Cairo University Faculty of Computers and Information



**CS352 – Software Engineering II**

**Phase 1**

**2017**

**LEARNO**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20140088 | Aya Mohammed Sayed Gad | ayoyagad96@gmail.com | 01140132399 |
| 20140121 | Radwa Ali Ahmed Mohammed | radwafci2014@gmail.com | 01006224168 |
| 20140139 | Sara Mahmoud Mohamed | saramahmoud251@gmail.com | 01276010818 |
| 20140068 | Amany Ashraf Abd El-Tawab | emy55emy66@gmail.com | 01142773139 |

**Staff:**

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

**Dr Khadiga Mohamed kelbedweihy@fci-cu.edu.eg**

**[T.A Omar Khaled]**

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

**Eng Ebtehal yahia 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

[Instructions [To be removed] 3](#_Toc476413280)

[Review Check List 3](#_Toc476413281)

[Testing 5](#_Toc476413282)

[Git repository link 6](#_Toc476413283)

# Review Check List

# Design and Code Checklist

**Design Principles**

1. Does the design follow SOLID principles? ∏ 30%

Related Issues:

Not Solid: The class doesn’t have one specific job to do.

No Interface segregation: the user is forced to implement interfaces that are not needed.

1. Does the design follow OOP rules? ∏ 50%

Related Issues:

Long Block of codes, High level of complexity and many redundant blocks of code.

1. Is the design simple and easy to modify? ∏ 70 %

Related Issues:

Input restrictions are ignored and the design is fragile ,hard to reuse

are not needed.

**Coding Standards**

1. Is the code understandable and readable? ∏ 60%

Related Issues:

complex code ,variable names are not clear and there exist some unused variables.

are not needed.

1. Does the code follow Java Coding Style? ∏ 40 % Related Issues: long block of codes
2. Is indentation used properly? ∏ 50% Related Issues: Bad formatting.
3. Do variable have good names? ∏ 70% Related Issues: non declarative names.

**Comments**

1. Is the code commented enough? ∏ 100% Related Issues: no comment was mentioned
2. Is every class and method commented? ∏ 100% Related Issues: no comment Do comments follow Javadoc style? ∏ 100% Related Issues: no comment Is Javadoc generated for all the code? ∏ 100% Related Issues: no comment Are there useless / wrong comments? ∏ 100% Related Issues: no comment

**Code Structure**

1. Does the code follow the design precisely? ∏ 50% Issues: No appropriate design
2. Are there very long classes or methods? ∏ 95% Related Issues: All the code is implemented in a single class.
3. Is there repeated code ?(put put in a function) ∏ 90%

Related Issues: Code Redundancy

**Error Handling**

1. Does the code handle errors and exceptions? ∏ 80 %

Related Issues: input restrictions are ignored

1. Is defensive programming used to avoid errors? ∏ 100%

Related Issues: Not implemented.

**Logic**

1. Do loops have correct conditions and bounds? ∏ 100%
2. Do loops always terminate? ∏ 100%

**Overall**

1. **Are the design and code of good quality?** ∏ 70%

Related Issues:

Many concepts are ignored and the code faces a lot of problems

And doesn’t seem easy to be used by Users..

# 

# Testing

* **In the following table, you should describe each testing functions you developed in each testing class and state the result of testing after executing testing class**
* **Number test cases in proper way.**
* **Example**
  1. **UserTesting class**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** |
| **1.** | **login(boolean result,String name,int pass,String type)** | **Testing function for login function in**  **controller entity. This test case test the normal login scenario**  **Assumption:**  **Username and passwords match one of the accounts already created before.** | **Passed** |
| **2.** | **login(boolean result,String name,int pass,String type)** | **Testing function for login function in**  **controller entity.**  **This test case test the normal login scenario**  **Assumption:**  **Username and passwords does NOT match any of the accounts already created** | **Failed** |
| **3.** | **addComment(boolean result , String gamename,String type)** | **Testing function for add\_comment function in**  **controller Entity. This test case test the normal scenario of adding comment**  **Assumption:**  **Game name exists/correct.** | **Passed** |
| **4.** | **addComment(boolean result , String gamename,String type)** | **Testing function for add\_comment function in**  **controller Entity.**  **This test case test the scenario of adding comment on a game that doesn’t exist/incorrect.**  **Assumption:**  **Game name doesn’t exists/incorrect.** | **Failed** |
| **5.** | **sign\_up(boolean result,String name ,String username , int password ,String mail , String state)** | **Testing function for sign\_up function in**  **controller entity.**  **This test case test the normal sign up scenario**  **Assumption:**  **Type entered is either a student or teacher.** | **Passed** |
| **6.** | **sign\_up(boolean result,String name ,String username , int password ,String mail , String state)password, mail, state)** | **Testing function for sign\_up function in**  **Controller Entity. This test case test the sign up scenario Assumption:**  **Type entered is incorrect.** | **Failed** |
| **7.** | **Rate( gname,type)** | **Testing function for rate function in**  **Controller Entity.**  **This test case tests the normal scenario of rating a game.**  **Assumption:**  **Game name exists / type exists** | **Passed** |
| **8.** | **Rate( gname,type)** | **Testing function for rate function in**  **Controller Entity.**  **This test case tests the scenario of rating a game.**  **Assumption:**  **Input range is negative or out of range.** | **Failed** |

# 

# Git repository link

<https://github.com/AmanyEldaly/Software-Project>