****

***Department of Engineering***

***School of Informatics & Engineering***

***Institute of Technology Blanchardstown***

***Dublin 15***

**Software Design & Quality**

**7th of October 2016**

# Table of Contents

[Table of Contents i](#_Toc469567862)

[Table of Figures i](#_Toc469567863)

[No table of figures entries found. i](#_Toc469567864)

[Introduction 1](#_Toc469567865)

[Problem Statement 1](#_Toc469567866)

[NOUN ANALYSIS 1](#_Toc469567867)

[Software Analysis 2](#_Toc469567868)

[Fully Dressed Use Case Model 1 2](#_Toc469567869)

[Fully Dressed Use Case Model 2 3](#_Toc469567870)

[Conceptual Model 4](#_Toc469567871)

[UML Class Conceptual Model 4](#_Toc469567872)

[Modelling Assumptions 4](#_Toc469567873)

[Behaviour Analysis 5](#_Toc469567874)

[High-level sequence diagram 5](#_Toc469567875)

[High-level Activity diagram 6](#_Toc469567876)

[High-level State diagram 7](#_Toc469567877)

[Software Design 8](#_Toc469567878)

[Detailed Sequence Diagrams 8](#_Toc469567879)

[Refined Class Diagram 11](#_Toc469567880)

[Implementation Recommendations & Notes 11](#_Toc469567881)

# Table of Figures

# No table of figures entries found.

# Introduction

In this task we are given a problem statement. In the statement given, it explains “to design the software for a multiple Choice Quiz application to aid students taking the module “Software Design & Quality”. We are asked to design and show a simple use case analysis and a conceptual model for this problem.

# Problem Statement

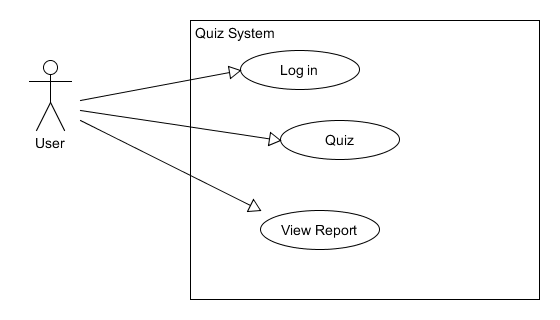
You are required to develop a C++ Multiple Choice Quiz application to aid students taking the module “Software Design & Quality” understand agile software development methodologies (specifically Kanban & SCRUM) and hence study for their January exams. Users will be presented with a randomly chosen set of 10 questions from a question bank and be presented with a choice of solutions. The user must then choose a correct answer and move to the next question. The application will generate a report of how well the student did after the quiz has completed. The student’s result is saved to their profile. The application can manage many student profiles and can generate an overall class report. The quiz may be administrated by an admin user. An admin user has full access to all student’s attempts, but a student user can only see their own attempts as such Student’s must login to their profile before starting the quiz. Students are allowed have multiple attempts which are all persistently saved against their individual profile. Questions and solutions are saved in an XML file which is read by the quiz application. You will need to research Agile developmental lifecycles to define your questions and you will need to investigate how to parse XML using C++

# Noun analysis

You are required to develop a C++ Multiple Choice Quiz application to aid students taking the module “Software Design & Quality” understand agile software development methodologies (specifically Kanban & SCRUM) and hence study for their January exams. Users will be presented with a randomly chosen set of 10 questions from a question bank and be presented with a choice of solutions. The user must then choose a correct answer and move to the next question. The application will generate a report of how well the student did after the quiz has completed. The student’s result is saved to their profile. The application can manage many student profiles and can generate an overall class report. The quiz may be administrated by an admin user. An admin user has full access to all student’s attempts, but a student user can only see their own attempts as such Student’s must login to their profile before starting the quiz. Students are allowed have multiple attempts which are all persistently saved against their individual profile. Questions and solutions are saved in an XML file which is read by the quiz application. You will need to research Agile developmental lifecycles to define your questions and you will need to investigate how to parse XML using C++

# Software Analysis

### Fully Dressed Use Case Model 1



#### Fully Dressed Description

#### Primary actor: student.

**Goal in context**:

**Level**: User level

**Stakeholders and Interests**:

User: wants to take quiz and see resulting report

Quiz: Shows quiz and generates student result in report.

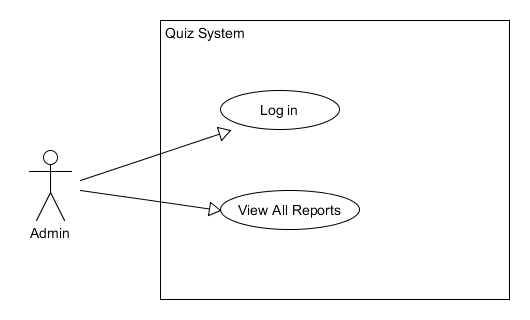
**Precondition**: Quiz is ready.

**Minimum Guarantee**: User sees quiz questions.

**Success Guarantees**: user has taken quiz and weather station is updating value to display new temperature.

**Trigger**: Student logs in.

### Fully Dressed Use Case Model 2



#### Fully Dressed Description

**Primary Actor:** Admin.

**Goal in context**: Admin logs in views overall class report.

**Level:** Admin Level.

**Stake Holders and Interests:** Admin wants to see all student results and the attempts on quiz.

**Preconditions:** The system is up and running with data from the students doing the quiz.

**Minimum Guarantee**: Admin sees all student’s results in report.

**Success Guarantees:** Admin access the quiz system’s overall class report.

**Trigger:** Admin Logs in to Quiz system

**Main success scenario:**

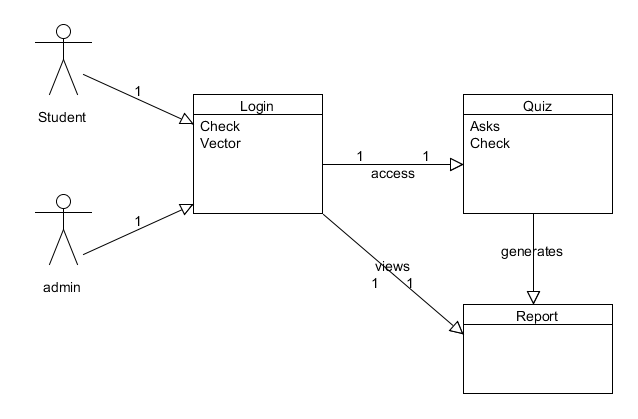
1. Admin logs in.
2. A report of all student’s attempts is displayed.
3. Admin hits any key.
4. System then logs the Admin out.

**Extensions:**

1a. Admin’s login details entered is wrong, and is asked for them again.

## Conceptual Model

### UML Class Conceptual Model



### Modelling Assumptions

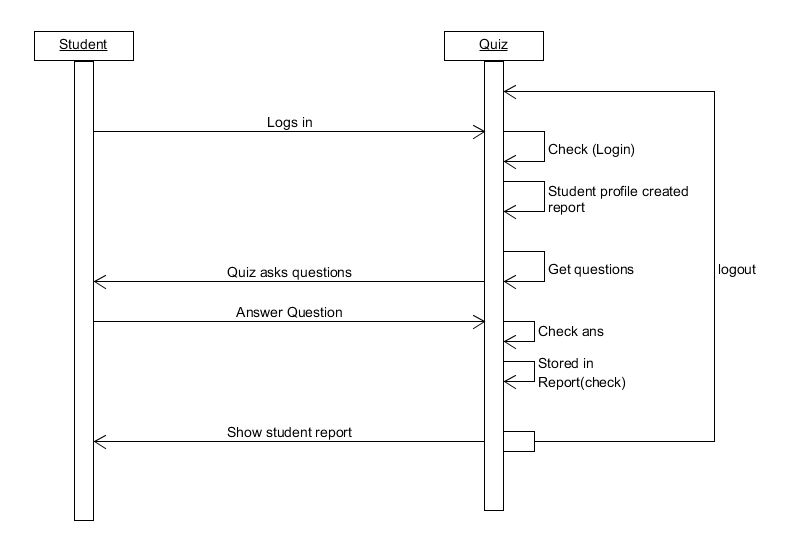
* Quiz class takes in questions + answers from the XML file.
* Only one student at a time using the system.
* Only one report per student after quiz.
* One report is added to the overall reports.

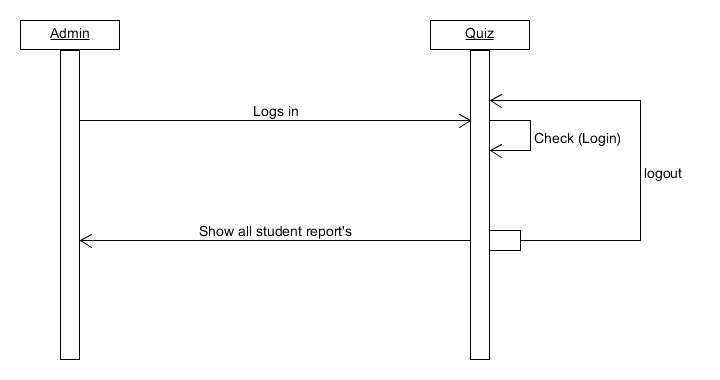
**System Design Observations and Recommendations**

* One student can only view their own report.

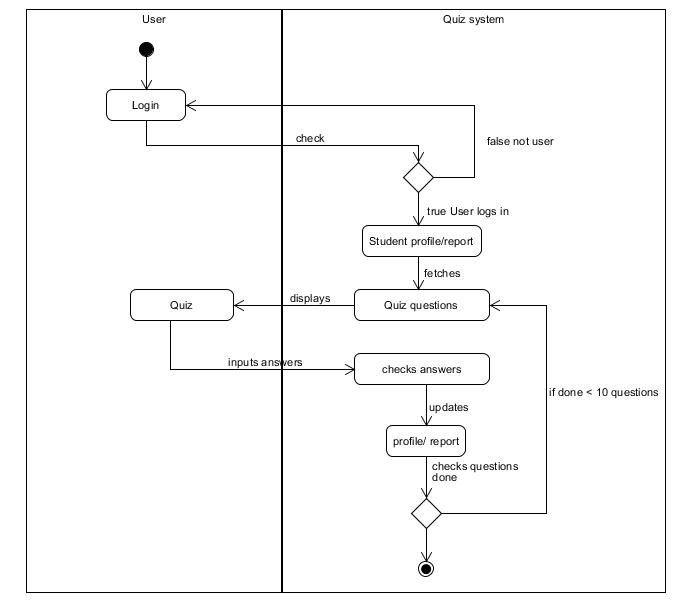
## Behaviour Analysis

### High-level sequence diagram



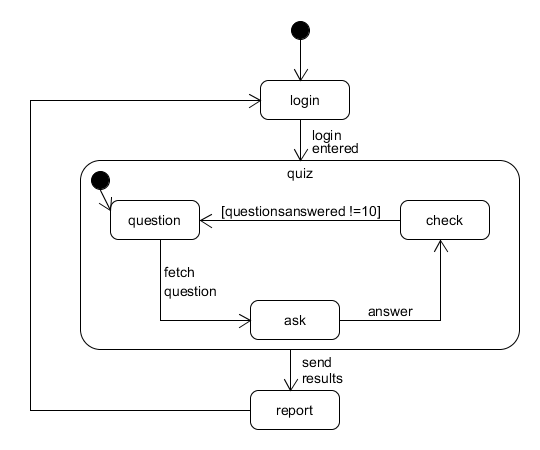


### High-level Activity diagram

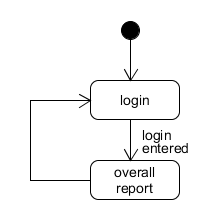


### High-level State diagram

User state Diagram.



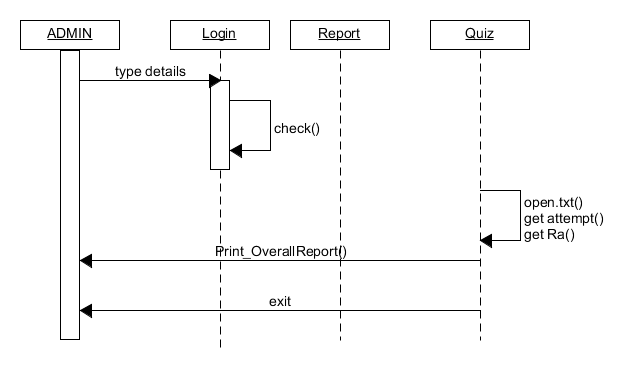
Admin State



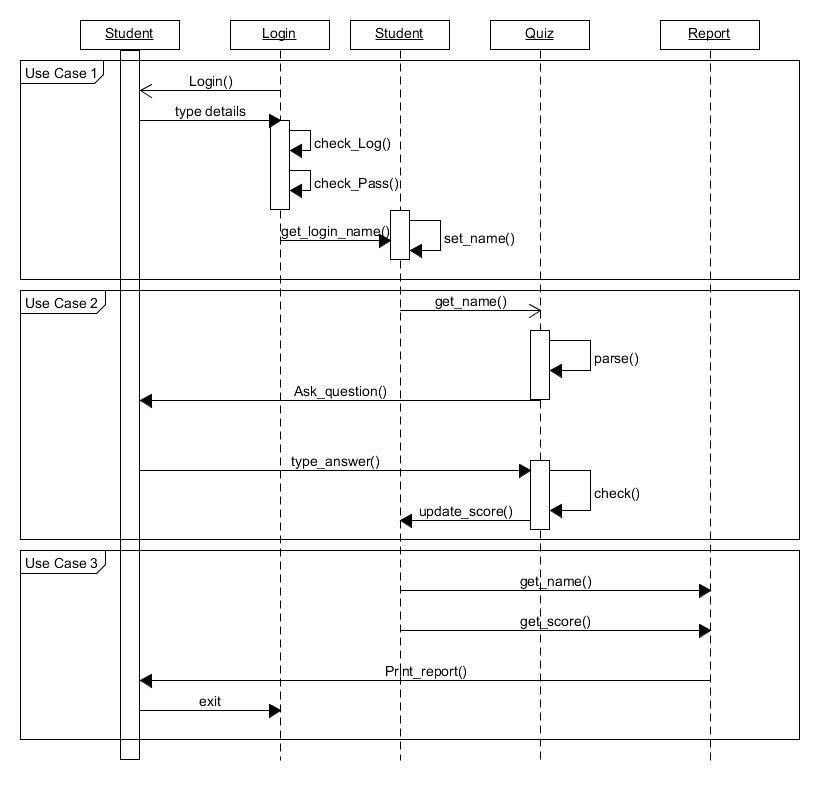
# Software Design

### Detailed Sequence Diagrams

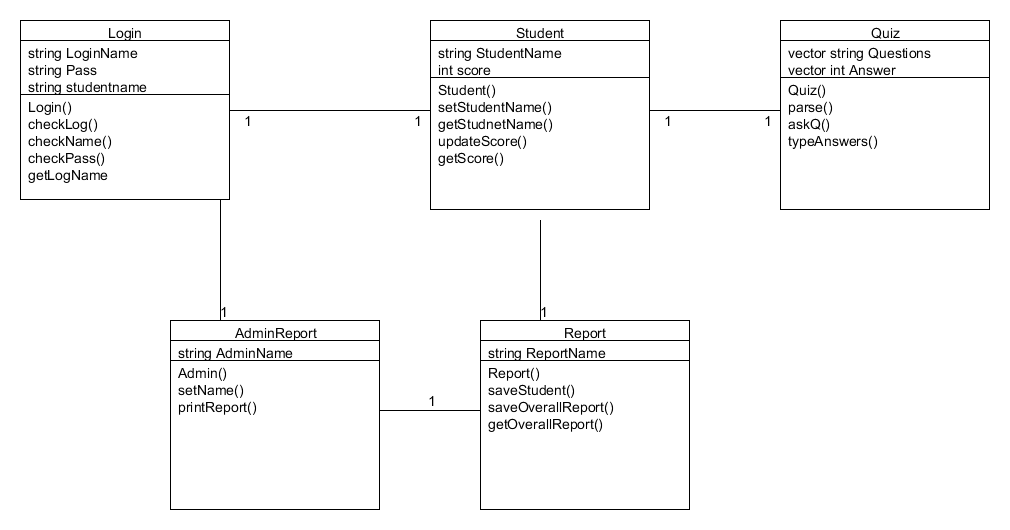
#### Sequence 1



**Sequence 2**



### Refined Class Diagram



### OCP Implementation Recommendations & Notes

The relationship outlined in the refined class diagram show that classes have one to one relationship. The Login class can be used by any user (i.e. a student or an admin) it is open for extension, like adding a new user, but is closed for modification. The quiz is only accessible by the student class but cannot changed anything from it nor the admin can change anything from it, it acts like an interface for the student and admin to use the methods.

An object of a class instantiated in main is always passed into another instant of another class. in this way classes work on an object that is passed to it and can update the data members of that object and return the object in main to be passed to another class.