Fitness Club Management System

Software Requirement specification Document

1)Introduction

1.1-Purpose

This document is written to describe the web app (Fitness Club Management System), interfaces, constrains, requirements and features.

1.2-Project scope

This web app will be used by admins, trainers and customers.

The scope of our project includes enrolling of a member in a Package in the fitness club and specifying to pay cash or credit, member can give feedback on his/her trainer,

Trainers can take members' attendance for the sessions every day and a trainer trains in only 1 package morning shift or evening shift.

Admins can view/add/delete users, search for users and view members' attendance taken by trainer.

Users can edit their profile. The project Scope also includes managing slot timing so system will allow a specific number of members to enroll in a session.

1.3-Glossary and Abbreviations

Word	Abbreviations & Glossary
Application	app
Software Requirement Pattern	SRP
User	Admin, Member, Trainer

1.4-List of system stakeholders

1-Admin

2-Member

3-Trainer

2)Functional Requirements

a) 2.1-User Requirements specification

1- Sign up

1.1 Member can sign up.

2-Login

2.1 Admin, Member and Trainer can login.

3-Logout

3.1 Admin, Member and Trainer can logout.

4-Package

4.1 Package details differs in number of months, number of jacuzzi, steam, sauna and spa sessions and differs in price.

4.2 Package may have discount

5-System

5.1 System allows a session to have maximum 30 member (session duration 2 hours)

Example: Not more than 30 members can enroll in class from 8:00 to 10:00

5.2 Sessions are from 8 am to 10 pm

5.3 System sends email with credentials when users are added by an admin.

6-Admin

6.1 Admin can add members.

6.2 Admin can delete members.

6.3 Admin can add trainers.

6.4 Admin can delete trainers.

6.5 Admin can modify members' session time.

6.6 Admin can change the shift of a trainer (morning or evening) and the package the trainer is training in.

6.7 Admin can add packages.

6.8 Admin can delete packages.

6.9 Admin can modify packages' information (number of months, number of jacuzzi, steam, sauna and spa sessions and price)

6.10 Admin can view the enrollment details of a member.

6.11 Admin can view shift details of trainer

6.12 Admin can view the member's attendance taken by the trainer

6.13 Admin can view and edit in (password/phone number)

6.14 Admin can add another admin

6.15 Admin can delete admin

6.16 Admin can view all admins

6.17 Admin can view all members

6.18 Admin can view all trainers

6.19 Admin can view all packages with details for each package

6.20 Admin can search for a member by typing name or email

6.21 Admin can search for a trainer by typing name or email

7-Member

7.1 Member can choose a package to enroll in and can view information of the package

7.2 Member can view and edit (profile picture/password/phone number)

7.3 Member can choose payment method (cash or online)

7.4 Member can choose a time to attend his session (from the available sessions)

7.5 Member can give a feedback on his/her trainer

7.6 Member can view list of trainers

7.7 Member can view list of packages

8-Trainer

8.1 Trainer can view and edit his/her (profile picture/password/phone number)

8.2 Trainer can take members' attendance daily.

8.3 Trainer should train in 1 package in morning or evening shift

b) 3.1-System Requirements specification

1- Sign up

1.1 Member can sign up by entering first name, last name, email, age, gender and password then press sign up button.

1.1.1 If any field is not filled an error is generated "this field is Required".

1.1.2 If the email already exists an error message is generated "email already exists".

1.1.3 If password is less than 8 characters an error is generated "password must be 8 characters or more".

1.1.4 If there are no errors the member would be directed to his home page.

2-Login

2.1 Admin, Member and Trainer can login by entering their email or password and pressing login button.

2.1.1 If email or password is wrong an error is generated "invalid email or password".

2.1.2 If any field is not filled an error is generated "this field is Required".

3-Logout

3.1 Admin, Member and Trainer can logout by pressing logout button.

4-Admin

4.1 Admin can add a member by entering first name, last name, email, age, gender and password then press sign up button.

4.1.1 If any field is not filled an error is generated "this field is Required".

4.1.2 If the email already exists an error message is generated "email already exists".

4.1.3 If password is less than 8 characters an error is generated "password must be 8 characters or more".

4.1.4 If there are no errors a message is generated "Member added successfully".

4.2 Admin can delete a member by viewing members from menu then click the delete button of the member you want to delete.

4.3 Admin can add a trainer by entering first name, last name, email, age, gender, password, choosing the trainer shift (morning or evening) and choosing the package number the trainer will train in then press Add trainer button.

4.3.1 If any field is not filled an error is generated "this field is Required".

4.3.2 If the email already exists an error message is

generated "email already exists".

4.3.3 If password is less than 8 characters an error is generated "password must be 8 characters or more".

4.3.4 If there are no errors a message is generated "Trainer added successfully".

4.4 Admin can delete a trainer by viewing all trainers from the menu then click the delete button of the trainer you want to delete.

4.5 Admin can modify members' session time through viewing all Members from view menu, click edit to modify, choose another session then click submit, then the admin will be directed to the page that views all members.

4.6 Admin can change the shift of a trainer (morning or evening) and the package the trainer is training in through viewing all trainers from menu, click edit to change the data then click submit

4.6.1 If there are members training with the trainer in his shift a message is generated "Cannot update information as there would be a conflict ".

4.6.2 If there are no members training with the trainer, the Shift and the package are changed successfully, then the admin will be directed to the page that views all trainer

4.7 Admin can add packages by filling price of package, discount (if exists, if no put zero), number of months, number of jacuzzi, steam, sauna and spa sessions, Package Number of Months and Package Description.

4.7.1 If any field is not filled an error is generated "this field is Required".

4.7.2 If there are no errors a message is generated "Package added successfully".

4.8 Admin can delete a package by viewing all packages from the menu then click the delete button of the package you want to delete.

4.9 Admin can modify packages' information (price of package, discount, number of months, number of jacuzzi, steam, sauna and spa sessions, package number of months and package description), then the admin will be directed to the page that views all packages.

4.10 Admin can view the enrollment details of members by viewing all members from menu.

4.11 Admin can view shift details of trainers by view all trainers from menu

4.12 Admin can view the member’s attendance taken by the trainer by clicking on view members' attendance taken by

trainer from menu then choosing trainer name, session time and date.

4.13 Admin can view his profile from menu by clicking view profile and can edit in (password/phone number) by clicking edit.

4.14 Admin can add another admin by entering first name, last name, email, age, gender and password then press Add admin button.

1.1.1 If any field is not filled an error is generated "this field is Required".

1.1.2 If the email already exists an error message is generated "email already exists".

1.1.3 If password is less than 8 characters an error is generated "password must be 8 characters or more".

4.3.5 If there are no errors a message is generated "Admin added successfully".

4.15 Admin can delete an admin by viewing all admins from the menu then click the delete button of the admin you want to delete.

4.16 Admin can view all admins from view menu.

4.17 Admin can view all members from view menu

4.18 Admin can view all trainers from view menu

4.19 Admin can view all packages from view menu

4.20 Admin can search for a member by viewing all members then typing name or email in search field then click search button

4.21 Admin can search a trainer by viewing all trainers then typing name or email in search field then click search button

4.22 Admin can search an admin by viewing all admins then typing name or email in search field then click search button

5-Member

5.1 Member can choose a package to enroll in by viewing all packages from menu click enroll button for chosen package, fill the information (choose session, payment method) then click submit.

5.1.1 If any field is not filled an error is generated "this field is Required".

5.1.2 If there are no errors, the member is directed to a page to view details about enrollment and payment details (cash or electronic)

5.2 Member can view his profile from menu by clicking view profile and can edit in (profile picture/password/phone number) by clicking edit.

5.3 Member can choose payment method (cash or electronic)

5.4 Member can choose a time to attend his/her session (from the available sessions)

5.5 Member can give a feedback on his/her trainer by clicking on give feedback then choose the trainer and write the feedback then click submit.

5.5.1 If no errors a message is generated "feedback recorded successfully"

5.5.2 If Member did not train with the chosen trainer then an error message is generated "Cannot give feedback on a trainer you did not train with"

5.6 Member can view a list of trainers by clicking on view trainers from the menu

5.7 Member can view a list of packages by clicking on view Trainers from the menu

8-Trainer

8.1 Trainer can view his profile from menu by clicking view profile and can edit in (profile picture/password/phone number) by clicking edit.

8.2 Trainer can take members' attendance daily by clicking on take members attendance in menu then choose the time of the session, click submit, then mark the members that

attended the session then submit.

8.2.1 If the attendance was taken before a message is generated "Attendance already taken"

8.2.2 If the trainer chose session that he/she did not train in a message is generated "you are not allowed to take attendance of this session"

8.3 Trainer should train in 1 package in morning or evening shift

9-Package

9.1 Package details differs in number of months, number of jacuzzi, steam, sauna and spa sessions and differs in price.

9.2 Package may have discount

10-System

10.1 System allows a session to have maximum 30 member (session duration 2 hours)

members in session <= 30

Example: Not more than 30 members can enroll in class from 8:00 to 10:00

10.2 Sessions are from 8 am to 10 pm

Sessions Time
From 8:00 to 10:00
From 10:00 to 12:00
From 12:00 to 14:00
From 14:00 to 16:00

From 16:00 to 18:00
From 18:00 to 20:00
From 20:00 to 22:00

10.3 System sends email with credentials when users are added by an admin.

c) Requirements ' Priorities (MoSCoW scheme)

3)Non-Functional Requirements

1-Performance

- System shall accommodate high number of users

- Responses to view information shall take no longer than 5 seconds to appear

-Quick access

2-Availability

-System is available for users 24 hours.

3- Security

-Software has many users and every user has access to specific information

4-Usability

-System is user friendly (easy to use), user can take 1 hour to get used with it

-System is accessed through the internet using web browser

5- Interface

-Interface text is understandable and convey the appropriate meaning

6-Effeciency

-Main time to repair, even if system fails it will recover and back up within an hour or less

7-Maintainability

-The Software system has a comfortable design that is easy to use.

8-Portability

-Well-Structured user manual

-Informative error messages.

-Well-formed graphical user interfaces

4) Design and Implementation Constraints

1- System shall use MySQL database.

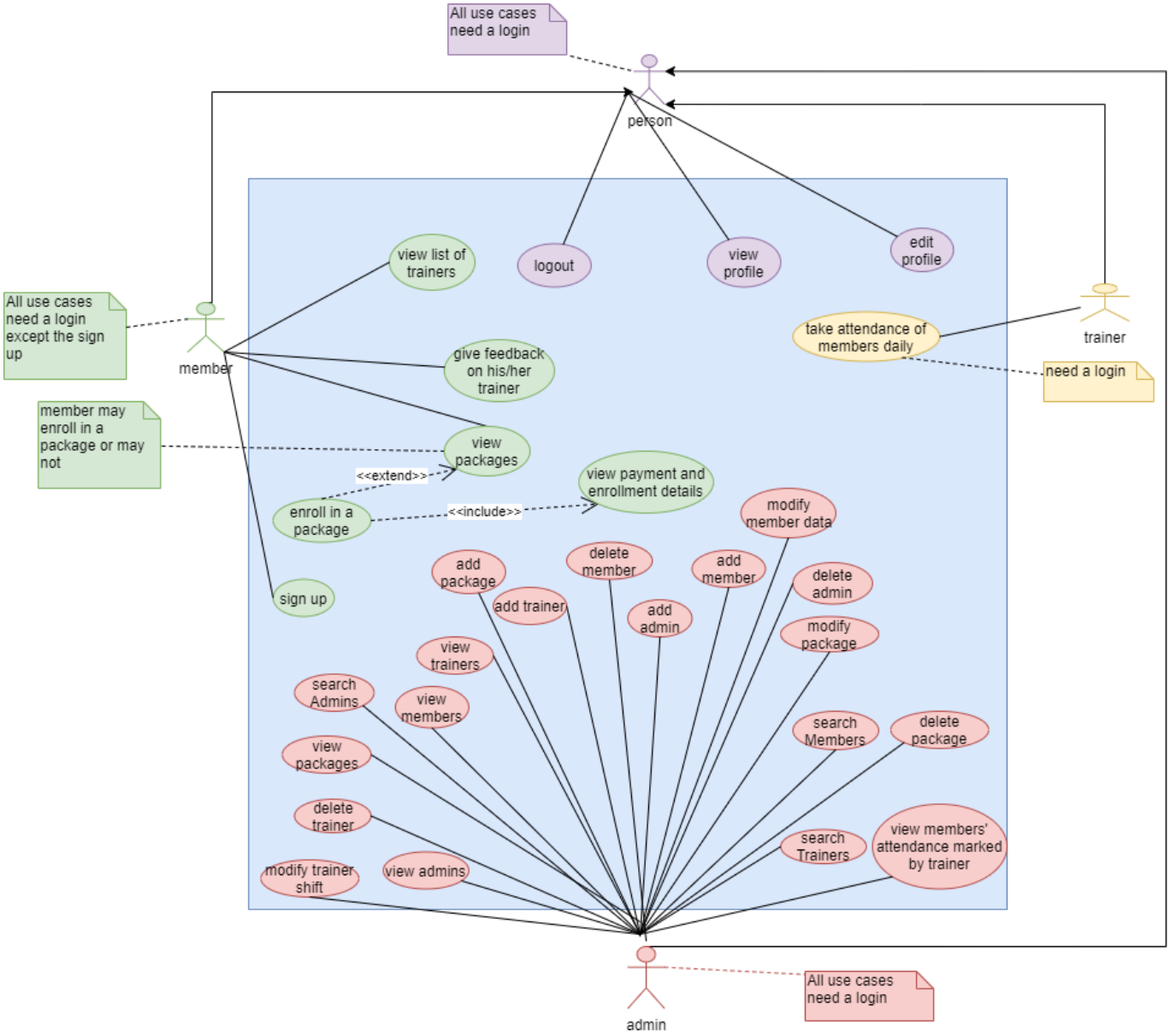
2- An Internet connection is required as it is a web-based application.

3- Xampp is used to record and read data from database.

4- PDF library is required.

5) Functional Diagrams

Use-Case Diagram



Detailed Use-Cases Description

Use case ID	UC 1
Use case name	Login
actors	Admin, Member, Trainer
Pre-condition	User has account
Basic flow	1-Open website 2-Enter Username and Password 3-Click on Login Button
Alternative flow	If username or password is incorrect: - an error is generated "invalid email or password", the user is prompted to re-enter information again. If username or password is not entered -System should generate an error "this field is Required".
Post-condition	User will be directed to his/her home page
Goal	Users login

Use case ID	UC 2
Use case name	Add member
actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-Login then click on member from add menu 2-Enter first name, last name, email, age, gender and password 3-Click on add member button
Alternative flow	If any information is not entered - System should generate an error "this field is Required". If email already exits - System should generate an error " email already exists ". If password is less than 8 characters - System should generate an error "password must be 8 characters or more"
Post-condition	Member is added

	- System should generate a message "Member added successfully".
Goal	Admin adds new member

Use case ID	UC 3
Use case name	Delete member
actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-choose members from view menu 2-search for the member you want to delete by typing his/her name or email 3-click on delete button
Alternative flow	none
Post-condition	Member is deleted
Goal	Admin deletes a member from the system

Use case ID	UC 4
Use case name	Add trainer
actors	Admin
Pre-condition	Admin is Logged Successfully
Basic flow	1-Login then click on member from add menu 2-Enter first name, last name, email, age, gender, password, choosing the trainer shift (morning or evening) and choosing the package number the trainer will train in 3-Click on add trainer button
Alternative flow	If any information is not entered - System should generate an error "this field is Required". If email already exits - System should generate an error " email already exists ". If password is less than 8 characters - System should generate an error "password must be 8 characters or more"

Post-condition	Trainer is added - System should generate a message "Trainer added successfully".
Goal	Admin adds a trainer

Use case ID	UC 5
Use case name	Delete trainer
actors	Admin
Pre-condition	Admin is Logged Successfully
Basic flow	1-choose trainers from view menu 2-search for the trainer you want to delete by typing his/her name or email 3-click on delete button
Alternative flow	none
Post-condition	Trainer is deleted
Goal	Admin deletes a trainer from the system

Use case ID	UC 6
Use case name	Add Admin
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-Login then click on member from add menu 2-Enter first name, last name, email, age, gender and password 3-Click on add admin button
Alternative flow	If any information is not entered - System should generate an error "this field is Required". If email already exits - System should generate an error " email already exists ". If password is less than 8 characters - System should generate an error "password must be 8 characters or more"
Post-condition	Admin is added

	- System should generate a message "Admin added successfully".
Goal	Admin adds an admin to the system

Use case ID	UC 7
Use case name	Sign up
Actors	Member
Pre-condition	none
Basic flow	1-open website 2-click on sign up button 3-Enter all information required 4-Click on sign up button
Alternative flow	If some information is not entered -Member is asked to enter all information
Post-condition	Signed up successfully Member will be directed to his/her home page
Goal	Member signs up and a new account is created

Use case ID	UC 8
Use case name	Modify member data
actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-choose members from view menu 2-search for the member you want to modify his/her information by typing his/her name or email 3-click on edit button 4-edit information then click submit
Alternative flow	none
Post-condition	Information updated successfully Admin is directed to the page that views all members
Goal	Admin modify member's data

Use case ID	UC 9
Use case name	Modify trainer data
actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-choose trainers from view menu 2-search for the trainer you want to modify his/her information by typing his/her name or email 3-click on edit button 4-edit information then click submit
Alternative flow	If changing the trainer shift or the package working in will make conflict (members train with him/her) -trainer's shift won't be updated
Post-condition	Shift of trainer updated successfully Admin is directed to the page that views all trainers
Goal	Admin changes trainer's shift

Use case ID	UC 10
Use case name	Add package
actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-Login then click on package from add menu 2-Enter price of package, discount, number of months, number of jacuzzi, steam, sauna and spa sessions, Package Number of Months and Package Description 3-Click on add package button
Alternative flow	If any information is not entered - System should generate an error "this field is Required".
Post-condition	Package is added - System should generate a message "Package added successfully".
Goal	Admin adds a new package to the system

Use case ID	UC 11
Use case name	Delete package

actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-choose packages from view menu 2-click on delete button of the package you want to delete
Alternative flow	none
Post-condition	Package is deleted
Goal	Admin deletes a package from the system

Use case ID	UC 12
Use case name	Modify package
actors	Admin
Pre-condition	Admin is Logged in Successfully
Basic flow	1-choose packages from view menu 2-click on edit button of the package you want to edit its information 3-edit information then click submit
Alternative flow	If any information is erased and is not entered - System should generate an error "this field is Required".
Post-condition	Package is modified Admin will be directed to the page that views all packages
Goal	Admin modify package's information

Use case ID	UC 13
Use case name	Edit Profile
actors	Admin, member, trainer
Pre-condition	Logged in
Basic flow	1-Click view profile 2- click on edit 3-update information

	4-click submit
Alternative flow	If any field is left empty -error message is generated "this field is required"
Post-condition	Profile is edited User is directed to view profile page
Goal	User edits his/her profile

Use case ID	UC 14
Use case name	View trainers
actors	Member, Admin
Pre-condition	Logged in successfully
Basic flow	1-choose trainers from view menu
Alternative flow	none
Post-condition	Trainers are viewed

Use case ID	UC 15
Use case name	View packages
actors	Member, Admin
Pre-condition	Logged in successfully
Basic flow	1-choose packages from view menu
Alternative flow	none
Post-condition	Packages are viewed
Goal	Admin or member views all the packages in the system

Use case ID	UC 16
Use case name	View members
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-choose members from view menu
Alternative flow	none
Post-condition	Members are viewed
Goal	Admin views all members in the system

Use case ID	UC 17
Use case name	View admins
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-choose admins from view menu
Alternative flow	none
Post-condition	Admins are viewed
Goal	Admin views all admins in the system

Use case ID	UC 18
Use case name	Give feedback on trainers
actors	Member
Pre-condition	Logged in successfully
Basic flow	1-click Give feedback 2-choose a trainer and write the feedback
Alternative flow	If any information is not entered - System should generate an error "this field is Required". If the member did not train with this trainer -System should generate an error "you can not give feedback on a trainer you did not train with"
Post-condition	Feedback recorded successfully
Goal	Member gives a feedback on the trainer he/she trains with

Use case ID	UC 19
Use case name	Take attendance of members
actors	Trainer
Pre-condition	Logged in successfully
Basic flow	1-click take members' attendance 2-choose session time then click submit 3-mark the members that attended the session then click submit

Alternative flow	If any information is not entered - System should generate an error "this field is Required". If attendance is taken -System should generate an error message "attendance already taken"
Post-condition	Attendance of members are taken
Goal	Trainer records the members' attendance (attended the session or not)

Use case ID	UC 20
Use case name	Search trainers
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-click trainers from view menu 2-search for the trainer you want his/her information by typing his/her name or email
Alternative flow	none
Post-condition	System views the results of the search
Goal	Admin searches for a trainer by typing his/her name or email

Use case ID	UC 21
Use case name	Search members
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-click members from view menu 2-search for the member you want his/her information by typing his/her name or email
Alternative flow	none
Post-condition	System views the results of the search
Goal	Admin searches for a member by typing his/her name or email

Use case ID	UC 22
Use case name	Search Admins
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-click admins from view menu 2-search for the admin you want his/her information by typing his/her name or email
Alternative flow	none
Post-condition	System views the results of the search
Goal	Admin searches for an admin by typing his/her name or email

Use case ID	UC 23
Use case name	View member’s attendance taken by trainer
actors	Admin
Pre-condition	Logged in successfully
Basic flow	1-click view member’s attendance 2-choose the trainer's name, date and session time 3-click submit
Alternative flow	If any information is not entered - System should generate an error "this field is Required". If the trainer did not train in this time -System will generate a message "No results to show"
Post-condition	Members attendance is viewed
Goal	Admin views the attendance of the members that the trainer takes

Use case ID	UC 24
Use case name	View profile
actors	Admin, member, trainer
Pre-condition	Logged in successfully
Basic flow	1-click view profile

Alternative flow	none
Post-condition	Profile viewed
Goal	User views his/her profile

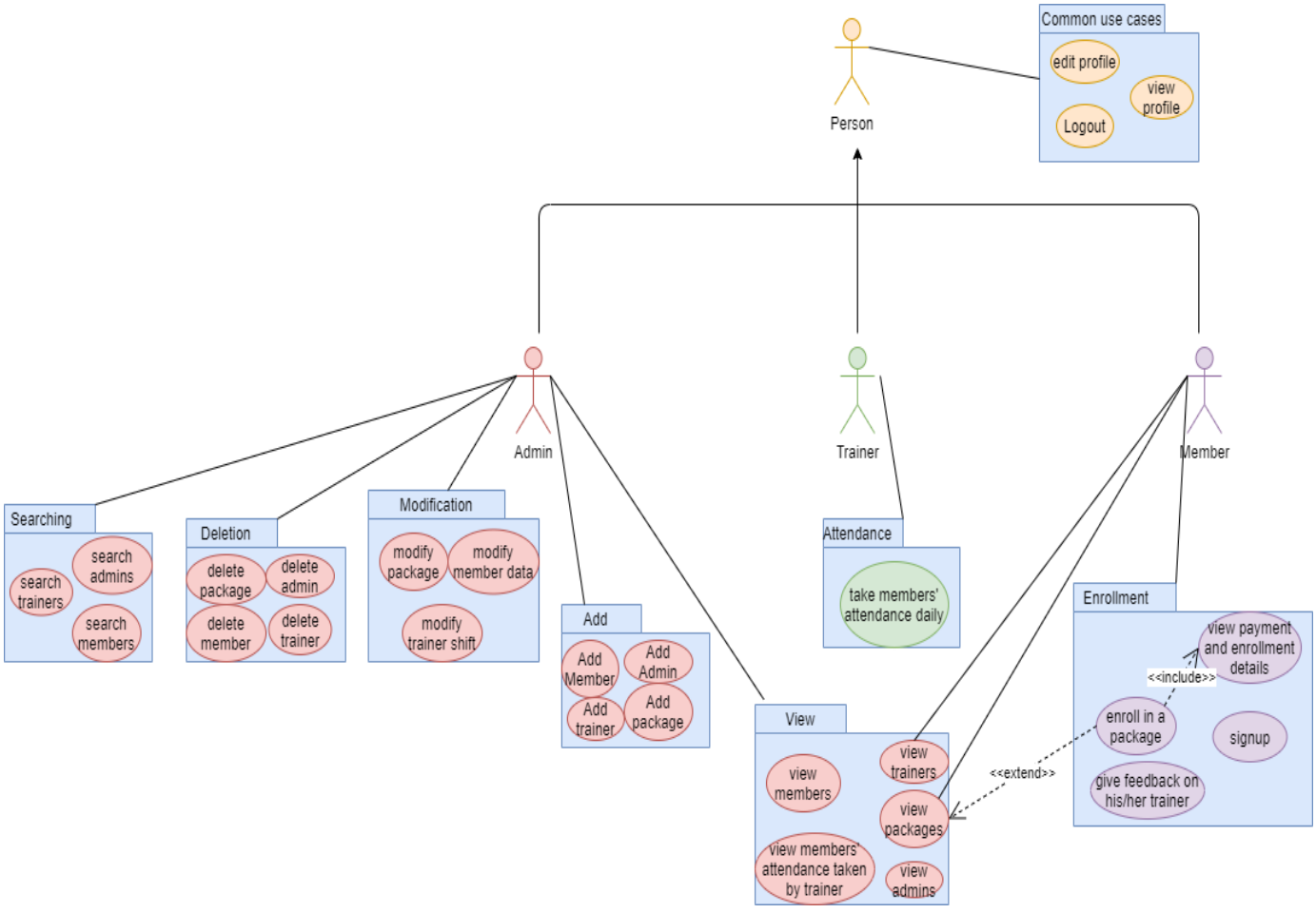
Use case ID	UC 25
Use case name	Logout
actors	Admin, member, trainer
Pre-condition	Logged in successfully
Basic flow	1-click on logout button
Alternative flow	none
Post-condition	Logged out
Goal	User logout

Use case ID	UC 26
Use case name	Enrollment in a package
actors	member
Pre-condition	Logged in successfully
Basic flow	1-click packages from view packages 2-click enroll on the package to be chosen 3-choose the session time and the payment method then click submit
Alternative flow	If member is currently enrolling in a package -System should generate an error message "you can not enroll in a package"
Post-condition	Enrolled in a package Member is directed to a page that contains information about the enrollment and payment method
Goal	Member enrolls in a package

Use case ID	UC 27
Use case name	Set profile picture
actors	member, trainer, admin
Pre-condition	Logged in successfully

Basic flow	1-click view profile 2-click edit profile 3- click browse and choose a photo
Alternative flow	none
Post-condition	Profile picture uploaded
Goal	User uploads a picture to his/her profile photo

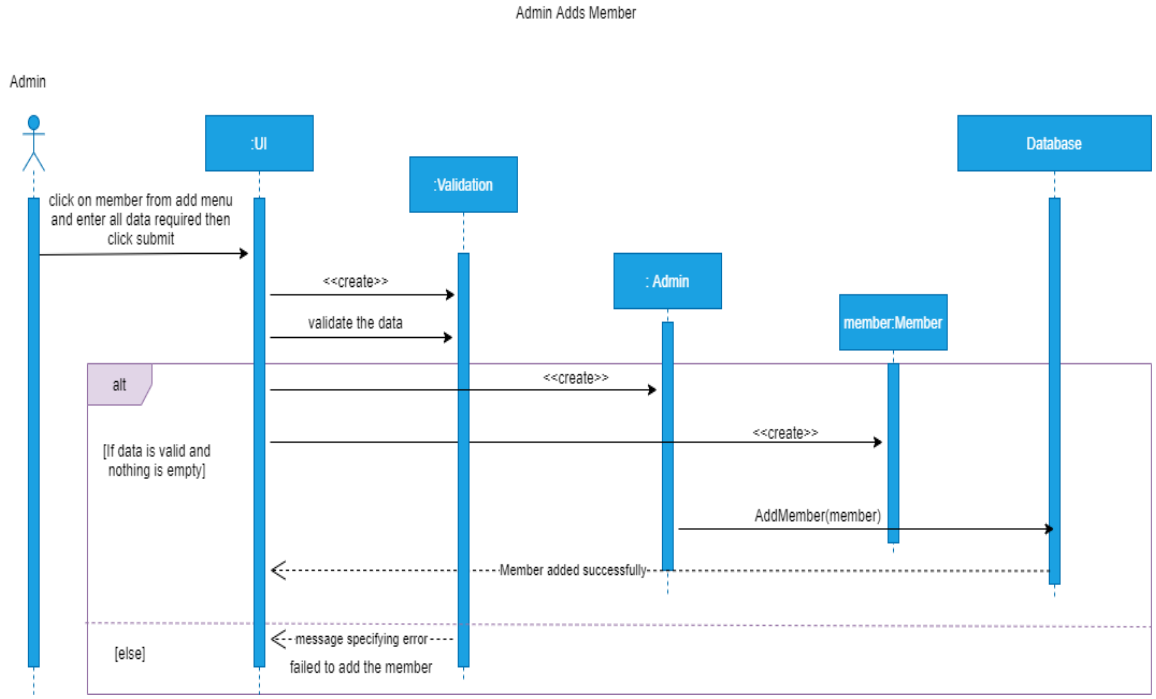
Package diagram grouping relevant Use-Cases into Packages



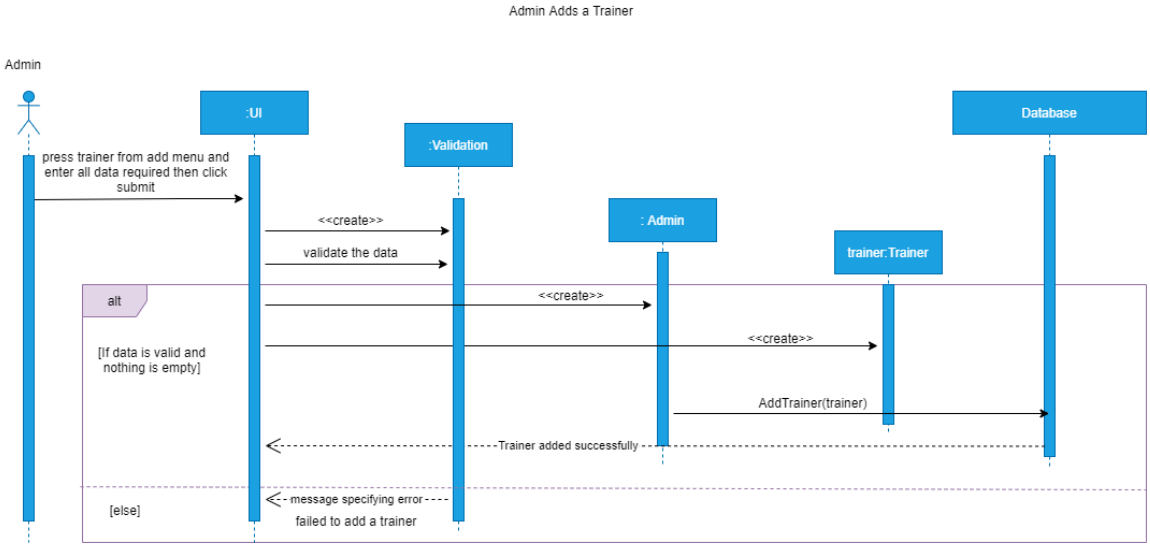
6) Structural and Behavioural Diagrams

Sequence Diagrams

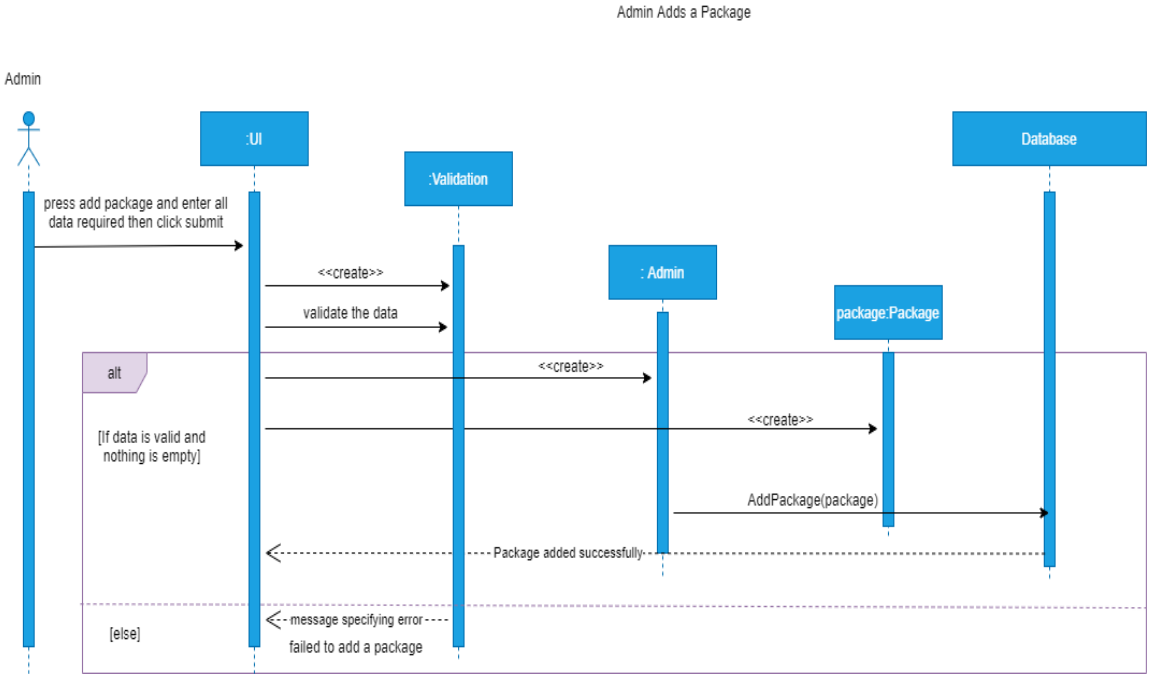
1-Admin adds Member



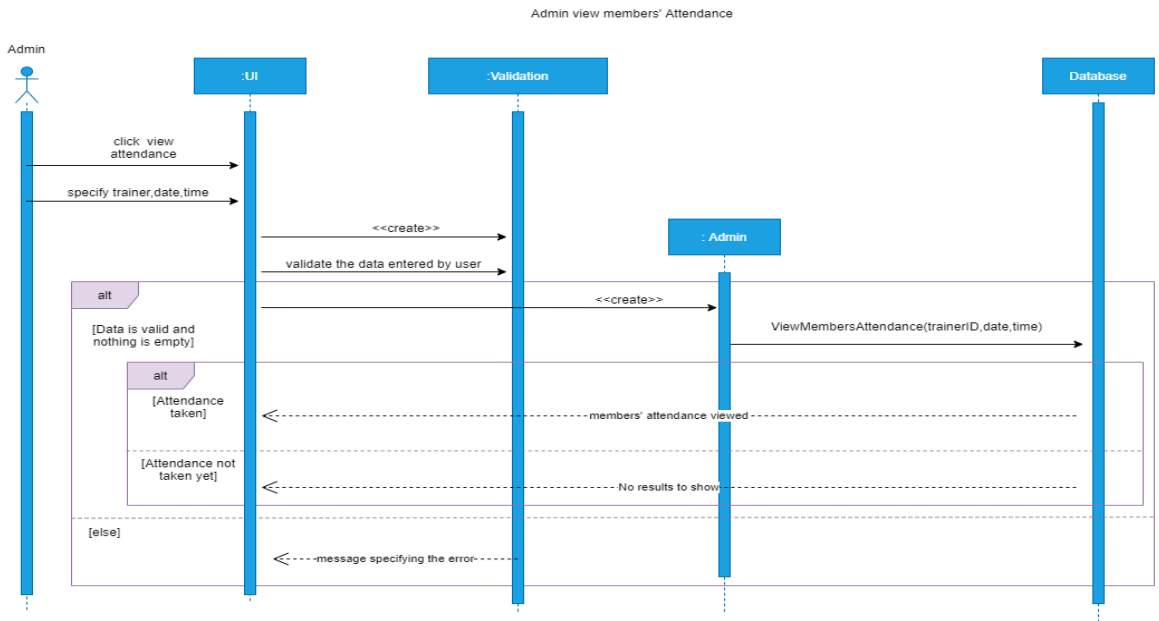
2-Admin adds a trainer



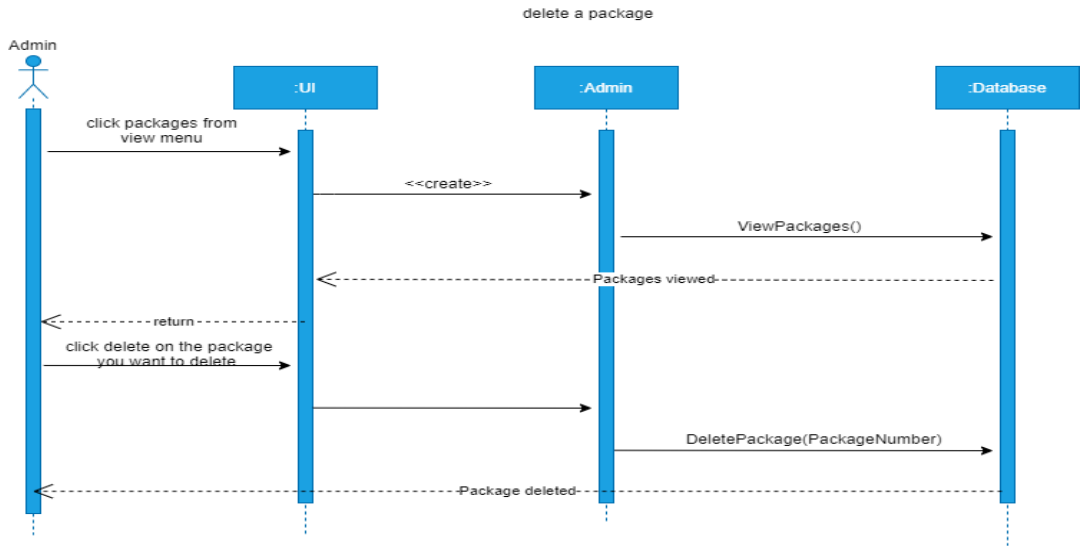
3-Admin adds a package



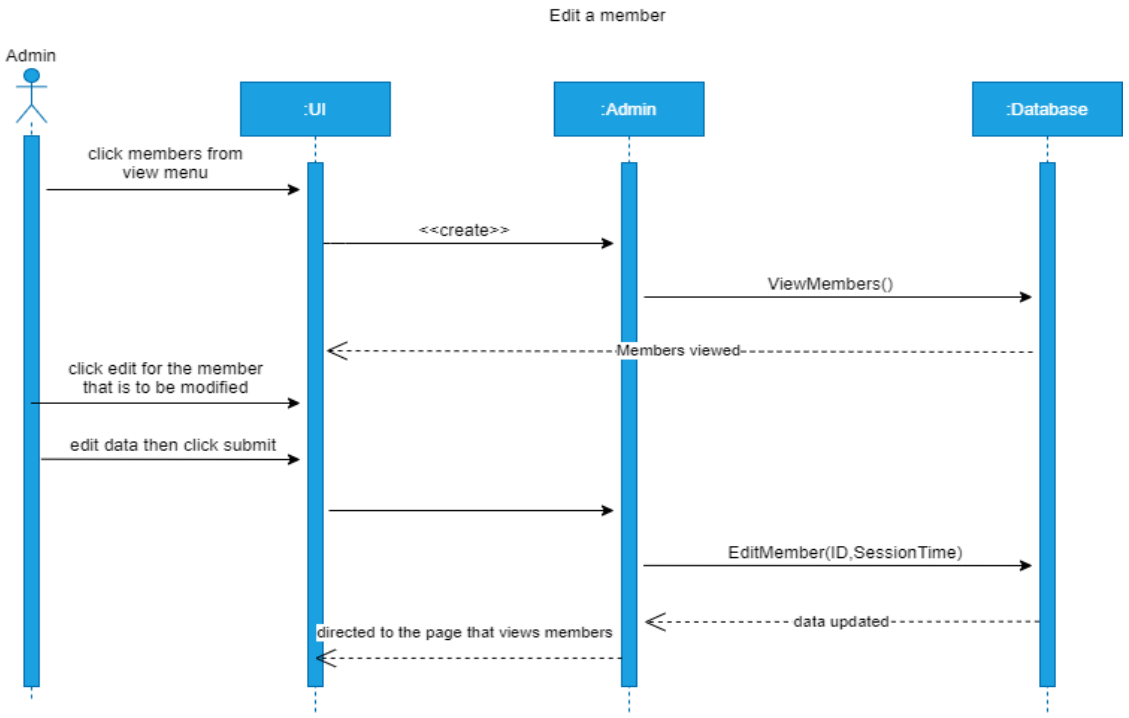
4-Admin view members' attendance



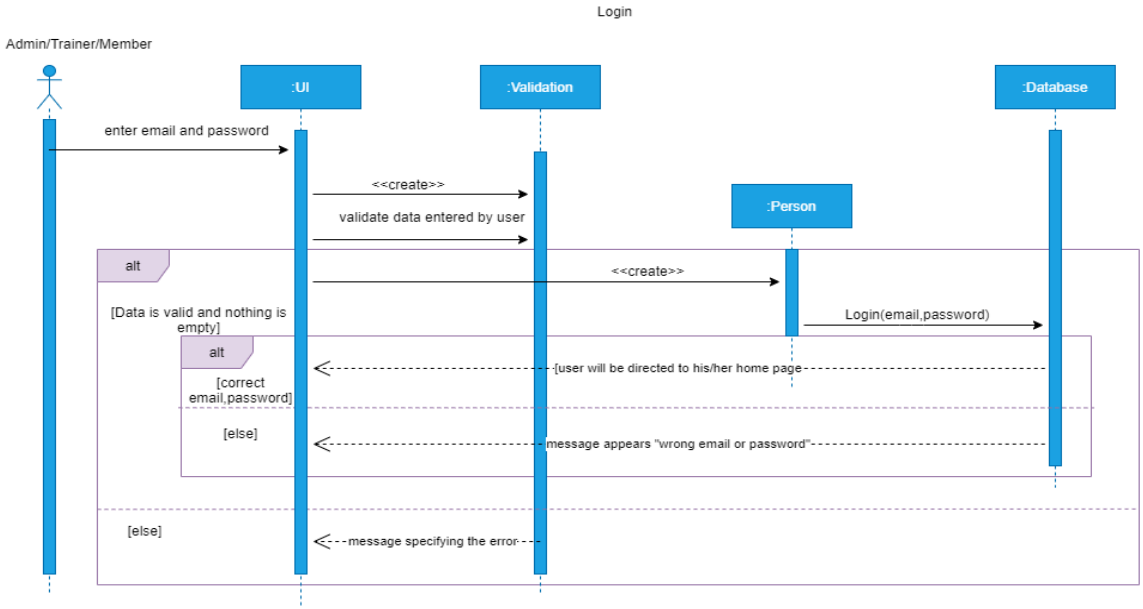
5-Admin deletes a package



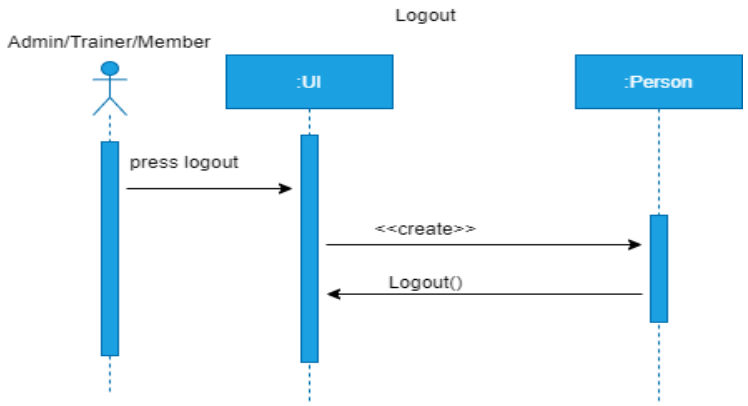
6-Admin edits member's data



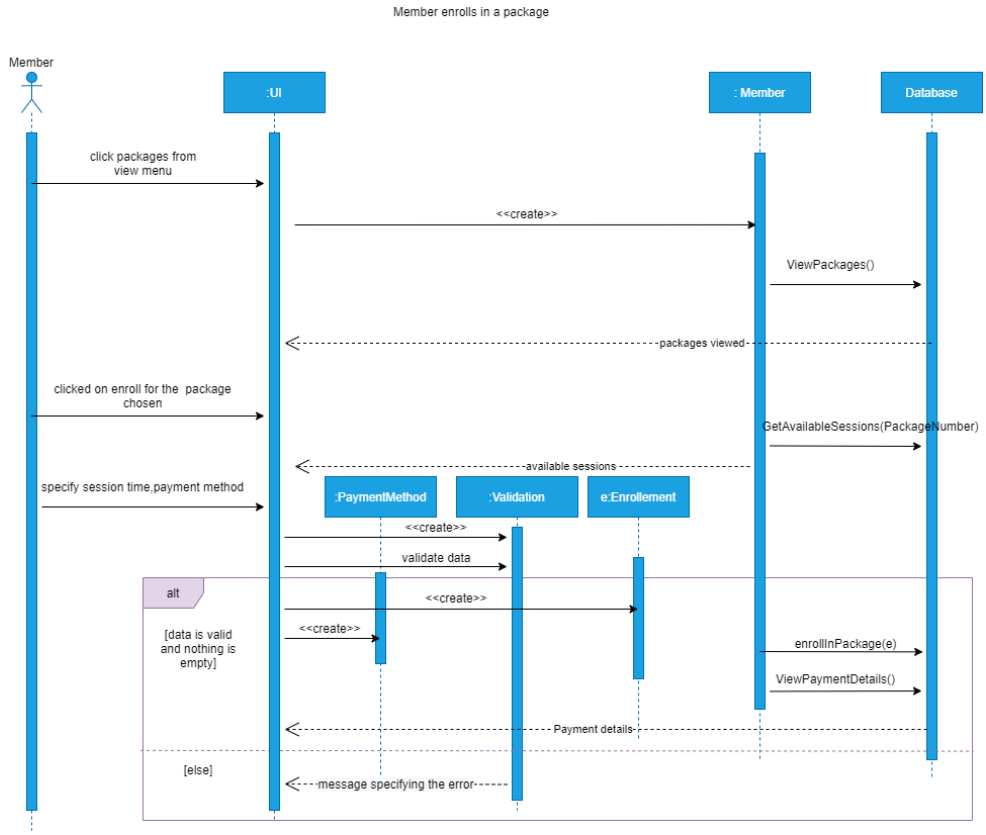
7-Login



8-Logout

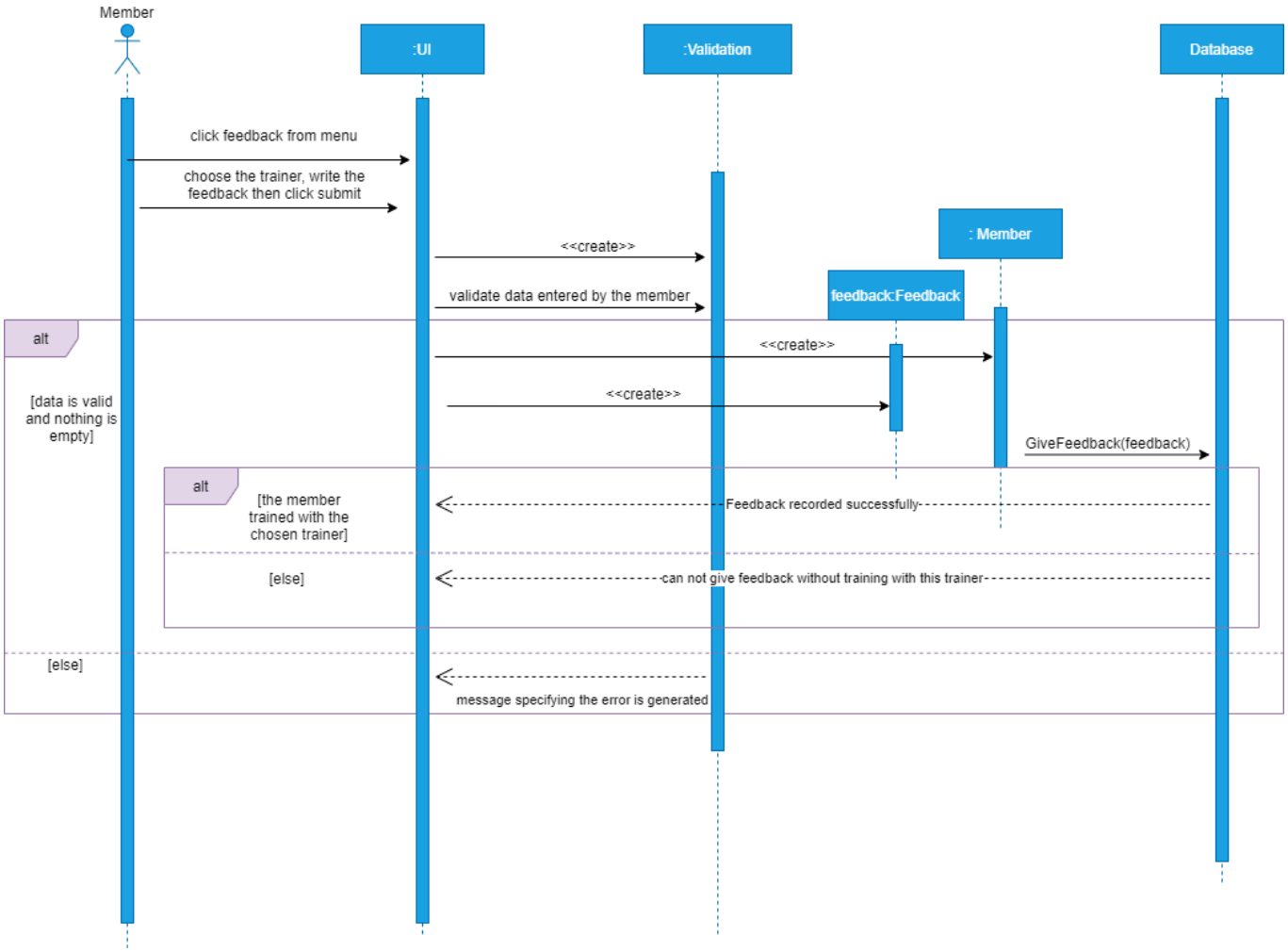


9-Member enrolls in a package

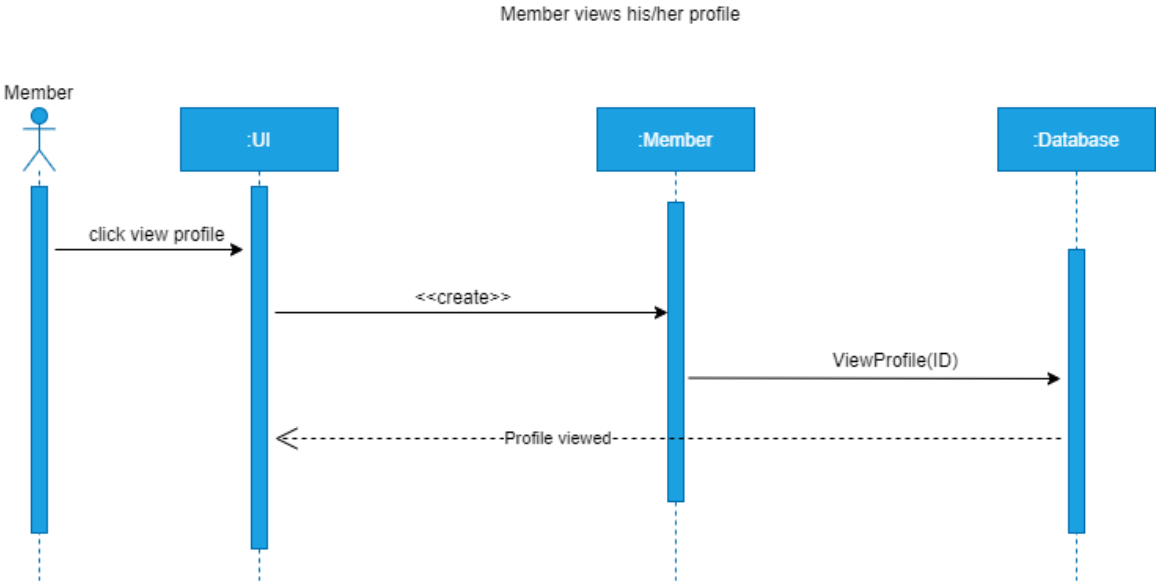


10-Member gives feedback

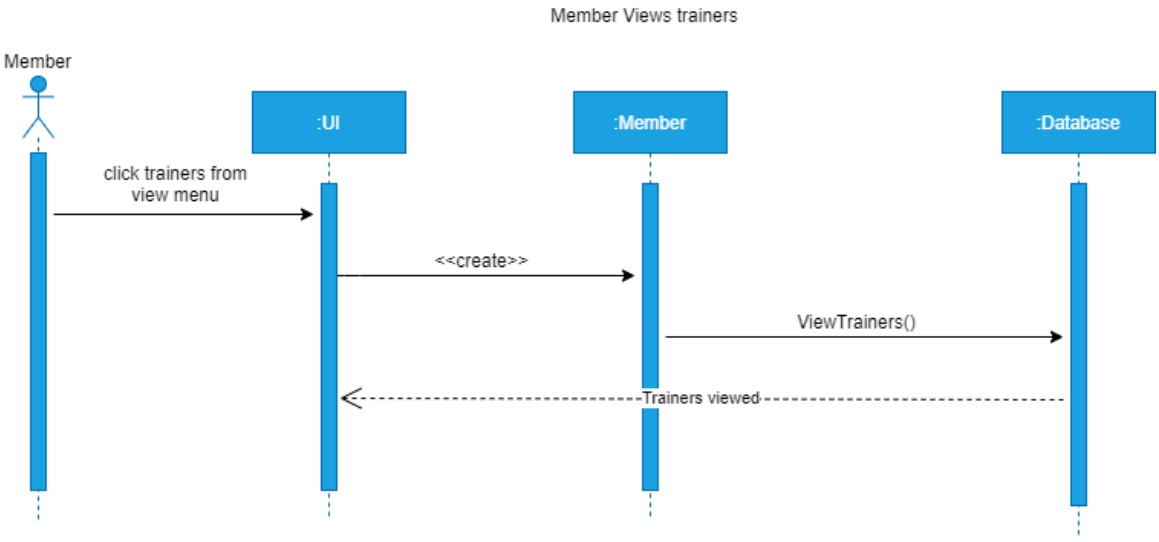
Member gives feedback



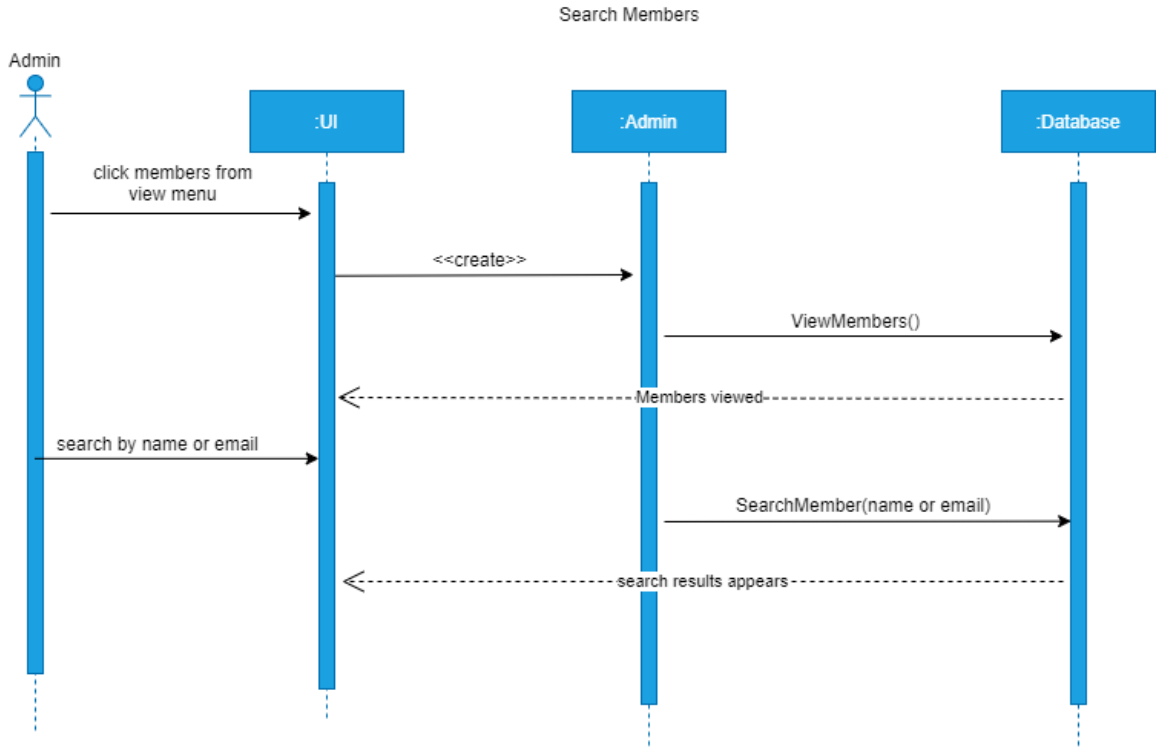
11-Member views his/her profile



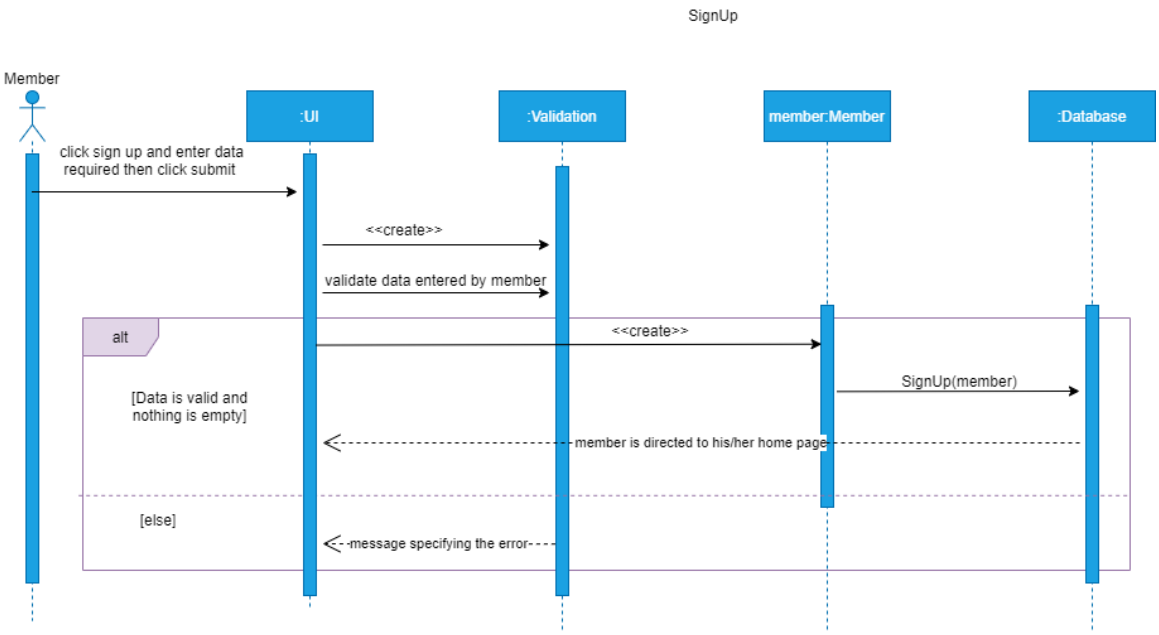
12-Member views all trainers



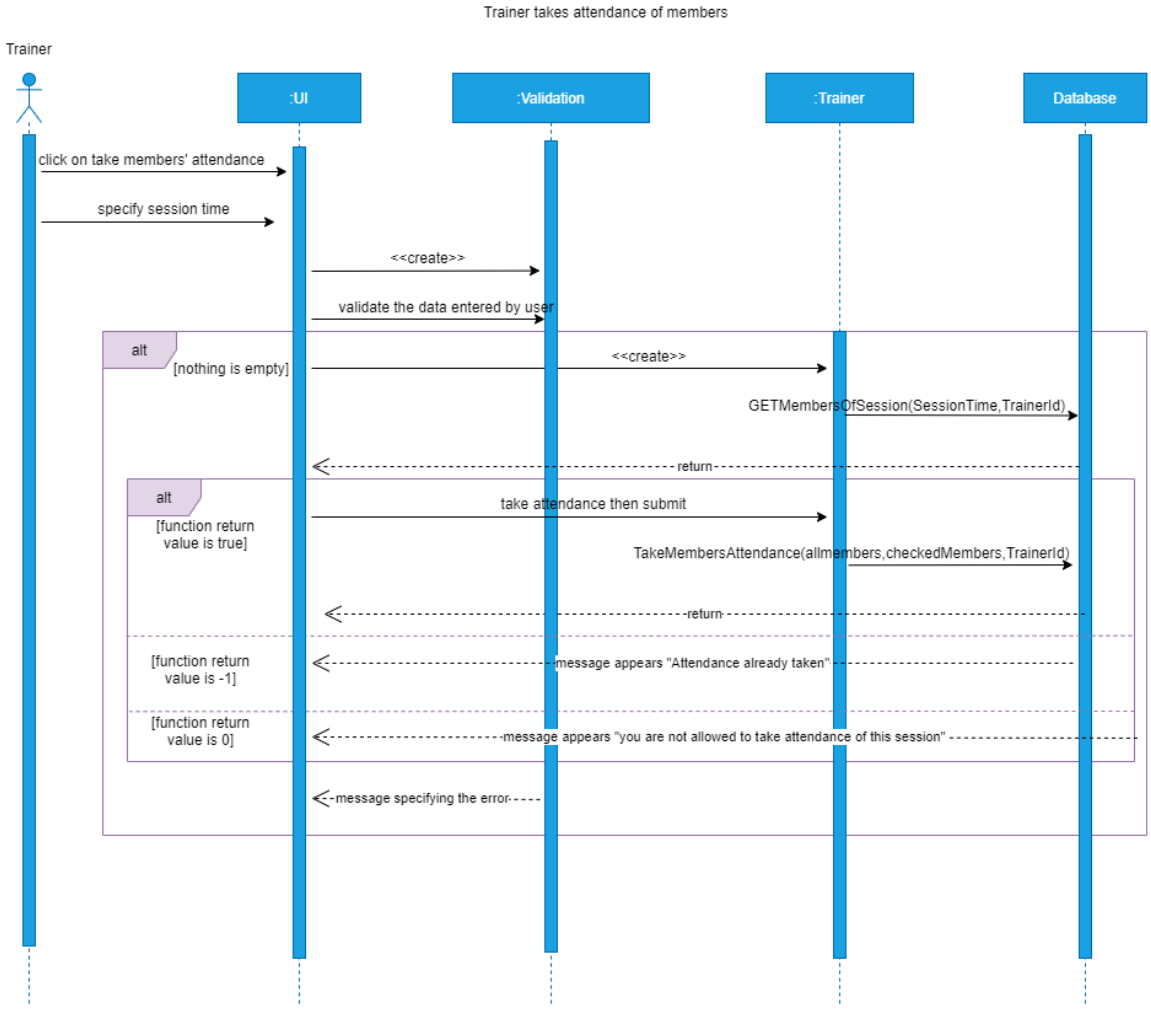
13-Admin searches for a member



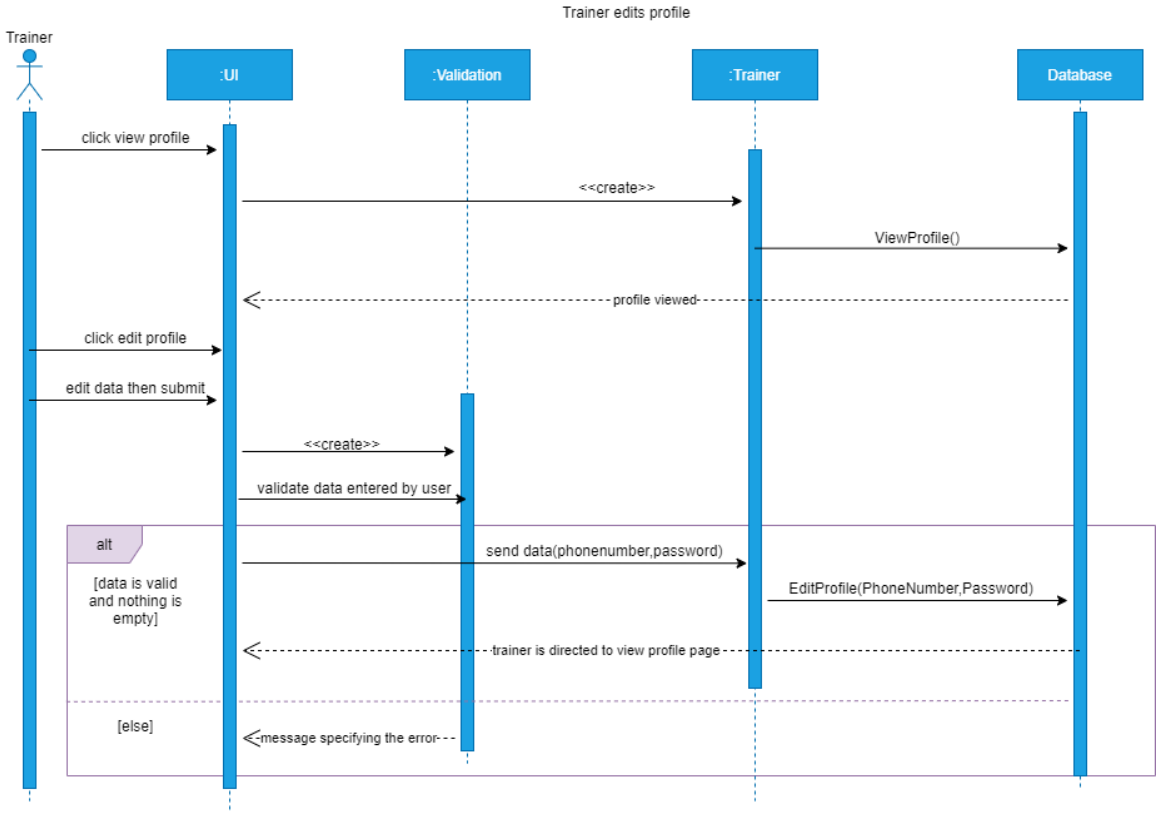
14-Signup



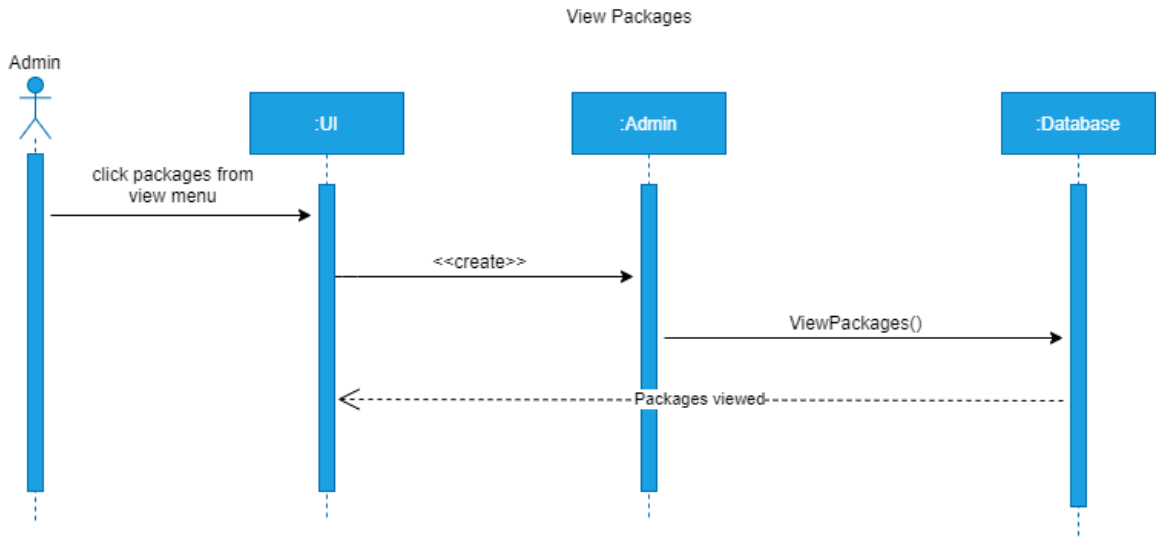
15-Trainer takes attendance of members



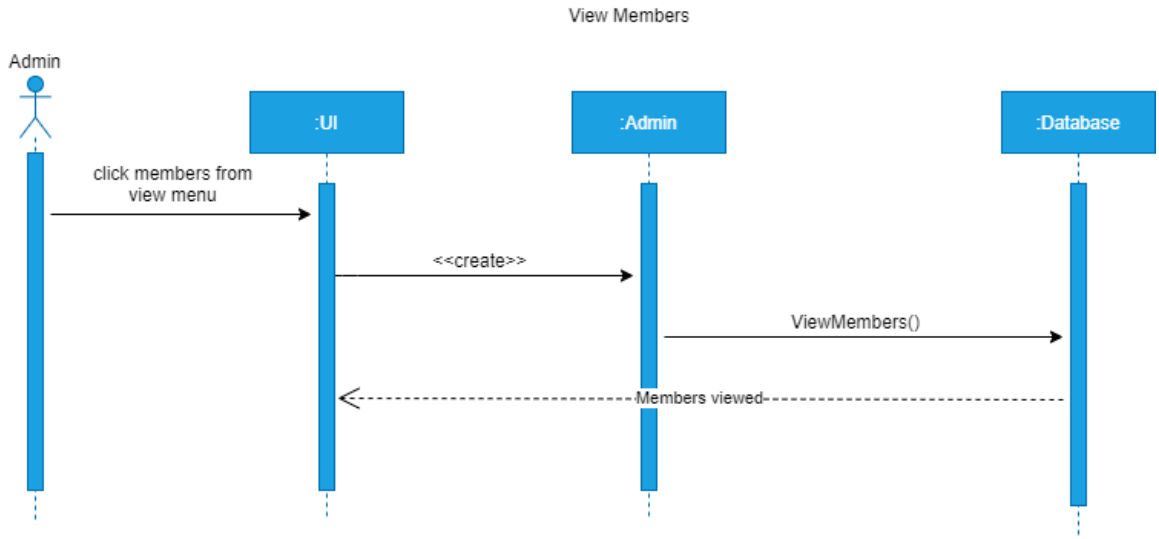
16-Trainer edits his profile



17-Admin views all packages

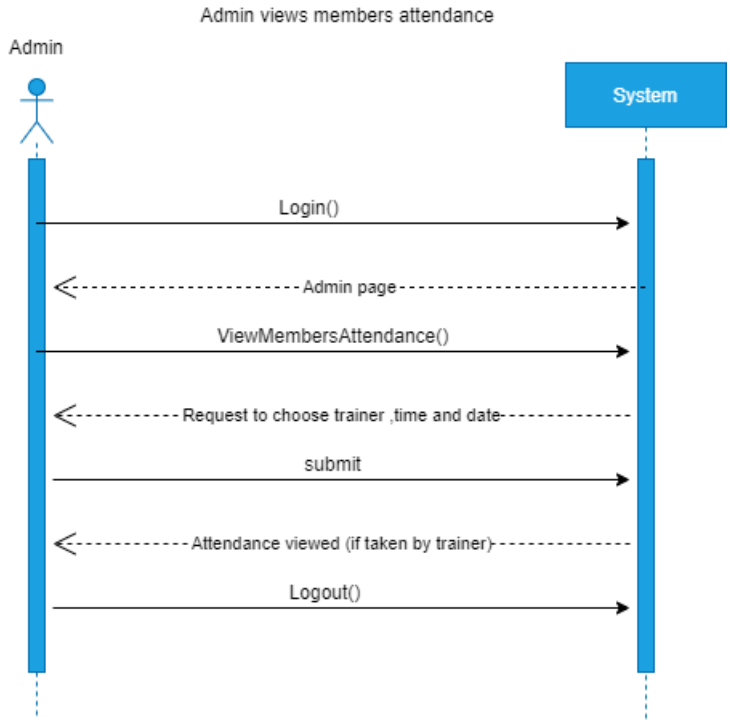


18-Admin views all members

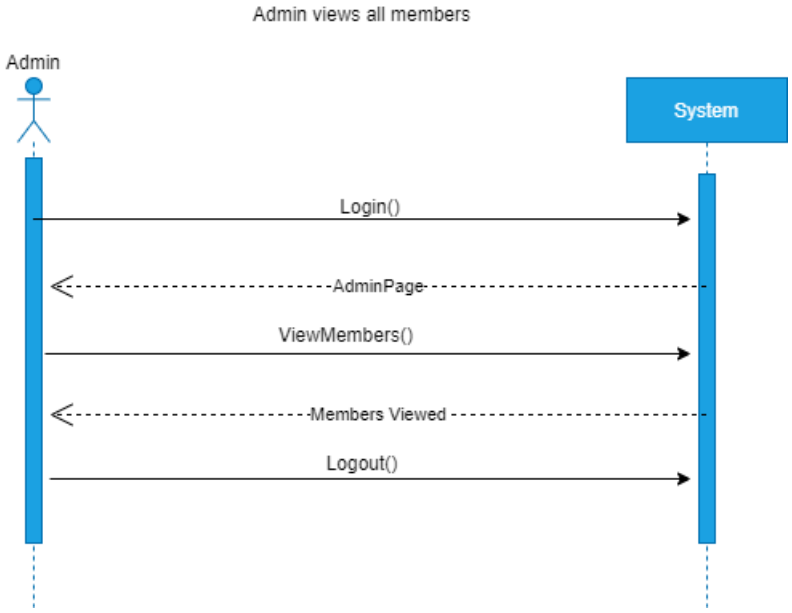


System Sequence Diagrams

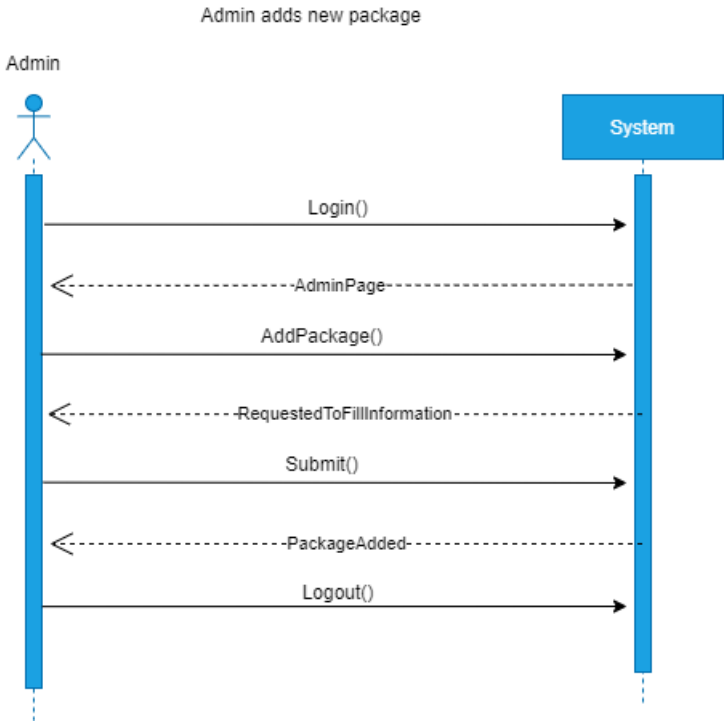
1-Admin views members attendance



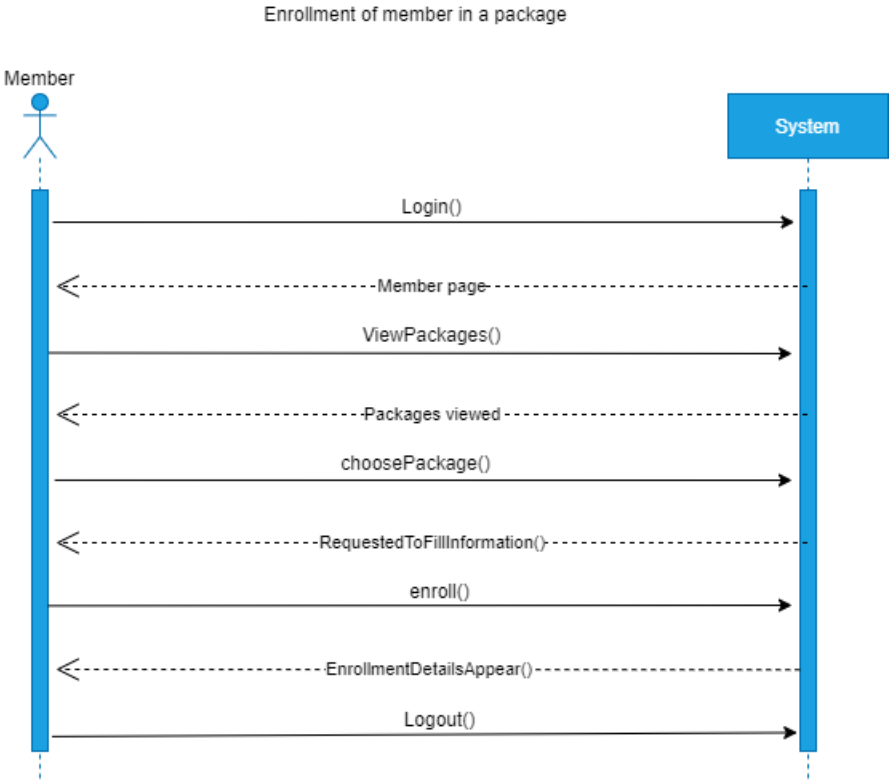
2-Admin views all members



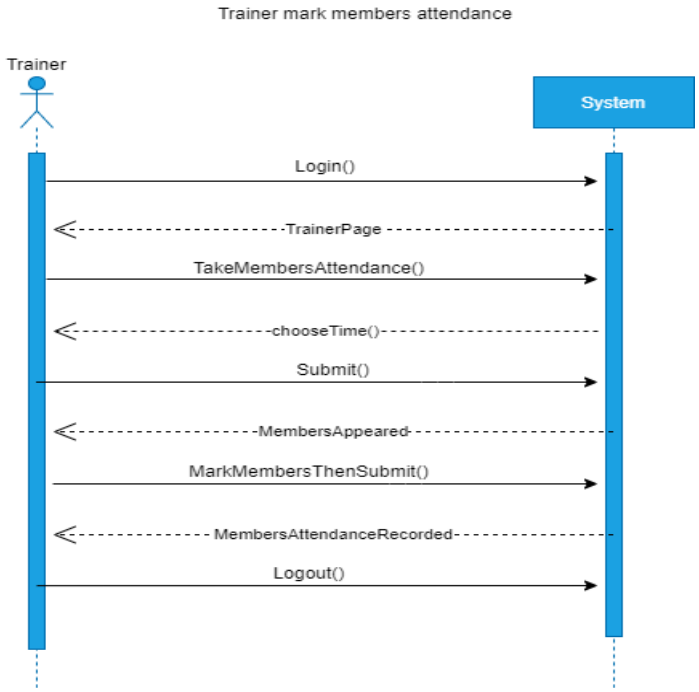
3-Admin adds a trainer



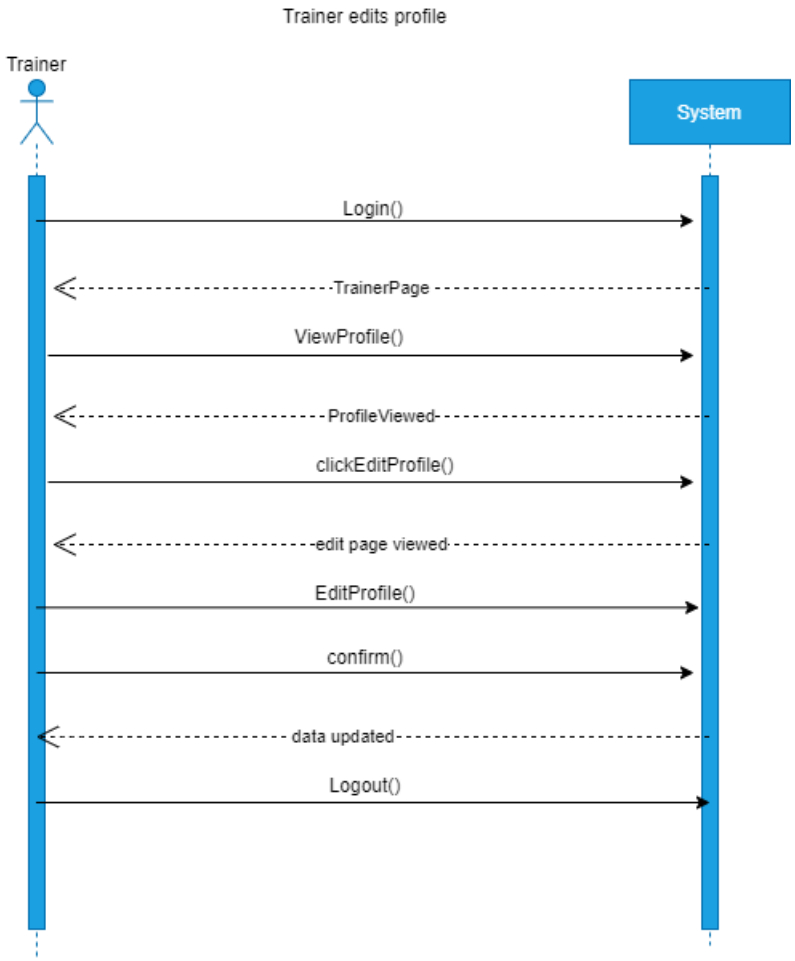
4-Member enrolls in a package



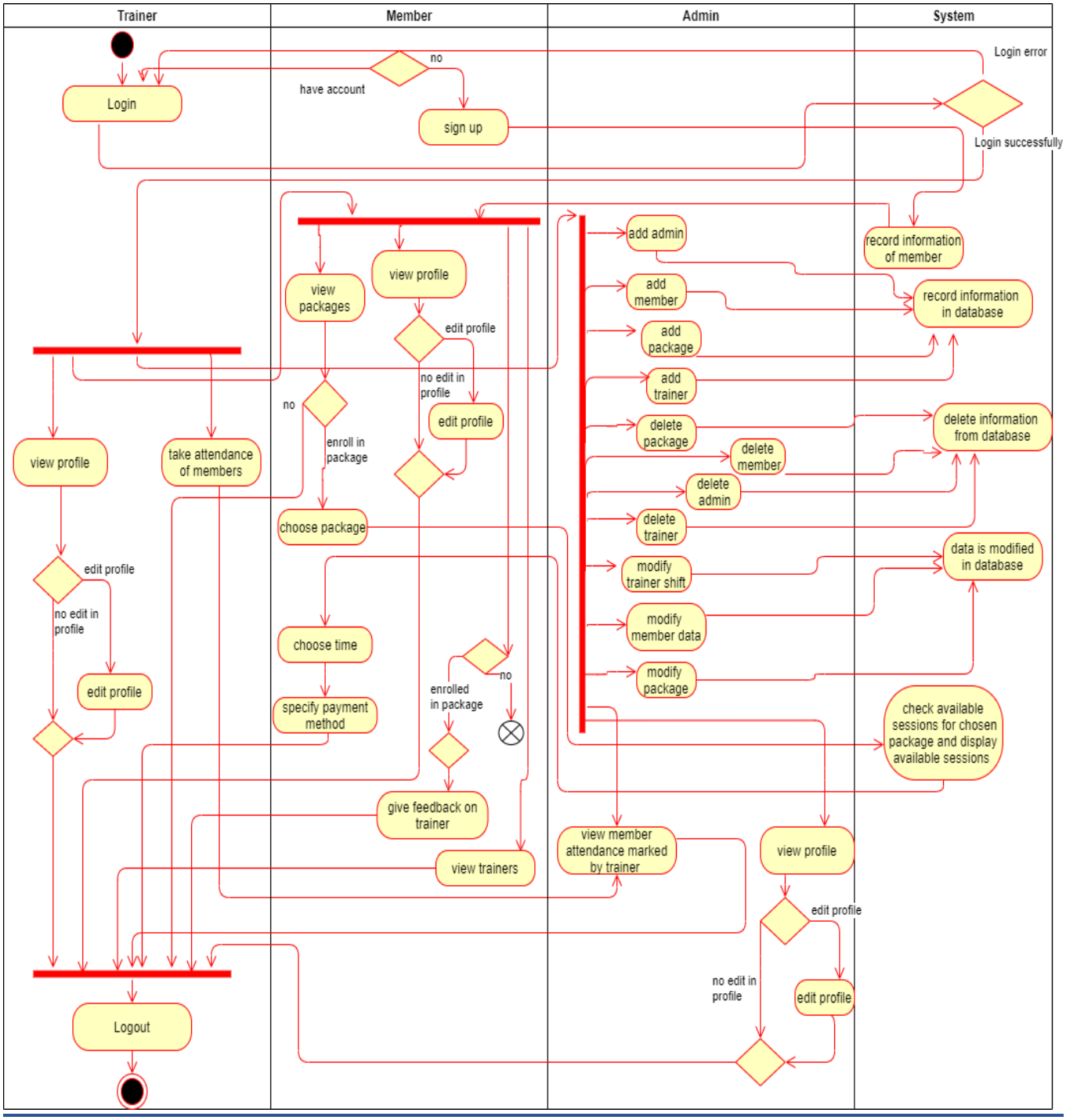
5-Trainer takes attendance of members



6-Trainer edits his/her profile



Activity Diagram



Design Patterns applied

1-Singlton Pattern

This pattern ensures that there is 1 instance from a class.

Context

-There exist some classes in which 1 instance only should be Created.

-Example: Database class

Problem

-How to guarantee that at any time there won't be more than 1 instance of singleton class and also provide a global point to access it through?

Forces

-Using a public constructor won't guarantee that there will be only 1 instance of the class.

-Singleton instance must be accessible to all classes that will use it so it must be public.

Solution

-The constructor must be private to guarantee that no other class will be able to create an instance of the singleton class.

-create a public static method that will check if there is no instance it will create one if there is it will store the reference

in a static private variable.

This pattern solved the problem of existing more than 1 instance at a time.

The design of the system became better as there is not many instances of the class on the same time but only 1.

There is only 1 instance opened for database.

2-Delegation Pattern

This pattern reuses the code as inheritance.

Context

-If there is a method in a class and you require only this method in another class, inheritance can not be the right concept to use as the is-a rule does not apply

Problem

-How can you reuse the method that already exists?

Forces

-Decrease the development cost by reusing the methods.

Solution

- The delegating method in the delegator class calls a method in the delegate class to perform a task. The relation between 2 classes must be association.

This Pattern solved problem of duplicating the code.

It affected the system design as it became better because There is no duplicate code and the methods is reused. Admin view trainers and packages and the member also does so. These 2 methods are implemented in admin and is called by admin Inside the member instead of duplicating the code.

3-Abstraction Occurrence Pattern

Context

-A set of related objects "occurrences", the members of such a set have some common information.

Problem

-what is the best way to represent such sets of occurrences?

Forces

-Representing the members of each set without duplicating the common information.

Solution

-Create abstraction class that will contain common data

-Create an occurrence class that represents the occurrences Of this abstraction

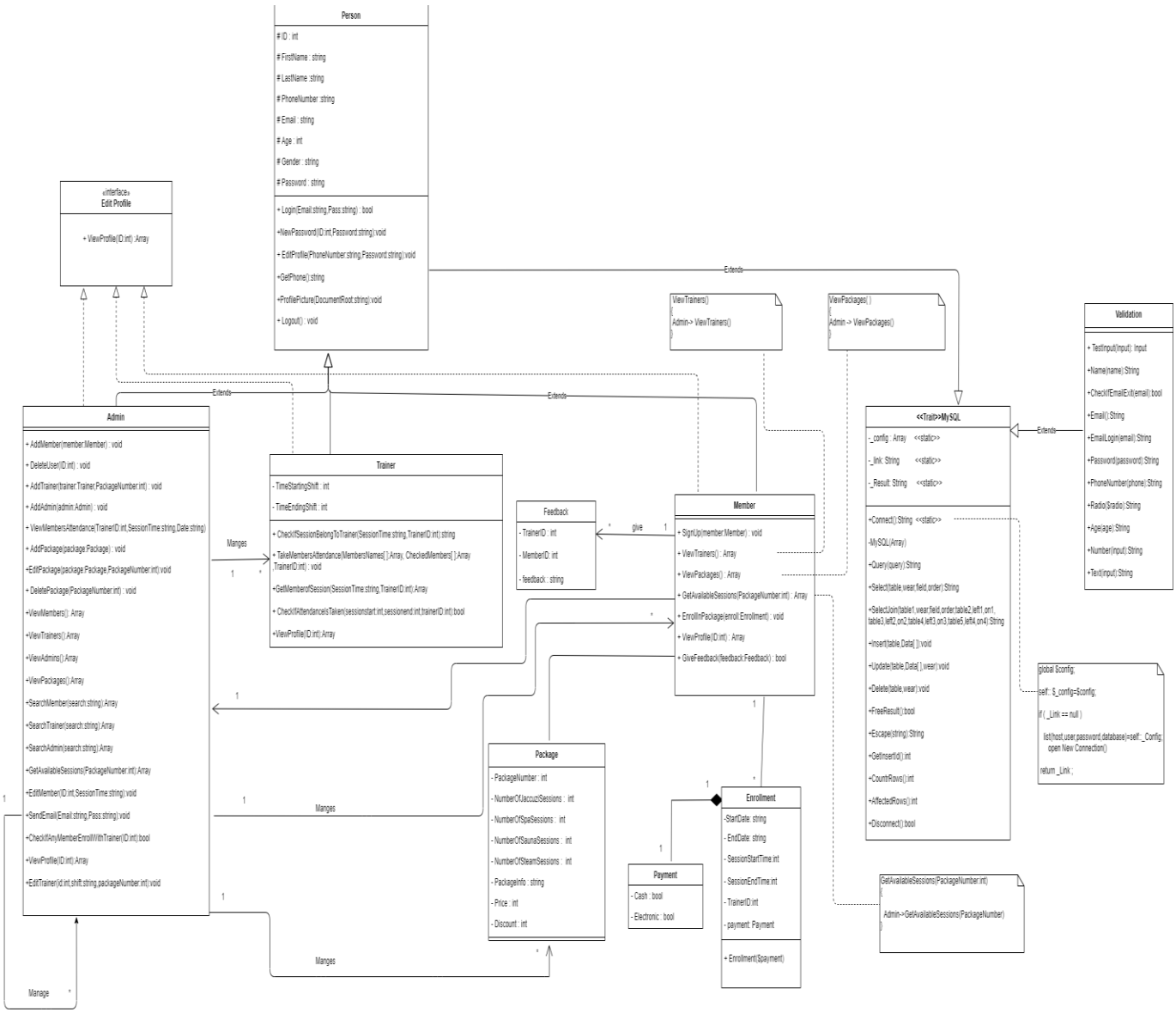
-Make the relation between the classes 1 to many

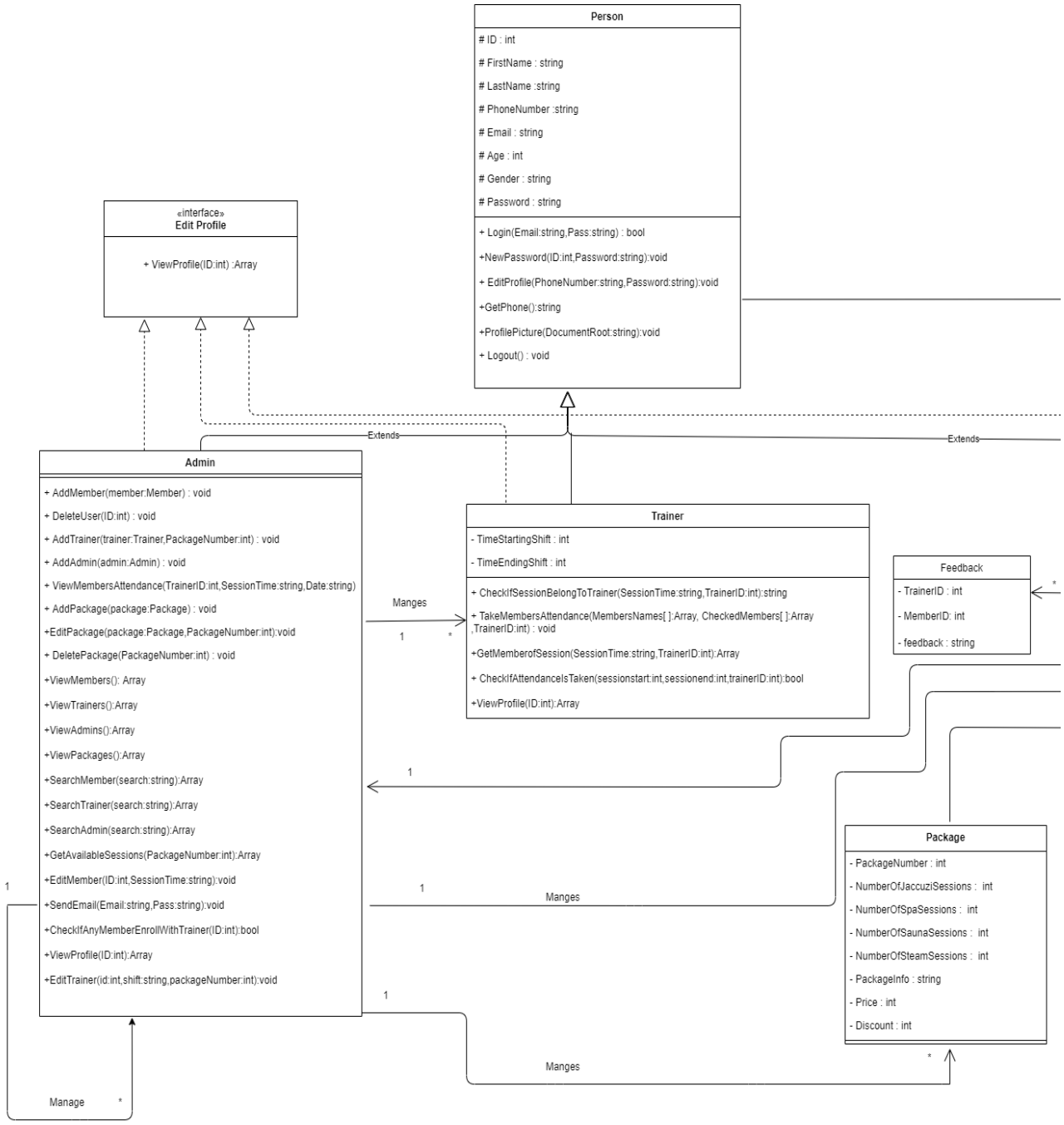
This pattern solved the problem of creating many objects and some of the objects contains many common information.

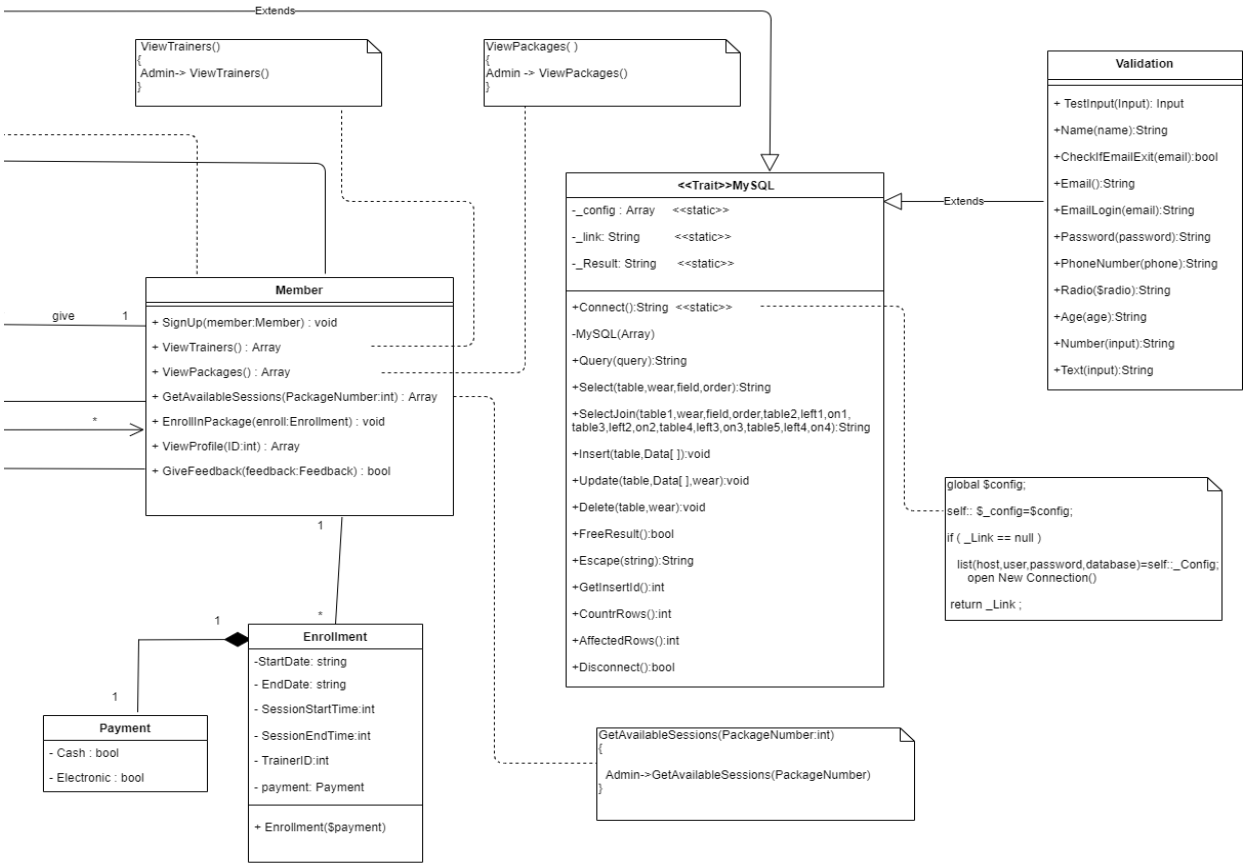
In this system the member can enroll more than once but not on the same time "multiple scheduling for the same event".

So the solution was to create a class for member and a class for the enrollment and the relation between 2 classes is 1 to many.

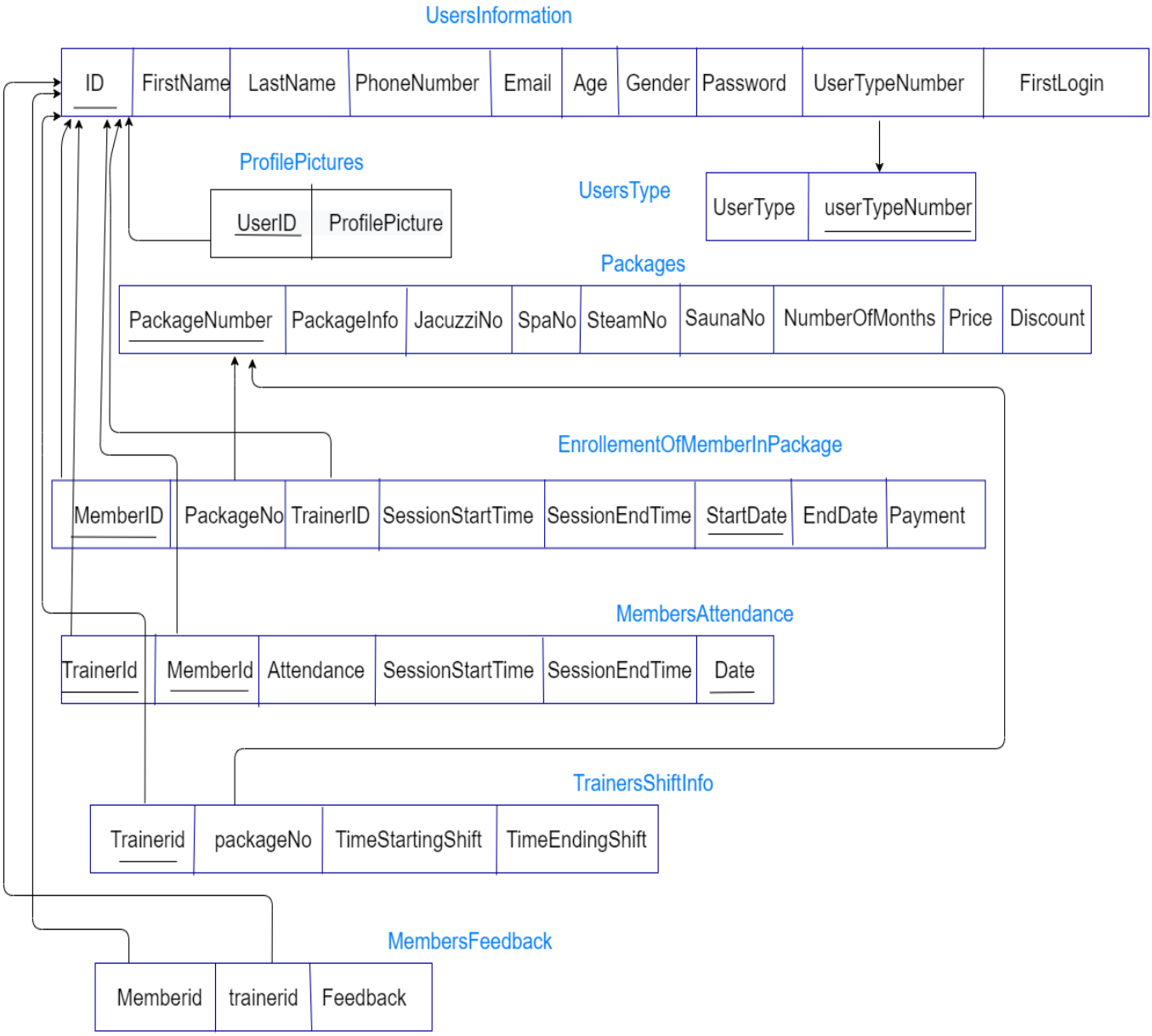
Class Diagram



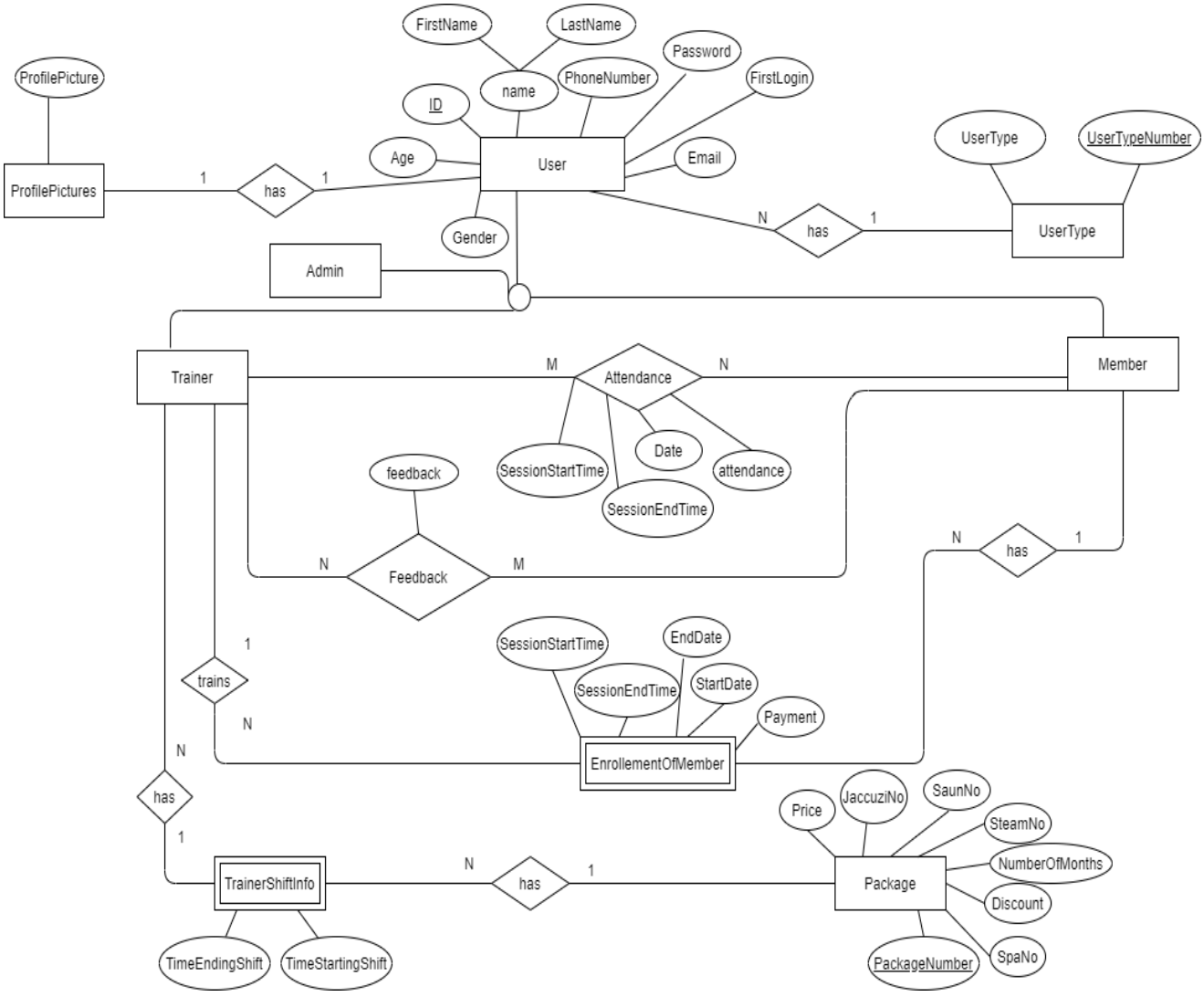

Person
ID : int
FirstName : string
LastName :string
PhoneNumber :string
Email : string
Age : int
Gender : string
Password : string
+ Login(Email:string,Pass:string) : bool
+NewPassword(ID:int,Password:string):void
+ EditProfile(PhoneNumber:string,Password:string):void
+GetPhone():string
+ProfilePicture(DocumentRoot:string):void
+ Logout() : void
«interface»
Edit Profile
+ ViewProfile(ID:int) :Array
Extends
Extends
Admin
+ AddMember(member:Member) : void
+ DeleteUser(ID:int) : void
+ AddTrainer(trainer:Trainer,PackageNumber:int) : void
+ AddAdmin(admin:Admin) : void
+ ViewMembersAttendance(TrainerID:int,SessionTime:string,Date:string)
+ AddPackage(package:Package) : void
+EditPackage(package:Package,PackageNumber:int):void
+ DeletePackage(PackageNumber:int) : void
+ViewMembers(): Array
+ViewTrainers():Array
+ViewAdmins():Array
+ViewPackages():Array
+SearchMember(search:string):Array
+SearchTrainer(search:string):Array
+SearchAdmin(search:string):Array
+GetAvailableSessions(PackageNumber:int):Array
+EditMember(ID:int,SessionTime:string):void
+SendEmail(Email:string,Pass:string):void
+CheckIfAnyMemberEnrollWithTrainer(ID:int):bool
+ViewProfile(ID:int):Array
+EditTrainer(id:int,shift:string,packageNumber:int):void
Manges
1
*
Trainer
- TimeStartingShift : int
- TimeEndingShift : int
+ CheckIfSessionBelongToTrainer(SessionTime:string,TrainerID:int):string
+ TakeMembersAttendance(MembersNames[ ]:Array, CheckedMembers[ ]:Array ,TrainerID:int) : void
+GetMemberofSession(SessionTime:string,TrainerID:int):Array
+ CheckIfAttendanceIsTaken(sessionstart:int,sessionend:int,trainerID:int):bool
+ViewProfile(ID:int):Array
Feedback
- TrainerID : int
- MemberID: int
- feedback : string
*
1
1
Manges
Package
- PackageNumber : int
- NumberOfJaccuziSessions : int
- NumberOfSpaSessions : int
- NumberOfSaunaSessions : int
- NumberOfSteamSessions : int
- PackageInfo : string
- Price : int
- Discount : int
*
1
Manges
1
Manage
*


Extends
ViewTrainers()
{
Admin-> ViewTrainers()
}
ViewPackages( )
{
Admin -> ViewPackages()
}
Validation
+ TestInput(Input): Input
+Name(name):String
+CheckIfEmailExit(email):bool
+Email():String
+EmailLogin(email):String
+Password(password):String
+PhoneNumber(phone):String
+Radio($radio):String
+Age(age):String
+Number(input):String
+Text(input):String
<<Trait>>MySQL
-_config : Array <<static>>
-_link: String <<static>>
-_Result: String <<static>>
+Connect():String <<static>>
-MySQL(Array)
+Query(query):String
+Select(table,wear,field,order):String
+SelectJoin(table1,wear,field,order,table2,left1,on1, table3,left2,on2,table4,left3,on3,table5,left4,on4):String
+Insert(table,Data[ ]):void
+Update(table,Data[ ],wear):void
+Delete(table,wear):void
+FreeResult():bool
+Escape(string):String
+GetInsertId():int
+CountrRows():int
+AffectedRows():int
+Disconnect():bool
Extends
Member
give
1
+ SignUp(member:Member) : void
+ ViewTrainers() : Array
+ ViewPackages() : Array
+ GetAvailableSessions(PackageNumber:int) : Array
+ EnrollInPackage(enroll:Enrollment) : void
+ ViewProfile(ID:int) : Array
+ GiveFeedback(feedback:Feedback) : bool
*
1
global $config;
self:: $_config=$config;
if (_Link == null)
list(host,user,password,database)=self::_Config;
open New Connection()
return _Link ;
1
*
Payment
- Cash : bool
- Electronic : bool
1
Enrollment
-StartDate: string
- EndDate: string
- SessionStartTime:int
- SessionEndTime:int
- TrainerID:int
- payment: Payment
+ Enrollment($payment)
GetAvailableSessions(PackageNumber:int)
{
Admin->GetAvailableSessions(PackageNumber)
}

Database tables



ERD


FirstName
LastName
Password
ProfilePicture
name
PhoneNumber
FirstLogin
ID
Age
User
Email
UserType
UserTypeNumber
ProfilePictures
1
has
1
Gender
N
has
1
UserType
Admin
Trainer
M
Attendance
N
Member
Date
feedback
SessionStartTime
attendance
SessionEndTime
N
Feedback
M
N
has
1
1
trains
SessionStartTime
EndDate
SessionEndTime
StartDate
Payment
N
EnrollementOfMember
N
has
Price
JaccuziNo
SaunNo
SteamNo
1
TrainerShiftInfo
N
has
1
Package
NumberOfMonths
Discount
TimeEndingShift
TimeStartingShift
PackageNumber
SpaNo