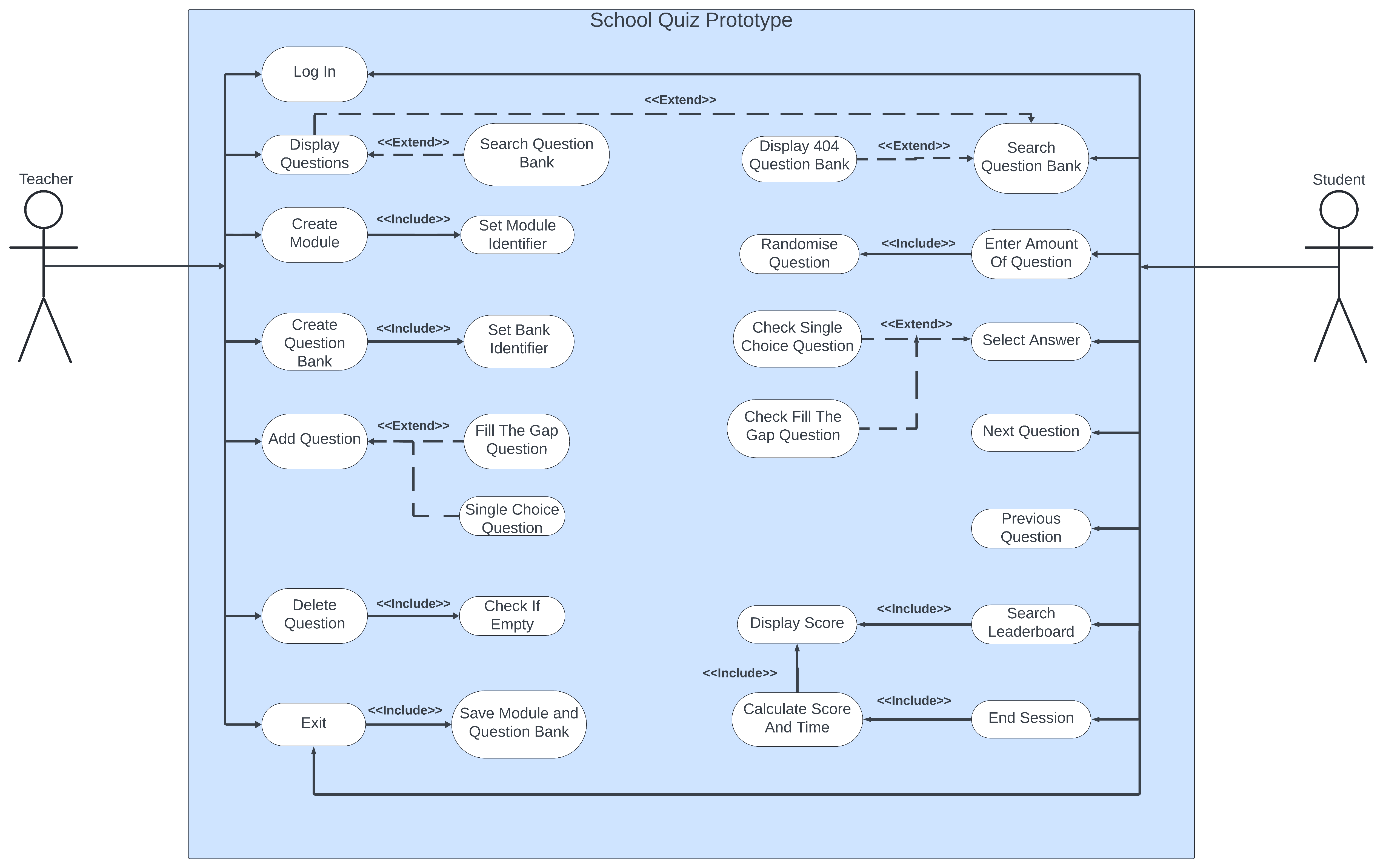
# Introduction

## 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, questions such as a picking the correct answer from a set of potential answers and fill in the blanks of a sentence. Which are designed to evaluate students’ knowledge of a 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 edit/remove questions banks and modules. Students can partake a quiz from a question bank by searching a specific module identifier, created by the teacher. After finishing the quiz or quitting at any point, the student will be displayed a score of how well they performed. Questions will be saved and loaded from files to allow quizzes to be saved and use later when rebooting the quiz program. Other features will be implemented such as questions will be given at random and students can move back and forth of questions they are doing in a quiz.

## Use Case Diagram

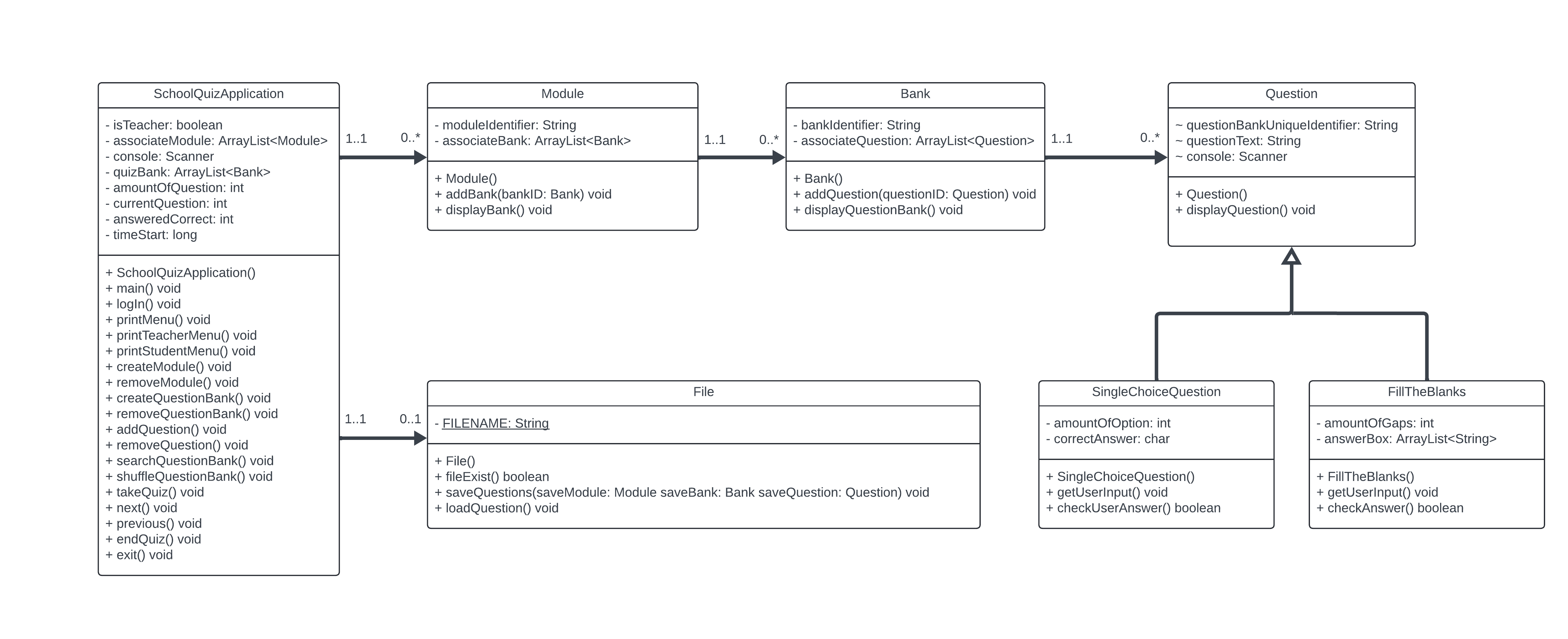
For my case diagram there will be two actors. A teacher and a student where the teacher can create the quiz and the student can partake the quiz. There will be one system which is the school quiz protype.



# Design

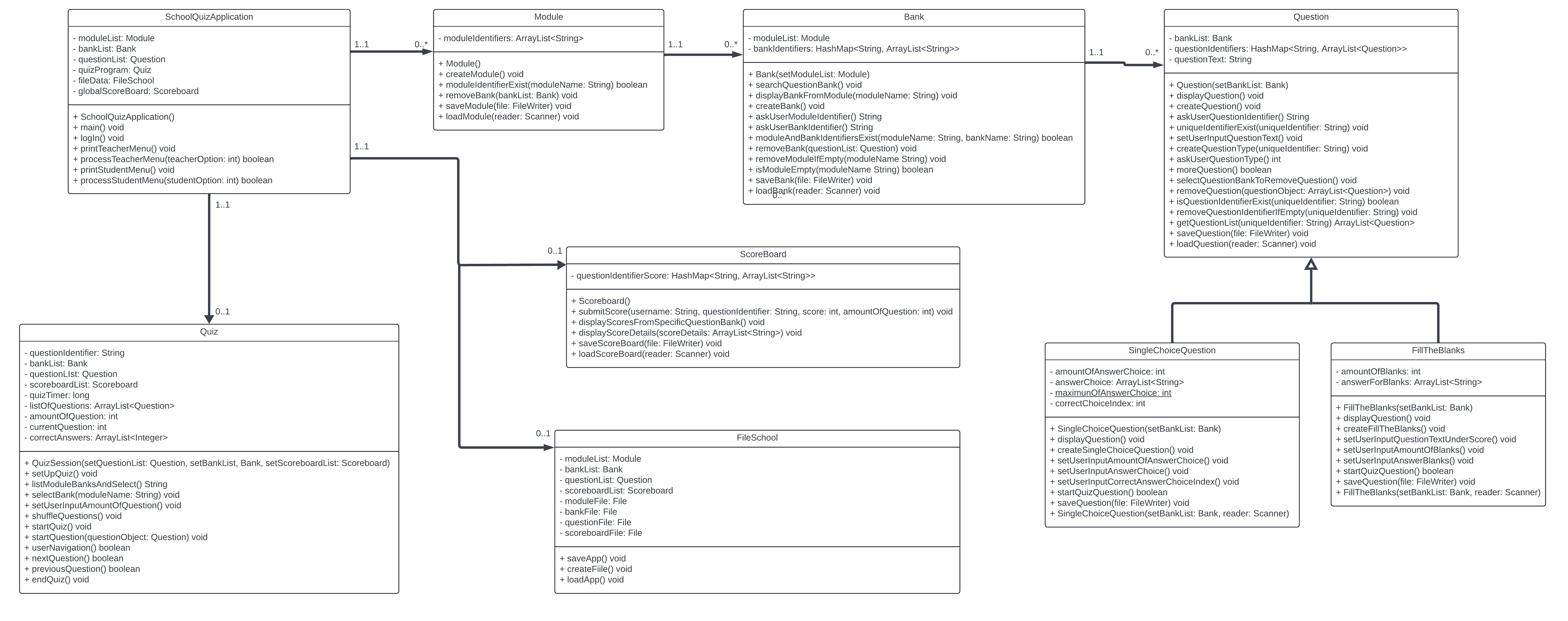
## Class Diagram

### 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 to the user and awaits for user input.

processTeacherMenu() – Processes the user input form teacher menu corresponding task..

printStudentMenu() – Displays the student menu to the user and awaits for user input.

processStudentMenu() – Processes the user input form student menu corresponding task..

### Module

Module() – Constructor to instantiate the module class.

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

moduleIdentifierExist() – Checks if the module identifier exist in moduleIdentifier ArrayList.

removeModule() – Removes user selected module identifier in moduleIdentifier.

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

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

### Bank

Bank() – Constructor to instantiate the bank class.

searchQuestionBank() – Displays specific question banks and returns user inputted module identifier.

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

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

askUserModuleIdentifier() – Asks for user input module identifier and must be valid before returning.

askUserBankIdentifier() – Asks for user input bank identifier and must be valid before returning.

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

removeBank() – User selects a bank to be removed from bankIdentifier.

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

isModuleEmpty() – Returns true if module key exist within bankIdentifier.

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

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

### Question

Question() – Constructor to instantiate question class.

displayQuestion() – Method for child class to override and be used to redirect to child method.

createQuestion() – Methods to allow the user to add and create question to questionIdentifier.

askUserQuestionIdentifier() – Asks user for question identifier and returns it.

addUniqueidentifierIfNew() - Adds question identifier to HashMap if it’s new.

setUserInputQuestionText() – User input Text to act as the question text.

askUserQuestionType() – Askes the user what question type is the next question to be created.

createQuestionType() – User creates a question of a specific type (Single Choice or Fill The Blanks).

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

selectQuestionBankToRemoveQuesiton() – User select bank and to remove a question from it.

removeQuestion() – Removes the question from bank.

IsQuestionIdentifierExist() – Checks if question identifier exist within questionIdentifier.

removeQuestionIdentifierIfEmpty() – Remove question identifier from questionIdentifier if empty.

getQuestionList() – Returns an ArrayList of different question type from a specific question identififer.

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

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

### SingleChoiceQuestion

SingleChoiceQuestion - Constructor to instantiate SingleChoiceQuestion class.

displayQuestion() – Overrides parent method and display SingleChoiceQuestion format.

createSingleChoiceQuestion() – Allows the user to create single choice question.

setUserInputAmountOfAnswerChoice – User input the amount of answer options there is.

setUserInputAnswerChoice – User input answer options.

setUserINputCorrectAnswerChoiceIndex() – Store user selected answer option index position.

startQuizSession() – Starts the single choice question and allows the user to partake quiz.

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() – Overrides parent method and display FillTheBlanks format.

createFillTheBlanks() – Allows the user to create fill the blanks question.

setUserInputQuestionTextUnderScore() – User input question text in fill the blanks format.

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

setUserInputAnswerBlanks() – User input answer for each blank.

startQuizSession() – Starts the fill the blanks question and allows the user to partake quiz.

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 before user can partake quiz session.

listModuleBanksAndSelect() – Displays Banks user can select from.

SelectBank() – User selects banks to partake question quiz from.

setUserInputAmountOfQuestion() – User sets the amount of question they want to partake.

shuffleQuestion() – Shuffle the orders of question displayed to user.

startQuiz() – User partake the quiz session.

StartQuestion() – Start question to the user.

userNavigation() – User can navigate questions or exit program.

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 from quiz to scoreboard.

displayScoresFromAQuestionBank() – User select question bank to display scoreboard.

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. Not to mention having to format my question save file in a way for my code to digest and load into the question types.

# Testing

## Test Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirements | Description | Input | Expected Output | Pass/Fail | Comments |
| A1.1 | FR1 | User select Add Module | Enter 2 | Adding module method should start | P | User must create a module before creating a bank else bank |
| A1.2 |  | User creates a module | Enter “Shrek” | Module identifier should be accepted because its length is equals to or less than 7 | P |  |
| A1.3 |  |  | Enter  “Pinocchio” | Module identifier should not be accepted because it exceeds 7 character | P |  |
| A1.4 |  | User select Add Question Bank | Enter 3 | Adding a bank identifier to a module method should start | P | Once module is created then bank can now link to it |
| A1.5 |  | User creates a bank | Enter  “Donkey” | Bank identifier should be accepted because it’s length is equals to ore less than 15 | P |  |
| A1.6 |  |  | Enter “ShrekAnd  -Donkey-  Adventure” | Bank identifier should no accepted because it exceeds 15 character | P |  |
| A1.7 |  | User must link valid bank to module | Enter “Shrek” | Since “Shrek” module exist so bank will link to Shrek module | P | Bank must link to an existing module. |
| A1.8 |  |  | Enter “NotShrek” | Since “NotShrek”  Module does not exist so bank will not link to module | P |  |
| A1.9 |  | Display all question bank from a module | Enter “Shrek” | Should display all question bank from Shrek module | P |  |
| 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 |  |  | Enter “NotShrek”  “Donkey” | Bank exist but does not link to that module | P |  |
| B1.4 |  |  | Enter  “Shrek”  “Catfish” | Bank does not exist but module does exist | P |  |
|  | FR2a |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## 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

Description automatically generated

### A1.4

A screenshot of a computer program

Description automatically generated

### A1.5

A screen shot of a computer

Description automatically generated

### A1.6

A computer screen shot of white text

Description automatically generated

### A1.7

A screenshot of a computer

Description automatically generated

### A1.8

A screen shot of a computer

Description automatically generated

### A1.9

A screenshot of a computer program

Description automatically generated

## FR2

### 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 screen shot of a computer

Description automatically generated

### B1.4

A screenshot of a computer program

Description automatically generated

# Evaluation