**Isaac Trujillo**

**September 17, 2017**

**CS414-P1**

**Use Case UC1: Hiring of Trainers**

**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
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.

**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: Registration of New Customers**

**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
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.

**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 of 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
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.

**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: Modification of Trainers**

**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. Manager accesses the user interface to modify trainer information
2. Manager selects trainer to be modified from a list of available trainers
3. Manager makes modifications to trainer information
4. 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

**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: Modification of Customers**

**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. Manager accesses the user interface to modify customer information
2. Manager selects customer to be modified from a list of available customers
3. Manager makes modifications to customer information
4. 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

**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: Modification of 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. Manager accesses the user interface to modify equipment inventory information
2. Manager selects equipment inventory to be modified from a list of available equipment inventory
3. Manager makes modifications to equipment inventory information
4. 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

**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: Creation of Workout Routines**

**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. 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.

**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: Modification of Workout Routines**

**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. Trainer accesses the user interface to modify a workout routine
2. Trainer selects the workout routine to be modified from a list of existing workout routines
3. Trainer makes all modifications
4. 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

**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: Workout Routine Assignment**

**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. Trainer accesses the user interface to assign a workout routine
2. Trainer selects one or more workout routines to be assigned from a list of existing workout routines
3. Trainer selects the customer to which the routines will be assigned
4. 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

**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: Customer Search**

**Scope:** Gym System

**Level:** user goal

**Primary Actor:** Trainer

**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

**Preconditions:**

* Trainer is authenticated

**Success Guarantee:** 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. Trainer accesses the search user interface to search for a customer
2. Trainer searches for customer A
3. Customer A appears showing all routines that are currently assigned

**Extensions:**

1. At any time, system fails
   1. Trainer accesses the user interface to search for a customer
   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 customer
   1. Trainer searches for non-existing customer
   2. Message is displayed in the search results stating that the customer was not found

**Special Requirements:** Only trainers 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: Routine Search**

**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. Message is displayed in the search results stating that the workout routine was not found

**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: User Login**

**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: User Account Creation**

**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

**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, including user role, and saves
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.

**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: User Account Modification**

**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

**Overview Diagram**

