كليـــــة .....

كونترول الفرقة .....

العـــام الجامعـــى 2019 / 2020 – دور مايـــو

الغلاف الخارجى للبحث

أولاً: البيانات الخاصة بالطالب			
الفرقة الدراسية	الثانية	التخصص	عام
اسم القسم	عام		
اسم المقرر	هندسة البرمجيات 1 - Software Engineering1		
استاذ المقرر	د/ عمرو غنيم		

ثانياً: البيانات الخاصة بالبحث				
عنوان البحث				
طبيعة المشاركة	بحث فردى		بحث جماعى	√
ارسال البحث	بواسطة البريد الالكتروني			
اسماء الطلاب المشاركين فى البحث (يكتب الاسم رباعيا)	م	الاسم رباعى	رقم الجلوس	الرقم القومى
	1	إسلام محمد محمود محمد	2281	30005272100655
	2	احمد بهاء حسن عبد الرحمن	2220	29904198800339
	3	احمد هشام عطية علي خضر	2264	30001122100074
	4			
	5			
تاريخ الإرسال	9 / 6 / 2020			

ثالثاً: البيانات الخاصة بالكونترول				
النتيجة	ناجح		راسب	
أعضاء لجنة تقييم البحث		الاسماء		التوقيع
	1			
	2			
	3			

فى حالة عدم قبول البحث يرجى ذكر الأسباب	- ..... - ..... - ..... - .....

PART 1: Overview & Software Requirements Specification

1)

- **Introduction to the Project:**

  -This is a fitness club website which has a lot of features to help not just the members joining it but also the trainers working in it and the administrators who run this fitness club.
  -For the members it aims to help them book the package or the session that suits them and also choose the trainer they want to take the session with in addition to writing reviews about trainers.
  -For the trainers it enables them to easily check what sessions they are assigned in and helps them to take the attendance of members easily for the sessions.
  -For the admins it gives them many privileges as they can easily make modifications to most of the existing data and view records of members, trainers, sessions and packages.
  So basically this website helps whoever uses it.

- **Glossary and abbreviations:**
  -MVC: Model, View and Controller
  -GUI: Graphical User Interface
- **System stakeholders:**

  -developers
  -production engineers
  -testers
  -users

- **References:**
  - Head first design patterns
  - Hands-on database (An introduction to database design and development) by: Steve Conger

2) → If we were to talk about **System Requirements Patterns,** we must Define what's **Requirements**:
The requirements of a system define what it needs to do-but not how.
The requirements define the problem that has to be solved: what the system is for and everything it needs in order to achieve that. They do not define the solution

System Requirements pattern:
an approach to specifying a particular type of requirement.
A requirement pattern is applied at the level of an individual requirement, to guide the specifying of a single requirement at a time.
Requirement pattern provide guidance like suggesting the information to include, giving advice, warning of common pitfalls, and it Saves Time

3) <u>Functional Requirements</u>:

This two tables showing **System Requirement pattern** used.

We used one of **Access Control Requirement Pattern**

Which is called **<Specific Authorization Requirement pattern**>

Category 2:

Functions that certain kinds of people are authorized to use. A requirement can either name an individual function or a class of functions

Requirement	Definition
Access only to data be granted when logged in	A user shall not have access to non-public functions or information if they have not logged in or have logged out.
Limited Casual visitor I.e. Member access	They shall not be able to even initiate any function that involves money (such as placing an order).
Employee maintenance access	The ability to maintain information about an employee (add, change, and remove) shall be limited Admin

Category 4:

Data To restrict access to certain information.

Requirement	Definition
Inquiries and reports not to show inaccessible data	No inquiry or report shall show data to which the current user does not have access. Where appropriate, inaccessible data shall be "filtered out." I.e. don’t show data from database that contain unauthorized data to the given category of users

FUNCTIONAL REQUERMENTS

A) User Requirements

In our System There are Three Actors which are :

-1- Admin :
a)Generate Reports
b)Manage Users of The System (Trainers and Members)
c)Manage Packages and Sessions
d)Notify Member about New items

2- Trainer :
a) Manage his Profile
b) Track Attendance Of his Members

3- Member :
a) Book in Particular Session Or Package
b) Review Trainer Performance
c) Manage his Profile
d) Have Multiple Payment Choices

B) System Requirement

-This project aims to Computerized gym management process , Booking process
The system should have capabilities to :
a) Generate reports
b) Choosing desired Trainer and Booking online
c) Manage Users Data
d) Tracking Member Progress
e) Track Member Satisfaction

3-C)	**REQUIREMNTS**	MOSCOW
A	1- USER MUST LOGIN 2- Member can Book Session or Packages 3-Admin can Generate Reports 4- Admin Add Session and Packages 5- Admin Can Manage Users	MUST
B	1- MEMBER should Sign up 2 Member Can Choose his payment Method 3- Trainer Or Admin Can View Member Attendance 4- Admin Can Send Notification	SHOULD
C	1- User Can update his profile 2-Member Can review Trainer	COULD
D	1- Member Or Trainer Cannot Change Session Or Package Data 2-- Member Or Trainer Cannot Change their UserName With Used one 3- Trainer Cannot Update his Salary	WON"T

4) Non-Functional Requirement:

A) The General Types/Categories of Non-Functional Requirements:

- Security
- Portability
- REUSABILITY
- MODIFIABILITY
- Usability
- Reliability / Availability

b) Non-Functional Requirements Specification

Portability
The ease with which a software system can be transferred from its current hardware or software environment to another

Reusability

The extent to which a portion of the software system can be reused in the same system or another

Modifiability

The degree to which changes to a software system can be developed and deployed efficiently and cost effectively

Usability:

The system uses a refined Graphic user Interface that facilitate the usage of the website services.

Reliability / Availability: System should be available the whole day and the whole year.

Security: - Users data should be secure. -User cannot view any other user data. -Trainer cannot view member's session or package. -Trainer can only change and view his data. -Trainer cannot change their shifts.

c) The Fit-Criteria for every Non-Functional Requirement:

1-**portability**:
can work on windows and Linux

2- **Reusability:**

a-each module can work efficiently and reused in another system

b- using Design pattern to make design flexible

c- reuse of functions and code

3-**Usability**:

GUI Can Pass user friendly Test

4- **Modifiability**:

system can be updated to meet user requirements

5-**Reliability** / **Availability**:

What are some of the ways you would expect our system to fail...

6-**Security**:

Only Admin can update Session and package data

Trainer cannot update his salary

User cannot view any other user data

D) Non-Functional Requirements affect the overall Architecture of the System:

As Seeking for security and separating data from login from view

We choose 3 tier architecture patterns as it increases availability

And reusability as 3 tier layers Allowed us to separate our system into 3 layers

Which is presentation (front end)

Business layer (which contains all the logic and business functions which drives application's core capabilities)

Data Access Layer (This is the data layer function, which receives the data from the business layer and performs the necessary operation into the database)

5) Domain requirements:

-This project aims to automate gym management process and fitness admission process as the admission process in gyms and Finding the desired Trainer are difficult. -This health club management system will help to overcome such problems by booking the desired program and trainer online and pay fees by electronic money transfer.

Generate reports to Admin

Allow admin to Manage all Resources

Admin can Notify Member about New items

2- Trainer:

a) Manage his Profile

b) Track Attendance Of his Members

3- Member:

a) Book in Particular Session or Package

b) Review Trainer Performance

6) Design Constraints:

Separate the system into layers to achieve flexibility, availability and Security

USING Design patterns to achieve Reusability and Modifiability

Implementation Constraints:

Implement project using PHP

Implement User Friendly GUI

7) System Evolution:

A) Anticipated changes:

1- Adding New Payment Methods to be sufficient customer needs

2- Adding new features for Trainer to improve User's benefits as (live sessions)

B)

1- Adding new Payment methods made me separate the payment process and apply Strategy Design Pattern

2- Making Trainer Class independent so it can be flexible to further Modification

8) Requirements discovery approaches:

Since there is no real client, we consider our Mentor as our Client

1-Requirement Planning:

We talked to our Mentor and have decided the pre-condition and post-condition for the vital requirements in our system, we also made appointments for Session for further discussing the Systems and keeping us updates with his needs

2-Requirement Elicitation:

We elicit Non-Functional Requirements from cover sheet such as Security and Reusability

3-Requirement Analysis and Documentation:

We take cover sheet Requirements and Analysis it to Functional, nonfunctional Requirements and Domain Requirements

4-Requirement Verification and Review:

We talked to our Mentor and Confirmed our doubts

5-Requirement Validation and Acceptance:

We talked to our Mentor and discussed most situations to check validations and His acceptance to our planning to the project

9) Requirements validation techniques:

We Used Two techniques:

1- **Test Case Generation:**

In This technique the Requirements are reviewed to the testers to conclude if the Requirements are testable and if it's hard or difficult or impossible to test any Requirement this Requirement should be Reconsidered.

We used this in our Project as the availability / Reliability non-functional Requirement is quite hard to test and should be reconsidered

2- **Requirements Review:**

In This technique the Requirements are reviewed by the Client

And Checks for any ambiguity

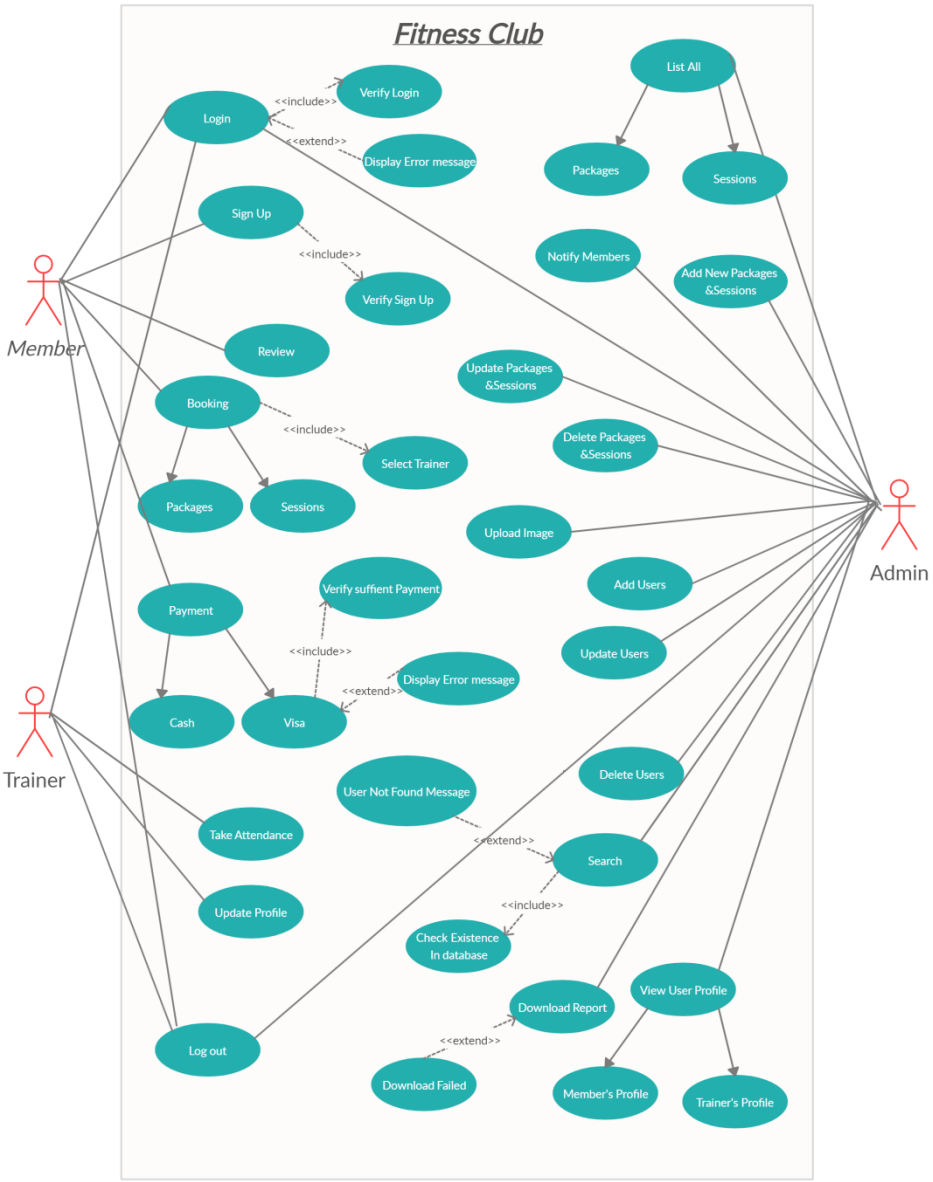
NOTE: Considerations for validation System Requirement pattern

Testing a specific authorization requirement is reasonably straightforward: can that kind of user access whatever it refers to Are all other kinds of users prevented from accessing it
An invaluable tool when testing authorizations is for all rejected authorization requests to be chronicled (logged). When reviewing requirements check that this feature is included. If not, insist on it

PART 2: System Design & Models

10)Functional Diagrams:

A) Use Case Diagrams



B) Detailed Use Case Documentation

Use Case Id:	1
Use Case Name:	Login
Description:	The member enters the website and login if he already has an account, if not he will sign up. The details entered is verified
Include Use Cases:	Verify Login use Case
Extends Use Cases:	Display an error Message use case
Initiator:	Any User of the system.
Preconditions:	User Must have a Valid Account.
Post Conditions:	The system displays the profile for the user.
Main Flow:	1)the user selects the login feature. 2)The system displays the Login form 3)The member Enters The user name and password

	4)The member Submit the user name and password 5)The system checks the user name and password in the data base. 6)The System Displays the User's Profile. 7) Use Case Ends
Alternative Flow:	5a) Un found User Name or password 1-The system displays an Error message 2-The System Prompt for User name and password 3-Use Case Resumes at Main flow step 2

Use Case Id:	2
Use Case Name:	Sign up
Description:	If the User doesn't have a membership in our Fitness Club. He has to sign up and fill in his details to be able to book and enjoy our website features.
Include Use Case:	Verify Sign up
Extends Use Case:	None.
Initiator:	Members.
Preconditions:	The user doesn't have a membership in The Fitness Club.
Post conditions:	The member has his membership in The Fitness Club.
Main Flow:	1)The user selects the sign-up feature. 2) The system Shows The form. 3)The user fills the form with his details. 4)The system Verify the details. 5)The system saves this data in the database. 6)Use Case Ends.
Alternative Flow:	4a) In case of Invalid Details. 1-The System Prompts an error message. 2-Use Case Resumes at Main flow step 2.

Use Case Id:	3
Use Case Name:	Booking
Description:	The user can see all the available packages with the cost and choose the suitable one for him. The user Have the ability to book either package or individual session. After that the user have to choose his trainer from the available trainers.
Include Use Cases:	Select Trainer
Extends Use Cases:	none
Primary Actors:	Members.
Secondary Actors:	Admin.
Preconditions:	The user must login in the system. The user wants to book a new package or session.

Post Conditions:	The user books the suitable choice for him. And also selects his trainer.
Main Flow:	1)Member choose the booking feature. 2)The system shows the available sessions and packages. 3)The user selects the suitable choice for him & selects his trainer. 4)The system saves the user's selection in the database. 5)The systems prompts Success message 6)the system shows payment screen 7)Use Case ends.
Alternative Flow:	--

Use Case Id:	4
Use Case Name:	Payment
Description:	After selecting the suitable choice & Trainer. The user is directed to the payment details. The user has the choice to either pay with visa card or by cash. If the user chose the visa selection, he has to enter his card details. If the user chose cash. The user had to pay at the fitness club.
Include Use Case:	*Verify sufficient payment.
Extends Use Case:	*Displays An error message.
Initiator:	Member.
Preconditions:	The user had already chosen a package or session to pay for. The Visa Card must have enough money for the payment process.
Post Conditions:	The user pays for his choice either by cash or visa.
Main Flow:	1)The system shows the payment form. 2)The user selects either Visa or Cash. 3)If the user selects Visa. the user enters his Card details. If Cash. The user will pay at the fitness Club. 4)The system verifies the payment. 5)The system saves the payment process in the database. 6)The system Prompt a confirmation message for the user. 7)Use Case ends.
Alternative Flow:	4a) If there is no enough Money in the card 1)The System prompts an error message for the user 2)Use case resumes from step 2

Use Case Id:	5
Use Case Name:	Review
Description:	The user has the ability to rate and Write review on his trainer's performance.
Include Use case:	None.
Extends Use Case:	None.
Initiator:	Members
Preconditions:	The user had already attended a session to have the ability to write a review.
Post Conditions:	The User writes a review on a trainer and rate his performance during the session.
Main Flow:	1)The user Choose the Review Feature. 2)The System shows all the user trainers. 3)The User selects a specific trainer. 4)The User Writes his reviews. 5)The User Presses Submit. 6)the system verifies the Data. 7)The system saves his review in the database. 8)Use Case Ends.
Alternative Flow:	5a) In case of Invalid Data. 1-the system displays an error message. 2-Use case resumes at step 4.

Use Case Id:	6
Use Case Name:	Notify Members
Description:	The Admin sends an email as a notification for the members for any updates and offers on the packages or sessions.
Initiator:	Admin.
Include Use Case:	None.
Extend Use Case:	None.
Preconditions:	New Updates Or offers in the sessions or packages information.
Post Conditions:	A Notification Email is sent to all the members in the club.
Main Flow:	1)The System calls the Notify when a modification is done and submitted by admin.

	2)The System sends automatically notification message for all member in the club. 3)Use Case ends.
Alternative Flow:	

Use Case Id:	7
Use Case Name:	Take Attendance.
Description:	Whenever a member attends a session, the trainer has to take attendance of this specific member and the system saves this data in the database. Also, The Admin can access the attendance sheet of each member.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	Trainer.
Preconditions:	A member had attended a session in the fitness club.
Post Conditions:	The Attendance of each individual member is taken by the Trainer.
Main Flow:	1)the trainer selects taking attendance feature. 2)the system displays the attendance form 3)the trainer writes the member name and the session name and the date. 4)the trainer submits the information. 5) the system verifies the data. 5)the system saves the attendance in the database 6)use case end
Alternative Flow:	4a) In case of Invalid Data. 1-the system displays an error message. 2- Use case resumes at step 3.

Use Case Id:	8
Use Case Name:	Update Profile.
Description:	The trainer has the right to change his profile details.
Include Use Case:	None
Extend Use Case:	None
Initiator:	Trainer
Preconditions:	The trainer wants to update his profile details (he can also reset his password)
Post conditions:	The trainer's profile updated
Main Flow:	1)the trainer selects update profile feature 2)the system retrieves his data from the database. 3)The system shows this Data to the trainer.

	4)The Trainer can change either his password or his details. If he presses reset button, the system displays a form with three text fields one for the old password and two for the new password and the trainer enters his old password and the new password and a confirmation for the new password, if the trainer presses Modify button, he will write his new modifications in the form. 5)the Trainer press submit button. 6)The data will be saved in the database. 7)the system sends a confirmation message. 8)use case end
Alternative Flow:	--

Use Case Id:	9
Use Case Name:	Log out.
Description:	After the user finishes his work on the system, he can close profile by logout.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	User (Member, Admin, Trainer)
Preconditions:	User need to close his profile.
Post conditions:	the user profile is closed and the system returned to login page.
Main Flow:	1)The user selects to logout. 2)The system displays the message to confirm logout. 3)The user confirms to logout. 4)The system displays the Home pages. 5)use case end
Alternative Flow:	

Use Case Id:	10
Use Case Name:	Search.
Description:	The Admin have the authority to search for specific Trainer by writing his name in the search engine. And the system checks if this trainer is found in the database or not.
Include Use Case:	Check Existence in database
Extend Use Case:	User Not Found Message
Initiator:	The Admin.

Preconditions:	The Admin wants to search for a specific Trainer and enters his name.
Post conditions:	The System shows the details of the required Trainer.
Main Flow:	1)The Admin selects Search button. 2)The Admin writes the name of the trainer in the search Bar. 3)the system retrieves this trainer data from the database if exists Or Show error message if not. 4)The system shows this Data to the Admin. 5)Use Case ends.
Alternative Flow:	

Use Case Id:	11
Use Case Name:	List
Description:	The Admin Can see the list of all the sessions and packages taking place in the fitness club.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	The Admin.
Preconditions:	The Admin has logged in and wants to access all the data of the sessions and packages from the database.
Post conditions:	The system lists to the admin all the sessions and packages.
Main Flow:	1)The Admin selects the List all packages and session button. 2)The System Retrieves the data of all packages and sessions from the database. 3)The System shows this data to the admin in a Tabular Form. 4)Use Case Ends.
Alternative Flow:	

Use Case Id:	12
Use Case Name:	Add Users
Description:	The Admin have the Authority to add a new User (either a Trainer or a member). The Admin Enters All the details Required in the system.
Include Use Case:	None
Extend Use Case:	None
Initiator:	Admin
Preconditions:	The Admin presses The Button Add To add a new user.
Post conditions:	The System Saves this data in the database.
Main Flow:	1)The admin selects the Add user Button. 2) the system will provide a form to fill it with the user details 3)The admin adds user details to the add user form. 4)The Admin Presses Submit. 5)The system verifies the details 6)The system saves the user data at database 7)end use the case.
Alternative Flow:	

Use Case Id:	13
Use Case Name:	Update Users
Description:	The Admin have the Authority To modify the User details Information (Either Member or Trainer) In the database.
Include Use Case:	None.
Extend Use Case:	None
Initiator:	Admin
Preconditions:	The Admin wants to modify User's Data and Presses the Update Button.
Post conditions:	The Data is Updated and saved in the database.
Main Flow:	1)The admin selects Update User Button. 2) the system will display text field to add id of the desired user 3) the admin enters id of a certain user to the text field 4)the system retrieves a data of the specific user from the data base. 5) The system shows this data to the admin to make any modifications 6) the admin makes his modifications then press submit 7)The system verifies the details 8)or the system saves the updated data In database

	9)end use the case.
Alternative Flow:	

Use Case Id:	14
Use Case Name:	Delete Users
Description:	The Admin Have the Authority to Delete users (Either Members or Trainers) From the database.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	Admin
Preconditions:	The Admin wants to delete the data of a specific User, and presses the delete Button.
Post conditions:	The Data Of this specific User is Deleted From the database.
Main Flow:	1)The admin selects the Delete Users Button 2) the system displays a text field to add id of the desired user. 3)the system deletes the user's data from the database. 4)The system saved the change at the database. 5)end use the case.
Alternative Flow:	

Use Case Id:	15
Use Case Name:	Delete Packages and sessions
Description:	The Admin Can Delete any session or package in the system.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	Admin
Preconditions:	The Admin Presses on the delete Button to delete a specific Package or session.
Post conditions:	The Specific Package Or session is deleted from the database.
Main Flow:	1)The Admin chooses The Delete Button. 2)The system asks if the user wants the Packages or sessions to make modifications on. 3)The Admin makes his choice. (either packages or sessions)

	4)The system displays an id text field for the admin to write which package or session to be deleted. 5)The Admin enters the id of the product to be deleted in the data base. 6)The admin presses the Button submit. 7)The systems delete That Specific Package or session from the database. 8)the System Prompts for the use a Success message. 9)The system Calls the notify members function. 10)Use Case Ends.
Alternative Flow:	

Use Case Id:	16
Use Case Name:	Update Packages and sessions
Description:	The Admin can Update the Information of any Package or session.
Include Use Case:	None
Extend Use Case:	None.
Initiator:	Admin.
Preconditions:	The Admin wants to modify on an existent Package or session.
Post conditions:	The Data is Modified And saved in the database.
Main Flow:	1)The Admin chooses the Update Button. 2)The system asks if the user wants the Packages or sessions to make modifications on. 3)The Admin makes his choice. (either packages or sessions) 4)The system displays an id text field for the admin to write the desired id of such a package or session. 5)The Admin enters the id of the product which needs modifications. 6)The system Retrieve the data from the database. 7)The system shows this data In Tabular Form. 8)The Admin makes the Required Modifications. 9)the Admin Presses Submit Button. 10)The System Verifies the New Data 11)The system saves this data in the database. 12)The System Display a Success message 13)The system Calls the notify members function. 14)Use Case Ends.

Alternative Flow:	

Use Case Id:	17
Use Case Name:	Add Packages and sessions
Description:	The Admin Can add a new package or session To the Fitness Club.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	The Admin.
Preconditions:	The Admin wants to add a new Product To the system and presses the Add Button.
Post conditions:	The New Data is Added in the system.
Main Flow:	1)The Admin chooses Add Button. 2)The system asks if the user wants the Packages or sessions to make modifications on. 3)The Admin makes his choice. (either packages or sessions) 4)The system displays A form to fill it with the details of either packages or sessions. 5)The admin writes the details of the new product. 6)The admin presses the Button submit. 7)The system Verifies this data in the database. 8)The systems save the new data in the data base. 9)the System Prompts for the use a Success message. 10)The system Calls the notify members function. 11)Use Case Ends.
Alternative Flow:	

Use Case Id:	18
Use Case Name:	Download Report
Description:	The Admin Receives a Monthly Report From the system with the number of members Joined Each Package or session. This Report Can be downloaded by the admin in PDF form.
Include Use Case:	None
Extend Use Case:	Download Failed.
Initiator:	The Admin
Preconditions:	The Admin wants to download the monthly Evaluation of the Packages and Sessions.
Post conditions:	The system Displays A Report with the required data and This data is Downloaded for the admin.
Main Flow:	1)The Admin Presses Download Monthly Report.

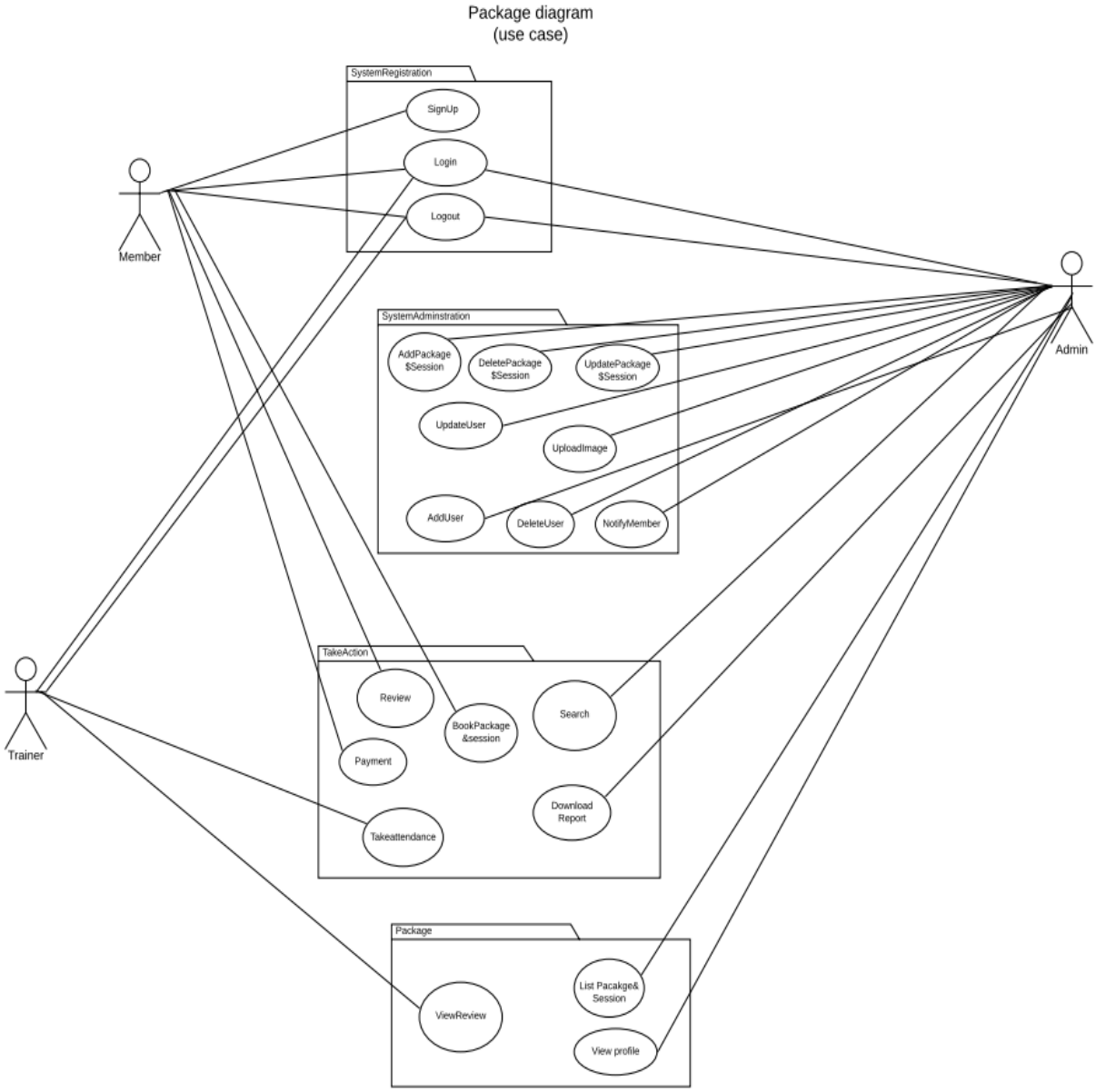
	2)The system Retrieves the report from the database 3)The System download this Report to the Admin's Computer. 4)Use Case Ends.
Alternative Flow:	2a) In case of Download Failed. 1-the system displays an error message. 2- Use case resumes at step 1.

Use Case Id:	19
Use Case Name:	Upload Image
Description:	The Admin can Upload Imaged to the Packages and sessions data to show the equipment which are used During the sessions.
Include Use Case:	None.
Extend Use Case:	None.
Initiator:	The Admin
Preconditions:	The Admin presses the upload button as to upload a new image in the sessions and packages.
Post conditions:	The Image Is Uploaded
Main Flow:	1)The Admin Presses Upload Images Button. 2)The system displays all the available images that can be uploaded 3)The Admin selects The Required Image. 4)The Admin Presses Submit. 5)The System Upload this image. 6)The system saves this image in the database. 7)The system displays A Success Message. 8)Use Case Ends.
Alternative Flow:	

Use Case Id:	20
Use Case Name:	View Profile
Description:	The Admin Can View The profile of a specific User (Member or Trainer)
Include Use Case:	None.
Extend Use Case:	None
Initiator:	The Admin
Preconditions:	The Admin Presses the View Button to view the profile of A specific User.
Post conditions:	The System Displays the Specific User's Data.
Main Flow:	1)The admin selects View Profile Feature

	2) the system displays text field to add id of the desired profile. 3)The Admin Writes the Id of the user. 4)The system Retrieves the Data From the database 5)The system Displays This data to the admin. 6)end use the case.
Alternative Flow:	

C) Package Diagram grouping use-cases into packages



11) **Structural & Behavioral Diagrams**

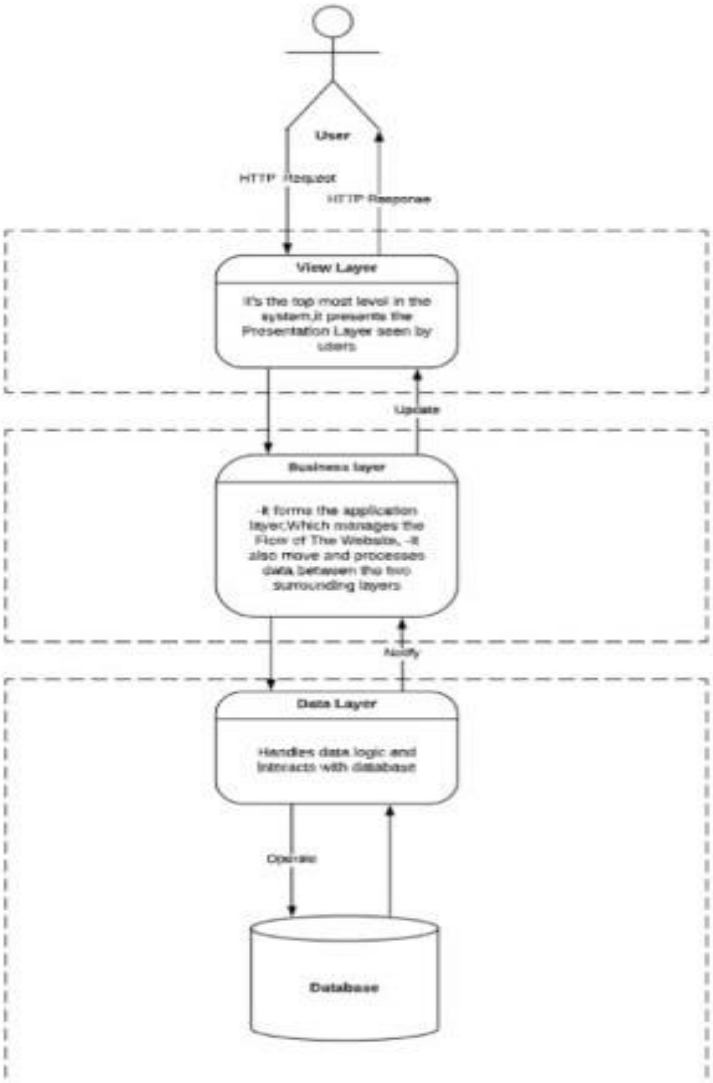
A) System Architecture (including applied Architectural Pattern(s), i.e., what patterns have you applied and why?)

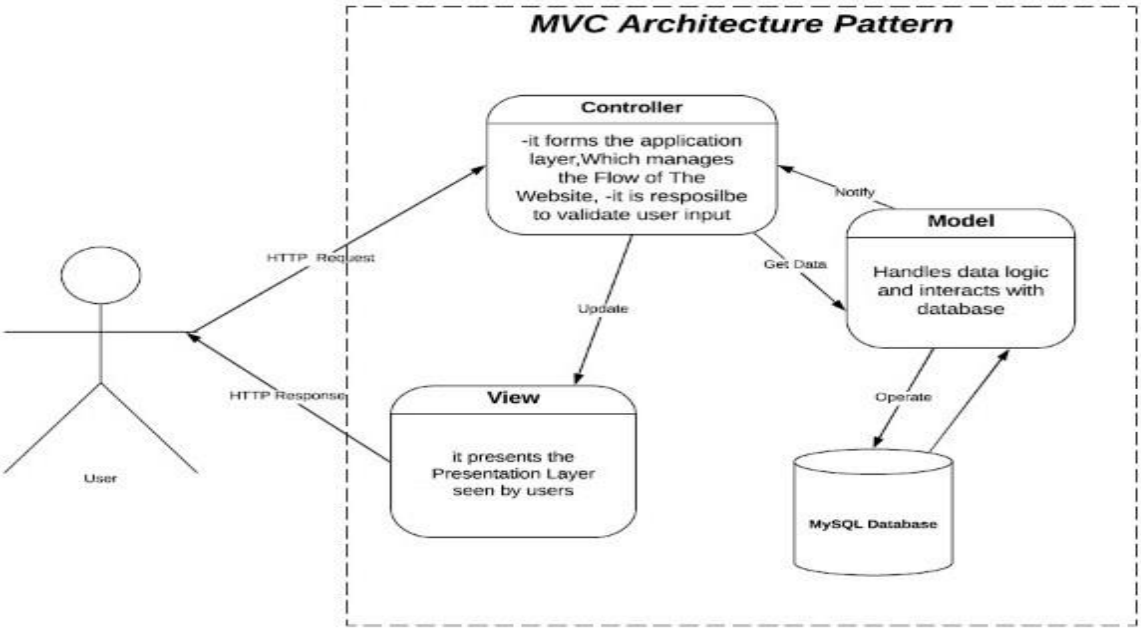
System Architectural Pattern

In this project we have applied the 3 tiers architectural pattern. Where Three-tier architecture is a software design pattern and a well-established software architecture.
the three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.
The three tiers architectural pattern consists of three sperate layers which are:

- ***Presentation tier*** *which* is also known as View layer, it Occupies the top level and displays information related to services available on a website in the form of a graphical user interface (GUI).
- ***Application tier*** which is also known as Logic or business layer, it controls the project's core functionalities and all the main operations that the project proposes.
- ***Data tier*** which is a database management system that provides access to application data.

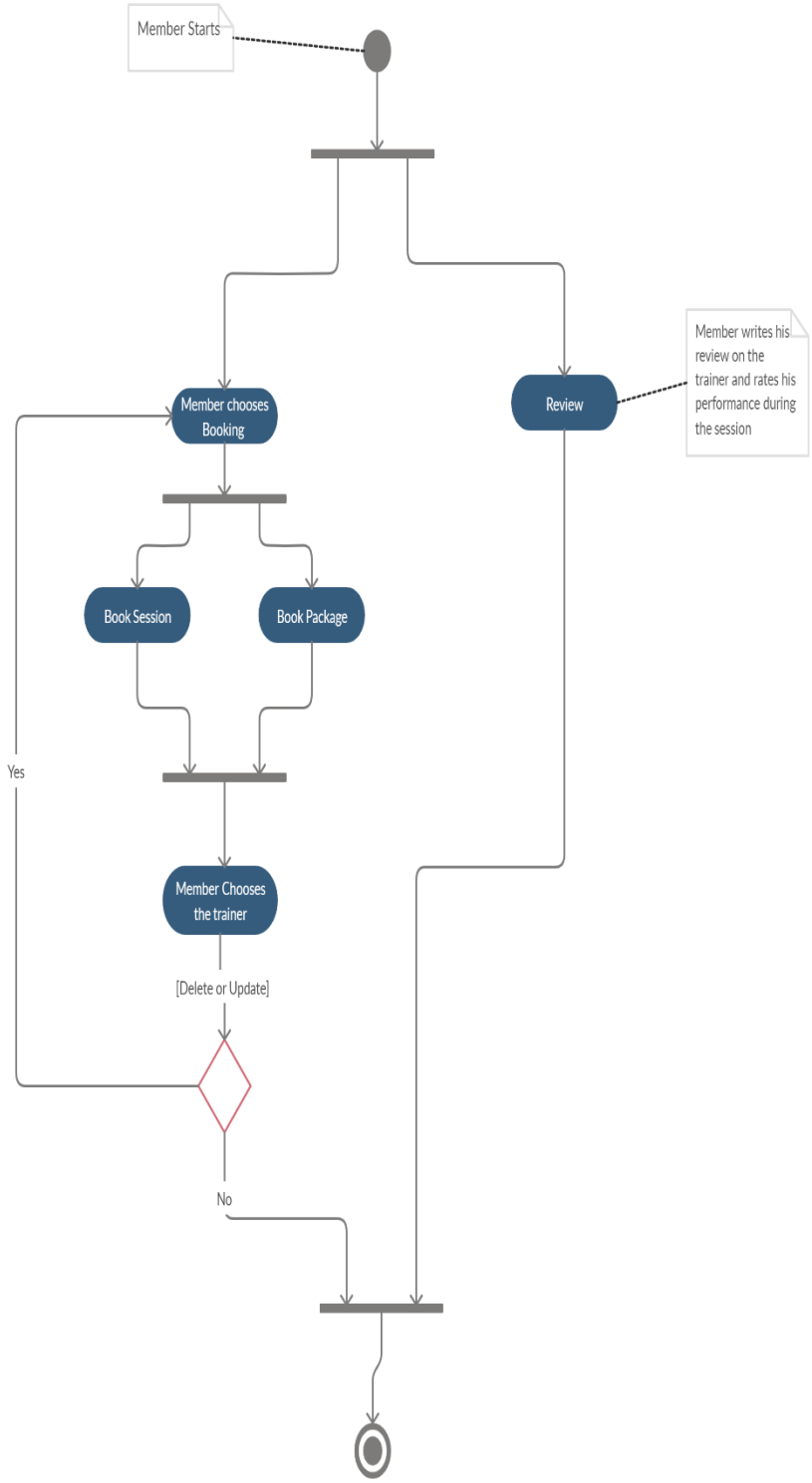
- The Reason of the choice of this specific pattern as it has lots of benefits and great capabilities. As the three tiers fully insulate users of the project from business rules, the underlying data storage, and concurrency issues, resulting complete encapsulation of the project data. As clients only interact with one single tier which is **View Tier** and therefore changes can be made in the database without having to touch a single line of code at the client.


User
HTTP Request
HTTP Response
View Layer
It's the top most level in the system,it presents the Presentation Layer seen by users
Update
Business layer
-It forms the application layer,Which manages the Flow of The Website, -It also move and processes data between the two surrounding layers
Notify
Data Layer
Handles data logic and interacts with database
Operate
Database

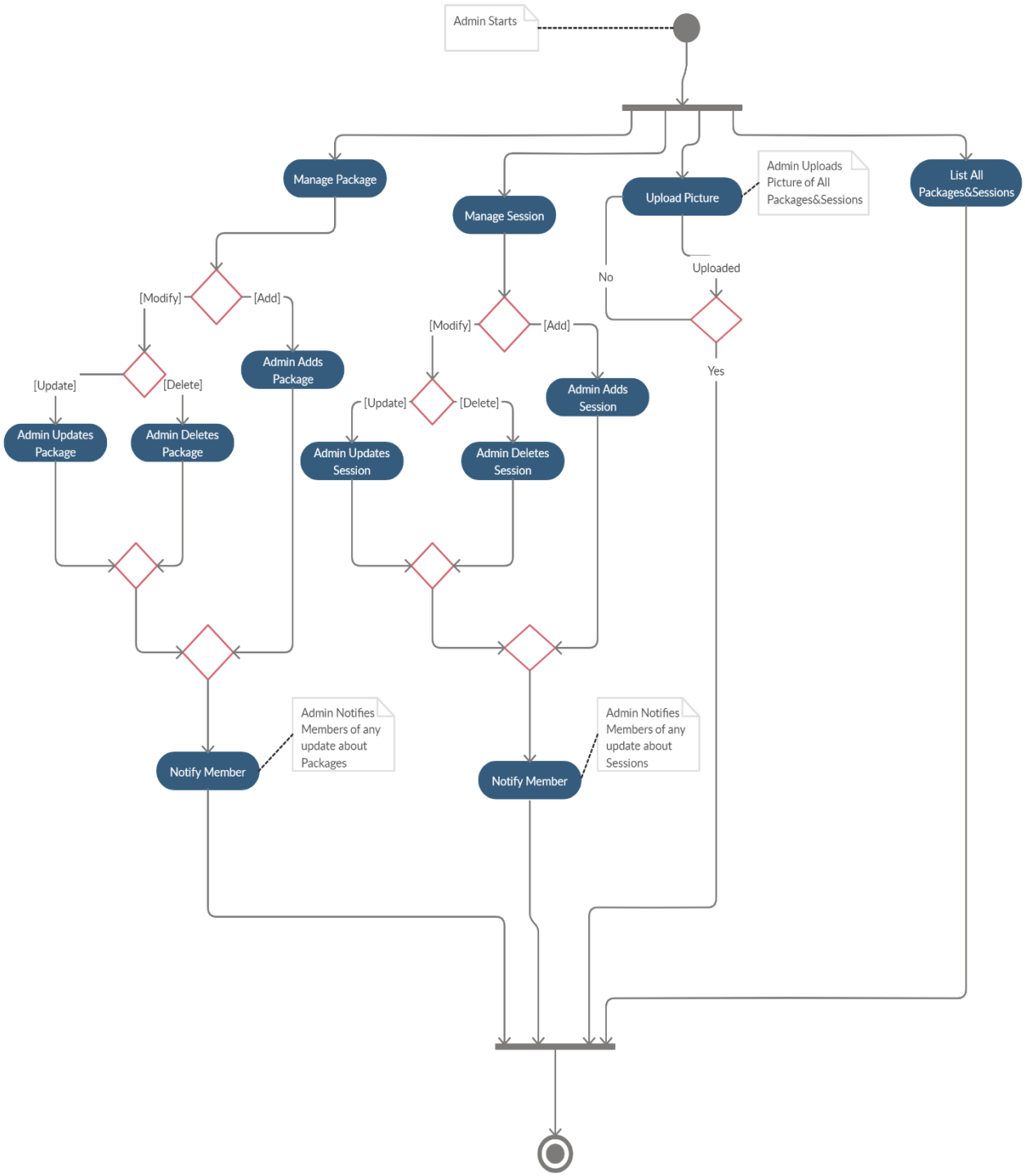


b) Activity Diagrams (for every major business process in the system, at least 6 diagrams.)

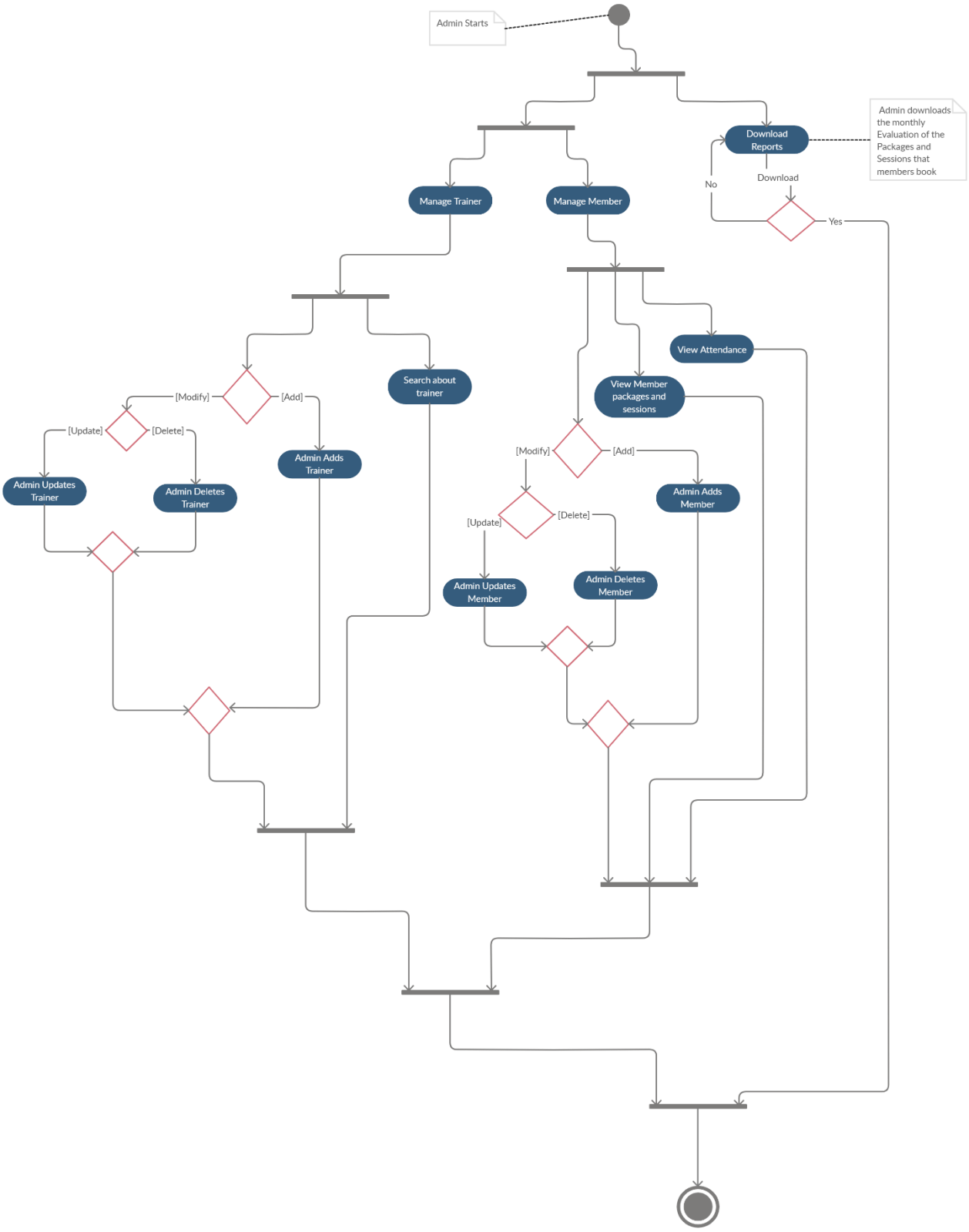
- **Booking Activity Diagram**



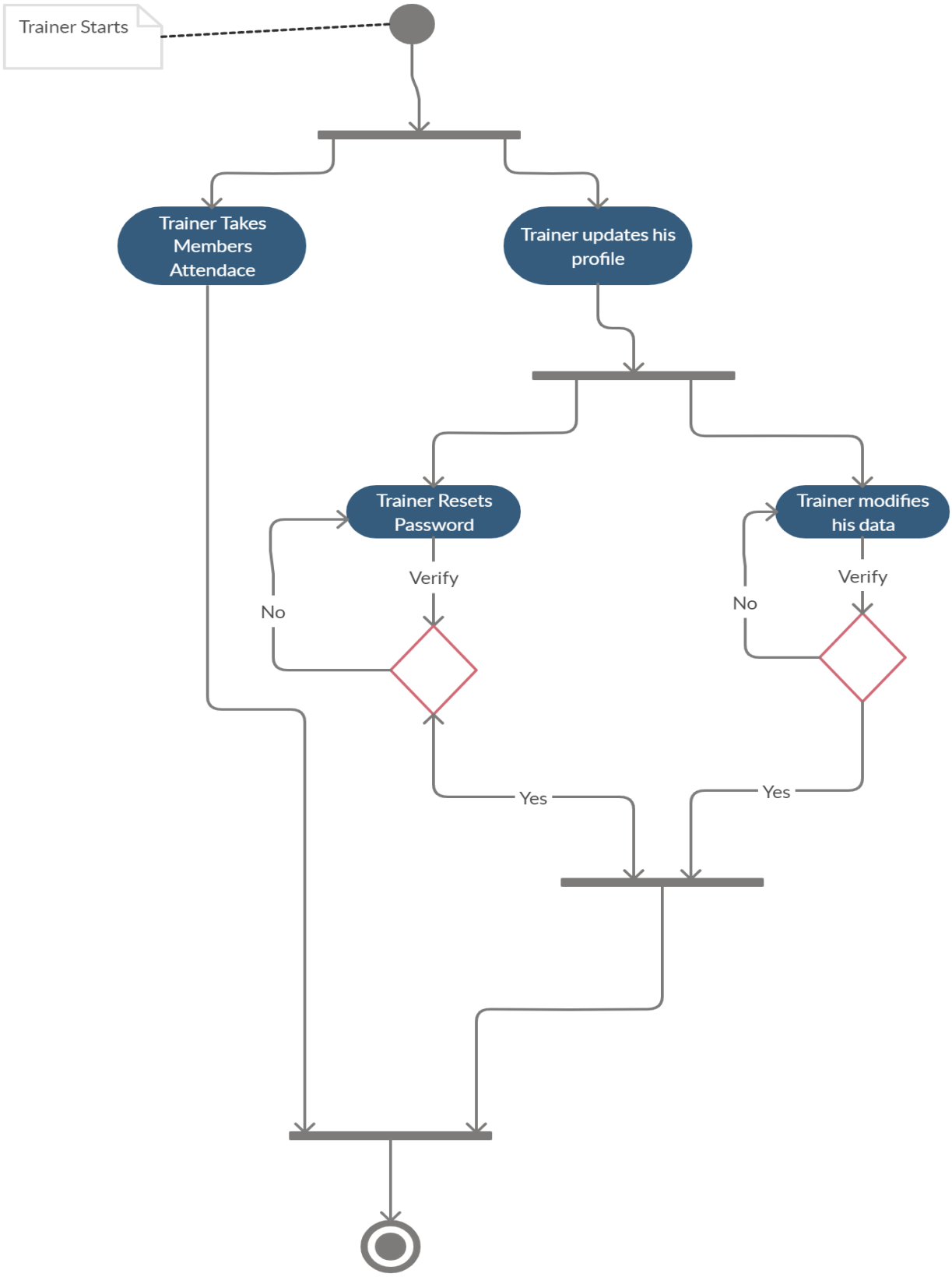
- **Manage Packages & Sessions-Notify-List-Upload Activity Diagram**



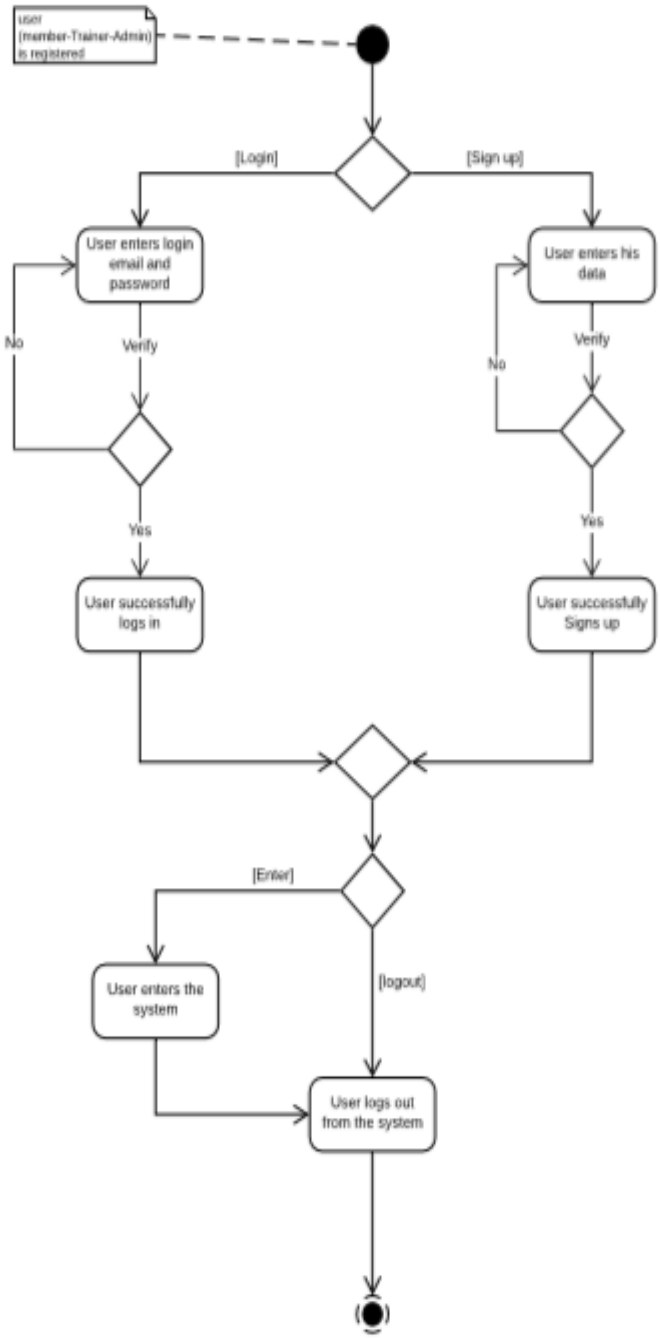
- **Manage Users-Report - Search Activity Diagram**



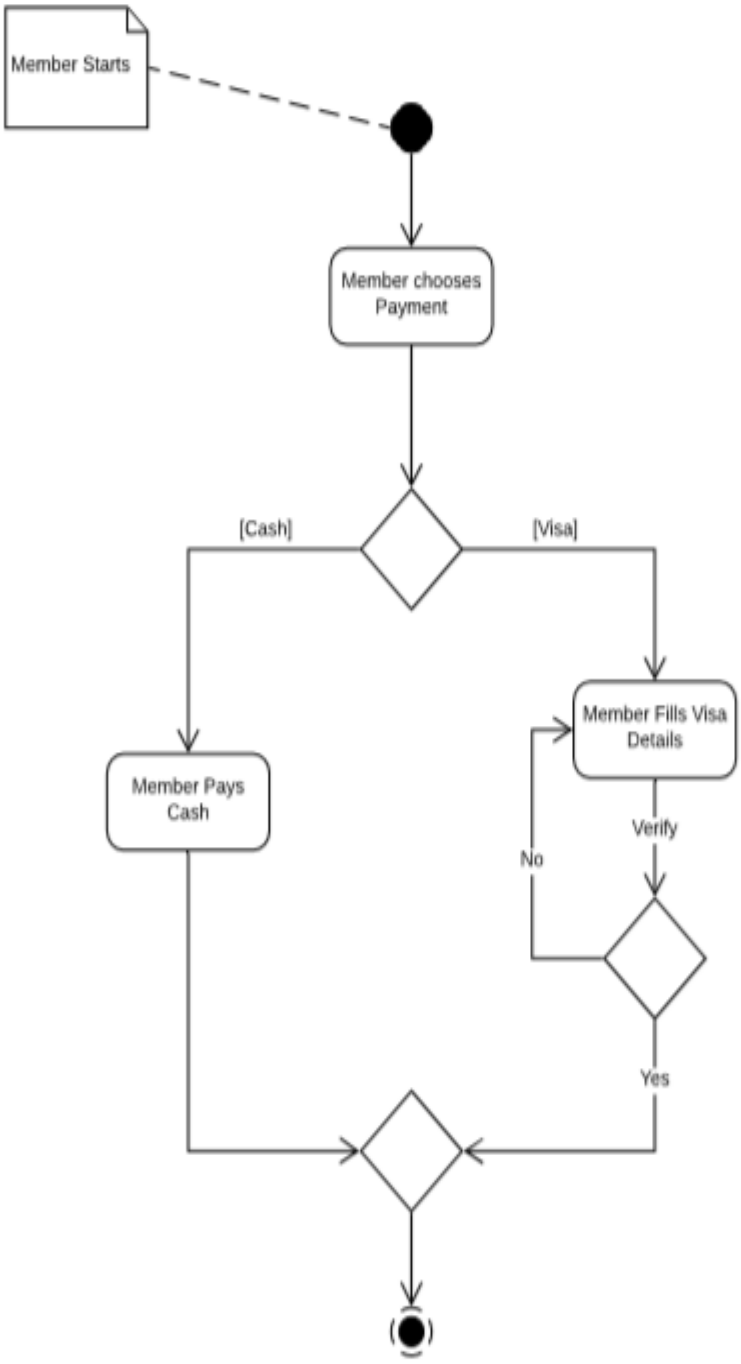
- **Update Profile-Take Attendance Activity diagram**



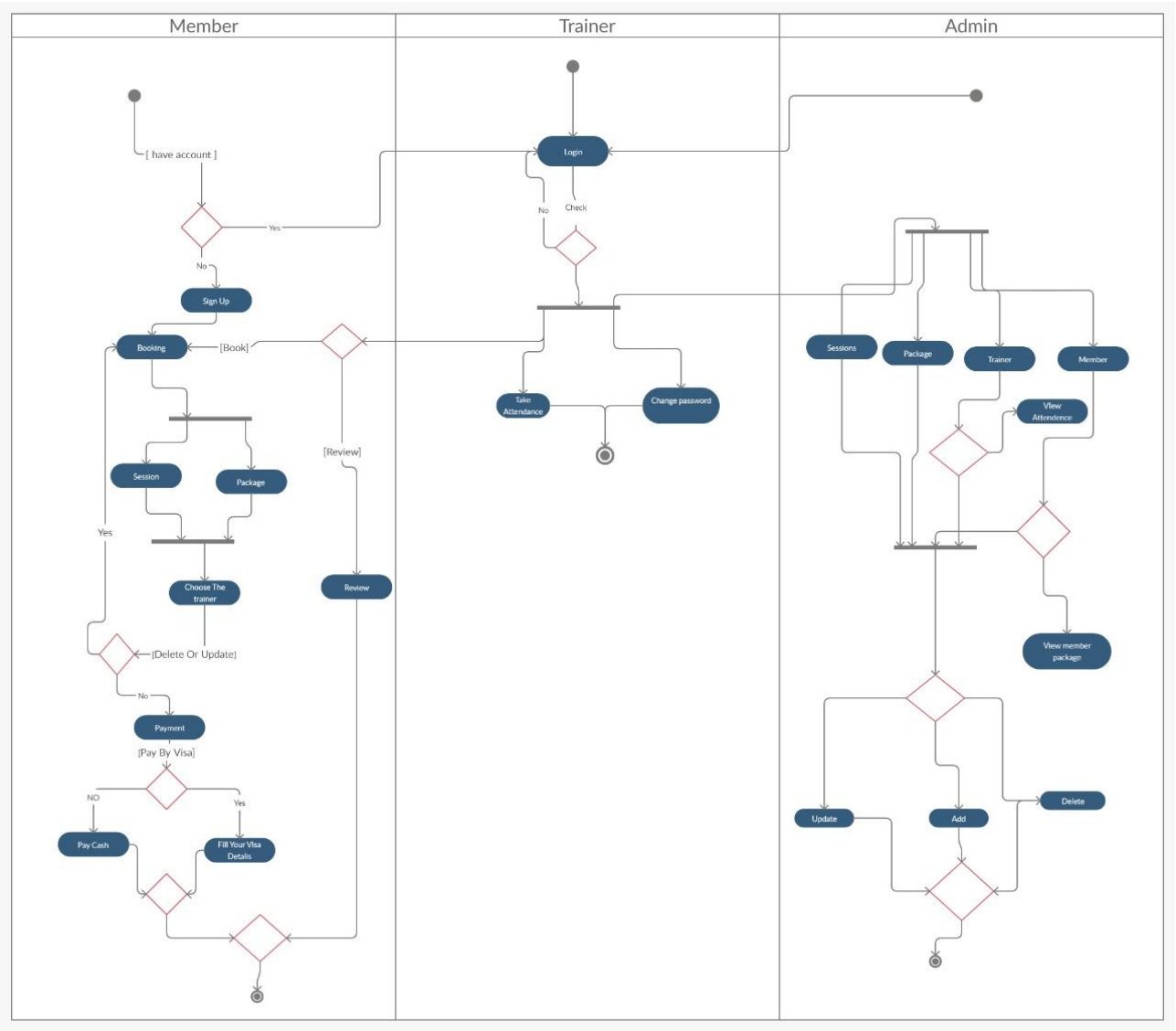
- **Register activity diagram**



- **Payment Activity Diagram**



- **First Activity diagram for all the system:**



c) Based on the Activity Diagrams, the List of User Interfaces required for the System, and the corresponding users of each interface.

1. Booking User Interface (Booking & review Activity Diagram).

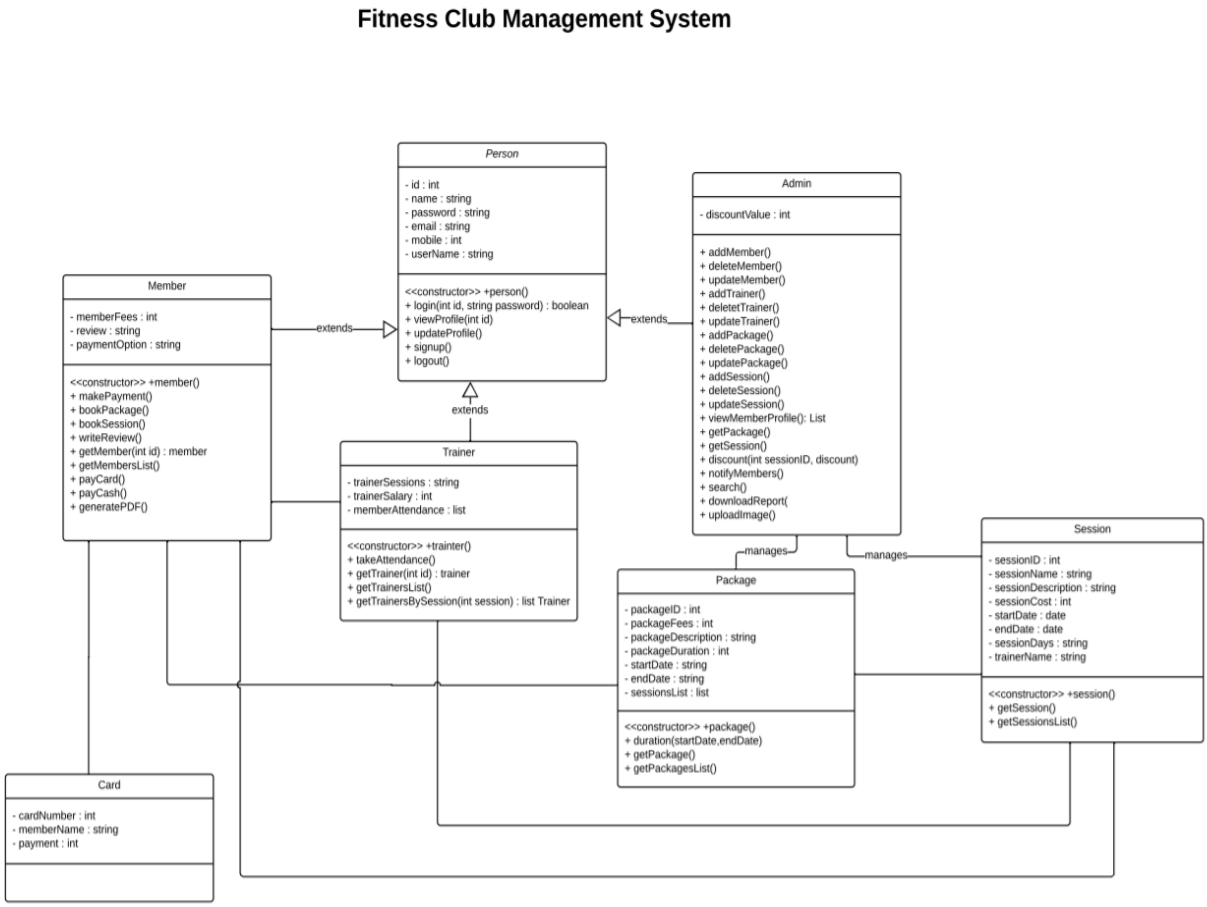
-Member interacts with Booking User interface to book either package or session and then UI shows the trainers for the member to choose trainer.

2. Review User Interface (Booking & review Activity Diagram).
   -Member interacts with Review UI to write or see reviews on his trainer and UI shows the reviews on the trainers to the member.
3. Packages & Sessions User Interface (manage package & session Activity diagram).
   -Admin interacts with package and session UI to manage them with adding or deleting or updating packages or sessions, Admin can also interact with Package or session UI with listing all the packages and sessions and he can also upload pictures of them to user interface.
4. Member & Trainer User Interface (Manage Users Activity diagram).
   - Admin interacts with Member and Trainer UI to manage them with adding or deleting or updating Member or Trainer, admin interacts with UI to download Reports, admin interacts with member UI to view their attendance and view their packages and sessions, admin interacts with trainer UI to search about trainer.
5. Payment User Interface (Payment Activity Diagram).
   -Member interacts with Payment User Interface to pay for his booking and he interacts with UI to choose cash or Visa.
6. Login User Interface (Register Activity Diagram).
   -Users (Member or Trainer or Admin) interact with login UI to login and enter the system.
7. Sign Up User Interface (Register Activity Diagram).
   - Users (Member or Trainer or Admin) interact with Sign Up UI to sign up and have a new account.
8. Log Out User Interface (Register Activity Diagram).
   - Users (Member or Trainer or Admin) interact with log out UI to log out from the system.
9. Trainer User Interface (Update Profile Activity Diagram).
   -Trainer interacts with Trainer UI to update his profile and set his password.

D) Software Analysis Pattern:

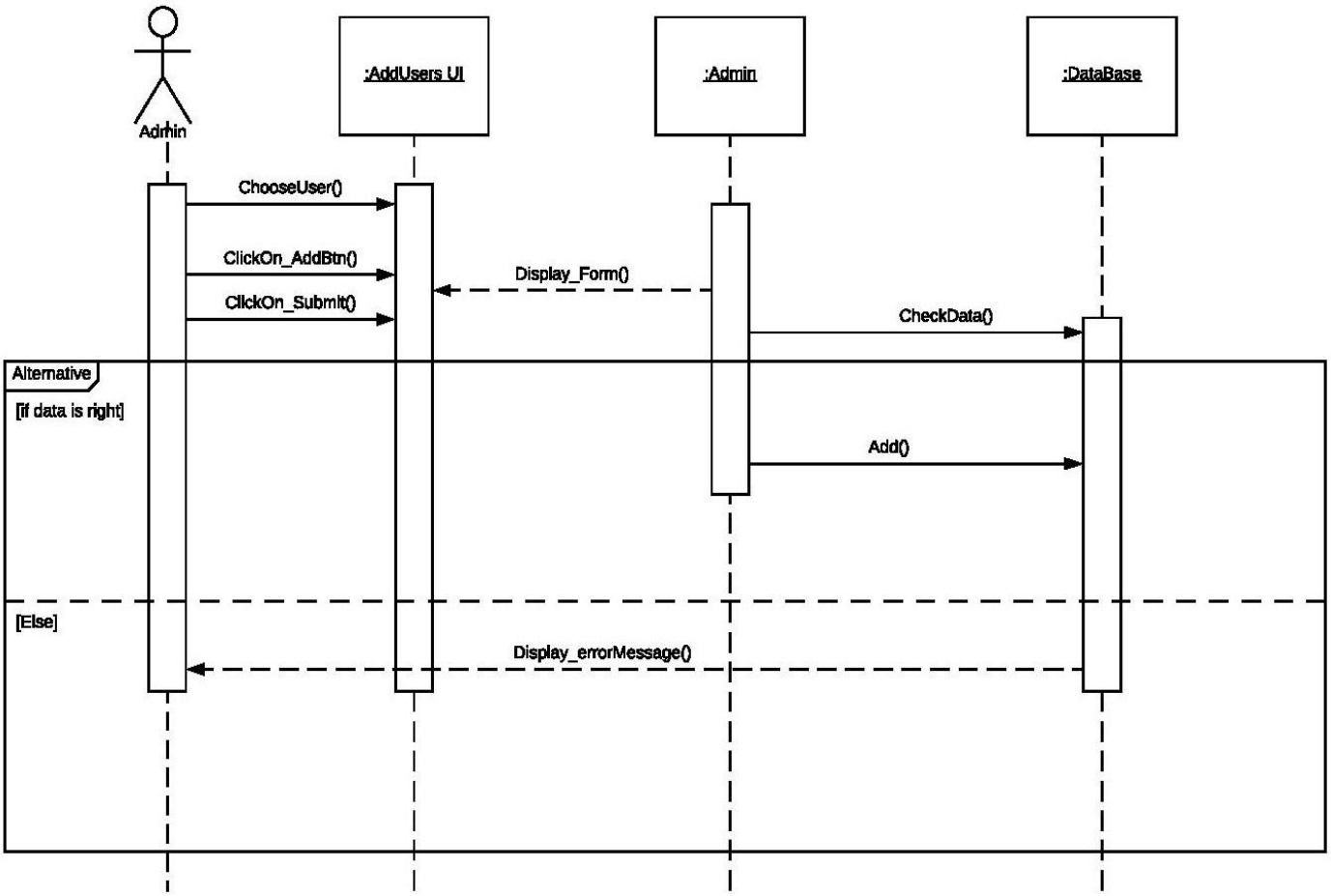
-Analysis patterns are groups of concepts that represent a common construction in business modeling. It may be relevant to only one domain, or it may span many domains.

E) Class diagram 1: The initial version

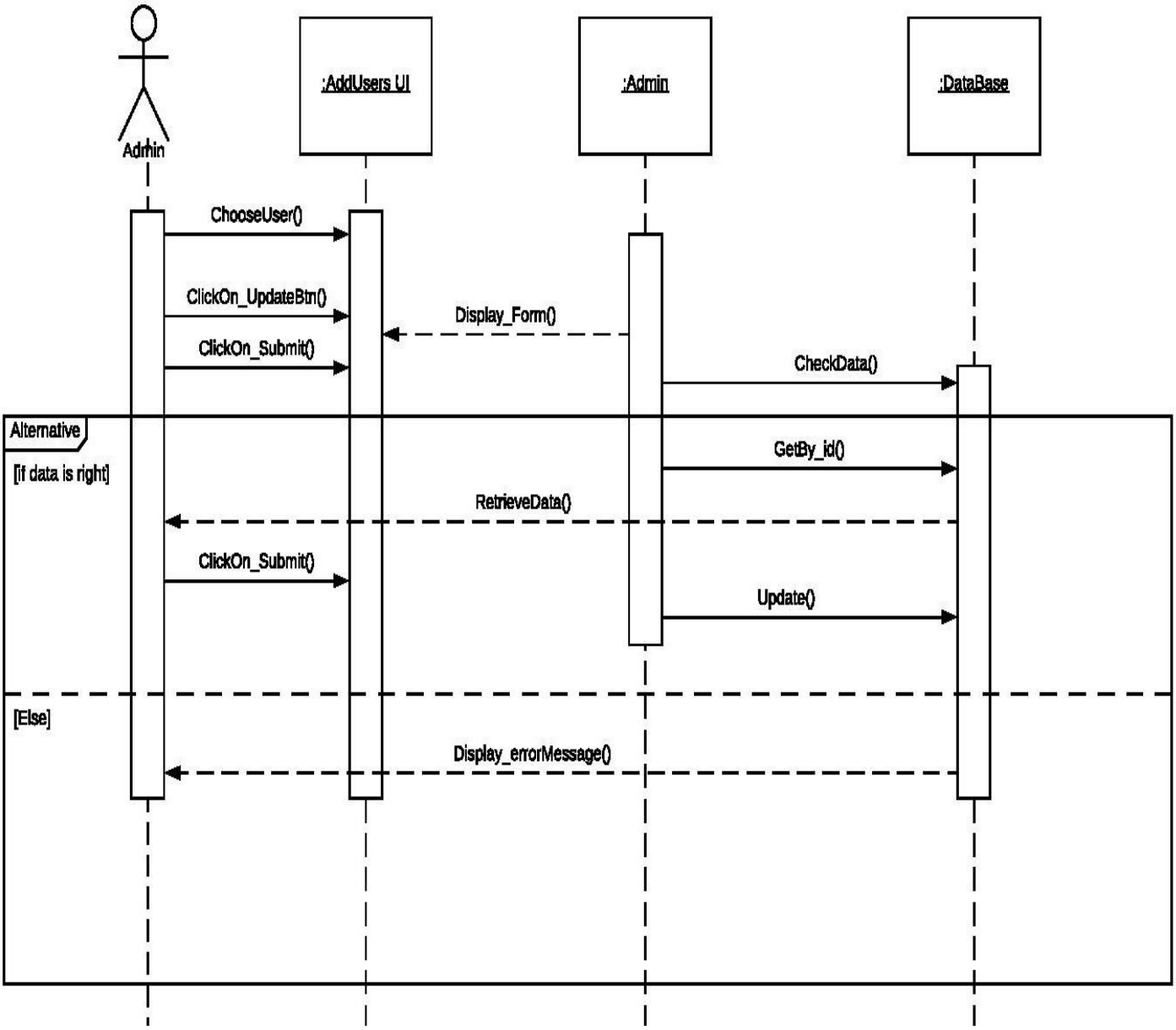


F) Sequence Diagrams:

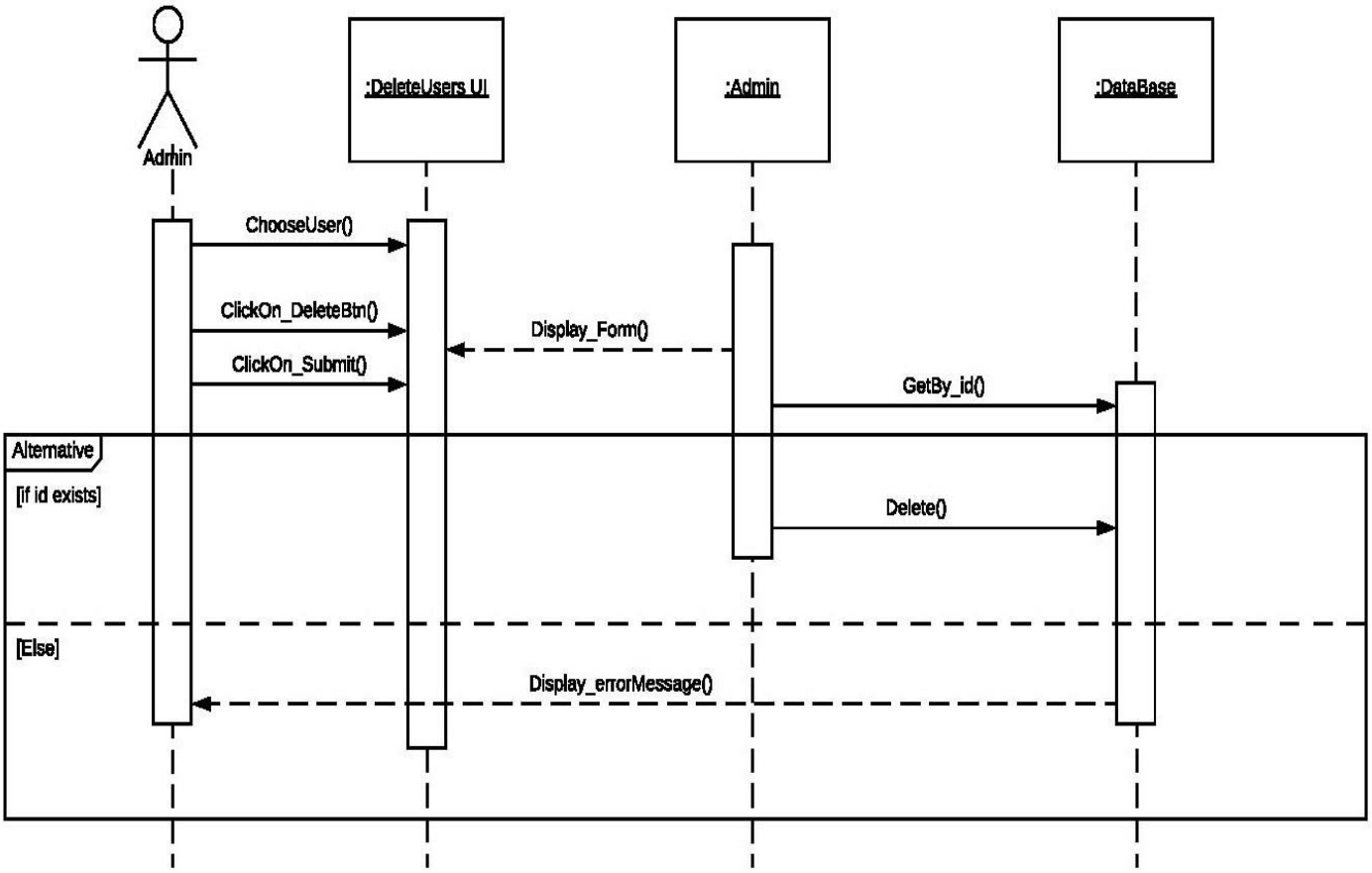
- **Add Users**



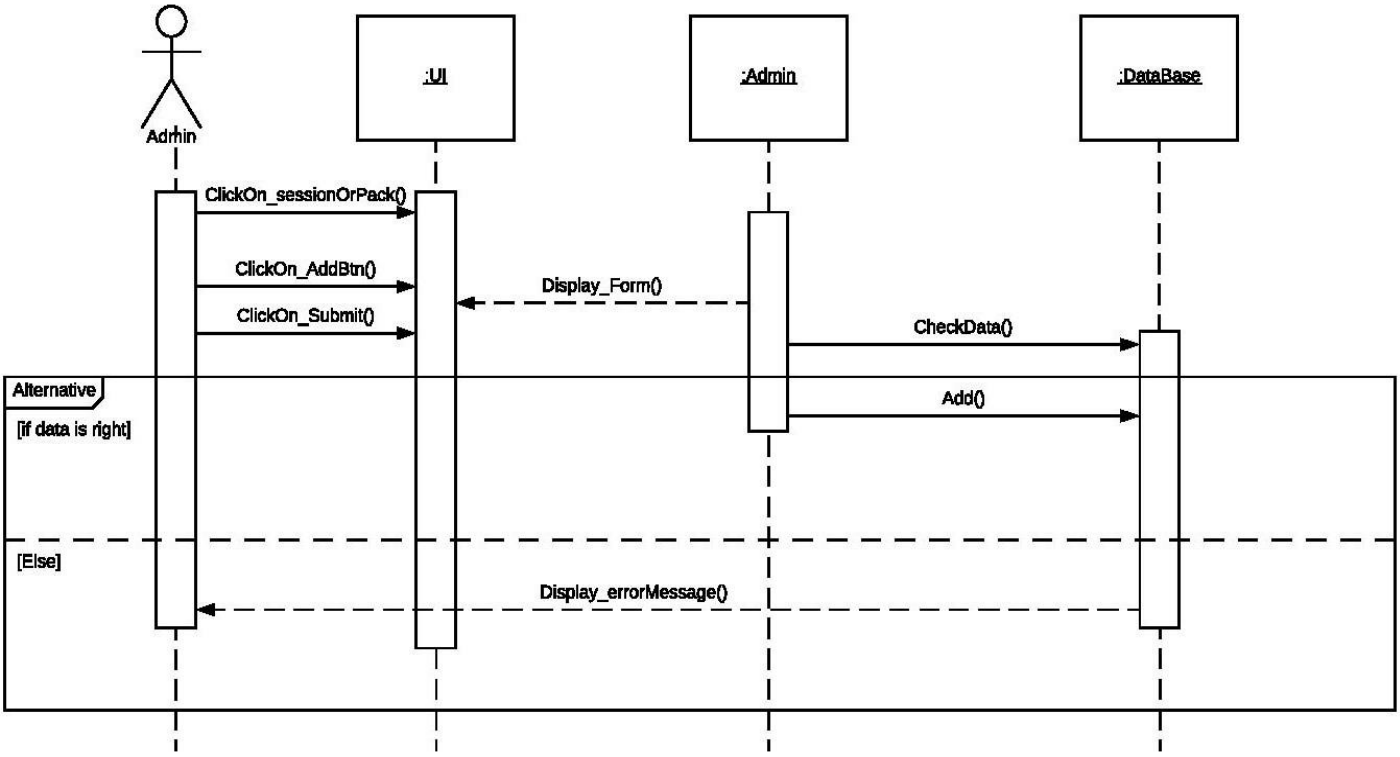
- **Update Users**



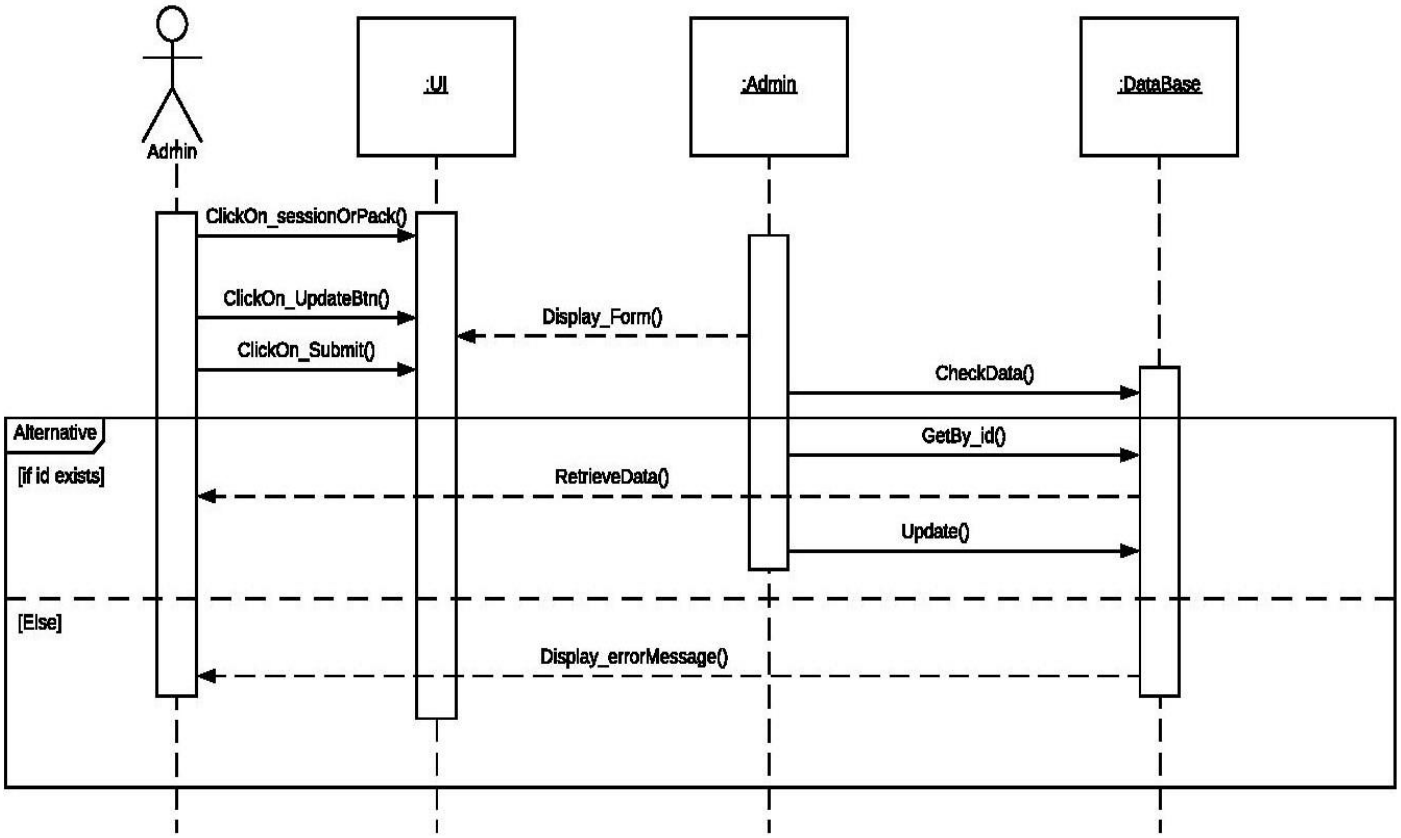
- **Delete Users**



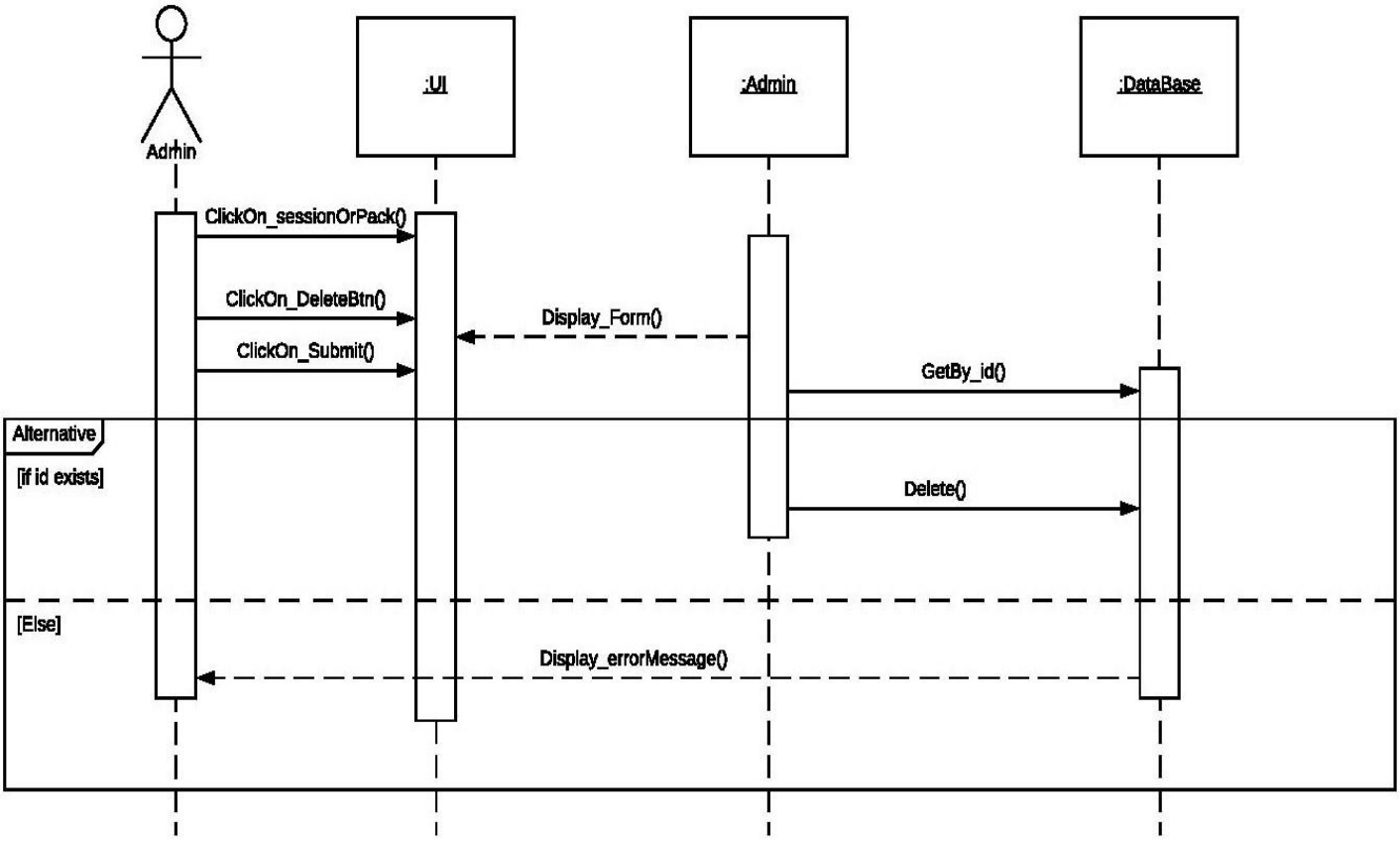
- **Add Packages & Sessions**



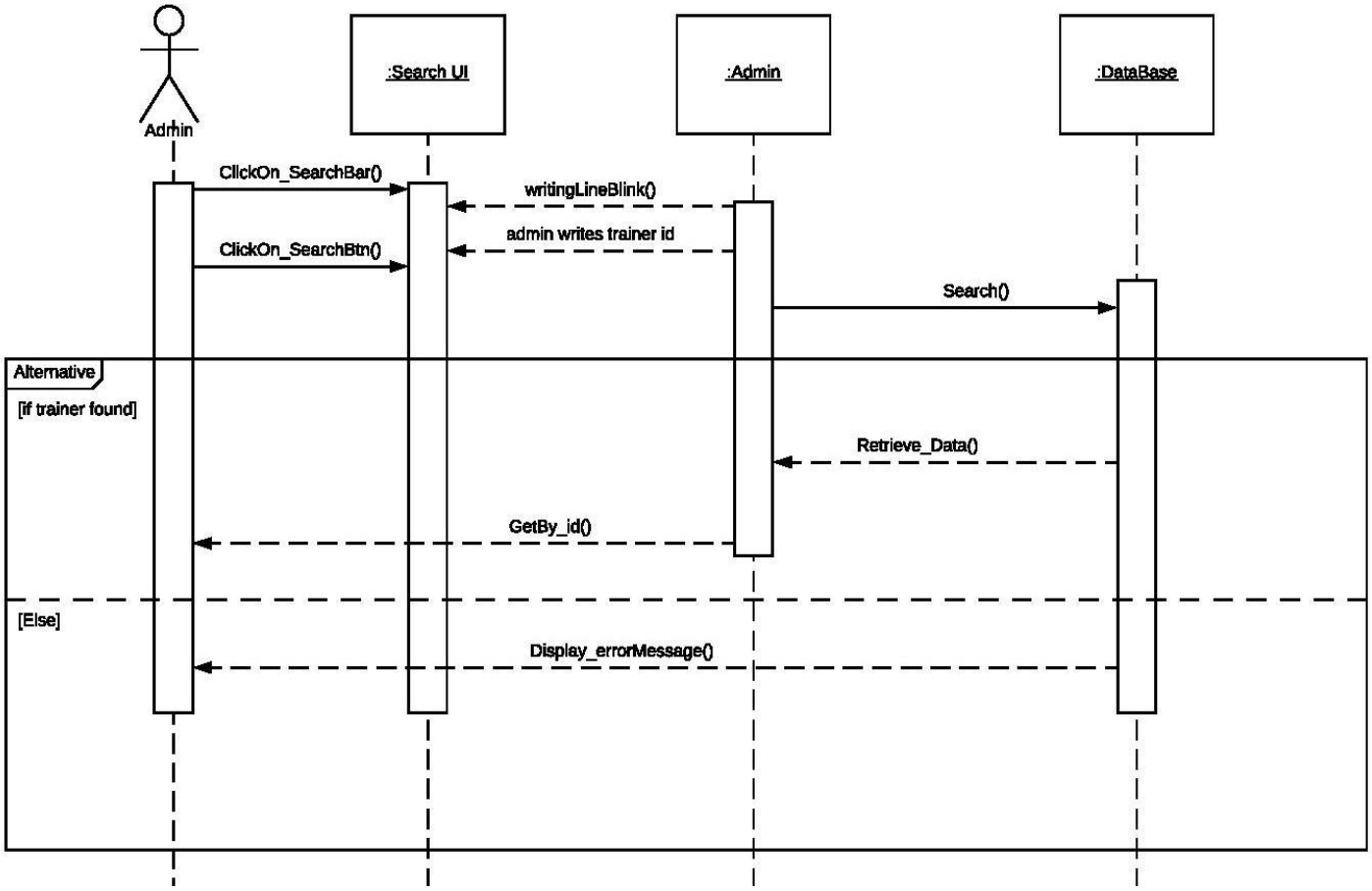
- **Update Packages & Sessions**



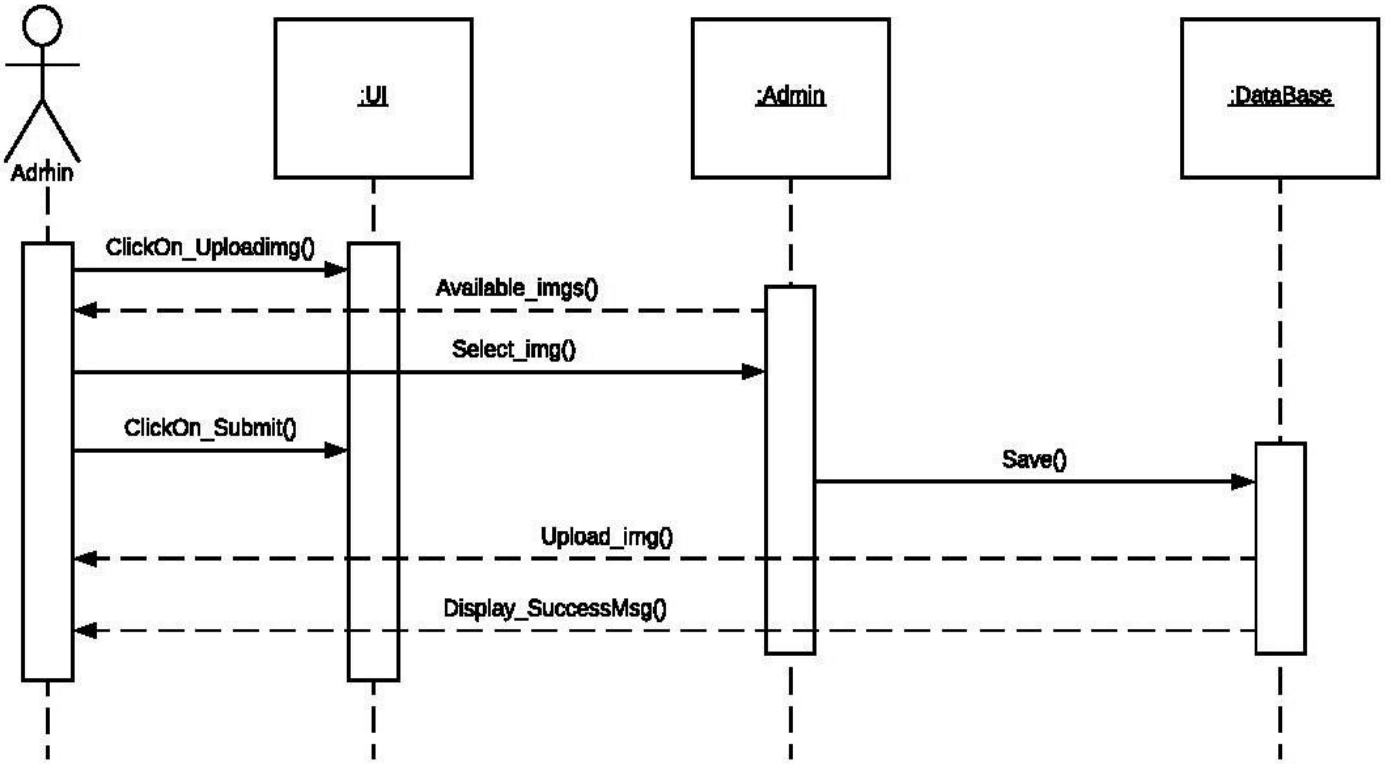
- **Delete Packages & Sessions**



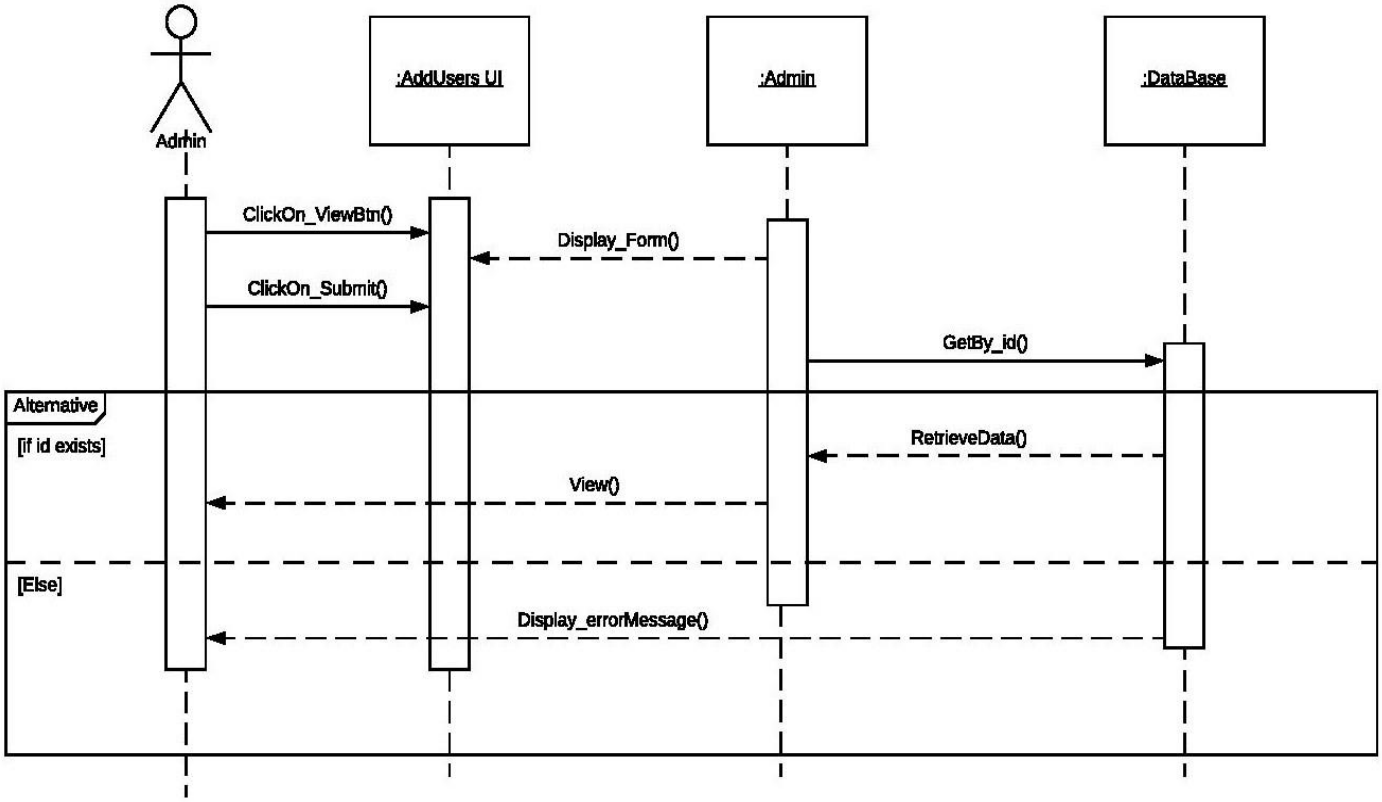
- **Search**



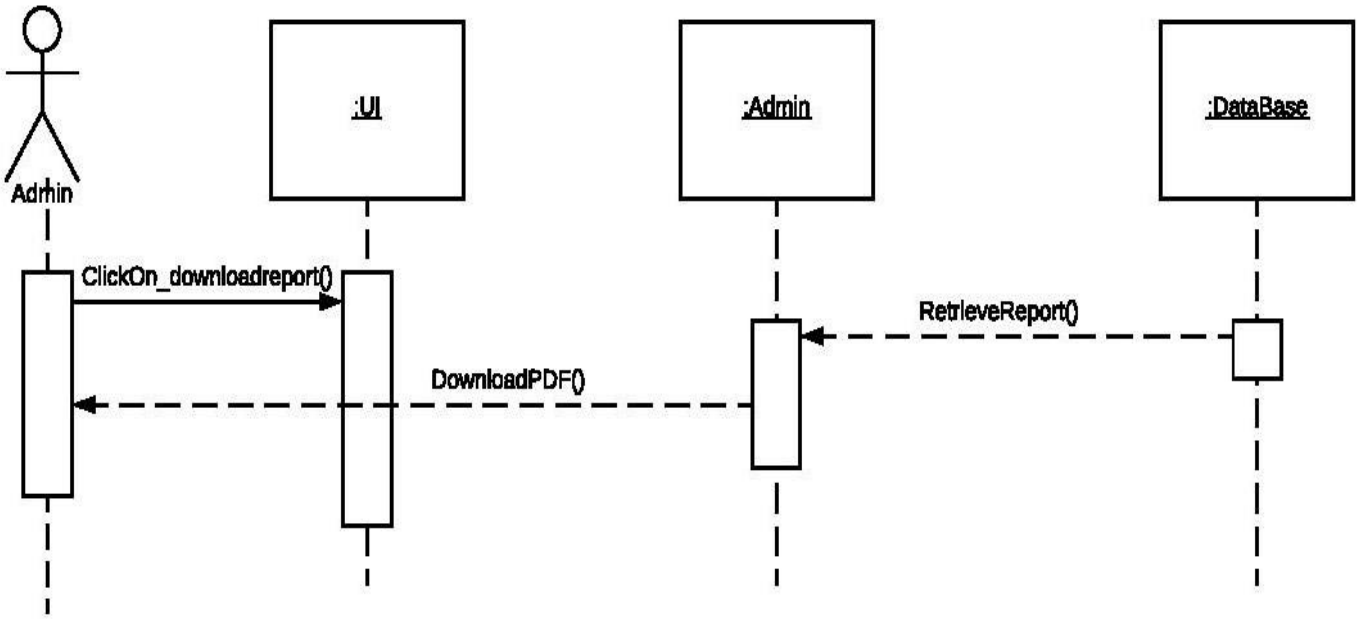
- **Upload Images**



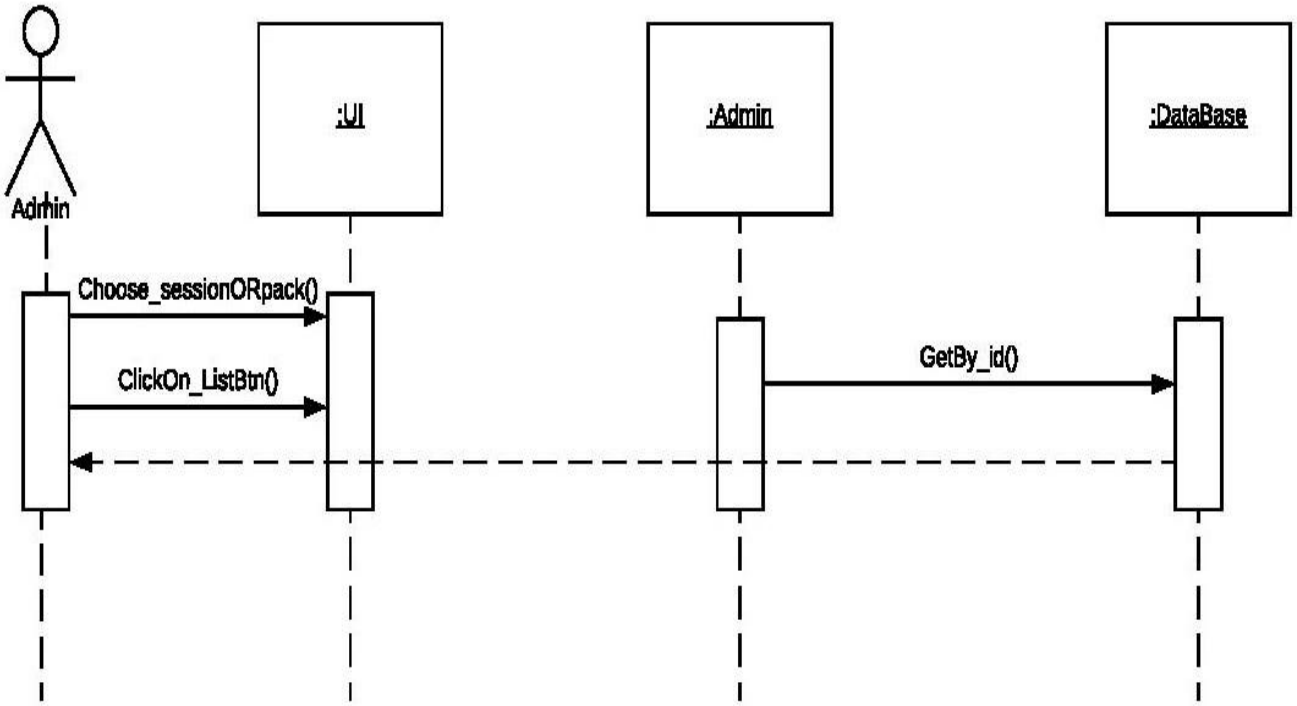
- **View**



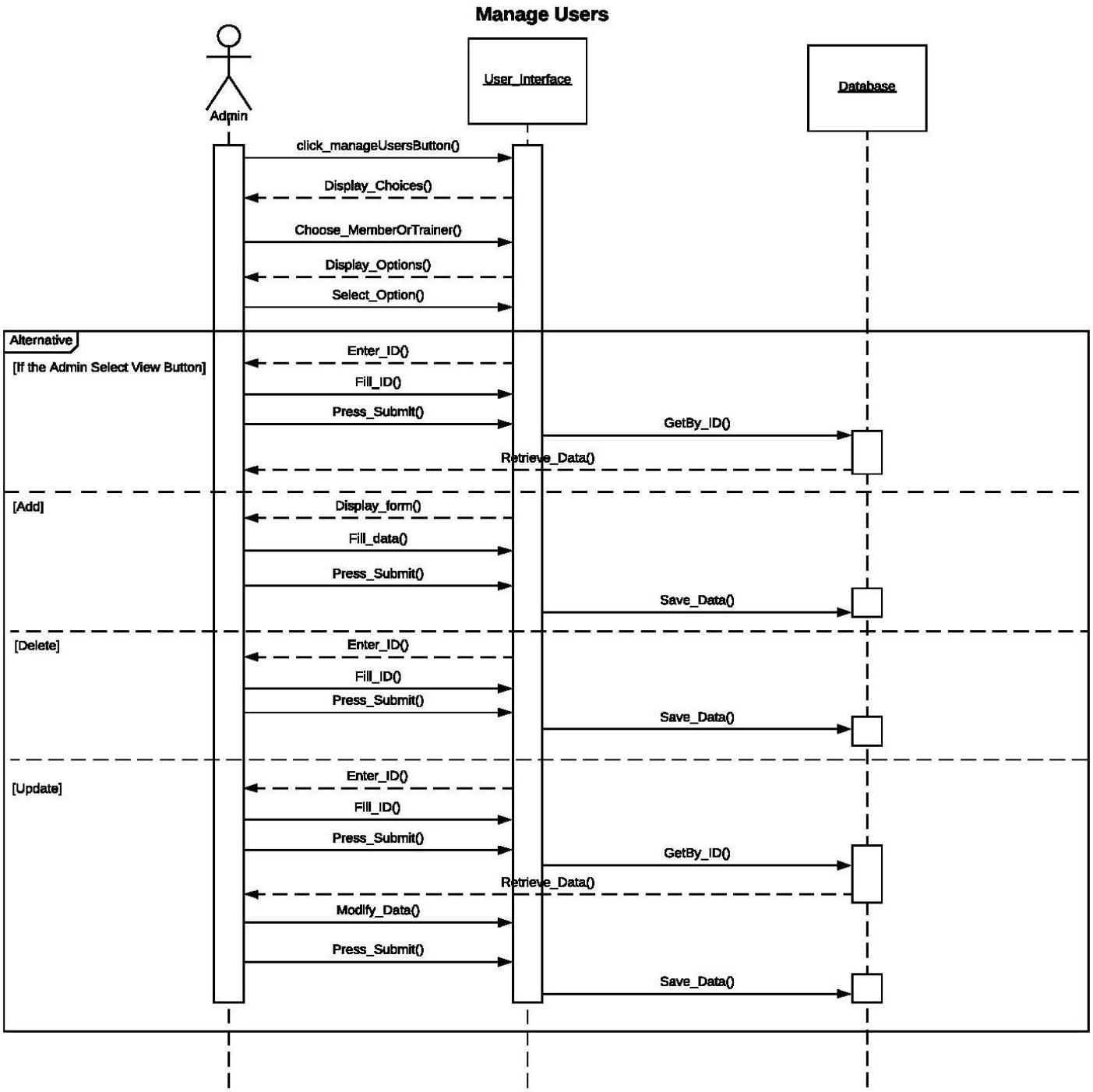
- **Download Report**



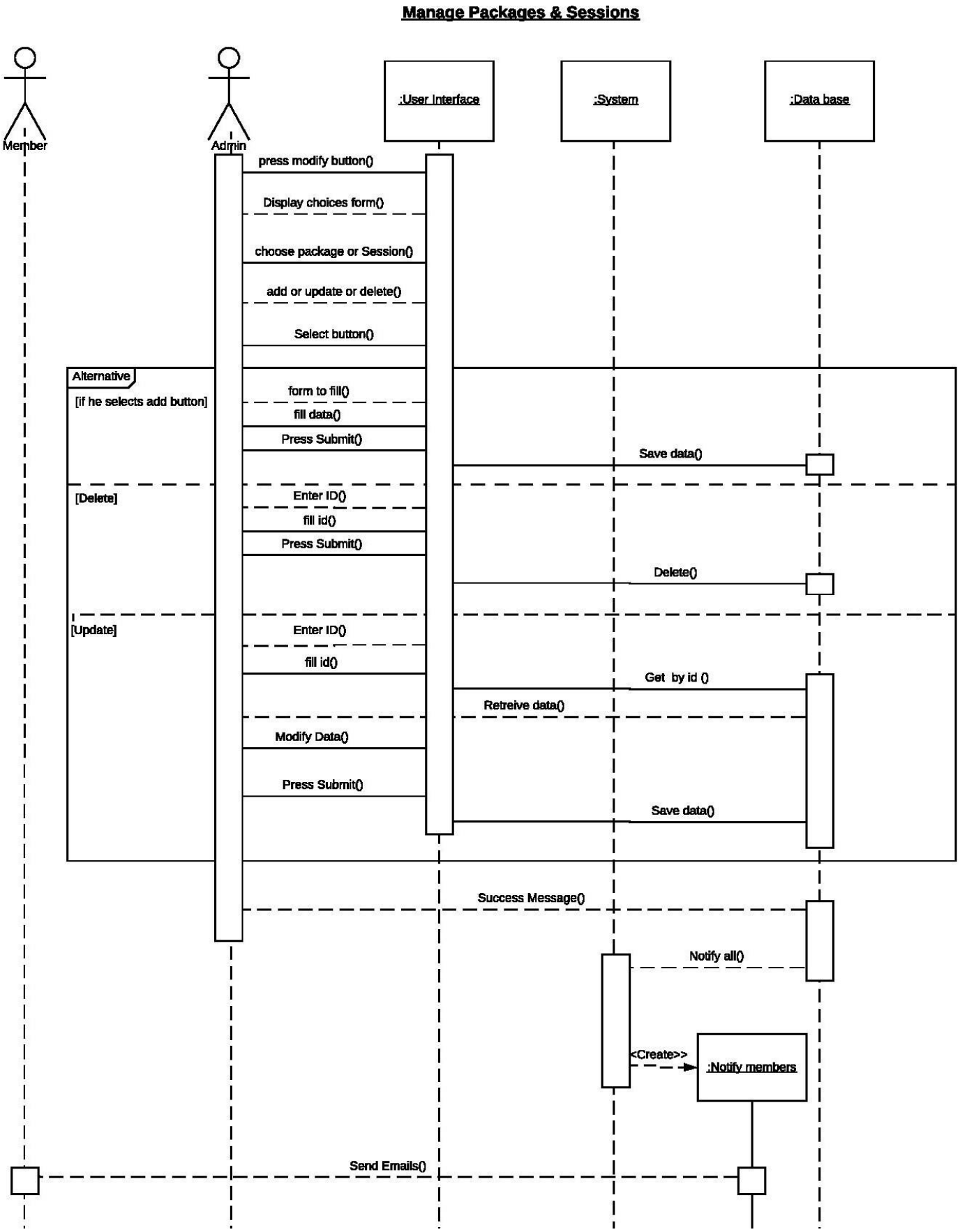
- **List sessions or packages**



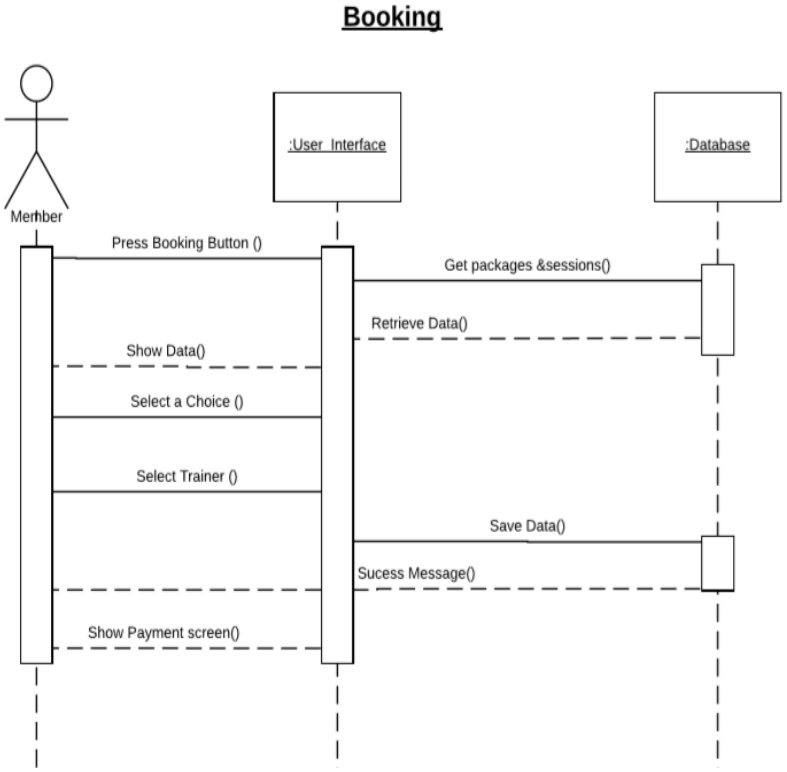
- **Manage Users**



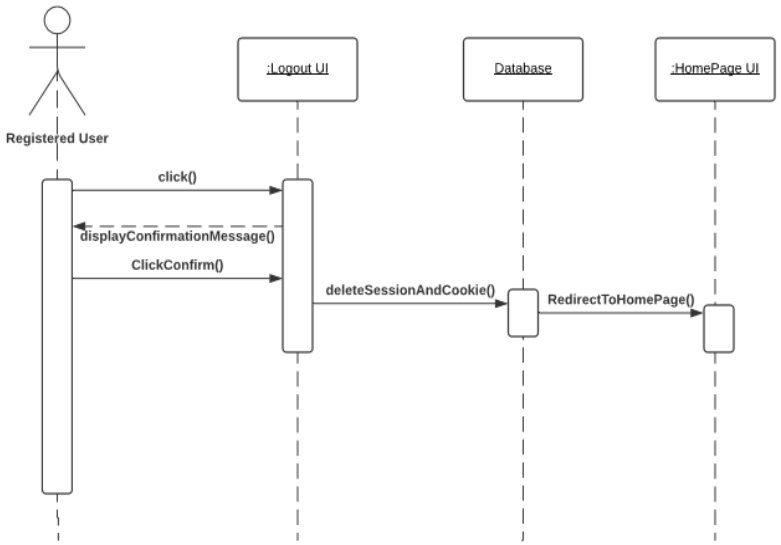
- **Manage Packages & Sessions**



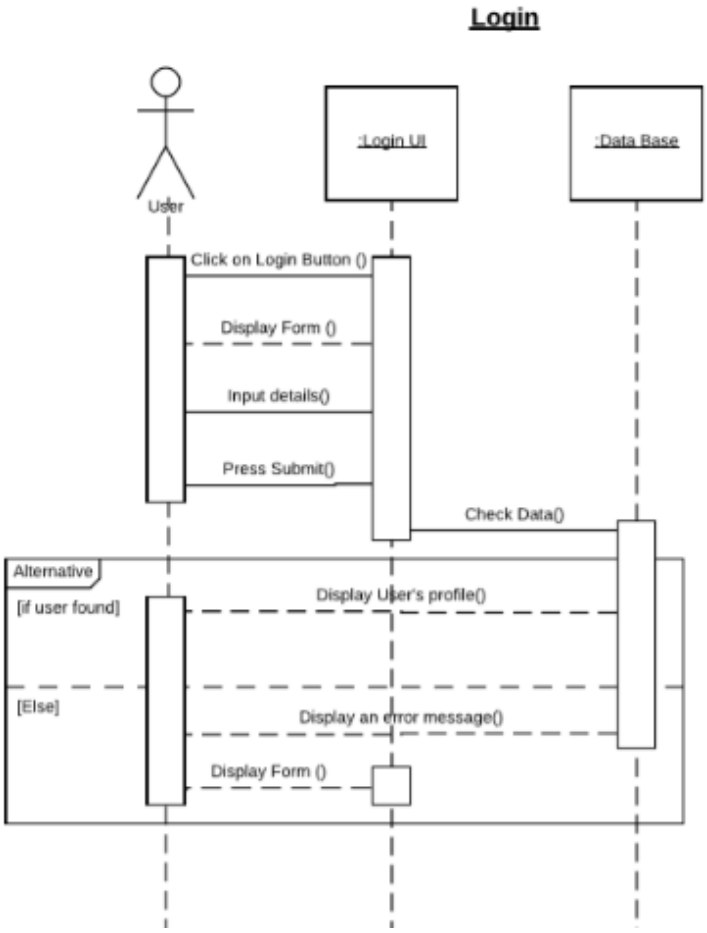
- **Booking**



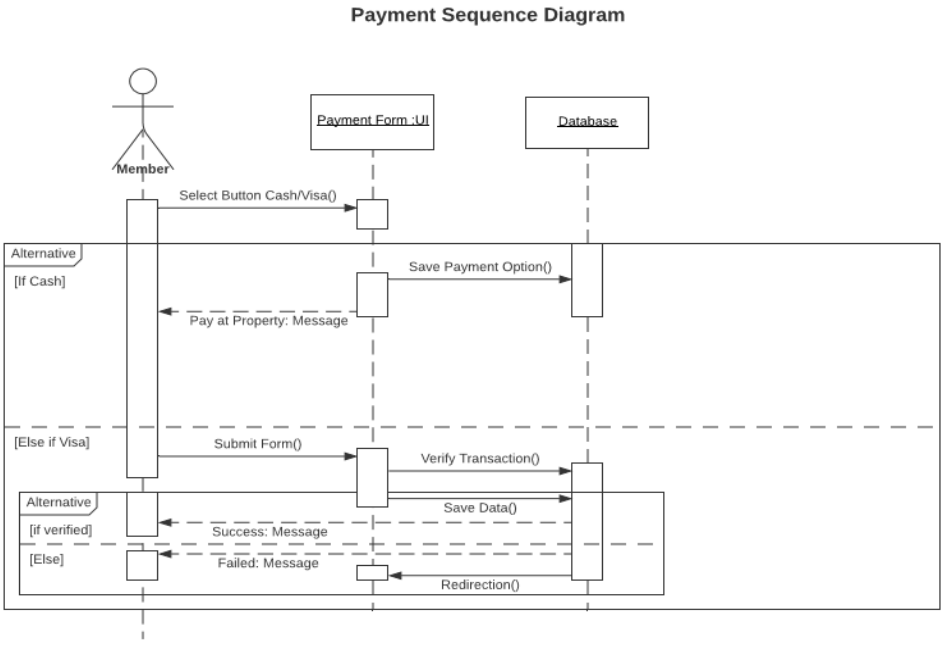
- **Logout**



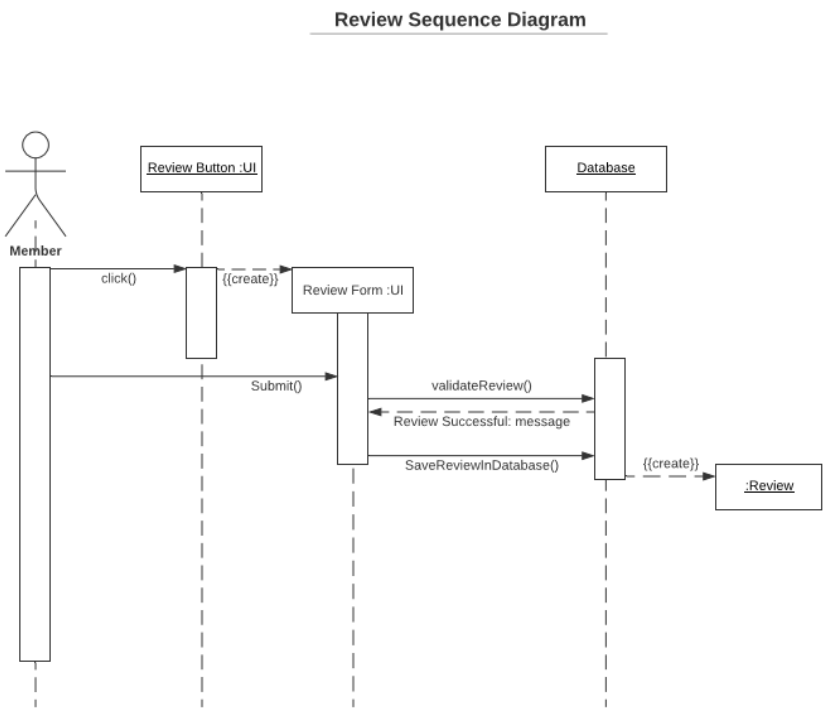
- **Login**



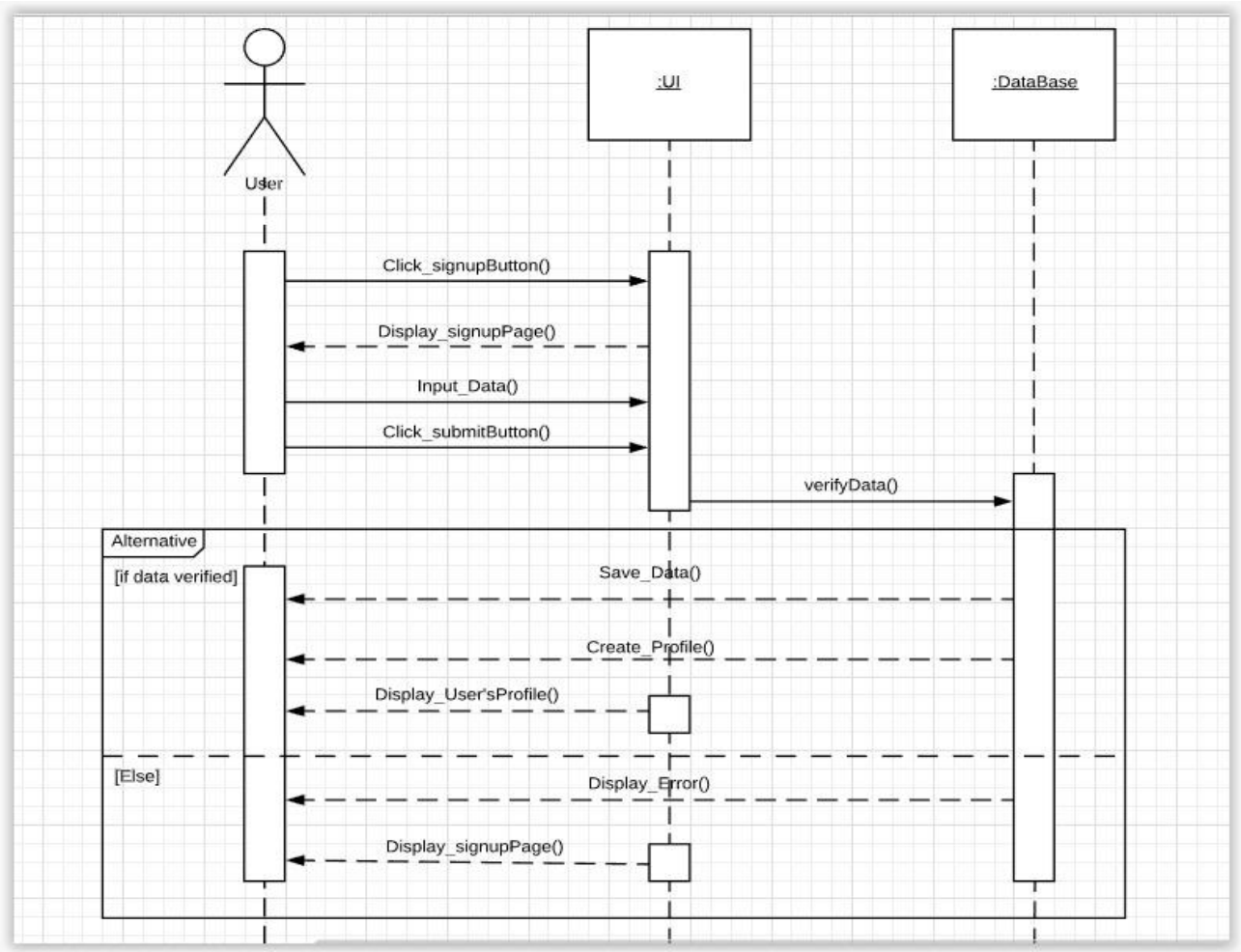
- **Payment**



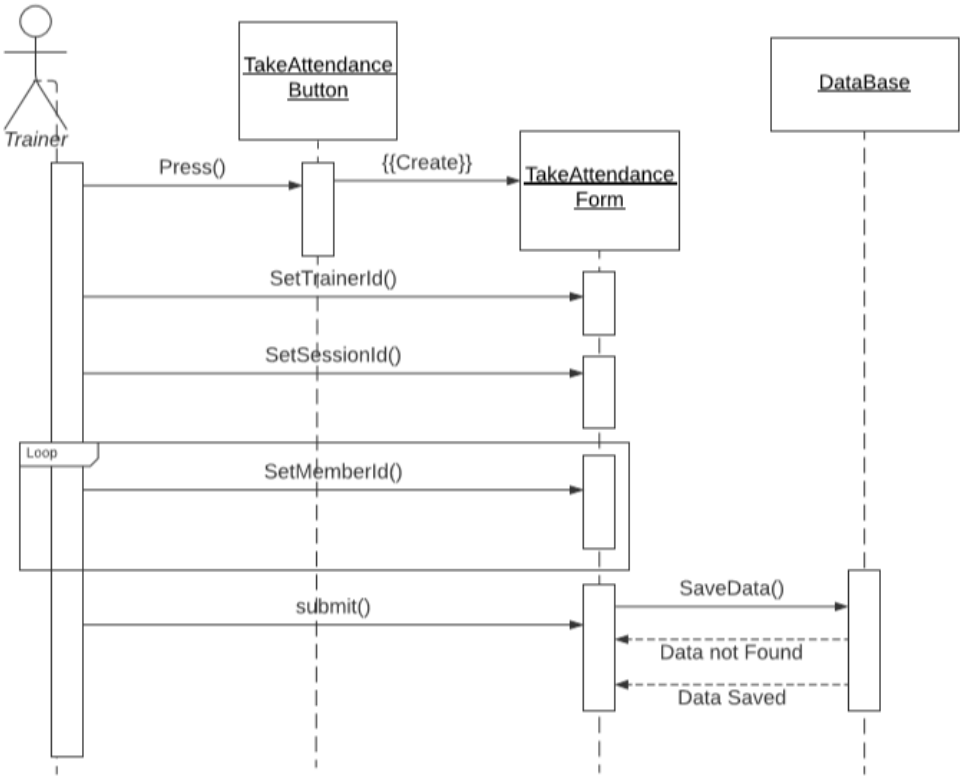
- **Payment**



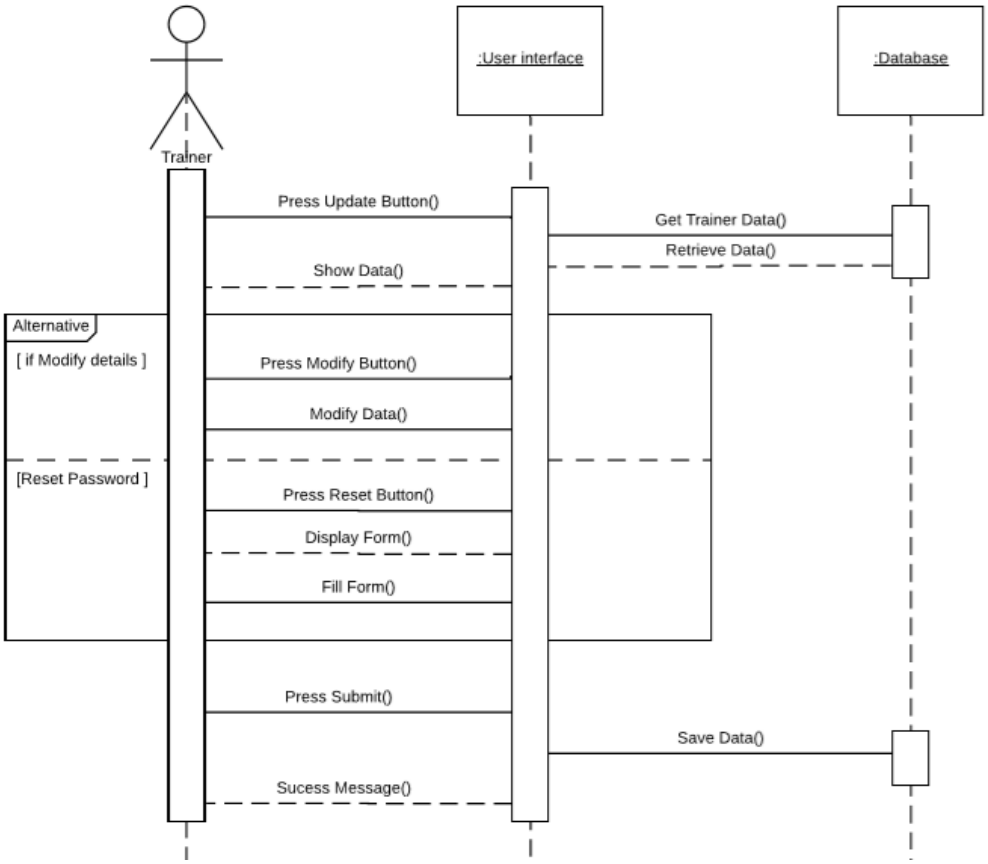
- **Sign up**



- **Take Attendance**

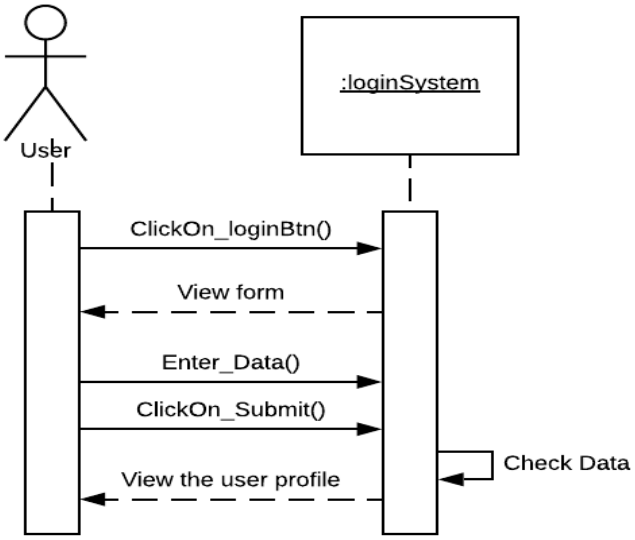


- **Update Profile**

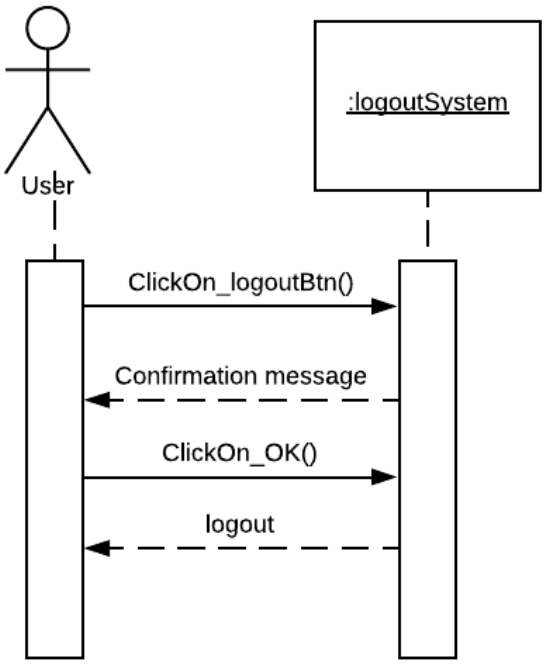


g) System Sequence Diagrams (SSDs)

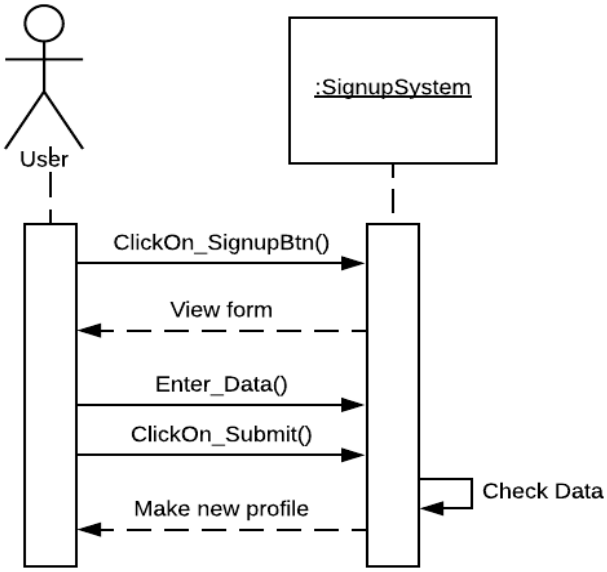
1) Login



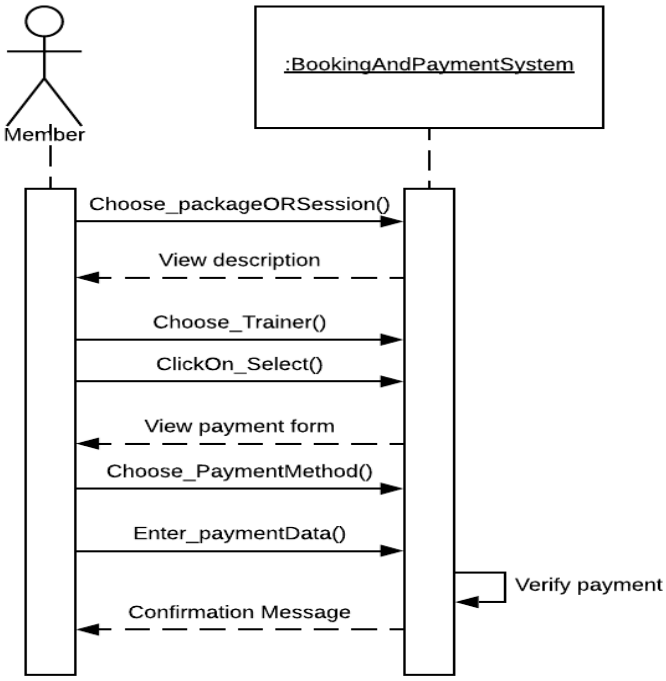
2) Logout



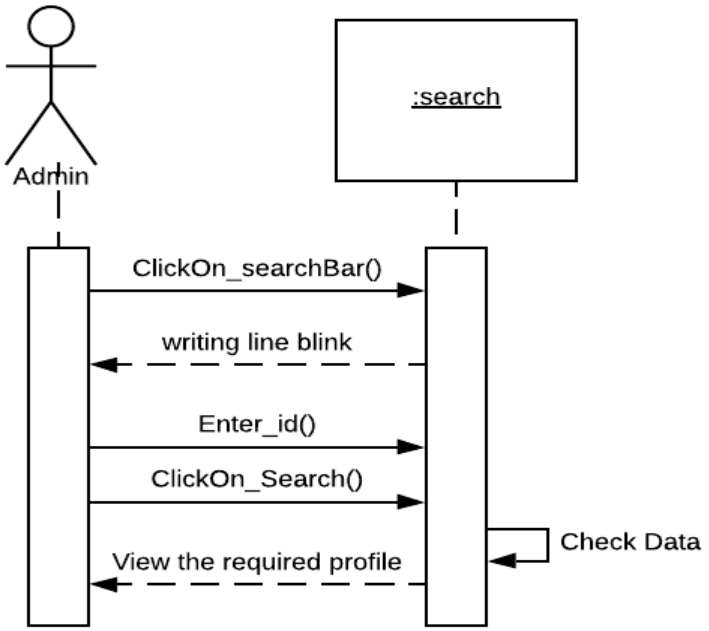
3) Signup



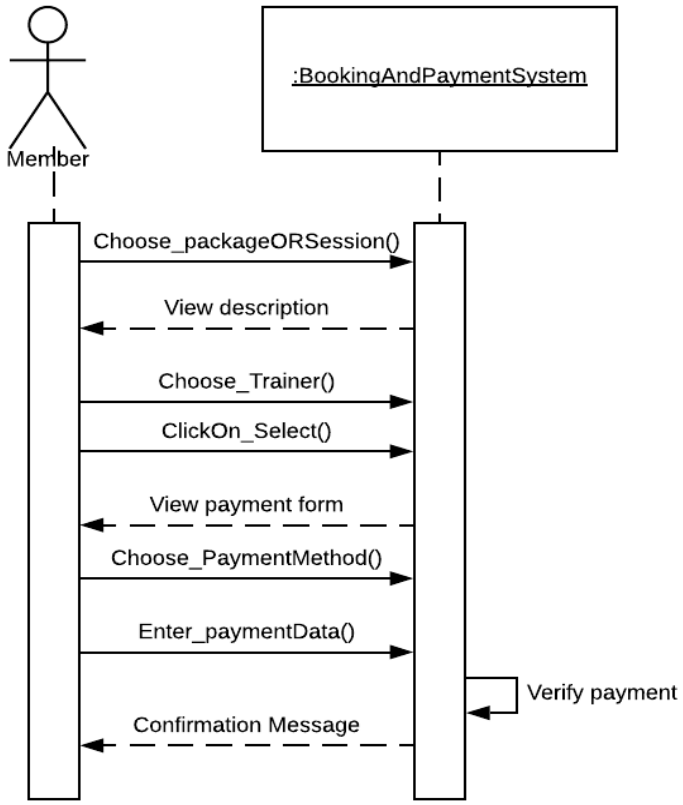
4) Booking & Payment



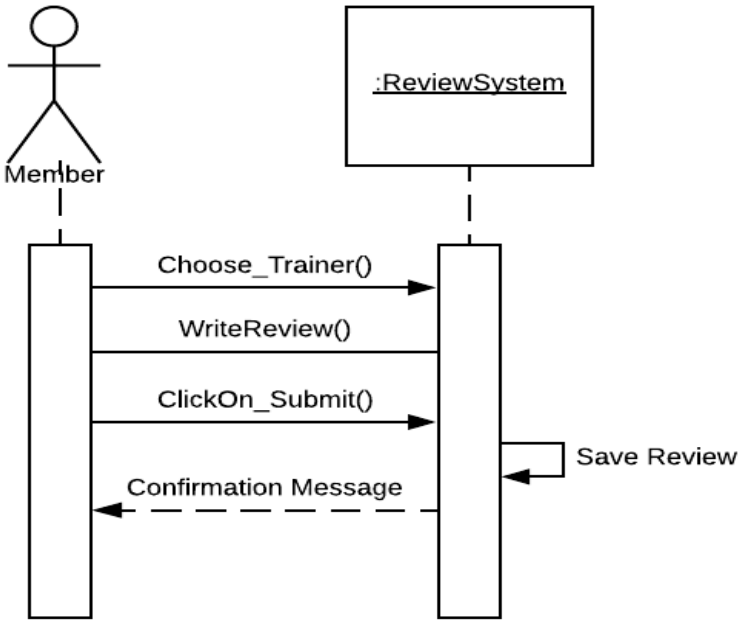
5) Search



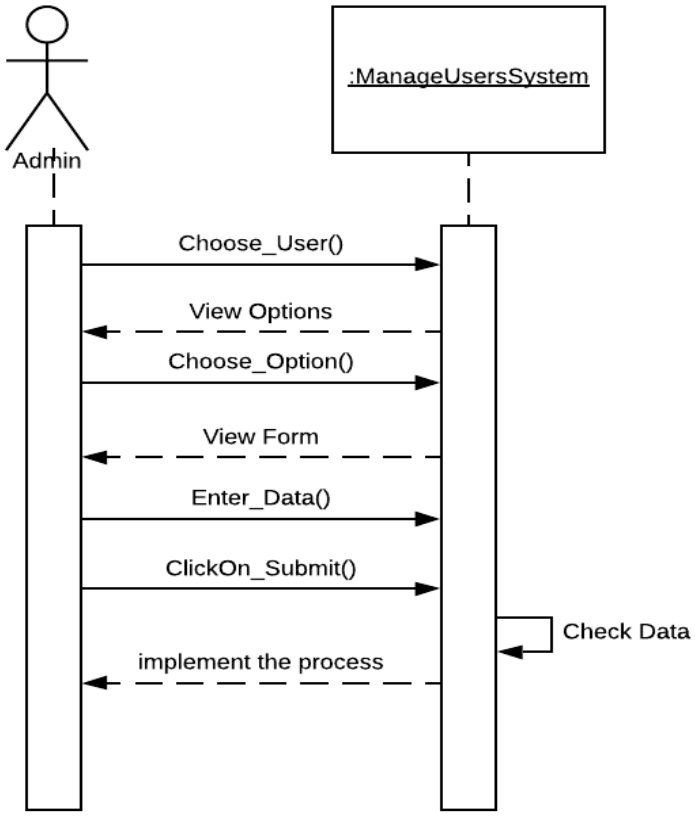
6) Booking & Payment



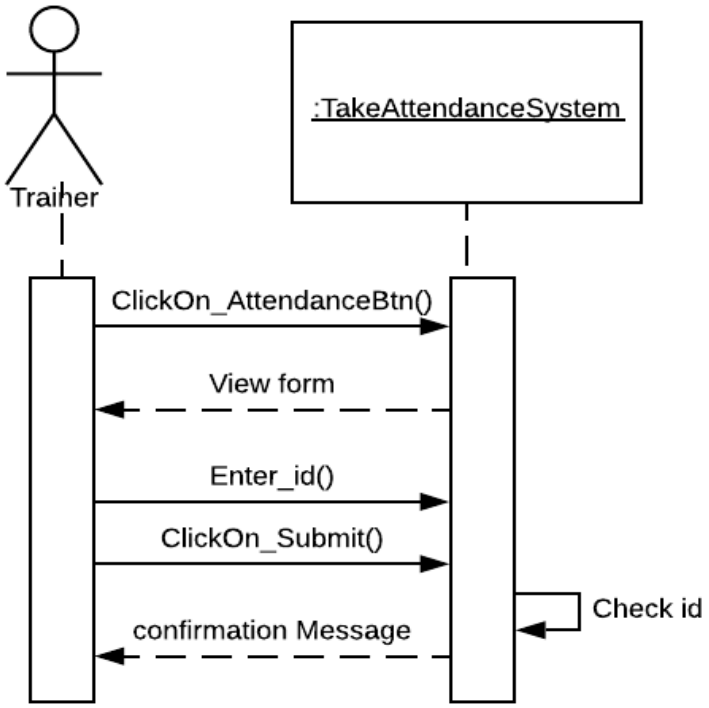
7) Review



8) Manage Users

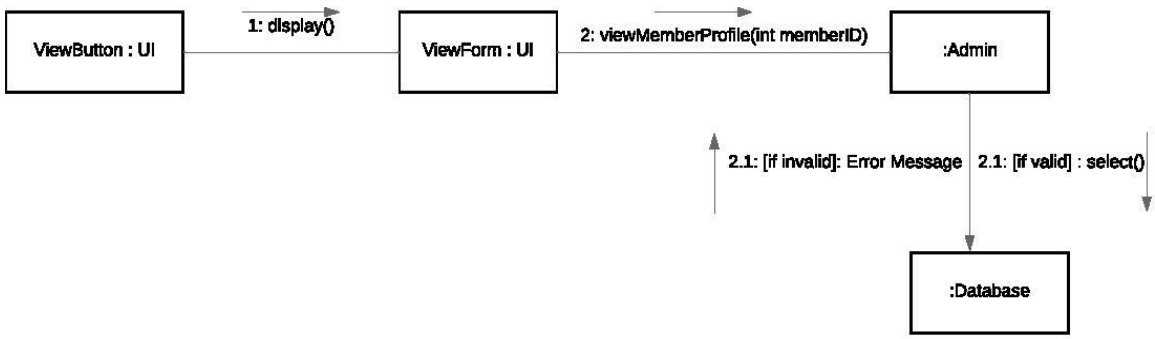


9) Take Attendance

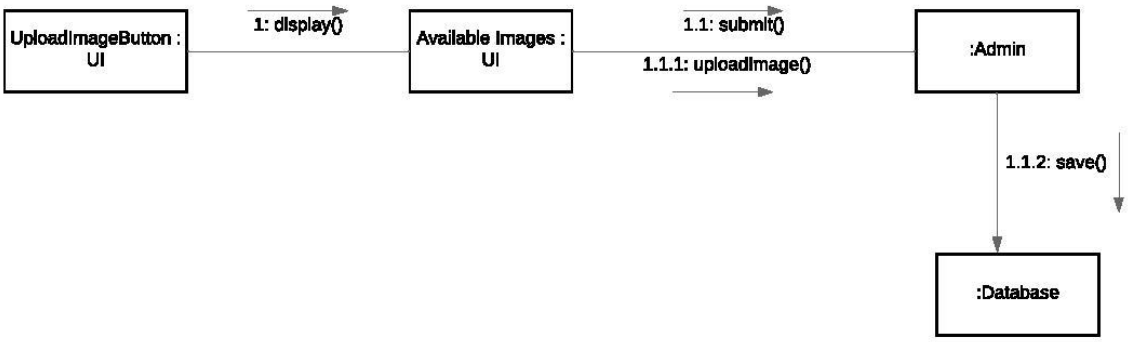


h) Collaboration/Communication Diagram(s)

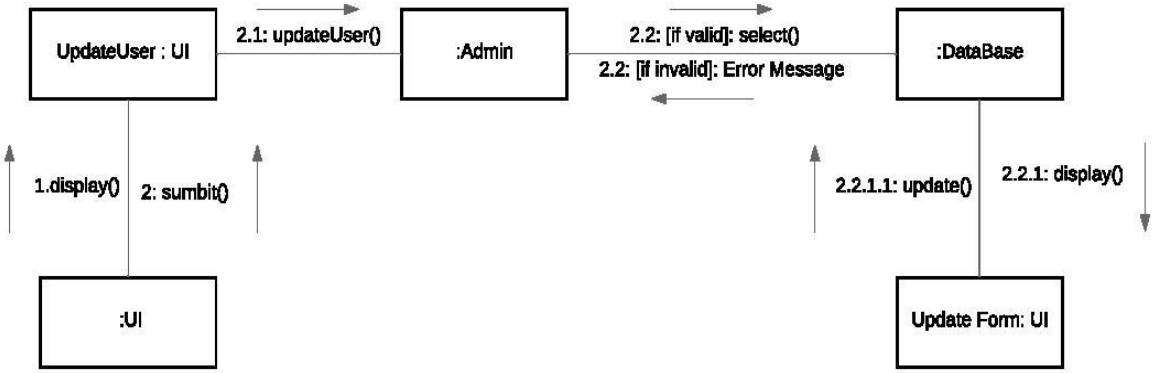
View Profile Collaboration Diagram



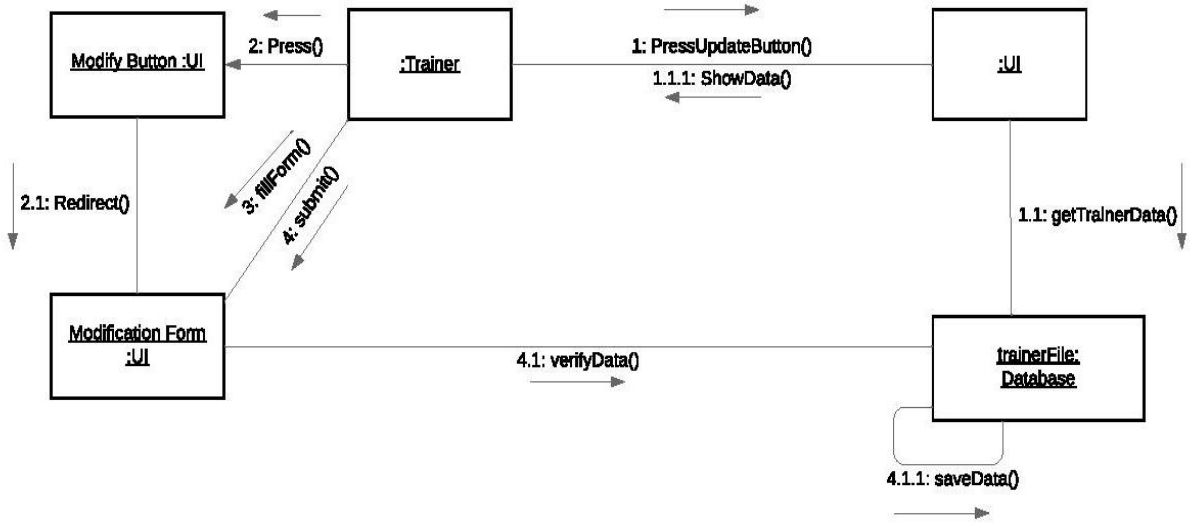
Upload Image Collaboration Diagram



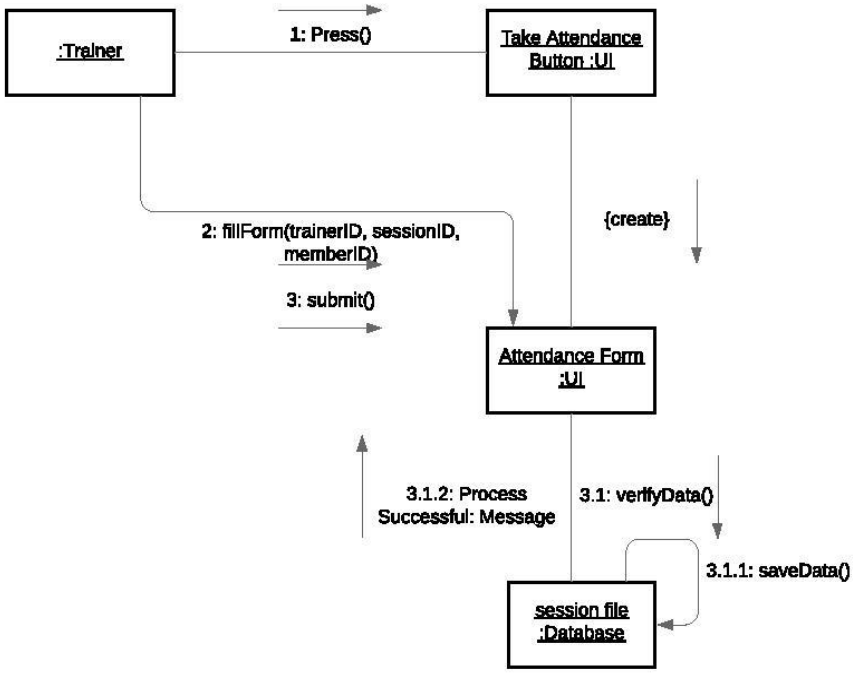
Update User Collaboration Diagram



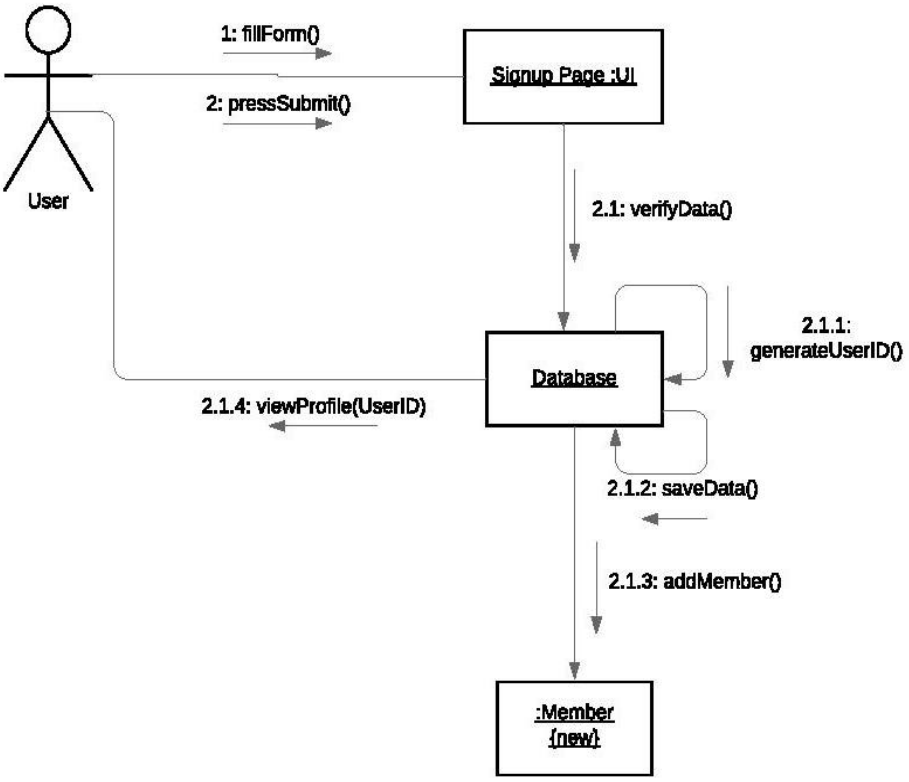
Update Profile Communication Diagram



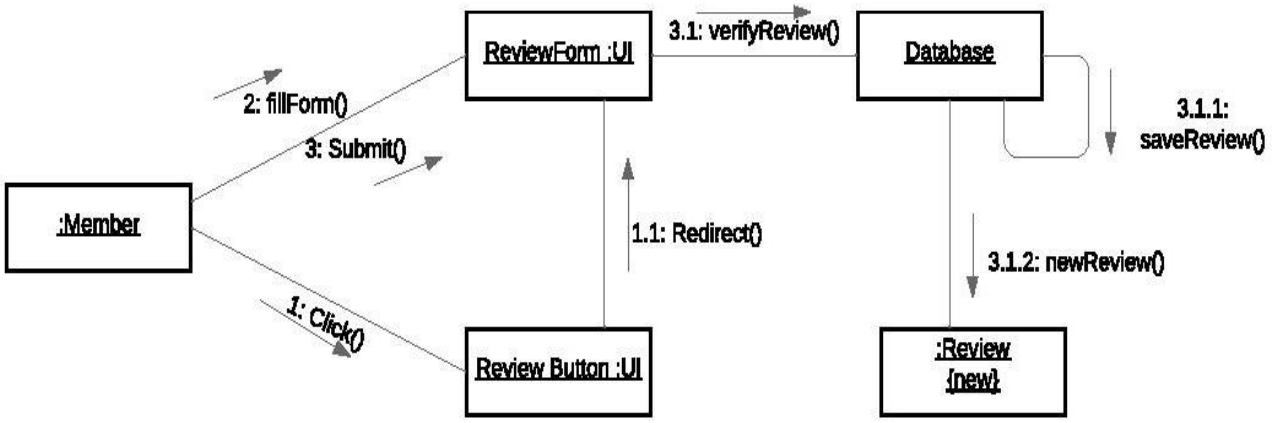
Take Attendance Communication Diagram



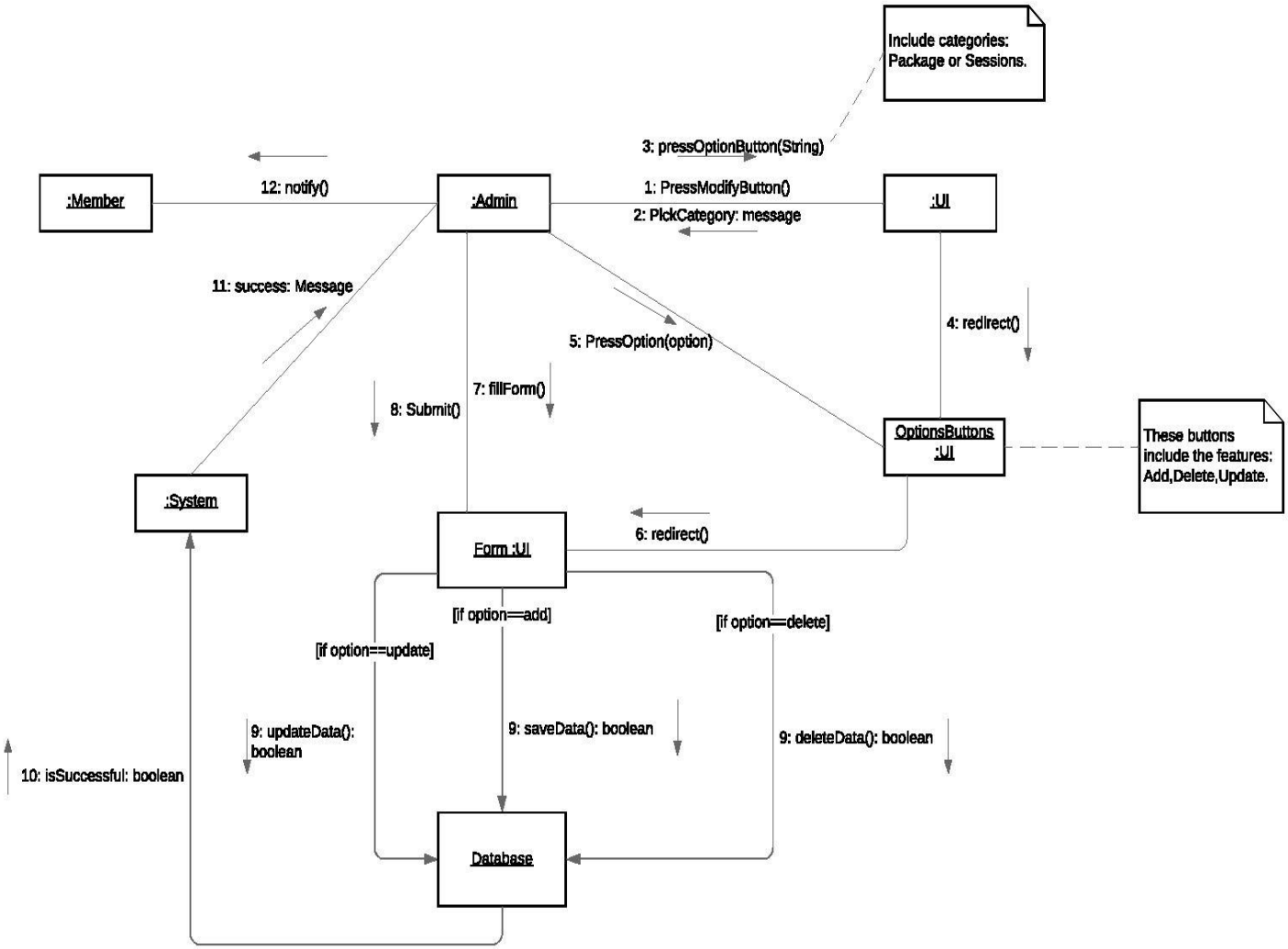
Signup Communication Diagram



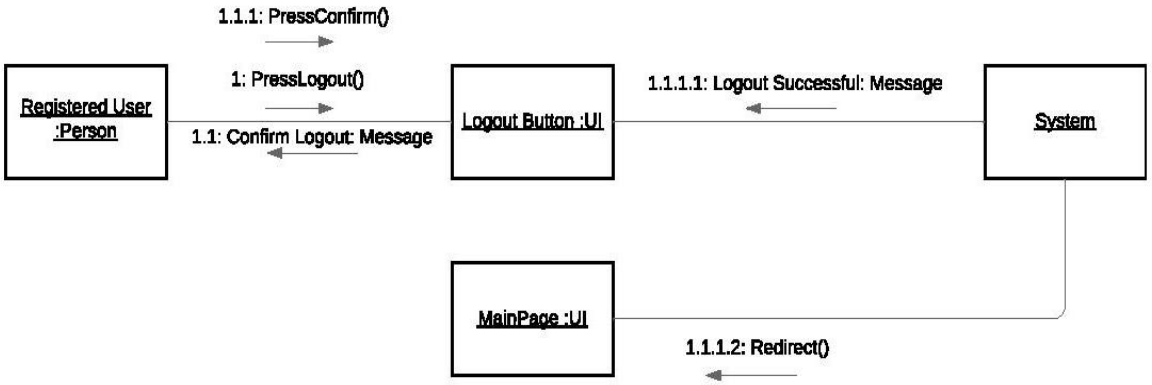
Review Communication Diagram



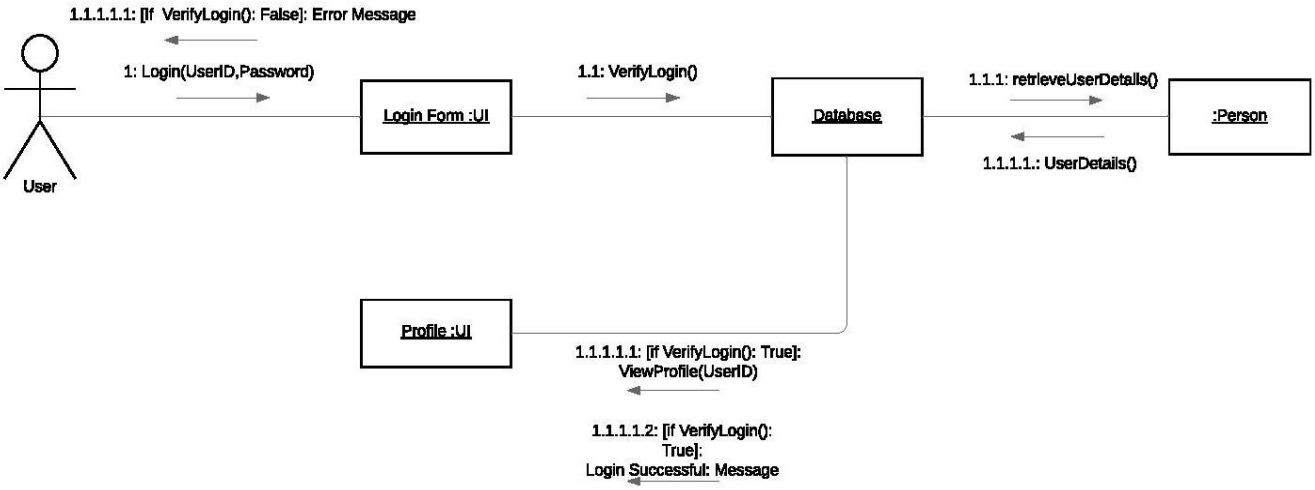
Manage Packages&Sessions Communication Diagram



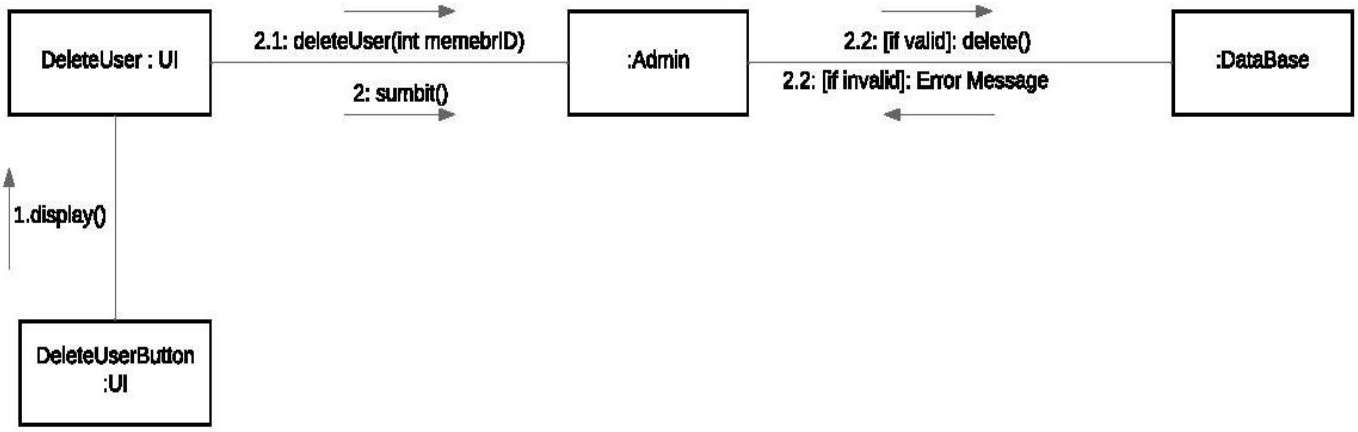
Logout Communication Diagram



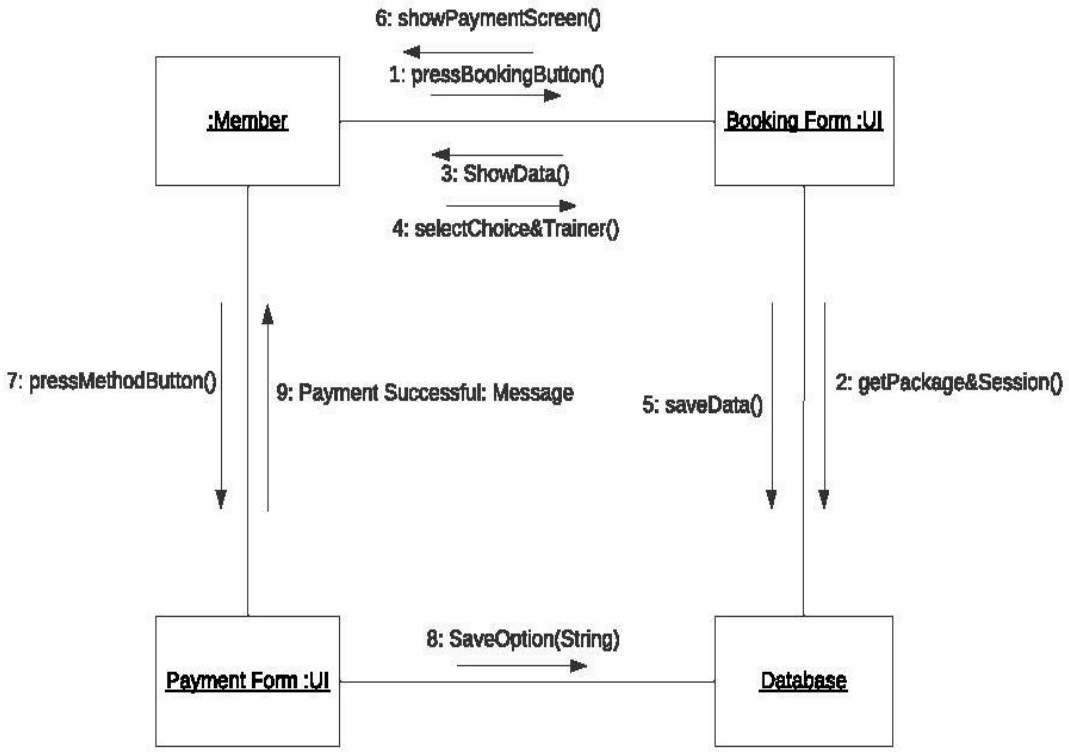
Login Communication Diagram



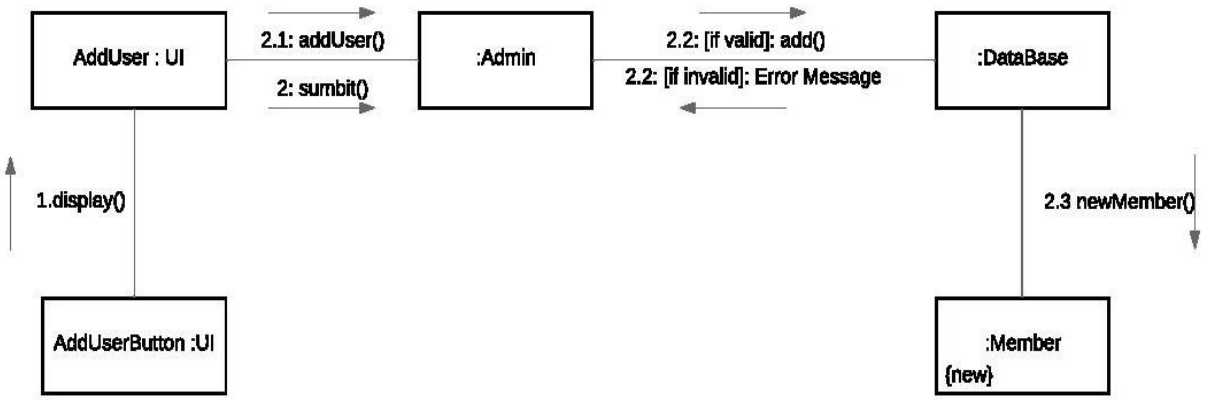
Delete User Collaboration Diagram



Booking&Payment Communication Diagram



Add User Collaboration Diagram



I) Which strategy (or strategies) did you use to implement the use-cases: One Central Class, Actor Class, or Use-Case class. Please explain your choice, and its potential advantages/disadvantages for your design.

Use Case Strategies

It is one of the designs issues that can face programmers when designing a Portable and applicable Software. There are 3 Strategies for implementing the use Cases and their corresponding Sequence and Collaboration Diagrams. These Strategies objectives is to know which object of a class will be used to receive the message coming from the user interface and handle these messages with the rest of the classes and their objects. And Start the sequence of message from the moment the user interacted with the system to the moment the system applies the desired interaction.

The Most useful Strategy To be implemented in that Fitness Club project is ***The Actor Class strategy****.*

This Strategy (just like the others) have diversity of advantages and also have disadvantages as well. It all depends on the System you are willing to implement and actually you can use a mixture of the available Strategies to fulfil the needs of the project.

The Reason of choosing this Specific Strategies is that it is more suitable and its disadvantages is more applicable than the other strategies. And can be solved in the perspective of this project.

<u>The Advantages of Actor Class Strategy</u>

1)It is clear from the designer's perspective, as it actually grants with the implementation of class diagram as well as the interaction diagrams (Sequence, System Sequence &Collaboration)

<u>The Disadvantages of Actor Class Strategy</u>

1) The Limitation of the usability of the defined class and multiple complications when two actors start the same interaction, as this is a possibility to have two actors having the same action as login or logout activity. So therefore, it would result Repetition in code in each Actor defined class.

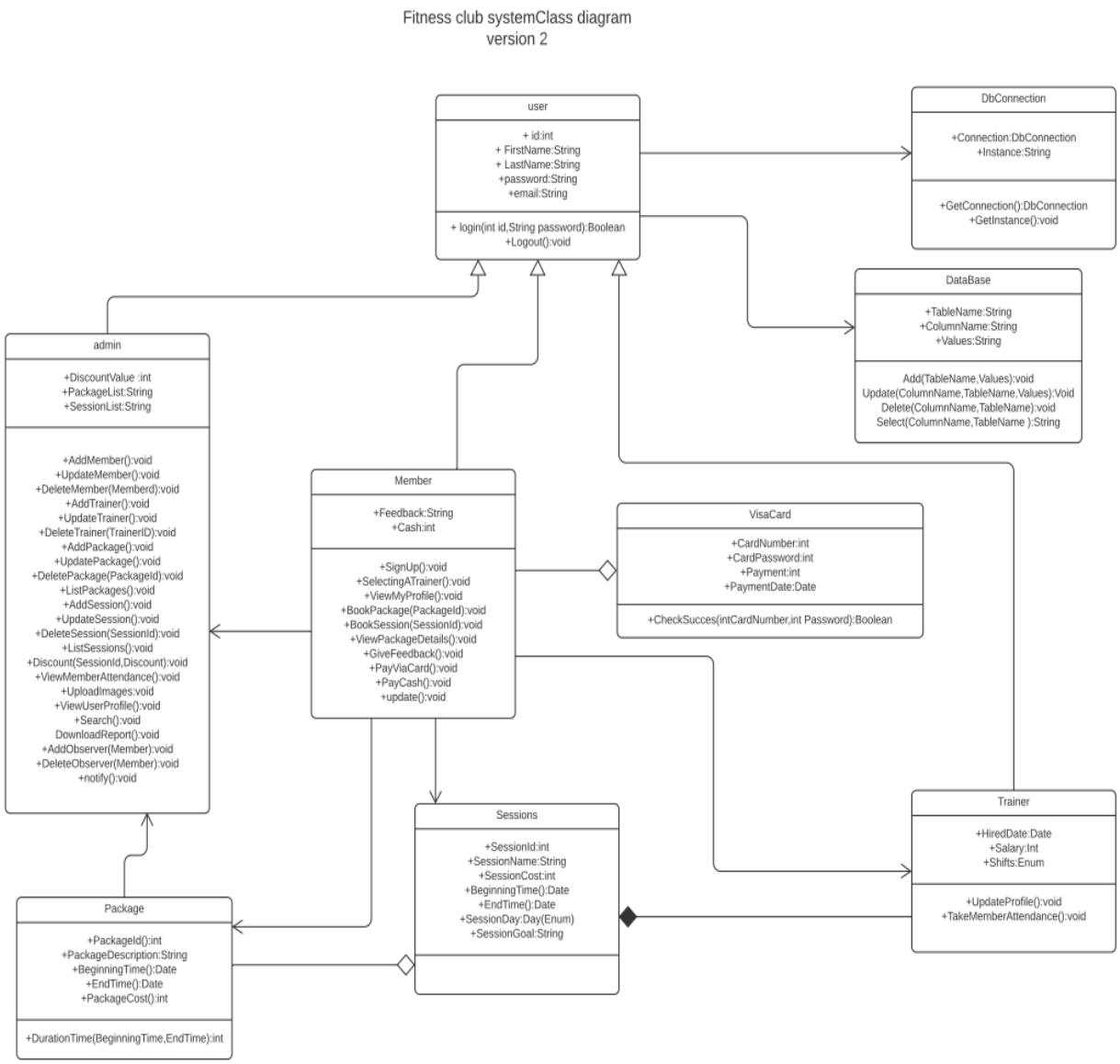
2)Difficulty to trace the classes, where as you will have an independent class in the class diagram with the suitable methods for each actor presence which interacts with this software.

<u>The Optimal solution of this Disadvantages in Fitness Club Project</u>

1)To overcome the 1st disadvantages of multiple users initiating the same activity, the project has a class ‘user’ for the common activity that all the actors make and to reduce the Repetition of code as well.

2)To overcome the 2nd disadvantage is that we classified our projects to three main actors to avoid the possibility of multiple actors and their corresponding class in the class diagram.

j) Class Diagram 2 : The Intermediate version



K) Design Patterns Applied with the Description

We applied 4 design patterns: the Observer, the Singleton, the Strategy and the Delegation.

1- The Observer Design Pattern Description

Classification Category: (Behavioral Patterns)

They describe interactions between objects and focus on developing patterns to improve how objects communicate with each other.

Context: Defines one-to-many dependency between a set of objects such that when any changes happen to a certain object, an automatic mechanism is created to notify and update all of its observers.

Problem:
• A one-to-many dependency between objects should be defined without making the objects tightly coupled.
• Should be ensured that when one object changes, all of its dependent objects are updated automatically.
• The Subject object should be able to notify all of its dependent objects if needed.
Forces:
• Coupling: Tightly coupled objects can be hard to implement in some scenarios, and reuse because they refer to and know about many different objects with different interfaces.
Solution:
• Subject contains a list of observers to notify of any change in its state, so it provides methods using which observers can register and unregister themselves and a method to notify all the observers of any change.

• Observer has a method that will be used by Subject to notify them of any updates.

References: The Observer pattern has been around long before OO programming. Most often it was referred to using the term "callback", often implemented with function pointers in various languages. It is one of the twenty-three well-known "Gang of Four" design patterns that describe how to solve design problems to design flexible and reusable object-oriented software, that is, objects that are easier to implement, change, test, and reuse.

2- The Singleton Design Pattern Description

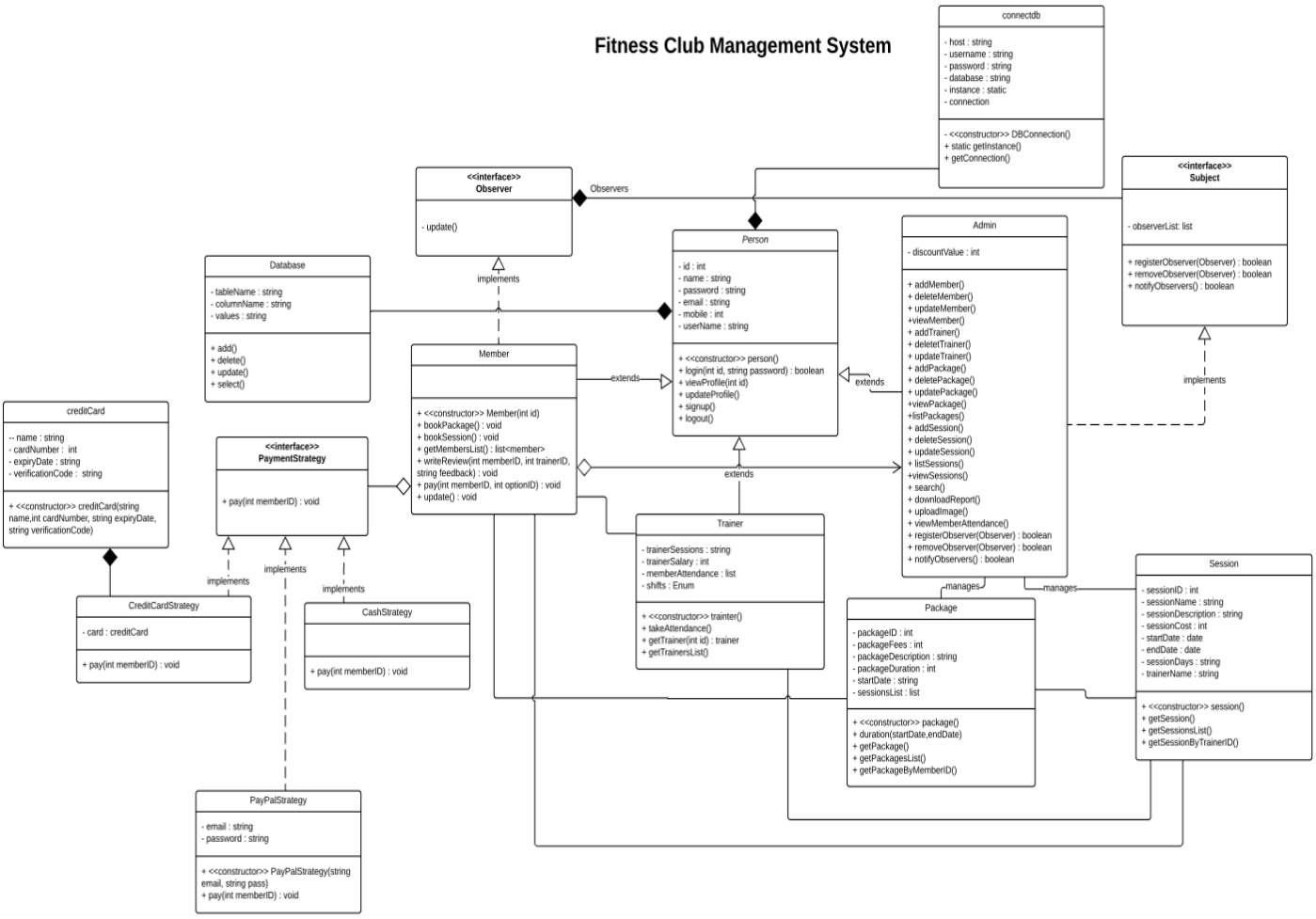
Classification Category: (Creational Patterns)
They are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation.

Context: Used when you need to ensure that you only have one instance of a certain class across the system and to provide a global point of access to it.

Problem:
• Should be ensured that the class has just a single instance (Whether to control access to some shared resource for example a database, or to do memory optimization).
• The sole instance of a class should be accessed easily, while also protecting it from being overwritten by any other class in the program.
Forces:
• Only one object: The Singleton pattern disables all other means of creating objects of a class except for the special creation method and only one object can exist, so we have to be absolutely sure of the number of instances we will need when we use Singleton.
• Saving memory to avoid memory leak.

Solution:
• By making the default constructor of the Singleton class private, to prevent other objects from using the new operator with the Singleton class.
• Creating a static method that acts as a constructor. This method calls the private constructor to create an object and saves it in a static field. So whenever that method is called, the same object is always returned.

L) Class Diagram 3 : The Final version



PART 3: Development Phase

→ Appendix:

https://github.com/ranabahaa/FitnessClub.git

12) Create and document front-end design for all functions

Pages:

- **Home Page**
  - Header-> Navigation bar (logo, home, about us, more -> [sessions, packages, our branches, contact us], login, register)
  - Login button takes the user to the login form then fill the id, email and password then his profile is displayed.

- Register button takes the user to the signup form the he fills the data required to make a profile on the website.

- First section: Motivating words.

- Second Section (about us) has two parts:
- First part: who are we and our vision
- Second part: Slide show that contains our trainers' photos.

- Third Section (sessions):
- This part shows our sessions.
- Booking button so that if any member wished to book an extra session this button will take him to a booking form where he has to fill the data and choose a trainer and also choose which way he will pay with.

- Fourth Section (packages):
- This part shows our packages.
- Booking button that leads the member to a booking form where he has to fill the data and choose the package that suits him.

- Footer:
- Contact Us (hotline/email)
- Our Branches.

- **Admin Profile Page**

- Header-> Navigation bar (logo, home, about us, more -> [sessions, packages, our branches, contact us], logout)

- Logout button exits the admin profile and takes him to the home page and he needs to login again if he wants to enter his profile.

- First Section (options Buttons):
- There is five buttons ->(member, trainer, packages, sessions, report)
  - Member button takes the admin to member modification page
  - Trainer button takes the admin to trainer modification page
  - Sessions button takes the admin to sessions modification page
  - Packages button takes the admin to package modification page

        - Report button takes the admin to the packages PDF report which can be downloaded.

- Second Section (review):

- This part shows the reviews written by members.

- **Member Modification Page**

  - First Section (four buttons):

  - View
    - It takes the admin to a form where he enters the member id to view his information.
  - Add
    - It takes the admin to a form where he fills the member data to add him to the database.
  - Update
    - It takes the admin to a form where he enters the member id then he goes to update form where he updates the data of this specific member.
  - Delete
    - It takes the admin to a form where he enters the member id to delete him from the database.

- **Trainer Modification Page**

  - First Section (six buttons):

  - search
    - It takes the admin to a form where he enters the trainer id to show this trainer’s record.
  - View
    - It takes the admin to a form where he enters the trainer id to view his information.
  - Add

        - It takes the admin to a form where he fills the trainer data to add him to the database.

    - Update
        - It takes the admin to a form where he enters the trainer id then he goes to update form where he updates the data of this specific trainer.
    - Delete
        - It takes the admin to a form where he enters the trainer id to delete him from database.

- **Sessions Modification Page**
  - First Section (five buttons):
    - list
        - This lists the sessions we have.

    - View
        - It takes the admin to a form where he enters the session id to view its information.

    - Add
        - It takes the admin to a form where he fills the session data and upload its photo then add it to the database.

    - Update
        - It takes the admin to a form where he enters the session id then he goes to update form where he updates the data of this specific session.

    - Delete

        - It takes the admin to a form where he enters the session id to delete it from database.

- **Packages Modification Page**
  - First Section (five buttons):
    - list
        - This lists the packages we have.

    - View
        - It takes the admin to a form where he enters the package id to view its information.

    - Add
        - It takes the admin to a form where he fills the package data to add it to the database.

    - Update
        - It takes the admin to a form where he enters the package id then he goes to update form where he updates the data of this specific session.

    - Delete
        - It takes the admin to a form where he enters the package id to delete it from database.

- **Member Profile Page**
  - Header-> Navigation bar (logo, home, about us, more -> [sessions, packages, our branches, contact us, change password], notifications, logout)
    - Logout button exits the member profile and takes him to the home page and he needs to login again if he wants to enter his profile.
    - Change password button takes the member to a form where he enters his old password, his new password and confirm the new password in order to set a new one.

- First Section (options Buttons):
  - There is two buttons ->(my package, my extra sessions)
    - My packages button display the member his packages.
    - My extra sessions button display the member his extra sessions.

- Second Section (review):
  - This part the member writes a review about a specific trainer.

- **Trainer Profile Page**

  - Header-> Navigation bar (logo, home, about us, more -> [sessions, packages, our branches, contact us, change password], logout)

    - Logout button exits the trainer profile and takes him to the home page and he needs to login again if he wants to enter his profile.
    - Change password button takes the trainer to a form where he enters his old password, his new password and confirm the new password in order to set a new one.

  - First Section (options Buttons):
    - There is two buttons ->(my sessions, take attendance)
      - My sessions button display the trainer his assigned sessions.
      - Take attendance button takes the trainer to a form where he can take the attendance of a member in a specific session by entering his id, the session id and the member id then this data is saved in the database.

13) Document an Implementation based on the abovementioned Requirements & Design:

A) User Role Management Module:

. Login Function

-when the user signs in the system automatically takes him to his profile where each user has his own privileges if he is admin, trainer or a member and if the user enters the data incorrectly the system will display an error message to him so that he can rewrite the data correctly.

- Also we use the session function in it to keep the user in from the moment he signs in to the moment he logs out.

```php
        function login($id,$mail,$pass){
            $connectdb  = connectdb::getInstance();
            $conn = $connectdb->getConnection();
            if(!empty($id) AND !empty($mail) AND !empty($pass)){

                $sqlmember ="SELECT Member_id and Member_email and Member_password FROM member WHERE
Member_id='$id' and Member_email = '$mail' and Member_password = '$pass'";
                $sqltrainer ="SELECT Trainer_id and Trainer_email and Trainer_password FROM trainer
WHERE Trainer_id='$id' and Trainer_email = '$mail' and Trainer_password = '$pass'";
                $sqladmin="SELECT Admin_id and Admin_email and Admin_password FROM admin WHERE
Admin_id='$id' and Admin_email = '$mail' and Admin_password = '$pass'";
                if(isset($sqltrainer)){
                    $result= mysqli_query($conn,$sqltrainer);
                    if(mysqli_num_rows($result)==1){
                        session_start();
                        $_SESSION['id']=$id;
                        $_SESSION['mail']=$mail;
                        $_SESSION['pass']=$pass;
                         header('location:indexTrainer.php');

                         }

                     }
                if(isset($sqlmember)){
                        $result= mysqli_query($conn,$sqlmember);
                        if(mysqli_num_rows($result)==1){
                            session_start();
                            $_SESSION['id']=$id;
                            $_SESSION['mail']=$mail;
                            $_SESSION['pass']=$pass;
                             header('location:indexMember.php');

                             }

                    }
                    if(isset($sqladmin)){
                        $result= mysqli_query($conn,$sqladmin);
                        if(mysqli_num_rows($result)==1){
                            session_start();
                            $_SESSION['id']=$id;
                            $_SESSION['mail']=$mail;
                            $_SESSION['pass']=$pass;
                             header('location:indexAdmin.php');

                             }
                             else{
                                 echo "email or password are wrong";
                            }

                    }
                }
                else{
                    echo "Please enter email and password";
                }


        }
```

.update profile function:

-the function allows both members and trainers to change the password of their profile whenever they needed.

- the method asks the user to write his old password then the new one and then the new one again as a confirmation after it find a match of the old password in the database, it compares the two new passwords if they match then the password will be changed and a message will be displayed for the user telling him that the password has been successfully changed.

```php
        function updateProfile($oldpass,$newpass,$confirmpass){
            $connectdb  = connectdb::getInstance();
            $conn = $connectdb->getConnection();
            $id=$_SESSION['id'];
            if(isset($id)){
                $query="SELECT Member_password FROM Member WHERE Member_id='$id'";
                $query2="SELECT Trainer_password FROM Trainer WHERE Trainer_id='$id'";
                if(isset($query)){
                    $result= mysqli_query($conn,$query);
                    while($row=mysqli_fetch_array($result)){
                        $pass=$row['Member_password'];
                        if($pass==$oldpass){
                            if($newpass==$confirmpass){
                                $updatequery="UPDATE Member SET Member_password='$confirmpass' WHERE
Member_id='$id'";
                                $update=mysqli_query($conn,$updatequery);
                                echo"updated successfully";
                            }
                            else{
                                echo"passwords don't match";
                            }

                        }
                    }

                }
                if(isset($query2)){
                    $result= mysqli_query($conn,$query2);
                    while($row=mysqli_fetch_array($result)){
                        $pass=$row['Trainer_password'];
                        if($pass==$oldpass){
                            if($newpass==$confirmpass){
                                $updatequery="UPDATE Trainer SET Trainer_password='$confirmpass' WHERE
Trainer_id='$id'";
                                $update=mysqli_query($conn,$updatequery);
                                echo"updated successfully";
                            }
                            else{
                                echo"passwords don't match";
                            }

                        }
                    }

                }
            }
            else{
                header('location:signinform.php');
            }


        }
```

.Take attendance function:

-in this function the trainer takes the attendance of the members in a specific session and it is recorded in the database.

```php
function TakeAttendance($trainerid,$memberid,$sessionid,$date,$count){
    $connectdb  = connectdb::getInstance();
    $conn = $connectdb->getConnection();
    if(!empty($trainerid) AND !empty($memberid) AND !empty($sessionid)){

        $sql ="INSERT INTO attendance(Trainer_id,Member_id,Session_id,Date,Count) VALUES
('$trainerid','$memberid','$sessionid','$date','$count')";
        $result= mysqli_query($conn,$sql);
           $_SESSION['message']="welcome";
            echo "attendance done successfuly";
        }
        else{
            echo "Please fill the data";
        }
}
```

b) User manipulation Module:

Add :-

We have created class Database for database functions which insert is one of these functions and its content is the query that insert values in database

At class Admin we create an object from database class and call insert function and pass the values as parameter to it at the body of the required function (add member, add trainer ,……)

```php
//Insert
static function insert($tablename,$value){
  $connectdb  = connectdb::getInstance();

  $conn = $connectdb->getConnection();

  $sql= "INSERT INTO $tablename VALUES ($value)";

  echo "sql command : $sql <br>";
   $q=mysqli_query($conn,$sql);
  if($q){
    return 1;
  }
  else{
  return 0;
  }
}
```

```php
    }

    public function AddTrainer($Id,$FirstName,$LastName,$Email,$Password,$Salary,$HiredDate,$Shift)
    {

     $Db = new DataBase();   //connect DB
     $TableName="trainer(Trainer_id,Trainer_Fname,Trainer_Lname,
     Trainer_email,Trainer_password,Trainer_salary,Hired_Date,Shifts)";
     $Values = "'$Id','$FirstName','$LastName','$Email',
     'Password','$Salary','$HiredDate','$Shift'"; //new trainer data
     $Db -> insert($TableName,$Values);
```

-Delete :-

In class database , Delete is one of its function and its content is the query that delete record from table in database. At class Admin we create an object from database class

and call delete function and send the record to parameter of the function at the body of the required function (Delete member ,Delete trainer ,……)

```
//Delete
static function delete($tablename,$id,$value){

$connectdb  = connectdb::getInstance();
$conn = $connectdb->getConnection();
            $sql = "DELETE FROM $tablename WHERE $id=$value";
            if (mysqli_query($conn, $sql)) {
        return 1;
            } else {
             return 0 ;
           }

           }


public function DeleteTrainer($Id)
{
    $Db = new DataBase();
    $Db->delete("trainer","Trainer_id",$Id);

}
```

-Update :-

In class database, update is one of its function and its content is the query that update values in database.

At class Admin we create an object from database class and call update function and give it new values and the place of values need to be updated at the body of the required function (update member, update trainer ,……)

```
//Update
static public function update($tablename,$records,$idName,$idValue)
{

  $connectdb  = connectdb::getInstance();

  $conn = $connectdb->getConnection();
  $count = 1;
  foreach ($records as $key => $value) {
    echo "$key = $value <br>";

      $query = "UPDATE $tablename SET $key = $value where $idName =  $idValue";
      echo "query to database $query <br>";
      $result = mysqli_query($conn,$query);
      echo "<br> $query";
      //echo "<br> mysqli_affected_rows($conn)<br>";
    if($bool = mysqli_affected_rows($conn) >= 1)
      {$count = $count +1;

      }

  }

  if($count == sizeof($records))
  {
    return 1;
  }
  else {
    return 0;
  }

}
//End of class
```

-Search :-

In class database , Select record is one of its function and its content is the query that Select record from table in database.

At class Admin we create an object from database class and call select record function and send the record to parameter of the function at the body of the required function (View member ,Search trainer ,……)

```
static function selectRecord($tablename,$id,$value){

        $connectdb  = connectdb::getInstance();

         $conn = $connectdb->getConnection();


   $query = "SELECT * FROM $tablename where $id= $value";

   $result = mysqli_query($conn,$query);

   $array = mysqli_fetch_all( $result , MYSQLI_ASSOC);
//echo  $conn->rowcount() ."<br>";
   //print_r($array);

return $array;
```

```
public function Search($Id)
{
    $Db = new DataBase();
    return $Db->selectRecord("trainer","Trainer_id",$Id);
}
```

- List:-

In class database , Select All is one of its function and its content is the query that Select all date from specific table in database.

At class Admin we create an object from database class and call select all function and send the table name to parameter of the function at the body of the required function (List session , list package)

```
static function selectAll($tablename)
{

  $connectdb  = connectdb::getInstance();

  $conn = $connectdb->getConnection();

      $query = "SELECT * FROM $tablename";

      $result = mysqli_query($conn,$query);
     if($bool = mysqli_affected_rows($conn) >= 1)
  {
  //  echo "<br>"."Affected rows: " . mysqli_affected_rows($conn)."<br>";
        $arrayOfResult = mysqli_fetch_all( $result ,MYSQLI_ASSOC);

   //print_r($arrayOfResult);
    }
     if( $bool >= 1)
   {
  return $arrayOfResult;
   }
  else
   {
    return 0;
    }
    //return $arrayOfResult;
  }
```

```
public function ListPackages()
{
    $Db = new DataBase();
    return $Db ->  selectAll("Packages");

}
```

c) Controlling resources module (package, session)

We control our product which represent as package or session by add new one or update and delete an existed session or product or to view a specific product or to list them all . some snapshots for the code below : -

Add :-
An example to add a new session to the system by adding its value to database

```
    public function AddSession($SessionId,$SessionName,$SessionCost,$Discount
        ,$StartTime,$EndTime,$SessionDays,$SessionGoal,$TrainerName)
{
   //connect DB
 $TableName="sessions(Session_id,Session_name,Session_cost,discount
 ,Start_time,End_time,Session_days,Session_goal,Trainer_name)";
 $Values = "'$SessionId','$SessionName','$SessionCost','$Discount',
 '$StartTime','$EndTime','$SessionDays','$SessionGoal','$TrainerName'";

 Database::insert($TableName,$Values);
```

Update :-
This example is to update data for an existed session at the system by updating its data from session table in database

```
public function updateSession($sessionDetails,$sessionId)
{
    $session =  array(' Session_name ' => " '$sessionDetails[0]' ",
        ' Session_cost '=>$sessionDetails[1],' discount '=>$sessionDetails[2],
        ' Start_time '=>" '$sessionDetails[3]' ",' End_time '=>"'$sessionDetails[4]'",
        ' Session_days '=>"'$sessionDetails[5]'",' Session_goal '=>"'$sessionDetails[6]'",
        ' Trainer_name '=>" '$sessionDetails[7]'" );
    Database::update("sessions",$session,'Session_id',$sessionId);
}
```

Delete:-
Also we can delete session or package by selecting our record need to be deleted and delete it form the database

```
    function deletePackage($id)

   $tableName = 'packages';
   $packageId = 'Package_id';

    Database:: deletefunction ($tableName,$packageId,$id) ;
```

View :-
We can view a single record of our products by select its id and displays all the data of this product

```
public function ViewSession($SId)
{
    $Db = new DataBase();
    return $Db->selectRecord("sessions","Session_id",$SId);
}
```

```php
<?php foreach ((array)$SId as  $session) {?>
 <div class="class-content center">
      <div class="one temp text-white" style="background-color: #060520e6; border:4px solid black;">

<div><?php echo "Session Id → " . ($session['Session_id']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Session Name → ".($session['Session_name']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Session Cost → ".($session['Session_cost']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Discount → ".($session['discount']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Start Time → ".($session['Start_time']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "End Time → ".($session['End_time']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Session Days → ".($session['Session_days']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Session Goals → ".($session['Session_goal']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Trainer Name → ".($session['Trainer_name']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div> <?php $Ad-> display($session['Session_id']); ?></div>
      </div>
      </div>
  <?php } ?>
```

This part is to display information to GUI

List:- As we can see a single record of our products also we can see all sessions by selecting them all from table of the database

```php
public function ListPackages()
{
    $Db = new DataBase();
    return $Db ->  selectAll("packages");

}
```

```php
<?php foreach ((array)$packages as  $package) {?>
 <div class="class-content center">
      <div class="one temp text-white" style="background-color: #060520e6; border:4px solid black;">

<div><?php echo "Package Id → " . ($package['Package_id']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Package Duration → ".($package['Package_duration']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Begining Date → ".($package['Beginning_date']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "End Date → ".($package['End_date']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Package Fee → ".($package['Package_fee']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Admin Id → ".($package['Admin_id']); ?></div>
<div style="height:1px; background-color:grey"></div>
<div><?php echo "Session Details → ".($package['sessionsDetails']); ?></div>

      </div>
      </div>
  <?php } ?>
```

D)Reservation & Rescheduling Module

The Reservation Module in The FCMS project is applied in the Member Module, Where The member has the capability to book A specific session or package from the available products in the database, Then the system shows to the user the available trainers for that session then the member can select any trainer he wants.
As well as that the member has the ability to choose his payment method and choose to pay by cash, Visa or PayPal.
The Booking featured is applied by 4 methods for each session booking and Packages Booking.
The 1st method displays all the product(session/packages) That the system provides to the user.

```php
//Display to the user ALL sessions in database
public function DisplaySession(){

   $connectdb  = connectdb::getInstance();
   $conn = $connectdb->getConnection();

//Retrive the id of avaiable trainers for that session
  $query="SELECT DISTINCT Session_id ,Session_name FROM sessions ";
  $result = mysqli_query($conn,$query);
  // $arrays=mysqli_fetch_all($result,MYSQLI_ASSOC);

   return $result;
}
```

The 2nd method Is when the member enters the specific Id of the desired choice of product(session/package), The method select Session Retrieves the id of the available trainer for that selected product.

```php
public function selectSession($sessionId){//when the user enters the session id

   $connectdb  = connectdb::getInstance();
   $conn = $connectdb->getConnection();
//Retrive the id of avaiable trainers for that session
  $query="SELECT TrainerID FROM sessions WHERE Session_id = $sessionId ";
  $result = mysqli_query($conn,$query);
  $arrays=mysqli_fetch_all($result,MYSQLI_ASSOC);
   return $arrays;
}//Returns the Assoc array containing the TrainerId
```

And as we need to show the user the names of those trainer not only the id, to make it more user friendly for the members to choose of the available Trainer name, so we go to the 3rd method which objective is to retrieves the available trainer names of that package. This method also invokes the final method needed in the booking Cycle which is Book Session() that insert the selection values of the member in the booking session/booking package tables in the database.

```php
//Retrive the avaiable trainers for that session
public function selectTrainer($trainername){
     $connectdb  = connectdb::getInstance();
   $conn = $connectdb->getConnection();
   $qu="SELECT Trainer_id FROM trainer WHERE Trainer_name = '$trainername' ";

   $results = mysqli_query($conn,$qu);
   $arrays=mysqli_fetch_assoc($results);
   $trainerId=$arrays['Trainer_id'];

   $id=$_SESSION['id'];//Member's Id from login

   $IdOfSession=$_SESSION['sessionId'];

   $valuesToInsert="$IdOfSession,$id,$trainerId";//Id of the member

    $member=new Member();
    $member->bookSession($valuesToInsert);

  // return $arrays;

  }//Returns the TrainerId of the selected Trainer by the user to be stored in the database
```

The Payment Is applied in the member module as in the booking the system Asks the Member to select a payment method and that selection is saved in the database as well.

```php
public function pay($member_if,$option){

   $member_id=$_SESSION['id'];
    if ($option == 'cash'){
                $way = new CashStrategy();
                $way -> pay($member_id,$option);
            }

            elseif ($option == 'visa'){
                $way = new CreditCardStrategy();
                $way -> pay($member_id,$option);
            }

            elseif ($option == 'paypal'){
                $way = new PayPalStrategyStrategy();
                $way -> pay($member_id,$option);
            }


}
```

Another Capability is supported by the system to the user is the writing Review Method, Where the member can give any feedback on the trainers by providing the name the trainer, he wants to write the review on as well as the feedback he wants to write.

```php
public function writeReview($trainer_name,$feedback){

        $connectdb  = connectdb::getInstance();
        $conn = $connectdb->getConnection();
        $qu="SELECT Trainer_id FROM trainer WHERE Trainer_name = '$trainer_name' ";
        $results = mysqli_query($conn,$qu);
        $arrays=mysqli_fetch_assoc($results);
        $trainer_id=$arrays['Trainer_id'];
        $member_id=$_SESSION['id'];
        $member=new Member();
        $db = new DataBase();
        $TableName="review(Member_id,Trainer_id,Review)";
        $Values = "$member_id,$trainer_id,'$feedback'";
        $db->insert($TableName,$Values);
       }
```

e) Generating Reports Module:

We Chose to generate PDF Document to Admin to give him the Data about Packages from the Database so:

I have used Fpdf library bec, it’s a simple one ,there are other libraries like Tcpdf and other libraries in framework of laraven but I chose the simplest one which is sufficient in our simple case
working with fpdf is simple you can just create an instance of the class fpdf with the required format parameters and use method add page to create a pdf page and then add table and text as you desire but this method is not effective bec, you will have to add a lot of options in header and footer
so, I simply created class pdf than inherent the class fpdf and simply override it’s required methods
which is header , footer, and made functions to adjust the table and its cells also called method from
select all from data layer which give the whole data as array then I loop in the array and put each data in the cell of the table then I simply call this method in the admin function generate pdf
NOTES :
making class of pdf that inherent the fpdf class make it more flexible for any updates and any options also it allows the reuse of code as I can reuse this class and generate this pdf in any module
I want .

Tools Project Preferences Help

AdminClass.php PDF.php

```php
require('FpdfLibrary/fpdf.php');

include('../DataLayer/Database.php');

class pdf extends FPDF
{

function Header()
{
    $this->SetFont('Arial','B',18);
    $this->Cell(180,5,'Package Report',0,0,'C');
    $this->Ln(10);


}
function Footer()
{
    // Position at 1.5 cm from bottom
    $this->SetY(-15);
    // Arial italic 8
    $this->SetFont('Arial','I',8);
    // Page number
    $this->Cell(0,10,'Page '.$this->PageNo().'/{nb}',0,0,'C');
}
// Load data
function headerTable()
{
    $this->SetFont('Arial','B',12);
    $this->Cell(10,10,'ID',1,0,'C');
    $this->Cell(30,10,'Duration',1,0,'C');
    $this->Cell(40,10,'Begining Date',1,0,'C');
    $this->Cell(30,10,'End Date',1,0,'C');
    $this->Cell(40,10,'Package Fee',1,0,'C');
    $this->Cell(40,10,'Creator Admin',1,0,'C');
```

tdocs/SoftwareFinal/FitnessClub/ApplicationLayer/PDF.php (FitnessClub) - Sublime Text (UNREGISTE

Tools Project Preferences Help

AdminClass.php PDF.php

```php
function headerTable()
{
    $this->SetFont('Arial','B',12);
    $this->Cell(10,10,'ID',1,0,'C');
    $this->Cell(30,10,'Duration',1,0,'C');
    $this->Cell(40,10,'Begining Date',1,0,'C');
    $this->Cell(30,10,'End Date',1,0,'C');
    $this->Cell(40,10,'Package Fee',1,0,'C');
    $this->Cell(40,10,'Creator Admin',1,0,'C');
    $this->Ln();
    //$this->Cell(20,10,'ID',1,0,'C');

}

function viewTable()
{
    $db = new Database();

    $stat = $db->selectAll("packages");
    foreach ($stat as $key => $value)
    {

        $this->Cell(10,10,"$value[Package_id]",1,0,'L');
        $this->Cell(30,10,"$value[Package_duration]",1,0,'L');
        $this->Cell(40,10,"$value[Beginning_date]",1,0,'L');
        $this->Cell(30,10,"$value[End_date]",1,0,'L');
        $this->Cell(40,10,"$value[Package_fee]",1,0,'L');
        $this->Cell(40,10,"$value[Admin_id]",1,0,'L');
        $this->Ln();
    }
}
//end class
}
```

```
/SoftwareFinal/FitnessClub/ApplicationLayer/AdminCla
ols  Project  Preferences  Help
AdminClass.php
    }

//Function to generate pdf
    public function generatePdf()
    {
        $pdf = new pdf();
        $pdf->AliasNbPages();
        $pdf->AddPage();
        $pdf->headerTable();
        $pdf->viewTable();
        $pdf->Output();


    }
    //end of function
```

F) Notification Module :

out Notification Module is based on **Observer Design Pattern** which is described in details in part 2 number K , Admin implement our subject interface in which he override 3 methods Add observer which in which he add the member in the table observers in database , delete observer in which he delete the member from table observers , notify in which he send notification message in the table notification to the observers , member class implements observer interface and override method update to select the notification specified by admin and put it in the menu

NOTE :

we made the admin add the member to the observer table by default when he add new member

also when he add new session automatically send notification to all observers

```php
<?php


interface Observer
{
    public function update();
}

interface Subject
{
    public  function addObserver(Observer $newObserver);
    public  function deleteObserver(Observer $removedObserver);
    public  function notify($sessionName);

}



?>
```

Tools Project Preferences Help

PersonClass.php signupform.php connectdb.php AdminClass.php indexAdmin.php

```php
class Admin implements Subject
{
    public   $observers = [];
    public $id;
    public $notificationId = 1;
    public function __construct($id)
    {
        $this->id = $id;

    }
    public  function addObserver( Observer $newObserver)
    {
        //array_push(Admin::$observers, $newObserver->member_id);
        $observer = array('Member_Id' => $newObserver->member_id,'Admin_Id'=>
            ;
        //echo "admin added Observer $newObserver->id"."<br>";
        print_r($observer);

        Database::insert("Observers",implode( ',',$observer));
    }


```

PersonClass.php | signupform.php | connectdb.php | AdminClass.php | indexAdmin.php | Observer.php

```php
    }
    public  function addObserver( Observer $newObserver)
    {
        //array_push(Admin::$observers, $newObserver->member_id);
        $observer = array('Member_Id' => $newObserver->member_id,'Admin_Id'=>$this->id)
            ;
        //echo "admin added Observer $newObserver->id"."<br>";
        print_r($observer);

        Database::insert("Observers",implode( ',',$observer));
    }


    public  function notify($sessionName)
    {

        $observers=DataBase::selectAll("Observers");
        foreach ($observers as $observer)
        {
            # code...
            # code...
            $observer = new member($observer['MemberId']);

            $array = array('Notification_Id'=>'NULL','Admin_Id'=> $this->id,'Member_Id'
                 => $observer->member_id,'Notification'=>" ' Check our New Session $
                sessionName '");
            echo "Observers id ";
            echo $observer->member_id;
                    $array = implode(",", $array);


            Database::insert("notification",$array);
        //  $this->notificationId++;
        }
```

```
PersonClass.php   signupform.php   connectdb.php   AdminClass.php   Memberclass.php
9   require_Once('..//ApplicationLayer/Observer.php');
10    //  include('Observer.php');
11  //if(require('../DataLayer/Database.php'))
12    //include('../DataLayer/Database.php');
13  require_Once('../DataLayer/Database.php');
14
15
16
17
18  class Member implements observer{
19    private $
20    private $
21    public $m
22    public $id ;
23    public $messegeArray =[];
24  //Update function of member
25     /*public function __constructor($mid){
26
27      $this ->member_id=$mid;
28     }*/
29   public function Member($id)
30    {
31      $this->member_id=$id;
32     // echo "the member id is $this->member_id<br>";
33
34
35     }
36    public function update()
37    {
38      $Db = new DataBase();
39    return $Db ->selectRecord('notification',"Member_Id",$this->member_id);
40    }
41    /*public function getNotifications()
```

Definition:
ApplicationLayer/Memberclass.php:29

g) File Uploaders:

→ uploading image :

The idea of the function is to uploading photos to MySQL database and the uploaded image is stored in a directory of the server and the file name is inserted in the database.

Later, the images are retrieved from the server based on the session id of the stored image in the database and display on the web page.

1) The requirements for the database creation :

1st create a column with data type **blob** for storing images

2nd assign a unique id -session id- for each image

2) The requirements in php script to uploading :

In MySQL image will be stored into binary form of data. We will use MySQL blob data type in MySQL table and in this datatype, we have store images in binary form. Here we have define one input file element and one submit button we can select file from this input element and click on submit button then after image will be send to php script and by using file_get_contents() function we can read selected image file in string format and after this we will insert that image into MySQL database. In MySQL database table we have store image in column with data type blob.

```
        function upload_image($SessionId)
  {
    # code...


//$connect = mysqli_connect("localhost", "ahmed", "12345", "project");
   $connectdb  = connectdb::getInstance();

   $connect = $connectdb->getConnection()

    $file = addslashes(file_get_contents($_FILES["image"]["tmp_name"]));
    $query = " UPDATE sessions SET name = ('$file') where Session_id = $SessionId";
    if(mysqli_query($connect, $query))
    {
        echo '("Image Inserted into Database")';
      }
```

→ Displaying image :

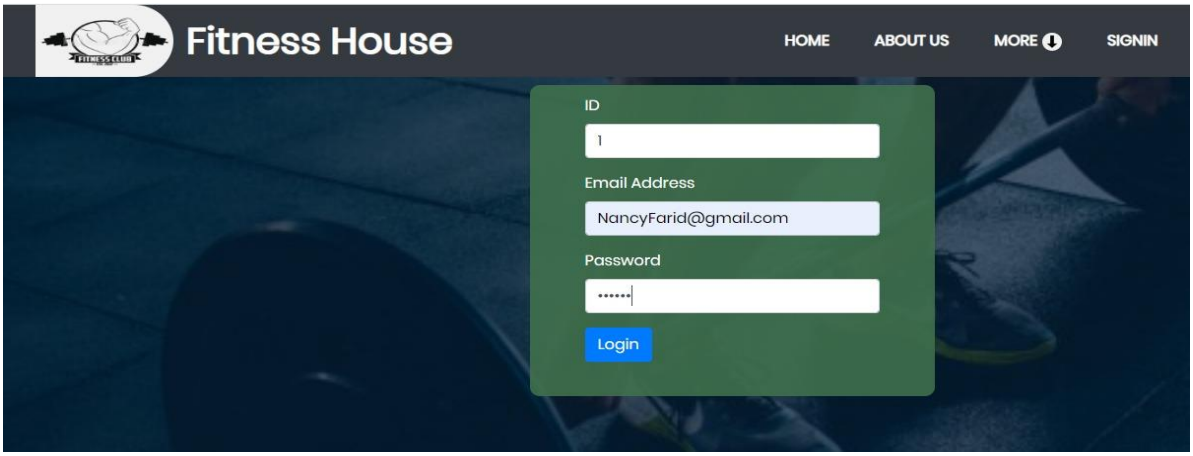
After this we want to display image on web page, so we have shown that inserted image into image tag by converting binary data into image by using base64encode() function. So, this way we can insert image into MySQL database by using php script.

```php
function display($SessionId){

        $connect = mysqli_connect("localhost", "ahmed", "12345", "project");
        $query = " SELECT name FROM sessions where Session_id = $SessionId " ;
       $result = mysqli_query($connect, $query);
       while($row = mysqli_fetch_array($result))
       {
            echo '
                <tr>
                    <td>
                        <img src="'.base64_encode($row['name'] ).'" height="200" width="200"
                            class="img-thumnail" />
                    </td>
                </tr>
            ';
       }


}
```

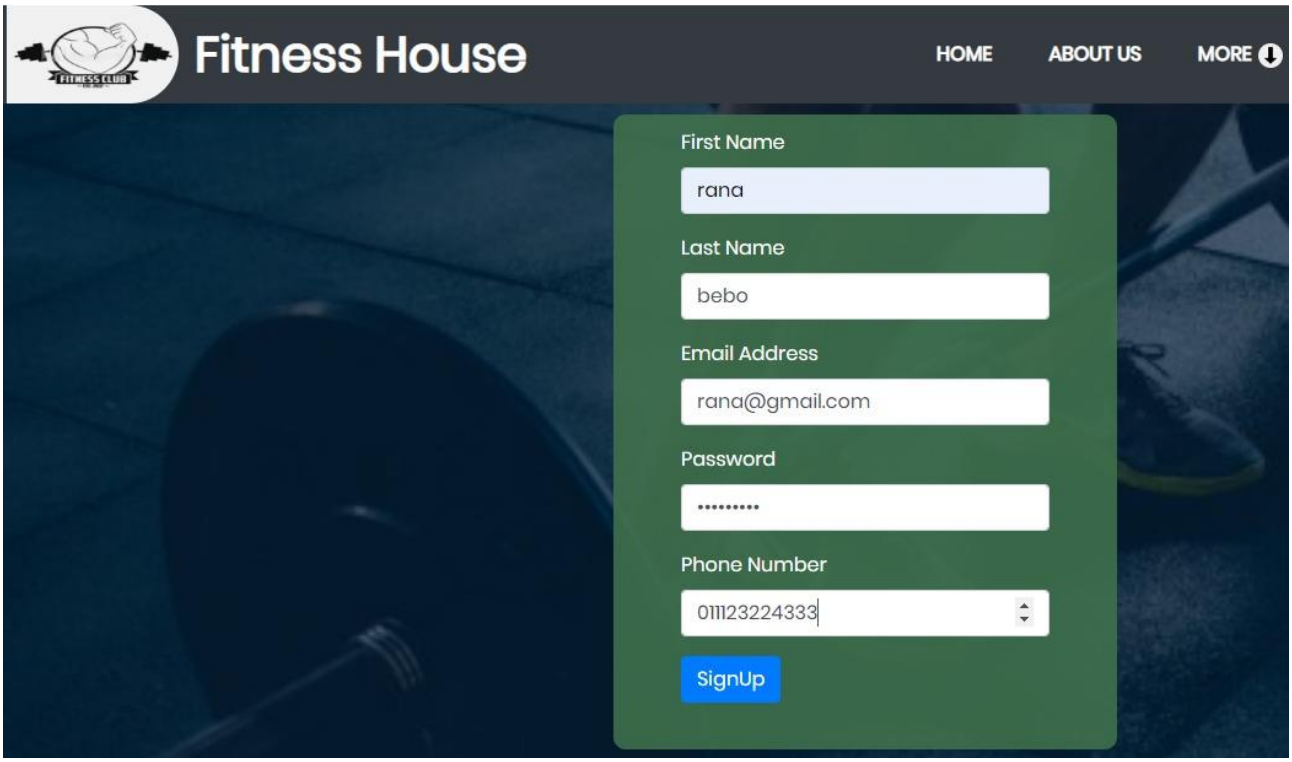
14) Design Some user interface:

1) user module:

- ***Login:***

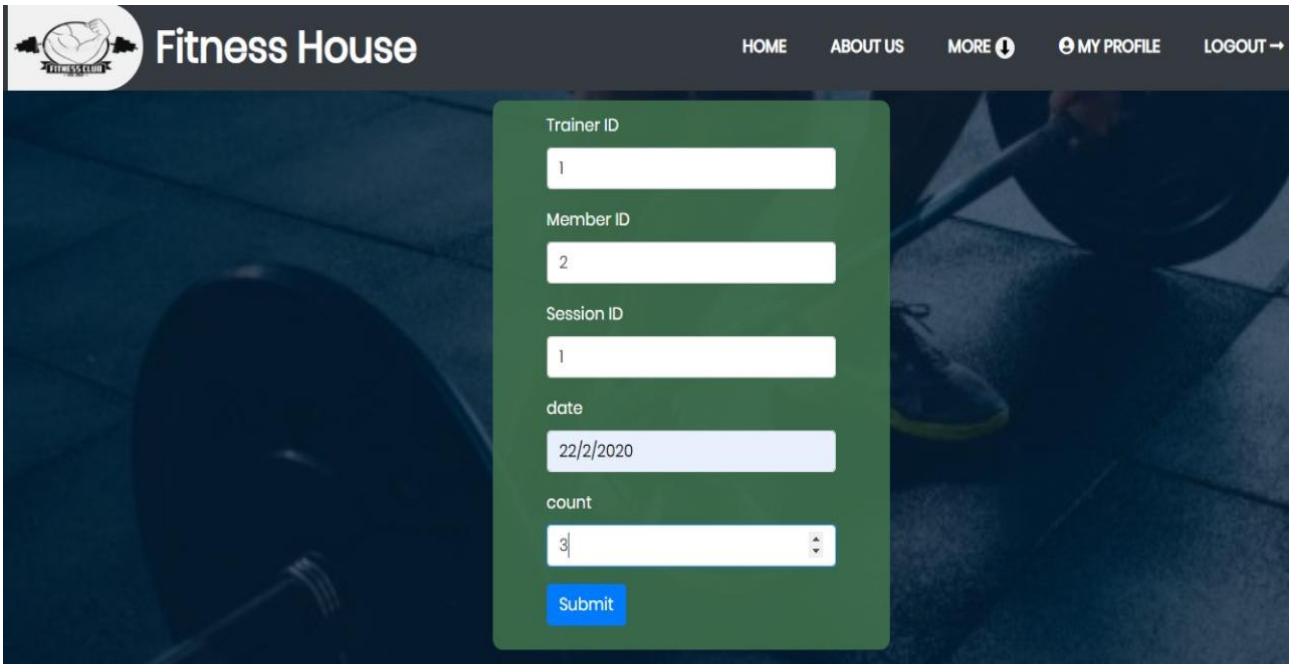


- ***Sign up:***

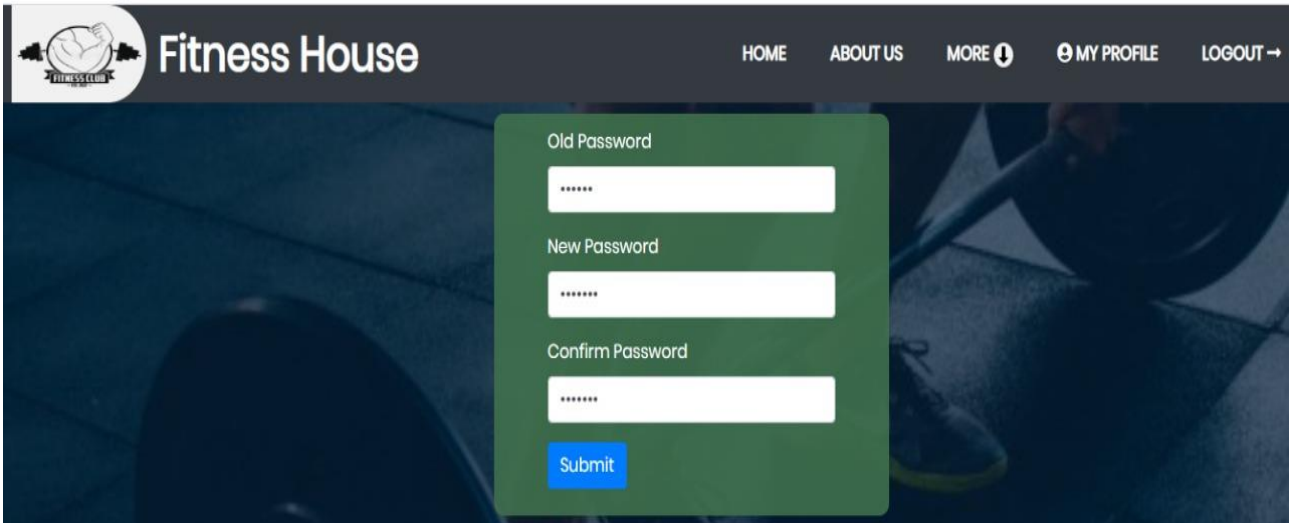


2) Trainer Module:

- ***<u>Take Attendance:</u>***

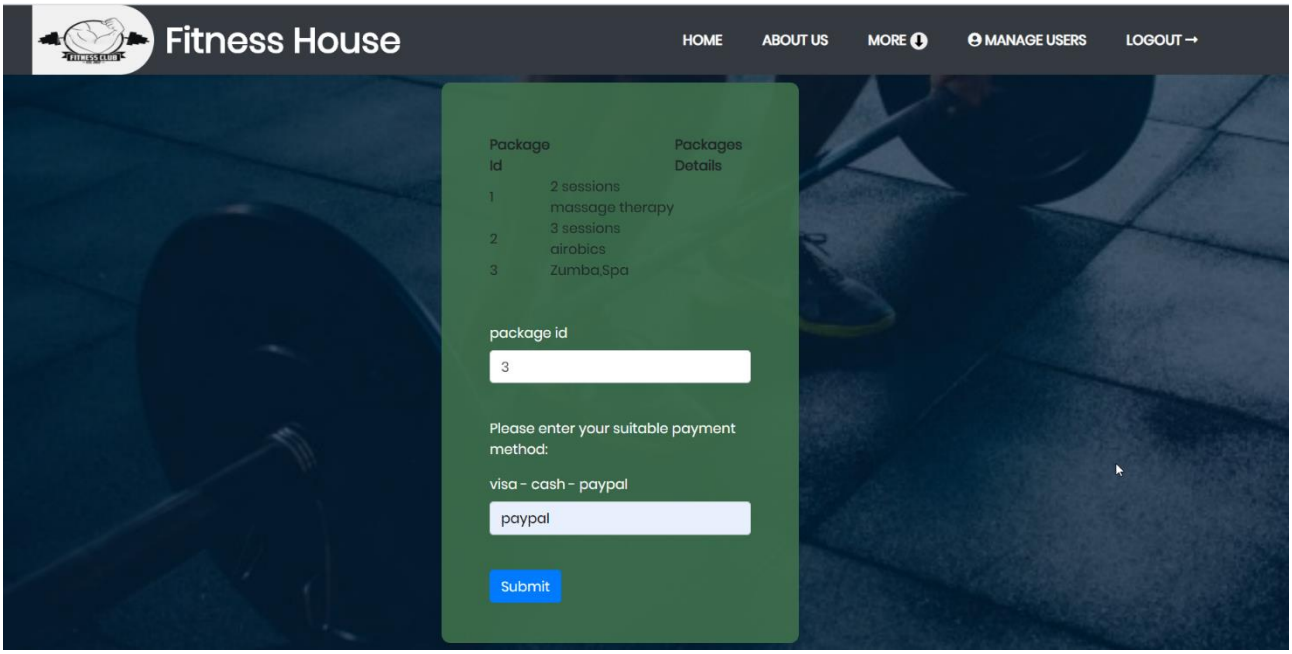


- ***<u>Update Profile:</u>***

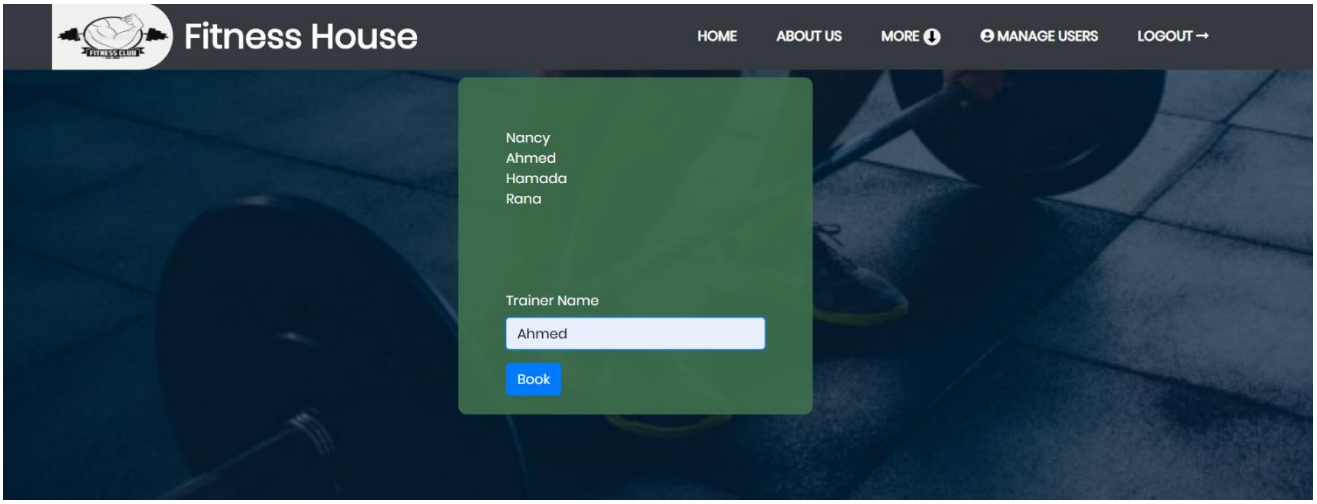


3) Member Module:

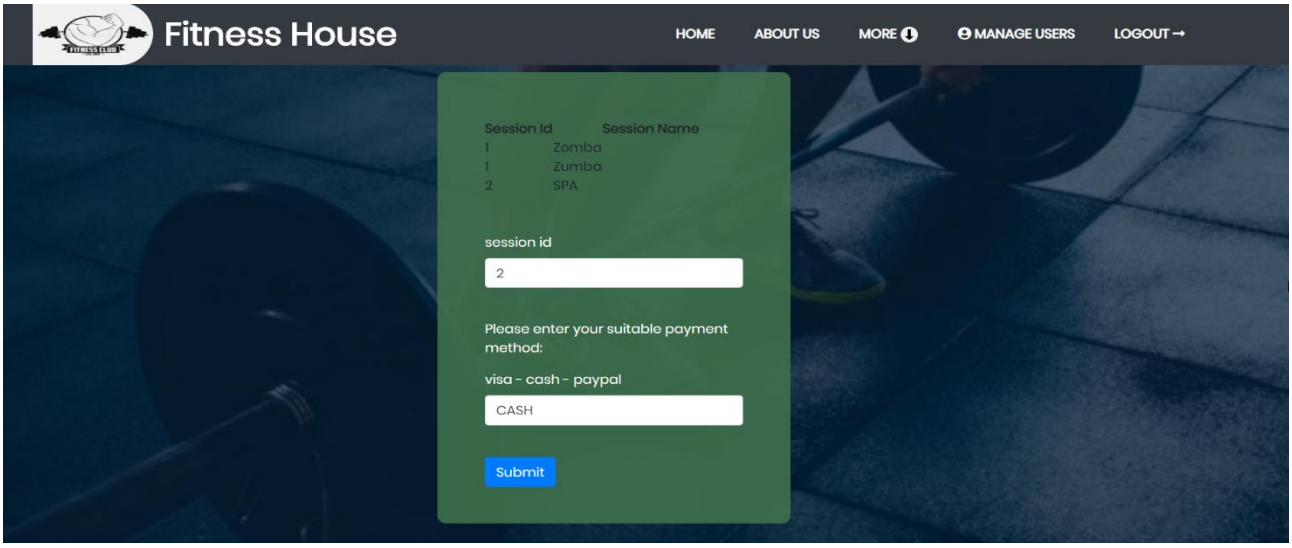
- ***<u>Booking Packages:</u>***



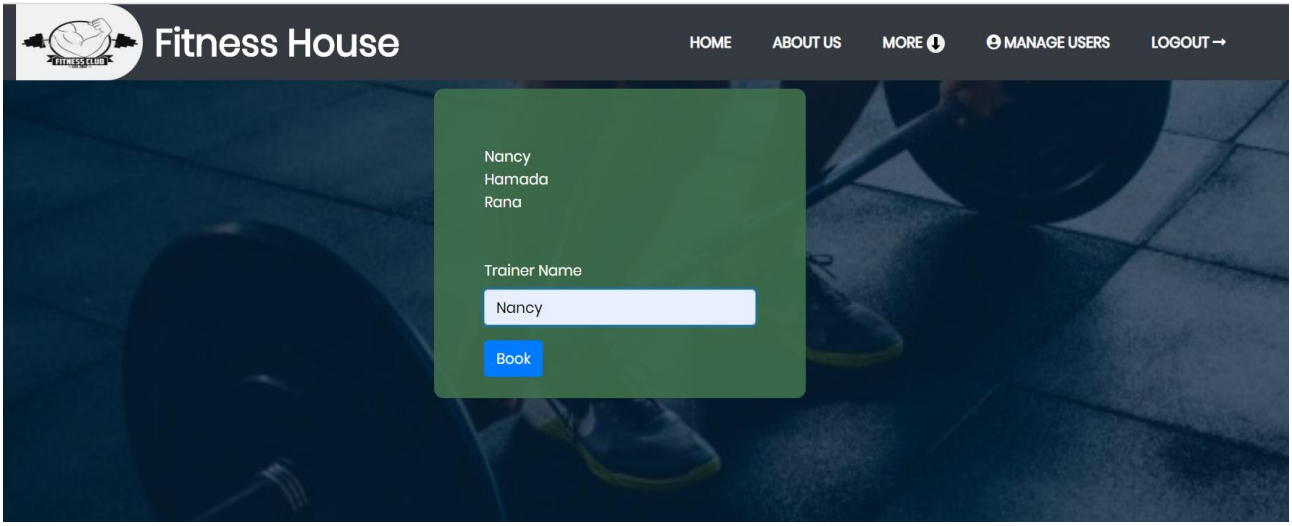
- ***<u>Select Trainers of Packages:</u>***



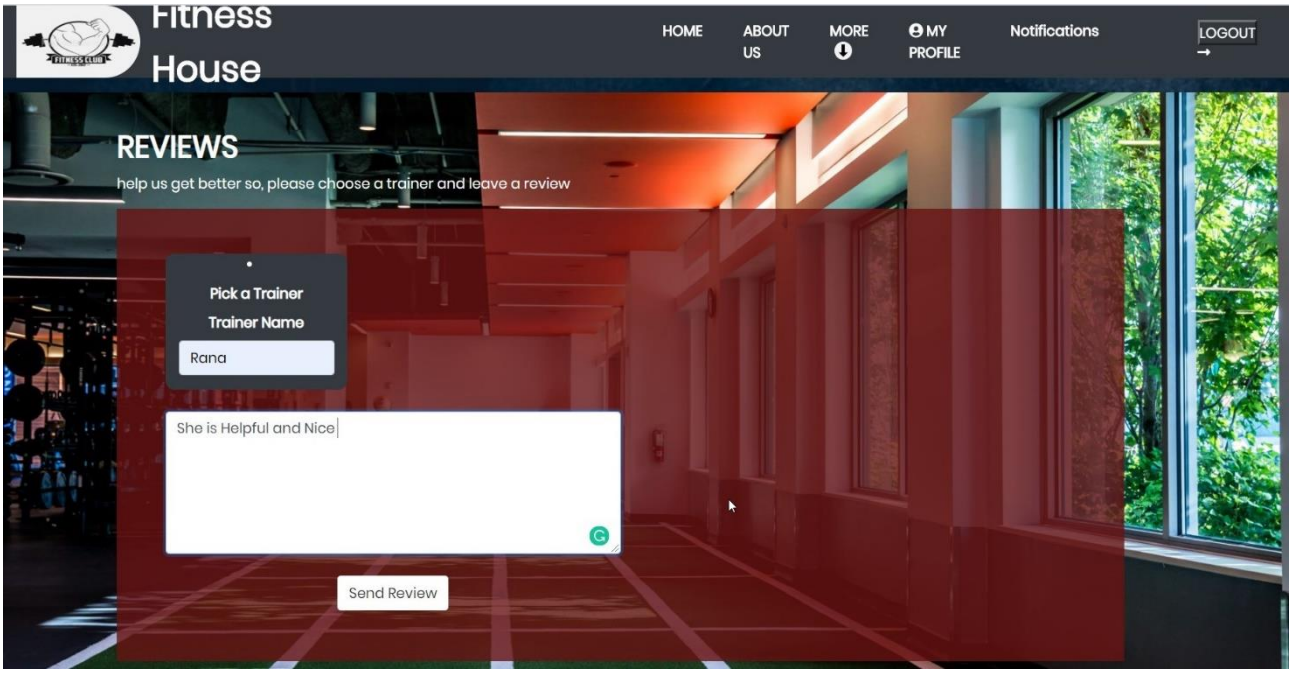
- ***Booking Sessions***



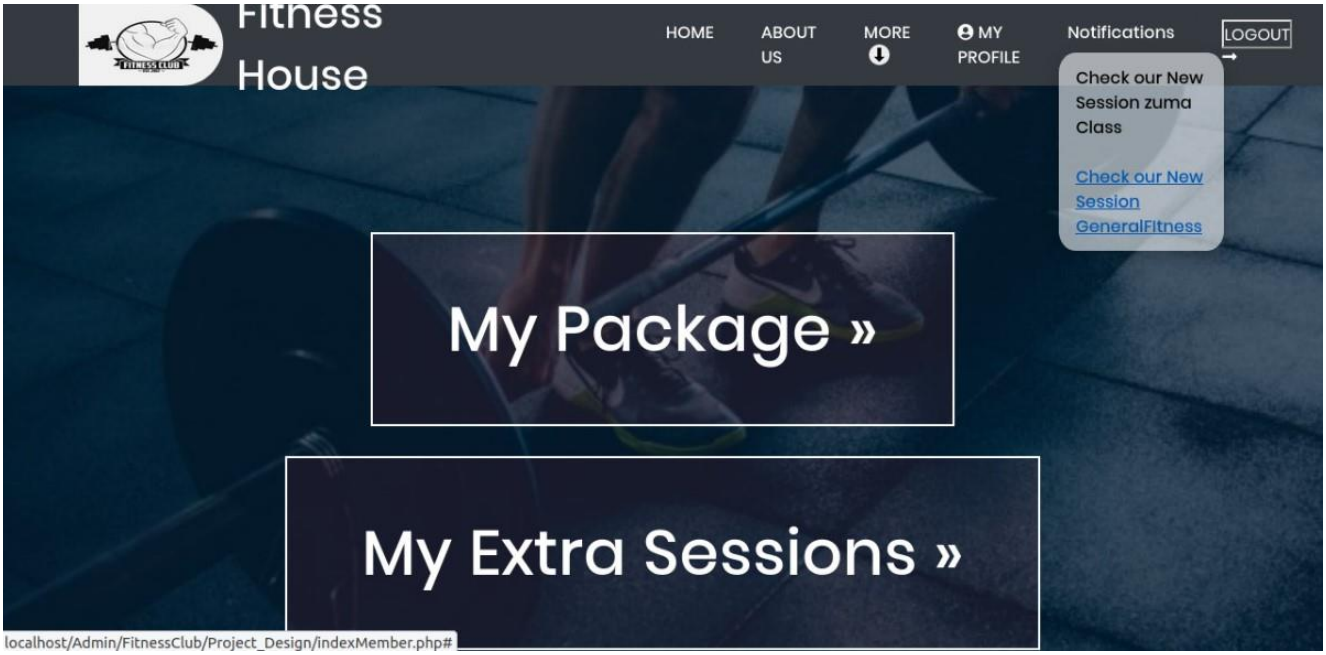
- ***Select Trainer of Session***



- ***Write Review***

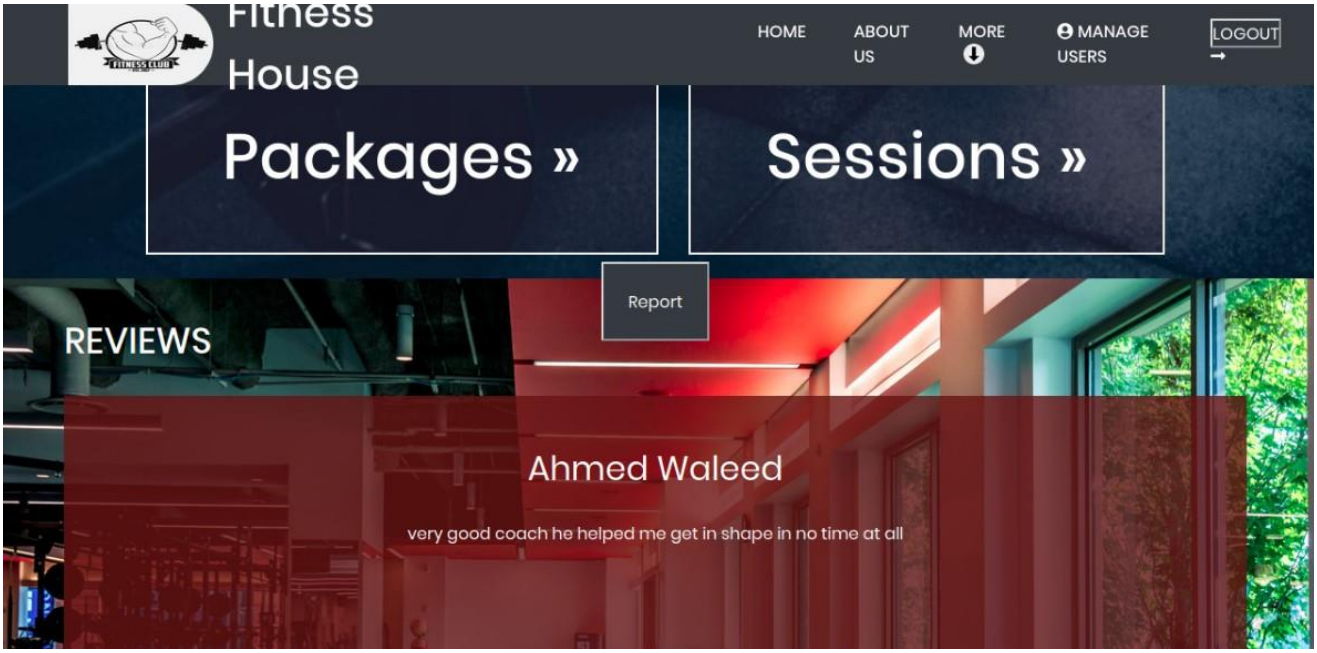


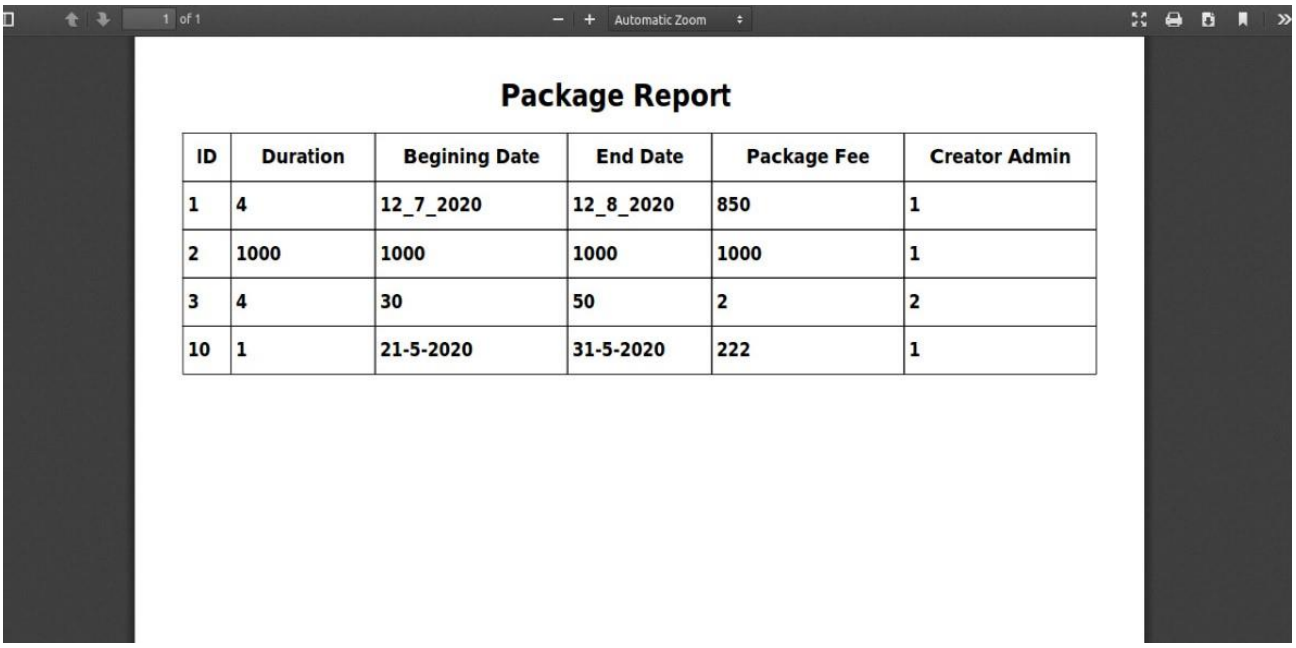
- **Notification in Member Module**



4) Admin Module:

- **Report as PDF**

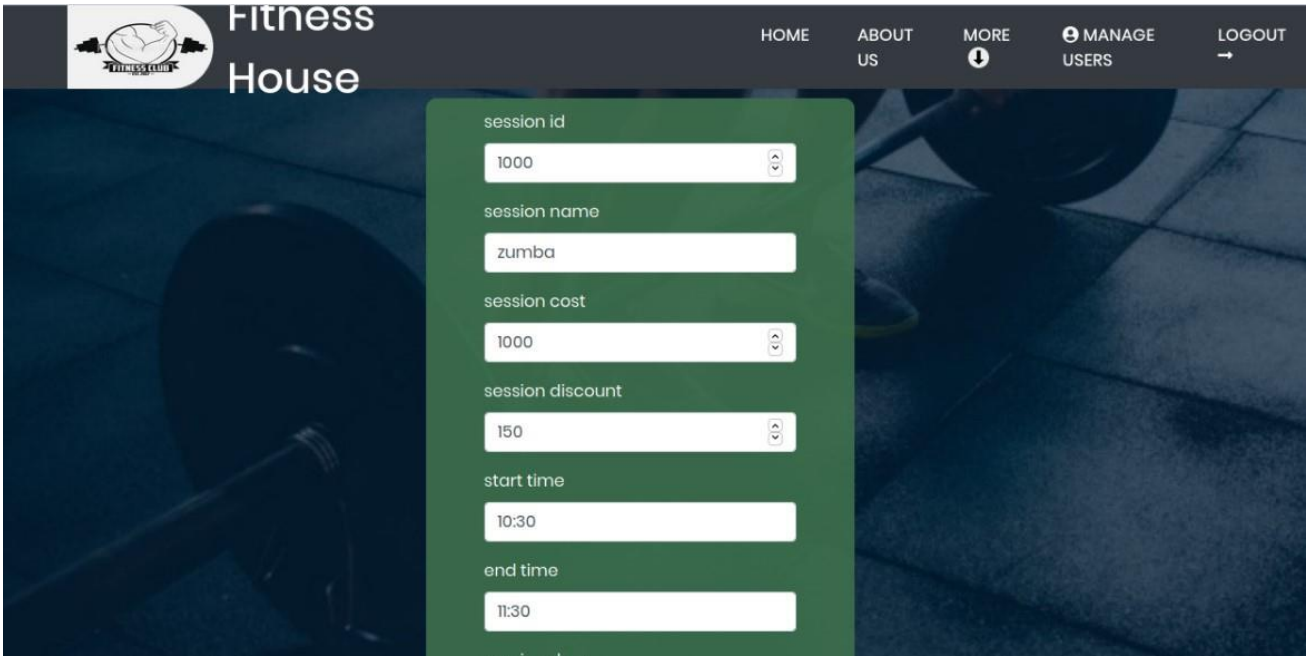




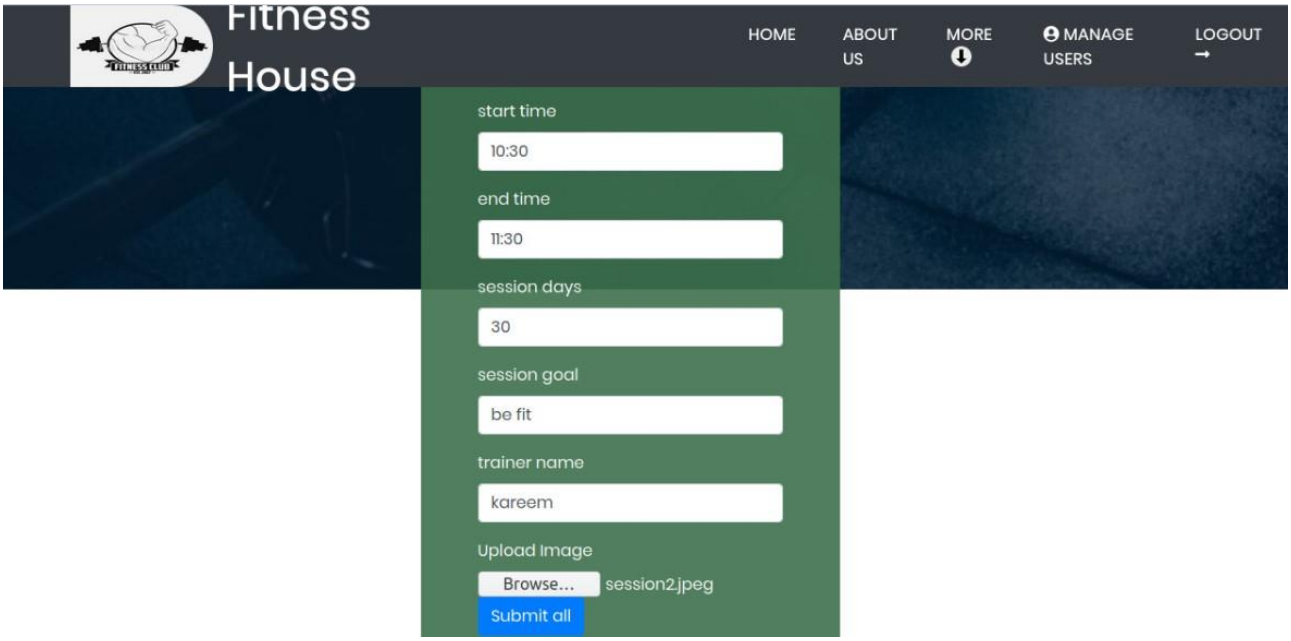
Package Report

ID	Duration	Begining Date	End Date	Package Fee	Creator Admin
1	4	12_7_2020	12_8_2020	850	1
2	1000	1000	1000	1000	1
3	4	30	50	2	2
10	1	21-5-2020	31-5-2020	222	1

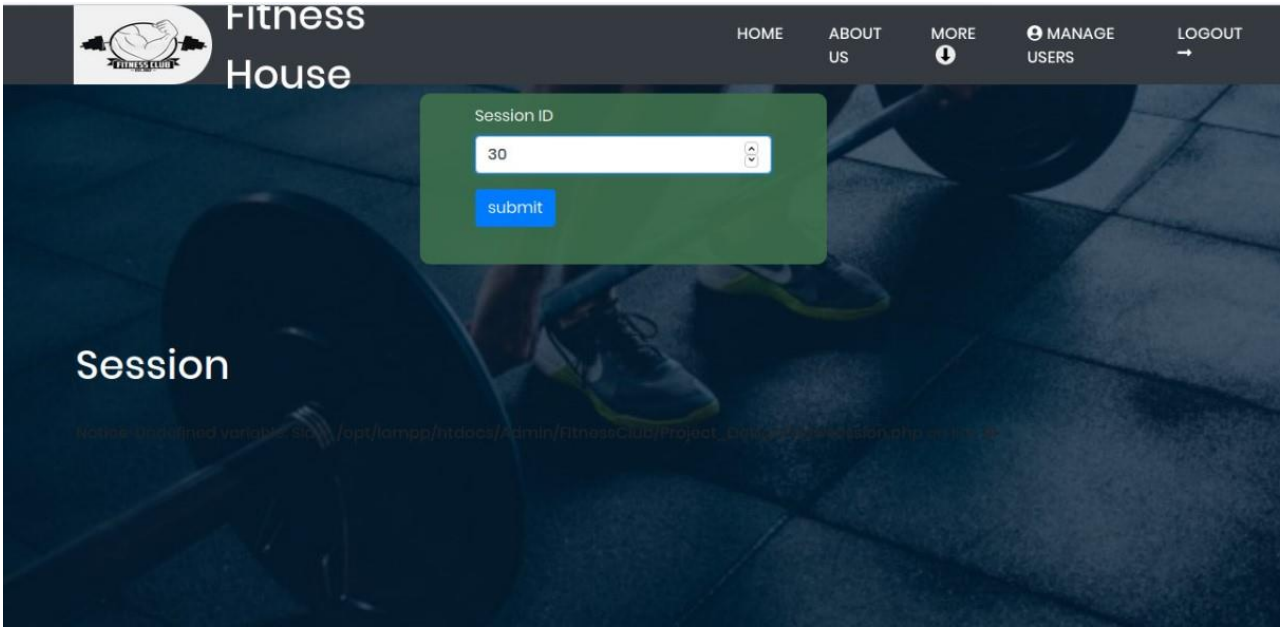
- **Manipulation of Product (Add/Update/Delete)**

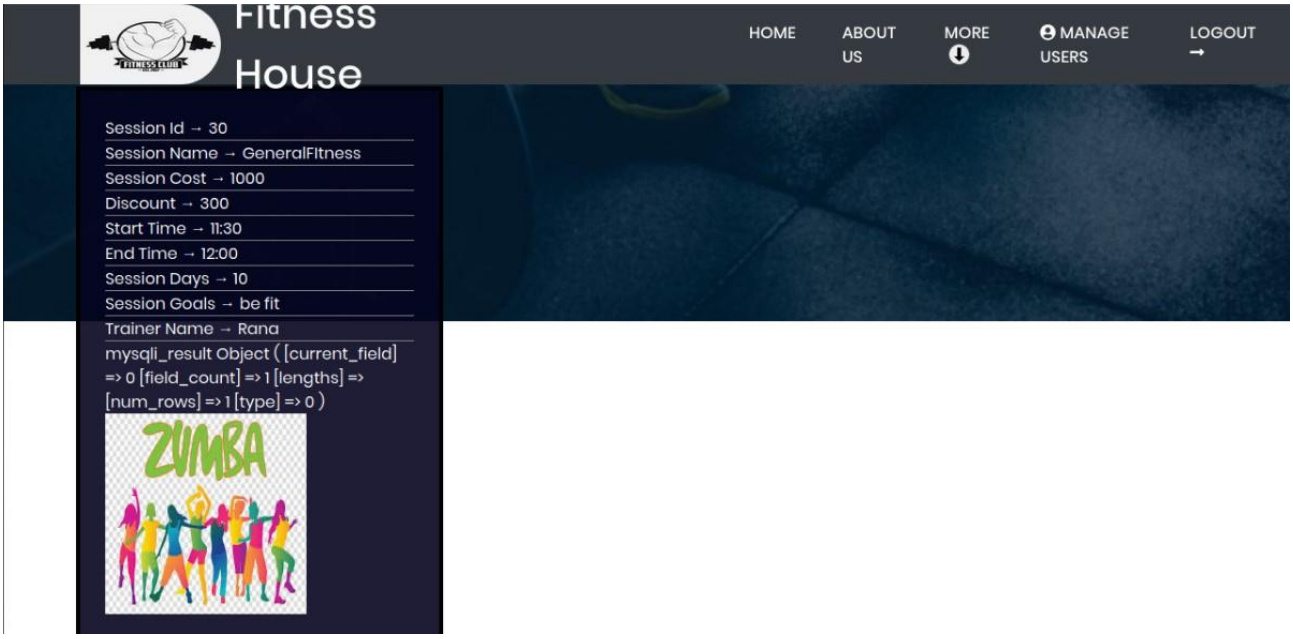


- **Upload Image In sessions**

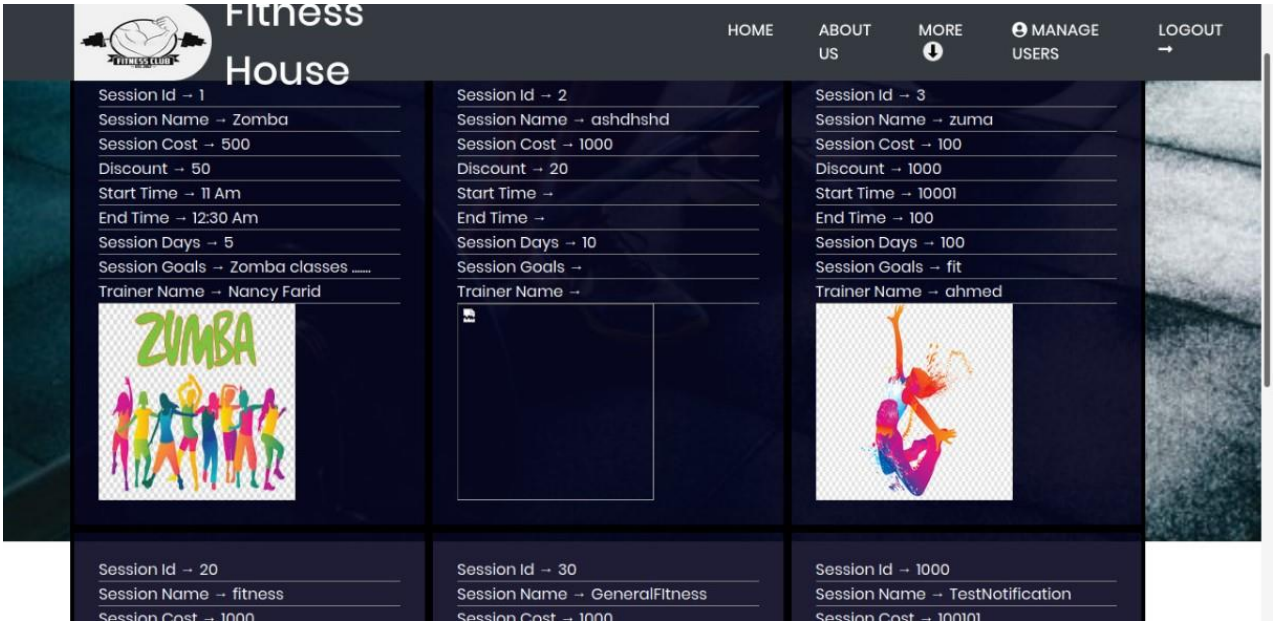
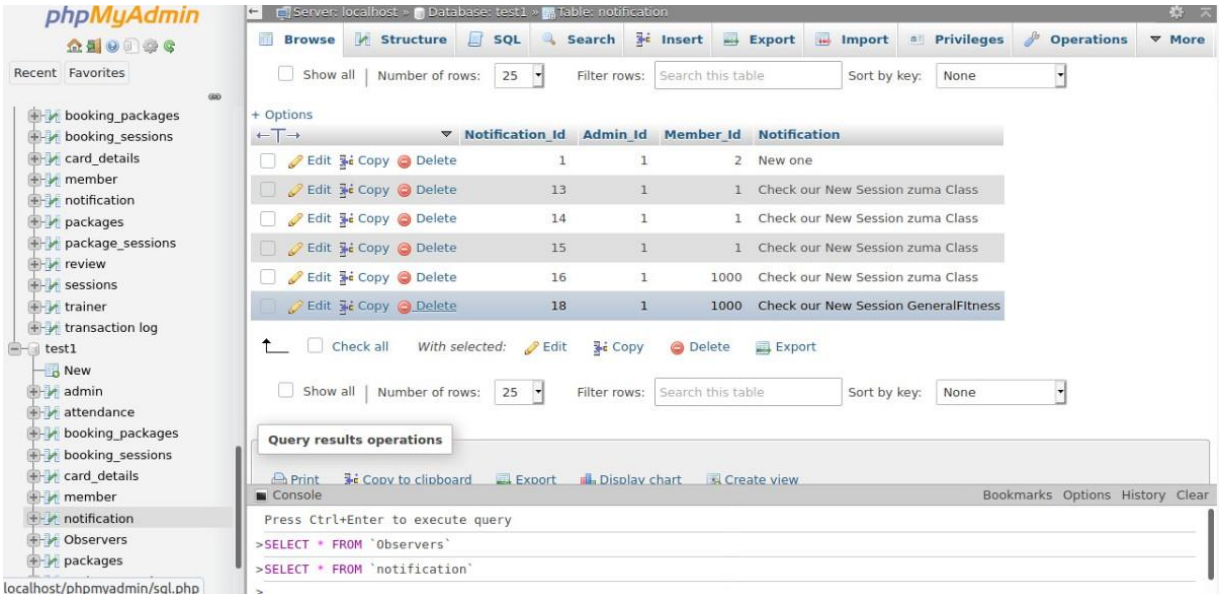


- **View Product**





- **List Products**



- **Notification Table**