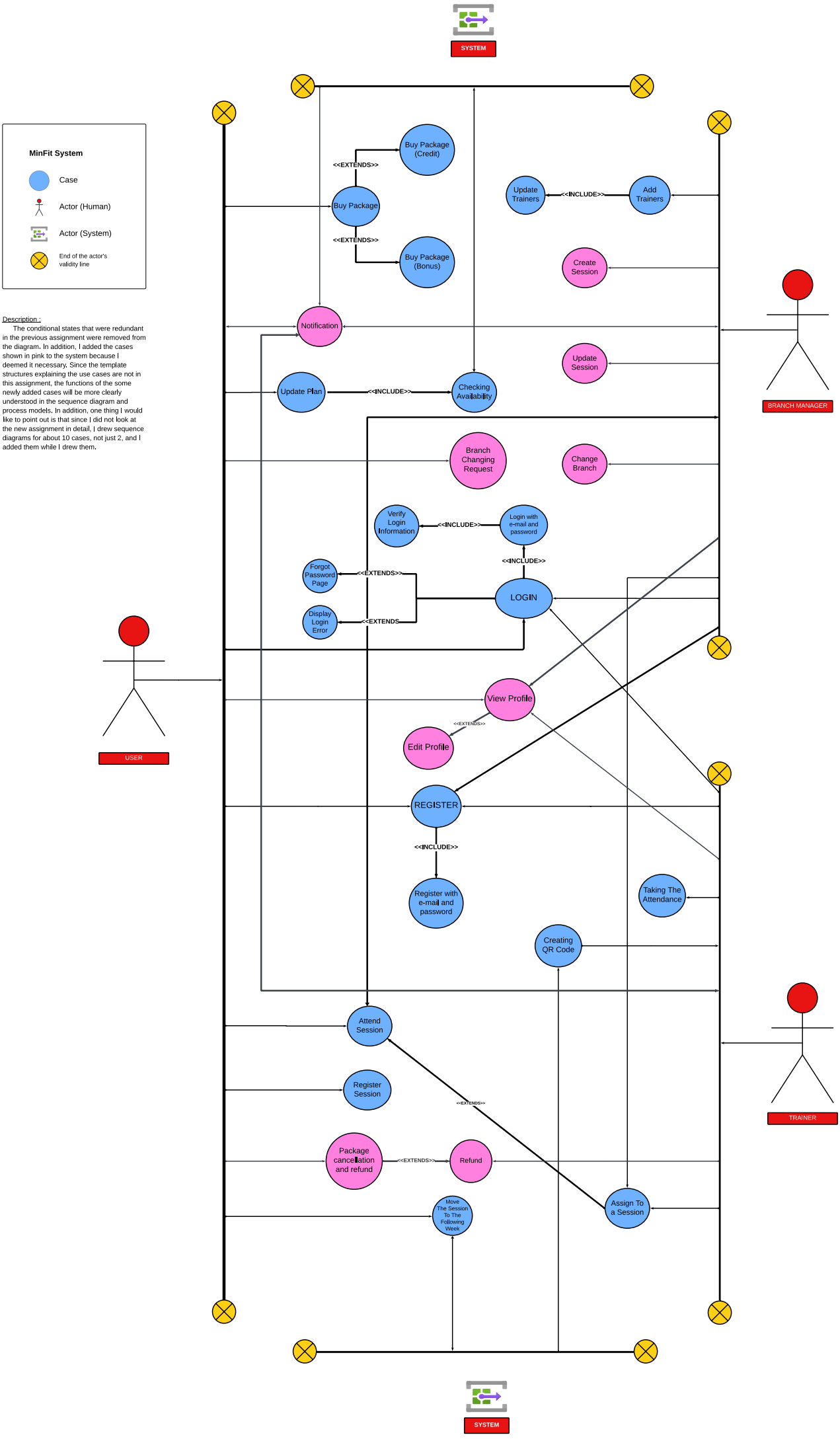
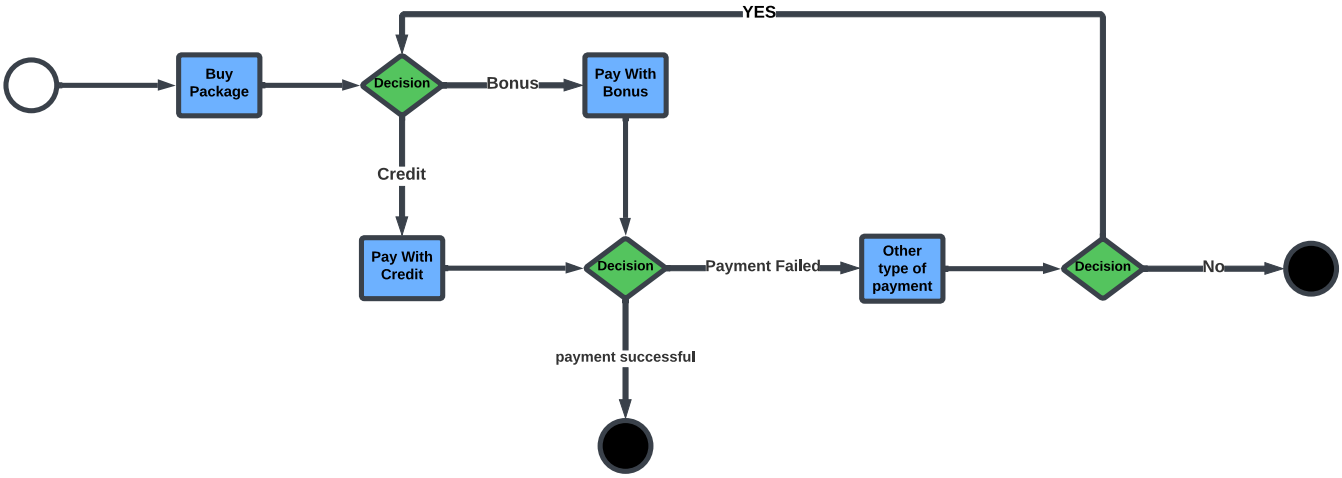
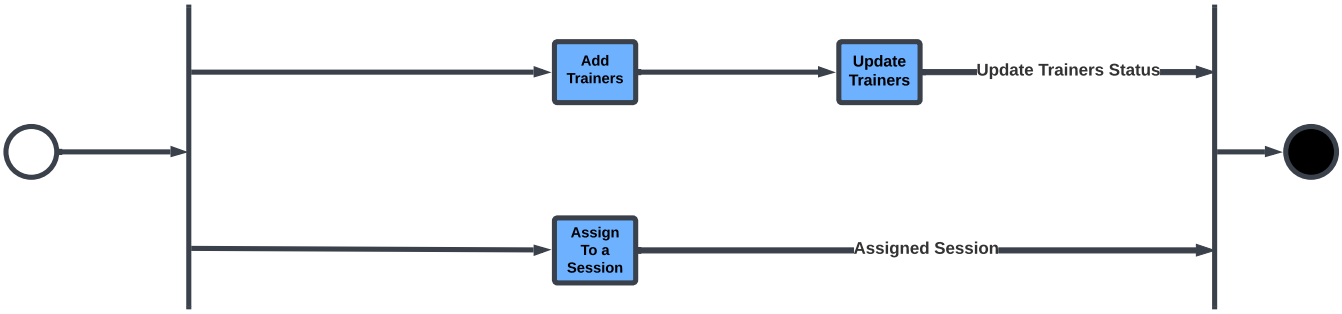

SYSTEM
MinFit System
Case
Actor (Human)
Actor (System)
End of the actor's validity line
Description :
The conditional states that were redundant in the previous assignment were removed from the diagram. In addition, I added the cases shown in pink to the system because I deemed it necessary. Since the template structures explaining the use cases are not in this assignment, the functions of the some newly added cases will be more clearly understood in the sequence diagram and process models. In addition, one thing I would like to point out is that since I did not look at the new assignment in detail, I drew sequence diagrams for about 10 cases, not just 2, and I added them while I drew them.
Buy Package (Credit)
<<EXTENDS>>
Buy Package
<<EXTENDS>>
Buy Package (Bonus)
Update Trainers
<<INCLUDE>>
Add Trainers
Create Session
Notification
Update Session
Update Plan
<<INCLUDE>>
Checking Availability
BRANCH MANAGER
Branch Changing Request
Change Branch
Verify Login Information
<<INCLUDE>>
Login with e-mail and password
<<INCLUDE>>
Forgot Password Page
<<EXTENDS>>
LOGIN
Display Login Error
<<EXTENDS
USER
View Profile
<<EXTENDS>>
Edit Profile
REGISTER
<<INCLUDE>>
Register with e-mail and password
Taking The Attendance
Creating QR Code
Attend Session
Register Session
<<EXTENDS>>
Package cancellation and refund
<<EXTENDS>>
Refund
Move The Session To The Following Week
Assign To a Session
TRAINER
SYSTEM

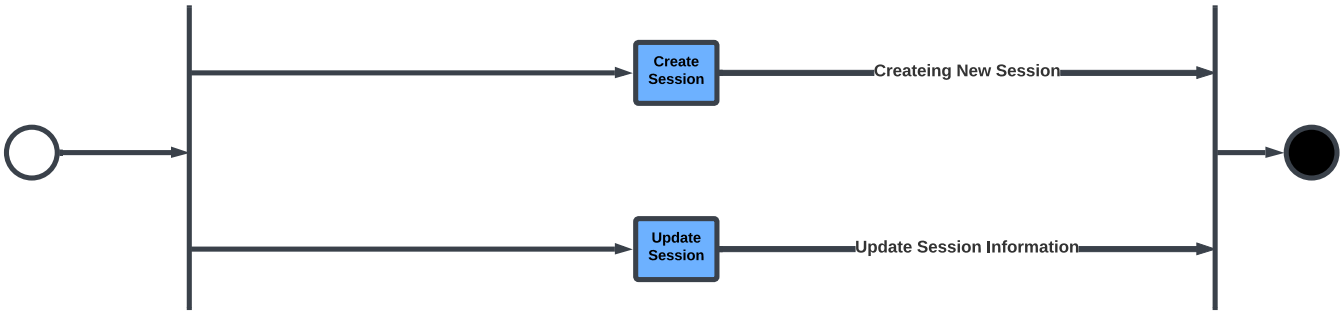
Process Model of Buy Package



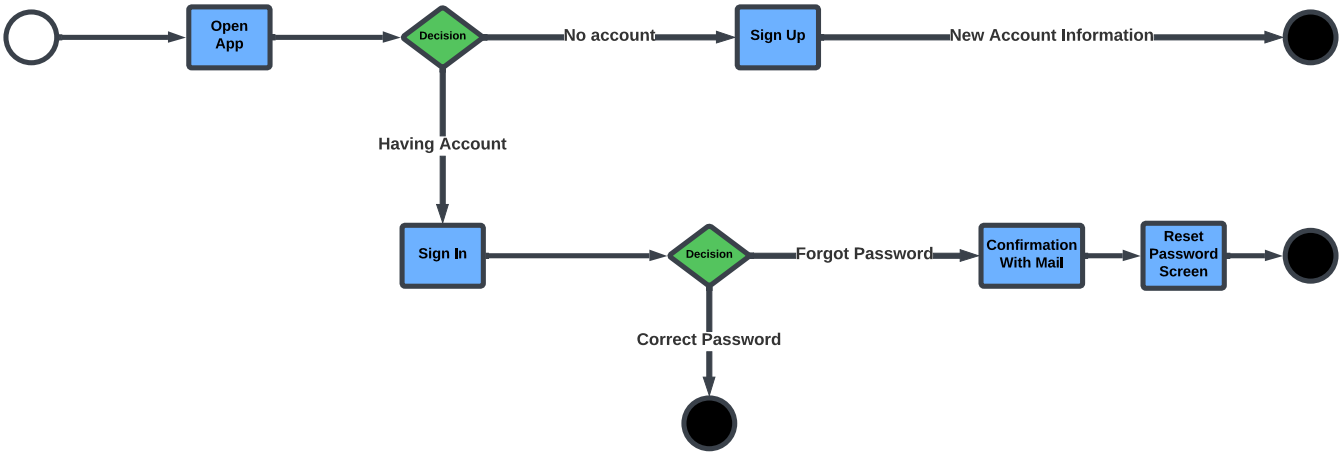
Branch Manager Manage Trainers



Branch Manager Manage Sessions

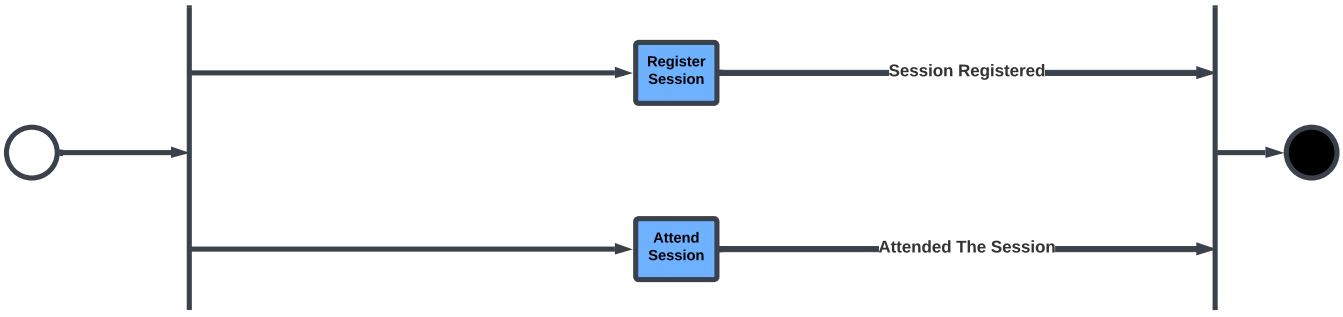


Login and Register

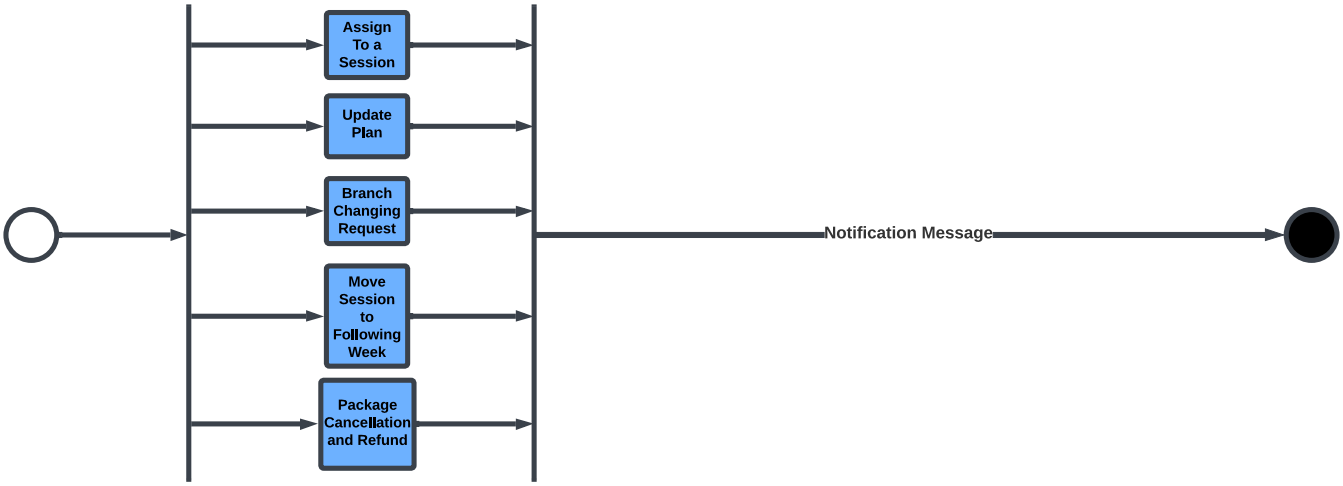


View / Edit Profile

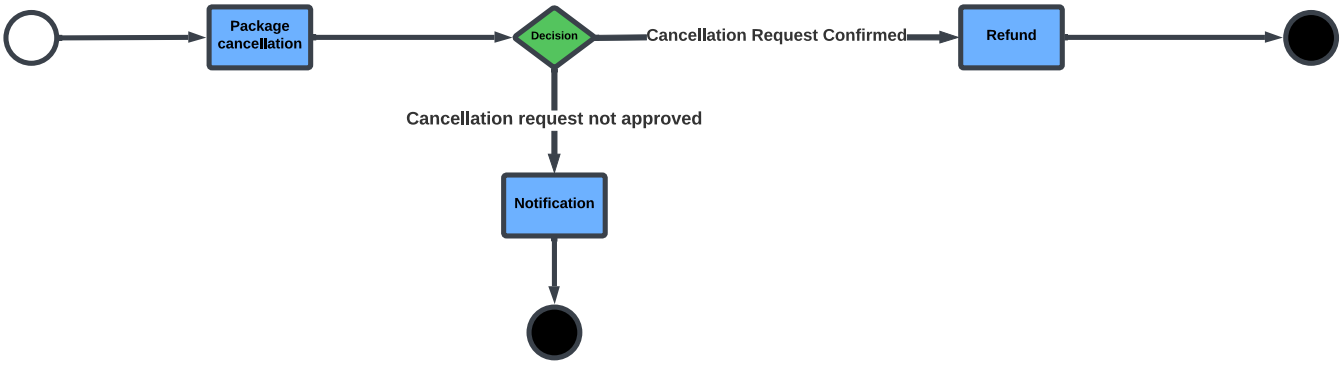
User Actvity on Sessions



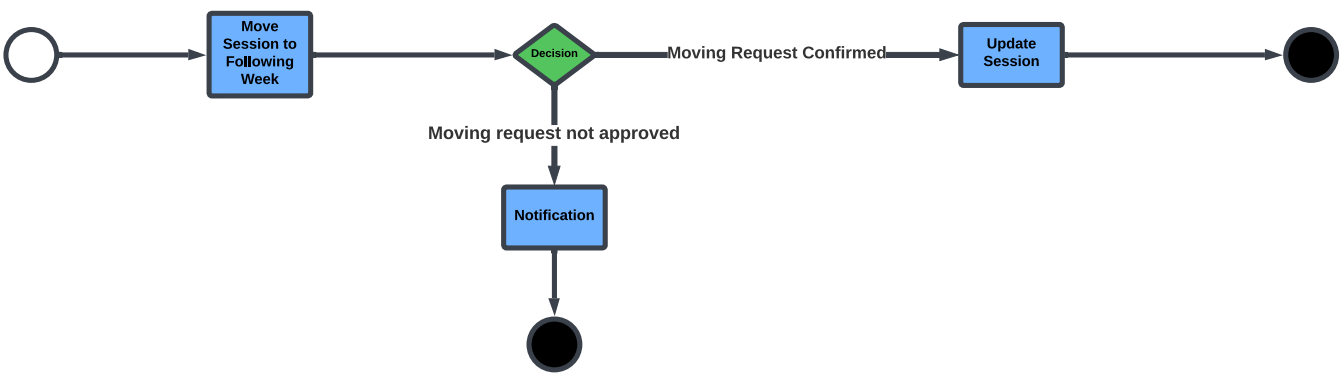
Notification



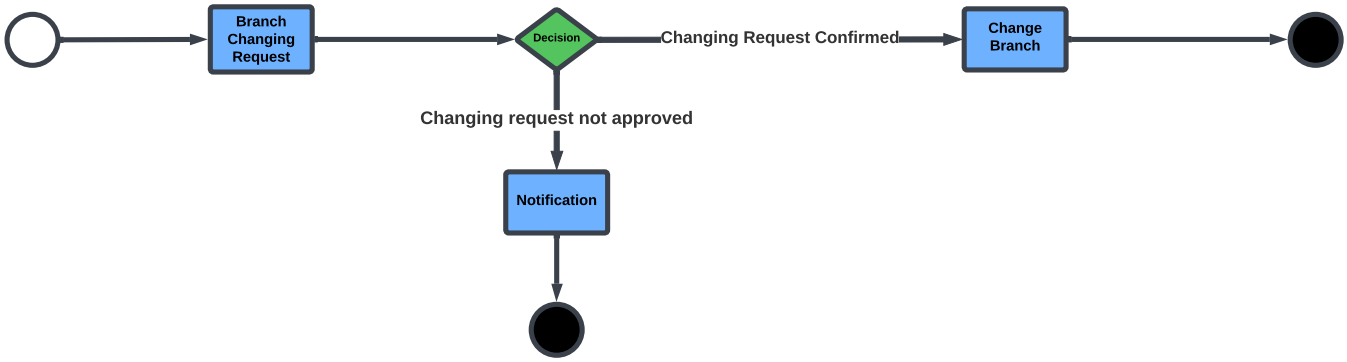
Package Cancellation and Refund



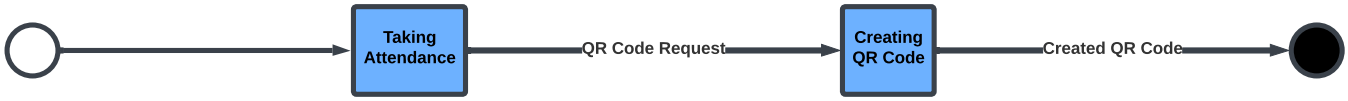
Move Session



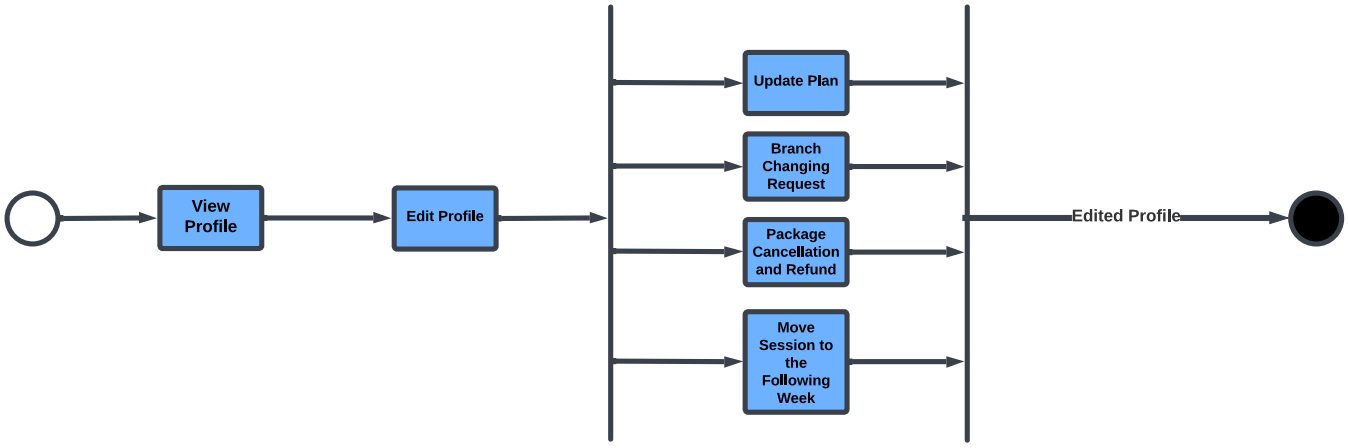
Branch Changing Request

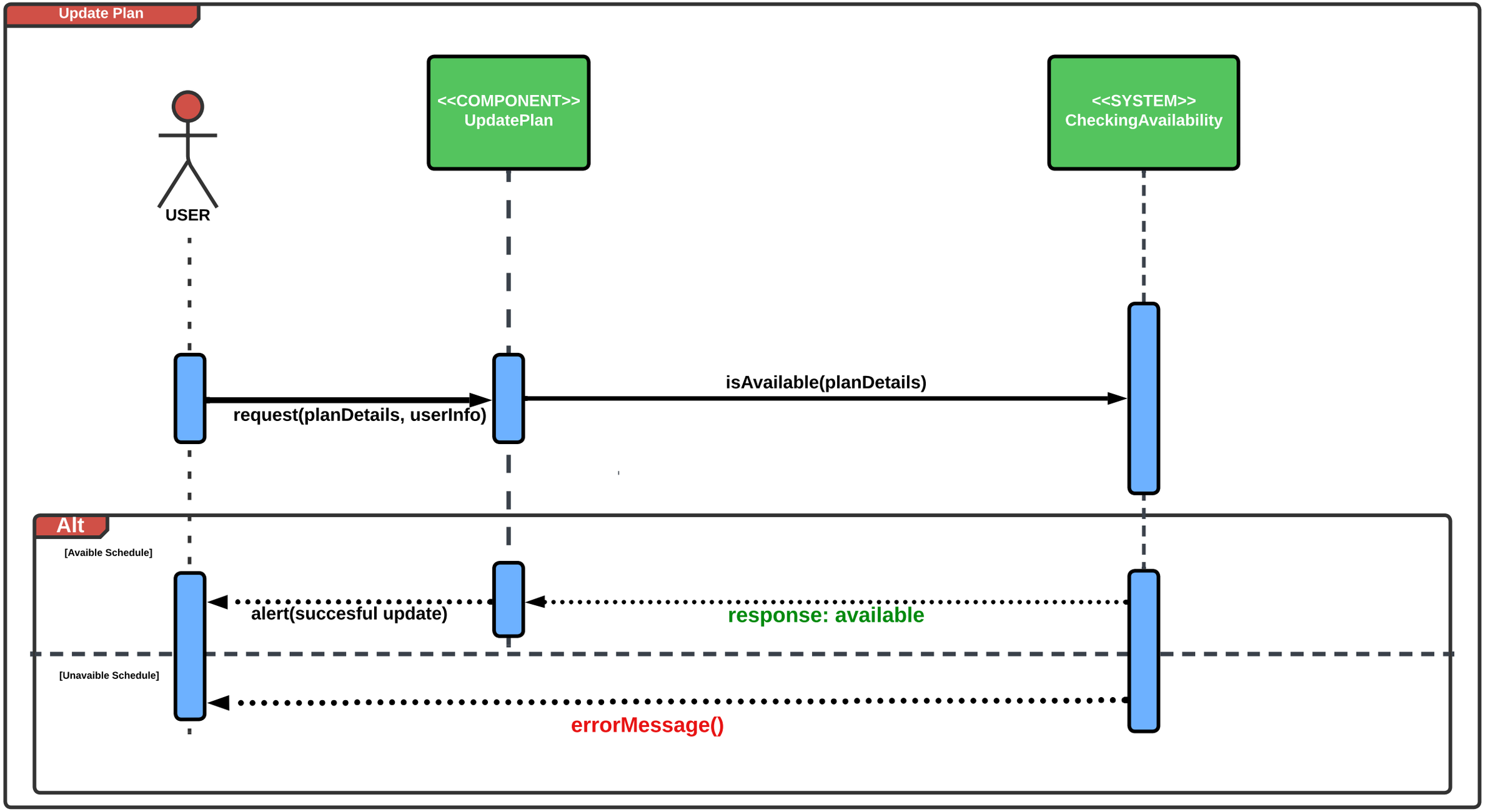


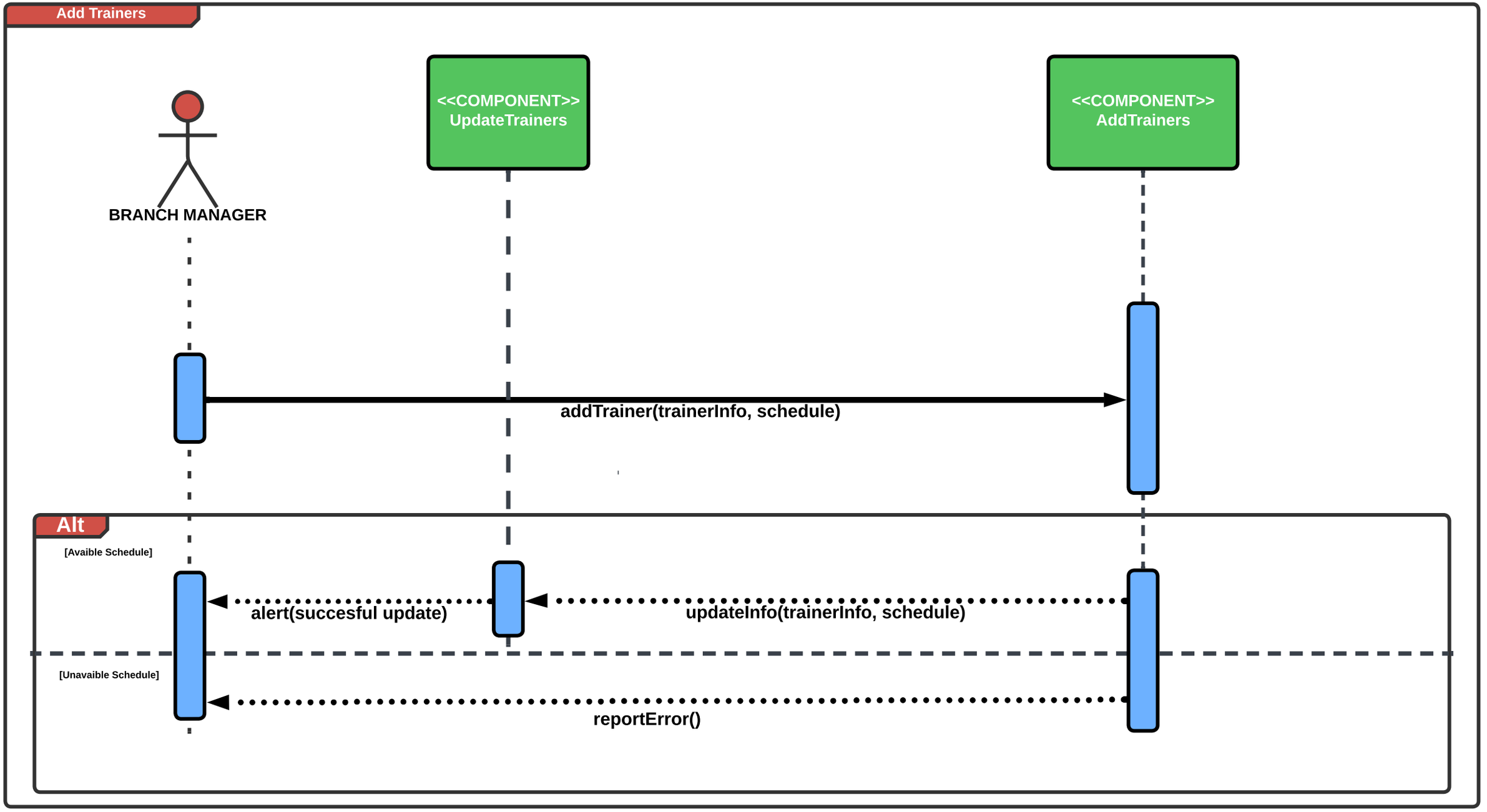
Taking Attendance / Creating QR Code

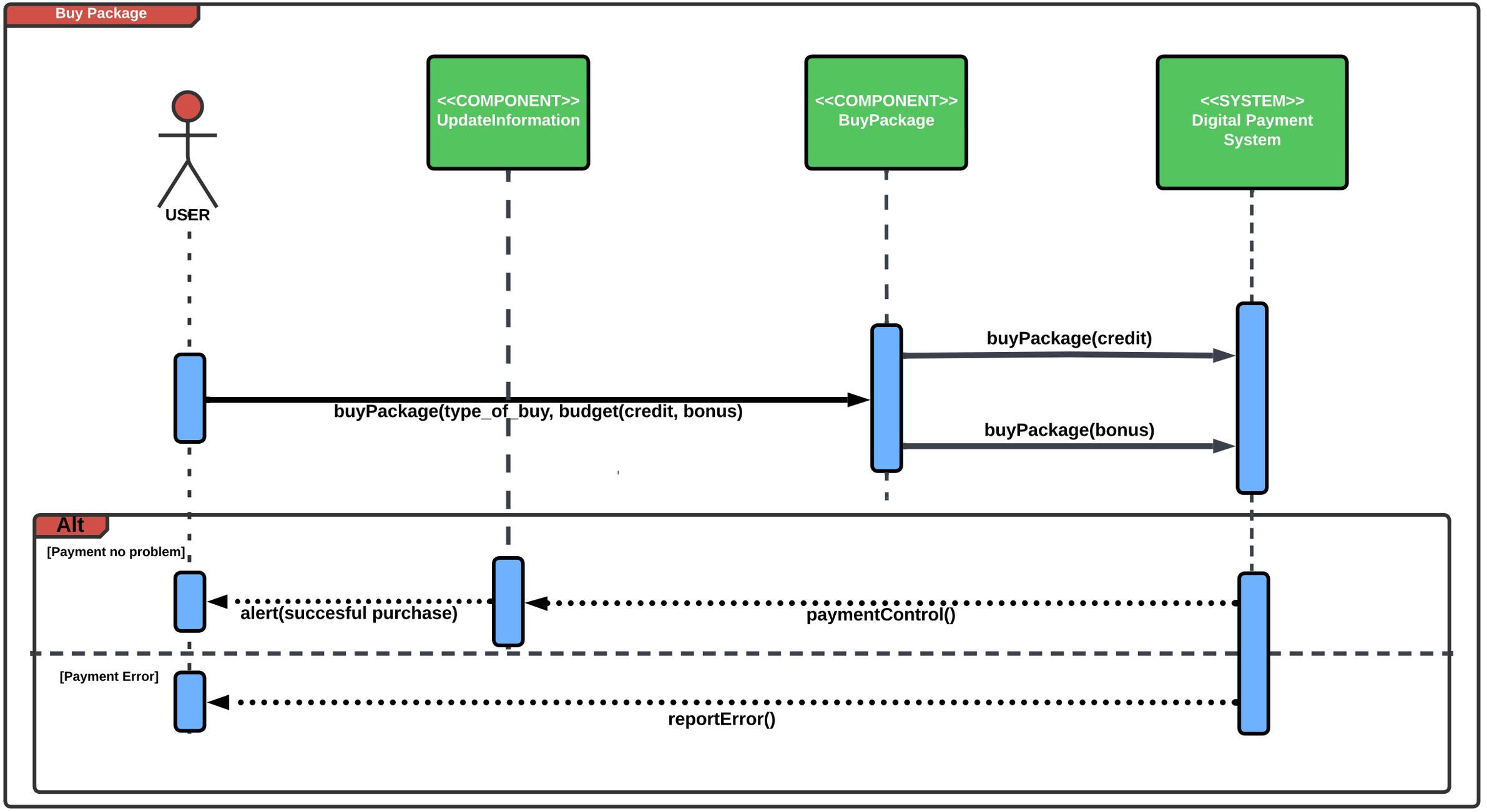


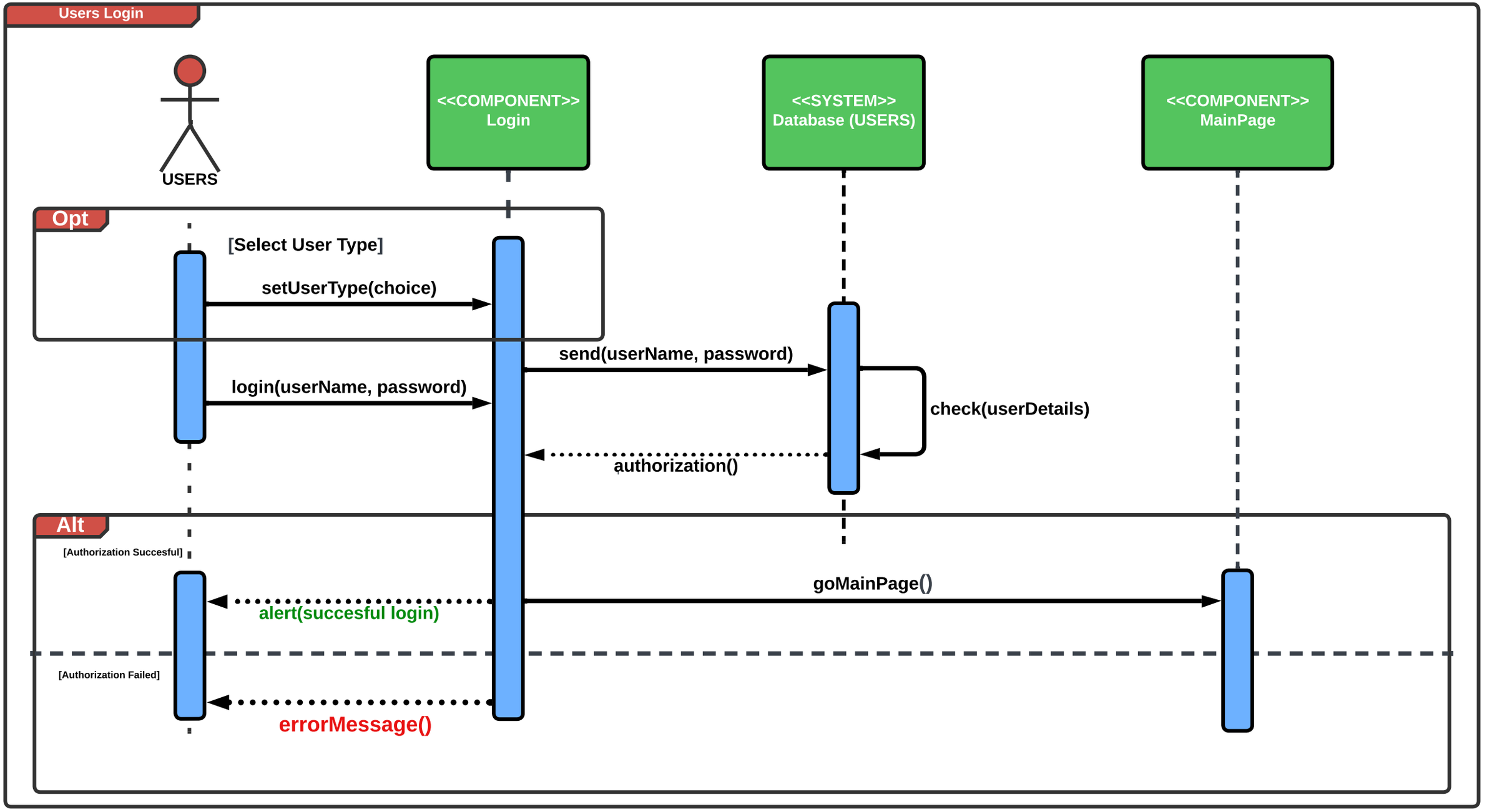
Process On Profile (User)

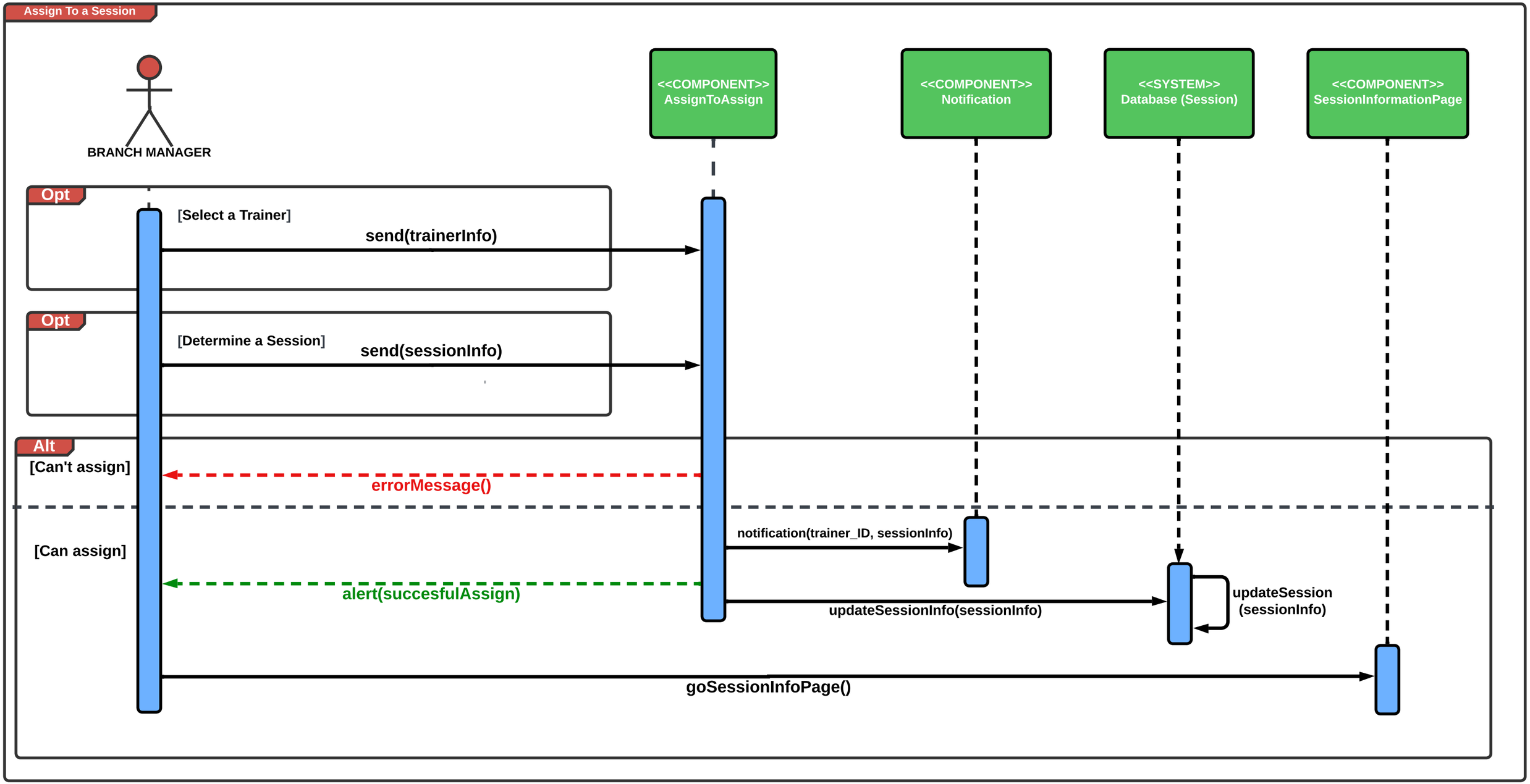


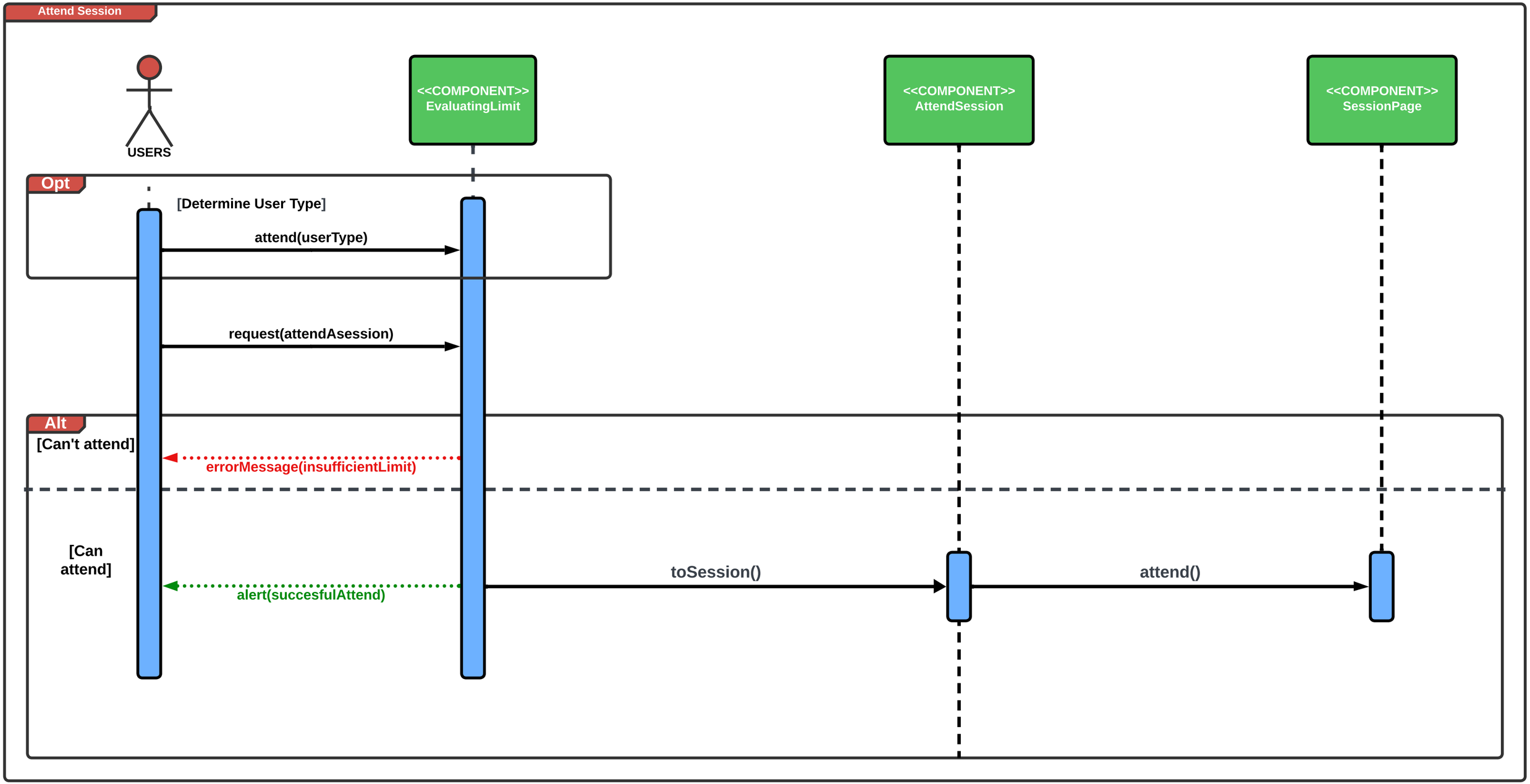

Update Plan
USER
<<COMPONENT>>
UpdatePlan
<<SYSTEM>>
CheckingAvailability
request(planDetails, userInfo)
isAvailable(planDetails)
Alt
[Avaible Schedule]
alert(succesful update)
response: available
[Unavaible Schedule]
errorMessage()

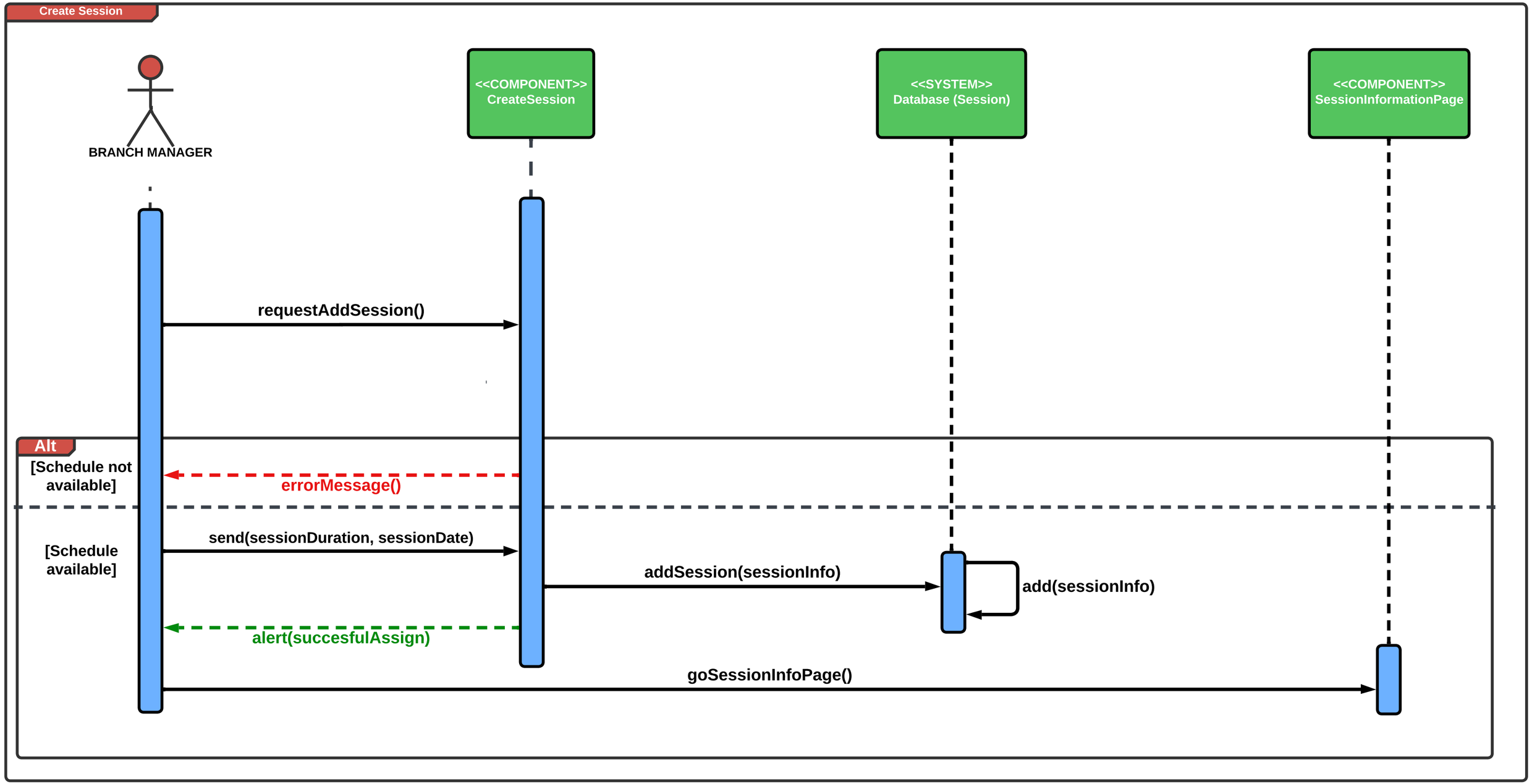

Add Trainers
BRANCH MANAGER
<<COMPONENT>>
UpdateTrainers
<<COMPONENT>>
AddTrainers
addTrainer(trainerInfo, schedule)
Alt
[Avaible Schedule]
alert(succesful update)
updateInfo(trainerInfo, schedule)
[Unavaible Schedule]
reportError()


Buy Package
USER
<<COMPONENT>>
UpdateInformation
<<COMPONENT>>
BuyPackage
<<SYSTEM>>
Digital Payment
System
buyPackage(credit)
buyPackage(type_of_buy, budget(credit, bonus)
buyPackage(bonus)
Alt
[Payment no problem]
alert(succesful purchase)
paymentControl()
[Payment Error]
reportError()


Users Login
USERS
<<COMPONENT>>
Login
<<SYSTEM>>
Database (USERS)
<<COMPONENT>>
MainPage
Opt
[Select User Type]
setUserType(choice)
send(userName, password)
login(userName, password)
check(userDetails)
authorization()
Alt
[Authorization Succesful]
goMainPage()
alert(succesful login)
[Authorization Failed]
errorMessage()


Assign To a Session
BRANCH MANAGER
<<COMPONENT>>
AssignToAssign
<<COMPONENT>>
Notification
<<SYSTEM>>
Database (Session)
<<COMPONENT>>
SessionInformationPage
Opt
[Select a Trainer]
send(trainerInfo)
Opt
[Determine a Session]
send(sessionInfo)
Alt
[Can't assign]
errorMessage()
[Can assign]
notification(trainer_ID, sessionInfo)
alert(succesfulAssign)
updateSessionInfo(sessionInfo)
updateSession
(sessionInfo)
goSessionInfoPage()


Attend Session
USERS
<<COMPONENT>>
EvaluatingLimit
<<COMPONENT>>
AttendSession
<<COMPONENT>>
SessionPage
Opt
[Determine User Type]
attend(userType)
request(attendAsession)
Alt
[Can't attend]
errorMessage(insufficientLimit)
[Can
attend]
alert(succesfulAttend)
toSession()
attend()


Create Session
BRANCH MANAGER
<<COMPONENT>>
CreateSession
<<SYSTEM>>
Database (Session)
<<COMPONENT>>
SessionInformationPage
requestAddSession()
Alt
[Schedule not available]
errorMessage()
[Schedule available]
send(sessionDuration, sessionDate)
addSession(sessionInfo)
add(sessionInfo)
alert(succesfulAssign)
goSessionInfoPage()

Moving Session Next Week

USER

<<SYSTEM>>
Checking
Availability

<<COMPONENT>>
UpdateSession

<<SYSTEM>>
Database (Session)

<<COMPONENT>>
UserProfilePage

movingRequest(newSessionInfo)

Alt

[Schedule not
available]

errorMessage(unavailableSchedule)

[Schedule
available]

updateSession(newSessionInfo,userInfo)

update(newData,userInfo)

update(data)

alert(updateSession)

goProfilePage()

Users Register

USERS

<<COMPONENT>>
Register

<<SYSTEM>>
Mail

<<COMPONENT>>
ValidateScreen

<<SYSTEM>>
Database (Users)

<<COMPONENT>>
ProfilePage

Opt

[Select User Type]

setUserType(choice)

register(userName, password)

Alt

[Password not matched]

errorMessage(invalidFormat)

[Password matched]

send(userInfo)

validate(mail)

sendCode()

alert(enterCode)

input(code)

Alt

[Email not confirmed]

errorMessage(invalidCode)

[Email confirmed]

send(userInfo)

addUser(userInfo)

alert(succesfulRegister)

goProfile()

Taking The Attendance

TRAINER

<<COMPONENT>>
TakingAttendance

<<SYSTEM>>
QR Code Generator

<<COMPONENT>>
QRCodePage

onClick(takeAttendance)

createQRCode(SessionInfo)

Alt

[Schedule not available]

errorMessage(invalidSession)

[Schedule available]

alert(QRCodeCreated)

goQRCodeScreen()

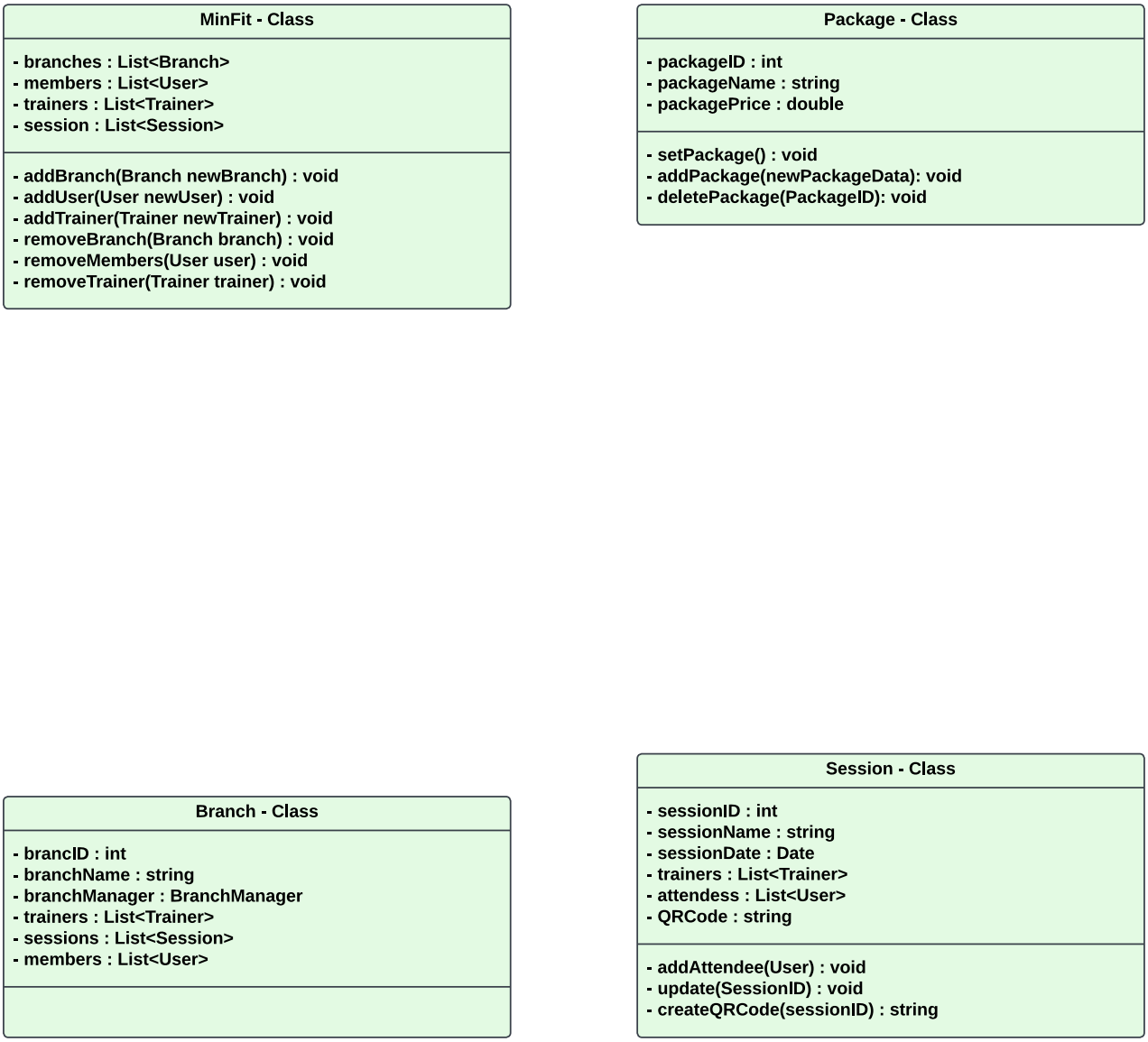
Alt

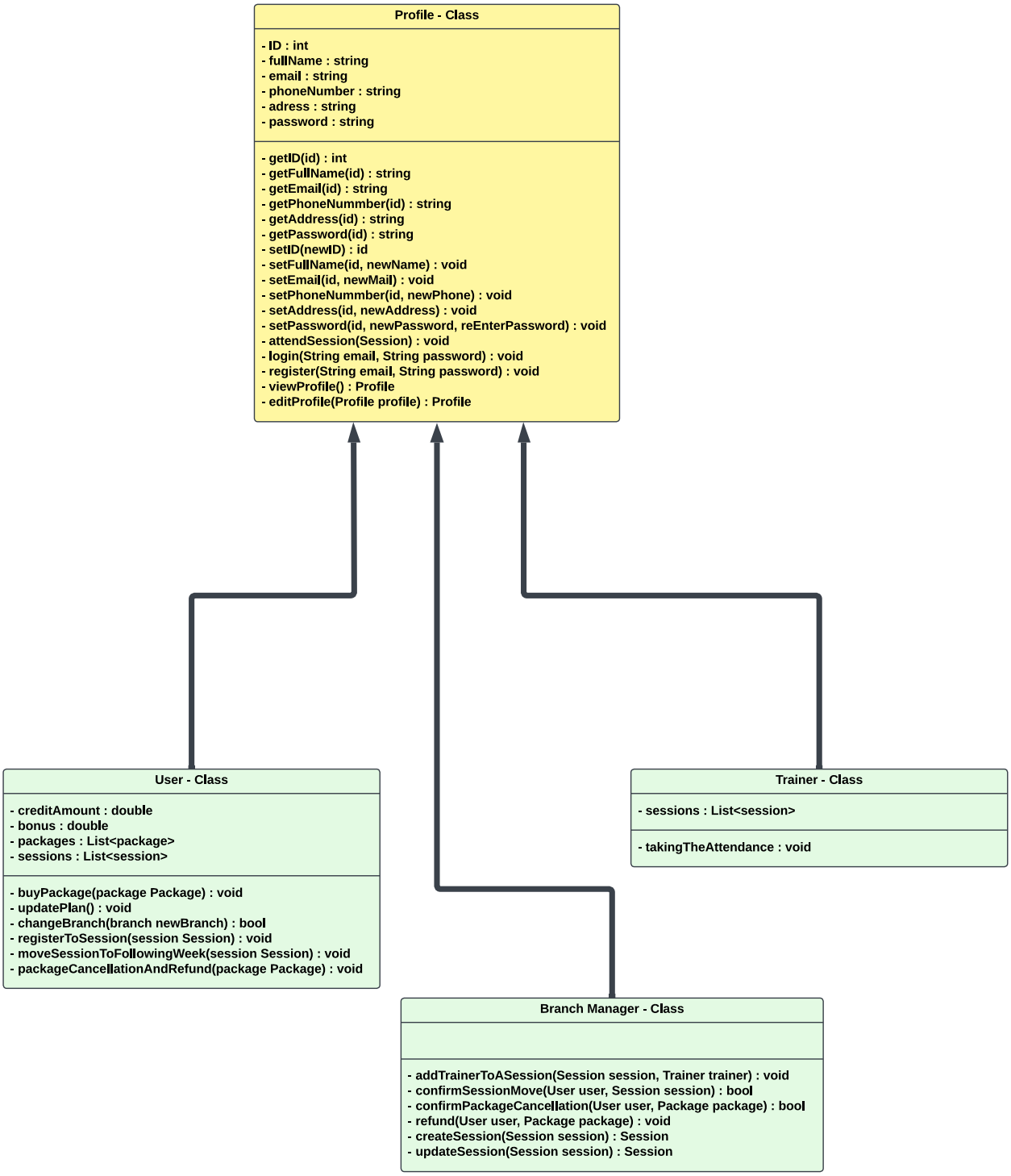
[Scan is Succesful]

alert(SuccesfulScan)

[Scan Error]

errorMessage(scanningError)


MinFit - Class
- branches : List<Branch>
- members : List<User>
- trainers : List<Trainer>
- session : List<Session>
- addBranch(Branch newBranch) : void
- addUser(User newUser) : void
- addTrainer(Trainer newTrainer) : void
- removeBranch(Branch branch) : void
- removeMembers(User user) : void
- removeTrainer(Trainer trainer) : void
Package - Class
- packageID : int
- packageName : string
- packagePrice : double
- setPackage() : void
- addPackage(newPackageData): void
- deletePackage(PackageID): void
Branch - Class
- brancID : int
- branchName : string
- branchManager : BranchManager
- trainers : List<Trainer>
- sessions : List<Session>
- members : List<User>
Session - Class
- sessionID : int
- sessionName : string
- sessionDate : Date
- trainers : List<Trainer>
- attendess : List<User>
- QRCode : string
- addAttendee(User) : void
- update(SessionID) : void
- createQRCode(sessionID) : string


Profile - Class
- ID : int
- fullName : string
- email : string
- phoneNumber : string
- adress : string
- password : string
- getID(id) : int
- getFullName(id) : string
- getEmail(id) : string
- getPhoneNummber(id) : string
- getAddress(id) : string
- getPassword(id) : string
- setID(newID) : id
- setFullName(id, newName) : void
- setEmail(id, newMail) : void
- setPhoneNummber(id, newPhone) : void
- setAddress(id, newAddress) : void
- setPassword(id, newPassword, reEnterPassword) : void
- attendSession(Session) : void
- login(String email, String password) : void
- register(String email, String password) : void
- viewProfile() : Profile
- editProfile(Profile profile) : Profile
User - Class
- creditAmount : double
- bonus : double
- packages : List<package>
- sessions : List<session>
- buyPackage(package Package) : void
- updatePlan() : void
- changeBranch(branch newBranch) : bool
- registerToSession(session Session) : void
- moveSessionToFollowingWeek(session Session) : void
- packageCancellationAndRefund(package Package) : void
Trainer - Class
- sessions : List<session>
- takingTheAttendance : void
Branch Manager - Class
- addTrainerToASession(Session session, Trainer trainer) : void
- confirmSessionMove(User user, Session session) : bool
- confirmPackageCancellation(User user, Package package) : bool
- refund(User user, Package package) : void
- createSession(Session session) : Session
- updateSession(Session session) : Session