Table 5.1-User_Information Class Description Form

Name	*User_Information*
Base Class(es)	
Purpose	*A User Information object contains all the user information that can be listed, kept, used by other modules or discarded.*
States	*Empty, full, or neither* *Inactive or Active*
Constructors	*Default: contains user information*
Operators **Mutators**	*setWorkout(workouts:Workouts)* *updateProfile(dob:String,height:Integer,weight:Integer)* *setPassword(newPassword: String)* *addBalance(balanceToAdd:Integer)* *addWorkout(newWorkout:Workout)* *setWorkouts(workouts:workout)* *validateWorkout(User_information,Workout, Activity)*
Accessors	*getBalance():Integer* *getUserBMI():Integer* *getValidatableWorkout():Bool*
Fields	*userID:String* *userDOB:integer* *userHeight: Integer* *userWeight: Integer* *password:String* *workouts:Integer* *usersCanValidate: List*

Table 6.1 – User_Information::getBalance

Prototype	*Integer:User_Information::getBalance ()*
Purpose	*Returns the balance of the current user*
Receives	*Will user information in the database to retrieve recurring user balance.*
Returns	*INTEGER – Returns the balance of the current user requested by the accessor's functions.*
Remarks	*The operation may fail if there's not balance available for the current user.*

Table 6.2 – User_Information::updateProfile

Prototype	*Bool User_Information::updateProfile (dob:String,height:Integer,weight:Integer)*
Purpose	*Update the user information fields for a user*
Receives	*dob- contains the date of birth of the user* *height- the height of the current user* *weight- contains the weight of the current user*
Returns	*TRUE if the operation succeeded* *FALSE if the data update information contains invalid values*
Remarks	*The operation may fail if the DOB,height or weight are invalid values for the profile.*

Table 6.3 – User_Information::setPassword

Prototype	*Bool User_Information::setPassword(newPassword)*
Purpose	*Add a new password for the current user*
Receives	*password-the password for the current user security settings*
Returns	*TRUE if the operation succeeded*
Remarks	*The operation may fail if the password is not valid (the password must follow security the minnimum ssecurity requirements).*

Table 6.4 – User_Information::addBalance

Prototype	*Integer User_Information::addBalance(balanceToAdd:Integer)*
Purpose	*Update the balance for the current user.*
Receives	*balanceToAdd- Contains the balance amount to be assigned for the current user.*
Returns	*Integer- user balance if the operation succeeded*
Remarks	*The operation may fail if the balance is not valid a valid amount (the balance must follow system policy).*

Table 6.4 – User_Information::addWorkout

Prototype	*Bool User_Information::addWorkout(newWorkout: Workout)*
Purpose	*Creates and adds the Workout routine for the current user.*
Receives	*newWorkout- Contains the workout profile for the user.*
Returns	*TRUE if the operation succeeded* *FALSE if the workout routine is not valid*
Remarks	*The operation may fail if the workout routine is not a (the routine must follow system policy).*

Table 6.5 – User_Information::getWorkout

Prototype	*Bool User_Information::setWorkout(workouts:Integer)*
Purpose	*Assigns the workout routine for the current user.*
Receives	*workout- Contains a list of all the workouts balance to be assigned to the current user.*
Returns	*TRUE if the operation succeeded* *FALSE if the amount or workout values are not valid*
Remarks	*The operation may fail if the workout information doesn't contains a valid numerical values (the workout must contain valid data).*

Table 6.6 – User_Information::getUserBMI

Prototype	*double User_Information::getUserBMI()*
Purpose	*Returns the calculation of the Body/Mass Index of the current user.*
Receives	*Will utilize user information from the database to calculate the current BMI index.*
Returns	*DOUBLE if the operation succeeded* *NULL if the user information values required for the calculation are not available.*
Remarks	*The operation may fail if the database doesn't have a valid numerical values for the calculation.*

Table 6.7 – User_Information::validateWorkout

Prototype	*Bool:User_Information::validateWorkout(User_information,Workout,Activity)*
Purpose	*Validates the current user Workout routine base on the user workout profile and validator input*
Receives	*User Workout – Contains the user activity profile* *Software routine will verify if the user completed the Workout or not.*
Returns	*TRUE if the operation succeeded* *FALSE if the user information values required for the validation are not completed.*
Remarks	*Validates the workouts/activities of the user*

Table 6.8 – User_Information::getValidatableWorkouts

Prototype	*Workout User_Information::getValidatableWorkouts(User_information)*
Purpose	*Retrieves the list of workout routines that can be validated by the current user.*
Receives	*Workout Routine List*
Returns	*List of Workout routines that can be validated by the current user.*
Remarks	*Contains a list of workout objects of that can be validated*

Table 6.9 – User_Information::createWorkout

Prototype	*User_Information::createWorkout(User_information,Workout: Integer,Activitiy : Integer)*
Purpose	*Creates the workout profile for the current user*
Receives	*User_Information:Workout – the user activities/workout profile input*
Returns	*TRUE if the operation succeeded* *FALSE if the workout profile information values required for the validation are not completed or invalid.*
Remarks	*Creates the workout profile which will be store in the data base.*