

**Index**

1. Koshary architecture
2. User story : Add Employee By Manager   
     
   2.1 User story : agile sample  
     
   2.2 User story : inputs  
     
   2.3 User story : business logic  
     
   2.4 User story : outputs  
     
   2.5 User story : UI/UX  
     
   2.6 User story : code sample
3. Problems of koshary architecture

3.1 first problem :  
 no separation of concerns   
  
3.2 second problem :   
the dependency between the business logic and user interface   
  
3.3 third problem :   
the dependency between the business logic and infrastructures

3.4 fourth problem :   
the evil of software : code duplications

**1.Koshary architecture :**

* Koshary architecture is just a joke like spaghetti code but from the architecture perspective which mean there are no actually architecture pattern exist or the legacy project.
* Before studying system design and architecture patterns we need first to know what are the problems that will be in our system when there are no architecture pattern used ?

A plate of food with spoons and bowls of food on a table

Description automatically generated

Koshary is a delicious Egyptian food that contains a mixture of different food elements in single dish.

Koshary architecture pattern is refer to the architecture pattern which single component ( usually the user interface component ) contain the implementation of all other components in the system like :  
business logic ,   
database ,   
notifications   
and .. etc.

A diagram of a business structure

Description automatically generated

**2. User story : Add new employee by manager**  
To better understanding, let’s see that with real user story and actual project ( employee management system ) with code sample :  
  
**2.1 User story : agile sample**

A person in a suit and tie

Description automatically generated

As a manager,

I want to add new employee to the system,

So I can save important information about the employee like   
- personal information : name , birth date   
- contact info : email , phone number  
- address : country , city , street  
- skills

**2.2 User story inputs :**

|  |  |
| --- | --- |
| **Data** | **Data type** |
| Employee Name | string |
| Employee Birthdate | Datetime |
| Email | string |
| Phone number | string |
| Country | string |
| City | string |
| Street and Building Number | string |
| Skills | List |

**2.3 User story business logic :**

**- Logic Workflow :**

|  |  |
| --- | --- |
| **User** | **System** |
| Manager opens screen called “Add new Employee”. | System display “Add new Employee” screen. |
| Manager enters the information of the employee on the UI elements and inputs. | - |
| Manager click on button called “Add new Employee” | System will :   1. if employee information is valid , then system will save the new employee and display success message 2. if employee information is not valid , then system will not save the new employee and display errors message |

* **Business rules:**

1. Employee age should be equal or greater than 21
2. Employee email must has correct format like [test@test.com](mailto:test@test.com) but the system should not accept other incorrect formats like test@test , test\_test.com.
3. Employee phone number is required and must be unique , system can not have two employees with the same phone number.
4. Employee must has at min 1 skill.

**2.4 User story outputs :**

|  |  |
| --- | --- |
| **Case** | **Result** |
| Case 1 :  input by manager apply the business rules | 1. Save new employee on database 2. Send Email Notification to Admin 3. Display success message on the screen for the manager |
| Case 2 :  Input by manager does not apply the business rules | 1. Display errors message on the screen for the manager |

* 1. **User story UI/UX :**
* **Default screen :**

A screenshot of a computer

Description automatically generated

* **Screen with errors message :**

A screenshot of a computer

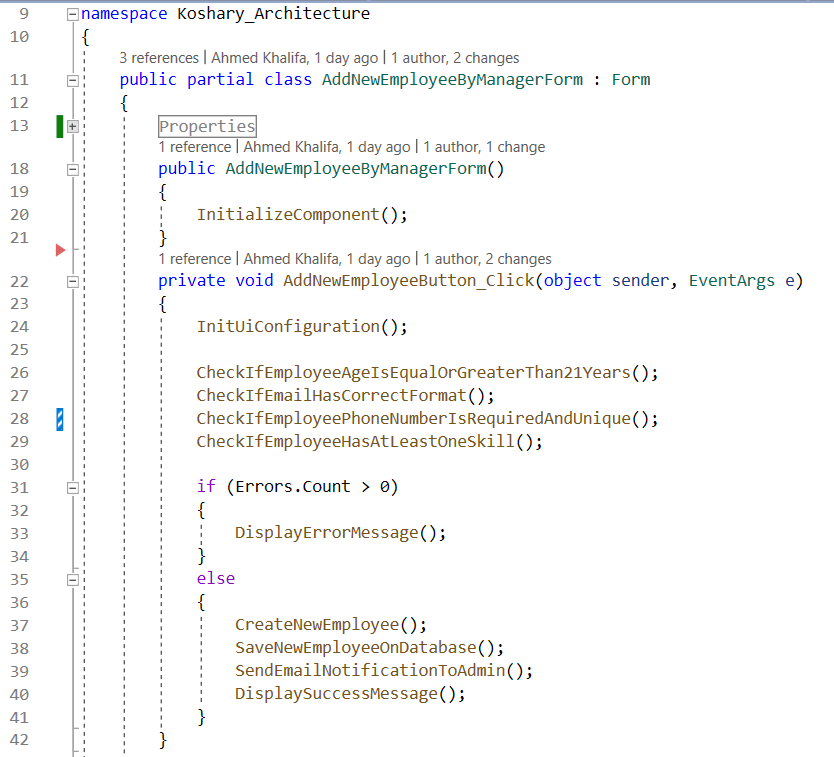
Description automatically generated

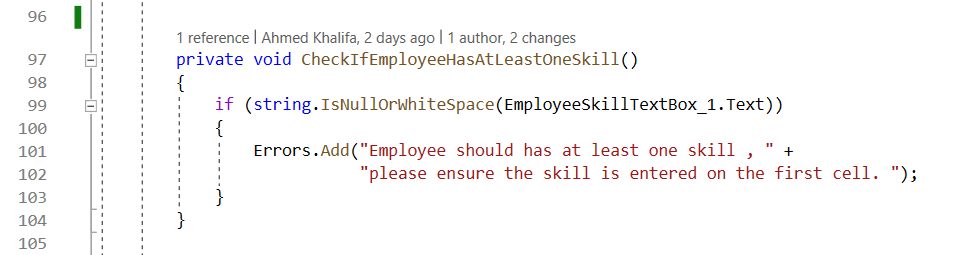
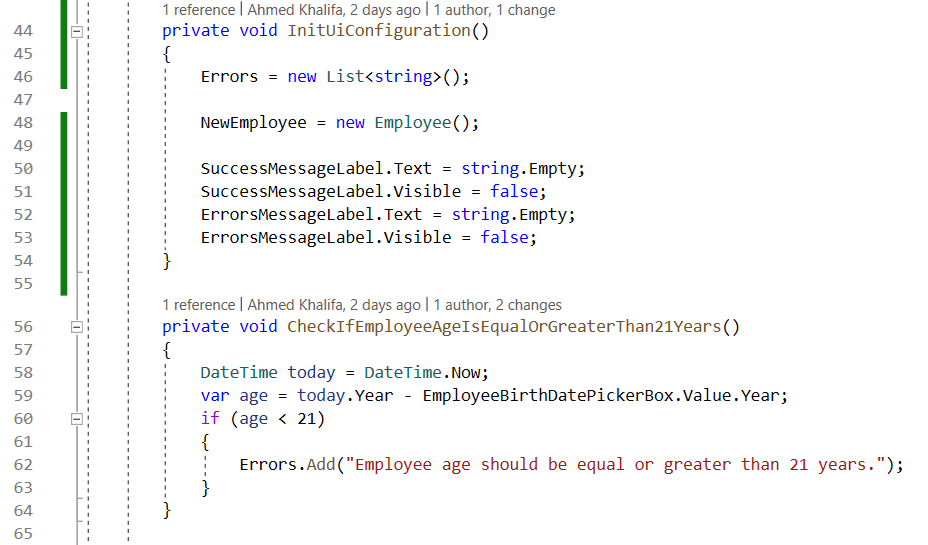
* **Screen with success message :**

A screenshot of a computer

Description automatically generated

* 1. **User story code sample :**

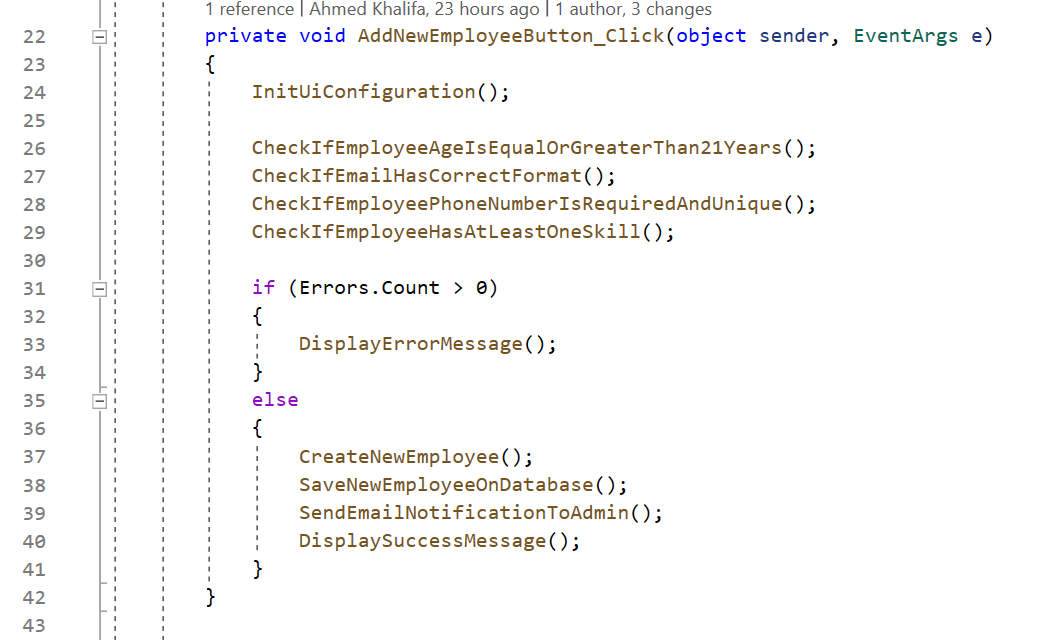
****

****

**3. Problems of koshary architecture :**  
This code may be apply some of the clean code principles like good naming , small functions but from the architecture perspective has a lot of problems.

**First problem ( No Separation Of Concern ) :**

The user interface component : **AddEmployeeByManagerForm** contain **the actual implementation** of all other components like business rules , database and email.



**User interface**

**User interface**

**Infrastructures**

**Business object creation**

**User interface**

**Business Rules**

The current architecture is not layered or has any modules to separate responsibilities from each other , just one layer ( user interface ) contain the implementation of all things.

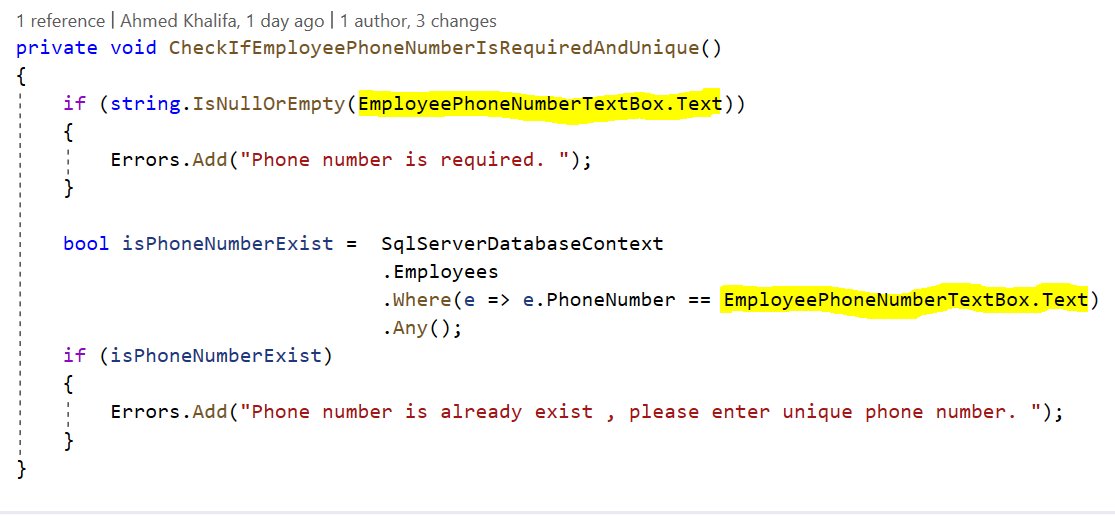
**Now if the company need to change the UI desktop framework form c# windows form that run only in the windows os to other UI desktop frameworks like .NET MAUI which is a cross platform that run on windows , mac , android and ios , this means the developers will not only change the code base of UI elements or framework , but they need to completely refactor the project from scratch.**

**Second problem :**

**business logic is very coupled and depended on the user interface**

Any change on the user interface code will effect on the business logic

For example :



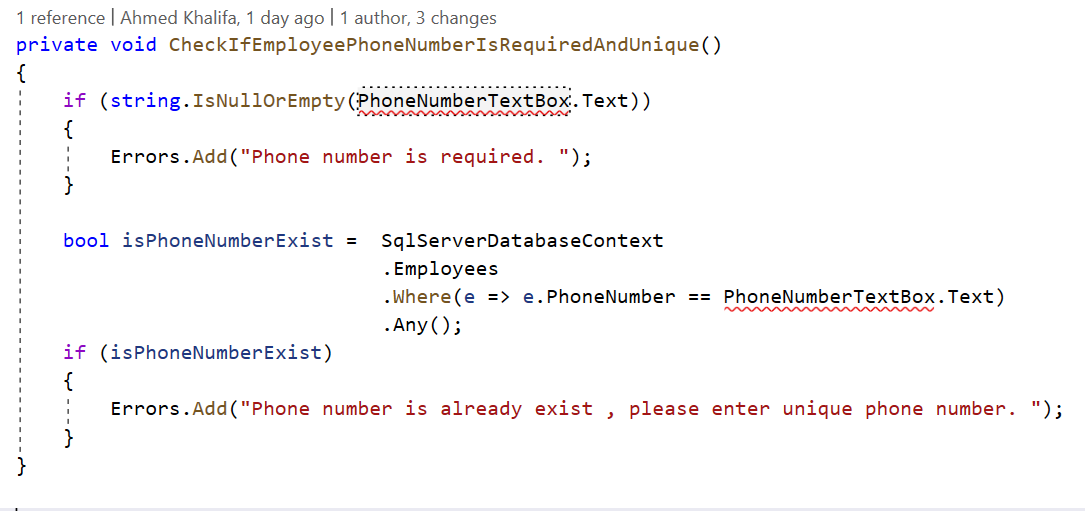
**Business Rule Function**

UI Element Reference

This business function contain reference to the ui element , this cause the business function will be coupled on the ui element.

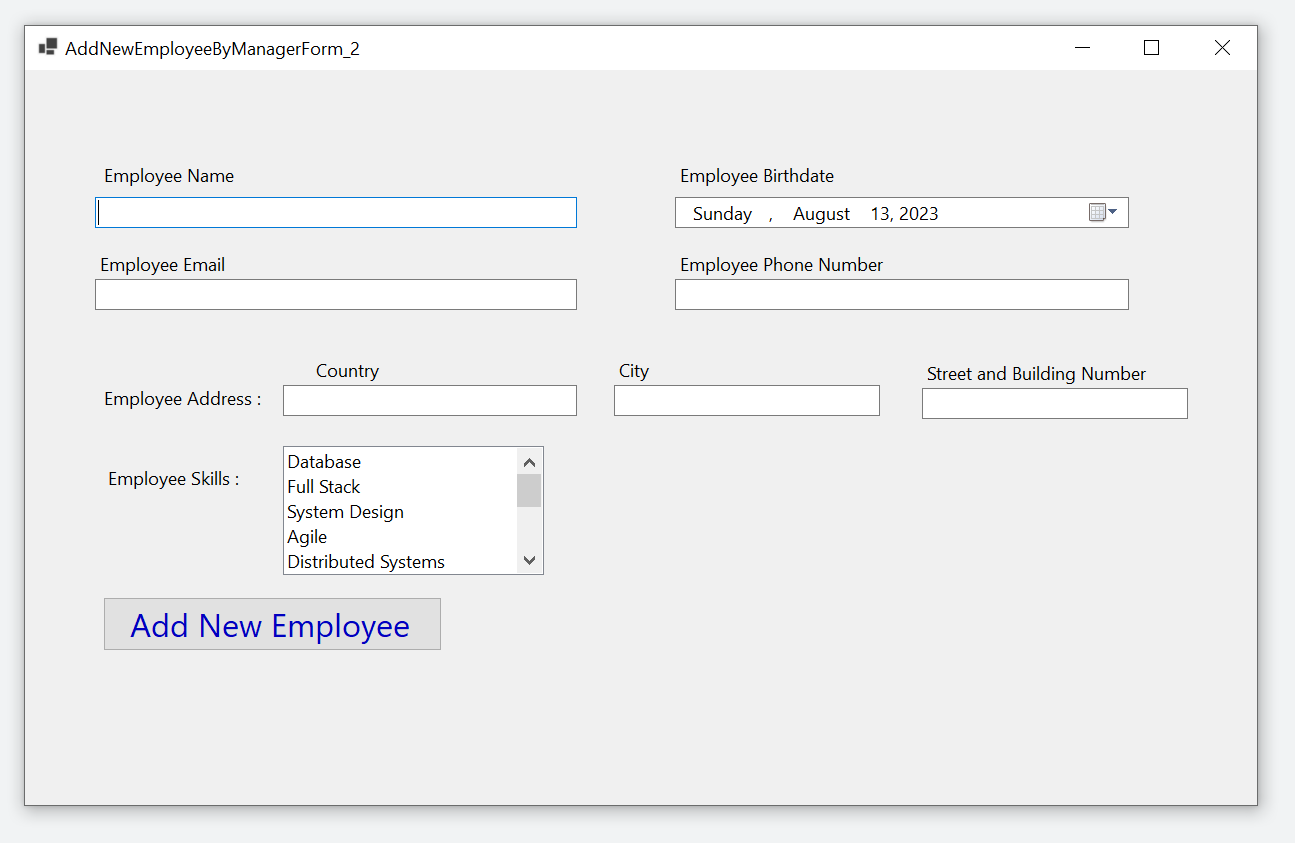
for example :

changing the ui element name from EmployeePhoneNumberTextBox to PhoneNumberTextBox will have effect side on this function and the developer need to revisit and fix this function :



You can think this is just a small fix , but in the real life and projects which have complex business logic and complex user interface the changing on the ui can have a very risk and dangerous effect on the business logic.

let’s see more complex example : change the skills text boxes to drop down menu that has multi select options :

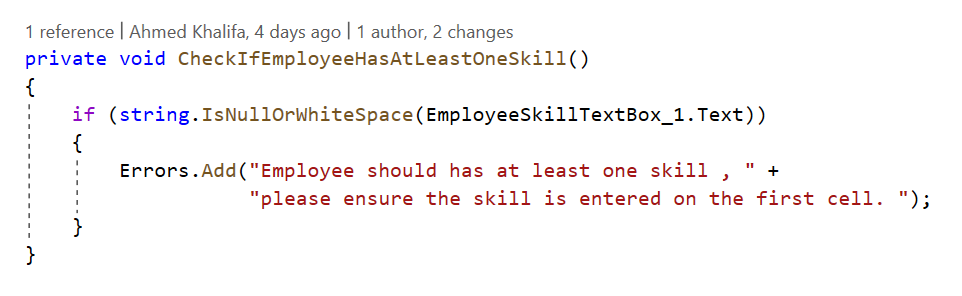


This ui change will have effect side on two business functions :

1. CheckIfEmployeeHasAtLeastOneSkill()

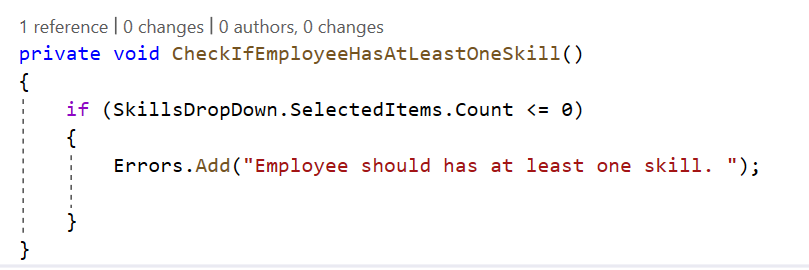
2. CreateNewEmployee()

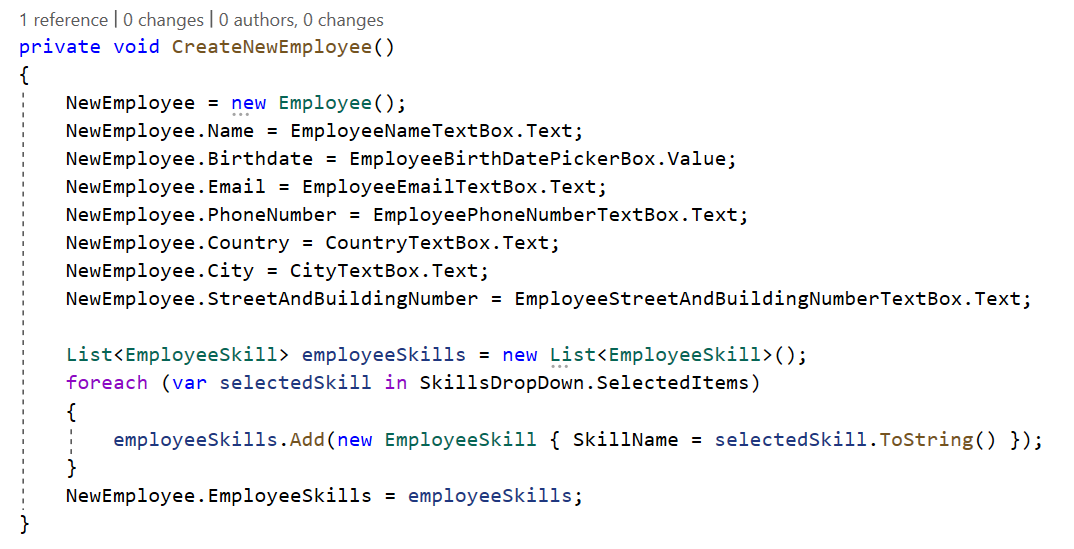
Old code :





New code :





**In the real projects with complex business logic and user interface , changing on the ui can cause a lot of conflicts and bugs when the business logic is very coupled and depended on the user interface , some developers can change the actual business logic by a mistake.**

**Third problem :**

**business logic is very coupled and depended on the infrastructure components like database :**

infrastructure is refer to :

1.the physical components in the system like databases or file system

2.external services like sms , emails , notifications and payments , this services are depended on external systems and usually the application interact with them through third party api , web services , or library or framework that the application setup them .

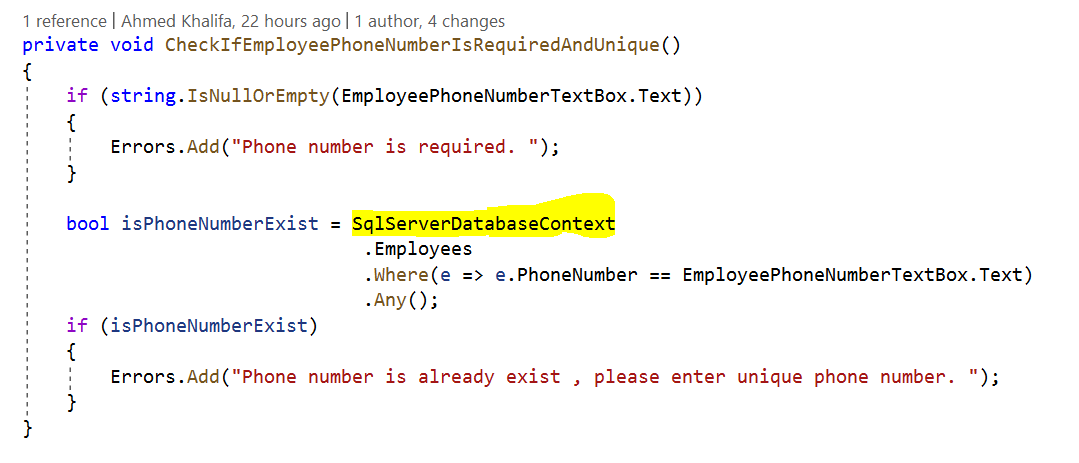
We will take database as infrastructure example : Applications usually need an orm framework to connect to the database so they can write or read data from the database.

the orm frameworks like entity framework core , dapper , ado.net are responsible :   
1- managing this communications   
2- Read and write operations   
2- mapping between database rows and application in-memory objects.

A diagram of a diagram

Description automatically generated

again in our business functions but this time we will discuss the infrastructure problem :



**Business Rule Function**

**Infrastructure :  
ORM Framework : EF core   
Database : SQL Server**

**A yellow text with black text

Description automatically generated**  
  
  
The two previous business functions contain **the actual implementation of the infrastructure code** for database , so the business logic will be very coupled on both the ORM framework and SQL Server database , so will have two cases :  
  
1- changing the ORM framework from entity framework core to for example dapper or ado.net  
2- changing the physical database from SQL Server to for example MongoDB or redis   
this two cases or any one of them will have a big effect on business function or logic although there are no actual change on the business itself , **business function should only change when there are change on the business itself not on the infrastructure component**

**Infrastructure :  
ORM Framework : EF core   
Database : SQL Server**

**Business Function**

This effect is happen because our code make the architecture look like this :

A screenshot of a computer

Description automatically generated

the outer layers or code will be very depended on the inner layers or code.  
so any change on the ORM Framework or the physical database will automatically effect on the business logic.

A diagram of a diagram

Description automatically generated

The new code after replacing SQL server by mongo database will modify the business functions and has a big effect on them :

A screenshot of a computer code

Description automatically generated

**Infrastructure :  
ORM Framework : MongoDriver  
Database : Mongo**

**Business Rule Function**

A screen shot of a computer code

Description automatically generated

**Infrastructure :  
ORM Framework : MongoDriver  
Database : Mongo**

**Business Function**

**Fourth problem : Code Duplications**

Now our company is growth and need web application beside the current desktop application, the two architectures will be like that :

A close-up of a computer

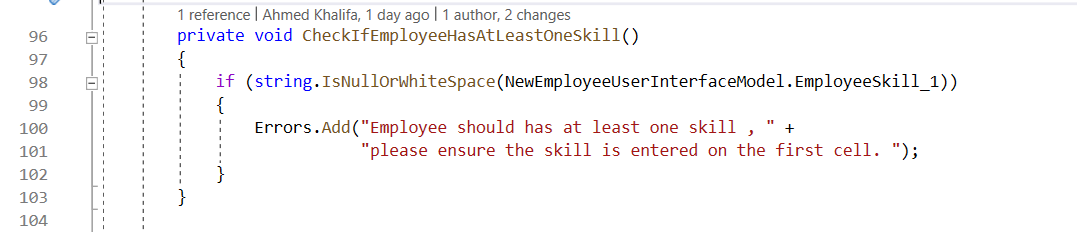
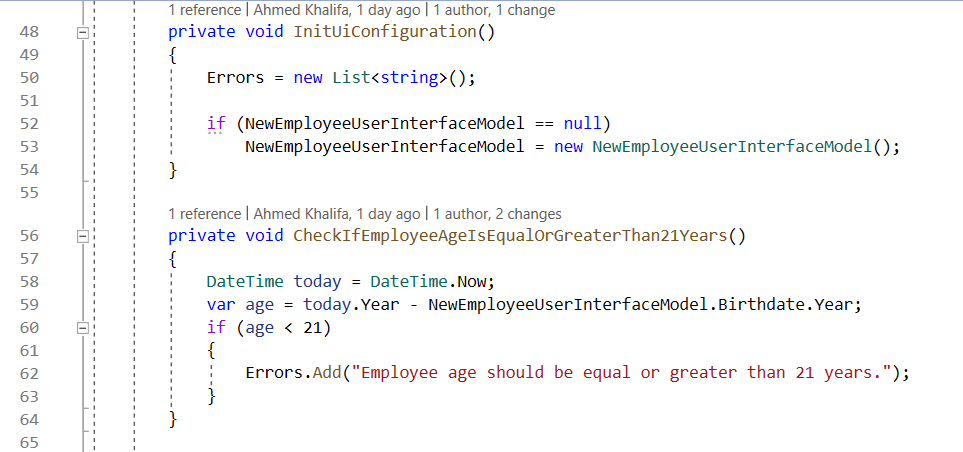
Description automatically generatedA close-up of a computer

Description automatically generated

The web application UI/UX :  
  
A screenshot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated  
Web application page model component code :



The code in two project will be duplicated , duplication is the evil of the software.

for example if you want to change business rule from min employee age 21 to 18 , the developer need to duplicated the code again in the two projects.

Desktop :

A screenshot of a computer

Description automatically generated

Web :

A yellow text with black text

Description automatically generated