Cairo University Faculty of Computers and Information



**CS352 – Software Engineering II**

**Phase 1 Template**

**2017**

NEFHAM

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20140031 | Ahmed Alaa Hussein | [Ahmedalaahussein00@gmail.com](mailto:Ahmedalaahussein00@gmail.com) | 01117730087 |
| 20140097 | Hazem mohamed Taha | [Hazem.fouda@yahoo.com](mailto:Hazem.fouda@yahoo.com) | 01211122871 |
| 20140199 | Ghada Gamal Ramadan | [sheblghada@gmail.com](mailto:sheblghada@gmail.com) | 01152191535 |
| 20140205 | Karim Ashraf Mohamed | [Karimallam\_96@outlook.com](mailto:Karimallam_96@outlook.com) | 01100896557 |
| 20140293 | Noura Mohi El-Dein | [Nouramohi712@gmail.com](mailto:Nouramohi712@gmail.com) | 01005016518 |

**Staff:**

**DrAmr Kamel** [a.kamel@fci-cu.edu.eg](mailto:a.kamel@fci-cu.edu.eg)

**DrKhadiga Mohamed kelbedweihy@fci-cu.edu.eg**

**Ragia Mohamed**

**TAs: Eng Mohamed Samir m.samir@fci-cu.edu.egEng Omar Khaled Ali Ragab o.khaled@fci-cu.edu.egEngRagia Mohamed r.mohamed@fci-cu.edu.eg**

**EngEbtehalyahia ebtehal.yahia@fci-cu.edu.eg**

**Eng Ahmed Emad ahmed.emad@fci-cu.edu.eg**

**Eng Amr Tarek a.tarek@fci.cu.edu.eg**

Contents

[Review Check List 3](#_Toc476413281)

[Testing 5](#_Toc476413282)

[Git repository link 6](#_Toc476413283)

# 

# Review Check List

**Design Principles**

1. Does the design follow SOLID principles? ∏ 80 %

Related Issues:

S:In class Game both createGame()& playGame() are together despite they have more than one concept.

O:It’s handled correctly as adding extension to code is easy.

L:Inhereted classes Teacher & Student are of the same type which is User.

I :fat interfaces exist in class Registration a functions are duplicates and can becompressedintoone function.

D:Dependency occurs between RegistrationController & GameController.

Issue #1

1. Does the design follow OOP rules? ∏ 95%

Related Issues: Classes are classified according to methods correctly , Setters & Getters exist , Inheritance relation between User with Teacher& Student correctly but withoutabstract methods.

Issue #2

1. Is the design simple and easy to modify? ∏ 100%

Related Issues: Totally easy to understand & modify.

Issue #3

**Coding Standards**

1. Is the code understandable and readable? ∏ 80%

Related Issues: Magic numbers exist without brief explanation to its usage in Class gameplayGame() . in Class Registration in createStudentAccount() , createTeacherAccount() ,loginAsStudent() , loginAsTeacher() , in Class registrationController in validateUserdata().

Issue #4

1. Does the code follow Java Coding Style? ∏ 50 %

Related Issues :Most of classes are not written by java coding style.

Issue #5

1. Is indentation used properly? ∏ 50 %

Related Issues: Spacing between statements are not handled properly.

Issue #6

1. Do variable have good names? ∏ 90 %

Related Issues: In reading from file , the used variable is always String s which doesn’t refer to its content.

Issue#7

**Comments**

1. Is the code commented enough? ∏ 0 %

Related Issues:None even one comment for explanation.

1. Is every class and method commented? ∏ 0 %

Related Issues: None.

1. Do comments follow Javadoc style? ∏ 0 %

Related Issues:None.

1. Is Javadoc generated for all the code? ∏ 0 %

Related Issues:None.

1. Are there useless / wrong comments? ∏ 0 %

Related Issues:None.

**Code Structure**

1. Does the code follow the design precisely? ∏ 50 %

Related Issues: Many functions are not implemented on classes but found on Class Diagram Model as :

Logout() in Registration , joinCompetition() & MakeComment() in Student , respondComments() in Teacher , getScore() is not found on Student , updateGame() & editGame() in Game.

Issue #8

1. Are there very long classes or methods? ∏ 70 %

Related Issues: Many methods are long as they are duplicates in Registration in .createStudentAccount() , createTeacherAccount() ,loginAsStudent() , loginAsTeacher().

Issue #9

1. Is there repeated code ?(put in a function) ∏ 70 %

Related Issues: LogInAsTeacher() , LogInAsStudent() , CreateStudentAcc () , CreateTeacherAcc() the 4 functions in Registration class , all contain the same implementation as they read from file , they can be replaced by file extension as a parameter and all be in only one function.

Issue #10

**Error Handling**

1. Does the code handle errors and exceptions? ∏ 30 %

Related Issues: Most of the code don’t handle catches after reading from file.

Issue #11

1. Is defensive programming used to avoid errors? ∏ 30 %

Related Issues: Poor check of file existence and ability to write on it & rarely closing Scanner.

Issue #12

**Logic**

1. Do loops have correct conditions and bounds? ∏ 80 %

Related Issues: loops are accurate while if statements are unnecessary in ValidateUserData() in RegistrationController & no break in others.

If statements are long in class RegistrationController as it takes many parameters, instead he could take object of User as a parameter.

Issue #13

1. Do loops always terminate? ∏ 100 %

Related Issues:Always.

Issue #14

**Overall**

1. **Are the design and code of good quality?** ∏ 90 %

Related Issues: Runs perfectly with GUI.

Issue #15

# Testing

1. **RegisterControllerTestClass**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** |
| **1.** | **validateUserData**  **(result , name , mail,**  **pass ,age , teacher)** | **Testing function for Signup function in RegisterationController entity. This test case tests the normal SignUp scenarios**  **Assumption: A filed cannot be left empty.** | **Failed**  **In all of the test cases.**  **Because:**  **Age: cannot be negative.**  **Email: Wrong format.** |
| **2.** | ***testEmailAndPassword***  **( mail , pass , teacher);** | **Testing function for login function. This test case tests the normal login scenario** | **Failed**  **In first test case because the password wrong and passed.**  **Passed**  **In the second and third test cases.** |

**2- GameControllerTestClass**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** |
| **1.** | ***validateGameData***  **(cat , name , techname , date , type , num , game)** | **Testing function for Create game function in GameController entity. This test case tests the normal add game sequence**  **Assumption: A filed cannot be left empty.** | **Passed in**  **the first and the second test cases.**  **Failed in the third test case because number of question cannot be negative.** |

**3.GameTest Class**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** |
| **1.** | ***playGame(name)*** | **Testing function for play game function in Game entity. This test case tests the normal play game sequence.** | **Passed in**  **All test cases.** |

# Git repository link

**https://github.com/AhmedAlaa20140031/NEFHAM-Testing**