Class Models

FH Mobile Application

Version 1.0

Prepared by

Omar Rivera Andrew Poirier Daven Amin Rick Rejeleene

**Table 1.1 Client Application**

|  |  |
| --- | --- |
| **Name** | *Client\_Application* |
| **Base Class** |  |
| **Purpose** | *The Purpose of the Client Application is it*  *helps the user login into the application.* |
| **States** | *Empty, full, or neither.*  *Inactive or Active* |
| **Constructors** | *Default* |
| **Operators Mutators**  **Accessors** | login()  resetPassword() refreshBalancesFromServer() updateProfile()  register()  getHelp() |
| **Fields** | *User\_information* |

**Table 2.1 – Client\_Application::login**

|  |  |
| --- | --- |
| **Prototype** | Client\_Application::login(User\_Information) |
| **Purpose** | *Helps the user login into the client application* |
| **Receives** | *The Login receives a string from the user.* |
| **Returns** | *The Login returns True if the login Id is true or False if the login Id is false.* |
| **Remarks** | *If Login ID matches it logins to the system.* |

**Table 2.2 – Client\_Application::resetPassword**

|  |  |
| --- | --- |
| **Prototype** | Client\_Application::resetPassword(user:User\_Information) |
| **Purpose** | *The purpose of the reset password is if the user forgets the password, he can reset it.* |
| **Receives** | *The reset password receives the string from the user, in this the userid is the string.* |
| **Returns** | *True if the userid matches*  *False if the userid doesn’t match.* |
| **Remarks** | *Helps to reset the password* |

**Table 2.3 – Client\_Application::register**

|  |  |
| --- | --- |
| **Prototype** | Client\_Application::register(user:User\_Information) |
| **Purpose** | *The purpose of Register is for a new user to register into the FH mobile App.* |
| **Receives** | *The Register receives the user name, password which is a string* |
| **Returns** | *void* |
| **Remarks** | *Helps to register for a new user.* |

**Table 2.4 – Client\_Application::refreshBalancesFromServer**

|  |  |
| --- | --- |
| **Prototype** | Client\_Application::refreshBalancesFromServer() |
| **Purpose** | *The purpose of the Refresh balance is it updates the balance for the user* |
| **Receives** | *The Refresh Balance receives Integers from the Server.* |
| **Returns** | *It returns Balance as Integers.* |
| **Remarks** | *The Refresh Balance helps to check the balance for the user.* |

**Table 2.5 – Client\_Application::getHelp**

|  |  |
| --- | --- |
| **Prototype** | Client\_Application::getHelp() |
| **Purpose** | *The Purpose of the Get Help is it brings up the instructions on how to use the application.* |
| **Receives** | *void* |
| **Returns** | *text* |
| **Remarks** | *Brings up the instructions and how to use the app.* |

**Table 2.6 – Client\_Application::updateProfile**

|  |  |
| --- | --- |
| **Prototype** | Cilent\_Application::updateProfile  *(User\_Information)* |
| **Purpose** | *If the User wants to change his information, he can click the update User information.* |
| **Receives** | *The Update user information receives the input as string* |
| **Returns** | *True if the update is successful*  *False if the update is successful.* |
| **Remarks** | *Helps to modify an already created User Information.* |

**Table 3.1 – Server\_Application**

|  |  |
| --- | --- |
| **Name** | *Server* |
| **Base Class(es**) |  |
| **Purpose** | *The server is used to connect to the database and store,delete or update information for the user, workouts, and bank in the application.* |
| **States** | *Change/Update, Validate, Create, Delete*  *Active or Inactive* |
| **Constructors** | *Default: makes server ready for data information and processing* |
| **Operators Mutators**  **Accessors** | *validateWorkout(user:user\_Information, workout:Workout, activity) updateBalance(balance:Integer) createOrUpdateUser(user:User\_Information) deleteUser(user:User\_Information) checkForMissedActivities() changePassword(User\_Session, :string) requestUsersInformation() addWorkout(user:User\_Information)*  *connectToDatabase(location: String,db:Credentials) connectClient(Credentials) getWorkoutsForUser(user:User\_Information) getBalanceForUser(user:User\_Information) getUsersCanValidate(user:User\_Information)* |
| **Fields** | *User\_Session* |

**Table 4.1 – Server\_Application:connectClient**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:connectClient(client:Credentials*  *)* |
| **Purpose** | *To connect the client to a user session in the server* |
| **Receives** | *The clients credentials to authenticate the server* |
| **Returns** | *TRUE if the credentials were valid*  *FALSE if the credentials were invalid* |
| **Remarks** | *The operation may fail if the user forgot to log out last time.* |

**Table 4.2 – Server\_Application:connectToDatabase**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:connectToDatabase(location:Str ing,db:Credentials)* |
| **Purpose** | *To connect the user to the database to access the user's saved information* |
| **Receives** | *The database location and database name as a string object*  *The database credentials from the server* |
| **Returns** | *TRUE if the user and server connect to the correct database with the correct credentials*  *FALSE if the user and server do not connect correctly or if the the credentials are invalid* |
| **Remarks** | *The operation may fail if the database cannot be found or the connection is interrupted at anytime with the current connection* |

**Table 4.3 – Server\_Application:validateWorkout**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:validateWorkout(user:User\_Info rmation, workout:Workout, workout:Activity)* |
| **Purpose** | *To connect the user to the database to access the user's workout history and information* |
| **Receives** | *The server receives the user's information along with the workout and the activity associated with that workout.* |
| **Returns** | *TRUE if the facilitator was able to validate the workout correctly*  *FALSE if the facilitator was unable to find the saved workout or if there are no workouts to validate* |
| **Remarks** | *The operation might fail if the user forgot to save the work out or if the workout does not have any activities associated with it.* |

**Table 4.4 – Server\_Application:getWorkoutsForUser**

|  |  |
| --- | --- |
| **Prototype** | *WORKOUTS:Server\_Application:getWorkoutsForUser(us er:userInformation)* |
| **Purpose** | *This will connect the server to the databse to receive the user's list of workouts from the database.* |
| **Receives** | *The server receives the user's information to find the list of workouts.* |
| **Returns** | *WORKOUTS if the the list of workouts for the user is found in the database*  *NULL if the user does not have any current workouts* |
| **Remarks** | *This operation might fail if the user is not working out for a few weeks or if it the first time and there are no workouts created.* |

**Table 4.5 – Server\_Application:getBalanceForUser**

|  |  |
| --- | --- |
| **Prototype** | *Int:Server\_Application:getBalanceForUser(user:User\_Inf ormation)* |
| **Purpose** | *This will connect the server to the bank database to access the user's current balance.* |
| **Receives** | *The server receives the user's information to find the user in the bank database* |
| **Returns** | *INTEGER if the user's balance was found in the bank database*  *NULL if the user currently does not have any funds available.* |
| **Remarks** | *This operation might fail if the user has not set up bank information with a starting balance.* |

**Table 4.6 – Server\_Application:updateBalance**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:updateBalance(balance:Integer)* |
| **Purpose** | *This will connect to the server and will update the users balance off of the current balance that was received* |
| **Receives** | *The server receives the current balance for the user connected* |
| **Returns** | *TRUE if the balance was updated successfully*  *FALSE if the balance was not updated successfully* |
| **Remarks** | *This operation might fail if the user does not have a balance or the user does not have a penalty/reward amount set in the system* |

**Table 4.7 – Server\_Application:createOrUpdateUser**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:createOrUpdateUser(user:User*  *\_Information)* |
| **Purpose** | *This will allow the user to update or create an account on the server for the application.* |
| **Receives** | *The server recieves the current user's information if they are updating. And it will receive new user information if a user is creating an account* |
| **Returns** | *TRUE if the user was created correctly or the information was updated correctly*  *FALSE if the user did not supply a required field when creating an account or the user does not update information with the correct information designated for that field* |
| **Remarks** | *This operation might not work if a user tries to create a user account with the same name that is already on the server.* |

**Table 4.8 – Server\_Application:deleteUser**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:deleteUser(user:User\_Informati on)* |
| **Purpose** | *This will allow the user to remove their account and information from the server* |
| **Receives** | *The user's information that is store on the server* |
| **Returns** | *TRUE if the user was deleted successfully*  *FALSE If the user was not deleted successfully.* |
| **Remarks** | *The operation will fail if the user tries to delete a user that does not exist on the server.* |

**Table 4.9 – Server\_Application:checkForMissedActivities**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:checkForMissedActivities()* |
| **Purpose** | *This will allow the user to check to see if they missed activities while performing their workout.* |
| **Receives** | *The server will receive the users information to check for missed activities.* |
| **Returns** | *TRUE if the user has missed any activities*  *FALSE if the user has completed all of their activities for the workout* |
| **Remarks** | *The operation will fail if the user does not have any activities listed under a workout* |

**Table 4.10 – Server\_Application:changePassword**

|  |  |
| --- | --- |
| **Prototype** | *Bool:Server\_Application:changePassword(user:User\_Info rmation ;string)* |
| **Purpose** | *This will allow the user to change the current password saved and used on the server* |
| **Receives** | *The server will receive the user's session information and a sting containing the new password* |
| **Returns** | *TRUE if the password was changed successfully FALSE if the user could not update the password sucessfully(length was not correct or did not pass password specifications)* |
| **Remarks** | *The operation might fail if the user log outs before the success or fail statement is sent back to the user.* |

**Table 4.11 – Server\_Application:requestUsersInformation**

|  |  |
| --- | --- |
| **Prototype** | *UserInformation:Server\_Application:requestUsersInform ation()* |
| **Purpose** | *This will allow the server to return all users information* |
| **Receives** | *All of the users that are currently on the server* |
| **Returns** | *TRUE if the information was passed correctly*  *FALSE if the user does not exist or the information was not passed correctly* |
| **Remarks** | *The operation might fail if there are no users on the server.* |

**Table 4.12 – Server\_Application:addWorkout**

|  |  |
| --- | --- |
| **Prototype** | *Bool:UserInformation:Server\_Application:addWorkout(us er:User\_Information)* |
| **Purpose** | *This allows the user to add a workout to their account* |
| **Receives** | *The user's information* |
| **Returns** | *TRUE if the workout was added*  *FALSE if the workout could not be added* |
| **Remarks** | *The operation might fail if the user tries to add a workout with an existing workout name* |

**Table 4.13 – Server\_Application:getUsersCanValidate**

|  |  |
| --- | --- |
| **Prototype** | *Bool:UserInformation:Server\_Application:getUsersCanV*  *alidateWorkout(user:User\_Information)* |
| **Purpose** | *Allows the user to set the users that can validate their workout.* |
| **Receives** | *The user's information* |
| **Returns** | *List of users that can validate workout* |
| **Remarks** | *This operation might fail if there are now users listed to validate a workout* |

**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**  **Accessors** | *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)*  *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\_informatio n,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.* |

**Table 7.1 - User\_Session**

|  |  |
| --- | --- |
| **Name** | *User\_Session* |
| **Base Class(es**) |  |
| **Purpose** | *Helps to receive the User Session* |
| **States** | *Empty or full.* |
| **Constructors** | *Default: empty session values*  *New: Contains the new session* |
| **Operators Mutators**  **Accessors** | user  getUser()  getSessionID() |
| **Fields** | *sessionID* |

**Table 8.1 – User\_Session:getSession**

|  |  |
| --- | --- |
| **Session ID** | *User\_Session::getSessionID()* |
| **Purpose** | *Gets the Session ID* |
| **Receives** | *void* |
| **Returns** | *Integer* |
| **Remarks** | *Gets the User Session ID* |

**Table 8.2 – User\_Session:getUser**

|  |  |
| --- | --- |
| **Use ID** | *User\_Session::getUser()* |
| **Purpose** | *Gets the User ID from Client Application* |
| **Receives** | *void* |
| **Returns** | *User\_Information* |
| **Remarks** | *Gets the User ID from Client Application* |

**Table 9.1 - Credentials**

|  |  |
| --- | --- |
| **Name** | ***Credentials*** |
| **Base Class(es**) |  |
| **Purpose** | *Helps client login to the server* |
| **States** | *Empty or full.* |
| **Constructors** | *Default: empty values*  *New: contains user information* |
| **Operators Mutators**  **Accessors** | *None*  getUserID()  getPassword() |
| **Fields** | userID  *password* |

**Table 10.1 – Credentials::getUser**

|  |  |
| --- | --- |
| **User ID** | Credentials::getUserID() |
| **Purpose** | *The Credentials gets the User ID from the*  *User.* |
| **Receives** | *Void* |
| **Returns** | *The Credentials returns the User Id as a*  *String.* |
| **Remarks** | *Gets the User ID from User* |

**Table 10.2 – Credentials::getUser**

|  |  |
| --- | --- |
| **Password** | Credentials::getPassword() |
| **Purpose** | *The Credentials gets the password from the*  *User.* |
| **Receives** | *Void* |
| **Returns** | *The Credentials returns the password as a*  *String* |
| **Remarks** | *Gets the User Password from the User.* |

**Table 11.1 - Workout**

|  |  |
| --- | --- |
| **Name** | ***Workout*** |
| **Base Class(es**) |  |
| **Purpose** | *Helps to receive workout information* |
| **States** | *Empty or full.* |
| **Constructors** | *Default: empty workout*  *New: contains workout information* |
| **Operators Mutators**  **Accessors** | None  getStartDate() getNumberOfWeeks getActivities() |
| **Fields** | *startDate*  *numberofWeeks activities* |

**Table 12.1 – Workout::getStartDate**

|  |  |
| --- | --- |
| **Start Date** | Workout::getStartDate() |
| **Purpose** | *Gets the Start date for the workout* |
| **Receives** | *Void* |
| **Returns** | *Integer* |
| **Remarks** | *Gives back the workout week* |

**Table 12.2 – Workout::getNumberOfWeeks**

|  |  |
| --- | --- |
| **Number of Weeks** | Workout::getNumberOfWeeks() |
| **Purpose** | *Gets the number of weeks for workout* |
| **Receives** | *None* |
| **Returns** | *Integer* |
| **Remarks** | *Gets the number of weeks the workout is going to be made.* |

**Table 12.3 – Workout::getActivities**

|  |  |
| --- | --- |
| **Activities** | Workout::getActivities() |
| **Purpose** | *Gets the type of Workout Activities* |
| **Receives** | *Void* |
| **Returns** | *Activities* |
| **Remarks** | *Gives the type of Activities the user wants to follow.* |

**Table 13.1 - <<Interface>> Activity**

|  |  |
| --- | --- |
| **Name** | ***Activity*** |
| **Base Class** |  |
| **Purpose** | *Helps to receive workout activities* |
| **States** | *Empty or full.* |
| **Constructors** | *Default: empty workout value.*  *New contains workout values.* |
| **Operators Mutators**  **Accessors** | getQuantity() getName() getDescription() getNumberPerWeek() |
| **Fields** | *void* |

**Table 14.1 – Activity::getQuantity**

|  |  |
| --- | --- |
| **Quantity** | Activity::getQuantity() |
| **Purpose** | *To get the User Workout quantities from the user* |
| **Receives** | *The User Session receives the workout quantities from the*  *Workout base class.* |
| **Returns** | *INT if there is a quantity*  *False if there is no quantity* |
| **Remarks** | *Gets the Workout quantity* |

**Table 14.2 – Activity::getName**

|  |  |
| --- | --- |
| **Workout Name** | Workout::getName() |
| **Purpose** | *To Get the Workout names from the Workout* |
| **Receives** | *The Workout name is received as a String from the workout base class* |
| **Returns** | *The Workout name as String* |
| **Remarks** | *Gets the workout Name* |

**Table 14.3 – Activity::getDescription**

|  |  |
| --- | --- |
| **Description** | Workout::getDescription() |
| **Purpose** | *Gets the Description of the Workout* |
| **Receives** | *Void* |
| **Returns** | *String* |
| **Remarks** | *Gets the Workout Description.* |

**Table 14.4 – Activity::getNumberPerWeek**

|  |  |
| --- | --- |
| **Number of Weeks** | Workout::getNumberPerWeek() |
| **Purpose** | *Gets the number of week* |
| **Receives** | *Void* |
| **Returns** | *Integer* |
| **Remarks** | *Gets the number of Weeks for workouts.* |