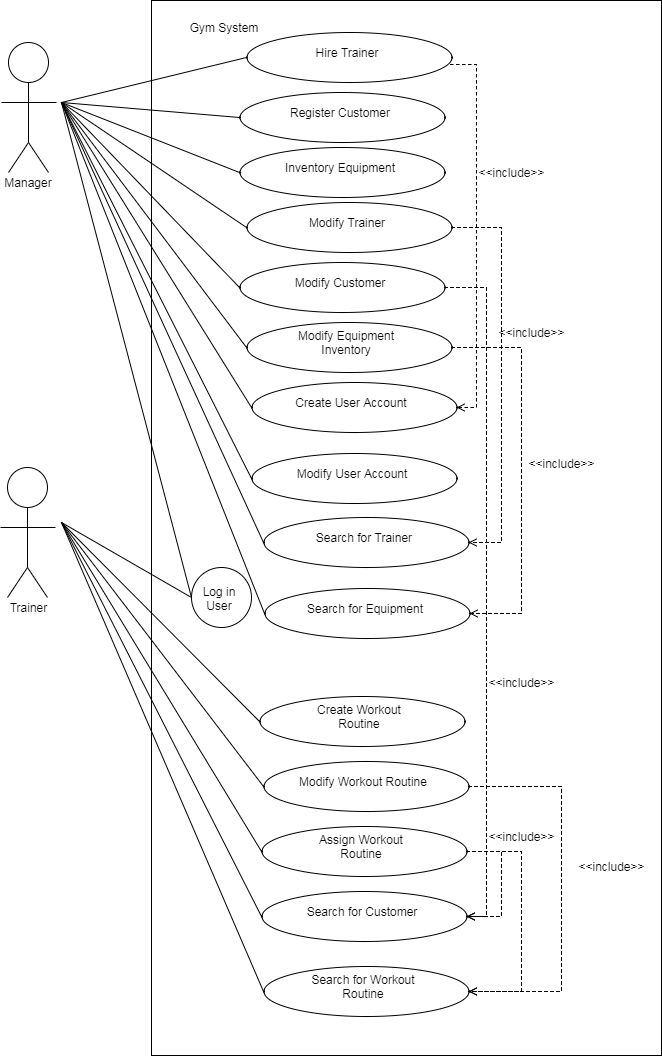
Isaac Trujillo

October 1, 2017

CS414-P1-Use-Cases



**Use Case UC1: Hire Trainer**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Wants to be able to enter all trainer information of a newly hired trainer
* Trainer: Wants his or her information to be entered into the system accurately

**Preconditions:**

* Trainer must not already be in the system
* Manager is authenticated

**Success Guarantee:** Trainer information entered is saved accurately

**Main Success Scenario:**

1. Manager accesses the user interface to enter trainer information
2. Manager enters all trainer data
   1. First name
   2. Last name
   3. Address
   4. Phone
   5. Email
   6. ID
   7. Health Insurance Provider
3. Manager saves information

**Extensions:**

1. Trainer already exists within the system
   1. Manager accesses the user interface to enter trainer information
   2. Manager enters all trainer data
   3. Manager attempts to save information
   4. System notifies the manager that the trainer already exists within the system and the information is not saved
2. At any time, system fails
   1. Manager accesses the user interface to enter trainer information
   2. During the time when a manager is entering trainer information, the system fails. No trainer information will be saved and the manager will have to enter all information again.
3. Action is cancelled
   1. Manager accesses the user interface to enter trainer information
   2. Manager cancels the action prior to saving
   3. No trainer information is saved

**Special Requirements:** Only managers should be allowed to enter trainer information

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC2: Register Customer**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Wants to be able to enter all customer information of a new customer
* Customer: Wants his or her information to be entered into the system accurately

**Preconditions:**

* Customer must not already be in the system
* Manager is authenticated

**Success Guarantee:** Customer information entered is saved accurately

**Main Success Scenario:**

1. Manager accesses the user interface to enter customer information
2. Manager enters all customer data
   1. First name
   2. Last name
   3. Address
   4. Phone
   5. Email
   6. ID
   7. Health Insurance Provier
3. Manager saves information and the customer now has an active membership

**Extensions:**

1. Customer already exists within the system
   1. Manager accesses the user interface to enter customer information
   2. Manager enters all customer data
   3. Manager attempts to save information
   4. System notifies the manager that the customer already exists within the system and the information is not saved
2. At any time, system fails
   1. Manager accesses the user interface to enter customer information
   2. During the time when a manager is entering customer information, the system fails. No customer information will be saved and the manager will have to enter all information again.
3. Action is cancelled
   1. Manager accesses the user interface to enter customer information
   2. Manager cancels the action prior to saving
   3. No customer information is saved

**Special Requirements:** Only managers should be allowed to enter customer information

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC3: Inventory Equipment**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Wants to be able to enter inventory information for gym equipment

**Preconditions:**

* Equipment item must not already be in the system
* Manager is authenticated

**Success Guarantee:** Equipment information entered is saved accurately

**Main Success Scenario:**

1. Manager accesses the user interface to enter equipment inventory information
2. Manager enters all data for the equipment item
   1. Name
   2. Picture
   3. Quantity
3. Manager saves information and the new equipment item is in the system

**Extensions:**

1. Equipment item already exists within the system
   1. Manager accesses the user interface to enter equipment information
   2. Manager enters all equipment item data
   3. Manager attempts to save information
   4. System notifies the manager that the equipment already exists within the system and the information is not saved
2. At any time, system fails
   1. Manager accesses the user interface to enter equipment inventory information
   2. During the time when a manager is entering equipment information, the system fails. No equipment information will be saved and the manager will have to enter all information again.
3. Action is cancelled
   1. Manager accesses the user interface to enter equipment information
   2. Manager cancels the action prior to saving
   3. No equipment information is saved

**Special Requirements:** Only managers should be allowed to enter inventory information

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC4: Modify Trainer**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Wants to be able to modify trainer information
* Trainer: Wants modifications made to his or her information to be saved into the system accurately

**Preconditions:**

* Trainers must exist within the system
* Manager is authenticated

**Success Guarantee:** Modifications made to trainer information is saved accurately

**Main Success Scenario:**

1. Include(Search for Trainer)
2. Manager accesses the user interface to modify trainer information
3. Manager selects trainer to be modified from a list of available trainers
4. Manager makes modifications to trainer information
5. Manager saves modifications

**Extensions:**

1. At any time, system fails
   1. Manager accesses the user interface to make trainer modifications
   2. During the time when a manager is making the modifications, the system fails. No modifications will be saved and the manager will have to enter all information again.
2. Two managers attempt to modify a trainer's information at the same time
   1. Manager A accesses the user interface to modify a trainer's information
   2. Manager B accesses the user interface to modify the same trainer's information
   3. Manager A saves his changes
   4. Manager B saves her changes
   5. Manager B's changes will override Manager A's changes
3. Action is cancelled
   1. Manager accesses the user interface to make trainer modifications
   2. Manager cancels the action prior to saving
   3. No trainer modifications are saved

**Special Requirements:** Only managers should be allowed to modify trainer information

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC5: Modify Customer**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Wants to be able to modify customer information
* Customer: Wants modifications made to his or her information to be saved into the system accurately

**Preconditions:**

* Customers must exist within the system
* Manager is authenticated

**Success Guarantee:** Modifications made to customer information is saved accurately

**Main Success Scenario:**

1. Include(Search for Customer)
2. Manager accesses the user interface to modify customer information
3. Manager selects customer to be modified from a list of available customers
4. Manager makes modifications to customer information
5. Manager saves modifications

**Extensions:**

1. At any time, system fails
   1. Manager accesses the user interface to make customer modifications
   2. During the time when a manager is making the modifications, the system fails. No modifications will be saved and the manager will have to enter all information again.
2. Two managers attempt to modify a customer's information at the same time
   1. Manager A accesses the user interface to modify a customer's information
   2. Manager B accesses the user interface to modify the same customer's information
   3. Manager A saves his changes
   4. Manager B saves her changes
   5. Manager B's changes will override Manager A's changes
3. Action is cancelled
   1. Manager accesses the user interface to make customer modifications
   2. Manager cancels the action prior to saving
   3. No customer modifications are saved

**Special Requirements:** Only managers should be allowed to modify customer information

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC6: Modify Equipment Inventory**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Wants to be able to modify equipment inventory information

**Preconditions:**

* Equipment inventory must exist within the system
* Manager is authenticated

**Success Guarantee:** Modifications made to equipment inventory information is saved accurately

**Main Success Scenario:**

1. Include(Search for Equipment)
2. Manager accesses the user interface to modify equipment inventory information
3. Manager selects equipment inventory to be modified from a list of available equipment inventory
4. Manager makes modifications to equipment inventory information
5. Manager saves modifications

**Extensions:**

1. At any time, system fails
   1. Manager accesses the user interface to make equipment inventory modifications
   2. During the time when a manager is making the modifications, the system fails. No modifications will be saved and the manager will have to enter all information again.
2. Two managers attempt to modify the same equipment inventory information at the same time
   1. Manager A accesses the user interface to modify equipment inventory information
   2. Manager B accesses the user interface to modify the same equipment inventory information
   3. Manager A saves his changes
   4. Manager B saves her changes
   5. Manager B's changes will override Manager A's changes
3. Action is cancelled
   1. Manager accesses the user interface to make equipment inventory modifications
   2. Manager cancels the action prior to saving
   3. No equipment inventory modifications are saved

**Special Requirements:** Only managers should be allowed to modify equipment inventory information

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC7: Create Workout Routine**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**Stakeholders and Interests:**

* Trainer: Wants to be able to create workout routines for his or her customers
* Customers: Want trainers to be able to create routines for them

**Preconditions:**

* Trainer is authenticated

**Success Guarantee:** Trainer is able to create a workout routine and all information entered is saved accurately

**Main Success Scenario:**

1. Trainer accesses the user interface to create a workout routine
2. Trainer fills in all necessary information for the each exercise within the workout routine, selecting available equipment to be used if needed
3. Trainer saves the workout routine and it is now available in the system

**Extensions:**

1. Trainer creates workout routine using pre-existing exercises
   1. Trainer accesses the user interface to create a workout routine
   2. Trainer selects pre-existing exercises from a list to add to the workout routing
   3. Trainer saves the workout routine and it is now available in the system
2. At any time, system fails
   1. Trainer accesses the user interface to create a workout routine
   2. During the time when a trainer is creating the workout routine, the system fails. No information will be saved and the trainer will have to enter all information again.
3. Action is cancelled
   1. Trainer accesses the user interface to create a workout routine
   2. Trainer cancels the action prior to saving
   3. No information for the workout routine is saved

**Special Requirements:** Only trainers should be allowed to create workout routines

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC8: Modify Workout Routine**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**Stakeholders and Interests:**

* Trainer: Wants to be able to modify existing workout routines for his or her customers
* Customers: Want trainers to be able to make needed modifications to workout routines

**Preconditions:**

* Trainer is authenticated

**Success Guarantee:** Trainer is able to modify an existing workout routine and all information entered is saved accurately

**Main Success Scenario:**

1. Include (Search for Workout Routine)
2. Trainer accesses the user interface to modify a workout routine
3. Trainer selects the workout routine to be modified from a list of existing workout routines
4. Trainer makes all modifications
5. Trainer saves the workout routine and the routine is updated accurately

**Extensions:**

1. At any time, system fails
   1. Trainer accesses the user interface to modify a workout routine
   2. During the time when a trainer is modifying the workout routine, the system fails. No information will be saved and the trainer will have to make all modifications again.
2. Two trainers attempt to modify a workout routine at the same time
   1. Trainer A accesses the user interface to modify a workout routine
   2. Trainer B accesses the user interface to modify the same workout routine
   3. Trainer A saves his changes
   4. Trainer B saves her changes
   5. Trainer B's changes will override Trainer A's changes
3. Action is cancelled
   1. Trainer accesses the user interface to modify a workout routine
   2. Trainer cancels the action prior to saving
   3. No modifications for the workout routine are saved

**Special Requirements:** Only trainers should be allowed to modify workout routines

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC9: Assign Workout Routine**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**Stakeholders and Interests:**

* Trainer: Wants to be able to modify assign workout routines to his or her customers
* Customers: Want trainers to be able to assign them workout routines

**Preconditions:**

* Trainer is authenticated
* At least one workout routine must exist
* At least one customer must exist

**Success Guarantee:** Trainer is able to assign an existing workout routine to a customer and the assignment is saved accurately

**Main Success Scenario:**

1. Include (Search for Workout Routine)
2. Include (Search for Customer)
3. Trainer accesses the user interface to assign a workout routine
4. Trainer selects one or more workout routines to be assigned from a list of existing workout routines
5. Trainer selects the customer to which the routines will be assigned
6. Trainer saves the assignment

**Extensions:**

1. At any time, system fails
   1. Trainer accesses the user interface to assign a workout routine to a customer
   2. During the time when a trainer is making the assignment, the system fails. No assignments will be saved and the trainer will have to make all assignments again
2. Customer already has the routine assigned
   1. Trainer attempts to assign a routine to a customer that already has the routine assigned
   2. When the trainer attempts to save the assignment, an error will be displayed and the assignment will not be saved
3. Action is cancelled
   1. Trainer accesses the user interface to assign a workout routine
   2. Trainer cancels the action prior to saving
   3. No assignment is saved

**Special Requirements:** Only trainers should be allowed to assign workout routines to customers

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC10: Search for Customer**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer, Manager

**Stakeholders and Interests:**

* Trainer: Wants to be able to search for a customer within the system to determine which workout routines are currently assigned to the customer
* Manger: Wants to be able to search for a customer within the system to modify customer data at any time

**Preconditions:**

* Manager / Trainer is authenticated

**Success Guarantee:** Manager / Trainer searches for a customer within the system. The customer is found and is displayed along with all workout routines assigned to the customer

**Main Success Scenario:**

1. Manager / Trainer accesses the search user interface to search for a customer
2. Manager / Trainer searches for customer A
3. Customer A appears showing all routines that are currently assigned

**Extensions:**

1. At any time, system fails
   1. Manager / Trainer accesses the user interface to search for a customer
   2. Manager / Trainer performs the search and results are displayed.
   3. During the time the system fails, search results are cleared and Manager / Trainer has to perform the search again
2. Manager / Trainer searches for non-existing customer
   1. Manager / Trainer searches for non-existing customer
   2. No results are listed in the search results list

**Special Requirements:** Only Manager / Trainer are allowed to search for customers

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC11: Search for Workout Routine**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**Stakeholders and Interests:**

* Trainer: Wants to be able to search for a workout routine within the system to determine which customers are currently assigned

**Preconditions:**

* Trainer is authenticated

**Success Guarantee:** Trainer searches for a workout routine within the system. The workout routine is found and is displayed along with all customers that are currently assigned

**Main Success Scenario:**

1. Trainer accesses the search user interface to search for a workout routine
2. Trainer searches for workout routine A
3. Workout routine A appears showing all customer that are currently assigned

**Extensions:**

1. At any time, system fails
   1. Trainer accesses the user interface to search for a workout routine
   2. Trainer performs the search and results are displayed.
   3. During the time the system fails, search results are cleared and trainer has to perform the search again
2. Trainer searches for non-existing workout routine
   1. Trainer searches for non-existing workout routine
   2. No results are listed in the search results list

**Special Requirements:** Only trainers are allowed to search for workout routines

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC12: Log in User**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager, Trainer

**Stakeholders and Interests:**

* Trainer: Needs to be able to log in to the system in order to perform trainer duties
* Manager: Needs to be able to log in to the system in order to perform manager duties

**Preconditions:**

* User must have an account within the system

**Success Guarantee:** User enters a user name and password, once verified as correct, user is allowed access to the system in the role to which the user assigned

**Main Success Scenario:**

1. User accesses the login user interface
2. User enters a username and password
3. Username and password are verified and the user is allowed access to the system

**Extensions:**

1. System fails after login
   1. User accesses the user interface to log in
   2. User enters credentials and is authenticated and granted access
   3. The system fails, user is logged out and is required to log in again
2. User supplies invalid username and password combination
   1. User accesses the user interface to log in
   2. User enter invalid username and password combination
   3. Notification is displayed saying login attempt was not valid and the user is not granted access to the system

**Special Requirements:** None

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC13: Create User Account**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Needs to be able to create user accounts within the system

**Preconditions:**

* User must not yet have an account within the system
* User must have either a manager or trainer entity within the system

**Success Guarantee:** Manager enters all user information and saves. The account is created within the system

**Main Success Scenario:**

1. Manager accesses the account creation user interface
2. Manager enters all user information and saves
   1. Username
   2. Password
3. User account is then available for use within the system

**Extensions:**

1. System fails during account creation
   1. Manager accesses the user interface to create an account
   2. The system fails before manager can save account, all information is lost and the Manager is required to enter all information again
2. User account already exists
   1. Manager accesses the user interface to create an account
   2. Manager enters all information and attempts to save the account
   3. A notification is displayed stating the account already exists and no information is saved.
3. Action is cancelled
   1. Manager accesses the user interface to create a user account
   2. Manager cancels the action prior to saving
   3. No user account is created

**Special Requirements:** Only Managers have the ability to create user accounts

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC14: Modify User Account**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Manager

**Stakeholders and Interests:**

* Manager: Needs to be able to modify user accounts within the system

**Preconditions:**

* User account must exist within the system

**Success Guarantee:** Manager modifies a user's account and all changes are accurately saved

**Main Success Scenario:**

1. Manager accesses the account modification user interface
2. Manager selects user account to be modified
3. Manager makes all necessary modifications and saves
4. All updated data is accurately saved

**Extensions:**

1. System fails during account modification
   1. Manager accesses the user interface to create an account
   2. Manager selects user account to be modified
   3. The system fails before manager can save the modifications, all information is lost and the Manager is required to make the modifications again

**Special Requirements:** Only Managers have the ability to modify user accounts

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC15: Search for Trainer**

**Scope:** Gym System

**Level:** user goal

**Primary Actor: Manager**

**Stakeholders and Interests:**

* Manager: Wants to be able to search for a trainer within the system to modify

**Preconditions:**

* Manager is authenticated

**Success Guarantee:** Manager searches for a trainer within the system. The trainer is found and is displayed

**Main Success Scenario:**

1. Manager accesses the search trainer interface to search for a trainer
2. Manager searches for trainer A
3. Trainer A appears showing all information

**Extensions:**

1. At any time, system fails
   1. Manager accesses the user interface to search for a trainer
   2. Manager performs the search and results are displayed.
   3. During the time the system fails, search results are cleared and manager has to perform the search again
2. Manager searches for non-existing trainer
   1. Manager searches for non-existing trainer
   2. No results are listed in the search results list

**Special Requirements:** Only Managers are allowed to search for trainers

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC16: Search for Equipment**

**Scope:** Gym System

**Level:** user goal

**Primary Actor: Manager**

**Stakeholders and Interests:**

* Manager: Wants to be able to search for a equipment within the system to modify

**Preconditions:**

* Manager is authenticated

**Success Guarantee:** Manager searches for equipment within the system. The equipment is found and is displayed

**Main Success Scenario:**

1. Manager accesses the search equipment interface to search for equipment
2. Manager searches for equipment A
3. Equipment A appears showing all information

**Extensions:**

1. At any time, system fails
   1. Manager accesses the user interface to search for equipment
   2. Manager performs the search and results are displayed.
   3. During the time the system fails, search results are cleared and manager has to perform the search again
2. Manager searches for non-existing equipment
   1. Manager searches for non-existing equipment
   2. No results are listed in the search results list

**Special Requirements:** Only Managers are allowed to search for equipment

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC17: Create Fitness Class**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**Stakeholders and Interests:**

* Trainer: Wants to be able to create fitness classes for his or her customers
* Customers: Want trainers to be able to create fitness classes for them

**Preconditions:**

* Trainer is authenticated

**Success Guarantee:** Trainer is able to create a fitness class and all information entered is saved accurately

**Main Success Scenario:**

1. Trainer accesses the user interface to create a fitness class
2. Trainer fills in all necessary information for the each exercise within the fitness class, assigning trainers and customers
3. Trainer saves the fitness class and it is now available in the system

**Extensions:**

1. At any time, system fails
   1. Trainer accesses the user interface to create a fitness class
   2. During the time when a trainer is creating the fitness class, the system fails. No information will be saved and the trainer will have to enter all information again.
2. Action is cancelled
   1. Trainer accesses the user interface to create a fitness class
   2. Trainer cancels the action prior to saving
   3. No information for the fitness class is saved

**Special Requirements:** Only trainers should be allowed to create fitness classes

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None

**Use Case UC18: Modify Fitness Class**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**Stakeholders and Interests:**

* Trainer: Wants to be able to modify fitness classes
* Customers: Want trainers to be able to modify fitness classes

**Preconditions:**

* Trainer is authenticated

**Success Guarantee:** Trainer is able to select and modify a fitness class and all information entered is saved accurately

**Main Success Scenario:**

1. Trainer accesses the user interface to select a fitness class
2. Trainer selects the fitness class and all class data is loaded to the UI.
3. Trainer modifies all necessary information
4. Trainer saves the fitness class and it is now available in the system

**Extensions:**

1. At any time, system fails
   1. Trainer accesses the user interface to modify a fitness class
   2. During the time when a trainer is modifying the fitness class, the system fails. No information will be saved and the trainer will have to enter all information again.
2. Action is cancelled
   1. Trainer accesses the user interface to modify a fitness class
   2. Trainer cancels the action prior to saving
   3. No modifications for the fitness class are saved

**Special Requirements:** Only trainers should be allowed to modify fitness classes

**Technology and Data Verifications List:**

* Java must be used
* Project must be on GitHub

**Frequency of Occurrence:** Could be continuous and at any time

**Miscellaneous:** None