COMP3005 – Final Project: Requirements & Assumptions

Authors: Thomas Wood, Kaif Ali

Application Assumptions:

- Problem Statement says: “This system will serve as a comprehensive platform catering to the diverse needs of club members …and administrative staff.” Therefore, this application will need to implement a user system that has standard (fitness) users and administrative users.

Fitness User Account:

- Problem Statement says: "Members should be able... [to input] ...health metrics." Therefore, we need to store same basic health details about the user.
    - We’ll assume health metrics means the users height, weight, and date of birth.
- Problem Statement says: “Once registered, they will gain access to a personalized dashboard that tracks their exercise routines… [and] …fitness achievements.” Therefore, we should implement a user dashboard for fitness users that lets them view update their exercise routines and fitness achievements (goals).
    - We will assume that fitness users should be able to add exercises that they wish to do.
    - We will also implement a way for users to update when they have last completed an exercise, they have added that way it will be easier for them to track their routines.
    - We should assume that a user has no limit to how many goals they can create.
- Problem Statement says: “Furthermore, members can register for group fitness classes, workshops, and other events….” Therefore, fitness users will need to be able to RSVP/Sign Up for events created by administrative staff.
    - We will assume fitness users can RSVP/Sign Up for multiple events.
- Problem Statement says: “The club’s unique selling point is its loyalty program; every transaction earns members loyalty points, which can be redeemed for future services.” Therefore, the fitness user should have a points value associated with their account and is updated with every transaction they pay. We will assume that “redeemed for future services” means fitness users can pay future transactions with points so long they have enough points to cover the cost of the transaction.
    - Additionally, we’ll assume that the points a user gets back from a transaction is decided by the administrative staff when they create the transaction.

- The Problem Statement mentions admins are able to monitor maintenance on the fitness equipment which we will assume means a ticketing system that allows fitness users to create and send tickets for administrative staff to view and complete.
    - We assume each ticket is created uniquely and cannot come from multiple fitness users.
    - We’ll assume a fitness user can create one or many tickets.
    - For simplicity tickets will not be linked back to the user who created the ticket in any way. Therefore, we will not need a bridge table for tickets made by each user.

- The Problem Statement mentions that administrative staff will be able to create transactions to charge fitness users for things such as members ships fees, we must assume the fitness user will need to have some sort of payment method linked to their account to pay for any transactions linked to their account.
    - To keep it simple we will assume each fitness user can only have one payment method.
    - We will assume there is no cost to create a fitness account. Therefore, fitness users will not need to add a payment method upon signing up.

<u>Admins:</u>

- The Problem Statement mentions that administrative staff need to be able to manage fitness equipment maintenance, billing, process payments for membership fees, and monitor club activities. Therefore, we should create a custom dashboard for admins that allows them to manage users, create and manage transactions/payments/billing, create events, and the ability to view and complete fitness equipment maintenance tickets.

- We will assume that as an administrative user you will not have access to the standard fitness user dashboard. Therefore, as an admin you will not be able to do any of the things a fitness user can do.

<u>Other:</u>

- We will assume the application should be web based that way users can access it on more than one device and aren't limited to just being able to use it on desktop computer.

Basic Requirements:

- Application will be web-based
    - Designed as a Three-Tier Architecture Application
        - Server side/Logic layer that will handle client GET, POST, PUT, and DELETE requests.
        - Client side/Presentation layer that will present the users data to their web browser and allow them to modify said data.
        - Data Layer using PostgreSQL
            - Responsible for storing all application data.
    - Viewable Pages
        - Global pages
            - / - Welcome page
            - /login
            - /logout
            - /event/:id – view details about an event
            - /exercise/:id – view details about an exercise
            - /register – create a new account

        - Fitness User Pages
            - / - Fitness user dashboard
            - /profile – View account info
            - /pay/:id – Pay a specific transaction
        - Admin
            - / - Admin dashboard
            - /admin-password-reset/:id – Reset a user’s password
- User Functionality
    - Admin
        - Ability to view all users.
            - Ability to reset a specified user’s password.
            - Toggle a specified user from being able to log in.
        - Ability to create a new fitness user/admin.
        - Ability to view and complete maintenance tickets.
        - Ability to create a new transaction for a specified user.
        - Ability to create a new event.
        - Ability to view upcoming and previous events.
            - View the users who have RSVP’d to a specific event.
    - Fitness User
        - Ability to add their basic health details to their account upon sign up.
        - Ability to track their fitness goals.
            - Ability to create new goals.
            - Ability to mark a goal as complete.
        - Ability to track exercises.
            - Ability to add a exercise they wish to do from a specified list of already created exercises.
            - Ability to complete exercise which will update the date they last did the exercise to today.
        - Ability to view upcoming events.
            - Ability to RSVP to each upcoming event.
        - Ability to view past events.
        - Ability to view their points balance.
        - Ability to view all of their transactions.
            - Ability to pay any unpaid transactions.
                - Ability to pay with points or linked payment method.
        - Ability to create a ticket.
        - Ability to view their profile.
            - Ability to update their basic health details.
            - Ability to view their payment method.
                - Ability to update their payment method if current method is expired.
            - Ability to add a new payment method if none exists.

- Required Tables:
  - **exerciseAdded** – bridge table to track what exercises each user has added to their fitness routine.
    - eid – exercise id
    - uid – user id
    - lastDone – when the exercise was last done by the user
  - **exercise** – contains exercises that users can view and add to their fitness event.
    - id – exercise id
    - name – name of exercise
    - info – contains the steps and targets muscles of the exercise
    - link – visual of the exercise
  - **ticket** – contains all active tickets submitted by fitness users.
    - id – ticket id
    - subject – ticket subject
    - description – ticket description
  - **rsvpActivities** – bridge table to keep track of what events a user has RSVP'd to.
    - aid – event id
    - uid – user id
  - **fitness_event** – contains all events and their details.
    - id – event id
    - name – event name
    - info – info about the event
    - when – when the event occurs
  - **payment** – contains all fitness user payment methods.
    - uid – user id
    - type – type of payment method
    - cardNum – card number
    - expiryYear – expiry year on card
    - expiryMonth – expiry month on card
    - cvc – card verification code
    - name – name on card
  - **user_transaction** – contains all transactions.
    - id – transaction id
    - uid – user id
    - date – transaction due date
    - type – type of transaction
    - amount – transaction amount
    - points – points given to the users account upon payment
    - paidByPoints – was the transaction paid in points?
    - paid – is the transaction paid?
  - **user_account_info** – contains each users basic health details.
    - uid – user id
    - height – height of user
    - weight – user weight

        - dateOfBirth – birthday of user
- **user_goals** – contains each users goals.
    - <u>uid</u> – user id
    - goals – all user goals in JSON
- **fitness_user** – contains all users.
    - id – user id
    - email – user email
    - password – user password
    - fName – users first name
    - lName – users last name
    - admin – is user an administrator?
    - joinDate – when account was created
    - points – user points balance
    - disabled – is user not allowed to login?