05-01-2025

LUNKI SYSTEM DESIGN

TEAM DELTA

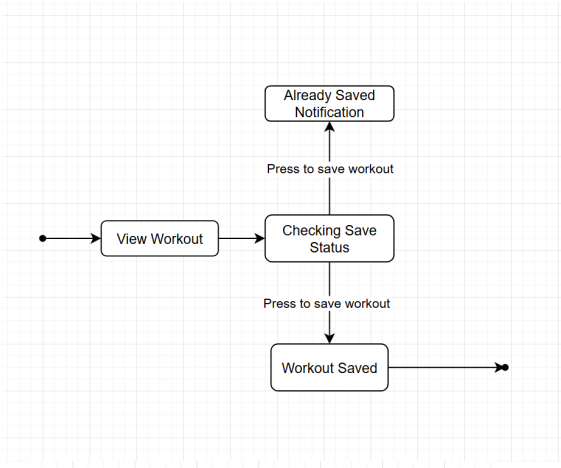
MATT BAUTISTA, BEN SEBIRI, ANGELO TURANO, CARLOS ESCOBAR

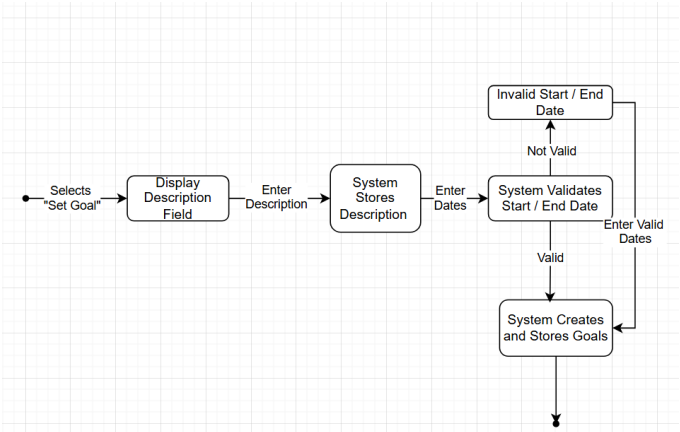
TABLE OF CONTENTS

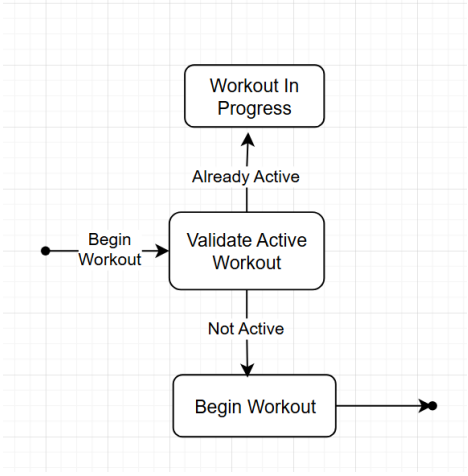
INTRODUCTION

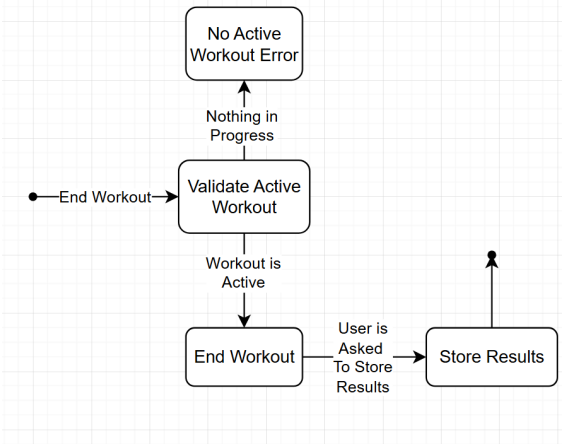
The following are design level diagrams that follow the logic of our fitness app, Lunki. They are intended to showcase a more in-depth look at how the app is supposed to operate, including its class structure, various important methods, and an overall view of the logic. Shown first is the state chart diagrams for each object, showcasing the different states objects can have depending on certain conditions. Shown next are our first-cut domain sequence diagrams, which expand upon the previous sequence diagrams seen in the System Requirements Document. Each chart goes through a specific use case and showcases the flow of data between objects, windows, and the actor. Similarly, shown last are our design level class diagrams. Each class diagram goes through a specific use case and shows the flow of data between classes while also showcasing specific methods. Pseudocode for these methods is shown last.

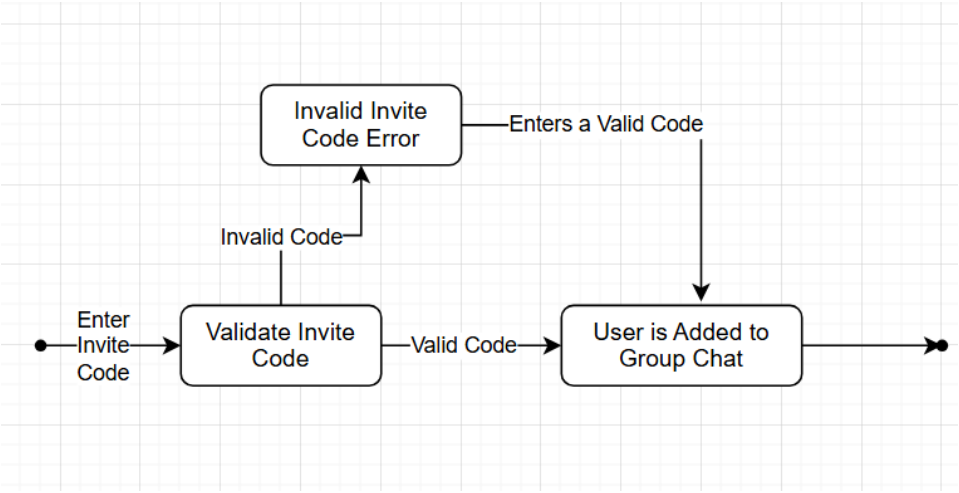
STATECHART DIAGRAMS

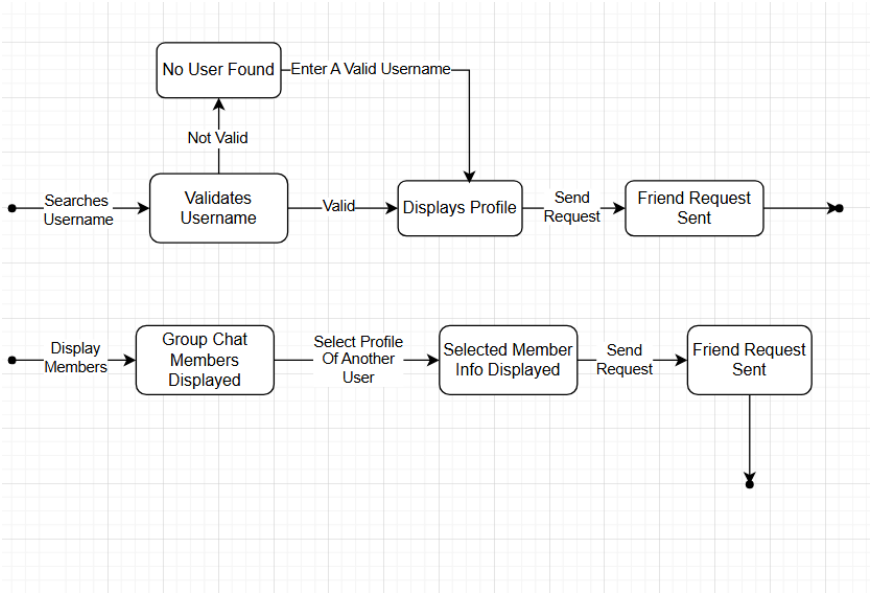


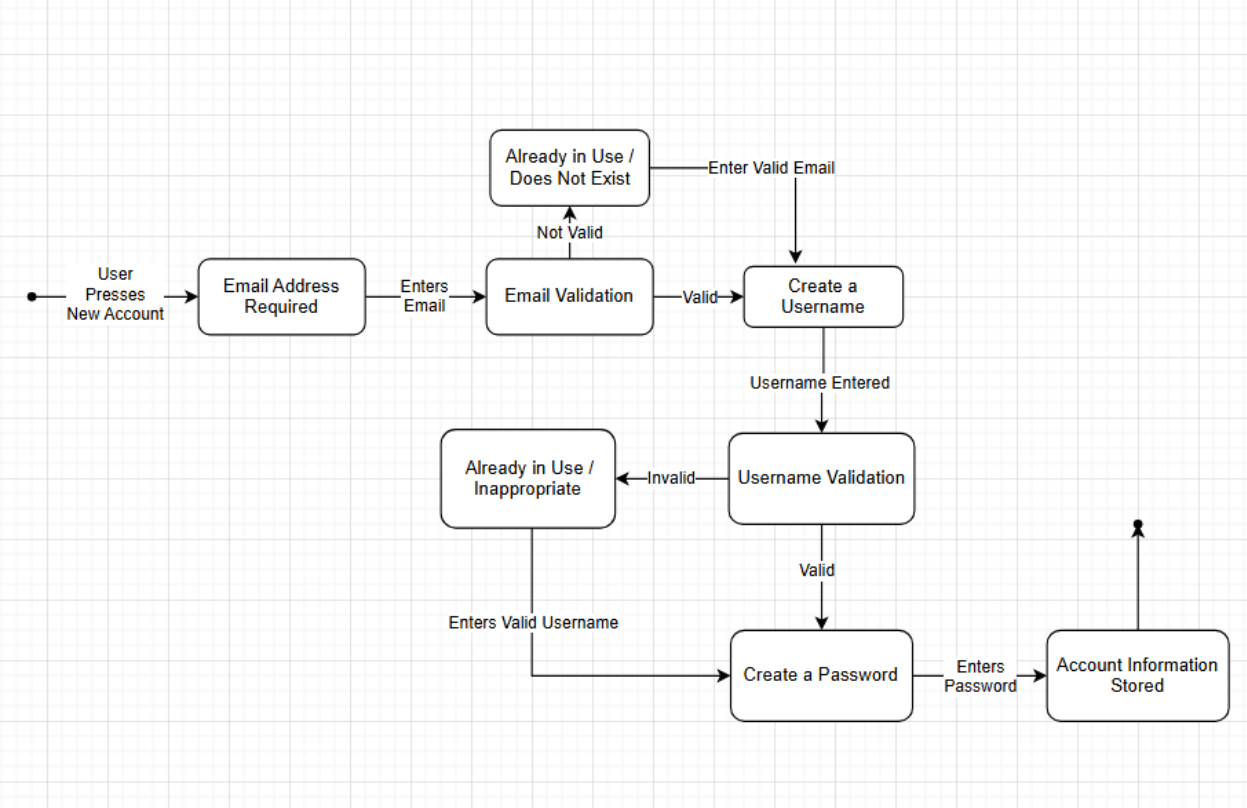


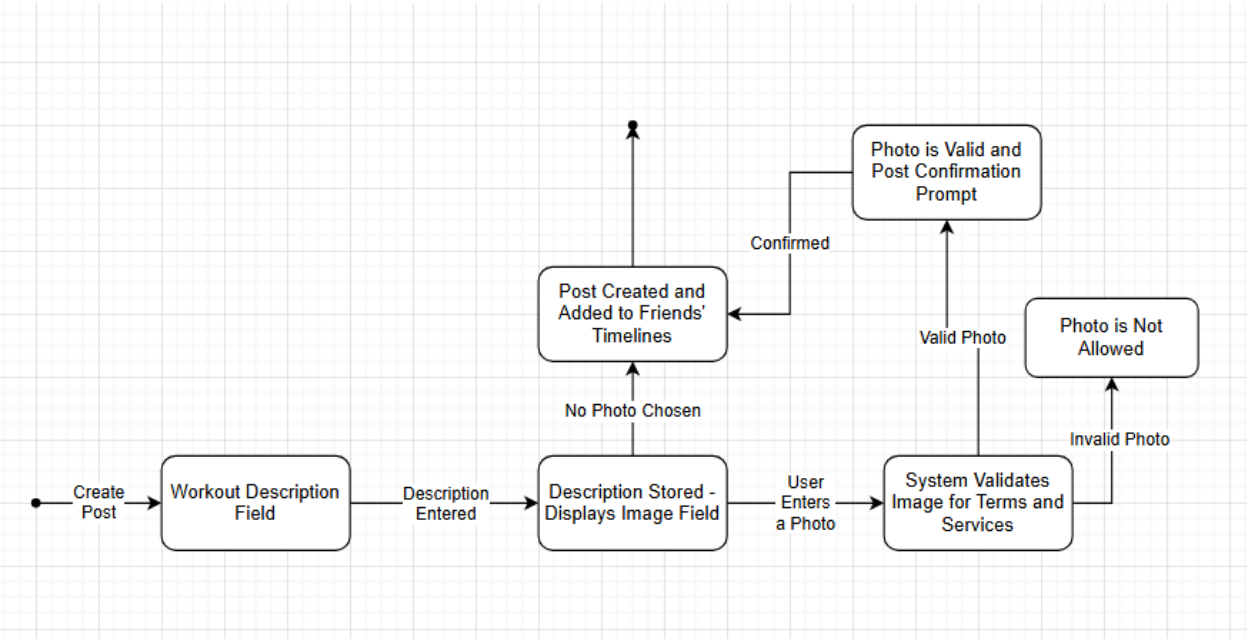




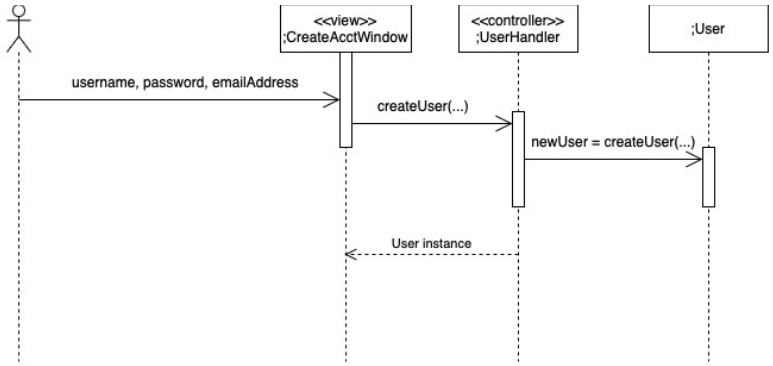

Invalid Invite Code Error
Enters a Valid Code
Invalid Code
Enter Invite Code
Validate Invite Code
Valid Code
User is Added to Group Chat

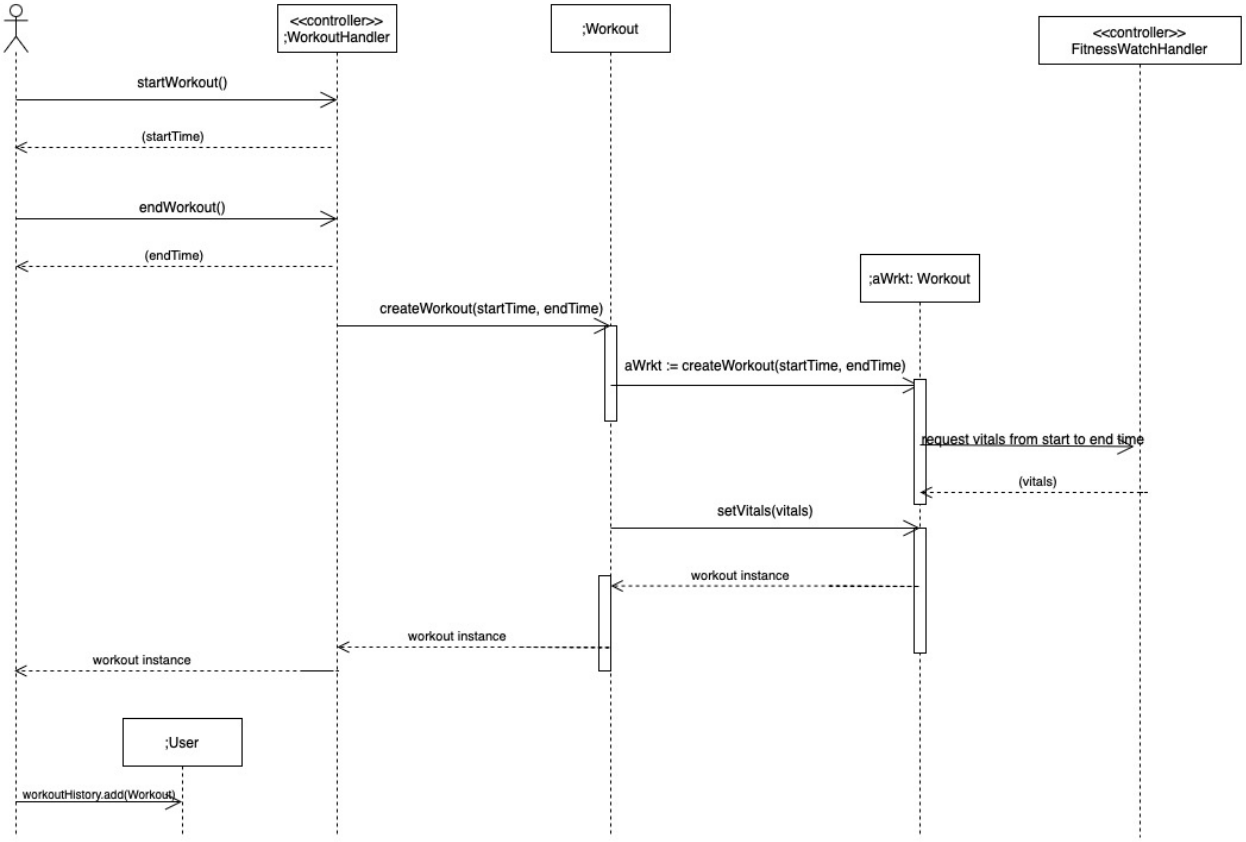

No User Found
Enter A Valid Username
Not Valid
Searches Username
Validates Username
Valid
Displays Profile
Send Request
Friend Request Sent
Display Members
Group Chat Members Displayed
Select Profile Of Another User
Selected Member Info Displayed
Send Request
Friend Request Sent

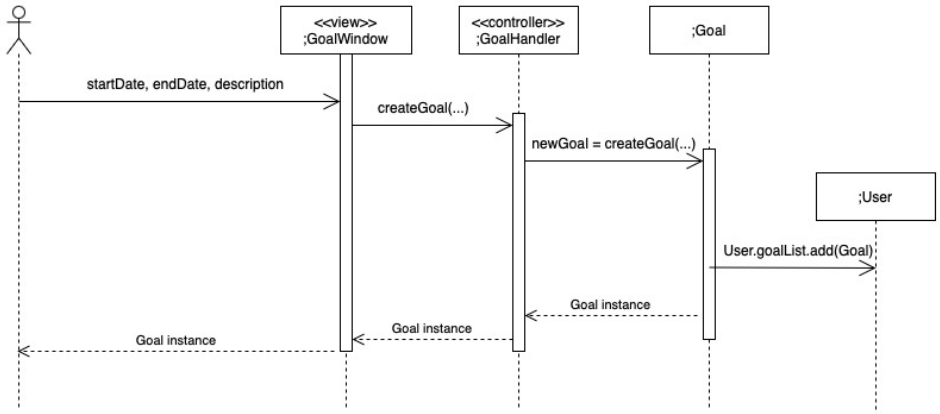


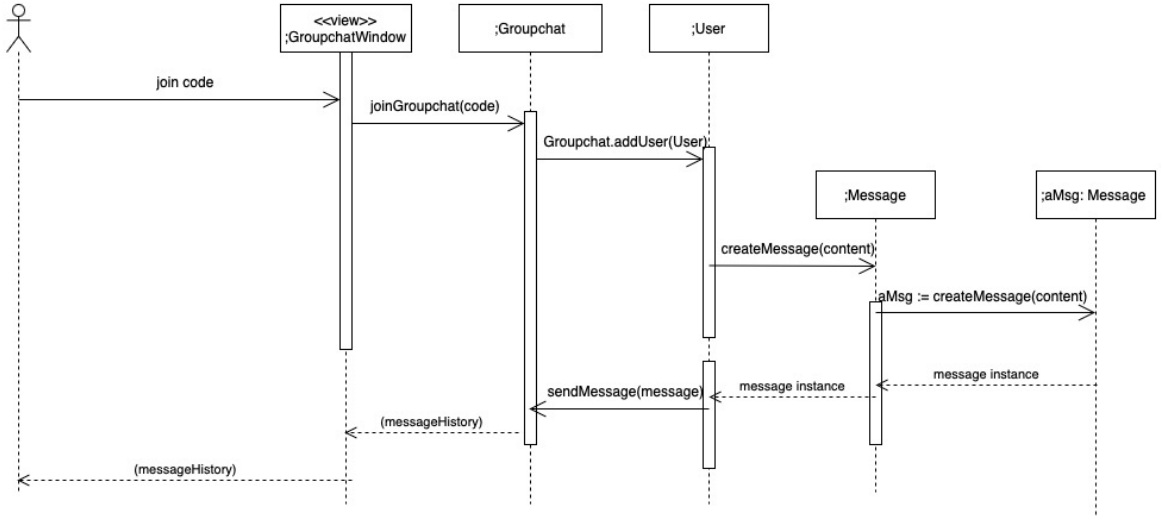


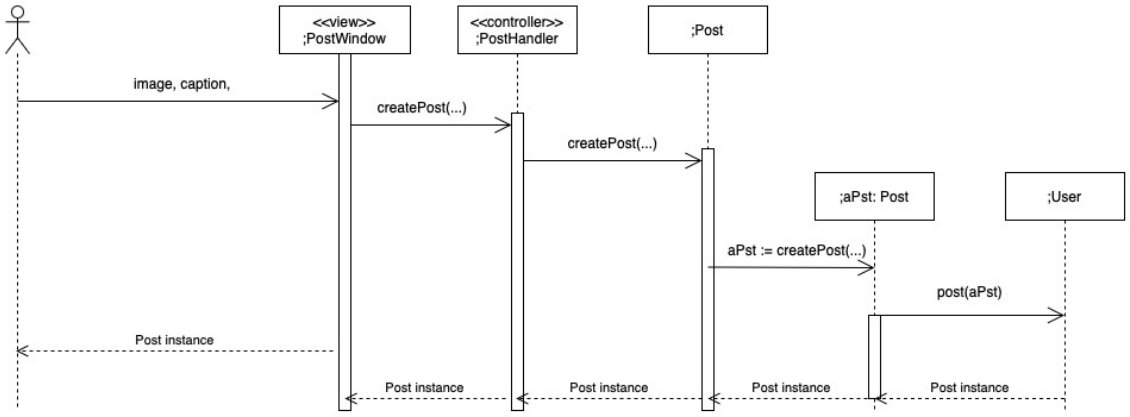
First Cut Domain Sequence Diagrams (one per use case)

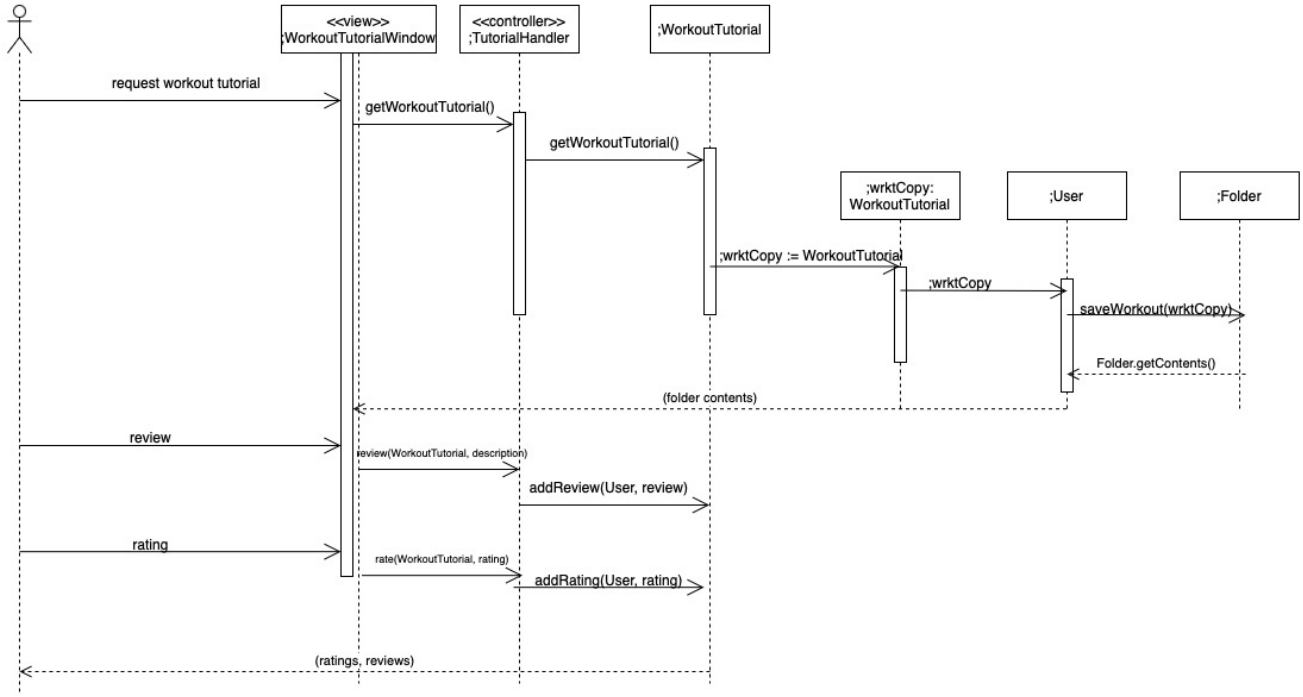

<<view>>
;CreateAcctWindow
<<controller>>
;UserHandler
;User
username, password, emailAddress
createUser(...)
newUser = createUser(...)
User instance


<<controller>>
;WorkoutHandler
;Workout
<<controller>>
FitnessWatchHandler
startWorkout()
(startTime)
endWorkout()
(endTime)
;aWrkt: Workout
createWorkout(startTime, endTime)
aWrkt := createWorkout(startTime, endTime)
request vitals from start to end time
(vitals)
setVitals(vitals)
workout instance
workout instance
workout instance
;User
workoutHistory.add(Workout)


<<view>>
;GoalWindow
<<controller>>
;GoalHandler
;Goal
;User
startDate, endDate, description
createGoal(...)
newGoal = createGoal(...)
User.goalList.add(Goal)
Goal instance
Goal instance
Goal instance


<<view>>
;GroupchatWindow
;Groupchat
;User
;Message
;aMsg: Message
join code
joinGroupchat(code)
Groupchat.addUser(User)
createMessage(content)
aMsg := createMessage(content)
message instance
message instance
sendMessage(message)
(messageHistory)
(messageHistory)

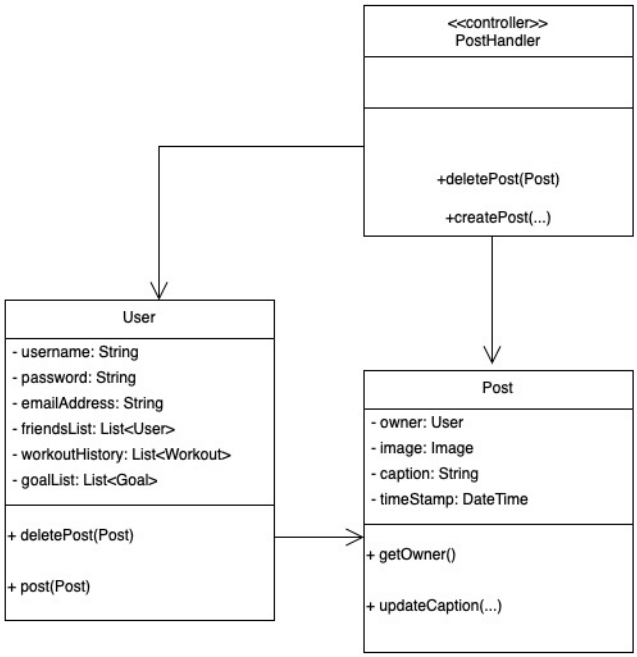

<<view>>
;PostWindow
<<controller>>
;PostHandler
;Post
;aPst: Post
;User
image, caption,
createPost(...)
createPost(...)
aPst := createPost(...)
post(aPst)
Post instance
Post instance
Post instance
Post instance
Post instance

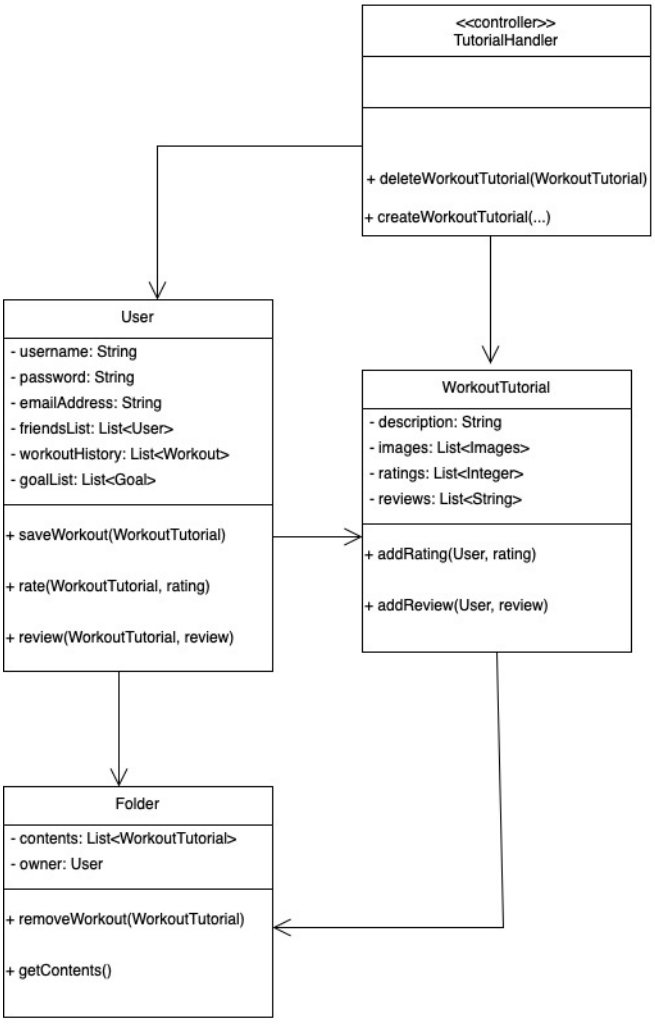


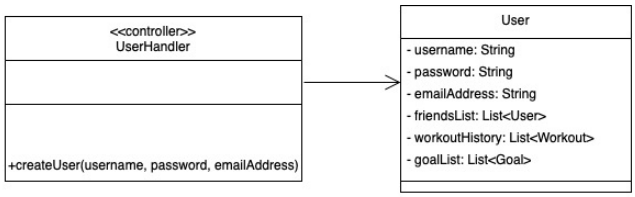
DESIGN LEVEL CLASS DIAGRAM

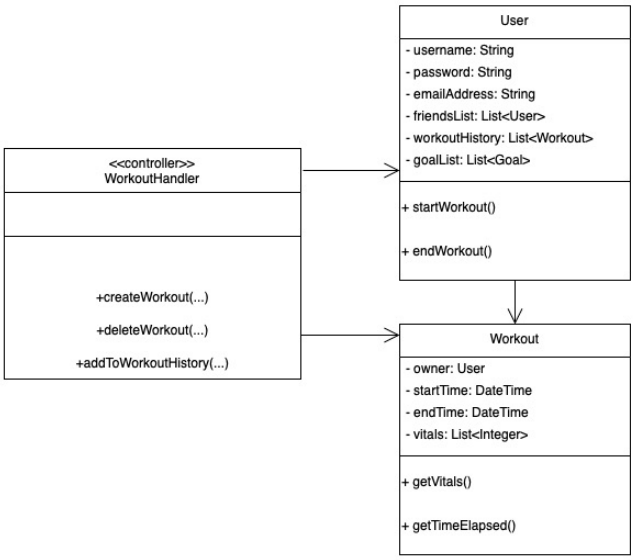
Refine your Class Diagram from your Analysis phase to contain the following additional information:

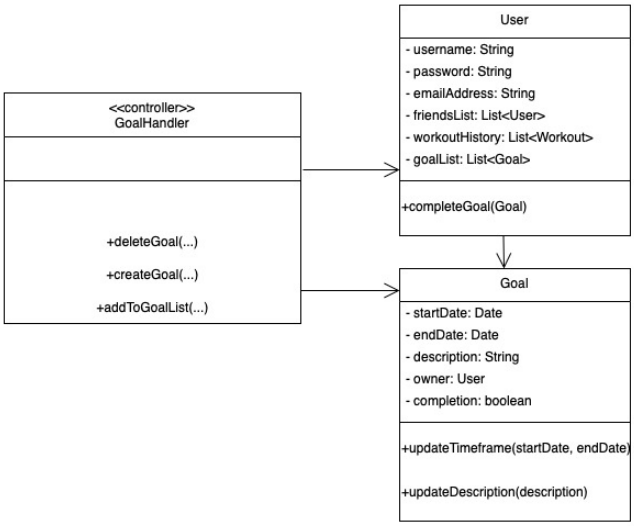
- *Attributes: data types, initial values, range of values (if appropriate)*
- *Methods: return types, parameters*
- *Pseudocode for each method*

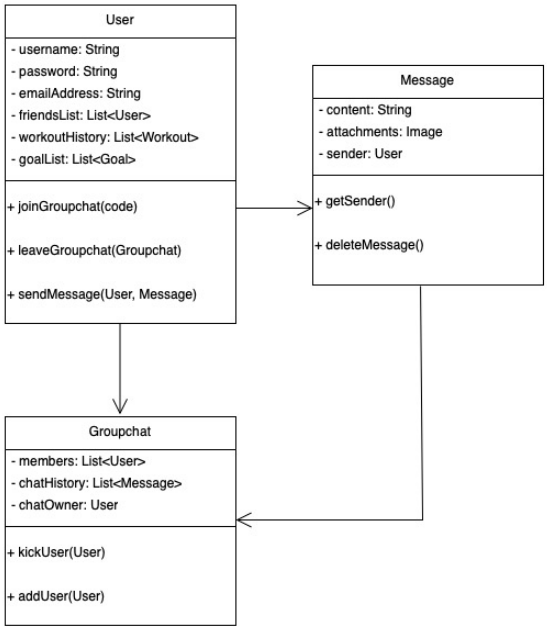

<<controller>>
PostHandler
+deletePost(Post)
+createPost(...)
User
- username: String
- password: String
- emailAddress: String
- friendsList: List<User>
- workoutHistory: List<Workout>
- goalList: List<Goal>
+ deletePost(Post)
+ post(Post)
Post
- owner: User
- image: Image
- caption: String
- timeStamp: DateTime
+ getOwner()
+ updateCaption(...)


<<controller>>
TutorialHandler
+ deleteWorkoutTutorial(WorkoutTutorial)
+ createWorkoutTutorial(...)
User
- username: String
- password: String
- emailAddress: String
- friendsList: List<User>
- workoutHistory: List<Workout>
- goalList: List<Goal>
+ saveWorkout(WorkoutTutorial)
+ rate(WorkoutTutorial, rating)
+ review(WorkoutTutorial, review)
WorkoutTutorial
- description: String
- images: List<Images>
- ratings: List<Integer>
- reviews: List<String>
+ addRating(User, rating)
+ addReview(User, review)
Folder
- contents: List<WorkoutTutorial>
- owner: User
+ removeWorkout(WorkoutTutorial)
+ getContents()


<<controller>>
UserHandler
+createUser(username, password, emailAddress)
User
- username: String
- password: String
- emailAddress: String
- friendsList: List<User>
- workoutHistory: List<Workout>
- goalList: List<Goal>


<<controller>>
WorkoutHandler
+createWorkout(...)
+deleteWorkout(...)
+addToWorkoutHistory(...)
User
- username: String
- password: String
- emailAddress: String
- friendsList: List<User>
- workoutHistory: List<Workout>
- goalList: List<Goal>
+ startWorkout()
+ endWorkout()
Workout
- owner: User
- startTime: DateTime
- endTime: DateTime
- vitals: List<Integer>
+ getVitals()
+ getTimeElapsed()


<<controller>>
GoalHandler
+deleteGoal(...)
+createGoal(...)
+addToGoalList(...)
User
- username: String
- password: String
- emailAddress: String
- friendsList: List<User>
- workoutHistory: List<Workout>
- goalList: List<Goal>
+completeGoal(Goal)
Goal
- startDate: Date
- endDate: Date
- description: String
- owner: User
- completion: boolean
+updateTimeframe(startDate, endDate)
+updateDescription(description)


User
- username: String
- password: String
- emailAddress: String
- friendsList: List<User>
- workoutHistory: List<Workout>
- goalList: List<Goal>
+ joinGroupchat(code)
+ leaveGroupchat(Groupchat)
+ sendMessage(User, Message)
Message
- content: String
- attachments: Image
- sender: User
+ getSender()
+ deleteMessage()
Groupchat
- members: List<User>
- chatHistory: List<Message>
- chatOwner: User
+ kickUser(User)
+ addUser(User)

//CREATING AN ACCOUNT

//User fills out username, password, and email fields

//IF fields are valid:

    //User account created with given information

//ELSE

    //User prompted to reenter information

//UPDATING USER INFO

//User fills out email change field

//IF Email is not taken:

    //User email updated

//ELSE:

    //User prompted to reenter email

//MANAGING FRIENDSHIPS

//User 1 fills out username field

//IF username corresponds to an active user:

    //Friend request sent to User 2

//ELSE:

    //User 1 prompted to enter a valid username

//IF User 2 accepts friend request:

    //User 2 is added to User 1's friendsList

    //User 1 is added to User 2's friendsList

//ELSE:

    //Friend request deleted

//CREATING A GOAL

//User enters start date, end date, and description of goal

//IF start date & end date are valid:

//Goal object created with specified information

//Goal completion set to false

//Goal added to User's goalList

//timer started

//ELSE:

//User is prompted to enter correct start/end date

//IF User completes goal:

//Goal completion set to true

//ELSE:

//WHILE Goal completion is false and 24 hours have elapsed:

//User is sent a push notification reminder

//timer reset

//STARTING AND LOGGING WORKOUTS

//User starts a workout

//Time of workout is logged

//User ends workout

//Time elapsed is calculated

//New workout object created with startTime, endTime, and owner

//IF User has a fitness watch connected to Lunki:

//Lunki fetches vital sign data from fitness watch

//Vitals information added to workout object (heart rate, calories, steps)

//RETURN workout Object

//ELSE:

//RETURN workout Object

//SENDING MESSAGES:

//User selects a valid groupchat or friend to message

//User enters message contents and optionally media (image, video, etc.)

//Message object created with specified info

//IF User is sending to a groupchat:

//Message object added to group chat's messageList

//ELSE:

//Message object sent to friend

//CREATING GROUPCHAT

//User selects friends

//Groupchat object created with User as owner and friends as members

//ADDING

//IF User is owner:

//User adds new friends to groupchat

//friend added to groupchat members

//ELSE:

//User invalid permissions message

//KICKING

//IF User is owner:

//User kicks specified friend

//friend removed from groupchat members

//ELSE:

//User invalid permissions message

//POSTING A WORKOUT

//IF User gives permission to Lunki to access camera roll:

//User enters image and caption in post fields

//IF image/caption follow guidelines:

//New post object created with User as owner, and given image/caption as fields

//post object added to User's postHistory

//(Not pseudocode but included for clarity: a user's feed consists of the posts within their postHistory list + all their friends' postHistory posts sorted chronologically)

//ELSE:

//User is warned of guidelines violation and prompted to provide a better image/caption

//ELSE:

//do nothing

//VIEWING/SAVING A WORKOUT TUTORIAL

//User selects workout tutorial

//Lunki fetches workoutTutorial object and displays its description, images, reviews, and ratings

//IF User selects save workout:

//A copy of the workoutTutorial object is created and added to the User's folder

//ELSE:

//do nothing

//RATING/REVIEWING A WORKOUT TUTORIAL

//User selects workout tutorial

//Lunki fetches workoutTutorial object and displays its information

//IF User selects review workout:

//User is prompted to enter a review (String)

//review added to the workoutTutorial object's reviews along with the User who left said review (Map)

//ELSE IF User selects rate workout:

//User is prompted to enter a rating (Integer)

//rating added to the workoutTutorial object's ratings along with the user who left said rating (Map)