UNIVERSITY OF COLORADO BOULDER

CSCI 5448: PROJECT PART 2

FITNESS MANAGER

Chaitanya Soma, Jordan Peters, Ramnarayan Krishnamurthy

10/13/2016

Table of Contents

TEAM 2

TITLE 2

PROJECT SUMMARY 2

PROJECT REQUIREMENT 2

Business Requirement 2

User Requirement 3

Functional Requirement 4

Non-Functional Requirement 4

USE CASES 5

Use Case Diagram 5

Use Case Documents 6

ACTIVITY DIAGRAM 41

DATA STORAGE 44

UI MOCKUP 44

SEQUENCE DIAGRAMS 50

CLASS DIAGRAMS 53

TEAM

Chaitanya Soma

Jordan Peters

Ramnarayan Krishnamurthy

TITLE

Fitness Manager

PROJECT SUMMARY

A personal fitness manager that allows users to create profile, track fitness goals, interact with trainers and provide feedback.

PROJECT REQUIREMENT

Business Requirement

ID	Requirement	Topic Area	User	Priority
BR-001	User profiles are deleted after 2 years of inactivity	Administration	All	Low
BR-002	All User information is stored in a database	Administration	All	Low

User Requirement

ID	Requirement	Topic Area	User	Priority
UR-001	As a Client, I need to be able to create a profile so that I can start tracking my fitness.	Profile	Client	High
UR-002	As a Trainer, I need to be able create a trainer profile so I can let clients find me.	Profile	Trainer	High
UR-003	As an Admin, I need to be able to create an admin profile so that I can monitor and manage user and trainer accounts.	Profile	Admin	High
UR-004	As a User, I need to be able to login to the system so that I can access all of its features	Login	All	High
UR-005	As a User, I need to be able to reset password through security question so that I can re-login in case I forget my password	Login	All	Medium
UR-006	As a User, I need to be able to reset password through my email so that I can re-login in case I forget my password	Login	All	Medium
UR-007	As a User, I need to able to update my profile so that I can keep my information up to date	Profile	User	Medium
UR-008	As a User, I need to be able to delete my profile so that I can leave the system	Profile	Client, Trainer	Low
UR-009	As an Admin, I need to be able to approve trainer profiles so I can verify the trainer credentials	Profile	Admin	Low
UR-010	As a Client, I need to add my daily fitness information so I can track my fitness progress	Health Tracking	Client	High
UR-011	As a Client, I need to modify my daily fitness information so I can modify my fitness progress	Health Tracking	Client	High
UR-012	As a Client, I need to delete my daily fitness information so I can remove entries	Health Tracking	Client	High
UR-013	As a Client, I need to add my fitness goals so I can track my them against my progress	Health Tracking	Client	High
UR-014	As a Client, I need to modify my fitness goals so I can update my goals as necessary	Health Tracking	Client	High
UR-015	As a Client, I need to delete my fitness goals so I can remove entries	Health Tracking	Client	High
UR-016	As a Client, I need to be able to schedule an appointment so that I can meet with a trainer that matches my requirement	Scheduling	Client	Medium
UR-017	As a Client, I need to be able to cancel an appointment so others can use that slot	Scheduling	Client	Medium
UR-018	As a Client, I need to be able to rate and review trainers to help other clients make an informed decisions	Reviews	Client	Medium

Functional Requirement

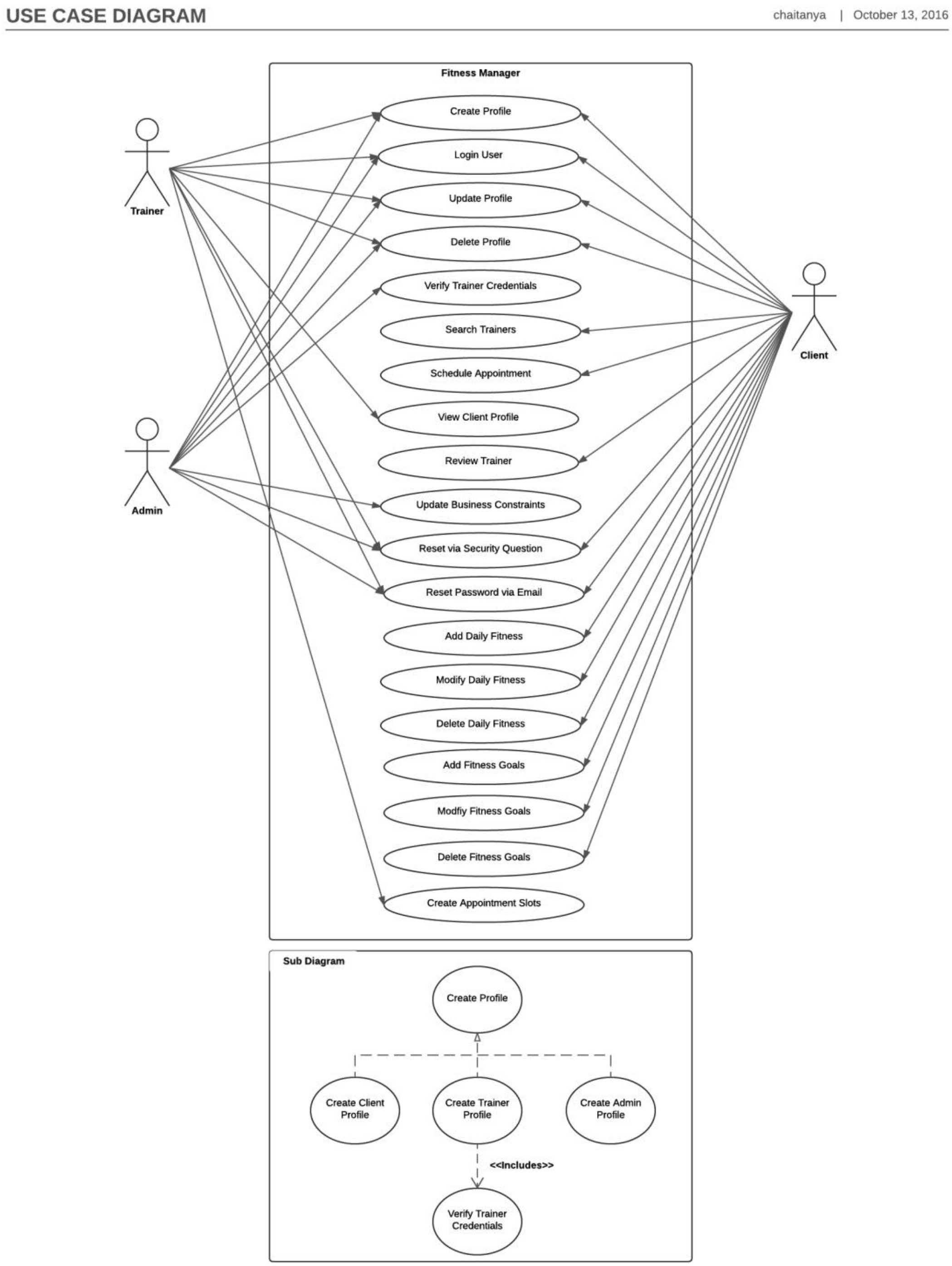
ID	Requirement	Topic Area	User	Priority
FR-001	Verification email sent when a profile is created	Profile	All	Medium
FR-002	Systems provides reminder notification to users for appointments	Scheduling	All	Medium

Non-Functional Requirement

N/A

USE CASES

Use Case Diagram



Use Case Documents

Use Case ID:	UC-01
Use Case Name:	Create Client Profile
Description:	Creates a client profile and stores client information in the system

Actor:	Client		
Pre-conditions:	Client clicks the “Create profile” button and selects “client” from the drop down menu.		
Post-conditions:	Client profile is successfully created and client can login with the username and password		
Frequency of Use:	Medium		
Flow of Events:		Actor Action	System Response
	1		Systems displays create profile page and fields to enter username and password
	2	Client enters username and password and clicks the submit button	Systems verifies the username and displays a page to enter client profile information
	3	Client enters	System verifies all required fields, creates a client account

		a) Name b) Email Address c) Location d) Gender e) Height f) Weight g) Age h) Password i) Security question And clicks the submit button	and sends a confirmation email to the client
	4	Client views the approval email	
Variations:			
Exceptions:	Step 2. If client’s username is not unique go back to step 1 Step 3. If client enters invalid information go back to system step 2		
Developer Notes:			

Use Case ID:	UC-02
Use Case Name:	Create Trainer Profile
Description:	Creates a trainer profile and stores trainer information in the system

Actor:	Trainer		
Pre-conditions:	Trainer clicks the “Create profile” button and selects “Trainer” from the drop down menu.		
Post-conditions:	Trainer profile is successfully created and trainer can login with the username and password. If trainer is rejected then reason for rejection is notified through email.		
Frequency of Use:	Low - Medium		
Flow of Events:		Actor Action	System Response
	1		Systems displays “create_profile_form" with fields to enter username and password
	2	Trainer enters username and password and clicks okay	Systems verifies the username and displays a “trainer_profile_form” to enter trainer profile information
	3	Trainer enters a) Name b) Email Address c) Location	System verifies all required fields and sends a notification email to the admin

		d) Gender e) Expertise f) Summary g) Experience h) Security question And clicks the submit button	
	4		After admin approval, send approval email to the user
	5	Trainer views the approval email	
Variations:			
Exceptions:	2 : If username already exists go back to step 1 of system response 3 : If profile information is incomplete or incorrect go back to step 2 of system response 4 : If admin rejects the profile, send rejection email to the user		
Developer Notes:			

Use Case ID:	UC-03
Use Case Name:	Create Admin Profile
Description:	Creates an admin profile and stores admin information in the system

<table>
<tr><td>Actor:</td><td colspan="3">Admin</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">Admin clicks the “Create profile” button and selects “admin” from the drop down menu.</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">Admin profile is successfully created and admin can login with the username and password</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="4">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>Systems displays create profile page and fields to enter username, password and “admin code”</td></tr>
<tr><td>2</td><td>Client enters username, password, admin code and click submit button</td><td>Systems verifies the admin code, username and displays a page to enter client profile information</td></tr>
<tr><td>3</td><td>Admin enters
a) Name
b) Email Address</td><td>System verifies all required fields, creates an admin account and sends a confirmation email</td></tr>
</table>

		c) Security question And clicks the submit button	to the admin
	4	Admin views the confirmation email	
Variations:			
Exceptions:	2. If admin code and username are invalid go back to step 1 3. If required fields are invalid or incomplete go back to system step 2		
Developer Notes:			

Use Case ID:	UC-04
Use Case Name:	Login User
Description:	User logs into the system

<table>
<tr><td>Actor:</td><td colspan="3">Client, Trainer and Admin</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">User is not already logged in
User is on the login page
User has an active account</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">User is logged into the system and can access the home page</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Very High</td></tr>
<tr><td rowspan="4">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>Systems displays “username” and “password” fields</td></tr>
<tr><td>2</td><td>User enters username, password and clicks login</td><td>System verifies the username and password and redirects the user to the homepage</td></tr>
<tr><td>3</td><td>User views home page</td><td></td></tr>
<tr><td>Variations:</td><td colspan="3">2. Reset password via security question, Reset via email</td></tr>
<tr><td>Exceptions:</td><td colspan="3"></td></tr>
</table>

Developer Notes:	

Use Case ID:	UC-05
Use Case Name:	Reset Password via Security Question
Description:	User resets the password

<table>
<tr><td>Actor:</td><td colspan="3">Client, Trainer and Admin</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">User is not already logged in
User is on the login page
User has an active account
User enters a valid username
User clicks reset password</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">User has a new password to log in to the system</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="3">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>System displays a page asking the user to “Reset via security question” or “Reset via email”</td></tr>
<tr><td>2</td><td>User selects “Reset via security question”</td><td>System displays the security question</td></tr>
</table>

	3	User answers the security question and clicks submit	System verifies the answer to the security question and asks user to enter new password
	4	User enters the new password and clicks submit	System updates the user password and displays the login page
	5	User views the login page	
Variations:			
Exceptions:	3. If user enters incorrect answer go back to system step 2		
Developer Notes:			

Use Case ID:	UC-06
Use Case Name:	Reset Password via email
Description:	User resets the password

<table>
<tr><td>Actor:</td><td colspan="3">Client, Trainer and Admin</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">User is not already logged in
User is on the login page
User has an active account
User enters a valid username
User clicks reset password</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">User has a new password to log in to the system</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="4">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>System displays a page asking the user to “Reset via security question” or “Reset via email”</td></tr>
<tr><td>2</td><td>User selects “Reset via security question”</td><td>System displays a page to enter username</td></tr>
<tr><td>3</td><td>User enters the username and clicks submit</td><td>System extract the user’s email address, sends an email with temporary password and</td></tr>
</table>

			displays a page to enter temporary password and new password
	4	User enters the temporary and new password and clicks submit	System verifies the information, updates the password and displays the login page
	5	User views the login page	
Variations:			
Exceptions:	4. If user enters incorrect temporary password go to system step 2		
Developer Notes:			

Use Case ID:	UC-07
Use Case Name:	Update Profile
Description:	User updates profile information

Actor:	Client, Trainer and Admin		
Pre-conditions:	User is already logged in User is on the home page User clicks update profile		
Post-conditions:	User profile information is updated		
Frequency of Use:	Low		
Flow of Events:		Actor Action	System Response
	1		Systems fetches the user profile information, and displays editable fields to the user
	2	User updates the profile information and clicks submit	1. System verifies all required fields and updates the profile information 2. System displays the updated fields to the user
	3	User can view the updated	

		profile information	
Variations:			
Exceptions:	2. If user enters incorrect information in the update profile fields go back to system step 1		
Developer Notes:			

Use Case ID:	UC-08
Use Case Name:	Delete Profile
Description:	Delete user profile and all associated information

Actor:	Client, Trainer and Admin		
Pre-conditions:	User is already logged in User is on the home page		
Post-conditions:	User profile and all associated information is permanently deleted. Login page is displayed		
Frequency of Use:	Low		
Flow of Events:		Actor Action	System Response
	1	User clicks “Delete Profile”	1. System deletes the user profile and all associated information. 2. System displays the login page to the user
	2	User views the login page	
Variations:			
Exceptions:			
Developer Notes:			

Use Case ID:	UC-09
Use Case Name:	Review Trainer
Description:	A Client can rate and review a Trainer following a session

Actor:	Client, Trainer		
Pre-conditions:	Client is logged in Client is on the “Rate & Review” page of the specific trainer		
Post-conditions:	Trainer’s Page displays the updated Rating/Review		
Frequency of Use:	Medium		
Flow of Events:		Actor Action	System Response
	1	Client clicks on Rate & Review that appears against the trainer’s name	System displays a page with an option of: 1. Rating out of 5 2. Writing a Review
	2.	Client enters the appropriate information and clicks sumits	System 1. Saves the rating and review 2. Updates the existing rating of the trainer 3. Displays the home page

	3	Client views the home page	
Variations:			
Exceptions:			
Developer Notes:	Rating is computed as a simple average of the existing rating and new rating provided		

Use Case ID:	UC-10
Use Case Name:	Search Trainer
Description:	A Client can search for a Trainer and sort by profile parameters

Actor:	Client		
Pre-conditions:	Client is logged in Client is on the “Search Trainers” page		
Post-conditions:	Client is provided with a list of the names of Trainers sorted by the parameter of choice		
Frequency of Use:	High		
Flow of Events:		Actor Action	System Response
	1		Displays the ”Search Trainers” page with checkboxes for sortable parameters
	2	Client selects the sortable parameters and enters the corresponding value	System 1. queries for the given parameter 2. Sorts the results 3. Displays names
	3	Client views the sorted trainer names	

Variations:	
Exceptions:	
Developer Notes:	

Use Case ID:	UC-11
Use Case Name:	Add Daily Fitness
Description:	Allow the client to add fitness information like amount of calories consumed, step count etc.

<table>
<tr><td>Actor:</td><td colspan="3">Client</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">Client is logged in
Client is in the “View daily fitness information” page</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">The system displays the updated daily fitness information to reflect the client's changes</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">High</td></tr>
<tr><td rowspan="4">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>System displays the daily fitness information with add, delete and update buttons</td></tr>
<tr><td>2</td><td>Client clicks “Add” button</td><td>System display an empty form/fields to add fitness information</td></tr>
<tr><td>3</td><td>Client enters fitness information based on the displayed fields and clicks</td><td>1. System Validates the information
2. The information is added</td></tr>
</table>

		the submit button	in database 3. Updated page is displayed
	4	Client Views the updated daily fitness information	
Variations:	3. If the fitness information is incorrect, go back to step 2 system response		
Exceptions:			
Developer Notes:			

Use Case ID:	UC-12
Use Case Name:	Modify Daily Fitness
Description:	Allow the client to modify daily fitness information like amount of calories consumed, step count etc.

<table>
<tr><td>Actor:</td><td colspan="3">Client</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">Client is logged in
Client is in the “View daily fitness information” page</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">The system displays the updated daily fitness information to reflect the client's changes</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="4">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>System displays the daily fitness information with add, delete and update buttons</td></tr>
<tr><td>2</td><td>client clicks “Modify” Button</td><td>System make the specific fitness info editable and displays the form to the user</td></tr>
<tr><td>3</td><td>User updates the fields/fitness information</td><td>1. System Validates the information
2. The information is</td></tr>
</table>

		and clicks submit button	updated in database 3. Updated page is displayed
	4	Client Views the updated daily fitness information	
Variations:			
Exceptions:	3. If the fitness information is incorrect, go back to step 2 system response		
Developer Notes:			

Use Case ID:	UC-13
Use Case Name:	Delete Daily Fitness
Description:	Allow the client to delete daily fitness information like amount of calories consumed, step count etc.

Actor:	Client		
Pre-conditions:	Client is logged in Client is in the “View daily fitness information” page		
Post-conditions:	The system displays the updated daily fitness information to reflect the client's changes		
Frequency of Use:	Low		
Flow of Events:		Actor Action	System Response
	1		System displays the daily fitness information with add, delete and update buttons
	2	If client clicks “Delete” button	1. The system deletes the corresponding field 2. The information is updated in database 3. Updated page is displayed

	3	Client views the updated daily fitness information	
Variations:			
Exceptions:			
Developer Notes:			

Use Case ID:	UC-14
Use Case Name:	Approve Profile
Description:	Admin validates trainer credentials and approves him to create a profile

<table>
<tr><td>Actor:</td><td colspan="3">Admin</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">Trainer filled in profile information and is waiting for admin to verify his/her profile
Admin is in the “approve profiles page”</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">Trainer profile is successfully create in the system</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="3">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td></td><td>System displays the trainer profile information, approve and reject buttons</td></tr>
<tr><td>2</td><td>Trainer clicks the approve button</td><td>The system marks the trainer profile as approved</td></tr>
<tr><td>Variations:</td><td colspan="3"></td></tr>
<tr><td>Exceptions:</td><td colspan="3">2. If the admin clicks denied, the trainer profile is set to denied</td></tr>
</table>

Developer Notes:	

Use Case ID:	UC-15
Use Case Name:	Schedule Appointment
Description:	A Client can schedule an appointment with a trainer that the trainer has created.

<table>
<tr><td>Actor:</td><td colspan="3">Client</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">Client is logged in
Client is on the “”Trainer’s Info” page
Trainer has created an appointment</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">Client has an appointment slot with the trainer booked</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Medium</td></tr>
<tr><td>Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td></td><td>1</td><td></td><td>Displays the “Trainer’s Info” page with all available appointments that trainer has set</td></tr>
<tr><td></td><td>2</td><td>Client selects any of the open appointments</td><td>1) System adds user to that appointment
2) System notifies trainer</td></tr>
<tr><td>Variations:</td><td colspan="3">Client selects an already booked appointment and is told they must select another option.</td></tr>
</table>

Exceptions:	
Developer Notes:	

Use Case ID:	UC-16
Use Case Name:	Create Appointment Slots
Description:	A trainer can create appointments that clients can sign up for.

Actor:	Trainer		
Pre-conditions:	Trainer is logged in Trainer is on the “”Create Appointment” page		
Post-conditions:	Client has an appointment slot with the trainer booked		
Frequency of Use:	High		
Flow of Events:		Actor Action	System Response
		Selects what times the appointments will happen, how long they will last and where they will happen.	1) Adds appointment slots to the database 2) Displays a confirmation
Variations:			
Exceptions:			
Developer Notes:			

Use Case ID:	UC-17
Use Case Name:	Add Fitness Goals
Description:	Add fitness goals to track progress

Actor:	Client		
Pre-conditions:	User is logged in User is on the home page		
Post-conditions:	User’s fitness goals for the month are populated and display on the home page		
Frequency of Use:	Medium		
Flow of Events:		Actor Action	System Response
	1	User clicks “Add Fitness Goals”	System displays the “fitness goals” page
	2	User enters the fitness goals, field-wise	1) System updates goals and saves the entries to the database 2) System redirects the User to the home page and displays the fitness goals
Variations:			

Exceptions:	
Developer Notes:	

Use Case ID:	UC-18
Use Case Name:	Modify Fitness Goals
Description:	User can modify the fitness goals

<table>
<tr><td>Actor:</td><td colspan="3">Client</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">User is logged in
User is on the home page</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">User’s fitness goals are updated and displayed on the homepage</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="2">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td>User clicks on “Modify Fitness Goals”</td><td>System displays an editable version of the fitness goals form</td></tr>
<tr><td></td><td>2</td><td>User enters the new goals</td><td>System updates the database with the new entries and redirects the user to the homepage</td></tr>
<tr><td>Variations:</td><td colspan="3"></td></tr>
<tr><td>Exceptions:</td><td colspan="3"></td></tr>
</table>

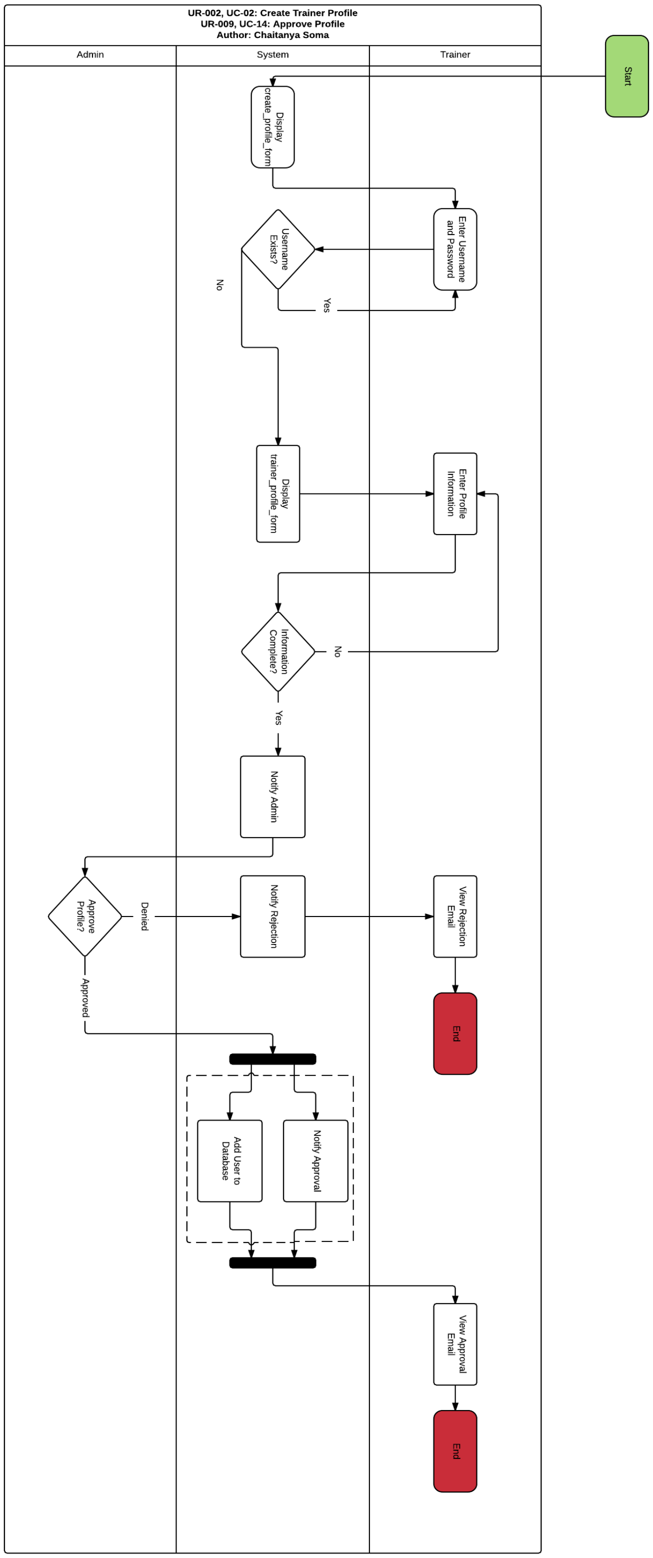
Developer Notes:	

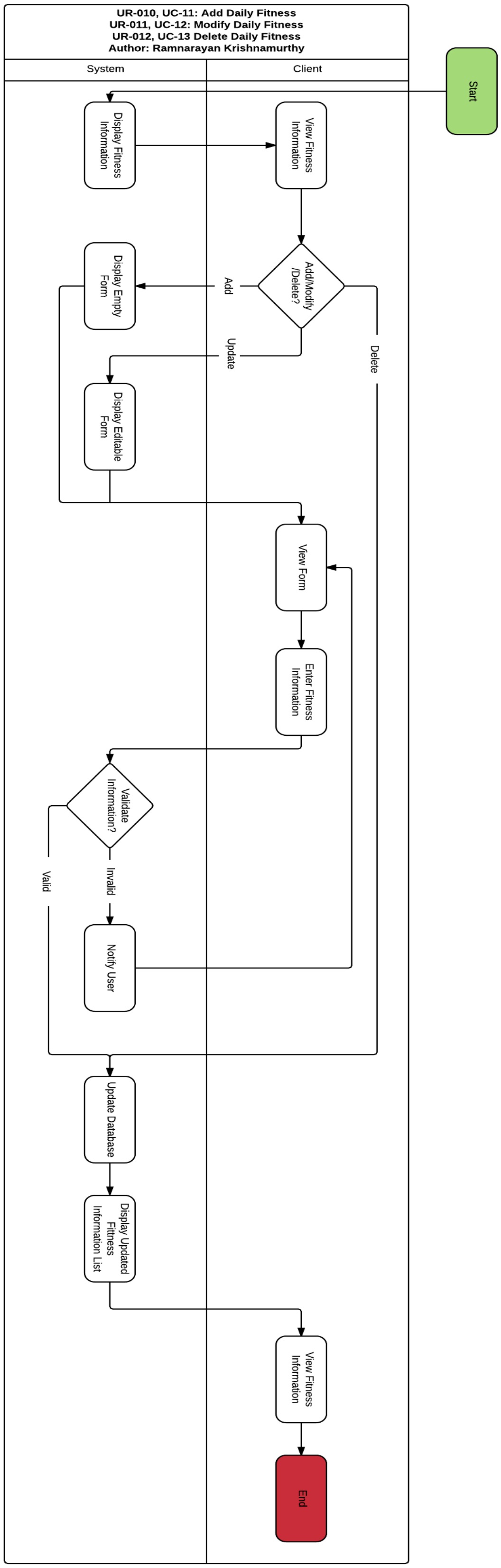
Use Case ID:	UC-19
Use Case Name:	Delete Fitness Goals
Description:	User can delete the set fitness goals

<table>
<tr><td>Actor:</td><td colspan="3">Client</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">User is logged in
User is on the home page</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">The respective User’s fitness goals are deleted</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Low</td></tr>
<tr><td rowspan="2">Flow of Events:</td><td></td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td>User clicks on “Delete Fitness Goals”</td><td>Systems deletes the fitness goals and updates the database and redirects the User to the homepage</td></tr>
<tr><td>Variations:</td><td colspan="3"></td></tr>
<tr><td>Exceptions:</td><td colspan="3"></td></tr>
<tr><td>Developer Notes:</td><td colspan="3"></td></tr>
</table>

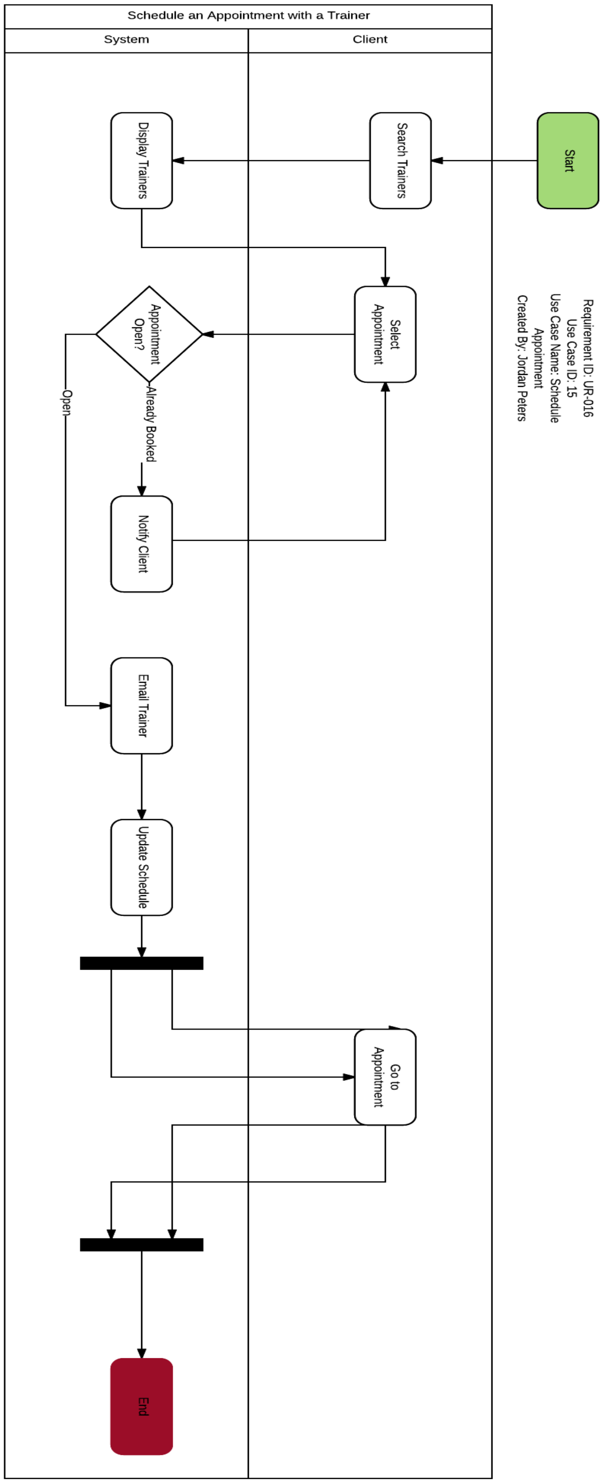
ACTIVITY DIAGRAM

(A full blow up of the activity diagrams are provided in the raw_images folder in github)





UR-010, UC-11: Add Daily Fitness
UR-011, UC-12: Modify Daily Fitness
UR-012, UC-13 Delete Daily Fitness
Author: Ramnarayan Krishnamurthy



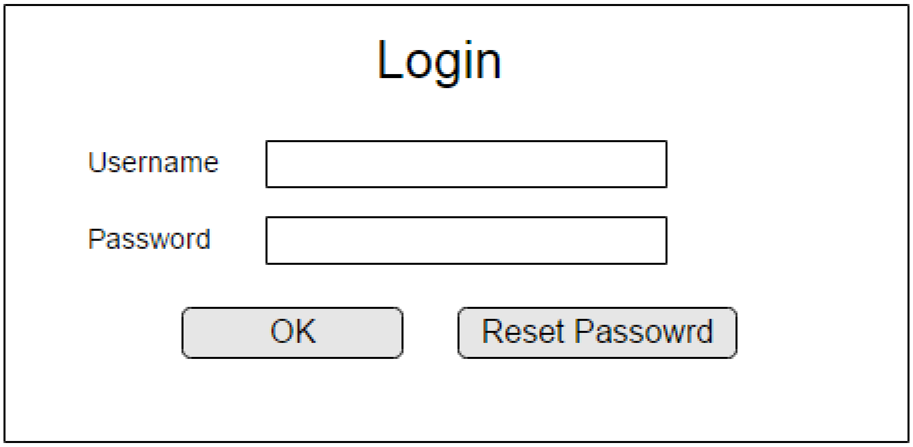
Requirement ID: UR-016
Use Case ID: 15
Use Case Name: Schedule Appointment
Created By: Jordan Peters

DATA STORAGE

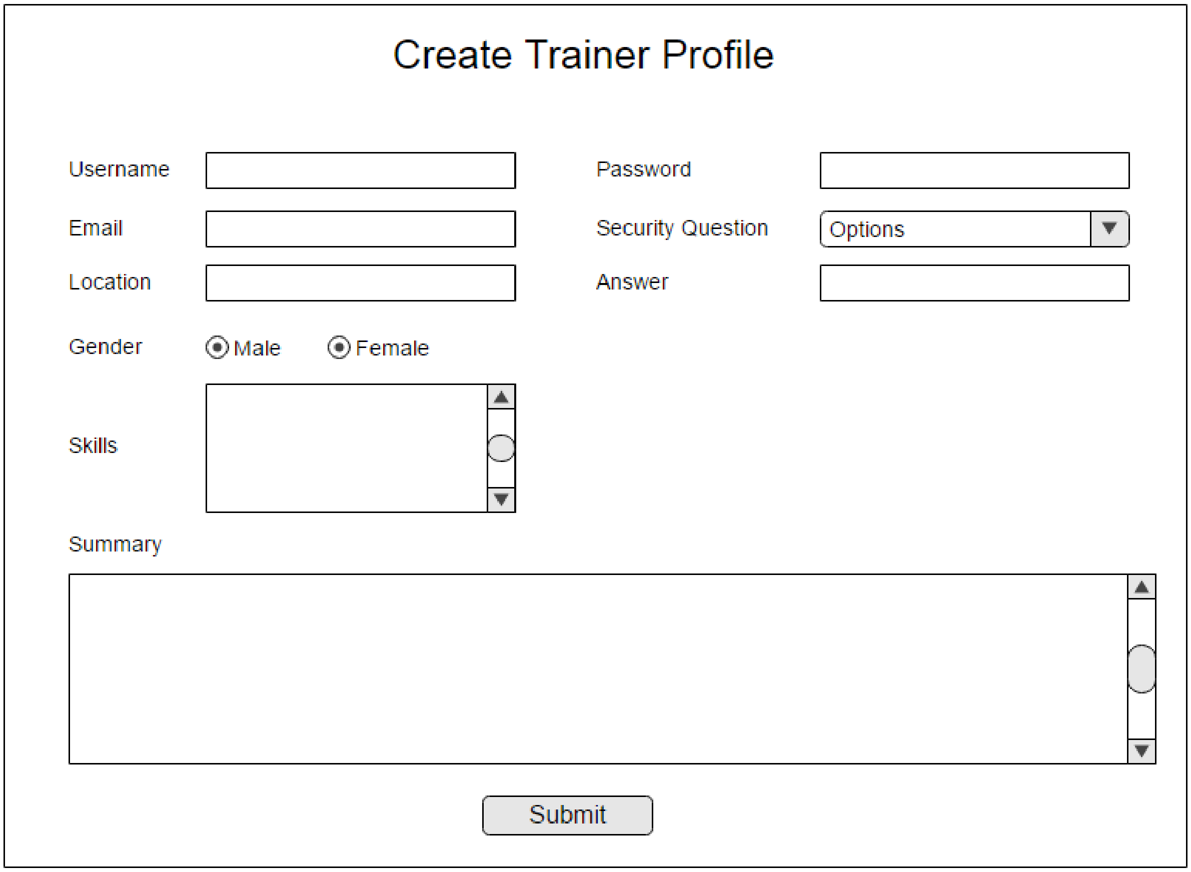
- We intend to use MongoDB, NoSQL database.
- All User Profiles and Information is stored in JSON format.

UI MOCKUP

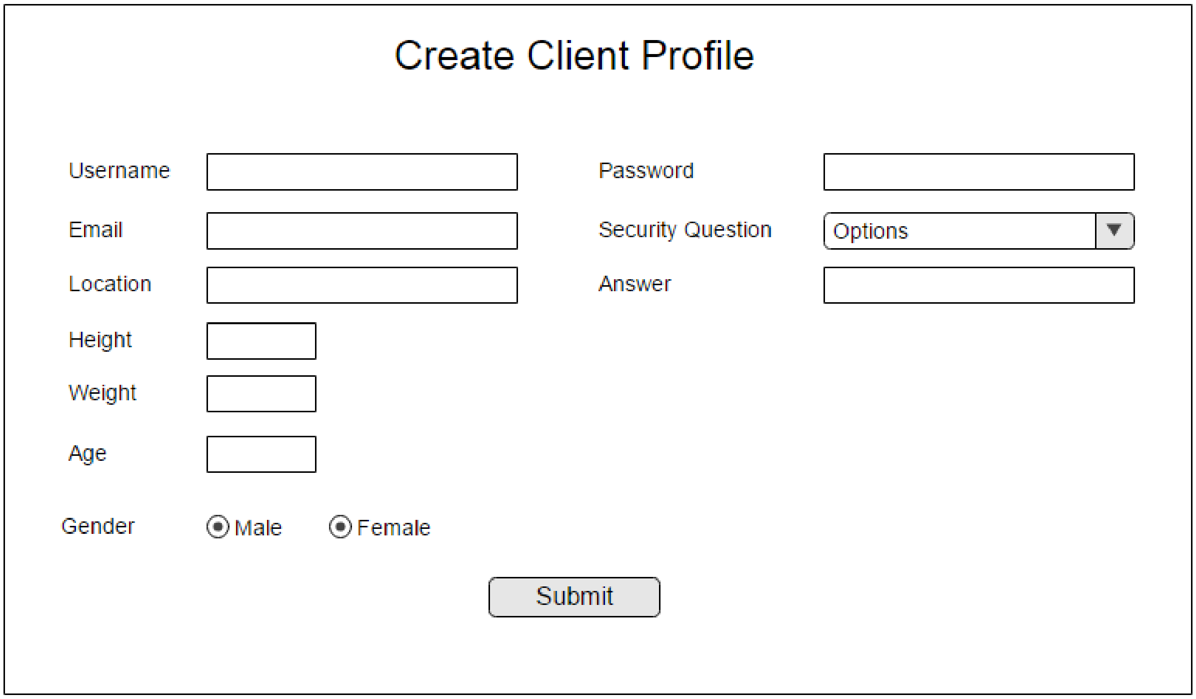
Login Page



Create Trainer Profile



Create Client Profile



Create Client Profile

Username

Password

Email

Security Question Options ▼

Location

Answer

Height

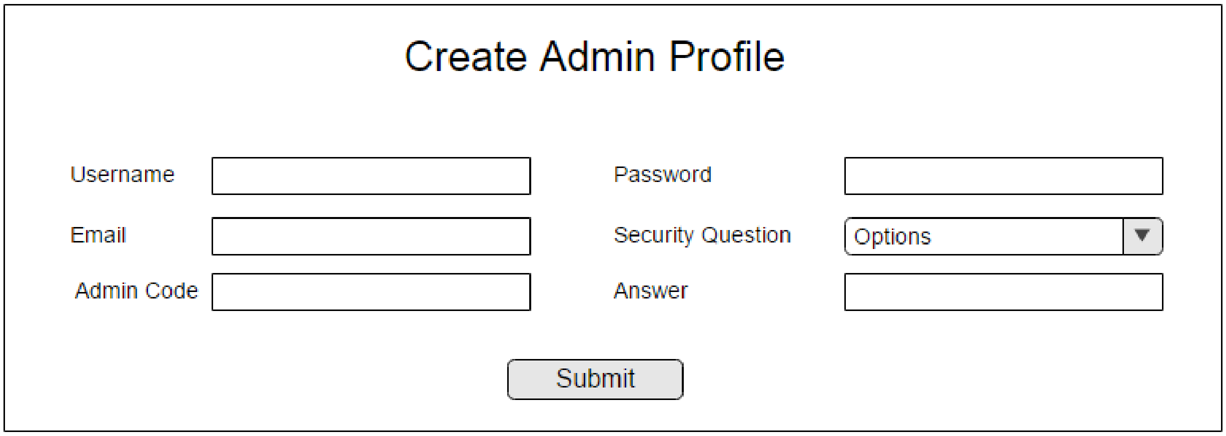
Weight

Age

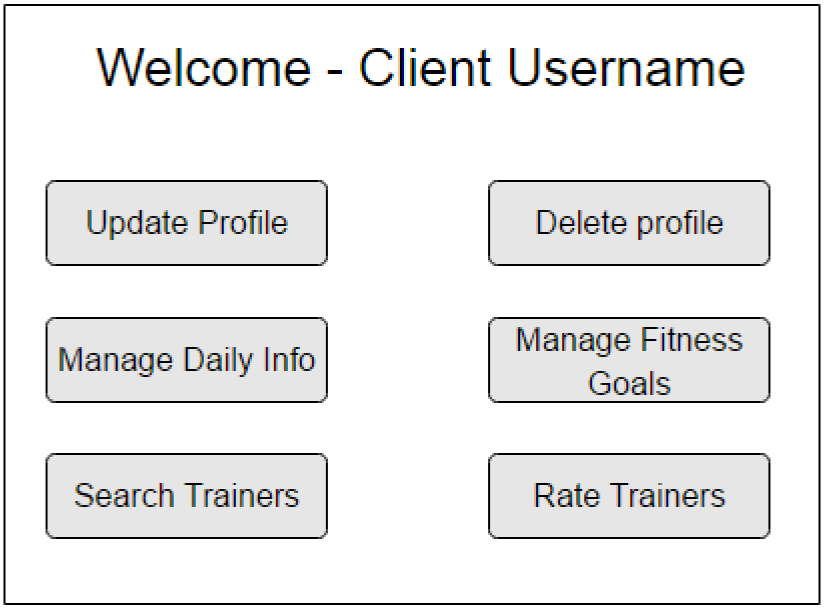
Gender ◉ Male ◉ Female

Submit

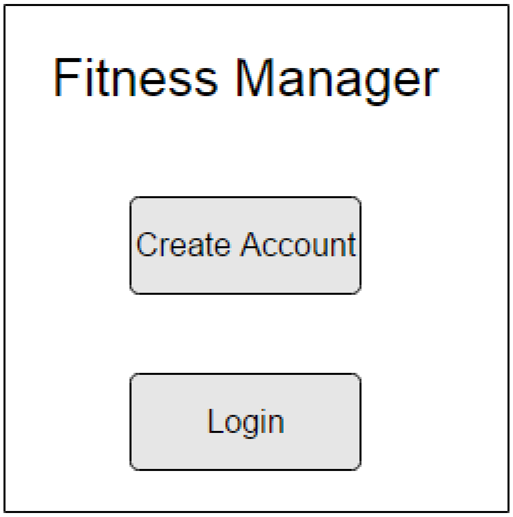
Create Admin Profile



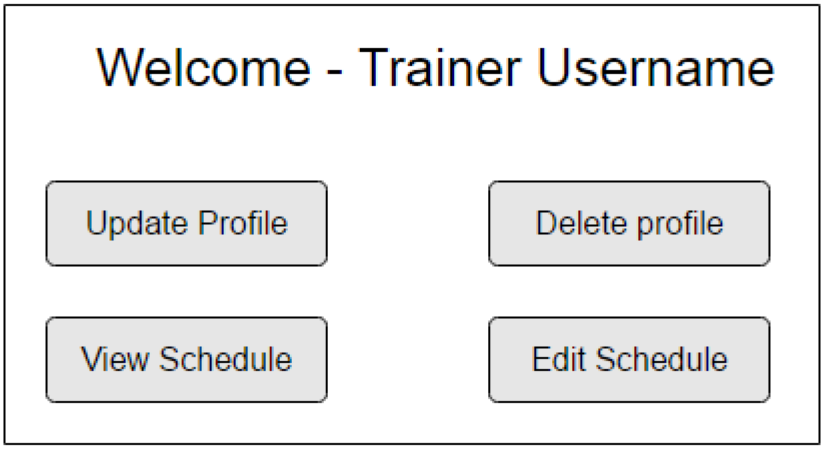
Client Homepage



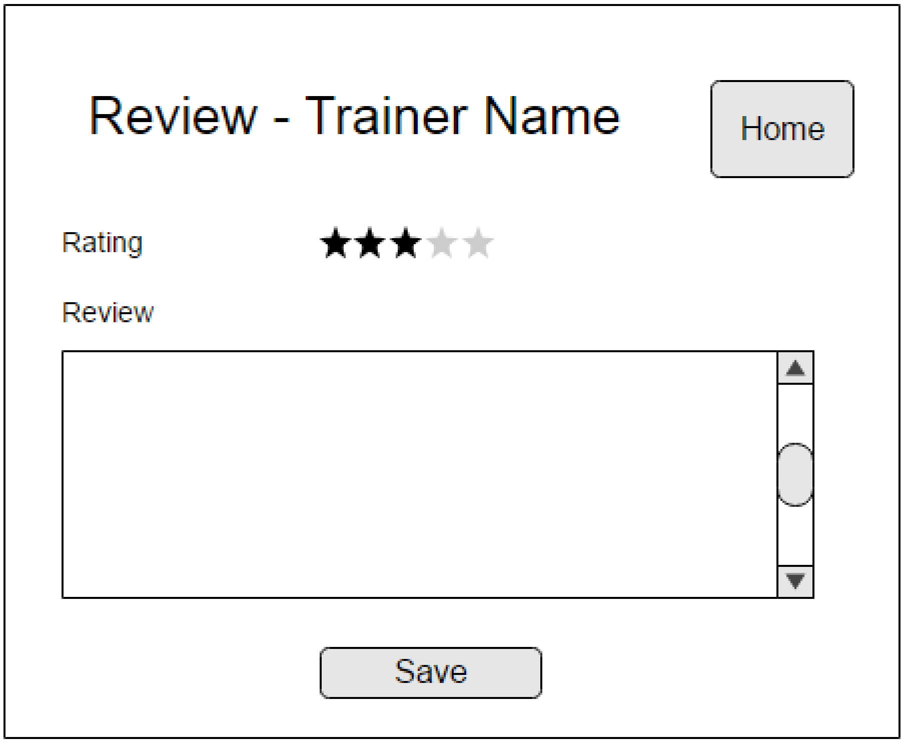
Getting Started



Trainer Homepage



Review Trainer

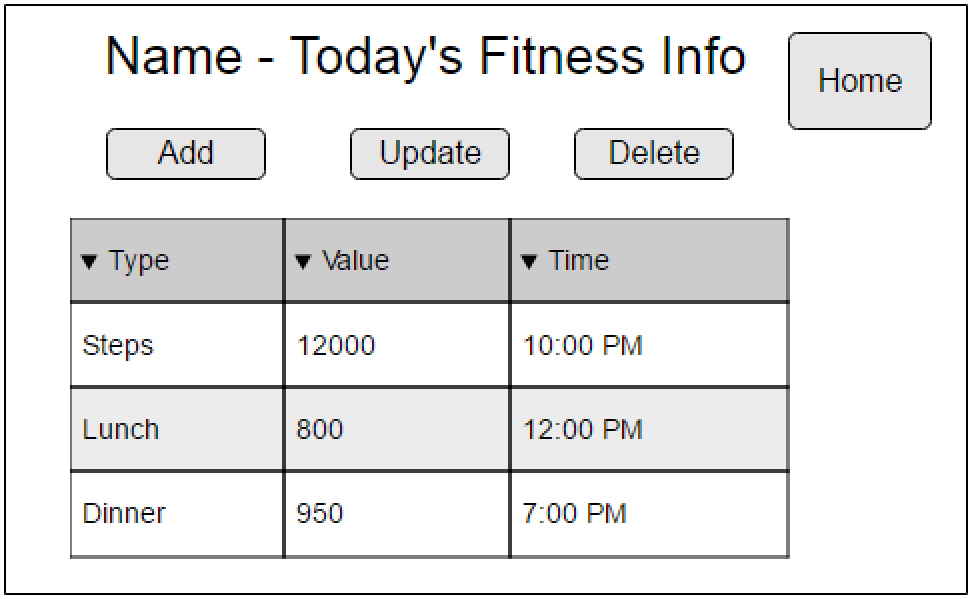


Review - Trainer Name

Rating ★★★☆☆

Review

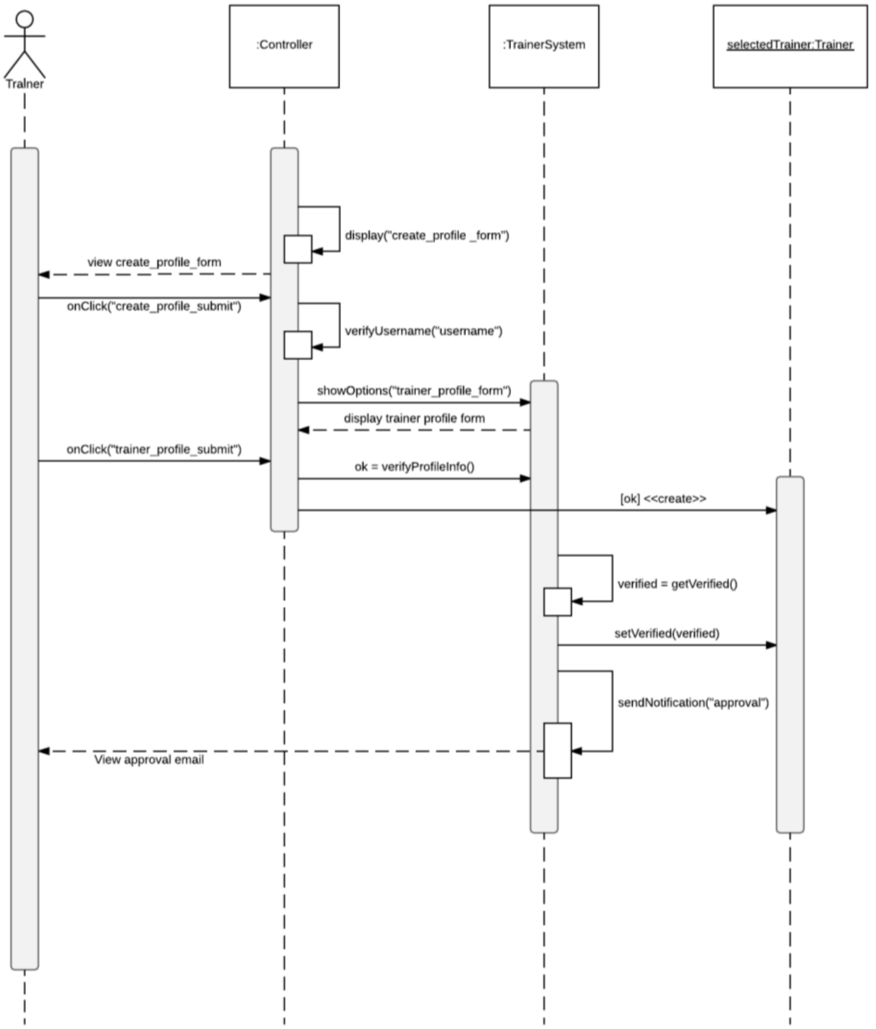
View Fitness Info



Name - Today's Fitness Info

▼ Type	▼ Value	▼ Time
Steps	12000	10:00 PM
Lunch	800	12:00 PM
Dinner	950	7:00 PM

SEQUENCE DIAGRAMS

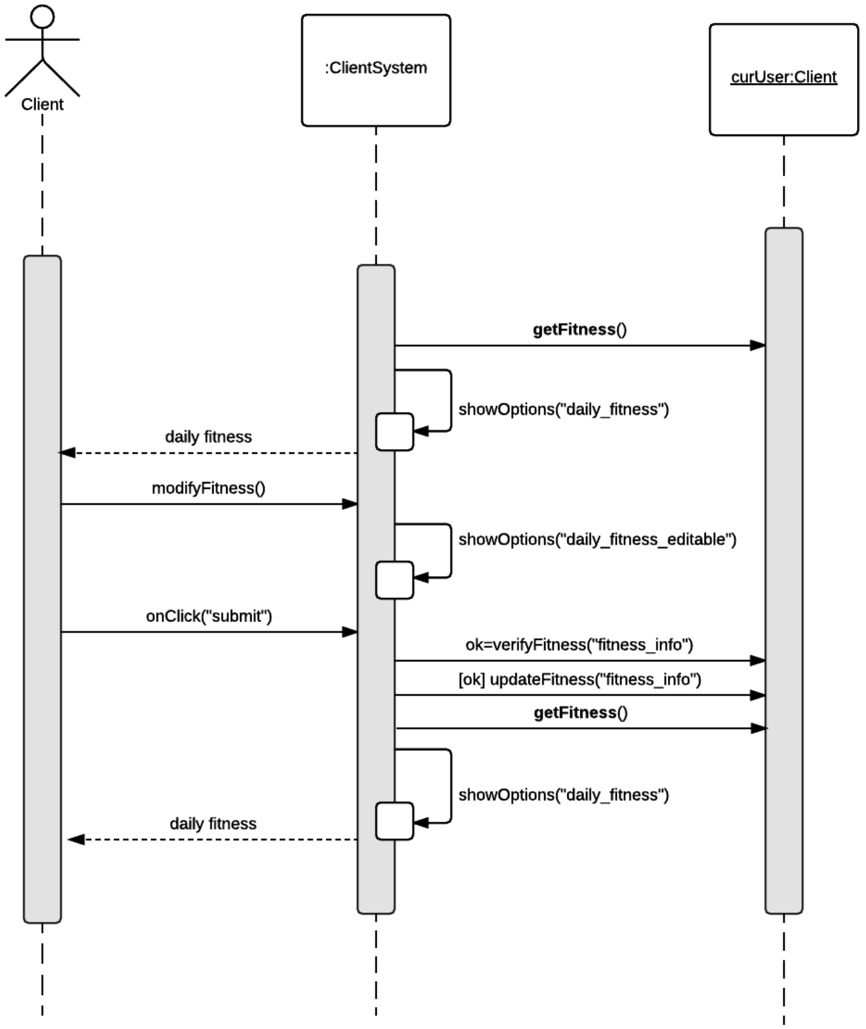


CREATE TRAINER PROFILE SEQUENCE DIAGRAM Chaitanya Soma | October 13, 2016

Trainer
:Controller
:TrainerSystem
selectedTrainer:Trainer
display("create_profile _form")
view create_profile_form
onClick("create_profile_submit")
verifyUsername("username")
showOptions("trainer_profile_form")
display trainer profile form
onClick("trainer_profile_submit")
ok = verifyProfileInfo()
[ok] <<create>>
verified = getVerified()
setVerified(verified)
sendNotification("approval")
View approval email

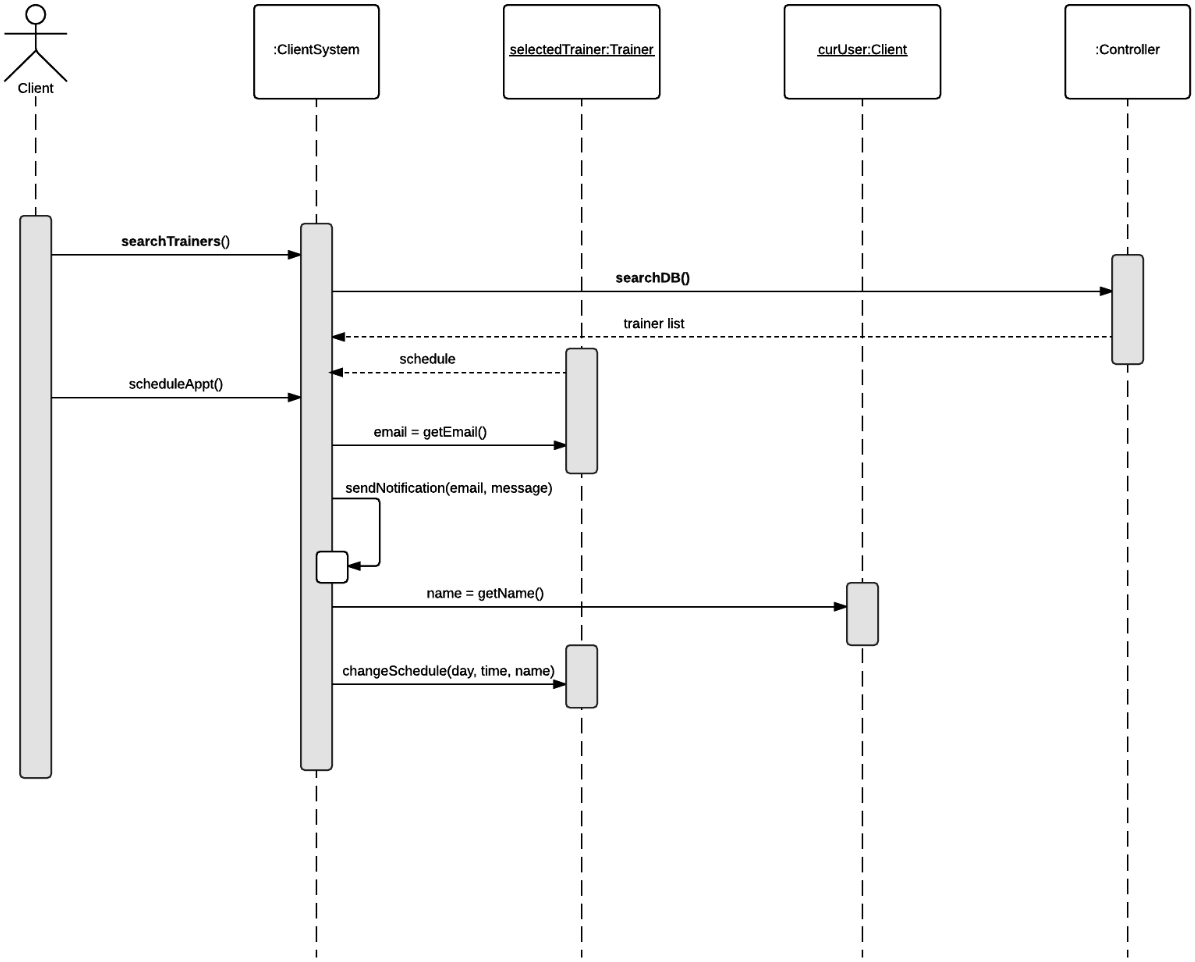
MODIFY DAILY FITNESS SEQUENCE DIAGRAM

Ramnarayan K | October 13, 2016


Client
:ClientSystem
curUser:Client
getFitness()
showOptions("daily_fitness")
daily fitness
modifyFitness()
showOptions("daily_fitness_editable")
onClick("submit")
ok=verifyFitness("fitness_info")
[ok] updateFitness("fitness_info")
getFitness()
showOptions("daily_fitness")
daily fitness

SCHEDULE AN APPOINTMENT WITH A TRAINER

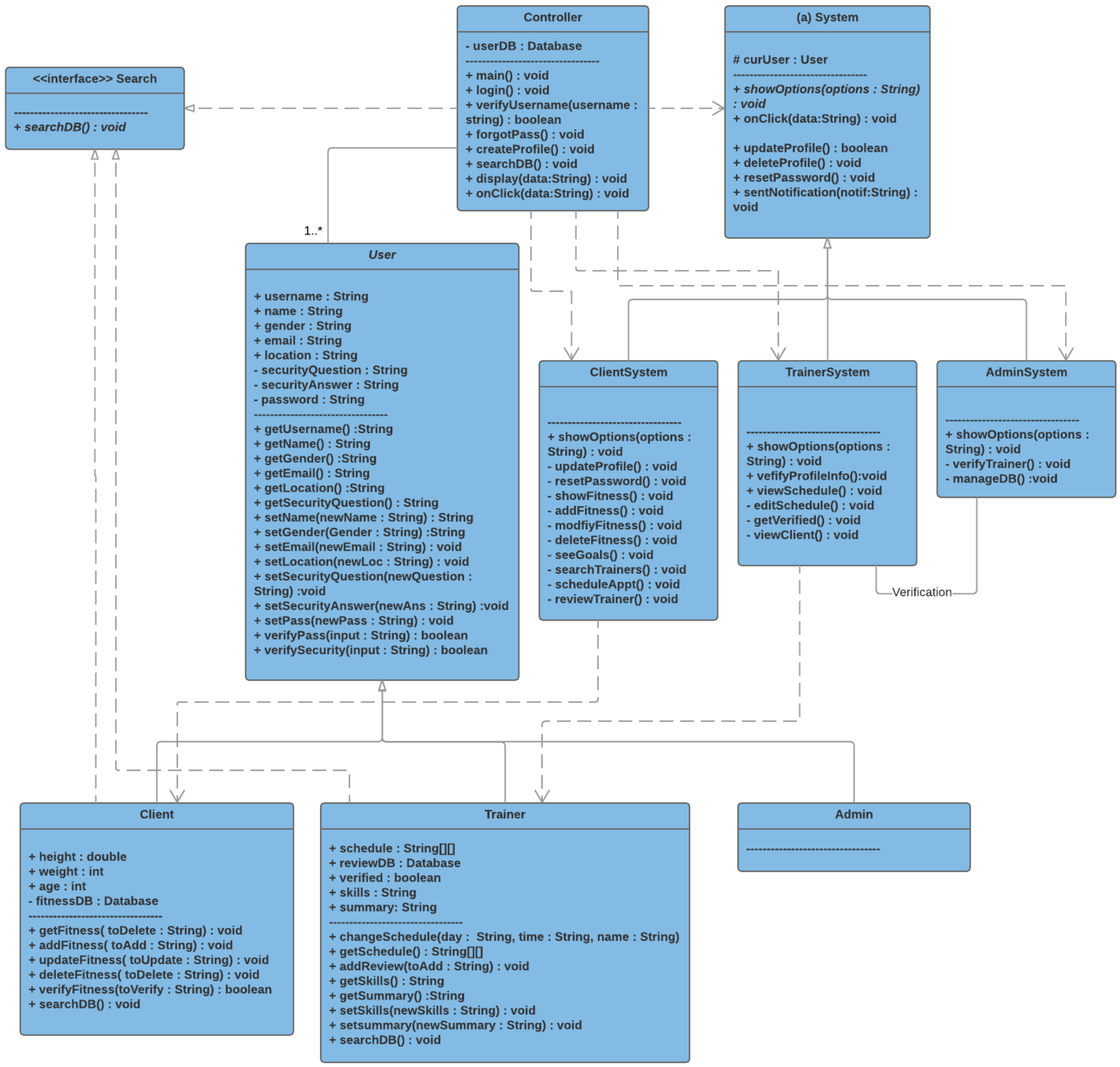
Author: Jordan Peters



CLASS DIAGRAMS

FITNESS MANAGER - CLASS DIAGRAM

chaitanya | October 13, 2016


Controller
- userDB : Database

+ main() : void
+ login() : void
+ verifyUsername(username : string) : boolean
+ forgotPass() : void
+ createProfile() : void
+ searchDB() : void
+ display(data:String) : void
+ onClick(data:String) : void
(a) System
# curUser : User

+ showOptions(options : String) : void
+ onClick(data:String) : void
+ updateProfile() : boolean
+ deleteProfile() : void
+ resetPassword() : void
+ sentNotification(notif:String) : void
<<interface>> Search

+ searchDB() : void
1..*
User
+ username : String
+ name : String
+ gender : String
+ email : String
+ location : String
- securityQuestion : String
- securityAnswer : String
- password : String

+ getUsername() :String
+ getName() : String
+ getGender() :String
+ getEmail() : String
+ getLocation() :String
+ getSecurityQuestion() : String
+ setName(newName : String) : String
+ setGender(Gender : String) :String
+ setEmail(newEmail : String) : void
+ setLocation(newLoc : String) : void
+ setSecurityQuestion(newQuestion : String) :void
+ setSecurityAnswer(newAns : String) :void
+ setPass(newPass : String) : void
+ verifyPass(input : String) : boolean
+ verifySecurity(input : String) : boolean
ClientSystem

+ showOptions(options : String) : void
- updateProfile() : void
- resetPassword() : void
- showFitness() : void
- addFitness() : void
- modfiyFitness() : void
- deleteFitness() : void
- seeGoals() : void
- searchTrainers() : void
- scheduleAppt() : void
- reviewTrainer() : void
TrainerSystem

+ showOptions(options : String) : void
+ vefifyProfileInfo():void
+ viewSchedule() : void
- editSchedule() : void
- getVerified() : void
- viewClient() : void
AdminSystem

+ showOptions(options : String) : void
- verifyTrainer() : void
- manageDB() :void
Verification
Client
+ height : double
+ weight : int
+ age : int
- fitnessDB : Database

+ getFitness(toDelete : String) : void
+ addFitness(toAdd : String) : void
+ updateFitness(toUpdate : String) : void
+ deleteFitness(toDelete : String) : void
+ verifyFitness(toVerify : String) : boolean
+ searchDB() : void
Trainer
+ schedule : String[][]
+ reviewDB : Database
+ verified : boolean
+ skills : String
+ summary: String

+ changeSchedule(day : String, time : String, name : String)
+ getSchedule() : String[][]
+ addReview(toAdd : String) : void
+ getSkills() : String
+ getSummary() :String
+ setSkills(newSkills : String) : void
+ setsummary(newSummary : String) : void
+ searchDB() : void
Admin
