Cairo University Faculty of Computers and Information



**CS352 – Software Engineering II**

**Phase 1 Template**

**2017**

**Project Team**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20140090 | Aya Mahmoud mouhamed | Ayaelhawary14@gmail.com |  |
| 20140300 | Heba | Heba\_ahmed\_eng@yahoo.com |  |
| 20140299 | neveen | Nevenreda34@gmail.com |  |
| 20140208 | ledia |  |  |
|  |  |  |  |
|  |  |  |  |

**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**

**[Write here your TA name only in your lab]**

**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**

Title page has no header or footer.

Docs must be supplied as pdf –files re-named as requested

Contents

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

[Review Check List 3](#_Toc476413281)

[Testing 5](#_Toc476413282)

[Git repository link 6](#_Toc476413283)

# 

# Instructions [To be removed]

* **IMPORTANT. Rename this document to CS352-LeaderID-TANAME-Phase2-a.docx**

**[Write TA name in your document name]**

**Examples:**

**CS352-20120001-MohamedSamir-Phase1.docx**

* **Remove the following notes and any red notes**

# Review Check List

* **Use the provided checklist as a starting point for your review. Add more items as issues arise during the review.**

**Design and Code Checklist**

**Design Principles**

1. Does the design follow SOLID principles? yes What %60 Related Issues:page5
2. Does the design follow OOP rules? yes What %70 Related Issues: page5
3. Is the design simple and easy to modify? yes What % 60 Related Issues:page6

**Coding Standards**

1. Is the code understandable and readable? ∏ What % 95 Related Issues: …….…
2. Does the code follow Java Coding Style? ∏ What % 95 Related Issues: …….…
3. Is indentation used properly? ∏ What % 98 Related Issues: …….…
4. Do variable have good names? ∏ What % 92 Related Issues: …….…

5-Coding Style:

Naming functions:-

Name did not begin with a verb. for example, function Registered login in class Database.

Name did not describe the purpose of the function . for example ,function setBoolTrue() in class Registered and function setCID() in class Game.

6-Indentation:

There is not a space after placing a comma between two function arguments. for example in function register(String name,String acc,String country,String email,String school,String password) , Registered login(String email,String p).

7-variables names :

They use shortcuts in some variables naming for example, variable name db of type database in a class Interface, variable name g of type game in class Database, function name parameter called acc in function register().

**Comments**

1. Is the code commented enough? no What % 0 Related Issues NO comments found …….…
2. Is every class and method commented? no What % 0 Related Issues: No comments for any class or method. It must find at least one line comment to explain what is the method do or what is the objective of this class or this method…….…
3. Do comments follow Javadoc style? no What %0 Related Issues: No using comments to follow any style…….…
4. Is Javadoc generated for all the code? no What %0 Related Issues: : No, we did not find any using for Javadoc because Javadoc is a tool which comes with JDK and it is used for generating Java code documentation in HTML format from Java source code, which requires documentation in a predefined format and we did not find that.…….…
5. Are there useless / wrong comments? yes What % …… Related Issues yes there is some sort of code put into comment, this cod useless …….…

**Code Structure**

1. Does the code follow the design precisely? ∏ What % 80 Related Issues: …….…
2. Are there very long classes or methods? ∏ What % 0 Related Issues: …….…
3. Is there repeated code ?(put put in a function) ∏ What % 5 Related Issues: …….…

13- Does the code follow the design precisely?

In class diagram there exists classes such as ( User, Guest, Problem, Student, Teacher)not used in sequence diagram and in code implementation.

In sequence diagram :-

1-AddGame use case there exist setPath() function that not implemented in class Game.

2-playGame use case there exist reqCategory() ,SelectGameCategory() which did not implemented in Interface ,Database classes respectively.

15-Is there repeated code ?(put put in a function)?

In class Database the function CreateGame() do nothing except calling function AddGame in the same class .

**Error Handling**

1. Does the code handle errors and exceptions? ∏ What % …… Related Issues: Related Issues: not all errors can handle like in add Game when need to enter number of question if user enter string the software will stop without show any message for user and handled in register when user enter password wrong or don’t match with the email and this code did not use exceptions statements like "try…catch"
2. Is defensive programming used to avoid errors? no What % …… Related Issues: No, because defensive code is check all input parameters for the correct type, length, and range of values, when user register if enter large string in name or any field the software don’t do any action(length),and according to correct type if you enter string rather than integer the software will stop. Defensive code making sure to check input and output fields and return values, but in this software if you fill field in register the program/software will stop. Defensive code review and improve error handling, in this software the program can't continue with any unexpected action.

**Logic**

1. Do loops have correct conditions and bounds? yes What % …… Related Issues: …….
2. Do loops always terminate? yes

**Overall**

1. **Are the design and code of good quality?** no What % 50

1-Does the design follow SOLID distribution?

1.1 single responsiblitity : this principle not exist because There is some class has more than one reason to change like:

1.1.1 Data base class will chang if:

1.1.1.1 DB\_interface change

1.1..1.2 reportpb Formates change

(SRP violate)

2-Open Closed principle:open for extention & closed for modification

2.1 this principle exist because user deals with DB class through the interface class,so if we need to change any class other class still as is.

3-Liskov substitution principle:this principle exist because subtypes are substitutable for thier base types as " User " inherit from it "registered" &"guest".

4-Interface segregation principle:this principle not exist because

the design has no segregation on its interface ,it is fat.if we want to extend inetrface

we are forced to implement the full interface and write some dumy method.

4.1 -Interface class has a huge amount of functionality,different types of functionality

must be placed in different interfaces.

5-Dependancy Inversion principle:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2-Does the design follow oop rule? yes

2.1 reuse ,they used inheritance.

2.2 object model optmization.

2.3 interface specification :not exist because they can't describe precisly each class interface.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

3-Is the design simple and easy to modify?

no,because if we change one class other class will change.ex,in class DB:

reportpb()

creategame()

replay()

getuser()

addgame()

all of these functions need attributes from another class.

# 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.** | **register(String name ,String acc,String country,String email,String school,String password)** | **Testing function for register as teacher .**  **This test case test the normal register as a teacher** | **Passed1** |
| **2.** | **register(String name ,String acc,String country,String email,String school,String password)** | **Testing function for register as teacher .**  **This test case test register without academic account** | **Failed** |
| **3.** | **register(String name ,String acc,String country,String email,String school,String password)** | **Testing function for register as teacher .**  **This test case test register with repeated email**  **the expected output should be this email already used.**  **The actual output success created account** | **Failed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** |
| **1.** | **login(String email ,String password)** | **Testing function for login as student.**  **This test case test the normal case** | **Passed** |
| **2.** | **login(String name ,password)** | **Testing function for login as teacher .**  **This test case test login with wrong email** | **passed** |
| **3.** | **login(String name ,password)** | **Testing function for login as teacher .**  **This test case test login with wrong password** | **passed** |

|  |  |  |
| --- | --- | --- |
| **Number** | **Testing function** | **Description** |
| **1.** | **addGame(Registered t)** | **Testing function for addGame function in Database class. This test case test the addGame scenarios**   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Normal  Scenario | Extreme case | Extreme case | Extreme case | | Game Name | **game** | **Integer value** | **nothing** | **nothing** | | Game  Category | **Science** | **History** | **nothing** | **programming** | | GameType | **MCQ** | **T&F** | **MCQ** | **MCQ** | | Questions  number | **3** | **1** | **-5** | **0** | | Result | **passed** | **failed** | **failed** | **failed** | |
| **2.** | **playGame(long id)** | **Testing function for playGame function in Database class. This test case test the playGame scenario**   |  |  | | --- | --- | | id | Result | | Any +ve number | **passed** | | 0 | **failed** | | -6 | **failed** | | String values(ledia) | **failed** | | nothing | **failed** | |

# Git repository link

**You should put here your git repository link**