Name: Ethan Fields Student #: 101245491

Project V2 - Project Report

2.1 *(Conceptual Design)*

An image of the ER-Diagram can be found in the Documentation folder in the GitHub Repository.

Assumptions regarding cardinality and participation types include:

- **Member-HealthMetrics:**
  - Each member has a unique healthMetrics record (mandatory participation)
  - Each healthMetrics record has a unique member assigned to it (mandatory participation)
  - Cardinality: One-to-One (1:1)
- **Member-Dashboard:**
  - Each member has a unique dashboard record (mandatory participation)
  - Each dashboard record has a unique member assigned to it (mandatory participation)
  - Cardinality: One-to-One (1:1)
- **Exercises-Exercise_Routines:**
  - Each exercise can be associated with many exercise routines (mandatory participation).
  - Each exercise routine can be associated with many exercises (mandatory participation).
  - Cardinality: Many-to-Many (M:N)
- **Routines-Exercise_Routines:**
  - A routine can be associated with many exercise routines (mandatory participation).
  - Each exercise routine must have exactly one routine (mandatory participation).
  - Cardinality: One-to-Many (1:N)
- **Dashboard-Routines:**
  - A dashboard may or may not have exactly one routine (optional participation).
  - Each routine must have exactly one dashboard entry (mandatory participation)
  - Cardinality: One-to-Many (1:1)

- **Members-BillingInfo:**
  - Each member has a unique billing information record (mandatory participation)
  - Each billing information record has a unique member assigned to it (mandatory participation)
  - Cardinality: One-to-One (1:1)
- **Trainers-Trainers_Availability:**
  - Each trainer may or may not have at least one availability record (optional participation)
  - Each availability record must belong to exactly one trainer (mandatory participation)
  - Cardinality: One-to-Many (1:N)
- **Trainers-Personal_Training_Sessions:**
  - Each trainer may or may not be assigned to at least one personal training session (optional participation)
  - Each personal training session must be conducted by exactly one trainer (mandatory participation)
  - Cardinality: One-to-Many (1:N)
- **Members-Personal_Training_Sessions:**
  - Each member may or may not be signed up for at least one personal training session (optional participation)
  - Each personal training session must belong to exactly one member (mandatory participation)
  - Cardinality: One-to-Many (1:N)
- **Room_Bookings-Personal_Training_Sessions:**
  - A room booking may or may not be associated with one personal training session (optional participation)
  - Each personal training session must be assigned to exactly one room booking (mandatory participation)
  - Cardinality: One-to-One (1:1)
- **Room_Bookings-Admin:**
  - Each room booking can be interacted with by at least one admin (mandatory participation)
  - Each admin can interact with at least one room booking (mandatory participation)
  - Cardinality: Many-to-Many (M:N)

Name: Ethan Fields Student #: 101245491

- **GroupClasses-Admin:**
  - Each group class can be interacted with by at least one admin (mandatory participation)
  - Each admin can interact with at least one group class (mandatory participation)
  - Cardinality: Many-to-Many (M:N)
- **GroupClasses-Room_Bookings:**
  - A room booking may or may not be associated with one group class (optional participation)
  - Each group class must be assigned to exactly one room booking (mandatory participation)
  - Cardinality: One-to-One (1:1)
- **Member_Group_Schedules-Members:**
  - A member group schedule must have exactly one member assigned to it (mandatory participation)
  - Each member may or may not be associated with at least one member group schedule (optional participation)
  - Cardinality: One-to-Many (1:N)
- **Member_Group_Schedules-GroupClasses:**
  - A member group schedule must have exactly one group class assigned to it (mandatory participation)
  - Each group class may or may not be associated with at least one member group schedule (optional participation)
  - Cardinality: One-to-Many (1:N)
- **ClubEquipment-Admin:**
  - Each club equipment record can be interacted with by at least one admin (mandatory participation)
  - Each admin can interact with at least one club equipment record (mandatory participation)
  - Cardinality: Many-to-Many (M:N)
- **ClubPayments-BillingInfo**
  - Each club payment record must be associated with exactly one billing information record (mandatory participation)
  - Each billing info may or may not be associated with a club payment record (optional participation)
  - Cardinality: One-to-Many (1:N)

- **ClubPayments -Admin:**
  - Each club payment record can be interacted with by at least one admin (mandatory participation)
  - Each admin can interact with at least one club payment record (mandatory participation)
  - Cardinality: Many-to-Many (M:N)

Here are additional assumptions regarding the implementation of the required functions made for this project:

- *User Registration* – The user only needs to provide a unique username, an unique email address, and must verify their passcode using "confirm Password". Once the user is registered, a prompt will appear and redirect the user to the login page. This registration only works for member users.
- *Profile Management* – The user can view and change their information in regards to general information, health stats, fitness goals, and payment information. The option to change passwords is not included.
- *Dashboard Display* – This dashboard displays the current weight, while showing how close the user is to their weight goal. It also includes a tracker for number of days exercised in the week, a trophy count representing the number of weekly goals consistently met, and a streak counter to count the number of days exercised on in a row. This number of days exercised is tracked by pressing a log button in the exercise section for the user's current exercise routine selected. There is also an option to change exercise routines at the bottom of the page.
- *Schedule Management* (for members) – The member can register for group classes, book with trainers, and view their personal schedules which shows two tables for personal training sessions and group classes separately. The member will have to pay a fee for registering to the group class, and if they deregister from it in their personal schedule, they will not be repaid. As for booking with a personal trainer, the member will view another form which will ask the user to select an available time with the trainer. If they select a time outside of the trainer's availability, a warning will be sent to the user. Otherwise, once they pick a valid time, they will pay 50$ to book the session. The member can then reschedule or cancel their training sessions from the personal schedule section.

- *Schedule Management* (for trainers)– The trainers can view the training sessions they have booked. They can choose to cancel the sessions if they wish. The trainer can also change their availability with a start time and end time on each day of the week.
- *Member Profile Viewing* – The trainer can search for the member by entering in a search term, and the hub will return results which are close to the search term provided. The search bar also can accept any letters regardless of capitalization. Once the results are returned, a number of results returned will be displayed, along with a brief overview of each member in the results. Each member will also have a "View Profile" button which show a div with information that could be found in the member profile page. Of course, their payment information with their credit card information and such is not shown.
- *Room Booking Management* – The admin user can manipulate the room number, the time start, and booking day.
- *Equipment Maintenance Monitoring* – There is equipment shown in a table which the user can either add, fix, or delete the existing equipment, along with being able to add new equipment entirely.
- *Class Schedule Updating* – The admin user can update either the instructor, the time start, the duration, and the class name of existing classes. They can also insert new classes, or delete other classes.
- *Billing and Payment Processing* – The admin user will see a table of payments that are made by the member users. The admin will either approve or disapprove the payment. Once a decision is made, it will affect the item being paid for.

- Additionally, trainers are considered as separate from instructors in the group classes. So, trainers' schedules will only look at personal training sessions.

2.2 *(Reduction to Relation Schemas)*

An image of the Relational Database Schema can be found in the Documentation folder in the GitHub Repository.

2.3 *(DDL File)*

The DDL File can be found in the SQL folder in the GitHub Repository.

2.4 *(DML File)*

The DML File can be found in the SQL folder in the GitHub Repository.

2.5 *(Implementation)*

Application Architecture

The application is designed to provide functionality of a fitness club web application. The architecture of the application follows a client-server model with a relational database management system (RDBMS) serving as the backend storage. The main idea is for there to be three separate sections of the website, which will support actions for members, trainers, and admin staff respectively. The main source code is stored inside of the '/code' folder.

Components in /code:

1. **/public**: This folder holds the bulk of the code for the client-side. This includes the scripts, CSS styling, and even some images for the design of the website:
   a) **/images**: These images are used for adding style to some of the webpages. It also includes an image used for the shortcut icon found on the browser tab.
   b) **/js**: The folder holds the scripts which are used for the majority of the backend for this project, and it is split into three separate folders (members, trainers, admin) for ease for transversality. It also includes logregscripts.js which handles the scripting for the login/register page, and the navscripts.js file which is used for traversing between webpages (HTML files) while the program is running.
   c) **/styles**: This folder holds the CSS files responsible for styling the website as needed. It includes separate files for member web pages, trainer web pages, admin web pages, and the login/register web page.
2. **/views**: This folder holds the .html files which will be used for loading up the webpages that will be needed for the remainder of this project. They are

separated into three separate folders (members, trainers, admin) to distinguish the web pages for each website.

3. **server.js**: The server is responsible for establishing a connection to the PostgreSQL database, and handling the middleware. Once a connection is formed, the server will be started on PORT 3000, and will provide links in the terminal for testing purposes.
4. **routes.js**: All of the routes for sending queries to the PostgreSQL database are stored in this file. This file is also responsible for allowing navigation between webpages.
5. **Relational Database**: This relational database serves as the persistent storage for the application's data. The chosen system for testing purposes is pg Admin 4, and it utilises PostgreSQL.

Technologies Used:

- **Programming Languages**: JavaScript, HTML, CSS
- **Frameworks/Libraries/Environments**: Express JS, Node.js, node-postgres(pg)
- **Relational Database Management System (RDBMS)**: PostgreSQL

2.6 *(Bonus Features)*

Bonus Features:

- Included a login page
- Included a password system
- Included a pop-up side bar navigation bar for members
- Implemented the profile web page in a way where there are different sections for the profile
- Application was implemented as a web Application

2.7 *(GitHub Repository)*

Link to repository: GitHub Project Repository

Note: The README and video link is included in the repository as well.