DreamTeam Project Architecture

Description of Components

WorkoutListModel

- This component is a model that stores each workout entered by a given user.
- Only the AddWorkoutController, EditWorkoutController, and GetWorkoutsController can communicate with WorkoutListModel.
    - AddWorkoutController can ask WorkoutListModel to store a new workout.
    - EditWorkoutController can ask WorkoutListModel to change information of a previous workout.
    - AddWorkoutController, GetWorkoutsController, and EditWorkoutController can ask WorkoutListModel for previous workouts.

AddWorkoutController

- This component is a controller for inputting new workout information.
- AddWorkoutController can only be accessed from NewWorkoutView and communicates with WorkoutListModel.
    - AddWorkoutController can receive data typed into the NewWorkoutView.
    - AddWorkoutController will then ask WorkoutListModel to store this data.

EditWorkoutController

- This component is a controller for changing data in a workout that was already added.
- EditWorkoutController can only be accessed from EditWorkoutView and communicates with WorkoutListModel.
    - EditWorkoutController can receive data typed into the EditWorkoutView.
    - EditWorkoutController will then ask WorkoutListModel to update with this data.

GetWorkoutsController

- This component is a controller for grabbing complete workout history.
- GetWorkoutsController can only be accessed from WorkoutHistoryView and communicates with WorkoutListModel.
    - GetWorkoutsController can ask WorkoutListModel for all previous workouts.
    - EditWorkoutController can send the workouts to WorkoutHistoryView.

NewWorkoutView

- This component is a view for inputting new workout information.
- NewWorkoutView communicates with HomeGUIView and AddWorkoutController.
    - NewWorkoutView can be accessed from a button on HomeGUIView.
    - AddWorkoutController can grab data typed into NewWorkoutView queries.

EditWorkoutView

- This component is a view for updating preexisting workout information.
- EditWorkoutView communicates with HomeGUIView and EditWorkoutController.
    - EditWorkoutView can be accessed from a button on HomeGUIView.
    - EditWorkoutController can grab data typed into EditWorkoutView queries.

WorkoutHistoryView

- This component is a view for observing past workout information.
- EditWorkoutView communicates with HomeGUIView and GetWorkoutsController.
    - EditWorkoutView can be accessed from a button on HomeGUIView.
    - EditWorkoutView receives and displays data from GetWorkoutsController.

UsersModel

- This component is a model that stores each user created account.
- CreateProfileController, EditProfileController, LoadProfileController, and ViewFriendsController can communicate with UsersModel.
    - CreateProfileController can ask UsersModel to store a new account.
    - EditProfileController can ask UsersModel to change information of an existing account.
    - LoadProfileController can ask UsersModel to check if a login attempt matches an account.
    - ViewFriendsController can ask UsersModel for accounts tagged as friends.

CreateProfileController

- This component is a controller for creating a new account.
- CreateProfileController can only be accessed from LoginView and communicates with UsersModel.
    - CreateProfileController can receive user input from LoginView.
    - CreateProfileController can ask UsersModel to store this input as a new account.

EditProfileController

- This component is a controller for editing an existing account.
- EditProfileController can only be accessed from ProfileView and communicates with UsersModel.
    - EditProfileController can receive user input from ProfileView.
    - CreateProfileController can ask UsersModel to update the current account with this input.

LoadProfileController

- This component is a controller for logging into an existing account.
- LoadProfileController can only be accessed from LoginView and communicates with UsersModel.
    - LoadProfileController can receive user input from LoginView.
    - LoadProfileController can ask UsersModel to check this input against all existing accounts.

ViewFriendsController

- This component is a controller for viewing a user's friends list.
- ViewFriendsController can only be accessed from FriendsView and communicates with UsersModel.
    - ViewFriendsController can ask UsersModel for information about all accounts marked as friends.
    - ViewFriendsController can update FriendsView

HomeGUIView

- This component is a view for navigating between feature
- HomeGUIView communicates with LoginView, ProfileView, FriendsView, NewWorkoutView, EditWorkoutView, and WorkoutHistoryView.
    - Each of the views listed above can be accessed by selecting an option on HomeGUIView.

LoginView

- This component is a view for creating accounts and logging in.
- LoginView communicates with HomeGUIView, CreateProfileController, and LoadProfileController.
    - LoginView can proceed to HomeGUIView
    - LoginView can send user input to CreateProfileController
    - LoginView can send user input to LoadProfileController

ProfileView

- This component is a view for observing and editing user account information.
- ProfileView communicates with HomeGUIView and EditProfileController.
    - ProfileView can be accessed from HomeGUIView.
    - ProfileView can receive data from and send updates to EditProfileController.

FriendsView

- This component is a view for observing account information from friends.
- ProfileView communicates with HomeGUIView and ViewFriendsController.

    - ProfileView can be accessed from HomeGUIView.
    - ProfileView can receive and display data from ViewFriendsController.

Workout

- This class creates a workout type.

WorkoutBuilder

- This class implements the builder design pattern and will build the workouts used in add workout edit workout and workout viewer.

UserInterface

- Holds general methods for types of users. For now we will just have a normal user but may implement a premium user with more methods.

UserNormal

- A standard user with the base UserInterface methods. Holds data from UserModel to create an instance of a User object.

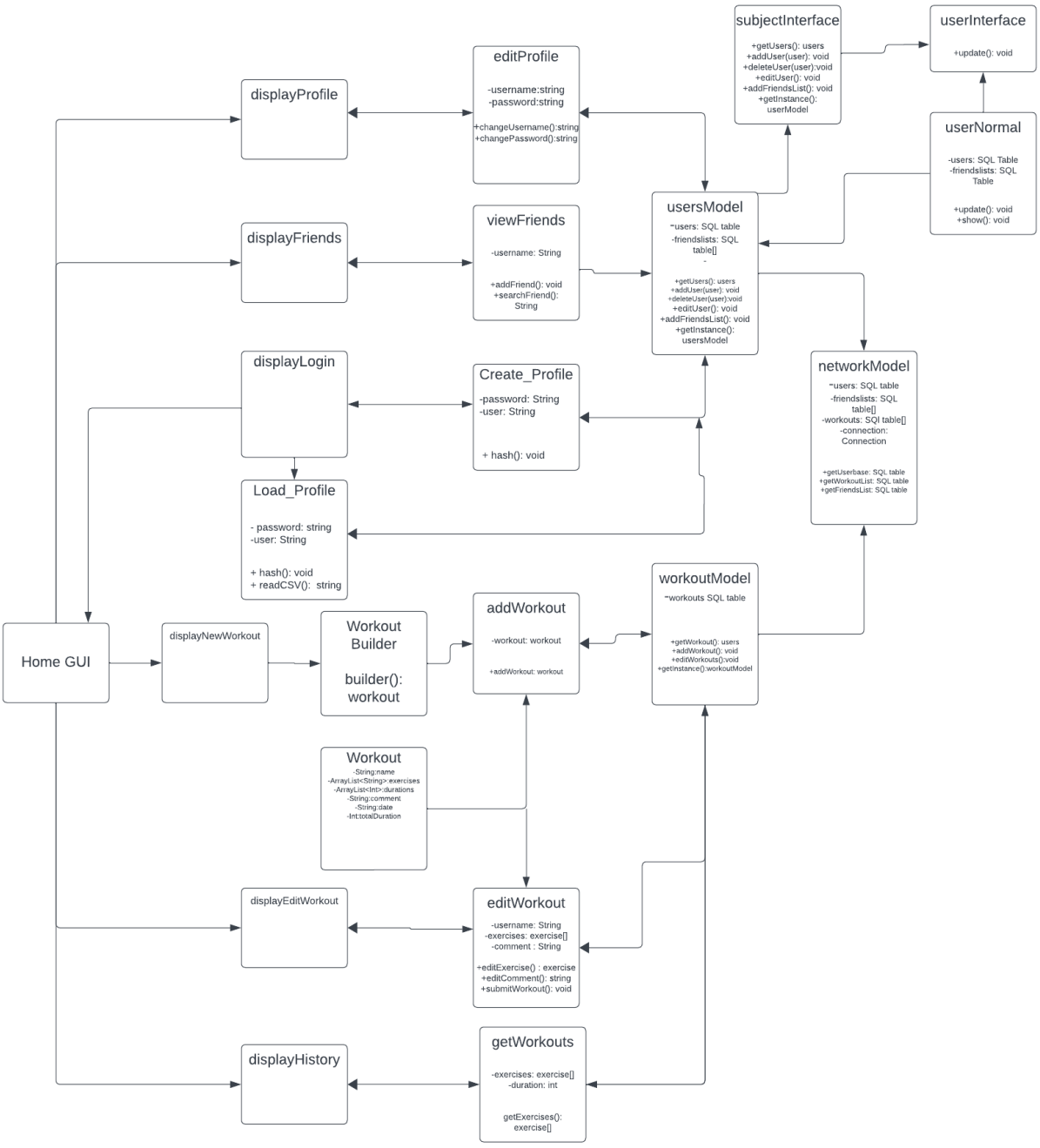
SubjectInterface

- An interface implemented by the UserModel. Holds base methods for adding and removing users.

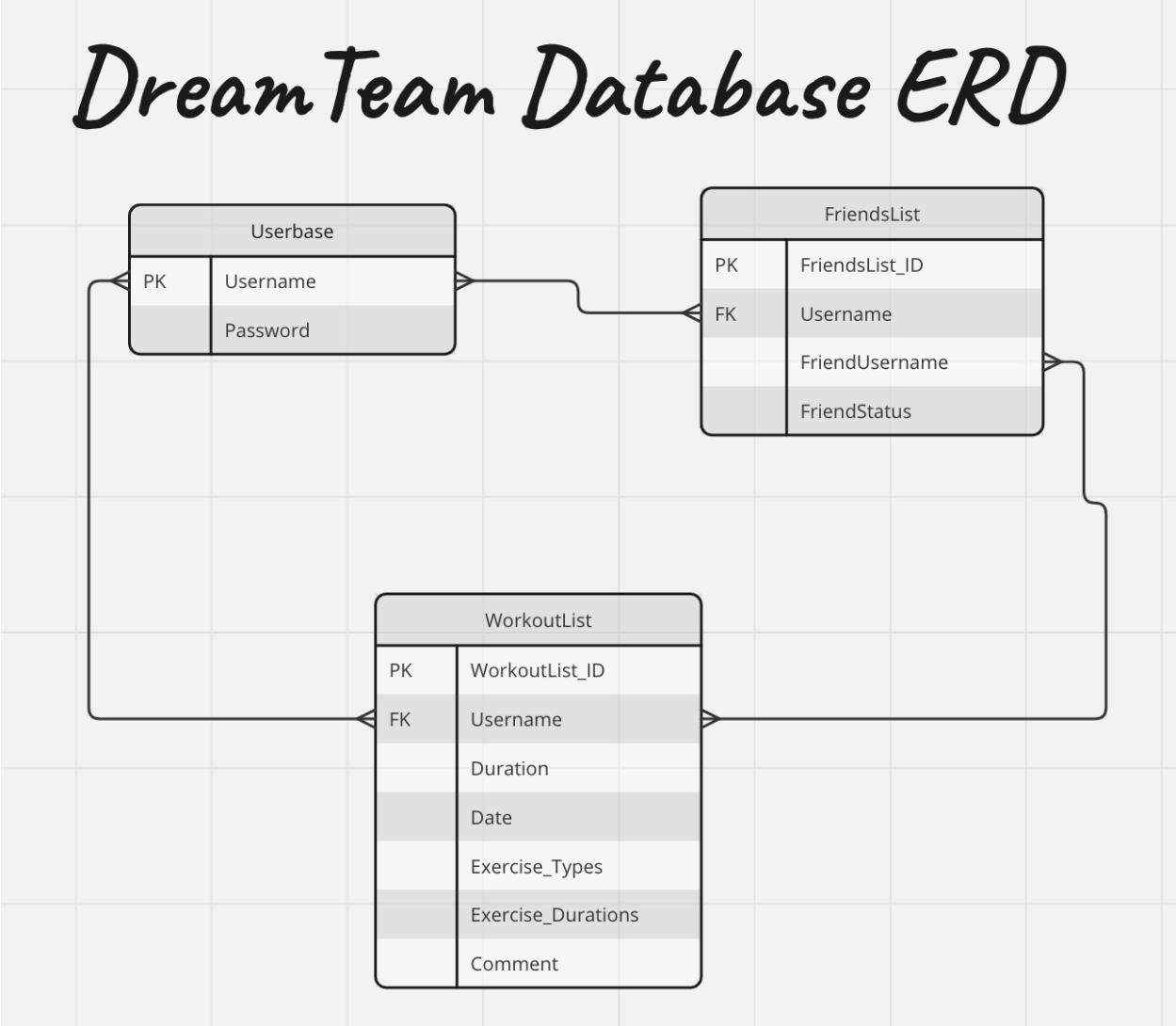
NetworkModel

- This component is a model that establishes and stores a SQL database connection
- Only the WorkoutModel and UserModel can communicate with NetworkModel.
    - UserModel can ask NetworkModel to get/set the userbase table
    - UserModel can ask NetworkModel to get/edit/create a friendslist table
    - WorkoutModel can ask NetworkModel to get/edit/create a workout table

UML Diagram


subjectInterface
+getUsers(): users
+addUser(user): void
+deleteUser(user):void
+editUser(): void
+addFriendsList(): void
+getInstance(): userModel
userInterface
+update(): void
editProfile
-username:string
-password:string
+changeUsername():string
+changePassword():string
displayProfile
userNormal
-users: SQL Table
-friendslists: SQL Table
+update(): void
+show(): void
usersModel
-users: SQL table
-friendslists: SQL table[]
-
+getUsers(): users
+addUser(user): void
+deleteUser(user):void
+editUser(): void
+addFriendsList(): void
+getInstance(): usersModel
viewFriends
-username: String
+addFriend(): void
+searchFriend(): String
displayFriends
displayLogin
Create_Profile
-password: String
-user: String
+ hash(): void
networkModel
-users: SQL table
-friendslists: SQL table[]
-workouts: SQl table[]
-connection: Connection
+getUserbase: SQL table
+getWorkoutList: SQL table
+getFriendsList: SQL table
Load_Profile
- password: string
-user: String
+ hash(): void
+ readCSV(): string
workoutModel
-workouts SQL table
+getWorkout(): users
+addWorkout(): void
+editWorkouts():void
+getInstance():workoutModel
addWorkout
-workout: workout
+addWorkout: workout
Home GUI
displayNewWorkout
Workout Builder
builder(): workout
Workout
-String:name
-ArrayList<String>:exercises
-ArrayList<Int>:durations
-String:comment
-String:date
-Int:totalDuration
displayEditWorkout
editWorkout
-username: String
-exercises: exercise[]
-comment : String
+editExercise() : exercise
+editComment(): string
+submitWorkout(): void
displayHistory
getWorkouts
-exercises: exercise[]
-duration: int
getExercises(): exercise[]

ERD Diagram



Demo Tables

(PK)

WorkoutList

WorkoutList_ID	Username	comment	Duration	Date	Exercise_type	Durations
668	Bob	I swam	30	2/14/23	Swim	90
782	BobBy	I Bike	60	4/19/23	Bike	60
721	Bob	I swam	60	12/15/23	Swim	60
779	BobBy	I Bike	90	12/13/23	Bike	90

Friends List

FriendsList_ID	Username	friendusername	Friend Status
2	Bob	Bobby	friends
4	Bob	Mom	sent
6	BobBy	Bob	friends
8	BobBy	Mom	sent

Component Stubs (each class in bold)

Create_Profile

```
Private user passwordHash(password,user) {

        //TODO Replace with actual algorithm
        return user;
}
```

Load_Profile

```
private String readfromProfileCSV(password,user) {

        //TODO Replace with actual algorithm
        return user_profile;

}
```

User_Model

```
private void writetoProfileCSV(password,user) {

        //TODO Replace with actual algorithm

}

private array getUsers() {

        //TODO Replace with actual algorithm
        return users_array;
}

private void addUser(password,user) {

        //TODO Replace with actual algorithm

}
```

```
private void editUser(password,user) {

        //TODO Replace with actual algorithm

}
```

Edit_Profile

```
public string changeUsername(password,user) {

        //TODO Replace with actual algorithm
        return new_username;

}

public string changePassword(password,user) {

        //TODO Replace with actual algorithm
        return new_password;

}
```

View_Friends

```
public void addFriend(user) {

        //TODO Replace with actual algorithm

}

public string searchFriend(user_input) {

        //TODO Replace with actual algorithm
        return user_matches;

}
```

Add_Workout

```
public exercise addExercise(type, duration) {

        //TODO Replace with actual algorithm
```

```
        return new_exercise;

}

public string addComment(input) {

        //TODO Replace with actual algorithm
        return comment;

}

public void submitWorkout(exercises[], string comment) {

        //TODO Replace with actual algorithm

}
```

Workouts_Model

```
public workouts[] getWorkouts() {

        //TODO Replace with actual algorithm
        return workouts_list;

}

public void addWorkouts() {

        //TODO Replace with actual algorithm

}
```

Edit_Workout

```
public exercise editExercise() {

        //TODO Replace with actual algorithm
        return updated;

}

public string editComment() {
```

```
        //TODO Replace with actual algorithm
        return updated;

}

public void submitWorkout() {

        //TODO Replace with actual algorithm

}
```

Get_Workouts

```
public exercise[] getExercises() {

        //TODO Replace with actual algorithm
        return exercise_list;

}
```

Workout

```
Public workout Workout(String: name, ArrayList<String>: exercises,ArrayList<Int>:
durations,String: comment,String: date,Int: totalDuration)[
        //TODO Replace with actual algorithm
        return workout;
}
```

WorkoutBuilder

```
Public workout workoutBuilder(){
        //TODO Replace with actual algorithm
        Return workout;
}
```

SubjectInterface

```
users getUsers();
void addUser(user);
```

```
void deleteUser(user);
void editUser();
void addFriendsList();
userModel getInstance();
```

UserInterface

```
void update();
```

UserNormal

```
void update(){
        //TODO Replace with actual algorithm
}
```

NetworkModel

```
Connection connect(){
        //TODO Replace with actual algorithm
}

void createUser(){
        //TODO Replace with actual algorithm
}

user getUser(User user){
        //TODO Replace with actual algorithm
}

void deleteUser(User user){
        //TODO Replace with actual algorithm
}

workout[] getWorkoutList(User user){
        //TODO Replace with actual algorithm
}
void updateWorkoutList(User user){
        //TODO Replace with actual algorithm
}

User[] getFriendsList(){
        //TODO Replace with actual algorithm
}
```

```
void updateFriendsList(){
	//TODO Replace with actual algorithm
}
```