Project Deliverable 2

BSEF19M022- Ashbal Fatima
BSEF19M020- Nosheen Hayat
BSEF19M024- Komal Riaz

Use Case Contracts

Operation:
SearchBy()
Cross Reference:
Use Cases: Search Services
Pre-Condition:
User searches for a narrator
Post-Condition:
Relative searches will be shown

Operation:
PayBill(Bill_Id: Bill_Id, Amount: integer)
Cross Reference:
Use Cases: Payment Management System
Pre-Condition:
The Buyer/service select a service/book to buy.
Post-Condition:
PayBill instance is created, and payment is made against it.

Operation:
PaymentVerification()
Cross Reference:
Use Cases: Payment Management System
Pre-Condition:
The buyer places the order and presses the checkout button.
Post-Condition:
A verification email is sent to the user. Payment is completed and the user is verified.

Operation:
Promotion()
Cross Reference:
Use Cases: Service/Book Promotion
Pre-Condition:
The Service Provider or Seller Pays for Promotion.
Post-Condition:
A promotion instance is created and that instance is added to the featured list.

Operation:
Profit()
Cross Reference:
Use Cases: Sell and Buy Book
Pre-Condition:
The Service Provider or Seller uploads a book.
Post-Condition:
From the original sale amount, profit will be calculated and added to the respective service provider or seller account.

Operation:
Commission()
Cross Reference:
Use Cases: Sell and Buy Book
Pre-Condition:
The Service Provider or Seller uploads a book.
Post-Condition:
From the original sale amount, the commission will be calculated and added to the client party's account.

Operation:
ProductSales()
Cross Reference:
Use Cases: Affiliate Sales
Pre-Condition:
The affiliate has a code that he added at the end of the product link and someone was redirected using that link.
Post-Condition:
An instance of sales will be created, which leads to an update in analytics and money will be transferred into the affiliate account.

Operation:
Library()
Cross Reference:
Use Cases: Buy Book
Pre-Condition:
A buyer bought a book.
Post-Condition:
An instance will be created which adds a book into the book list of library.

Architecture:

It is, of course, very early and impossible to anticipate all the things when we create our web application. Especially, in these turbulent times, when the

market, user needs, and demands, business goals, and technology can change very abruptly. The same goes for our decision for choosing an architecture for our application.

What architecture are we using?

We are using MVC as our architecture pattern.

WHY MVC?

MVC has been broadly adopted as a design pattern for WWW applications in major programming languages, Even though initially, it was developed for desktop computing. A lot of web frameworks have been created that enforce the pattern.

During the traditional approach of programming, there was a lack of maintainability, testability as well as scalability in the application development because the UI coding, business logic, and applications data domain were written into a single file.

With the arrival of the MVC approach, an application can be created where different aspects of the application (input logic, business logic, and UI logic) are divided into separate layers while providing a loose coupling between these elements. The pattern specifies the location of each kind of logic within the application. The UI logic, input logic, and business logic belong in the view, controller, and model respectively. This segregation helps in managing complexity by focusing on one aspect of the implementation at a time. For example, without depending on the input logic, you can focus on the view.

Following are some advantages and disadvantages of the MVC pattern:

Advantages:

- **Parallel development**: There can be parallel working on the model, controller, and view by multiple developers.
- **Loose coupling:** This feature makes the MVC framework flexible thus simplifying testing, maintenance, and troubleshooting procedures.
- **Easy modification:** As the responsibilities are separated from each other, future modification or development is easier i.e. product's scalability is increased.
- **A model with multiple views:** There can be Models having multiple views.
- **High Cohesion:** MVC permits logical grouping of related actions together on a controller and grouping of views for a specific model together. This feature is associated with traits like robustness, reliability, and reusability.

Disadvantages:

- **Code navigability** — The framework navigation can be complex because new layers of abstraction are introduced and users are required to adapt to the decomposition criteria of MVC.
- **Consistency problems:** The data is scattered, due to the breaking down of a feature into three artifacts. Thus, developers are required to maintain the consistency of multiple representations at once.

Conclusion:

After the above discussion, it can be deduced that in comparison to the traditional approach, using MVC architecture software development becomes smoother & robust due to its advantages of code reuse, parallel application development, etc. Nowadays, all the popular platforms of development like .NET, PHP, Java, have facilitated the use of MVC architecture.